



Ricardo Cabello
aka mrdoob

three.js

справочное руководство

three.js - это небольшая по размеру кросбраузерная JavaScript библиотека/API используемая для создания и отображения анимированной трехмерной компьютерной графики в веб-браузере. Скрипты **three.js** можно использовать в связке с элементом canvas HTML5, SVG или WebGL.

В **three.js** возможно создавать трехмерные анимации, ускоряемые графическим процессором, используя язык JavaScript как часть вебсайта, не полагаясь на собственные плагины браузеров. Это стало возможным благодаря появлению WebGL.

Библиотека предоставляет рендереры <canvas>, <svg>, CSS3D и WebGL.

В three.js включены следующие функции:

Визуализаторы (рендереры): Canvas, SVG и WebGL.

Эффекты: [анаглиф](#),

собственных глаз менять фокусное расстояние.

Если есть желание попробовать,

выполните следующую последовательность действий:' +'

Смотрите на изображение.

Не отрывая взгляда от изображения, начинайте сдвигать фокусное расстояние ближе к носу (скосите глаза).

' +'

Две части изображения начнут накладываться одна на другую, образуя третью, которая появится в центре.

' +'

Сосредоточьтесь на появившейся картинке и сфокусируйте на ней взгляд.

Если у вас все получилось, то изображение в центре вы увидите трехмерным.')"> [перекрестный взгляд \(cross-eyed\)](#) и [параллаксный барьер \(parallax barrier\)](#).

Сцены: добавление и удаление объектов во время выполнения; туман

Камеры: перспективная и ортографическая

Контроллеры: [трекбол](#), FPS, path и другие Controllers: trackball, FPS, path and more

Анимация: armatures, forward kinematics, inverse kinematics, morph and keyframe

Освещение: фоновое, направленное, точечное и прожекторное ambient, direction, point and spot lights

Тени: отбрасывание и получение cast and receive

Материалы: PBR, Phong, Lambert, smooth shading, textures and more

Формы: доступ к полноценному языку программирования шейдеров ([OpenGL Shading Language - GLSL](#))

Capabilities: блики от линз, передача глубины и обширная библиотека пост-обработки lens flare, depth pass and extensive post-processing library

Объекты: сетки (meshes), частицы (particles), спрайты (sprites), линии, ленты (ribbons), кости (bones) и многое другое - все с all with Level of detail

Геометрия: плоскость, куб, сфера, тор, 3D текст и другие

Modifiers: lathe, extrude and tube

Загрузчики данных: бинарный, графический, JSON и сценарный Data loaders: binary, image, JSON and scene

Утилиты: полный набор временных и трехмерных математических функций, включая усеченный конус (усеченная параллельными плоскостями часть фигуры), матрицу, Quaternionian, UV и другие Utilities: full set of time and 3D math functions including frustum, matrix, Quaternionian, UVs and more

Экспорт и импорт: утилиты для создания JSON-файлов, совместимых с [three.js](#) из: Blender, openCTM, FBX, Max, и OBJ

Сопровождение: разрабатывается документация по API, общественный форум и вики в полном объеме documentation is under construction, public forum and wiki in full operation

Примеры: Свыше 150 файлов с кодами примеров плюс шрифты, модели, текстуры, звуки и другие поддерживаемые файлы

Отладка: [Stats.js](#), [WebGL Inspector](#), [Three.js Inspector](#)

three.js работает во всех браузерах, поддерживающих WebGL.

Выпускается ***three.js*** под [лицензией MIT](#).

ВВЕДЕНИЕ

Создание сцены

Цель этого раздела - дать краткое вступление в ***three.js***. Начнем его созданием сюжета с вращающимся кубом. На случай если вы запутаетесь и понадобится помочь, рабочий пример кода [приводится ниже](#).

Прежде чем начать

Перед использованием ***three.js***, нужно определиться, где мы будем его показывать. Сохраните следующий HTML-код как файл на своем компьютере вместе с копией файла [`three.js`](#) в папке `js/` и откройте его в своем браузере.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset=utf-8>
    <title>My first Three.js app</title>
    <style>
      body { margin: 0; }
      canvas { width: 100%; height: 100% }
    </style>
  </head>
  <body>
    <script src="js/three.js"></script>
    <script>
      // Здесь будет ваш Javascript код.
    </script>
  </body>
</html>
```

Это все. Весь код ниже идет в пустой тег `<script>`.

Создание сцены

Для того, чтобы была возможность что-либо отображать с ***three.js***,

нужны три вещи: сцена (scene), камера (camera) и визуализатор (renderer) - также называемый рендерер (по звучанию английского слова), чтобы была возможность показывать сцену, снятую камерой.

```
var scene = new THREE.Scene();
var camera = new THREE.PerspectiveCamera( 75, window.innerWidth / w
var renderer = new THREE.WebGLRenderer();
renderer.setSize( window.innerWidth, window.innerHeight );
document.body.appendChild( renderer.domElement );
```

Давайте воспользуемся моментом и поясним, что здесь происходит. На данном этапе у нас имеется набор из сцены (переменная scene), камеры (переменная camera) и визуализатора (переменная renderer). В **three.js** имеется несколько разных камер. Сейчас мы будем использовать PerspectiveCamera (т.е. камеру с отображением перспективы). Первым параметром у нее является которым ограничена зона видимости камеры, объекты не попадающие

'+' в нее не будут видны.

Различают поле зрения по горизонтали и по вертикали.

В three.js угол fov - это поле зрения по вертикали и измеряется в градусах.');" onmouseout="hide()">fov (поле зрения или просмотра).

Второй параметр - это

aspect ratio - отношение сторон');" onmouseout="hide()">aspect (соотношение сторон, пропорции или формат кадра). Почти всегда нужно использовать отношение ширины элемента к его высоте или получите такой же результат как при просмотре старых фильмов на широкоэкранных телевизорах - изображение будет выглядеть сплющенным.

Следующими двумя параметрами являются near (ближняя) и far (дальняя) плоскости отсечения. Это значит, что объекты дальше, чем значение far или ближе, чем near не будут показаны. Пока о них можно не беспокоиться, но для увеличения быстродействия в своем приложении можно использовать другие значения.

Примечание переводчика: как говорится "вместо тысячи слов", вот на рисунке показаны все параметры камеры.

[... читать далее](#)

Далее идет визуализатор (рендерер). Вот тут то и происходит волшебство. В дополнение к используемому здесь WebGLRenderer, в Three.js имеется несколько других, зачастую используемых в качестве резервного варианта для пользователей со старыми браузерами или тех, у кого нет поддержки WebGL по другим причинам.

Помимо создания экземпляра визуализатора (рендерера), также нужно установить размеры в которых будет отображаться наше приложение. Хорошей идеей будет использование ширины и высоты области, которую нам хочется занять нашим приложением - в данном случае, это ширина и высота элемента <canvas>, т.е. полный размер (помните, у <canvas> и ширина (width) и высота (height) равны 100%) окна браузера. При выполнении ресурсоёмких приложений также можно задать параметру **setSize** меньшие значения, вроде **window.innerWidth/2** и **window.innerHeight/2**, которые сделают визуализацию приложения в половинном размере.

Если нужно сохранить размеры приложения, но отобразить его с более низким разрешением, можно сделать это, вызвав параметр **setSize** со значением **false** в качестве параметра **updateStyle**.

Например, код

```
setSize(window.innerWidth/2, window.innerHeight/2, false)
```

сделает визуализацию приложения в половинном разрешении, с учетом того, что наш <canvas> имеет 100% ширину и высоту от размеров окна браузера.

И наконец, что не менее важно, к нашему HTML-документу добавляем элемент **renderer'a**. Это элемент <canvas>, на котором рендерер и отображает сцену.

Так то все хорошо, но где этот обещанный куб?

Сейчас мы его добавим.

```
var geometry = new THREE.BoxGeometry( 1, 1, 1 );
var material = new THREE.MeshBasicMaterial( { color: 0x00ff00 } );
var cube = new THREE.Mesh( geometry, material );
scene.add( cube );
```

```
camera.position.z = 5;
```

Для создания куба нам потребуется **BoxGeometry**, т.е. геометрия куба. Это объект, содержащий все точки (вершины - **vertices** ед. число vertex) и стороны (границы - **faces**) куба. В дальнейшем мы рассмотрим это более подробно.

Примечание переводчика: Здесь и далее слово «геометрия» означает совокупность всех геометрических свойств объекта, т.е. все элементы, которые составляют каркас объекта - вершины, грани, а также их цвет и расположение.

[... читать далее](#)

В дополнение к геометрии куба понадобится и материал для его окрашивания. В Three.js имеется несколько материалов, но в данном случае будем использовать **MeshBasicMaterial**. Все материалы принимают свойства объекта, что были к нему применены. Чтобы сохранить простоту вещей, предоставим только атрибут цвета **0x00ff00**, т.е. зеленый цвет. Это работает точно также как в CSS или Photoshop (шестнадцатиричные значения цвета).

Третья вещь, что нам потребуется, это **Mesh**, т.е. сетка. Mesh представляет собой объект, который принимает геометрию и применяет к ней материал, который потом можно вставить в нашу сцену и свободно перемещаться вокруг.

По умолчанию, при вызове `scene.add()`, добавляемая нами вещь будет располагаться в координатах **(0,0,0)**. Чтобы этого избежать, мы в последней строке просто немножко смещаем камеру.

Визуализация (рендеринг) сцены

Если скопировать код, приведенный выше, и вставить в заранее созданный HTML-файл, то вы ничего не увидите. Потому что на самом деле пока еще нечего отображать. Для того, чтобы что-нибудь появилось нужно то, что называется **render loop** - циклом визуализации.

```
function render() {  
    requestAnimationFrame( render );  
    renderer.render( scene, camera );  
}  
render();
```

Этот код создаст цикл, который будет 60 раз в секунду вызывать визуализатор для прорисовки сцены. Если вы новичок в написании браузерных игр, то можете сказать: "Почему бы нам просто не вызвать JavaScript-функцию

т.е. времени между вызовами функции перерисовки сцены');"
onmouseout="hide()">setInterval?" Конечно же, можно это сделать, но у функции
- позволяет синхронизировать все анимации со
встроенными механизмами обновления страницы.

То есть, сгруппированы будут не только JavaScript-анимации,
но и CSS-анимации и другие браузерные перерисовки. При этом
графический ускоритель используется максимально эффективно
и исключается повторная обработка одних и тех же участков
страницы.

А значит – меньше будет загрузка CPU, да и сама анимация
станет более плавной.');"

onmouseout="hide()">requestAnimationFrame имеется несколько преимуществ. Пожалуй, самым важным из них является то, что анимация приостанавливается при переходе пользователя на другую вкладку браузера, а следовательно не тратится драгоценная вычислительная мощность и заряд аккумулятора.

Примечание переводчика: Вот ещё одна картинка для лучшего понимания процесса визуализации в three.js.

[... читать далее](#)

Анимация куба

Если весь код, приведенный выше, вставлен в заранее созданный файл, вы должны увидеть зеленый куб. Давайте сделаем его чуть поинтереснее - покрутим его.

Добавьте следующий код сразу перед вызовом `renderer.render` функции `render`:

```
cube.rotation.x += 0.1;  
cube.rotation.y += 0.1;
```

Этот код будет выполняться в каждом кадре (60 раз в секунду) и

задаст кубу хорошую анимацию вращения. В принципе, все что нужно перемещать или изменять во время работы приложения, должно проходить через цикл рендеринга. Конечно, там можно вызвать и другие функции, но постарайтесь не делать функцию `render` в несколько сотен строк.

Итог

Поздравляем! Вы только что получили свое первое `three.js` приложение. Как видите, просто нужно было где-то начать.

Полный код показан ниже. Чтобы получить более полное представление о его работе, поиграйтесь с ним (т.е. попробуйте поменять те или иные параметры и понаблюдайте за получающимися изменениями).

Код: [... показать](#)

Импорт с помощью модулей

Несмотря на то, что импорт *three.js* с помощью тега `<script>` отличный способ быстро получить и начать работать с библиотекой, у него имеется несколько недостатков для продолжительных проектов, например:

- Нужно вручную выбирать и подключать копию библиотеки как часть исходного кода проекта.
- Обновление версии библиотеки также "ручной" процесс.
- При проверке новой версии данной библиотеки управление различия в управлении версиями завалены множеством строк сборочного файла.

Использование диспетчера зависимостей типа `npm` обходит эти условия позволяя просто скачивать и импортировать желаемую версию библиотеки на вашу машину.

Установка при помощи `npm`

Примечание переводчика: `npm` - сокращ. от Node.js Package Manager — менеджер пакетов, входящий в состав [Node.js](#)

three.js выпускается как модуль [`npm`](#). Это значит, что для включения *three.js* в проект, нужно просто запустить "`npm install three`".

Импорт модуля

Предположим, что вы объединяете файлы с помощью такого инструмента как [Webpack](#) или [Browserify](#), которые позволяют *Assuming that you're bundling your files with a tool such as Webpack or Browserify, which allow you to "require('modules') in the browser by bundling up all of your dependencies."* Теперь можно внести модуль в свои исходные файлы и продолжить использовать его в обычном режиме.

```
var THREE = require('three');

var scene = new THREE.Scene();
...
```

Также есть возможность использования импорта синтаксиса ES6:

```
import * as THREE from 'three';

const scene = new THREE.Scene();
...
```

или, если хотите импортировать только отдельные части библиотеки ***three.js***, например Scene:

```
import { Scene } from 'three';

const scene = new Scene();
...
```

Предостережения

В настоящее время таким способом невозможно импортировать файлы в каталог "examples/js". Это происходит из-за того, что некоторые из файлов зависимы от загрязнения глобального пространства имен THREE. Более подробные сведениясмотрите на [Transform `examples/js` to support modules #9562](#).

Поддержка браузерами

Обзор

Three.js может использовать WebGL для визуализации (рендеринга) сцен во всех современных браузерах. Для старых браузеров, особенно Internet Explorer 10 и ниже, может потребоваться откат к одному из оставшихся [визуализаторов \(рендереров\)](#) (CSS2DRenderer, CSS3DRenderer, SVGRenderer, CanvasRenderer). Кроме того, может потребоваться включение некоторых полифиллов, особенно при использовании файлов из каталога [/examples](#).

«Полифилл» (англ. «polyfill» - poly - много, fill - наполнять, заполнять) ... [читать далее](#)

Примечание: если поддержка устаревших браузеров не требуется, то не рекомендуется использовать другие визуализаторы-рендереры, отличные от WebGLRenderer, так как они работают медленнее и поддерживают меньшее количество функций.

Браузеры, поддерживающие WebGL

WebGL поддерживается следующими браузерами: Google Chrome 9+, Firefox 4+, Opera 15+, Safari 5.1+, Internet Explorer 11 и Microsoft Edge. Узнать, какие браузеры поддерживают WebGL можно на [«могу ли я пользоваться WebGL»\);" onmouseout="hide\(\)">Can I use WebGL](#).

Используемые в three.js функции на языке JavaScript или

Вот несколько функций, используемых в **three.js**. Для некоторых из них может потребоваться дополнение полифиллами.

| ФУНКЦИИ FEATURE | ОБЛАСТЬ ПРИМЕНЕНИЯ USE SCOPE | МОДУЛИ MODULES |
|--------------------|---------------------------------|---|
| Typed Arrays | Исходник | BufferAttribute, BufferGeometry и т.д. |
| Web Audio API | Исходник | Audio, AudioContext, AudioListener и т.д. |
| WebVR API | Исходник | WebVRManager и т.д. |

| | | |
|------------------|----------|---|
| Blob | Исходник | FileLoader, etc. |
| Promise | Примеры | GLTFLoader, GLTF2Loader, WebVR, VREffect и т.д. |
| Fetch | Примеры | ImageBitmapLoader и т.д. |
| File API | Примеры | GLTFFormatter и т.д. |
| URL API | Примеры | GLTFLoader и т.д. |
| Pointer Lock API | Примеры | PointerLockControls |

Полифиллы

Просто импортируйте полифиллы, исходя из своих требований. Если в качестве примера взять IE9, то потребуются полифиллы как минимум к этим функциям:

- Typed Arrays
- Blob

Рекомендуемые полифиллы

- [core-js](#)
- [typedarray.js](#)
- [ES6-Promise](#)
- [Blob.js](#)
- [fetch](#)

Обнаружение совместимости браузера с WebGL

Несмотря на то, что проблема встречается все реже и реже, некоторые устройства или браузеры все ещё не могут работать с WebGL.

Описанный ниже метод проверяет, поддерживается ли WebGL и, если это не так, выводит сообщение пользователю.

Добавьте с своему javascript-коду [Detector.js](#) и запускайте следующий код прежде чем пытаться что-либо отобразить.

```
if (Detector.webgl) {  
    // Initiate function or other initializations here  
    // Здесь располагается инициализация функции или другие инициализации  
    animate();  
} else {  
    var warning = Detector.getWebGLErrorMessage();  
    document.getElementById('container').appendChild(warning);  
}
```

Как все это запустить локально

Если используются только конструкции, предусмотренные в Three.js, и никаких текстур дополнительно не загружается, веб-страницы должны работать прямо из файловой системы, просто сделайте двойной клик в файловом менеджере по HTML файлу и он должен появиться и начать работать в браузере (в адресной строке будет видно `file:///yourFile.html`).

Содержимое, загружаемое из внешних файлов

Если модели или текстуры загружаются из внешних файлов, из-за ограничений безопасности по

с англ. «Принцип одинакового источника») — это важная концепция безопасности для некоторых языков программирования на стороне клиента, таких как JavaScript.

'+' Политика разрешает сценариям, находящимся на страницах одного сайта, доступ к методам и свойствам друг друга без ограничений, но предотвращает доступ к большинству методов и свойств для страниц на разных сайтах.

'+' Однаковые источники — это источники, у которых совпадают три признака: домен, порт, протокол.');" onmouseout="hide()">правилу ограничения домена браузера, загрузка из файловой системы завершится ошибкой и выдачей сообщения с описанием исключения безопасности.

Примечание переводчика: Вот ссылки на статьи в Википедии про [Правило ограничения домена](#) и [Same Origin Policy](#).

Существует два способа решения этой проблемы:

1. Изменить для локальных файлов правила безопасности в браузере. Это позволит получить доступ к своей странице как: `file:///yourFile.html`
2. Запускать файлы из локального веб-сервера. Это позволит получить доступ к своей странице как:
`http://localhost/yourFile.html`

При использовании первого варианта имейте в виду, что если

используете тот же самый браузер и для обычного веб-серфинга, можно открыть некоторые уязвимости. Для безопасности, можно создать отдельный профиль/ярлык браузера только для локального применения. Давайте рассмотрим каждый вариант по очереди.

Изменение политики безопасности локальных файлов в браузере Safari

С помощью панели настройки включите меню разработки, через Advanced (дополнительно) => "Show develop menu in menu bar" (показывать меню разработки в панели меню).

Затем в меню "Develop" (разработка) браузера выберите "Disable local file restrictions" (отключить ограничения локальных файлов), стоит также отметить, что Safari имеет несколько странное поведение при работе с кэшем, так что в том же меню желательно использовать опцию "Disable caches" (отключить кэш); если редактирование и отладка ведутся с помощью Safari.

Chrome

Вначале закройте все работающие экземпляры браузера Chrome. Главное слово здесь "все".

В операционной системе Windows можно проверить число используемых экземпляров Chrome через Панель задач. Кроме того, если в системном трее видно иконку Chrome, можно открыть его контекстное меню и кликнуть 'Exit' - выйти. Это должно закрыть все экземпляры браузера Chrome.

Затем запустите исполняемый файл Chrome с флагом командной строки:

```
chrome --allow-file-access-from-files
```

В ОС Windows, возможно, самым простым способом является создание специального ярлыка, в котором добавлен флаг, указанный выше (правый клик по иконке ярлыка вызовет контекстное меню, в котором нужно сделать левый клик на строке

Свойства (Properties). В открывшемся окне свойств ярлыка, в строке Объект (Target) и нужно добавить упомянутый флаг).

В операционной системе Mac OSX это делается с помощью
`open /Applications/Google\ Chrome.app --args --allow-file-access-Firefox`

1. Наберите в адресной строке `about:config`
2. Найдите параметр
`security.fileuri.strict_origin_policy`
3. Установите его как `false`

Запуск локального сервера

На многих языках программирования имеются встроенные простые HTTP-серверы. Они не настолько полнофункциональны, как реально работающие сервера, подобные [Apache](#) или [NGINX](#), тем не менее они должны быть достаточны для тестирования приложения three.js.

Запуск сервера Python

Если установлен [Python](#), то его должно быть достаточно для запуска сервера из командной строки (из рабочего каталога):

```
//Python 2.x
python -m SimpleHTTPServer

//Python 3.x
python -m http.server
```

Он будет обслуживать файлы из текущего каталога в localhost по 8000 порту, то есть, в адресной строке наберите:

```
http://localhost:8000/
```

Запуск сервера Ruby

Если установлен Ruby, можно получить тот же результат, запустив взамен следующее:

```
ruby -r webrick -e "s = WEBrick::HTTPServer.new(:Port => 8000, :D
```

Запуск сервера PHP

В PHP также имеется встроенный сервер, начиная с версии php 5.4.0:

```
php -S localhost:8000
```

Запуск сервера Node.js

В Node.js имеется простой пакет HTTP сервера. Для установки:

```
npm install http-server -g
```

Для запуска (из локального каталога):

```
http-server . -p 8000
```

Запуск сервера lighttpd на Mac

Lighttpd - это очень легковесный веб-сервер общего назначения. Сейчас расскажем о его установке на OSX с помощью HomeBrew. В отличие от других серверов, обсуждаемых здесь, lighttpd - это полноценный сервер, готовый к реальной работе.

1. Устанавливаем сервер через homebrew

```
brew install lighttpd
```
2. В каталоге, где нужно запустить веб-сервер, создаем файл настройки с названием lighttpd.conf. [Вот здесь](#) имеется пример.
3. В файле настройки заменяем значение параметра server.document-root на каталог, в котором нужно обслуживать файлы.
4. Запускаем его

```
lighttpd -f lighttpd.conf
```
5. Переходим на `http://localhost:3000/` и он будет обслуживать статические файлы из выбранного каталога.

Другие простейшие варианты обсуждаются например на [Stack Overflow](#).

Рисование линий

Предположим, нужно нарисовать линию или круг, без каркасной [сетки](#). Вначале нужно установить [рендерер \(визуализатор\)](#), [сцену](#) и камеру (смотрите страницу [Создание сцены](#)).

Вот код, который для этого будет использоваться:

```
var renderer = new THREE.WebGLRenderer();
renderer.setSize(window.innerWidth, window.innerHeight);
document.body.appendChild(renderer.domElement);

var camera = new THREE.PerspectiveCamera(45, window.innerWidth / wi
camera.position.set(0, 0, 100);
camera.lookAt(new THREE.Vector3(0, 0, 0));

var scene = new THREE.Scene();
```

Следующее, что нужно сделать, это определить материал. Для линий можно использовать [LineBasicMaterial](#) или [LineDashedMaterial](#).

```
//create a blue LineBasicMaterial (создадим синий LineBasicMaterial)
var material = new THREE.LineBasicMaterial({ color: 0x0000ff });
```

После определения материала нужно определиться с [Geometry](#) или [BufferGeometry](#) с некоторым количеством вершин (рекомендуется использовать BufferGeometry как более производительную, однако для упрощения здесь будет использована Geometry):

```
var geometry = new THREE.Geometry();
geometry.vertices.push(new THREE.Vector3(-10, 0, 0));
geometry.vertices.push(new THREE.Vector3(0, 10, 0));
geometry.vertices.push(new THREE.Vector3(10, 0, 0));
```

Обратите внимание, что линии проведены между каждой последующей парой вершин, но не между первой и последней (то есть, эта линия не замкнута).

Теперь, когда есть точки для двух линий и материал, можно сложить все вместе для формирования линии.

```
var line = new THREE.Line(geometry, material);
```

Все что осталось, вывести (добавить) её на сцену и вызвать [рендерер](#).

```
scene.add(line);
```

```
renderer.render(scene, camera);
```

Теперь вы должны увидеть стрелку из двух синих линий, направленную вверх.

Создание текста

Часто бывает так, что в приложении three.js нужно использовать текст - вот несколько способов, как можно это сделать.

1. DOM + CSS

Использование HTML - это наиболее простой и быстрый способ добавления текста. This is the method used for descriptive overlays in most three.js examples.

Можно добавить содержимое в

```
<div id="info">Description</div>
```

и использовать CSS-разметку для размещения в абсолютно независимой позиции и поверх всего остального по z-индексу, особенно если приложение three.js работает в полноэкранном режиме.

```
#info {  
    position: absolute;  
    top: 10px;  
    width: 100%;  
    text-align: center;  
    z-index: 100;  
    display: block;  
}
```

2. Нарисовать текст на canvas и использовать его как тексту

Используйте этот способ, если хочется просто нарисовать текст на плоскости, на своей three.js-сцене.

3. Создать модель текста в выбранной вами программе 3D-г и экспортовать её в three.js

Используйте этот способ, если предпочитаете работать со своей программой 3D-графики и вставьте эту модель в three.js.

4. Procedural Text Geometry

Если предпочитаете работать исключительно в three.js или создавать If you prefer to work purely in THREE.js or to create procedural and dynamic 3D text geometries, you can create a mesh whose geometry is an instance of THREE.TextGeometry:

```
new THREE.TextGeometry( text, parameters );
```

Для того, чтобы это сработало, для TextGeometry будет нужен экземпляр THREE.Font, установленный в качестве значения его параметра "font" - шрифт. Посмотрите страницу [TextGeometry](#) для получения дополнительных сведений о том, как это можно сделать с описанием каждого принимаемого параметра и списком [JSON](#)-шрифтов, поставляемых в составе самого THREE.js.

Примеры

[WebGL / geometry / text](#)

[canvas / geometry / text](#)

[WebGL / shadowmap](#)

Если шрифт не работает или нужно использовать шрифт, которого здесь нет, существует [руководство со скриптом на языке Python](#) для [Blender'a](#), который позволяет экспортить текст в понятном для Three.js формате JSON.

Руководство по миграции и стилю кода

Руководство [по миграции](#) есть, по переводу программ three.js, сделанных в одной версии, для работы в другой версии three.js хранится на [wiki](#). Оно содержит список изменений каждой версии three.js, начиная с релиза r45.

Его можно найти [здесь](#).

Примечание переводчика: Вики (англ. wiki) — веб-сайт, ... читать далее

Весь код и примеры на three.js написаны в стиле кода от Mr.doob. Конечно, можно использовать любой, предпочтаемый вами для работы, стиль, но при добавлении кода в библиотеку или в примеры, требуется следовать этому руководству. Подробности можно найти [здесь](#).

ЧАСТО ЗАДАВАЕМЫЕ ВОПРОСЫ

Какой формат импорта/экспорта лучше всего поддерживает

TODO

Почему в примерах присутствуют теги meta для окна просмо

```
<meta name="viewport" content="width=device-width, user-scalable=no"
```

Эти тэги управляют размерами окна просмотра (viewport) и масштабом для браузеров мобильных телефонов (где содержимое страницы может отображаться в другом размере, чем видимое окно просмотра).

<http://developer.apple.com/library/safari/#documentation/AppleApplication>

https://developer.mozilla.org/en/Mobile/Viewport_meta_tag (англ.)

https://developer.mozilla.org/ru/docs/Mozilla/Mobile/Viewport_meta_tag
на русском

Как можно сохранить масштаб сцены при изменении размер

Хочется чтобы все объекты, независимо от их расстояния до камеры, выглядели одинаково, даже при изменении размера окна. Ключевое уравнение для решения этой задачи - это формула для видимой высоты (visible_height) на заданном расстоянии от камеры (distance_from_camera):

```
visible_height = 2 * Math.tan( ( Math.PI / 180 ) * camera.fov / 2 )
```

Если увеличивается высота окна на определенный процент, то и видимая высота на всех расстояниях также увеличивалась на тот же самый процент. Этого нельзя сделать изменением положения камеры. Вместо этого следует изменять поле просмотра камеры (параметр fov - field-of-view). [Пример](#).

Почему часть моего объекта невидима?

Это может происходить из-за выбора грани. У граней имеется ориентир, который определяет какая сторона является передней и видимой, а какая - задней и невидимой. И в обычных условиях при выборе удаляется задняя, невидимая сторона. Если проблема в этом, измените сторону материала на THREE.DoubleSide.

```
material.side = THREE.DoubleSide
```

Ссылки на полезные ресурсы

Ниже приведена коллекция ссылок, которые возможно будут полезными при изучении three.js.

Если вы нашли что-либо, что хочется сюда добавить, или считаете, что одна из приведенных ссылок больше не уместна или не работает, не стесняйтесь кликнуть кнопку 'edit' (редактировать) справа вверху и внести некоторые изменения!

Отметьте также, что three.js довольно быстро развивается, многие из этих ссылок будут содержать устаревшие сведения - если что-то работает не так, как ожидалось или как указывается в этой ссылке, проверьте консоль браузера на наличие предупреждений и ошибок, соответствующие разделы этой справки и особенно страницу [DeprecatedList](#) (список устаревших элементов API).

В дополнение к этой странице, mrdoob поддерживает коллекцию ссылок, связанных с three.js в Google+. Посмотри их [здесь](#).

Справочные форумы

Three.js официально использует [Stack Overflow](#) для запросов справочной информации.

Если в чем-либо требуется помочь, обращайтесь туда. НЕ НУЖНО создавать проблемы просьбами помочь на Github'e.

Учебники и курсы

Начало работы с three.js

- [Beginning with 3D WebGL](#) от [Rachel Smith](#).
- [Animating scenes with WebGL and three.js](#) анимация сцен с WebGL и three.js

Более расширенные и дополняющие статьи и курсы

- [Collection of tutorials](#) набор учебников от [CJ Gammon](#).
- [Glossy spheres in three.js](#) блестящие сферы в three.js.

- [Interactive 3D Graphics](#) - бесплатный курс по Udacity, обучающий основам 3D графики и использующий в качестве инструмента кодирования three.js.
- [Aerotwist](#) учебники от [Paul Lewis](#).
- [Learning Three.js](#) – a blog with articles dedicated to teaching three.js
- [Animated selective glow in Three.js](#) от [BKcore](#)

Учебники на других языках

- [Building A Physics Simulation Environment](#) построение среды моделирования физических процессов - учебник three.js на японском

Дополнительная документация

- [Three.js walking map](#) - a graphical breakdown of the structure of a three.js scene.

Новости и обновления

- [Three.js на reddit](#)
- [WebGL на reddit](#)
- [Learning WebGL Blog](#) – Авторитетный источник новостей о WebGL.
- [Three.js posts](#) на Google+ – частые сообщения о Three.js

Примеры

- [Professor Stemkoskis Examples](#) (Примеры профессора Стемкоскис) - Сборник дружественных примеров для начинающих, построенных с использованием three.js версии r60a.
- [Официальные примеры three.js](#) - эти примеры сохраняются как часть репозитория three.js и всегда используют последнюю версию three.js.
- [Official three.js dev branch examples](#) - Same as the above, except these use the dev branch of three.js, and are used to check that everything is working as three.js being developed.

Инструменты

- [physgl.org](#) - javascript front-end with wrappers to three.js, чтобы донести графику WebGL до студентов, изучающих физику и математику. to bring WebGL graphics to students learning physics and math.
- [link:[Whitestorm.js](http://whitestormjs.xyz/)] – A wrapper around Three.js and custom [link:[physi.js](https://github.com/chandlerprall/Physis)].
- [Three.js Inspector](#)
- [ThreeNodes.js](#).

Старые ссылки

Эти ссылки хранятся в исторических целях - в них тоже можно найти что-то полезное, но имейте ввиду, содержащаяся в них информация относится к очень старым версиям three.js.

- [AlterQualia at WebGL Camp 3](#)
- [Yomotsus Examples](#) (примеры) - коллекция примеров, с использованием three.js версии r45.
- [Introduction to Three.js](#) - введение в Three.js от [Ilmari Heikkinen](#) (слайдшоу).
- [WebGL and Three.js](#) от [Akihiro Oyamada](#) (слайдшоу).
- [Fast HTML5 game development using three.js](#) от [BKcore](#) (видео).
- [Trigger Rally](#) от [jareiko](#) (видео).
- [ThreeFab](#) - редактор сцен, поддерживался приблизительно до three.js версии r50.
- [Max to Three.js workflow tips and tricks](#) от [BKcore](#)
- [A whirlwind look at Three.js](#) (Беглый взгляд на Three.js) от [Paul King](#)

СЛЕДУЮЩИЕ ДЕЙСТВИЯ

Как это все обновлять

Все объекты по умолчанию автоматически обновляют свои матрицы, если были добавлены на сцену при помощи

```
var object = new THREE.Object3D;  
scene.add( object );
```

или если они являются дочерними по отношению к другому объекту, уже добавленному на сцену:

```
var object1 = new THREE.Object3D;  
var object2 = new THREE.Object3D;  
  
object1.add( object2 );  
scene.add( object1 ); //object1 and object2 will automatically update  
//object1 и object2 будут автоматически обновлять свои матрицы
```

Впрочем, если известно что объект будет статичным, можно запретить автоматическое обновление и обновлять матрицу трансформации вручную, когда потребуется.

```
object.matrixAutoUpdate = false;  
object.updateMatrix();
```

Geometries

Примечание переводчика: Здесь рассматривается обновление геометрических параметров объекта, т.е. вершины, грани, их расположение, нормали, цвета и т.д.

BufferGeometry

BufferGeometries сохраняют информацию (такую как положение вершин, индексы граней, нормали, цвета, текстурные координаты (и координатами на текстуре (U, V - эти буквы обозначают оси двумерной текстуры,
потому что «X», «Y» и «Z» уже используются для обозначения осей 3D-объекта
в пространстве модели). Значения U и V обычно изменяются от 0 до 1.); " onmouseout="hide()">UV) и все атрибуты, установленные пользователем) в буферах, которые являются типованными массивами (здесь описание этих массивов на русском языке). Это делает работу BufferGeometries в общем-то быстрее, по сравнению с обычными Geometries за счет того, что с ними сложнее работать.

Что касается обновления BufferGeometries, то самое главное для понимания, не стоит изменять размер буферов (это очень затратно, по существу равнозначно созданию новой геометрии). Однако можно обновлять содержимое буферов.

Это значит, если известно, что какой-нибудь атрибут вашей BufferGeometry будет расти (например, количество вершин), то следует изначально выделить буфер, достаточно большой, для хранения любого числа новых вершин, которые могут быть созданы. Конечно, это также означает, что для вашей BufferGeometry существует максимальный размер - нет способа создать BufferGeometry, которую можно было бы успешно расширять до бесконечности.

В качестве примера попробуем нарисовать линию, которая увеличивается во время визуализации. Выделим место в буфере

для 500 вершин, но сначала, при помощи метода [BufferGeometry.drawRange](#), нарисуем только две.

```
var MAX_POINTS = 500;

// geometry (геометрия)
var geometry = new THREE.BufferGeometry();

// attributes (атрибуты)
var positions = new Float32Array( MAX_POINTS * 3 ); // 3 vertices per
geometry.addAttribute( 'position', new THREE.BufferAttribute( posit

// draw range (рисуем ряд точек)
var drawCount = 2; // draw the first 2 points, only (рисуем только)
geometry.setDrawRange( 0, drawCount );

// material (материал)
var material = new THREE.LineBasicMaterial( { color: 0xff0000, line

// line (линия)
var line = new THREE.Line( geometry, material );
scene.add( line );
```

Далее будем случайным образом добавлять точки к линии при помощи шаблона в виде:

```
var positions = line.geometry.attributes.position.array;

var x, y, z, index;
x = y = z = index = 0;

for ( var i = 0, l = MAX_POINTS; i < l; i ++ ) {

    positions[ index ++ ] = x;
    positions[ index ++ ] = y;
    positions[ index ++ ] = z;

    x += ( Math.random() - 0.5 ) * 30;
    y += ( Math.random() - 0.5 ) * 30;
    z += ( Math.random() - 0.5 ) * 30;

}
```

Если нужно изменить **количество точек**, отображаемых после первой визуализации, сделайте следующее:

```
line.geometry.setDrawRange( 0, newValue );
```

Если, после первой визуализации, нужно изменить значения данных

положения, следует установить флаг needsUpdate, вот так:

```
line.geometry.attributes.position.needsUpdate = true;  
// required after the first render (запрашивается после первой виз
```

[Вот пУтанка](#), представленная анимированной линией, которую можно приспособить под свои нужды.

Примеры:

[WebGL / custom / attributes](#)

[WebGL / buffergeometry / custom / attributes / particles](#)

[Geometry](#)

Следующие флаги управляют обновлением различных атрибутов геометрии. Устанавливайте флаги только для атрибутов, требующих обновления, так как обновления - затратная вещь. После изменения буферов эти флаги автоматически сбрасываются обратно к значению false. Если требуется продолжать обновление буферов, нужно сохранить их как true. Обратите внимание, что это относится только к [Geometry](#), а не к [BufferGeometry](#).

```
var geometry = new THREE.Geometry();  
geometry.verticesNeedUpdate = true;  
geometry.elementsNeedUpdate = true;  
geometry.morphTargetsNeedUpdate = true;  
geometry.uvsNeedUpdate = true;  
geometry.normalsNeedUpdate = true;  
geometry.colorsNeedUpdate = true;  
geometry.tangentsNeedUpdate = true;
```

Кроме этого, в версиях, предшествующих [r66](#), сеткам (mesh) необходимо включать флаг dynamic (для сохранения внутренних типизированных массивов):

```
// removed after r66 (удаляется после версии r66)  
geometry.dynamic = true;
```

Пример:

[WebGL / geometry / dynamic](#)

Материалы

Все однотипные значения могут быть свободно изменены All uniforms values can be changed freely (например, цвета, текстуры, непрозрачность и так далее), значения отправляются в шейдер с каждым кадром. values are sent to the shader every frame.

Примечание переводчика: В области компьютерной графики, шейдер - это компьютерная программа
[... читать далее](#)

Параметры связанные с GLstate также могут быть изменены в любой момент (depthTest, blending, polygonOffset, и т.д.).

Плоское (flat) / плавное (smooth) shading is baked into normals.
Требуется сброс буфера нормалей (смотрите выше).

Следующие свойства нельзя просто так изменить во время выполнения (после того, как материал был визуализирован хотя бы раз):

- типы и число uniforms
- типы и число источников освещения
- наличие или отсутствие

текстуры (texture)
тумана (fog)
цвета вершин (vertex colors)

создания кожи, встречается также написание скиннинг)

- это один из этапов постановки 3d-персонажа, когда модель персонажа привязывается (скинится) к скелету.

'+'Делается это для того, чтобы при движении скелета двигалась и

сама модель персонажа.

'+'Это достаточно трудоемкий процесс, поскольку нужно правильно

назначить вес (англ. weight) для каждой вершины модели.

Чем больше вес, тем больше влияет конкретная кость

на конкретную вершину 3d-модели.';'"

onmouseout="hide()">скининга (skinning)

анимации, визуальный эффект, создающий впечатление плавной

трансформации одного объекта в другой. Встречается в трёхмерной

и двухмерной (как растровой, так и векторной) графике.
' +'Для создания эффекта используются как минимум два изображения,

на которых художник задаёт в зависимости от использующегося

программного обеспечения опорные фигуры или ключевые точки

(т. н. маркеры, или метки), которые помогают компьютеру выполнить

правильный морфинг, то есть создать изображения промежуточных

состояний (интерполируя имеющиеся данные).

' +'Морфинг также часто используется для создания анимации, когда не

стоит задача добиться эффекта превращения одного объекта в другой,

а требуется лишь выстроить промежуточные состояния между двумя

(и более) ключевыми положениями анимируемого объекта.';" onmouseout="hide()">морфинга (morphing)

теневой карты (shadow map)

вектор цвета в формате RGBA. Альфа-компонент определяет

непрозрачность материала в диапазоне от 1.0 означающего полную

непрозрачность до 0.0, означающего полную прозрачность.
' +'Для того чтобы создавать прозрачные и полупрозрачные

объекты,

необходимо разрешить тестировать буфер альфа-канала и включить

механизм под названием альфа-смешивание.
' +'При включённом альфа-тесте сравнивается входящее значение
альфа-канала с эталонным значением. Фрагмент принимается или
отклоняется в зависимости от результатов сравнения.');"
onmouseout="hide()">альфа-теста (alpha test)

Изменения в них требуют создания новой шейдерной программы.
Нужно будет установить

`material.needsUpdate = true`

Имейте в виду, что это может быть довольно медленно и вызывать подергивание кадров (особенно в Windows, поскольку шейдерная компиляция в DirectX медленнее, чем в OpenGL).

Для повышения плавности работы можно в некоторой степени имитировать изменения этих функций, с помощью «фиктивных» значений, таких как освещение с нулевой интенсивностью, белых текстур или тумана с нулевой плотностью.

Можно свободно изменять материал, используемый для частей геометрии, однако нельзя изменять способ разделения объекта на части (в соответствии с материалами грани). You can freely change the material used for geometry chunks, however you cannot change how an object is divided into chunks (according to face materials).

Если во время выполнения нужны разные конфигурации материалов:

Если число материалов / частей невелико, можно заранее предварительно разделить объект (например, для человека - волосы (hair) / лицо (face) / тело (body) / верхняя одежда (upper clothes) / брюки (trousers), для автомобиля - перед (front) / боковые стороны (sides) / верх (top) / стекла (glass) / шины (tire) / салон (interior)).

Если число велико (к примеру каждое лицо/грань может быть потенциально различным), рассмотрите другое решение, такое как использование атрибутов/текстур для приведения к другому внешнему виду.

Примеры:

[WebGL / materials / cars](#)

[WebGL / webgl_postprocessing / dof](#)

Текстуры

Если текстуры изображения, элемента canvas, видео и данных были изменены, то у них должен быть установлен следующий флаг:

Image, canvas, video and data textures need to have the following flag set if they are changed:

```
texture.needsUpdate = true;
```

Обновление целей визуализации произойдет автоматически. Render targets update automatically.

Примеры:

[WebGL / materials / video](#)

[WebGL / rtt](#)

Камеры

Положение в пространстве и направление съемки камеры обновляются автоматически. Если нужно изменить параметры

- `fov` (поле просмотра)
- `aspect` (соотношение сторон)
- `near` (ближняя плоскость отсечения)
- `far` (дальняя плоскость отсечения)

то требуется пересчитать матрицу проекции:

```
camera.aspect = window.innerWidth / window.innerHeight;  
camera.updateProjectionMatrix();
```

Матричные преобразования

В [*three.js*](#) для кодирования 3-мерных преобразований - перемещения (изменения положения), вращения и масштабирования используются **матрицы**. Каждый экземпляр [Object3D](#) имеет свойство [matrix](#), в котором хранится положение, угол поворота и масштаб этого объекта. На этой странице описывается как обновлять преобразование (трансформацию) объекта.

Преимущества свойств и matrixAutoUpdate

Существует два способа обновления преобразования объекта:

1. Изменить свойства объекта `position`, `quaternion` и `scale`, и позволить Three.js пересчитать матрицу объекта с этими свойствами:

```
object.position.copy(start_position);  
object.quaternion.copy(quaternion);
```

По умолчанию, свойство `matrixAutoUpdate` устанавливается равным `true`, так что матрица будет пересчитана автоматически. Если объект статичен или нужно вручную определять когда будет происходить пересчет матрицы, наилучшую производительность можно получить установкой этого свойства как `false`:

```
object.matrixAutoUpdate = false;
```

И после изменения каких-либо свойств, вручную обновить матрицу:

```
object.updateMatrix();
```

2. Непосредственно изменить матрицу объекта. Класс [Matrix4](#) имеет различные методы для изменения матрицы:

```
object.matrix.setRotationFromQuaternion(quaternion);  
object.matrix.setPosition(start_position);  
object.matrixAutoUpdate = false;
```

Обратите внимание, что в этом случае свойство `matrixAutoUpdate` **должно** быть установлено как `false`, при этом следует убедиться, что **не** было вызова `updateMatrix`. Вызов `updateMatrix` перебьет изменения матрицы, сделанные вручную, пересчитав матрицу для

position, scale и так далее.

Матрицы объекта и world

An object's [page:Object3D.matrix matrix stores the object's transformation **relative** to the object's [page:Object3D.parent parent; to get the object's transformation in **world** coordinates, you must access the object's [page:Object3D.matrixWorld.

При изменениях в преобразовании родительского или дочернего объекта можно запросить обновление свойства [matrixWorld](#) дочернего объекта вызовом метода [updateMatrixWorld](#).

Вращение и кватернионы

Three.js предоставляет два способа представления трехмерных вращений: [углы Эйлера](#) и [Quaternions](#), а также методы конвертирования между ними. Three.js provides two ways of representing 3D rotations: [Euler angles](#) and [Quaternions](#), as well as methods for converting between the two. Euler angles are subject to a problem called "gimbal lock," where certain configurations can lose a degree of freedom (preventing the object from being rotated about one axis). По этой причине, вращение объекта **всегда** сохраняется в его свойстве [quaternion](#). For this reason, object rotations are **always** stored in the object's [quaternion](#).

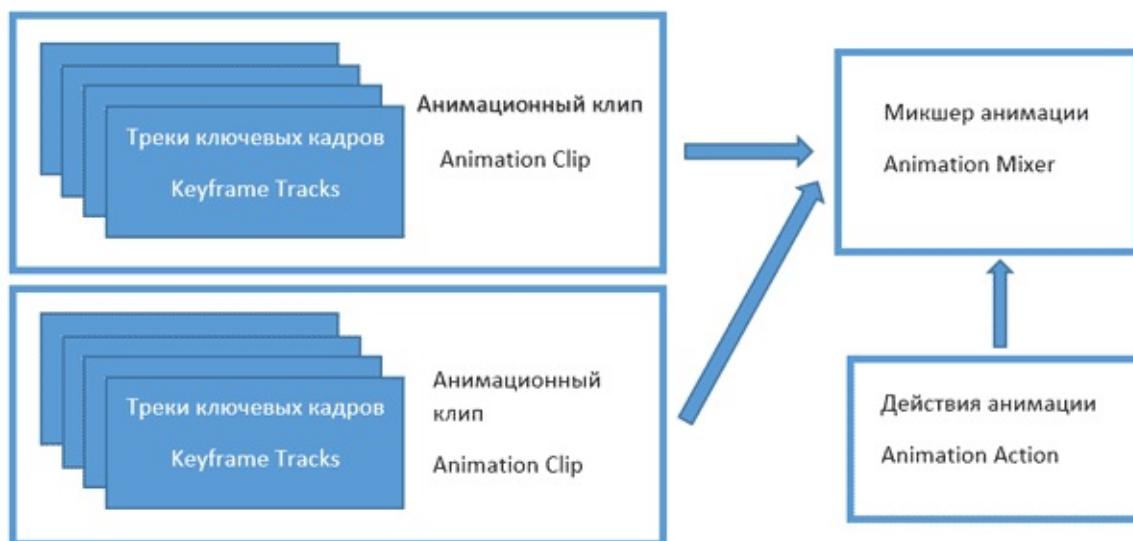
Предыдущие версии библиотеки включали в себя свойство useQuaternion, которое, будучи установленное как false, приводило к расчету [матрицы](#) объекта из углов Эйлера. Эта практика устарела - взамен следует использовать метод [setRotationFromEuler](#) method, which will update the quaternion.

Система анимации

Обзор

В рамках анимационной системы ***three.js*** можно анимировать различные свойства модели: кости (bone) [skinned and rigged model](#), [цели морфинга \(morph targets\)](#), различные свойства материала (цвета, непрозрачность, логику), видимость и преобразования. Свойствами анимации могут быть постепенное появление (faded in), постепенное исчезновение (faded out), плавное появление на фоне плавного исчезновения (crossfaded) и деформация (warped). «Веса» (weight) и временные масштабы (time scales) различных одновременных анимаций как на одном и том же объекте, так и на разных объектах, могут быть изменены независимо друг от друга. Можно синхронизировать различные анимации как на одном и том же объекте, так и на разных объектах.

Чтобы достичь всего этого в единственной однородной системе, система анимации ***three.js*** [полностью изменилась в 2015 году](#) (помните об устаревшей информации!), и теперь его архитектура, походит на Unity/Unreal Engine 4. На этой странице дается краткий обзор основных компонентов этой системы и способов их совместной работы.



Анимационные клипы Animation Clips

Если имеется успешно импортированный анимированный 3D объект (неважно имеются ли в нем кости или цели морфинга, или и то и другое) - например, экспортенный из Blender'a с помощью [Blender exporter](#) и загруженный на сцену [**three.js**](#) загрузчиком [JSONLoader](#), - то одним из свойств геометрии загружаемой сетки (mesh) должен быть массив, поименованный как "animations", содержащий [AnimationClip](#)'ы для данной модели (смотрите ниже список возможных загрузчиков). If you have successfully imported an animated 3D object (it doesn't matter if it has bones or morph targets or both) - for example exporting it from Blender with the [link:[JSONLoader\] - , one of the geometry's properties of the loaded mesh should be an array named "animations", containing the \[page:\[AnimationClip AnimationClips\\] for this model \\(see a list of possible loaders below\\).\]\(#\)](https://github.com/mrdoob/three.js/tree/master/utils/exporters/blender Blender exporter] and loading it into a three.js scene using [page:<a href=)

Каждый «AnimationClip» обычно содержит данные для определенной активности объекта. Если сеткой, например является персонаж, то для цикла ходьбы может быть один AnimationClip, для прыжка - второй, третий для уклонения в сторону и так далее. Each *AnimationClip* usually holds the data for a certain activity of the object. If the mesh is a character, for example, there may be one AnimationClip for a walkcycle, a second for a jump, a third for sidestepping and so on.

Треки ключевых кадров Keyframe Tracks

Внутри такого «AnimationClip» данные для каждого свойства анимации хранятся в отдельном [KeyframeTrack](#). Допустим, персонифицированный объект имеет [скелет](#) и один трек ключевого кадра может хранить данные изменений положения кости предплечья во времени, другой трек - данные изменения поворота этой же самой кости, а третий отслеживать положение, поворот или изменение масштаба другой кости и так далее. Понятно, что AnimationClip может состоять из множества подобных треков.

Предположим, что у модели имеются [цели морфинга](#) (например, одна цель морфинга показывает приветливое лицо, а другая - сердитое), каждый трек хранит сведения о том, как [воздействие \(influence\)](#) некоторой цели морфинга изменяется во время выполнения клипа. Assumed the model has [page:Geometry.morphTargets morph targets] (for example one morph target showing a friendly face and another showing an angry face), each track holds the information as to how the [page:Mesh.morphTargetInfluences influence] of a certain morph target changes during the performance of the clip.

Микшер анимации (Animation Mixer)

Сохраненные данные формируют только основу анимации - фактическое воспроизведение контролируется [AnimationMixer](#). Можно представить это не только как игрока для анимации, но и как симуляцию аппаратного обеспечения, например, реальной микшерной консоли, которая может одновременно управлять несколькими анимациями, смешивая и объединяя их. The stored data form only the basis for the animations - actual playback is controlled by the [page:AnimationMixer]. You can imagine this not only as a player for animations, but as a simulation of a hardware like a real mixer console, which can control several animations simultaneously, blending and merging them.

Действия анимации (Animation Actions)

Собственно сам «AnimationMixer» имеет только очень мало (общих) свойств и методов, потому что им можно управлять с помощью [AnimationActions](#). The *AnimationMixer* itself has only very few (general) properties and methods, because it can be controlled by the [page:AnimationAction AnimationActions]. Настройкой «AnimationAction» можно определять когда конкретный «AnimationClip» будет воспроизводиться, устанавливаться в паузу или быть остановленным на одном из микшеров, будет ли он повторяться и если будет, как часто, должен ли он выполняться с

затуханием или масштабироваться по времени, и другими дополнительными особенностями вроде кроссфейдинга или синхронизации. By configuring an `*AnimationAction*` you can determine when a certain `*AnimationClip*` shall be played, paused or stopped on one of the mixers, if and how often the clip has to be repeated, whether it shall be performed with a fade or a time scaling, and some additional things, such crossfading or synchronizing.

Анимация групп объектов

Если нужно чтобы группа объектов приобрела совместно используемое состояние анимации, можно использовать [`AnimationObjectGroup`](#).

Поддерживаемые форматы и загрузчики

Обратите внимание, не все форматы моделей включают анимацию (в частности OBJ не включает), и что только некоторые загрузчики `three.js` поддерживают последовательности [`AnimationClip`](#). Note that not all model formats include animation (OBJ notably does not), and that only some three.js loaders support [page:`AnimationClip` `AnimationClip`] sequences. Several that *do* support this animation type:

- [`THREE.JSONLoader`](#)
- [`THREE.ObjectLoader`](#)
- `THREE.BVHLoader`
- `THREE.FBXLoader`
- [`THREE.GLTF2Loader`](#)
- `THREE.MMDLoader`
- `THREE.SEA3DLoader`

Обратите внимание, что в настоящее время 3ds max и Maya не могут экспортить несколько анимаций (то есть, анимаций, которые не находятся на одном и том же временном промежутке) непосредственно в один файл.

Пример

```
var mesh;  
  
// Create an AnimationMixer, and get the list of AnimationClip inst
```

```
// Создаем AnimationMixer и получаем список экземпляров AnimationClip
var mixer = new THREE.AnimationMixer( mesh );
var clips = mesh.animations;

// Update the mixer on each frame (обновляем микшер в каждом кадре)
function update () {
    mixer.update( deltaSeconds );
}

// Play a specific animation (проигрываем конкретную анимацию)
var clip = THREE.AnimationClip.findByName( clips, 'dance' );
var action = mixer.clipAction( clip );
action.play();

// Play all animations (проигрываем все анимации)
clips.forEach( function ( clip ) {
    mixer.clipAction( clip ).play();
} );
```

Сборочные инструменты

Проверка с помощью менеджера пакетов, вх

В данной статье рассказывается как получить *three.js* в среде [node.js](#), так чтобы можно было выполнять автоматические проверки. Тестирование можно запускать из командной строки или с помощью автоматизированных инструментов непрерывной интеграции [непрерывная интеграция — это практика разработки программного обеспечения, которая заключается в слиянии рабочих копий в общую основную ветвь разработки несколько раз в день и выполнении частых автоматизированных сборок проекта для скорейшего выявления и решения интеграционных проблем.'\);"](#) [onmouseout="hide\(\)">\(CI\)](#), вроде [Travis](#).

Более подробно про непрерывную интеграцию можно посмотреть в [Википедии](#) или на [Хабрахабре](#).

Сокращенный вариант

Если вам привычна работа с node и npm,

```
$ npm install three --save-dev
```

и к своей проверке добавьте

```
var THREE = require('three');
```

Создание проекта для тестирования с нуля

Если эти инструменты вам незнакомы, вот краткое руководство (для Linux процесс установки будет немного отличаться от работы в Windows, но команды npm идентичны).

Basic setup

1. Устанавливаем npm и node.js. Кратчайший путь обычно выглядит приблизительно так

```
$ sudo apt-get install -y npm nodejs-legacy  
# fix any problems with SSL in the default registry URL  
$ npm config set registry http://registry.npmjs.org/
```

2. Делаем каталог нового проекта

```
$ mkdir test-example; cd test-example
```

3. Запрашиваем прм для создания файла нового проекта to
create a new project file for you:

```
$ npm init
```

and accept all defaults by hitting Enter on all the prompts. This will create package.json.

4. Try and start the test feature with

```
$ npm test
```

This will fail, which is expected. If you look in the package.json, the definition of the test script is

```
"test": "echo \"Error: no test specified\" && exit 1"
```

Добавляем

многофункциональная среда для тестирования JavaScript, работающая на Node.js и в браузере, что делает асинхронное тестирование простым и интересным.
' +' Тестирование Mocha запускается поочередно, позволяя гибко и точно сообщать, а также сопоставлять неперехваченные исключения с правильными тестовыми примерами.');" onmouseout="hide()">mocha

Будем использовать [mocha](#).

1. Устанавливаем mocha с помощью

```
$ npm install mocha --save-dev
```

Обратите внимание, что node_modules/ создан и все зависимости окажутся там. Также отметьте, что package.json был обновлен: добавлено и обновлено свойство devDependencies при помощи --save-dev. Notice that node_modules/ is created and your dependencies appear in there. Also notice that your package.json has been updated: the property devDependencies is added and updated by the use of --save-dev.

2. Edit package.json to use mocha for testing. When test is invoked, we just want to run mocha and specify a verbose

reporter. By default this will run anything in test/ (not having directory test/ can run into npm ERR!, create it by mkdir test)

```
"test": "mocha --reporter list"
```

3. Перезапускаем тест с помощью

```
$ npm test.
```

Теперь он должен быть успешным, This should now succeed, reporting 0 passing (1ms) or similar.

Добавление three.js

1. Let's pull in our three.js dependency with

```
$ npm install three --save-dev
```

Если нужна другая версия three.js, используйте

```
$ npm show three versions
```

чтобы посмотреть, какая доступна. to see what's available. To tell npm the right one, use

```
$ npm install three@0.84.0 --save
```

(0.84.0 in this example). --save makes this a dependency of this project, rather than dev dependency. Для более подробных сведенийсмотрите документацию [здесь](#).

2. Mocha will look for tests in test/, so let's

```
$ mkdir test.
```

3. Finally we actually need a JS test to run. Let's add a simple test that will verify that the three.js object is available and working. Create test/verify-three.js containing:

```
var THREE = require('three');
var assert = require("assert");

describe('The THREE object', function() {
  it('should have a defined BasicShadowMap constant', function() {
    assert.notEqual('undefined', THREE.BasicShadowMap);
  }),

  it('should be able to construct a Vector3 with default values', function() {
    var vec3 = new THREE.Vector3();
    assert.equal(0, vec3.x);
  })
})
```

- Finally let's test again with \$ npm test. This should run the tests above and succeed, showing something like:

```
The THREE object should have a defined BasicShadowMap cons  
The THREE object should be able to construct a Vector3 wit  
2 passing (8ms)
```

Добавляем свой собственный код

Нужно сделать три вещи:

- Написать тест для ожидаемого поведения своего кода и разместить его в test/. [Вот пример](#) из реального проекта.
- Экспортируйте свой действующий код таким образом, чтобы nodejs мог его видеть, для использования в сочетании с require. Смотрите [пример здесь](#). Export your functional code in such a way that nodejs can see it, for use in conjunction with require. See it [link:[here](https://github.com/air/encounter/blob/master/js/Physics.js)].
- Require your code into the test file, in the same way we did a require('three') in the example above.

Пункты 2 и 3 будут зависеть от того, как вы управляете своим кодом. В примере Physics.js, приведенном ниже, Items 2 and 3 will vary depending on how you manage your code. In the example of Physics.js given above, the export part is right at the end. We assign an object to module.exports:

```
=====  
// make available in nodejs  
=====  
if (typeof exports !== 'undefined')  
{  
    module.exports = Physics;  
}
```

Работа с зависимостями

Если вы уже пользовались чем-то умным, вроде [Он оптимизирован для работы в браузере, но может использоваться и в других средах JavaScript, вроде Rhino и Node. Использование модульного загрузчика скрипта, подобного RequireJS, повышает скорость](#)

и качество кода.");" onmouseout="hide()" target="_blank">require.js или browserify, пропустите эту часть.

Обычно, проект three.js запускается в браузере. Следовательно, загрузка модуля браузером выполняет множество скриптовых тегов. Отдельные файлы проекта не должны беспокоиться о зависимостях. Module loading is hence done by the browser executing a bunch of script tags. Your individual files don't have to worry about dependencies. Однако, в контексте nodejs нет файла index.html, увязывающего все вместе, так что In a nodejs context however, there is no index.html binding everything together, so you have to be explicit.

Если экспортируется модуль, зависящий от других файлов, нужно указать node о их загрузке.

Вот один из подходов:

1. В начале кода модуля проверьте, находитесь ли вы в среде node.js.
2. Если это так, явно объявите зависимости.
3. Если нет, то вероятно вы в браузере, так что больше ничего не нужно делать.

Пример кода из Physics.js:

```
//=====
// setup for server-side testing (установка проверки на стороне сервера)
//=====
if (typeof require === 'function') // test for nodejs environment (
{
    var THREE = require('three');
    var MY3 = require('./MY3.js');
}
```

АНИМАЦИЯ

AnimationAction

AnimationActions планирует (составляет расписание, по которому происходит) выполнение анимаций, которые хранятся в [AnimationClips](#).

Примечание: Большинство методов AnimationAction могут быть объединены в цепочки.

Обзор различных элементов системы анимации ***three.js*** можно найти в статье "[Системы анимации](#)" раздела "[Следующие шаги](#)" данного руководства.

Конструктор

```
AnimationAction(  
    mixer - микшер, смеситель' );" onmouseout="hide()">mixer, clip, loca
```

mixer - микшер, смеситель');" onmouseout="hide()">mixer - AnimationMixer, который управляется данным действием.

clip - AnimationClip, что содержит данные анимации для этого действия.

localRoot - корневой объект, в котором выполняется данное действие.

Примечание: Вместо непосредственного вызова этого конструктора следует создавать экземпляр AnimationAction при помощи метода [AnimationMixer.clipAction](#), поскольку для улучшения производительности он предоставляет кэширование.

Свойства

```
зажать (приостановить) по завершении' );" onmouseout="hide()">.cla
```

Если clampWhenFinished установлен как true, анимация автоматически будет [приостановлена](#) на своем последнем кадре. Если clampWhenFinished установлен как false, то когда закончится последний цикл действия, свойство [enabled](#) будет автоматически переключено в false, так что дальше это действие не будет оказывать никакого влияния. Значением по умолчанию является false.

Примечание: clampWhenFinished не оказывает влияния, если действие прервано (свойство работает только в случае, когда последний цикл действия реально завершился).

[.enabled](#)

Установка enabled как false отключает данное действие, так что оно не будет оказывать никакого влияния. Значением по умолчанию является true.

Если действие включено повторно, анимация продолжится с её текущего времени, т.е. от значения свойства [time](#) (установка enabled как false не сбрасывает действие).

Примечание: Установка enabled как true не делает автоматического перезапуска анимации. Установка enabled как true сразу же перезапустит анимацию, если выполняется следующее условие: свойство [paused](#) установлено как false и в то же время это действие не было отключено (выполнением команды [stop](#) или [reset](#)), а также ни [weight](#), ни [timeScale](#) не равны 0.

[.loop](#)

Циклический режим (может быть изменен с помощью метода [setLoop](#)). Значением по умолчанию является константа [THREE.LoopRepeat](#) (с бесконечным числом повторений - свойство [repetitions](#))

Значение должно быть одной из следующих констант:

[THREE.LoopOnce](#) - однократное воспроизведение клипа,
[THREE.LoopRepeat](#) - проигрывает клип выбранное количество раз (указанное в параметре [repetitions](#)), всякий раз

переходя с конца клипа непосредственно на его начало, [THREE.LoopPingPong](#) - воспроизводит клип выбранное количество раз (указанное в параметре `repetitions`), проигрывая его поочередно вперед и назад.

[.paused](#)

Установка значения `paused` как `true` приостанавливает выполнение данного действия, устанавливая шкалу эффективного времени в 0. Значением по умолчанию является `false`.

[.repetitions](#)

Число повторных выполнений [AnimationClip](#) в ходе данного действия. Может быть установлено с помощью метода [setLoop](#). Значением по умолчанию является [Infinity](#).

Установка данного числа не имеет эффекта если [циклический режим](#) установлен как [THREE.LoopOnce](#), т.е. однократное воспроизведение клипа.

Примечание: Первый запуск не учитывается. Например, если `repetition` установлен как 2, общее количество выполнений будет равно 3.

[.time](#)

Локальное (частное) время данного действия (в секундах, начиная с 0).

Значение получается "зажатым" или ограниченным в диапазоне `0...clip.duration` (в соответствии с состоянием цикла). Его можно масштабировать относительно общего времени микшера, изменяя значение свойства [timeScale](#) (применением метода [setEffectiveTimeScale](#) или [setDuration](#)).

[.timeScale](#)

Коэффициент масштабирования для свойства [time](#). Значение 0 приводит к приостановке анимации. Отрицательные значения вызывают воспроизведение анимации в обратном направлении.

Значением по умолчанию является 1.

Свойства/методы связанные со свойством timeScale (и соответственно с time) следующие: [getEffectiveTimeScale](#), [halt](#), [paused](#), [setDuration](#), [setEffectiveTimeScale](#), [stopWarping](#), [syncwith](#), [warp](#).

[.weight](#)

Степень влияния данного действия (в интервале [0, 1]).

Значения между 0 (нет влияния) и 1 (полноценное воздействие) могут использоваться для смешивания нескольких действий. Значением по умолчанию является 1.

Свойства/методы связанные со свойством weight следующие:

[crossFadeFrom](#), [crossFadeTo](#), [enabled](#), [fadeIn](#), [fadeOut](#), [getEffectiveWeight](#), [setEffectiveWeight](#), [stopFading](#).

[.zeroSlopeAtEnd](#)

Включает гладкую интерполяцию без разделения клипов в начале, в цикле и в конце. Значением по умолчанию является true.

[.zeroSlopeAtStart](#)

Включает гладкую интерполяцию без разделения клипов в начале, в цикле и в конце. Значением по умолчанию является true.

Методы

[cross fade from - перекрестное затухание из'\);](#) " onmouseout="hide()
[fade out action - затухающее действие'\);](#) " onmouseout="hide()">fad

Метод вызывает [постепенное нарастание](#) данного действия и одновременное затухание другого действия, за время заданного (параметр durationInSeconds) интервала времени. Этот метод может быть включен в цепочку.

Если warpBoolean установлен как true, будут применено

дополнительное [деформирование](#) (плавные изменения временных масштабов).

Примечание: Как и у fadeIn/fadeOut, затухание начинается/завершается с весом (т.е. свойством [weight](#)) равным 1.

```
cross fade to - перекрестное затухание в');" onmouseout="hide()">  
fade in action - появляющееся действие');" onmouseout="hide()">fa
```

Метод вызывает [постепенное затухание](#) данного действия и одновременное нарастание другого действия, за время заданного (параметр durationInSeconds) интервала времени. Этот метод может быть включен в цепочку.

Если warpBoolean установлен как true, будут применено дополнительное [деформирование](#) (плавные изменения временных масштабов).

Примечание: Как и у fadeIn/fadeOut, затухание начинается/завершается с весом (т.е. свойством [weight](#)) равным 1.

```
.fadeIn( durationInSeconds )
```

Метод постепенно увеличивает [вес](#) данного действия от 0 до 1, за указанный интервал времени. Этот метод может быть включен в цепочку.

```
fade out - постепенное исчезновение, затухание');" onmouseout="hi
```

Метод постепенно уменьшает [вес](#) данного действия от 0 до 1, за указанный интервал времени. Этот метод может быть включен в цепочку.

```
получить масштаб действующего времени');" onmouseout="hide()">.ge
```

Метод возвращает действующий масштаб времени (с учетом текущих состояний деформации и [приостановки](#).

```
- получить действующий вес');" onmouseout="hide()">.getEffectiveW
```

Метод возвращает действующий вес (с учетом текущих состояний затухания и [возможного включения](#).

`get clip - получить клип';" onmouseout="hide()">.getClip()`

Возвращает клип, содержащий данные анимации для данного действия.

`get mixer - получить микшер';" onmouseout="hide()">.getMixer()`

Возвращает микшер, который ответственен за воспроизведение данного действия.

`get root - получить корень';" onmouseout="hide()">.getRoot()`

Возвращает корневой объект, на котором выполняется данное действие.

`halt - остановка';" onmouseout="hide()">.halt(.durationInSeconds`

Снижает скорость данной анимации до 0, постепенным уменьшением значения [timeScale](#) (начиная с его текущего значения), в течение заданного интервала времени (durationInSeconds). Этот метод может быть включен в цепочку.

`.isRunning()`

Возвращает значение `true`, если отсчет времени действия (свойство [time](#)) в настоящий момент уже запущен.

Кроме того, для выполнения активации в микшере (смотрите [isScheduled](#)) должны быть выполнены следующие условия: свойство [paused](#) должно быть установлено как `false`, [enabled](#) должно быть `true`, значение [timeScale](#) должно быть отличным от 0 и не должно быть запланировано отложенного запуска ([startAt](#)).

Примечание: Значение `true` у `isRunning` не обязательно означает, что анимацию на самом деле будет видно. Это будет только в случае, если у свойства [weight](#) установлено

ненулевое значение.

.isScheduled()

Возвращает значение true, если данное действие активировано в микшере.

Примечание: Это не обязательно значит, что данная анимация на самом деле работает (сравните дополнительные условия для метода [isRunning](#)).

play - играть'); " onmouseout="hide()">.play()

Указывает микшеру активировать действие. Этот метод может быть включен в цепочку.

Примечание: Активация данного действия не обязательно означает, что анимация начнется тотчас же: Activating this action doesn't necessarily mean that the animation starts immediately: If the action had already finished before (by reaching the end of its last loop), or if a time for a delayed start has been set (via [page]:startAt startAt), a [page]:reset reset must be executed first. Some other settings ([page]:paused paused=true, [page]:enabled enabled=false, [page]:weight weight=0, [page]:timeScale timeScale=0) can prevent the animation from playing, too.

reset - сбросить'); " onmouseout="hide()">.reset()

Сбрасывает действие. Этот метод может быть включен в цепочку. Метод устанавливает свойство [paused](#) как false, [enabled](#) как true, [time](#) сбрасывается в 0, прерывает любое запланированное затухание и деформирование, удаляет внутренний счетчик цикла и планирование отложенного запуска.

Примечание: Метод [reset](#) всегда вызывается при вызове метода [stop](#), но собственно сам [reset](#) не вызывает [stop](#). Это означает, если нужно вызывать оба метода, сброса и остановки, не вызывайте [reset](#); вместо него вызывайте [stop](#).

```
.setDuration( durationInSeconds )
```

Устанавливает продолжительность отдельного цикла данного действия (путем настройки timeScale и останавливает любую запланированную деформацию - warp). Этот метод может быть включен в цепочку.

```
set_effective_time_scale - установить действующий масштаб времени'); " onmouseout="hide()">>.setEffective
```

Метод устанавливает timeScale и останавливает любую запланированную деформацию (warp). Этот метод может быть включен в цепочку.

Если paused установлено как `false`, действующий масштаб времени (внутреннее свойство) также будет установлен в это же значение; в противном случае действующий масштаб времени (непосредственно влияющий на анимацию в данный момент) будет установлен в 0. the effective time scale (an internal property) will also be set to this value; otherwise the effective time scale (directly affecting the animation at this moment) will be set to 0.

Примечание: `*paused*` will not be switched to `*true*` automatically, if `*timeScale*` is set to 0 by this method.

```
set_effective_weight - установить действующий вес'); " onmouseout=
```

Устанавливает weight и останавливает любое запланированное затухание. Sets the [page]:weight weight and stops any scheduled fading. Этот метод может быть включен в цепочку.

Если свойство enabled установлено как `true` If [page]:enabled enabled is true, the effective weight (an internal property) will also be set to this value; otherwise the effective weight (directly affecting the animation at this moment) will be set to 0.

Примечание: `*enabled*` will not be switched to `*false*` automatically, if `*weight*` is set to 0 by this method.

```
set_loop - установка цикла'); " onmouseout="hide()">>.setLoop( loop
```

Метод устанавливает режим цикла и количество повторений.

Этот метод может быть включен в цепочку.

```
start at - начинать с');" onmouseout="hide()">>.startAt( startTime
```

Метод определяет время задержки старта анимации (обычно передается как [AnimationMixer.time](#) + [deltaTimeInSeconds](#)).

Этот метод может быть включен в цепочку.

Примечание: The animation will only start at the given time, if [.*startAt*](#) is chained with [page].play play, or if the action has already been activated in the mixer (by a previous call of [.*play*](#), without stopping or resetting it in the meantime).

```
stop - остановить');" onmouseout="hide()">>.stop()
```

Указывает микшеру деактивировать (отключить) данное действие. Этот метод может быть включен в цепочку.

Это действие будет немедленно остановлено и полностью [сброшено](#).

Примечание: Можно остановить все активные действия на том же самом микшере сразу, при помощи метода [mixer.stopAllAction](#).

```
stop fading - остановить затухание');" onmouseout="hide()">>.stopF
```

Метод останавливает любое запланированное затухание ([fadeIn](#)), примененное в данном действии. Этот метод может быть включен в цепочку.

```
stop warping - остановить деформацию');" onmouseout="hide()">>.sto
```

Метод останавливает любую запланированную деформацию ([warp](#)), примененную в данном действии. Этот метод может быть включен в цепочку.

syncWith это сокращ. англ. слов

```
synchronize with - синхронизировать с');" onmouseout="hide()">>.sy
```

```
other action - другое действие');" onmouseout="hide()">>otherActio
```

Метод синхронизирует данное действие с переданным в аргументе otherAction другим действием. Этот метод может быть включен в цепочку.

Синхронизация выполняется установкой значений [time](#) и [timeScale](#) данного действия к соответствующим значениям другого действия (прекращение любой запланированной деформации).

Примечание: Последующие изменения аргументов time и timeScale не будут обнаруживаться.

```
warp - деформация');" onmouseout="hide()">>.warp( startTimeScale,
```

Метод изменяет скорость воспроизведения внутри переданного интервала времени, постепенным изменением значения [timeScale](#) от startTimeScale до endTimeScale. Этот метод может быть включен в цепочку.

События

Есть два события, показывающие когда завершается отдельный цикл действия и, соответственно, все действие целиком. На них можно реагировать с помощью:

```
mixer.addEventListener( 'loop', function( e ) { ... } );
    // properties of e: type, action and loopDelta
    // свойствами для е должны быть тип, действие и loopDelta

mixer.addEventListener( 'finished', function( e ) { ... } );
    // properties of e: type, action and direction
    // свойствами для е должны быть тип, действие и направление
```

Исходники

[AnimationAction.js на github.com](#)

AnimationClip

`AnimationClip` - это многократно используемый набор треков ключевых кадров, представляющих анимацию.

Обзор различных элементов системы анимации `three.js` можно найти в статье "[Системы анимации](#)" раздела "[Следующие шаги](#)" данного руководства.

Конструктор

```
AnimationClip( name, duration, tracks )
```

name - имя данного клипа.

duration - продолжительность данного клипа (в секундах). Если передано отрицательное значение, продолжительность будет рассчитываться из переданного массива `tracks`.

tracks - массив [треков ключевых кадров](#).

Примечание: Вместо создания экземпляра `AnimationClip` непосредственно с помощью конструктора, для создания анимационных клипов можно использовать один из его статических методов: от JSON ([parse](#)), из Instead of instantiating an `AnimationClip` directly with the constructor, you can use one of its static methods to create `AnimationClips`: from JSON ([page:.parse parse]), from morph target sequences ([page:.CreateFromMorphTargetSequence CreateFromMorphTargetSequence], [page:.CreateClipsFromMorphTargetSequences CreateClipsFromMorphTargetSequences]) or from animation hierarchies ([page:.parseAnimation parseAnimation]) - if your model doesn't already hold `AnimationClips` in its geometry's animations array.

Свойства

```
.duration
```

Продолжительность данного клипа (в секундах). Оно может быть вычислено из The duration of this clip (in seconds). This can be calculated from the [page:.tracks tracks] array via [page:.resetDuration

`resetDuration.`

[.name](#)

Имя данного клипа. Конкретный клип может быть найден с помощью метода

[find by name - найти по имени'\);](#) " onmouseout="hide()">[find](#)

[.tracks](#)

Массив, содержащий [KeyframeTrack](#) для каждого свойства, которые анимируются с помощью данного клипа.

[unique identifier - универсальный уникальный идентификатор'\);](#) " onmouseout="hide()">.uuid

[unique identifier - универсальный уникальный идентификатор'\);](#) " onmouseout="hide()">UUID данного экземпляра клипа. Он присваивается автоматически и не должен быть изменен.

Вот [статья Википедии про UUID](#) на русском языке.

Методы

[optimize - оптимизировать'\);](#) " onmouseout="hide()">.optimize()

Оптимизирует каждый трек путем удаления равнозначных последовательных ключей (которые являются общими в последовательности целей морфинга) Optimizes each track by removing equivalent sequential keys (which are common in morph target sequences).

[.resetDuration\(\)](#)

Метод устанавливает [продолжительность](#) клипа как продолжительность самого длинного [KeyframeTrack](#).

trim - обрезать, подстригать');" onmouseout="hide()">>.trim()

Метод обрезает все треки по времени продолжительности клипа.

Статические методы

создать клипы из последовательности целей морфинга');" onmouseout

Returns an array of new AnimationClips created from the [page:Geometry.morphTargets morph target sequences of a geometry, trying to sort morph target names into animation-group-based patterns like "Walk_001, Walk_002, Run_001, Run_002 ..." This method is called by the [page:JSONLoader internally, and it uses [page:.CreateFromMorphTargetSequence CreateFromMorphTargetSequence.

create from morph target sequence - создать из последовательности целей морфинга');" onmouseout="hide()">>.Cre

Returns a new AnimationClip from the passed [page:Geometry.morphTargets morph targets array of a geometry, taking a name and the number of frames per second.

Примечание: Параметр fps обязательен, но скорость анимации в AnimationAction может быть переопределена методом animationAction.setDuration.

find by name - найти по имени');" onmouseout="hide()">>.findByName

Метод разыскивает AnimationClip по названию, принимая в качестве своего первого параметра либо массив анимационных клипов, либо сетку (Mesh), либо геометрию (Geometry), которые содержат массив, поименованный как "animations".

parse - разбор, синтаксический анализ, парсинг');" onmouseout="hi

Parses a JSON representation of a clip and returns an AnimationClip.

parse animation - разобрать (т.е. сделать анализ) анимацию');" оп

Метод производит разбор формата иерархии анимации и возвращает AnimationClip. Parses the animation.hierarchy format and returns an AnimationClip.

.toJSON(
clip - клип, обрезать');" onmouseout="hide()">clip)

Метод принимает AnimationClip и возвращает JSON-объект.

Исходники

[AnimationClip.js на github.com](#)

AnimationMixer

AnimationMixer - это проигрыватель для анимации конкретного объекта на сцене. Если на сцене независимо анимируются множество объектов, для всех объектов может использоваться один AnimationMixer.

Обзор различных элементов системы анимации **three.js** можно найти в статье "[Системы анимации](#)" раздела "[Следующие шаги](#)" данного руководства.

Конструктор

AnimationMixer(
root object - корневой объект');" onmouseout="hide()">rootObject)

root object - корневой объект');"

onmouseout="hide()">rootObject - объект, анимация которого должна воспроизводиться этим микшером.

Свойства

.time

Общее (глобальное) время микшера (в секундах; начиная с 0 в момент создания микшера).

.timeScale

Коэффициент масштабирования для общего (глобального) времени микшера.

Примечание: Установка параметра `timeScale` микшера в 0 и последующий возврат его к 1 - это возможность приостановки/возобновления (`pause/unpause`) всех действий, управляемых этим микшером.

Методы

clip action - действие клипа');" onmouseout="hide()">.clipAction(
clip - клип');" onmouseout="hide()">clip,
optional root - дополнительный корень');" onmouseout="hide()">opt

Возвращает [AnimationAction](#) для клипа, указанного в параметре `clip`, с дополнительным (необязательным) использованием корневого объекта, отличного от корневого объекта по умолчанию данного микшера. Первым параметром может быть либо объект [AnimationClip](#), либо имя/название `AnimationClip`.

Если действие, связанное с этими параметрами, еще не существует, данным методом оно будет создано.

Примечание: Вызов этого метода несколько раз с одними и теми же параметрами всегда возвращает один и тот же экземпляр клипа.

existing action - существующее действие');" onmouseout="hide()">.
clip - клип');" onmouseout="hide()">clip,
optional root - дополнительный корень');" onmouseout="hide()">opt

Возвращает существующее Returns an existing [AnimationAction](#) for the passed clip, optionally using a root object different from the

mixer's default root.

Первым параметром может быть либо объект [AnimationClip](#), либо имя AnimationClip.

```
get_root - получить корень' );" onmouseout="hide()">.getRoot()
```

Возвращает корневой объект микшера.

```
stop_all_action - остановить все действие' );" onmouseout="hide()">
```

Деактивирует на этом микшере все ранее запланированные действия.

```
update - обновление' );" onmouseout="hide()">.update( deltaTimeInS
```

Advances the global mixer time and updates the animation.

This is usually done in the render loop, passing [page:Clock.getDelta clock.getDelta scaled by the mixer's [page:.timeScale timeScale]).

```
.uncacheClip(  
clip - клип' );" onmouseout="hide()">clip )
```

Высвобождает все ресурсы памяти от клипа.

```
.uncacheRoot(  
root - корень' );" onmouseout="hide()">root )
```

Высвобождает все ресурсы памяти от корневого объекта.

```
.uncacheAction(  
clip - клип' );" onmouseout="hide()">clip,  
optional root - дополнительный корень' );" onmouseout="hide()">opt
```

Высвобождает все ресурсы памяти от действия.

Исходники

[AnimationMixer.js на github.com](#)

AnimationObjectGroup

Группа объектов, которая получает общее состояние анимации. Обзор различных элементов системы анимации ***three.js*** можно найти в статье "[Системы анимации](#)" раздела "[Следующие шаги](#)" данного руководства.

Применение

Add objects you would otherwise pass as 'root' to the constructor or the [page:AnimationMixer.clipAction clipAction method of [page:AnimationMixer AnimationMixer] and instead pass this object as 'root'.

Обратите внимание, что объекты этого класса в микшере представлены как один объект, поэтому управление кэшем отдельных объектов должен выполняться в группе.

Ограничения

Свойства анимации должны быть совместимы среди всех объектов группы.

Отдельным свойством можно управлять либо через целевую группу, либо непосредственно, но нельзя одновременно и тем и другим способами.

Конструктор

```
AnimationObjectGroup( obj1, obj2, obj3, ... )
```

obj - произвольное число сеток (meshes), с одинаковым состоянием анимации.

Свойства

```
.stats
```

Объект, который содержит некоторую информацию о данной

`AnimationObjectGroup` (total number, number in use, number of bindings per object)

`[property:String uuid`

The [link:http://en.wikipedia.org/wiki/Universally_unique_identifier UUID of this *`AnimationObjectGroup`*. It gets automatically assigned and shouldn't be edited.

Методы

`.add(obj1, obj2, obj3, ...)`

Метод добавляет произвольное число объектов в данную `AnimationObjectGroup`.

`.remove(obj1, obj2, obj3, ...)`

Удаляет произвольное число объектов из данной `AnimationObjectGroup`.

`.uncache(obj1, obj2, obj3, ...)`

Высвобождает все ресурсы оперативной памяти для переданных объектов данной `AnimationObjectGroup`.

Исходники

[AnimationObjectGroup.js на github.com](#)

AnimationUtils

Объект с различными функциями для помощи при анимации, используется внутри. An object with various functions to assist with animations, used internally.

Методы

`.arraySlice(array, from, to)`

Это тоже самое, что и [`Array.prototype.slice`](#), но при этом работает на типизированных массивах.

`array` (массив) - это массив, у которого берется часть, т.е. `slice` (часть, кусок),
`from` (от) - точка, с которой начинается `slice`,
`to` (до) - точка, которой заканчивается `slice`.

[`.convertArray\(array, type, forceClone \)`](#)

Конвертирует массив в указанный тип.

[`.flattenJSON\(jsonKeys, times, values, valuePropertyName \)`](#)

Метод используется для разбора форматов ключевых кадров AOS. Used for parsing AOS keyframe formats.

[`.getKeyframeOrder\(times \)`](#)

Метод возвращает массив, по которому могут быть отсортированы времена и значения. Returns an array by which times and values can be sorted.

[`.isTypedArray\(object \)`](#)

Возвращает значение `true`, если объект является типизированным массивом.

[`.sortedArray\(values, stride, order \)`](#)

Сортирует массив, ранее возвращенный методом [`getKeyframeOrder`](#).

Исходники

[AnimationUtils.js на github.com](#)

KeyframeTrack

Трек ключевых кадров (KeyframeTrack) это, согласованная по времени показа, последовательность [`ключевых кадров \(keyframes\)`](#), которые состоят из списков моментов времени и связанных с ними значений, и которые используются для анимации определенного

свойства объекта.

Примечание переводчика: вот перевод начала вышеприведенной статьи о ключевых кадрах.
... [читать далее](#)

Обзор различных элементов системы анимации *three.js* можно найти в статье "[Системы анимации](#)" раздела "[Следующие шаги](#)" данного руководства.

В отличие от иерархии анимации [модели в формате JSON](#), KeyframeTrack не сохраняет свои отдельные ключевые кадры как объекты в массиве keys (ключи) (удерживая совместно моменты времени и значения для каждого кадра в одном месте). In contrast to the animation hierarchy of the [link:<https://github.com/mrdoob/three.js/wiki/JSON-Model-format-3>] JSON model format a *KeyframeTrack* doesn't store its single keyframes as objects in a "keys" array (holding the times and the values for each frame together in one place).

Вместо этого, в KeyframeTrack имеются два массива: массив [времён](#), где хранятся значения времени для всех ключевых кадров данного трека в последовательном порядке, и массив [значений](#), содержащий соответствующие изменяемые значения свойства анимации.

Instead of this there are always two arrays in a *KeyframeTrack*: the [page:.times times array stores the time values for all keyframes of this track in sequential order, and the [page:.values values array contains the corresponding changing values of the animated property.

Отдельное значение, принадлежащее конкретному моменту времени, может быть не только простым числом, но (например) вектором (если анимируется положение объекта) или кватернионом (если анимируется вращение объекта). По этой причине массив значений (который является также плоским массивом) A single value, belonging to a certain point of time, can not only be a simple number, but (for example) a vector (if a position is animated) or a quaternion (if a rotation is animated). For this reason the values array (which is a flat

array, too) might be three or four times as long as the times array.

В соответствии с различными возможными типами анимированных значений существует несколько подклассов KeyframeTrack, наследующих большинство свойств и методов:

- [BooleanKeyframeTrack](#)
- [ColorKeyframeTrack](#)
- [NumberKeyframeTrack](#)
- [QuaternionKeyframeTrack](#)
- [StringKeyframeTrack](#)
- [VectorKeyframeTrack](#)

Некоторые примеры создания вручную [клипов анимации](#) с разными видами ключевых кадров можно найти в [файле](#). Поскольку явные значения определены только для дискретных точек времени, хранящихся в массиве времен, все промежуточные значения между ними должны быть интерполированы. Since explicit values are only specified for the discrete points of time stored in the times array, all values in between have to be interpolated.

Имя трека важно для соединения данного трека с конкретным свойством анимированного узла (сделанного [PropertyBinding](#)). The track's name is important for the connection of this track with a specific property of the animated node (done by [page:PropertyBinding]).

Конструктор

```
KeyframeTrack( name, times, values, interpolation )
```

name - идентификатор для KeyframeTrack, т.е. имя трека ключевых кадров.

times - массив времен ключевых кадров, внутренне преобразованный в [Float32Array](#). Вот [статья Википедии о Float32Array](#) на русском языке.

values - массив значений, связанных с массивом времен, внутренне преобразованный в [Float32Array](#). Вот [статья Википедии о Float32Array](#) на русском языке.

interpolation - тип используемой интерполяции. Возможные

значения смотрите в статье "[Анимационные константы](#)". Значением по умолчанию является [InterpolateLinear](#).

Свойства

.name

The track's name can refer to [page:Geometry.morphTargets morph targets or [page:SkinnedMesh bones or possibly other values within an animated object. See [page:PropertyBinding.parseTrackName for the forms of strings that can be parsed for property binding:

The name can specify the node either using its name or its uuid (although it needs to be in the subtree of the scene graph node passed into the mixer). Or, if the track name starts with a dot, the track applies to the root node that was passed into the mixer.

Usually after the node a property will be specified directly. But you can also specify a subproperty, such as .rotation[x, if you just want to drive the X component of the rotation via a float track.

You can also specify bones or multimaterials by using an object name, for example: .bones[R_hand.scale; the red channel of the diffuse color of the fourth material in a materials array - as a further example - can be accessed with .materials[3.diffuse[r].

PropertyBinding will also resolve morph target names, for example: .morphTargetInfluences[run.

Примечание: Название трека необязательно должно быть уникальным. Несколько треков могут управлять одним и тем же свойством. The track's name does not necessarily have to be unique. Multiple tracks can drive the same property. The result should be based on a weighted blend between the multiple tracks according to the weights of their respective actions.

.times

Тип значения - [Float32Array](#). Вот [статья Википедии](#) на русском языке. преобразован из массива времен, переданного в конструкторе. converted from the times array which is passed in the constructor.

.values

A [link:https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Float32Array Float32Array, converted from the values array which is passed in the constructor.

[property:Constant DefaultInterpolation]

The default interpolation type: [page:Animation InterpolateLinear].

[property:Constant TimeBufferType]

[link:https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Float32Array Float32Array, the type of the buffer internally used for the times.

[property:Constant ValueBufferType]

[link:https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Float32Array Float32Array, the type of the buffer internally used for the values.

Методы

[.createInterpolant\(\)](#)

Метод, в зависимости от значения параметра интерполяции, переданного в конструкторе, создает Creates a [page:LinearInterpolant LinearInterpolant, [page:CubicInterpolant CubicInterpolant or [page:DiscreteInterpolant DiscreteInterpolant, depending on the value of the interpolation parameter passed in the constructor.

[.getInterpolation\(\)](#)

Возвращает тип интерполяции.

[method:Number getValueSize()]

Returns the size of each value (that is the length of the [page:.values values array divided by the length of the [page:.times times array).

[method:DiscreteInterpolant InterpolantFactoryMethodDiscrete([.re](#)

Creates a new [page:DiscreteInterpolant DiscreteInterpolant from the [page:KeyframeTrack.times times and [page:KeyframeTrack.times values. A Float32Array can be passed which will receive the results. Otherwise a new array with the appropriate size will be created

automatically.

```
[method:null InterpolantFactoryMethodLinear( .result )
```

Creates a new [page:LinearInterpolant LinearInterpolant from the [page:KeyframeTrack.times times and [page:KeyframeTrack.times values. A Float32Array can be passed which will receive the results. Otherwise a new array with the appropriate size will be created automatically.

```
[method:null InterpolantFactoryMethodSmooth( .result )
```

Create a new [page:CubicInterpolant CubicInterpolant from the [page:KeyframeTrack.times times and [page:KeyframeTrack.times values. A Float32Array can be passed which will receive the results. Otherwise a new array with the appropriate size will be created automatically.

```
[method:null optimize()
```

Removes equivalent sequential keys, which are common in morph target sequences. Called automatically by the constructor.

```
[method:null scale()
```

Scales all keyframe times by a factor.

Примечание: This is useful, for example, for conversions to a certain rate of frames per seconds (as it is done internally by [page:AnimationClip.CreateFromMorphTargetSequence animationClip.CreateFromMorphTargetSequence).

```
.setInterpolation( interpolationType )
```

Метод устанавливает тип интерполяции. Для выбора смотрите [Анимационные константы](#).

```
.shift( timeOffsetInSeconds )
```

Передвигает все ключевые кадры (keyframes) либо вперед, либо назад во времени.

```
.trim( startTimeInSeconds, endTimeInSeconds )
```

Метод удаляет ключевые кадры перед startTime и после endTime, без изменения каких-либо значений в диапазоне [startTime, endTime].

```
.validate\( \)
```

Performs minimal validation on the tracks. Called automatically by the constructor.

This method logs errors to the console, if a track is empty, if the [page]:.valueSize value size is not valid, if an item in the [page]:.times times or [page]:.values values array is not a valid number or if the items in the *times* array are out of order.

Статические методы

```
[method:KeyframeTrack parse( [page:JSON json )
```

Parses a JSON object and returns a new keyframe track of the correct type.

```
[method:JSON toJSON( [page:KeyframeTrack track )
```

Converts the track to JSON.

Исходники

[KeyframeTrack.js на github.com](#)

[KeyframeTrack →](#)

BooleanKeyframeTrack

Трек логических значений ключевых кадров.

Конструктор

```
BooleanKeyframeTrack( name, times, values )
```

name - (required) identifier for the KeyframeTrack.

times - (required) array of keyframe times.

values - values for the keyframes at the times specified.

Свойства

Для ознакомления с унаследованными свойствами смотрите описание [KeyframeTrack](#).

```
.DefaultInterpolation
```

The default interpolation type to use, [page:AnimationInterpolateDiscrete].

.ValueBufferType

A normal Array (no Float32Array in this case, unlike *ValueBufferType* of [page:KeyframeTrack]).

.ValueTypeName

String 'bool'.

Методы

Для ознакомления с унаследованными методами смотрите описание [KeyframeTrack](#).

.InterpolantFactoryMethodLinear()

The value of this method here is 'undefined', as it does not make sense for discrete properties.

.InterpolantFactoryMethodSmooth()

The value of this method here is 'undefined', as it does not make sense for discrete properties.

Исходники

[BooleanKeyframeTrack.js на github.com](#)

[KeyframeTrack →](#)

ColorKeyframeTrack

Трек значений ключевых кадров, представляющих изменения цвета. Сама по себе основная реализация данного подкласса не имеет ничего особенного. Тем не менее, это место для параметризации цветового пространства.

Конструктор

`ColorKeyframeTrack(name, times, values)`

name - (обязателен) идентификатор KeyframeTrack.
times - (обязателен) массив времен ключевого кадра.
values - значения для ключевых кадров в указанное время.
[page:Constant interpolation - the type of interpolation to use. See
[page:Animation Animation Constants for possible values. Default is
[page:Animation InterpolateLinear.

Свойства

Для ознакомления с унаследованными свойствами смотрите описание [KeyframeTrack](#).

.ValueTypeName

Строка 'color' - цвет.

Методы

Для ознакомления с унаследованными методами смотрите описание [KeyframeTrack](#).

Исходники

[ColorKeyframeTrack.js на github.com](#)

[KeyframeTrack →](#)

NumberKeyframeTrack

Трек числовых значений ключевых кадров.

Конструктор

NumberKeyframeTrack(name, times, values)

[page:String name - (required) identifier for the KeyframeTrack.
[page:Array times - (required) array of keyframe times.
[page:Array values - values for the keyframes at the times specified.
[page:Constant interpolation - the type of interpolation to use. See
[page:Animation Animation Constants for possible values. Default is
[page:Animation InterpolateLinear.

Свойства

Для ознакомления с унаследованными свойствами смотрите описание [KeyframeTrack](#).

[property:String ValueTypeName
String 'number'.

Методы

Для ознакомления с унаследованными методами смотрите описание [KeyframeTrack](#).

Исходники

[NumberKeyframeTrack.js на github.com](#)

[KeyframeTrack →](#)

QuaternionKeyframeTrack

A Track of quaternion keyframe values.

Конструктор

QuaternionKeyframeTrack([name](#), [times](#), [values](#))

[page:String name (required) identifier for the KeyframeTrack.
[page:Array times (required) array of keyframe times.
[page:Array values values for the keyframes at the times specified.
[page:Constant interpolation the type of interpolation to use. See
[page:Animation Animation Constants for possible values. Default is
[page:Animation InterpolateLinear.

Свойства

Для ознакомления с унаследованными свойствами смотрите описание [KeyframeTrack](#).

[property:Constant DefaultInterpolation

The default interpolation type to use, [page:Animation InterpolateLinear].

[property:String ValueTypeName
String 'quaternion'.

Методы

Для ознакомления с унаследованными методами смотрите описание [KeyframeTrack](#).

[method:null InterpolantFactoryMethodLinear() Returns a new [page:QuaternionLinearInterpolant QuaternionLinearInterpolant based on the [page:KeyframeTrack.values values, [page:KeyframeTrack.times times and [page:KeyframeTrack.valueSize valueSize of the keyframes.

Исходники

[QuaternionKeyframeTrack.js на github.com](#)

[KeyframeTrack →](#)

StringKeyframeTrack

Трек строковых значений ключевых кадров.

Конструктор

StringKeyframeTrack(name, times, values)
[page:String name - (required) identifier for the KeyframeTrack.
[page:Array times - (required) array of keyframe times.
[page:Array values - values for the keyframes at the times specified.
[page:Constant interpolation - the type of interpolation to use. See [page:Animation Animation Constants for possible values. Default is [page:Animation InterpolateDiscrete.

Свойства

Для ознакомления с унаследованными свойствами смотрите описание [KeyframeTrack](#).

[property:Constant DefaultInterpolation]

The default interpolation type to use, [page:Animation InterpolateDiscrete].

[property:Array ValueBufferType]

A normal Array (no Float32Array in this case, unlike *ValueBufferType* of [page:KeyframeTrack]).

```
[property:String ValueTypeName
```

```
String 'string'.
```

Методы

Для ознакомления с унаследованными методами смотрите описание [KeyframeTrack](#).

```
[method:null InterpolantFactoryMethodLinear()
```

The value of this method here is 'undefined', as it does not make sense for discrete properties.

```
[method:null InterpolantFactoryMethodSmooth()
```

The value of this method here is 'undefined', as it does not make sense for discrete properties.

Исходники

[StringKeyframeTrack.js на github.com](#)

[KeyframeTrack](#) →

VectorKeyframeTrack

Трек векторных значений ключевых кадров.

Конструктор

```
VectorKeyframeTrack( name, times, values )
```

[page:String name - (required) identifier for the KeyframeTrack.

[page:Array times - (required) array of keyframe times.

[page:Array values - values for the keyframes at the times specified.

[page:Constant interpolation - the type of interpolation to use. See

[page:Animation Animation Constants for possible values. Default is

[page:Animation InterpolateLinear.

Свойства

Для ознакомления с унаследованными свойствамисмотрите описание [KeyframeTrack](#).

```
.ValueTypeName
```

String 'vector'.

Методы

Для ознакомления с унаследованными методамисмотрите описание [KeyframeTrack](#).

Исходники

[VectorKeyframeTrack.js на github.com](#)

PropertyBinding

This holds a reference to a real property in the scene graph; used internally.

Конструктор

```
PropertyBinding( rootNode, path, parsedPath )
```

-- rootNode: -- path -- parsedPath (optional)

Свойства

```
.path
```

[.parsedPath](#)

[.node](#)

[.rootNode](#)

[.BindingType](#)

[.Versioning](#)

[.GetterByBindingType](#)

[.SetterByBindingTypeAndVersioning](#)

Методы

[.getValue\(targetArray, offset \)](#)

[.setValue\(sourceArray, offset \)](#)

[.bind\(\)](#)

Создает пару getter / setter (получатель / установщик) для свойства в Create getter / setter pair for a property in the scene graph. Используется внутри методами [getValue](#) и [setValue](#).

[.unbind\(\)](#)

Отвязывает пару getter / setter (получатель / установщик) для

свойства в Unbind getter / setter pair for a property in the scene graph.

[method:Constructor Composite(targetGroup, path, optionalParsedP
Create a new Composite PropertyBinding.

[method:Constructor create(root, path, parsedPath)
Create a new Composite PropertyBinding (if root is an
[page:AnimationObjectGroup] or PropertyBinding.

[method:Constructor parseTrackName(trackName)
Matches strings in the following forms:
-- nodeName.property
-- nodeName.property[accessor]
-- nodeName.material.property[accessor]
-- uuid.property[accessor]
-- uuid.objectName[objectIndex].propertyName[propertyIndex]
-- parentNode/nodeName.property
-- parentNode/parentNode/nodeName.property[index]
-- .bone[Armature.DEF_cog].position
-- scene:helium_balloon_model:helium_balloon_model.position

[method:Constructor findNode(root, nodeName)
Find a node in a node tree or [page:Skeleton Skeleton.

Исходники

[PropertyBinding.js на github.com](#)

PropertyMixer

Buffered scene graph property that allows weighted accumulation; used internally.

Конструктор

```
PropertyMixer( binding, typeName, valueSize )
```

-- binding
-- typeName
-- valueSize

Свойства

[.binding](#)

[.buffer](#)

Буфер размером Buffer with size

[value size - размер значения'\);](#) " onmouseout="hide()">value

Свойство имеет такую схему:

[[incoming](#) | [accu0](#) | [accu1](#) | [orig](#)]

Интерполяторы могут использовать свойство buffer в качестве своего result и затем данные передавать в incoming.

Interpolators can use .buffer as their .result and the data then goes to 'incoming'.

accu0 и accu1 are used frame-interleaved for the cumulative result and are compared to detect changes.

orig сохраняет исходное состояние свойства.

[.cumulativeWeight](#)

Значение по умолчанию равно 0.

[.valueSize](#)

[.referenceCount](#)

Значение по умолчанию равно 0.

[.useCount](#)

Значение по умолчанию равно 0.

Методы

`.accumulate(accuIndex, weight)`

Accumulate data in [page:PropertyMixer.buffer buffer[accuIndex]
'incoming' region into 'accu[i]'.

Если вес (параметр `weight`) равен 0, то не делается ничего.

`.apply(accuIndex)`

Apply the state of `buffer` 'accu[i]' to the binding when accus differ.

`.saveOriginalState()`

Запоминает состояние связанного свойства и копирует его в оба накопителя (accu).

`.restoreOriginalState()`

Метод применяет к привязке состояние, ранее взятое с помощью метода `saveOriginalState`.

Исходники

[PropertyMixer.js на github.com](#)

ЗВУКОВОЕ СОПРОВОЖДЕНИЕ

Audio

[Object3D →](#)

Создает глобальный, т.е. не имеющий конкретного расположения, аудио объект.

Для его работы используется [Web Audio API](#). Вот [Web Audio API](#) на русском языке.

Пример

```
// Create an AudioListener and add it to the camera
// создаем AudioListener и добавляем его к камере
var listener = new THREE.AudioListener();
camera.add( listener );

// create a global audio source (создаем глобальный аудиоисточник)
var sound = new THREE.Audio( listener );

// Load a sound and set it as the Audio object's buffer
// загружаем звук и устанавливаем его как буфер объекта Audio
var audioLoader = new THREE.AudioLoader();
audioLoader.load( 'sounds/ambient.ogg', function( buffer ) {
    sound.setBuffer( buffer );
    sound.setLoop(true);
    sound.setVolume(0.5);
    sound.play();
});
```

Другие примеры:

[webaudio / sandbox](#)
[webaudio / visualizer](#)

Конструктор

```
Audio(
  listener - слушатель, приемник);" onmouseout="hide()">listener )
```

listener — экземпляр [AudioListener](#) (этот аргумент обязателен).

Свойства

автоматическое воспроизведение');" onmouseout="hide()">.autoplay

Свойство определяет, будет ли звуковое сопровождение запускаться автоматически. Значением по умолчанию является false.

.context

AudioContext для listener, заданного в конструкторе. Вот AudioContext на русском языке.

.filters

Свойство определяет, воспроизводится ли звук в настоящий момент. Значением по умолчанию является пустой массив.

.gain

GainNode созданный при помощи метода AudioContext.createGain().

.hasPlaybackControl

Можно ли управлять воспроизведением при помощи play(), pause() и т.п. методами. Значением по умолчанию является true.

.isPlaying

Свойство показывает, воспроизводится ли звук в данный момент.

.offset

Смещение времени внутри аудиобуфера, с которого должно начаться воспроизведение. Это тоже самое, что и параметр offset из AudioBufferSourceNode.start. Значение по умолчанию равно 0.

.playbackRate

Множитель скорости воспроизведения. Значение по умолчанию

равно 1.

.startTime

Момент, с которого начинается воспроизведение звука. Значение по умолчанию равно 0.

.source

[AudioBufferSourceNode](#) созданный при помощи метода [AudioContext.createBufferSource\(\)](#).

.sourceType

Тип источника звука. Значением по умолчанию является строка '[empty](#)'.

.type

Строковое значение, обозначающее тип установленного 'Audio'.

Методы

.connect()

Подключение к [Audio.source](#). Использование метода внутреннее при инициализации и при установке / удалении фильтров.

.disconnect()

Отключение от [Audio.source](#). Использование метода внутреннее при установке / удалении фильтров.

.getFilter()

Возвращает первый элемент из массива [filters](#).

.getFilters()

Возвращает массив [filters](#).

.getLoop()

Возвращает значение свойства [source.loop](#) (свойство

определяет, зациклено ли воспроизведение).

`.getOutput()`

Возвращает [gainNode](#).

`множитель (скорости) воспроизведения');" onmouseout="hide()">.get`

Возвращает значение [playbackRate](#).

`.getVolume(value)`

Возвращает текущее значение громкости.

`.play()`

Если свойство [hasPlaybackControl](#) установлено как true, то этот метод запускает воспроизведение.

`.pause()`

Если свойство [hasPlaybackControl](#) установлено как true, то этот метод приостанавливает воспроизведение, т.е. переводит в режим "пауза".

`.onEnded()`

Вызывается автоматически по окончании воспроизведения.

Устанавливает свойство [isPlaying](#) как false.

`.setBuffer(audioBuffer)`

Устанавливает [source](#) в audioBuffer, и назначает [sourceType](#) в buffer.

Кроме этого запускает воспроизведение, если свойство [autoplay](#) установлено как true.

`.setFilter(filter)`

Добавляет фильтр в массив [filters](#).

`set filters - установить фильтры');" onmouseout="hide()">.setFilt`

| value - массив из фильтров.
Устанавливает в качестве массива `filters` массив, приведенный в параметре `value`.

`.setLoop(value)`

Устанавливает свойство `source.loop` в состояние, указанное в `value` (будет ли зациклено ли воспроизведение).

`.setNodeSource(audioNode)`

Устанавливает `source` в `audioBuffer`, и назначает `sourceType` в `audioNode`.

Также устанавливает `hasPlaybackControl` как `false`.

`множителя (скорости) воспроизведения');" onmouseout="hide()">.set`

Если `hasPlaybackControl` включен, то свойство `playbackRate` устанавливается как `value`.

`.setVolume(value)`

Устанавливает уровень громкости.

`.stop()`

Если `hasPlaybackControl` включен, то воспроизведение останавливается, `startTime` сбрасывается в 0, а `isPlaying` устанавливается как `false`.

Исходники

[Audio.js на github.com](#)

AudioAnalyser

Создает объект `AudioAnalyser`, использующий для анализа аудиоданных `AnalyserNode`

Для его работы используется [Web Audio API](#). [Web Audio API](#) на

русском языке.

Пример

```
// Create an AudioListener and add it to the camera
// Создаем AudioListener и добавляем его к камере
var listener = new THREE.AudioListener();
camera.add( listener );

// create an Audio source (создаем источник Audio)
var sound = new THREE.Audio( listener );

// Load a sound and set it as the Audio object's buffer
// загружаем звук и устанавливаем его как буфер объекта Audio
var audioLoader = new THREE.AudioLoader();
audioLoader.load( 'sounds/ambient.ogg', function( buffer ) {
    sound.setBuffer( buffer );
    sound.setLoop(true);
    sound.setVolume(0.5);
    sound.play();
});

// Create an AudioAnalyser, passing in the sound and desired fftSize
// создаем AudioAnalyser, с передачей sound и желаемого fftSize
var analyser = new THREE.AudioAnalyser( sound, 32 );

// Get the average frequency of the sound (получаем среднюю частоту
analyser.getAverageFrequency();
```

Другие примеры:

[webaudio / sandbox](#)
[webaudio / visualizer](#)

Конструктор

`AudioAnalyser(audio, fftSize)`

Создает новый [AudioAnalyser](#).

Свойства

[.analyser](#)

Это [AnalyserNode](#), используемый для анализа звука.

Примечание переводчика: вот краткий перевод описания AnalyserNode из
вышеприведенной статьи ... [читать далее](#)

.fftSize

Ненулевое значение степени двойки до 2048, представляющее размер FFT (Fast Fourier Transform) быстрого преобразования Фурье, используемого для определения частотной области.

Подробности смотрите на [этой странице](#).

Примечание переводчика: вот краткий перевод описания fftSize из вышеприведенной статьи ... [читать далее](#)

Посмотрите также [этот материал](#) о спектрах звукового сигнала, быстром преобразовании Фурье, семплах, бинах и т.д.

.data

Используемый для хранения данных анализа, массив 8-разрядных беззнаковых целых чисел, размер которого определяется свойством

[частотных бинов \(bin - лоток, ячейка, подборка\) анализатора'\);](#)" onmouseout="hide()" target="_blank">>analyser.frequencyBinCount.

Примечание переводчика: вот краткий перевод описания analyser.frequencyBinCount из вышеприведенной статьи ... [читать далее](#)

Методы

.getFrequencyData()

Данный метод использует метод Web Audio с названием [getByteFrequencyData](#).

Вот перевод части описания метода по ссылке выше: метод getByteFrequencyData интерфейса AnalyserNode копирует данные текущей частоты в массив беззнаковых целых чисел (Uint8Array), переданный с ним.

Если в массиве элементов меньше, чем в AnalyserNode.frequencyBinCount, избыточные элементы отбрасываются. Если же элементов больше, чем требуется, избыточные элементы игнорируются.

.getAverageFrequency()

Метод получает среднее значение частот, возвращаемых

методом [getFrequencyData](#).

Исходники

[AudioAnalyser.js на github.com](#)

AudioContext

Здесь содержатся методы для настройки [AudioContext](#). Вот [AudioContext](#) на русском языке.

Используется внутри классов [AudioListener](#) и [AudioLoader](#).

Для его работы используется [Web Audio API](#). Вот [Web Audio API](#) на русском языке.

Методы

[`.getContext\(\)`](#)

Возвращает значение переменной context (если оно определено) во внешней области, в противном случае устанавливает в неё новый [AudioContext](#).

[`.setContext\(value \)`](#)

Устанавливает в переменную context во внешней области, значение, указанное в параметре value.

Исходники

[AudioContext.js на github.com](#)

[Object3D →](#)

AudioListener

AudioListener представляет виртуального [слушателя](#) всех звуковых эффектов на сцене, как имеющих определенное положение, так и не имеющих такового.

Приложение three.js обычно создает один экземпляр AudioListener. Это обязательный параметр конструктора для аудио-объектов, таких как [Audio](#) и [PositionalAudio](#).

В большинстве случаев, объект-слушатель является дочерним элементом камеры. Таким образом, изменение положения камеры в трехмерном пространстве представляет и изменение положения слушателя.

Пример

```
// Create an AudioListener and add it to the camera
// создаем AudioListener и добавляем его к камере
var listener = new THREE.AudioListener();
camera.add( listener );

// create a global audio source (создаем глобальный аудиоисточник)
var sound = new THREE.Audio( listener );

// Load a sound and set it as the Audio object's buffer
// загружаем звук и устанавливаем его как буфер объекта Audio
var audioLoader = new THREE.AudioLoader();
audioLoader.load( 'sounds/ambient.ogg', function( buffer ) {
    sound.setBuffer( buffer );
    sound.setLoop(true);
    sound.setVolume(0.5);
    sound.play();
});
```

Другие примеры:

[webaudio / sandbox](#)
[webaudio / timing](#)
[webaudio / visualizer](#)

Конструктор

`AudioListener()`

Создает новый AudioListener.

Свойства

[context - контекст, среда, фон'\);](#) " onmouseout="hide()">.context

Заданный в конструкторе, [AudioContext слушателя](#). Вот [статья о AudioContext](#) на русском языке.

.gain

[GainNode](#) созданный при помощи метода [AudioContext.createGain](#).

.filter

Значением по умолчанию является null.

Методы

.getInput()

Возвращает значение [gainNode](#).

.removeFilter()

Устанавливает свойство [filter](#) как null.

.getFilter()

Возвращает значение свойства [filter](#).

.setFilter([page:AudioNode value])

Устанавливает свойство [filter](#) равным значению, указанному в параметре value.

.getMasterVolume()

Возвращает значение уровня громкости.

.setMasterVolume(value)

Устанавливает значение уровня громкости.

Исходники

[AudioListener.js на github.com](#)

PositionalAudio

Создает позиционированный аудиообъект (т.е. источник звука, имеющий конкретное расположение).

Для его работы используется [Web Audio API](#). Вот [Web Audio API](#) на русском языке.

Пример

```
// Create an AudioListener and add it to the camera
// создаем AudioListener и добавляем его к камере
var listener = new THREE.AudioListener();
camera.add( listener );

// Create the PositionalAudio object (passing in the listener)
// создаем объект PositionalAudio (passing in the listener)
var sound = new THREE.PositionalAudio( listener );

// Load a sound and set it as the PositionalAudio object's buffer
// загружаем звук и устанавливаем его как буфер объекта PositionalA
var audioLoader = new THREE.AudioLoader();
audioLoader.load( 'sounds/song.ogg', function( buffer ) {
    sound.setBuffer( buffer );
    sound.setRefDistance( 20 );
    sound.play();
});

// Create an object for the sound to play from
// создаем объект для воспроизведения звука
var sphere = new THREE.SphereGeometry( 20, 32, 16 );
var material = new THREE.MeshPhongMaterial( { color: 0xff2200 } );
var mesh = new THREE.Mesh( sphere, material );
scene.add( mesh );

// Finally add the sound to the mesh
// наконец добавляем звук к сетке
mesh.add( sound );
```

[Другие примеры:](#)

[webaudio / sandbox](#)
[webaudio / timing](#)

Конструктор

PositionalAudio([listener](#))

| [listener](#) — экземпляр [AudioListener](#) (обязательный аргумент).

Свойства

Для ознакомления с унаследованными свойствамисмотрите описание класса [Audio](#).

[.panner](#)

Это [PannerNode](#) для [PositionalAudio](#).

Примечание переводчика: panner - устройство для панорамирования звука, т.е. для размещения источников звука в виртуальном пространстве, скорости распространения и направленности звукового сигнала. Вот неполный перевод вышеприведенной статьи о PannerNode.

[... читать далее](#)

Методы

Для ознакомления с унаследованными методамисмотрите описание класса [Audio](#).

[.getOutput\(\)](#)

Возвращает [panner](#).

[- получить справочное расстояние'\);](#) " onmouseout="hide()">.getRefD

Возвращает значение параметра [panner.refDistance](#).

Примечание переводчика: Свойство refDistance интерфейса PannerNode это значение типа double, представляющее справочное (эталонное, опорное) расстояние для уменьшения громкости при удалении источника звука от слушателя. Данное значение используется при всех моделях, зависимости от расстояния, т.е. при всех значениях свойства distanceModel.

Значение по умолчанию свойства refDistance равно 1.

[- установить справочное расстояние'\);](#) " onmouseout="hide()">.setRe

Устанавливает значение [panner.refDistance](#).

[коэффициент спада \(уменьшения'\);](#) " onmouseout="hide()">.getRollof

Возвращает значение [panner.rolloffFactor](#).

Примечание переводчика: Свойство `rolloffFactor` интерфейса `PannerNode` это значение типа `double`, характеризующее как быстро уменьшается громкость при удалении источника звука от слушателя. Данное значение используется при всех моделях, зависимости от расстояния, т.е. при всех значениях свойства `distanceModel`.

Значение по умолчанию свойства `rolloffFactor` равно 1.

[коэффициент спада \(уменьшения'\);](#) " onmouseout="hide()">>.setRolloffFactor(

Устанавливает значение [panner.rolloffFactor](#).

[модель \(зависимости от\) расстояния'\);](#) " onmouseout="hide()">>.getDistanceModel()

Возвращает значение [panner.distanceModel](#).

Примечание переводчика: Свойство `distanceModel` интерфейса `PannerNode` является вычисляемым значением, определяющим какой алгоритм будет использоваться для снижения громкости источника звука при его удалении от слушателя.

Возможными значениями являются:

`linear` (линейная): Модель линейной зависимости от расстояния рассчитывает коэффициент усиления, вызванный расстоянием, в соответствии с:

`1 - rolloffFactor * (distance - refDistance) / (maxDistance - refDistance)`

`inverse` (обратная): Модель обратной зависимости от расстояния рассчитывает коэффициент усиления, вызванный расстоянием, в соответствии с:

`refDistance / (refDistance + rolloffFactor * (distance - refDistance))`

`exponential` (экспоненциальная): Модель экспоненциальной зависимости от расстояния рассчитывает коэффициент усиления, вызванный расстоянием, в соответствии с:

`pow(distance / refDistance, -rolloffFactor)`.

Значением по умолчанию для `distanceModel` является `inverse`.

[модель \(зависимости от\) расстояния'\);](#) " onmouseout="hide()">>.setDistanceModel(

Устанавливает значение [panner.distanceModel](#).

[.getMaxDistance\(\)](#)

Возвращает значение [panner.maxDistance](#).

Примечание переводчика: Свойство `maxDistance` интерфейса `PannerNode` это значение типа `double`, представляющее максимальное расстояние между источником звука и слушателем, после которого уровень громкости дальше не

снижается. Данное значение используется только в модели с линейной зависимостью от расстояния, т.е. когда `distanceModel` установлена как `linear`.
Значение по умолчанию свойства `maxDistance` равно 10000.

[.setMaxDistance\(value \)](#)

Устанавливает значение [panner.maxDistance](#).

Исходники

[PositionalAudio.js на github.com](#)

КАМЕРЫ

Camera

[Object3D →](#)

Абстрактный базовый класс для камер. При создании новой камеры этот класс всегда будет унаследован.

Конструктор

`Camera()`

Создает новый объект Camera. Обратите внимание, что этот класс не предназначен для непосредственного вызова; скорее всего вам нужна [камера с перспективной проекцией](#) или [камера с ортографической проекцией](#).

Свойства

Для просмотра общих свойств посмотрите базовый класс [Object3D](#).

`.isCamera`

Используется для проверки, является ли данный объект, или производные от него, камерами. По умолчанию значение равно `true`.

Не нужно его изменять, так как внутри `three.js` оно используется рендерером (визуализатором) для оптимизации.

`.layers`

Слои, частью которых является камера. Это свойство, наследуемое от [Object3D](#).

Чтобы быть видимыми, при отображении области просмотра камеры, объекты должны иметь с камерой как минимум один общий (совместно используемый) слой.

`matrix_world_inverse - инверсия матрицы мира');" onmouseout="hide`

Это инверсия matrixWorld. MatrixWorld содержит Matrix, в которой имеется универсальное преобразование Camera. This is the inverse of matrixWorld. MatrixWorld contains the Matrix which has the world transform of the Camera.

```
projection matrix - матрица проекции');" onmouseout="hide()">>.pro
```

Это матрица, содержащая проекцию изображения.

Методы

Для просмотра общих методов посмотрите базовый класс [Object3D](#).

```
clone - клон, двойник');" onmouseout="hide()">>.clone( )
```

Возвращает новую камеру с теми же свойствами, что и данная.

```
copy - копия');" onmouseout="hide()">>.copy(
```

```
source - источник');" onmouseout="hide()">>source )
```

Копирует свойства исходной (source) камеры в данную камеру.

```
get world direction - получить направление мира');" onmouseout="h  
optional target - дополнительная цель');" onmouseout="hide()">>opt
```

Возвращает трехмерный вектор ([Vector3](#)) представляющий направление в трехмерном пространстве, куда смотрит камера.

Примечание: Это не положительная, а отрицательная часть оси z камеры, в отличие от
get world direction - получить направление мира');"
onmouseout="hide()">>getWorldDirection базового класса (Object3D).

Если задан вектор

```
optional target - дополнительная цель');" onmouseout="hide()">>target
```

результат будет скопирован в этот вектор (который можно также использовать повторно), в противном случае будет создан новый вектор.

Исходники

[Camera.js на Гитхабе](#)

[Object3D →](#)

CubeCamera

Создает 6 камер, которые представляют визуализатор [WebGLRenderTargetCube](#).

Пример

```
// Create cube camera (создаем кубическую камеру)
var cubeCamera = new THREE.CubeCamera( 1, 100000, 128 );
scene.add( cubeCamera );

// Create car (создаем машину)
var chromeMaterial = new THREE.MeshLambertMaterial( { color: 0xffff00 } );
var car = new Mesh( carGeometry, chromeMaterial );
scene.add( car );

// Update the render target cube (обновляем представление заданного)
car.setVisible( false );
cubeCamera.position.copy( car.position );
cubeCamera.updateCubeMap( renderer, scene );

// Render the scene (визуализация сцены)
car.setVisible( true );
renderer.render( scene, camera );
```

Другие примеры:

[materials / cubemap / dynamic](#)
[materials / cubemap / dynamic2](#)
[shading / physical](#)

Конструктор

`CubeCamera(near, far, cubeResolution)`

 | near -- расстояние до ближней плоскости отсечения.

 | far -- расстояние до дальней плоскости отсечения.

 | cubeResolution -- параметр устанавливает длину ребер куба.

Создает CubeCamera, которая содержит 6 [камер с перспективной проекцией](#), каковые затем представляют визуализатор [WebGLRenderTargetCube](#).

Свойства

Для просмотра общих свойств посмотрите базовый класс [Object3D](#).

[.renderTarget](#)

The cube texture that gets generated.

Методы

Для просмотра общих методов посмотрите базовый класс [Object3D](#).

[.update\(renderer, scene \)](#)

[renderer](#) -- текущий WebGL визуализатор (рендерер).

[scene](#) -- текущая (т.е. действующая в данный момент) сцена.

Этот метод вызывается для обновления свойства [renderTarget](#).

[.clear\(renderer, color, depth, stencil \)](#)

Метод вызывается для очистки [renderTarget](#)'овских буферов цвета, глубины и/или трафаретов. Буфер цвета устанавливается для текущего прозрачного цвета рендера. The color buffer is set to the renderer's current clear color. По умолчанию аргументы равны `true`.

Исходники

[CubeCamera.js на Гитхабе](#)

[Object3D](#) → [Camera](#) →

Ортографическая камера

Камера с [ортографической проекцией](#).

При этом способе проецирования, размер объекта в отображаемой картинке остается постоянным, независимо от расстояния между

ним и камерой.

Между прочим, это может быть полезным при отображении двухмерных (2D) сцен и элементов интерфейса пользователя (UI).

Пример

```
var camera = new THREE.OrthographicCamera( width / - 2, width / 2,  
scene.add( camera );
```

Другие примеры:

[camera / orthographic](#)
[camera / orthographic2](#)
[camera](#)
[interactive / cubes / ortho](#)
[materials / cubemap / dynamic](#)
[postprocessing / advanced](#)
[postprocessing / dof2](#)
[postprocessing / godrays](#)
[rtt](#)
[shaders / tonemapping](#)
[shadowmap](#)
[terrain / dynamic](#)

Конструктор

```
OrthographicCamera( left, right, top, bottom, near, far )
```

left — левая плоскость отсечения области видимости камеры.

right — правая плоскость отсечения области видимости камеры.

top — верхняя плоскость отсечения области видимости камеры.

bottom — нижняя плоскость отсечения области видимости камеры.

near — ближняя плоскость отсечения области видимости камеры.

far — дальняя плоскость отсечения области видимости камеры.

Вместе они определяют [область просмотра](#) камеры (в виде усеченной пирамиды). ... [показать](#)

Свойства

Описание общих свойств смотрите в статье о базовом классе [Camera](#).

Обратите внимание, чтобы изменения вступили в силу, после внесения изменений в большинство этих свойств, следует вызвать

метод [updateProjectionMatrix](#).

[.bottom](#)

Нижняя плоскость отсечения области просмотра камеры.

[.far](#)

Дальняя плоскость отсечения области просмотра камеры.

Значение по умолчанию равно 2000.

Допустимый диапазон значений находится между текущим значением ближней плоскости отсечения (свойство [near](#)) и бесконечностью.

[.isOrthographicCamera](#)

Свойство используется для проверки, является ли данный класс и производные от него классы, ортографическими камерами.

Значением по умолчанию является `true`.

Это свойство нельзя изменять, так как оно используется внутри `three.js` визуализатором для оптимизации.

[.left](#)

Левая плоскость отсечения области просмотра камеры.

[.near](#)

Ближняя плоскость отсечения области просмотра камеры.

Значением по умолчанию является `0.1`.

Допустимый диапазон значений находится между `0` и текущим значением дальней плоскости отсечения (свойство [far](#)).

Обратите внимание, что, в отличие от [камеры с перспективной проекцией](#), для ближней плоскости отсечения

`OrthographicCamera` значение `0` является допустимым.

[.right](#)

Правая плоскость отсечения области просмотра камеры.

[.top](#)

Верхняя плоскость отсечения области просмотра камеры.

.view

Значение свойства устанавливается методом [setViewOffset](#). Значением по умолчанию является null.

.zoom

Получает или устанавливает коэффициент масштабирования камеры. Значение по умолчанию равно 1.

Методы

Описание общих методов смотрите в статье о базовом классе [Camera](#).

.setViewOffset(fullWidth, fullHeight, x, y, width, height)

fullWidth — полная ширина при установке нескольких областей просмотра

fullHeight — полная высота при установке нескольких областей просмотра

x — горизонтальное смещение субкамеры

y — вертикальное смещение субкамеры

width — ширина субкамеры

height — высота субкамеры

Метод устанавливает смещение при большой области просмотра. Это полезно для многооконных или многомониторных/многомашинных настройках. Пример подобного использования смотрите в описании [камеры с перспективной проекцией](#).

.clearViewOffset()

Удаляет любые смещения, установленные методом [setViewOffset](#).

.updateProjectionMatrix

Метод обновляет матрицу проецирования. Он должен быть вызван после любого изменения параметров.

```
_ – то есть, в текстовый формат описания объекта,  
основанный на JavaScript');;" onmouseout="hide()">>.toJSON()
```

Метод возвращает данные камеры в формате [JSON](#).

Исходники

[OrthographicCamera.js](#) на Гитхабе

[Object3D](#) → [Camera](#) →

PerspectiveCamera

Камера с [перспективной проекцией](#). Статья Википедии о [перспективе](#) на русском языке.

Этот режим проекции предназначен чтобы наиболее полно сымитировать человеческое зрение. Это самый распространенный режим проецирования, используемый для визуализации (рендеринга) трехмерной (3D) сцены.

Пример

```
var camera = new THREE.PerspectiveCamera( 45, width / height, 1, 10  
scene.add( camera );
```

Другие примеры:

[geometry / birds](#)
[geometry / cube](#)
[animation / skinning / blending](#)
[animation / skinning / morph](#)
[effects / stereo](#)
[interactive / cubes](#)
[loader / collada / skinning](#)

Конструктор

```
PerspectiveCamera( fov, aspect, near, far )
```

fov — вертикальный угол области просмотра камеры.

aspect — соотношение сторон области видимости камеры.

near — ближняя плоскость отсечения области видимости камеры.

far — дальняя плоскость отсечения области видимости камеры.

Вместе они определяют [область просмотра](#) камеры (в виде

усеченной пирамиды). ... показать

Свойства

Описание общих свойств смотрите в статье о базовом классе [Camera](#). Обратите внимание, чтобы изменения вступили в силу, после внесения изменений в большинство этих свойств, следует вызвать метод [updateProjectionMatrix](#).

[.aspect](#)

Соотношение сторон области видимости камеры, т.е. ширина окна (canvas) деленная на высоту окна (canvas). Значение по умолчанию равно 1 (квадратное окно / canvas).

[.far](#)

Дальняя плоскость отсечения области просмотра камеры.

Значение по умолчанию равно 2000.

Допустимый диапазон значений находится между текущим значением ближней плоскости отсечения (свойство [near](#)) и бесконечностью.

[.filmGauge](#)

Размер кинопленки, используемый для большей оси. Значение по умолчанию равно 35 (миллиметров). Этот параметр не оказывает никакого влияния на проекционную матрицу, за исключением случая, когда свойство [filmOffset](#) установлено в ненулевое значение.

[.filmOffset](#)

Смещение от центра по горизонтали в тех же единицах, что и у свойства [filmGauge](#). Значение по умолчанию равно 0.

[.focus](#)

Расстояние до объекта, используемое для стереоскопии и эффектов глубины резкости (depth-of-field). Вот статьи из Википедии на [английском](#) и [русском](#) языках. Этот параметр не

влияет на матрицу проекции, если не используется [стереокамера](#). Значение по умолчанию равно 10.

[.fov](#)

Вертикальный угол области просмотра камеры, снизу доверху, в градусах. Значение по умолчанию равно 50.

[.isPerspectiveCamera](#)

Свойство используется для проверки, является ли данный класс и производные от него классы, камерами с перспективной проекцией. Значением по умолчанию является `true`.

Это свойство нельзя изменять, так как оно используется внутри *three.js* визуализатором для оптимизации.

[.near](#)

Ближняя плоскость отсечения области просмотра камеры.

Значение по умолчанию равно `0.1`.

Допустимый диапазон значений находится между числом, большим 0, и текущим значением дальней плоскости отсечения (свойство [far](#)).

Обратите внимание, в отличие от [камеры с ортографической проекцией](#), значение `0` для ближней плоскости отсечения PerspectiveCamera недопустимо.

[.view](#)

Свойство может быть или описанием области видимости камеры в виде усеченной пирамиды, или `null`. Значение этого свойства устанавливается методом [setViewOffset](#), а удаляется методом [clearViewOffset](#).

[.zoom](#)

Возвращает или устанавливает коэффициент масштабирования камеры.

Методы

Описание общих методов смотрите в статье о базовом классе [Camera](#).

`.clearViewOffset()`

Удаляет любые смещения, установленные при помощи метода [setViewOffset](#).

`получить действующий угол поля просмотра');" onmouseout="hide()">>`

Возвращает текущий вертикальный угол поля просмотра в градусах, с учетом [zoom](#).

`.getFilmHeight()`

Возвращает высоту изображения на кинопленке. Если значение свойства [aspect](#) меньше или равно единице (книжный формат), результат равен [filmGauge](#).

`.getFilmWidth()`

Возвращает ширину изображения на кинопленке. Если значение свойства [aspect](#) больше или равно единице (альбомный формат), результат равен [filmGauge](#).

`.getFocalLength()`

Метод возвращает фокусное расстояние по текущему значению свойства [fov](#) в соответствии со значением свойства [filmGauge](#).

`.setFocalLength(focalLength)`

Метод устанавливает свойство [fov](#) по фокусному расстоянию (заданному в параметре `focalLength`) в соответствии с текущим значением свойства [filmGauge](#).

По умолчанию фокусное расстояние задается для 35-миллиметровой (полнокадровой) камеры.

`.setViewOffset(fullWidth, fullHeight, x, y, width, height)`

`fullWidth` — полная ширина при установке нескольких областей просмотра

`fullHeight` — полная высота при установке нескольких

областей просмотра

x — горизонтальное смещение субкамеры

y — вертикальное смещение субкамеры

width — ширина субкамеры

height — высота субкамеры

Метод устанавливает смещение при большой области просмотра. Это годится для многооконных или многомониторных/многомашинных установках.

Например, если имеются два ряда мониторов друг над другом, по три монитора в каждом ряду (3×2), причем каждый монитор (обозначены буквами A, B, C, D, E и F) с разрешением 1920x1080 пикселей и они располагаются в сетке вот таким образом:

```
+---+---+---+
| A | B | C |
+---+---+---+
| D | E | F |
+---+---+---+
```

то для каждого монитора будем вызывать его следующим образом:

```
var w = 1920;
var h = 1080;
var fullWidth = w * 3;
var fullHeight = h * 2;

// A
camera.setViewOffset( fullWidth, fullHeight, w * 0, h * 0, w, h )
// B
camera.setViewOffset( fullWidth, fullHeight, w * 1, h * 0, w, h )
// C
camera.setViewOffset( fullWidth, fullHeight, w * 2, h * 0, w, h )
// D
camera.setViewOffset( fullWidth, fullHeight, w * 0, h * 1, w, h )
// E
camera.setViewOffset( fullWidth, fullHeight, w * 1, h * 1, w, h )
// F
camera.setViewOffset( fullWidth, fullHeight, w * 2, h * 1, w, h )
```

Обратите внимание, мониторы могут быть разного размера и не располагаться в сетке.

[.updateProjectionMatrix](#)

Метод обновляет матрицу проецирования. Он должен быть

вызван после любого изменения параметров.

```
_ – то есть, в текстовый формат описания объекта,  
основанный на JavaScript');" onmouseout="hide()">>.toJSON()
```

Метод возвращает данные камеры в формате JSON.

Исходники

[PerspectiveCamera.js на Гитхабе](#)

StereoCamera

Сдвоенная [камера с перспективной проекцией](#), используемая для получения эффектов, подобных [3D Anaglyph](#) или [Parallax Barrier](#). Вот эти эффекты на русском языке: [Анаглиф](#) и [Параллаксный барьер](#).

Пример

```
effects / anaglyph  
effects / parallaxbarrier  
effects / stereo
```

Этот класс камер используется внутри файлов

```
examples/js/effects/AnaglyphEffect.js (исходный файл эффекта анаглифа)  
examples/js/effects/ParallaxBarrierEffect.js (исходник параллаксного барьера)  
examples/js/effects/StereoEffect.js (исходник стереоэффекта)
```

используемых, в свою очередь, в вышеупомянутых примерах.

Конструктор

```
StereoCamera( )
```

Свойства

```
.aspect
```

Значение по умолчанию равно 1.

- разделение глаз');;" onmouseout="hide()">>.eyeSep

Значение по умолчанию равно 0.064.

cameraL - сокращ. англ. слов

camera left - левая камера');;" onmouseout="hide()">>.cameraL

Левая камера. This is added to [page:Layers layer 1] - objects to be rendered by the left camera must also be added to this layer.

cameraR - сокращ. англ. слов

camera right - правая камера');;" onmouseout="hide()">>.cameraR

Правая камера. This is added to [page:Layers layer 2] - objects to be rendered by the left camera must also be added to this layer.

Методы

.update(camera)

Обновление стереокамер, основанное на камере, переданной в качестве параметра.

Исходники

[StereoCamera.js на Гитхабе](#)

КОНСТАНТЫ

Анимационные константы

Циклические режимы (loop modes)

[однократное воспроизведение клипа'\);](#) " onmouseout="hide()">THREE.LoopOnce
[клип проигрывается выбранное количество раз \(указанное в параметре repetitions\),](#)
[всякий раз переходя с конца клипа непосредственно на его начало'\);](#) " onmouseout="hide()">THREE.LoopRepeat
[воспроизводится выбранное количество раз \(указанное в параметре repetitions\),](#)
[проигрывая его поочередно вперед и назад'\);](#) " onmouseout="hide()">THREE.LoopPingPong

Режимы интерполяции (interpolation modes)

[THREE.InterpolateDiscrete](#)
[THREE.InterpolateLinear](#)
[THREE.InterpolateSmooth](#)

Режимы окончания (ending modes)

[THREE.ZeroCurvatureEnding](#)
[THREE.ZeroSlopeEnding](#)
[THREE.WrapAroundEnding](#)

Исходники

[constants.js на github.com](#)

Константы ядра

Номер версии (revision number)

[THREE.REV](#)

Номер текущей версии *three.js*.

Кнопки мышки (mouse buttons)

[THREE.MOUSE.LEFT](#)
[THREE.MOUSE.MIDDLE](#)
[THREE.MOUSE.RIGHT](#)

Исходники

[constants.js на github.com](#)

Константы настраиваемых уравнений смешивания

Пример

[materials / blending / custom](#)

Применение

Константы работают со всеми типами материалов. Вначале устанавливается режим смешивания материалов (THREE.CustomBlending), затем устанавливается желаемые уравнение смешивания (Blending Equation), исходный фактор (Source Factor) и целевой фактор (Destination Factor).

```
var material = new THREE.MeshBasicMaterial( {color: 0x00ff00} );
material.blending = THREE.CustomBlending;
material.blendEquation = THREE.AddEquation; // по умолчанию
material.blendSrc = THREE.SrcAlphaFactor; // по умолчанию
material.blendDst = THREE.OneMinusDstAlphaFactor; // по умолчанию
```

Уравнения смешивания (blending equations)

[THREE.AddEquation](#)
[THREE.SubtractEquation](#)
[THREE.ReverseSubtractEquation](#)
[THREE.MinEquation](#)
[THREE.MaxEquation](#)

Исходные факторы (source factors)

[THREE.ZeroFactor](#)
[THREE.OneFactor](#)
[THREE.SrcColorFactor](#)
[THREE.OneMinusSrcColorFactor](#)
[THREE.SrcAlphaFactor](#)
[THREE.OneMinusSrcAlphaFactor](#)
[THREE.DstAlphaFactor](#)
[THREE.OneMinusDstAlphaFactor](#)
[THREE.DstColorFactor](#)
[THREE.OneMinusDstColorFactor](#)
[THREE.SrcAlphaSaturateFactor](#)

Целевые факторы (destination factors)

Все исходные факторы работают и как целевые, за исключением `THREE.SrcAlphaSaturateFactor`.

Исходники

[constants.js на github.com](#)

Константы режима рисования

Это действующие значения для свойства [Mesh.drawMode](#), и управляют тем, как список вершин будет интерпретирован после отправки в графический процессор (GPU).

Обратите внимание, что это все работает только когда `Mesh.geometry` является [BufferGeometry](#). Изменение констант, когда `Mesh.geometry` является [Geometry](#), не окажет никакого влияния.

Режимы рисования (draw modes)

[THREE.TrianglesDrawMode](#)

Этот режим устанавливается по умолчанию и приводит к тому, что каждые три последовательные вершины (v_0, v_1, v_2), (v_2, v_3, v_5), ... будут интерпретированы как отдельный треугольник.

Если число вершин не кратно 3, избыточные вершины

игнорируются.

[THREE.TriangleStripDrawMode](#)

Этот режим приведет к последовательности треугольников, соединенных в ленту, представленную как (v0, v1, v2), (v2, v1, v3), (v2, v3, v4), ... так что каждый предыдущий треугольник имеет две общие вершины с предыдущим треугольником.

[THREE.TriangleFanDrawMode](#)

В результате этого режима появятся последовательности треугольников, в каждом из которых имеется одна общая вершина (подобно лопастям вентилятора - fan), представленные как (v0, v1, v2), (v0, v2, v3), (v0, v3, v4), и т.д.

Примечание: Этот режим не поддерживается [DirectX10](#). (Вот [DirectX10 на русском языке](#).) Браузеры Chrome и Firefox представляют WebGL на Windows при помощи [ANGLE](#), преобразовывая внутри себя этот режим в поддерживаемый, что, скорее всего приведет к снижению производительности этих браузеров.

Примечание переводчика: ANGLE (Almost Native Graphics Layer Engine - практически встроенный движок графического слоя, а перевод самой аббревиатуры ANGLE - угол) это графический движок слоя абстракций, разработанный компанией Google, с открытым исходным кодом под лицензией BSD. Этот API в основном предназначен для обеспечения высокопроизводительной совместимости OpenGL с компьютерами Windows и веб-браузерами вроде Chromium, путем перевода вызовов OpenGL в Direct3D, который гораздо лучше поддерживается драйверами. Для ANGLE существует два внутренних визуализатора: самый старый использует Direct3D 9.0c, в то время как новый использует Direct3D 11.

из статьи англ. Википедии, перевод мой.

Применение

```
var geometry = new THREE.Geometry();

geometry.vertices.push(
    new THREE.Vector3( -10,  10,  0 ),
    new THREE.Vector3( -10, -10,  0 ),
    new THREE.Vector3( 10, -10,  0 ),
    ...
);
geometry.faces.push( new THREE.Face3( 0, 1, 2 ), ... );

var material = new THREE.MeshBasicMaterial( { color: 0xffff00 } );
```

```
var mesh = new THREE.Mesh( geometry, material );
mesh.drawMode = THREE.TrianglesDrawMode; // default (по умолчанию)

scene.add( mesh );
```

Исходники

[constants.js на github.com](#)

Константы материалов

Эти константы определяют свойства, общие для всех типов материала, за исключением операций комбинирования текстур (Texture Combine Operations), которые применяются только к [MeshBasicMaterial](#), [MeshLambertMaterial](#) и [MeshPhongMaterial](#).

Сторона (side)

[THREE.FrontSide](#)
[THREE.BackSide](#)
[THREE.DoubleSide](#)

Эти константы определяют какая сторона грани будет отображаться - передняя (front), задняя (back) или обе. По умолчанию используется [FrontSide](#).

Цвета (colors)

[THREE.NoColors](#)
[THREE.FaceColors](#)
[THREE.VertexColors](#)

[NoColors](#) используется по умолчанию и применяется ко всем граням (поверхностям) цвет материала.

[FaceColors](#) окрашивает грани (т.е. ячейки каркаса) в соответствии со значением цвета [color](#) каждой треугольной грани [Face3](#).

[VertexColors](#) окрашивает грани в соответствии со значением [vertexColors](#) каждой из вершин грани [Face3](#). Это массив из трех значений [color](#), по одному для каждой вершины грани.

Смотрите пример [geometry / colors](#).

Режим смешивания (blending mode)

```
THREE.NoBlending
THREE.NormalBlending
THREE.AdditiveBlending
THREE.SubtractiveBlending
THREE.MultiplyBlending
THREE.CustomBlending
```

Эти константы управляют исходным и целевым уравнениями для RGB и Alpha материала These control the source and destination blending equations for the material's RGB and Alpha sent to the WebGLRenderer for use by WebGL.

[NormalBlending](#) используется по умолчанию.

Обратите внимание, что [CustomBlending](#) должен быть установлен для использования [Custom Blending Equations](#).

Смотрите пример [materials / blending](#).

Режим глубины (depth mode)

```
THREE.NeverDepth
THREE.AlwaysDepth
THREE.LessDepth
THREE.LessEqualDepth
THREE.GreaterEqualDepth
THREE.GreaterDepth
THREE.NotEqualDepth
```

Which depth function the material uses to compare incoming pixels Z-depth against the current Z-depth buffer value. Если результатом этого сравнения будет значение true, пиксель будет прорисован. If the result of the comparison is true, the pixel will be drawn.

[page:Materials NeverDepth will never return true.]

[page:Materials AlwaysDepth will always return true.]

[page:Materials LessDepth will return true if the incoming pixel Z-depth is less than the current buffer Z-depth.]

[page:Materials LessEqualDepth is the default and will return true if the incoming pixel Z-depth is less than or equal to the current buffer Z-depth.]
[page:Materials GreaterEqualDepth will return true if the incoming pixel Z-depth is greater than or equal to the current buffer Z-depth.]

[page:Materials GreaterDepth will return true if the incoming pixel Z-depth is greater than the current buffer Z-depth.]

[page:Materials NotEqualDepth will return true if the incoming pixel Z-

depth is equal to the current buffer Z-depth.

Операции комбинирования (совмещения) текстур

[THREE.MultiplyOperation](#)

[THREE.MixOperation](#)

[THREE.AddOperation](#)

Эти операции определяют каков будет результат при совмещении цвета поверхности с картой цвета среды (если существует), для таких материалов как [MeshBasicMaterial](#), [MeshLambertMaterial](#) и [MeshPhongMaterial](#).

[MultiplyOperation](#) применяется по умолчанию и перемножает карту цвета среды и цвет поверхности.

[MixOperation](#) при смещивании двух цветов используется отражение.

[AddOperation](#) добавляет два цвета.

Исходники

[constants.js на github.com](#)

Константы WebGLRenderer

Режимы выбраковки граней (cull face modes)

[THREE.CullFaceNone](#)

[THREE.CullFaceBack](#)

[THREE.CullFaceFront](#)

[THREE.CullFaceFrontBack](#)

Эти константы используются методом [setFaceCulling](#)

WebGLRenderer'a.

[CullFaceNone](#) отключает выбраковку граней.

[CullFaceBack](#) отбрасываются задние грани (эта константа применяется по умолчанию).

[CullFaceFront](#) отбрасываются передние грани.

[CullFaceFrontBack](#) отбрасываются обе грани, и передняя и задняя.

Направление передней грани Front Face Direction

[THREE.FrontFaceDirectionCW](#)

[THREE.FrontFaceDirectionCCW](#)

Эти константы используются методом [setFaceCulling](#) WebGLRenderer'a.

[FrontFaceDirectionCW](#) sets the winding order for polygons to clockwise.

[FrontFaceDirectionCCW](#) sets the winding order for polygons to counter-clockwise (default).

Типы тени Shadow Types

[THREE.BasicShadowMap](#)

[THREE.PCFShadowMap](#)

[THREE.PCFSOFTShadowMap](#)

Эти константы определяют свойство [shadowMap.type](#)

WebGLRenderer'a.

[BasicShadowMap](#) дает нефильтрованные карты теней - самые быстрые, но и самые низкокачественные. gives unfiltered shadow maps - fastest, but lowest quality.

тестов на затенение в окрестностях искомого фрагмента и для затенения используется среднее значение этих тестов, таким образом, граница тени будет более «мягкой».)"> [PCFShadowMap](#) фильтры теневых карт используют алгоритм Percentage-Closer Filtering - PCF (значение по умолчанию).

реалистичными и приближенными к жизни. Очертания теней растворяются по мере удаления от объекта, отбрасывающего тень. Использование технологии PCSS позволяет сделать очертания теней естественными и избежать «лесенок».)">

[PCFSOFTShadowMap](#) фильтры теневых карт используют алгоритм Percentage-Closer Soft Shadows (PCSS).

Tone Mapping

Отображение тонов цвета (tone mapping) - это процесс преобразования диапазона яркостей HDR к LDR диапазону, отображаемому устройством вывода, например, монитором или принтером, так как вывод HDR изображений на них потребует преобразования динамического диапазона и цветового охвата модели HDR в соответствующий динамический диапазон LDR, чаще всего модель RGB. Ведь диапазон яркости, представленный в HDR, очень широк, это несколько порядков абсолютного динамического

диапазона единовременно, в одной сцене. А диапазон, который можно воспроизвести на привычных устройствах вывода (мониторах, телевизорах), составляет лишь около двух порядков динамического диапазона.

Преобразование из HDR в LDR и называется tone mapping, оно выполняется с потерями и имитирует свойства человеческого зрения.

```
THREE.NoToneMapping
THREE.LinearToneMapping
THREE.ReinhardToneMapping
THREE.Uncharted2ToneMapping
THREE.CineonToneMapping
```

Эти константы определяют свойство [toneMapping](#) WebGLRenderer'a. Они используются для приведения внешнего вида изображения в расширенном динамическом диапазоне (high dynamic range - HDR) к возможному отображению их в низком динамическом диапазоне (low dynamic range - LDR) стандартного компьютерного монитора или экрана мобильного устройства.

[NoToneMapping](#) отключает отображение тонов.

Значением по умолчанию является [LinearToneMapping](#).

Смотрите пример [отображения тонов](#).

Исходники

[constants.js на github.com](#)

Константы текстур

Режимы картирования (Mapping Modes)

```
THREE.UVMapping
THREE.CubeReflectionMapping
THREE.CubeRefractionMapping
THREE.EquirectangularReflectionMapping
THREE.EquirectangularRefractionMapping
THREE.SphericalReflectionMapping
THREE.CubeUVReflectionMapping
THREE.CubeUVRefractionMapping
```

Эти константы определяют режим отображения текстуры. These define the texture's mapping mode.

Значением по умолчанию является [THREE.UVMapping](#), при которой текстура накладывается на поверхность объекта в соответствии с координатами UV сетки (mesh).

Остальные константы определяют типы The rest define environment mapping types.

[page:Constant CubeReflectionMapping] and [page:Constant CubeRefractionMapping] are for use with a [page:CubeTexture CubeTexture], which is made up of six textures, one for each face of the cube.

[page:Constant CubeReflectionMapping] is the default for a [page:CubeTexture CubeTexture].

[page:Constant EquirectangularReflectionMapping] and [page:Constant EquirectangularRefractionMapping] are for use with an equirectangular environment map.

[page:Constant SphericalReflectionMapping] is for use with a spherical reflection map.

Смотрите пример [materials / envmaps](#).

Режимы укладки текстур (Wrapping Modes)

[THREE.RepeatWrapping](#)

[THREE.ClampToEdgeWrapping](#)

[THREE.MirroredRepeatWrapping](#)

Эти константы определяют свойства текстур `wrapS` и `wrapT`, которые устанавливаютстыковку текстур по горизонтали и вертикали.

С константой [RepeatWrapping](#) текстура будет попросту повторяться до бесконечности.

По умолчанию применяется константа [ClampToEdgeWrapping](#).

Последний пиксель текстуры растягивается до края сетки (mesh).

С константой [MirroredRepeatWrapping](#) текстура будет повторяться бесконечно, зеркально отображаясь при каждом повторе.

Фильтры увеличения (Magnification Filters)

THREE.NearestFilter

THREE.LinearFilter

For use with a texture's [page:Texture.magFilter magFilter] property, these define the texture magnification function to be used when the pixel being textured maps to an area less than or equal to one texture element (texel).

[page:constant NearestFilter] returns the value of the texture element that is nearest (in Manhattan distance) to the specified texture coordinates.

[page:constant LinearFilter] is the default and returns the weighted average of the four texture elements that are closest to the specified texture coordinates, and can include items wrapped or repeated from other parts of a texture, depending on the values of [page:Texture.wrapS wrapS] and [page:Texture.wrapT wrapT], and on the exact mapping.

Фильтры минимизации (Minification Filters)

THREE.NearestFilter

THREE.NearestMipMapNearestFilter

THREE.NearestMipMapLinearFilter

THREE.LinearFilter

THREE.LinearMipMapNearestFilter

THREE.LinearMipMapLinearFilter

For use with a texture's [page:Texture.minFilter minFilter] property, these define the texture minifying function that is used whenever the pixel being textured maps to an area greater than one texture element (texel).

In addition to [page:constant NearestFilter] and [page:constant LinearFilter], the following four functions can be used for minification:

[page:constant NearestMipMapNearestFilter] chooses the mipmap that most closely matches the size of the pixel being textured and uses the [page:constant NearestFilter] criterion (the texel nearest to the center of the pixel) to produce a texture value.

[page:constant NearestMipMapLinearFilter] chooses the two mipmaps

that most closely match the size of the pixel being textured and uses the [page:constant NearestFilter] criterion to produce a texture value from each mipmap. The final texture value is a weighted average of those two values.

[page:constant LinearMipMapNearestFilter] chooses the mipmap that most closely matches the size of the pixel being textured and uses the [page:constant LinearFilter] criterion (a weighted average of the four texels that are closest to the center of the pixel) to produce a texture value.

[page:constant LinearMipMapLinearFilter] is the default and chooses the two mipmaps that most closely match the size of the pixel being textured and uses the [page:constant LinearFilter] criterion to produce a texture value from each mipmap. The final texture value is a weighted average of those two values.

See the [example:webgl_materials_texture_filters materials / texture / filters] example.

Типы

```
THREE.UnsignedByteType  
THREE.ByteType  
THREE.ShortType  
THREE.UnsignedShortType  
THREE.IntType  
THREE.UnsignedIntType  
THREE.FloatType  
THREE.HalfFloatType  
THREE.UnsignedShort4444Type  
THREE.UnsignedShort5551Type  
THREE.UnsignedShort565Type  
THREE.UnsignedInt248Type
```

Для использования со свойством текстуры [type](#), которое должно соответствовать правильному формату.

Значением по умолчанию является [UnsignedByteType](#).

Форматы

```
THREE.AlphaFormat  
THREE.RGBFormat
```

```
THREE.RGBAFormat  
THREE.LuminanceFormat  
THREE.LuminanceAlphaFormat  
THREE.RGBEFormat  
THREE.DepthFormat  
THREE.DepthStencilFormat
```

For use with a texture's [page:Texture.format format] property, these define how elements of a 2d texture, or *texels*, are read by shaders.

[page:constant AlphaFormat] discards the red, green and blue components and reads just the alpha component. The [page:Texture.type type] must be [page:constant UnsignedByteType].

[page:constant RGBFormat] discards the alpha components and reads the red, green and blue components. The [page:Texture.type type] must be [page:constant UnsignedByteType] or [page:constant UnsignedShort565Type].

[page:constant RGBAFormat] is the default and reads the red, green, blue and alpha components. The [page:Texture.type type] must be [page:constant UnsignedByteType], [page:constant UnsignedShort4444Type] or [page:constant THREE.UnsignedShort5551Type].

[page:constant LuminanceFormat] reads each element as a single luminance component. This is then converted to a floating point, clamped to the range [0,1], and then assembled into an RGBA element by placing the luminance value in the red, green and blue channels, and attaching 1.0 to the alpha channel. The [page:Texture.type type] must be [page:constant UnsignedByteType].

[page:constant LuminanceAlphaFormat] reads each element as a luminance/alpha double. The same process occurs as for the [page:constant LuminanceFormat], except that the alpha channel may have values other than *1.0*. The [page:Texture.type type] must be [page:constant UnsignedByteType].

[page:constant RGBEFormat] is identical to [page:constant

`RGBAFormat`].

`[page:constant DepthFormat]` reads each element as a single depth value, converts it to floating point, and clamps to the range [0,1]. The `[page:Texture.type type]` must be `[page:constant UnsignedIntType]` or `[page:constant UnsignedShortType]`. This is the default for `[page:DepthTexture DepthTexture]`.

`[page:constant DepthStencilFormat]` reads each element is a pair of depth and stencil values. The depth component of the pair is interpreted as in `[page:constant DepthFormat]`. The stencil component is interpreted based on the depth + stencil internal format. The `[page:Texture.type type]` must be `[page:constant UnsignedInt248Type]`.

Note that the texture must have the correct `[page:Texture.type type]` set, as described above. See [this page](#) for details.

DDS / ST3C Compressed Texture Formats

```
THREE.RGB_S3TC_DXT1_Format  
THREE.RGBA_S3TC_DXT1_Format  
THREE.RGBA_S3TC_DXT3_Format  
THREE.RGBA_S3TC_DXT5_Format
```

For use with a `[page:CompressedTexture CompressedTexture]`'s `[page:Texture.format format]` property, these require support for the [WEBGL_compressed_texture_s3tc](#) extension.

According to [WebglStats](#), as of February 2016 over 80% of WebGL enabled devices support this extension.

There are four [S3TC](#) formats available via this extension. These are:
`[page:constant RGB_S3TC_DXT1_Format]`: A DXT1-compressed image in an RGB image format.

`[page:constant RGBA_S3TC_DXT1_Format]`: A DXT1-compressed image in an RGBA image format with a simple on/off alpha value.

`[page:constant RGBA_S3TC_DXT3_Format]`: A DXT3-compressed image in an RGBA image format. Compared to a 32-bit RGBA texture, it offers 4:1 compression.

`[page:constant RGBA_S3TC_DXT5_Format]`: A DXT5-compressed image in an RGBA image format. It also provides a 4:1 compression, but

differs to the DXT3 compression in how the alpha compression is done.

PVRTC Compressed Texture Formats

```
THREE.RGB_PVRTC_4BPPV1_Format  
THREE.RGB_PVRTC_2BPPV1_Format  
THREE.RGBA_PVRTC_4BPPV1_Format  
THREE.RGBA_PVRTC_2BPPV1_Format
```

For use with a [page:CompressedTexture CompressedTexture]'s [page:Texture.format format] property, these require support for the [WEBGL_compressed_texture_pvrtc](#) extension.

According to [WebglStats](#), as of February 2016 less than 8% of WebGL enabled devices support this extenstion. PVRTC is typically only available on mobile devices with PowerVR chipsets, which are mainly Apple devices.

There are four [PVRTC](#) formats available via this extension. These are:
[page:constant RGB_PVRTC_4BPPV1_Format]: RGB compression in 4-bit mode. One block for each 4?4 pixels.

[page:constant RGB_PVRTC_2BPPV1_Format]: RGB compression in 2-bit mode. One block for each 8?4 pixels.

[page:constant RGBA_PVRTC_4BPPV1_Format]: RGBA compression in 4-bit mode. One block for each 4?4 pixels.

[page:constant RGBA_PVRTC_2BPPV1_Format]: RGBA compression in 2-bit mode. One block for each 8?4 pixels.

ETC Compressed Texture Format

```
THREE.RGB_ETC1_Format
```

For use with a [page:CompressedTexture CompressedTexture]'s [page:Texture.format format] property, these require support for the [WEBGL_compressed_texture_etc1](#) extension.

According to [WebglStats](#), as of February 2016 just over 13% of WebGL enabled devices support this extenstion.

Кодирование Encoding

```
LinearEncoding  
THREE.sRGBEncoding  
THREE.GammaEncoding  
THREE.RGBEEncoding  
THREE.LogLuvEncoding  
THREE.RGBM7Encoding  
THREE.RGBM16Encoding
```

```
THREE.RGBDEncoding  
THREE.BasicDepthPacking  
THREE.RGBADepthPacking
```

Для использования со свойством [encoding](#) Texture.

If the encoding type is changed after the texture has already been used by a material, you will need to set [page:Material.needsUpdate Material.needsUpdate] to *true* to make the material recompile.

Значением по умолчанию является [LinearEncoding](#). Значения, отличные от данных, действительны только для карт материала, envMap и emissiveMap.

Исходники

[constants.js на github.com](#)

ОСНОВНЫЕ ЭЛЕМЕНТЫ THREE.JS

В этом разделе описаны основные классы библиотеки *three.js*.

[BufferAttribute](#)

[BufferGeometry](#)

[Clock](#) - класс для отслеживания времени.

[DirectGeometry](#) - данный класс используется внутри *three.js* для конвертирования из [Geometry](#) в [BufferGeometry](#).

[EventDispatcher](#) - диспетчер событий, класс для управления (отслеживания и запуска) событиями JavaScript.

[Face3](#) - класс, определяющий параметры треугольной грани - ячейки арматуры, из которых составлена геометрическая фигура.

[Geometry](#) - основной класс для всех геометрий кроме BufferGeometry. Также можно использовать напрямую для построения собственных и нестандартных геометрических элементов.

[InstancedBufferAttribute](#)

[InstancedBufferGeometry](#)

[InstancedInterleavedBuffer](#)

[InterleavedBuffer](#)

[InterleavedBufferAttribute](#)

[Layers](#) - слои для управления видимостью трехмерных объектов. Можно использовать до 32 слоёв.

[Object3D](#) - базовый класс для большинства объектов *three.js*, предоставляющий набор свойств и методов для управления объектами в трехмерном пространстве.

[Raycaster](#)

[Uniform](#)

BufferAttribute

Данный класс сохраняет данные атрибута (такие как положения

вершин, индексы граней, нормали, цвета, текстурные координаты (объекта (X, Y, Z) и координатами на текстуре (U, V - эти буквы обозначают оси двумерной текстуры, потому что «X», «Y» и «Z» уже используются для обозначения осей 3D-объекта в пространстве модели). Значения U и V обычно изменяются от 0 до 1.)" onmouseout="hide()">UV) и все атрибуты, установленные пользователем), связанного с BufferGeometry, который позволяет более эффективное прохождение всех этих данных к блок графических вычислений или графический процессор или попросту - процессор видеокарты'" onmouseout="hide()">GPU. Посмотрите эту страницу для дополнительных сведений и примеров использования.

Данные хранятся как векторы произвольной длины (определяемые itemSize), и вообще в методах, описанных ниже, если передаются с индексом, то он автоматически умножается на длину вектора.

Конструктор

BufferAttribute(array, itemSize,

нормированный, т.е. приведенный к какой-то норме'" onmouseout="hid

array -- должен быть TypedArray (тиปизированным массивом).

Используется для создания экземпляра буфера.

В этом массиве должно быть (itemSize * numVertices) элементов, где numVertices - это число вершин в соответствующей BufferGeometry.

itemSize -- число значений массива, связанного с конкретной вершиной. Например, если данный атрибут хранит 3-компонентный вектор (такой как position - положение, normal - нормаль, или color - цвет), тогда itemSize должен быть равен 3.

нормированный, т.е. приведенный к какой-то норме'" onmouseout="hid

-- (дополнительный, необязательный атрибут) - показывает как

базовые данные в буфере сопоставлены со значениями в коде

для программирования шейдеров. Синтаксис языка базируется

программирования ANSI C, однако, из-за его специфической на

из него были исключены многие возможности, для упрощения я:

производительности. В язык включены дополнительные функции
например для работы с векторами и матрицами. Основное преимущество
перед другими шейдерными языками — переносимость кода между
платформами и ОС.)" onmouseout="hide()">GLSL. Например, если массив является экземпляром `UIInt16Array`, а атрибут `normalized` установлен как `true`, то значения от 0 до +65535 в данных массива будут отображены в атрибуте GLSL как значения от `0.0f` до `+1.0f`. Массив `Int16Array` (со знаком) от -32767 до +32767 будет сопоставлен как `-1.0f` - `+1.0f`. Если `normalized` установлен как `false` значения будут конвертированы в числа с плавающей запятой, которые содержат точное значение, т.е. 32767 станет `32767.0f`.

Свойства

.array

Массив удерживает данные, хранящиеся в буфере.

.count

Хранит длину массива деленную на .itemSize.

Если буфер хранит 3-компонентный вектор (как например положение, нормали или цвет), то в этом случае будет подсчитано число таких хранящихся векторов.

.dynamic

Определяет, будет ли буфер динамическим или нет. Значение по умолчанию равно `false`.

При установленном значении `false`, GPU сообщается, что скорее всего, содержимое буфера, будет часто использоваться, но не часто изменяться. Это соответствует флагу `gl.STATIC_DRAW`.

Если же установлено значение `true`, GPU сообщается, что содержимое буфера, вероятно будет как часто использоваться, так и часто изменяться. Это соответствует флагу `gl.DYNAMIC_DRAW`.

Примечание переводчика: перевод части статьи по ссылке выше
`gl.STATIC_DRAW`: содержимое буфера, возможно, будет часто использоваться, но

не часто изменяется. Содержимое записывается в буфер, но не считывается.
gl.DYNAMIC_DRAW: содержимое буфера, возможно, будет часто использоваться и
часто изменяться. Содержимое записывается в буфер, но не считывается.

.isBufferAttribute

Используется для проверки является ли данный класс или производные от него BufferAttribute. Значение по умолчанию равно true.

Не изменяйте это свойство, так как оно используется для оптимизации внутри *three.js*.

.itemSize

Длина векторов, сохраняемых в [array](#).

.name

Дополнительное имя для экземпляра данного атрибута. По умолчанию это пустая строка.

.needsUpdate

Флаг, показывающий что данный атрибут был изменен и должен быть повторно отправлен в графический процессор (GPU).

Установите его как true при изменении значения массива.

Установка этого значения как true также увеличивает [version](#).

.normalized

Указывает каким образом базовые данные в буфере соответствуют значениям в коде GLSL. Подробностисмотрите выше, в [описании конструктора](#).

.onUploadCallback

Функция обратного вызова, выполняемая после того, как визуализатор (renderer) передал данные массива атрибутов в GPU.

.updateRange

Объект содержит:

offset: позиция, с которой начинается обновление, значение по умолчанию равно 0.

count: значение по умолчанию равно -1, что означает не использовать диапазон обновления.

Данное свойство может быть использовано для отдельного обновления некоторых компонентов хранящихся векторов (например, только для компонента относящегося к цвету).

- универсальный уникальный идентификатор'" onmouseout="hide()">>.

- универсальный уникальный идентификатор'"

onmouseout="hide()">UUID данного экземпляра объекта. Он присваивается автоматически и не должен редактироваться.

.version

Номер версии, увеличивающийся каждый раз, когда свойство needsUpdate устанавливается как true.

Методы

.clone()

Возвращает копию данного bufferAttribute.

.copyArray(array)

Копирует, заданный здесь, массив (который может быть обычным массивом или типизированным - TypedArray) в массив array.

Если копируется типованный массив (TypedArray), ознакомьтесь на TypedArray.set с примечаниями по требованиям. Вот TypedArray.set на русском языке.

.copyAt(index1, bufferAttribute, index2)

Копирует вектор из bufferAttribute[index2] в array[index1].

copy colors array - копировать массив (значений) цвета'" onmouse

Копирует массив, представленный значениями цвета в формате RGB, в [array](#).

```
copy indices array - копировать массив индексов'")" onmouseout="hi
```

Копирует массив, представленный индексами [Face3](#), в [array](#).

```
copy vector2s array - копировать массив двумерных векторов'")" onm
```

Копирует массив, представленный векторами [Vector2](#), в [array](#).

```
copy vector3s array - копировать массив трехмерных векторов'")" on
```

Копирует массив, представленный векторами [Vector3](#), в [array](#).

```
copy vector4s array - копировать массив четырехмерных векторов'")"
```

Копирует массив, представленный векторами [Vector4](#), в [array](#).

```
- получить координату по оси X'" onmouseout="hide()">.getX( index )
```

Возвращает компоненту x вектора по заданному индексу.

```
- получить координату по оси Y'" onmouseout="hide()">.getY( index )
```

Возвращает компоненту y вектора по заданному индексу.

```
- получить координату по оси Z'" onmouseout="hide()">.getZ( index )
```

Возвращает компоненту z вектора по заданному индексу.

```
- получить компонент W'" onmouseout="hide()">.getW( index )
```

Возвращает компоненту w вектора по заданному индексу.

```
.onUpload ( callback )
```

Устанавливает значение свойства [onUploadCallback](#).

В примере [WebGL / Buffergeometry](#) этот метод используется для высвобождения памяти после того, как буфер был передан GPU.

`.set(value, offset)`

`value` -- массив или типизированный массив, из которого копируются значения.

`offset` -- индекс массива `array`, с которого начинается копирование, (дополнительный, необязательный аргумент).

Метод вызывает метод `TypedArray.set(value, offset)` на данный `array`.

В частности, посмотрите эту страничку насчет требований к `value` в `TypedArray`. [Вот здесь](#) она на русском языке.

`.setArray(array)`

Устанавливает массив `array` в типизированный массив `TypedArray`, переданный в аргументе.

После установки массива, `needsUpdate` должен быть установлен как `true`.

`.set dynamic - установить (свойство) dynamic'" onmouseout="hide()`

Метод устанавливает значение свойства `dynamic`.

`.setX(index, x)`

Устанавливает компонент x вектора по заданному индексу.

`.setY(index, y)`

Устанавливает компонент y вектора по заданному индексу.

`.setZ(index, z)`

Устанавливает компонент z вектора по заданному индексу.

`.setW(index, w)`

Устанавливает компонент w вектора по заданному индексу.

`.setXY(index, x, y)`

Устанавливает компоненты вектора x, и y по заданному индексу.

```
.setXYZ( index, x, y, z )
```

Устанавливает компоненты вектора x, y, и z по заданному индексу.

```
.setXYZW( index, x, y, z, w )
```

Устанавливает компоненты вектора x, y, z и w по заданному индексу.

Исходники

[BufferAttribute.js на github.com](#)

[BufferAttribute →](#)

Типы BufferAttribute

В *three.js* имеются девять типов BufferAttribute. Они соответствуют [типованным массивам](#) в JavaScript.

```
THREE.Float64BufferAttribute
THREE.Float32BufferAttribute
THREE.Uint32BufferAttribute
THREE.Int32BufferAttribute
THREE.Uint16BufferAttribute
THREE.Int16BufferAttribute
THREE.Uint8ClampedBufferAttribute
THREE.Uint8BufferAttribute
THREE.Int8BufferAttribute
```

Конструктор

Все вышеперечисленное вызывается одним и тем же способом.

```
TypedBufferAttribute( array, itemSize )
```

array -- это может быть типизированный или нетипизированный (обычный) массив. Он будет преобразован в массив указанного типа.

itemSize -- число значений массива, которые должны быть увязаны с конкретной вершиной.

Свойства

Унаследованные свойства смотрите на странице [BufferAttribute](#).

Методы

Унаследованные методы смотрите на странице [BufferAttribute](#).

Исходники

[BufferAttribute.js на github.com](#)

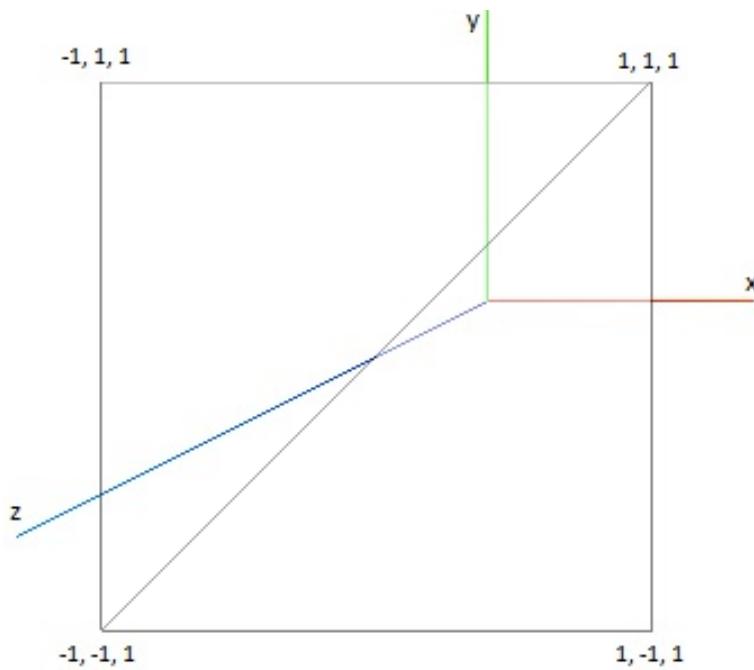
BufferGeometry

Данный класс является эффективной альтернативой [Geometry](#), так как он сохраняет внутри буферов все данные, включая положения вершин, индексы граней, нормали, цвета, текстурные координаты (объекта (X, Y, Z) и координатами на текстуре (U, V - эти буквы обозначают оси двумерной текстуры, потому что «X», «Y» и «Z» уже используются для обозначения осей 3D-объекта в пространстве модели). Значения U и V обычно изменяются от 0 до 1.)" [onmouseout="hide\(\)">UV](#)) и все атрибуты, установленные пользователем); это снижает затраты на прохождение всех этих данных к

блок графических вычислений или графический процессор или попросту - процессор видеокарты)" [onmouseout="hide\(\)">GPU](#).

При этом, работать с BufferGeometry сложнее, чем с [Geometry](#); вместо того, чтобы обращаться к данным расположения как к объектам [Vector3](#), а к данным о цвете как к объектам [Color](#) и так далее, нужно обращаться к необработанным данным (raw data) из соответствующего буфера атрибутов. Это делает BufferGeometry наиболее подходящим для статических объектов, где вам не нужно управлять геометрическими свойствами объекта после его создания.

Пример



```
var geometry = new THREE.BufferGeometry();
// create a simple square shape. We duplicate the bottom left and t
// vertices because each vertex needs to appear once per triangle.
// создадим простую форму - квадрат. Нижняя левая и верхняя правая
// повторяются, поскольку в обоих треугольниках есть каждая из этих
var vertices = new Float32Array( [
-1.0, -1.0, 1.0,
 1.0, -1.0, 1.0,
 1.0, 1.0, 1.0,
 1.0, 1.0, 1.0,
-1.0, 1.0, 1.0,
-1.0, -1.0, 1.0
] );

// itemSize = 3 because there are 3 values (components) per vertex
// itemSize = 3, потому что имеется 3 значения (компоненты, т.е. со
geometry.addAttribute( 'position', new THREE.BufferAttribute( verti
var material = new THREE.MeshBasicMaterial( { color: 0xff0000 } );
var mesh = new THREE.Mesh( geometry, material );
```

Другие примеры:

[Complex mesh with non-indexed faces](#) (сложная сетка с неиндексирован

[Lines](#) (линии),
[Indexed Lines](#) (индексированные линии),
[Particles](#) (частицы), и
[Raw Shaders](#) (необработанные шейдеры).

Обращение к атрибутам Accessing attributes

WebGL хранит данные, связанные с отдельными вершинами геометрии в **атрибутах**. WebGL stores data associated with individual vertices of a geometry in **attributes**. Примеры включают положение вершины, вектор нормали вершины, цвет вершины и т.д. Examples include the position of the vertex, the normal vector for the vertex, the vertex color, and so on. При использовании [Geometry](#), [рендерер](#) берет на себя заботу о переносе этой информации в буферы типизированного массива и отправлении этих данных в шейдер. When using [page:Geometry], the [page:WebGLRenderer renderer] takes care of wrapping up this information into typed array buffers and sending this data to the shader. С BufferGeometry все эти данные хранятся в буферах, связанных с отдельными атрибутами. With BufferGeometry, all of this data is stored in buffers associated with an individual attributes. Это значит, что для получения данных о положении, связанных с вершиной (например), нужно вызвать метод [.getAttribute](#) для доступа к [атрибуту position](#), затем обращаться к отдельным координатам x, y, и z. This means that to get the position data associated with a vertex (for instance), you must call [page:.getAttribute] to access the 'position' [page:BufferAttribute attribute], then access the individual x, y, and z coordinates of the position.

Различными элементами этого класса устанавливаются следующие атрибуты:

[position](#) (itemSize: 3)

Содержит координаты x, y, и z каждой вершины данной геометрии. Устанавливается при помощи метода [.fromGeometry\(\)](#).

[normal](#) (itemSize: 3)

Содержит компоненты x, y и z векторов нормалей к каждой вершине в данной геометрии. Устанавливается при помощи метода [.fromGeometry\(\)](#).

color (itemSize: 3)

Содержит каналы красного, зеленого и синего цвета каждой вершины в данной геометрии. Устанавливается при помощи метода [.fromGeometry\(\)](#).

index (itemSize: 1)

Allows for vertices to be re-used across multiple triangles; this is called using "indexed triangles," and works much the same as it does in [page:Geometry]: each triangle is associated with the index of three vertices. This attribute therefore stores the index of each vertex for each triangular face.

If this attribute is not set, the [page:WebGLRenderer renderer] assumes that each three contiguous positions represent a single triangle.

In addition to the the built-in attributes, you can set your own custom attributes using the addAttribute method. With [page:Geometry], these attributes are set and stored on the [page:Material]. In BufferGeometry, the attributes are stored with the geometry itself. Note that you still need to set the attributes information on the material as well, but the value of each attribute is stored in the BufferGeometry.

Конструктор

BufferGeometry()

Конструктор создает новый BufferGeometry, а также задает некоторым свойствам значение по умолчанию.

Свойства

.attributes

У этой хеш-карты имя атрибута устанавливается как идентификатор, а в качестве значения устанавливается [буфер](#). Вместо непосредственного обращения к данному свойству, для доступа к атрибутам этой геометрии используйте методы [add attribute - добавить атрибут"\)"](#) [onmouseout="hide\(\)">addAttribute](#) и

[get attribute - получить атрибут"\)" onmouseout="hide\(\)">getAttribute](#)

This hashmap has as id the name of the attribute to be set and as value the buffer to set it to. Rather than accessing this property directly, use addAttribute and getAttribute to access attributes of this geometry.

Примечание переводчика: Вот чуть более подробно о хешировании, хеш-таблицах (хеш-картах) и ассоциативных таблицах. ... [читать далее](#)

[bounding_box - ограничительная рамка'" onmouseout="hide\(\)">.boun](#)

Ограничительная рамка для bufferGeometry, которая может быть рассчитана при помощи метода [.computeBoundingBox\(\)](#).

Значением по умолчанию является null.

[bounding_sphere - ограничительная сфера'" onmouseout="hide\(\)">.b](#)

Ограничительная сфера для bufferGeometry, которая может быть рассчитана при помощи метода [.computeBoundingSphere\(\)](#).

Значением по умолчанию является null.

[.drawRange](#)

Используется для определения, какая часть геометрии должна быть визуализирована. Это свойство не устанавливается напрямую, взамен используйте метод [.setDrawRange](#).

Значением по умолчанию является

{ start: 0, count: Infinity }.

[.groups](#)

Разделите геометрию на группы, каждая из которых будет отображаться в отдельном вызове рисования WebGL. Это позволит использовать с bufferGeometry массив материалов. Split the geometry into groups, each of which will be rendered in a separate WebGL draw call. This allows an array of materials to be used with the bufferGeometry.

Каждая группа представляет собой объект в виде:

{ [start: Integer](#), [count: Integer](#), [materialIndex: Integer](#) }

где start указывает индекс первой вершины в этом вызове

рисования, count определяет количество включенных вершин, а materialIndex устанавливает индекс используемого массива материала.

Для добавления групп, вместо непосредственного изменения массива groups, используйте метод [.addGroup](#).

.id

Уникальное число для данного экземпляра bufferGeometry.

.index

Описание этого свойства смотрите выше, в разделе "[Accessing Attributes](#)". Значением по умолчанию является null.

.isBufferGeometry

Используется для проверки, является ли данный класс и производные от него буферными геометриями (BufferGeometry). Значением по умолчанию является true. Это свойство нельзя изменять, поскольку оно используется для оптимизации внутри *three.js*.

.morphAttributes

Хеш-карта [BufferAttribute](#)'ов, содержащая сведения о [morphTargets](#) (целях морфинга) геометрии.

.name

Необязательное (дополнительное) имя для данного экземпляра bufferGeometry. Значением по умолчанию является пустая строка.

- универсальный уникальный идентификатор')" onmouseout="hide()">>.

- универсальный уникальный идентификатор')" onmouseout="hide()">>UUID данного экземпляра объекта. Он присваивается автоматически и не должен изменяться.

Методы

В данном классе возможно применение методов [EventDispatcher](#).

```
.addAttribute( name,  
attribute - атрибут)" onmouseout="hide()">attribute )
```

Метод добавляет атрибут в заданную геометрию. Использование этого метода лучше использования свойства атрибутов, так как для ускорения перебора поддерживается внутренний массив [атрибутов](#).

```
.addGroup( start, count, materialIndex )
```

Добавляет группу в заданную геометрию; подробности смотрите в свойстве [groups](#).

```
.applyMatrix(  
matrix - матрица)" onmouseout="hide()">matrix )
```

Передает преобразование матрицы непосредственно в координаты вершин. Bakes matrix transform directly into vertex coordinates.

```
.center( )
```

Центрирует геометрию исходя из ограничительной рамки.

```
clone - клон, двойник, аналог)" onmouseout="hide()">.clone()
```

Создает клон данной BufferGeometry.

```
copy - копия, копировать)" onmouseout="hide()">.copy( bufferGeom
```

Копирует другую BufferGeometry в данную BufferGeometry.

```
.clearGroups( )
```

Удаляет все группы.

```
.computeBoundingBox( )
```

Вычисляет ограничительную рамку геометрии и обновляет атрибут [.boundingBox](#).

Ограничительные рамки не рассчитываются по умолчанию. Они должны быть вычислены явно, в противном случае они равны null.

```
- вычислить ограничительную сферу'" onmouseout="hide()">.compute
```

Вычисляет ограничительную сферу геометрии и обновляет атрибут [.boundingSphere](#).

Ограничительные сферы не рассчитываются по умолчанию. Они должны быть вычислены явно, в противном случае они равны null.

```
- вычислить нормали вершин'" onmouseout="hide()">.computeVertexN
```

Вычисляет нормали вершин путем усреднения нормалей граней.

```
.dispose\(\)
```

Выводит объект из памяти.

Этот метод вызывается когда во время работы приложения нужно удалить определенную bufferGeometry.

```
от англ. слов from direct geometry - из прямой геометрии'" onmouseout="hide()">
```

Заполняет этот BufferGeometry данными из объекта [DirectGeometry](#).

Примечание: [DirectGeometry](#) используется в основном в качестве посредника для конвертирования между [Geometry](#) и [BufferGeometry](#).

```
от англ. слов from geometry - из геометрии'" onmouseout="hide()">
```

Заполняет этот BufferGeometry данными из объекта [Geometry](#).

```
от англ. слов get attribute - получить атрибут'" onmouseout="hide()">
```

Возвращает атрибут с указанным именем.

от англ. слов get index - получить индекс'" onmouseout="hide()">

Возвращает .index - индекс буфера.

от англ. слов look at - смотреть на'" onmouseout="hide()">.lookA

| vector - A world vector to look at.

Поворачивает геометрию к лицевой точке в пространстве.

Обычно выполняется как одноразовая операция, и не в процессе цикла. Используйте метод Rotates the geometry to face point in space. This is typically done as a one time operation, and not during a loop Use Object3D.lookAt for typical real-time mesh usage.

merge(bufferGeometry, offset)

Объединение с другой BufferGeometry с дополнительным смещением, от которого начинается слияние.

.normalizeNormals()

Каждый вектор нормали геометрического элемента будет иметь единичную величину. Это исправит освещение геометрических поверхностей.

от англ. слов remove attribute - удалить атрибут'" onmouseout="h

Удаляет атрибут с указанным именем.

rotate X - поворот (по оси) X'" onmouseout="hide()">rotateX(rad

Поворот геометрии вокруг оси x. Обычно выполняется как одноразовая операция, и не в процессе цикла.

Для обычного поворота сетки (mesh) в реальном времени используйте свойство Object3D.rotation.

rotate Y - поворот (по оси) Y'" onmouseout="hide()">rotateY(rad

Поворот геометрии вокруг оси Y. Обычно выполняется как одноразовая операция, и не в процессе цикла.

Для обычного поворота сетки (mesh) в реальном времени используйте свойство [Object3D.rotation](#).

```
rotate Z - поворот (по оси) Z'" onmouseout="hide()">rotateZ( rad
```

Поворот геометрии вокруг оси Z. Обычно выполняется как одноразовая операция, и не в процессе цикла.

Для обычного поворота сетки (mesh) в реальном времени используйте свойство [Object3D.rotation](#).

```
scale - масштаб'" onmouseout="hide()">scale( x, y, z )
```

Масштабирует данные геометрии. Обычно выполняется как одноразовая операция, и не в процессе цикла.

Для обычного масштабирования сетки (mesh) в реальном времени используйте свойство [Object3D.scale](#).

```
.setIndex(  
index - индекс'" onmouseout="hide()">index )
```

Устанавливает [.index](#) - индекс буфера.

```
.setDrawRange( start, count )
```

Устанавливает значение свойства буфера [.drawRange](#).

Подробнее смотрите в описании этого свойства.

```
set from object - установить из объекта'" onmouseout="hide()">.s
```

Устанавливает атрибуты из [Object3D](#) для данной BufferGeometry.

```
set from points - установить из точек'" onmouseout="hide()">.set
```

Устанавливает атрибуты данной BufferGeometry из массива точек.

Notation, – то есть, в текстовый формат описания объекта, основанный на JavaScript'" onmouseout="hide()">>.toJSON()

Возвращает необработанное представление объекта BufferGeometry.

to non indexed - в неиндексированном (виде)'" onmouseout="hide()">

Возвращает неиндексированную версию индексированной BufferGeometry.

translate - перевод, перенос, перемещение'" onmouseout="hide()">>

Перевод (смещение) координат геометрии. Обычно выполняется как одноразовая операция, и не в процессе цикла.

Для обычного перевода координат сетки (mesh) в реальном времени используйте свойство Object3D.position.

update from object - обновить из объекта'" onmouseout="hide()">>.

Обновляет для данной BufferGeometry атрибуты из Object3D.

Исходники

[BufferGeometry.js на github.com](#)

Clock

Объект для отслеживания времени. Он использует метод performance.now(), если такая возможность имеется, в противном случае он обращается к менее точному методу Date.now(). Вот описание методов performance.now() и Date.now() на русском языке.

Конструктор

`Clock(autoStart)`

`autoStart` — определяет, будет ли отсчет времени запускаться автоматически, (дополнительный, необязательный аргумент). Значением по умолчанию является `true`.

Свойства

`autoStart`

Если значение свойства установлено как `true`, при первом вызове обновления автоматически запускается отсчет времени. Значением по умолчанию является `true`.

`.startTime`

Здесь хранится время последнего вызова метода `start`. Если отсчет времени запущен, то в этом свойстве хранится время начала отсчета.

`.oldTime`

Это свойство хранит время последнего вызова методов `start`, `getElapsedTime` или `getDelta`.

`.elapsedTime`

Свойство отслеживает полное время работы часов (отсчет всего времени).

`.running`

Данное свойство отслеживает, запущены часы или нет.

Методы

`.start()`

Запускает отсчет времени. Также устанавливает значения свойств `startTime` и `oldTime` в текущее время, обнуляет значение свойства `elapsedTime` и устанавливает значение свойства `running` как `true`.

.stop()

Останавливает отсчет времени и устанавливает значение свойства oldTime на текущее время.

.getElapsedTime()

Метод получает секунды, прошедшие после запуска отсчета времени и устанавливает значение свойства oldTime на текущее время.

Если отсчет времени не запущен, а значение свойства autoStart установлено как true, также запускает отсчет времени.

.getDelta()

Метод получает секунды, прошедшие после установки значения свойства oldTime и устанавливает значение oldTime на текущее время.

Если отсчет времени не запущен, а значение свойства autoStart установлено как true, также запускает отсчет времени.

Исходники

[Clock.js на github.com](#)

DirectGeometry

Данный класс используется внутри **three.js** для конвертирования из Geometry в BufferGeometry.

Конструктор

DirectGeometry()

Создает новую DirectGeometry.

Свойства

.id

Уникальное число для данного экземпляра DirectGeometry.

.name

Дополнительное, необязательное имя. Значением по умолчанию является пустая строка.

.type

Строка 'DirectGeometry'.

.indices

Инициализируется как пустой массив, заполняется методом [.fromGeometry\(\)](#).

.vertices

Инициализируется как пустой массив, заполняется методом [.fromGeometry\(\)](#).

в данной ее точке - прямая, проходящая через эту точку и перпендикулярной касательной прямой (касательной плоскости) в этой точке кривой (плоская гладкая кривая имеет в каждой точке единственную нормаль, расположенную в плоскости кривой.)" onmouseout="hide()">.normals

Инициализируется как пустой массив, заполняется методом [.fromGeometry\(\)](#).

.colors

Инициализируется как пустой массив, заполняется методом [.fromGeometry\(\)](#).

буква s означает, что координаты во множественном числе')" onmouseout="hide()">.s

Инициализируется как пустой массив, заполняется методом [.fromGeometry\(\)](#).

Примечание переводчика: UV это текстурные координаты, соответствующие координатам на поверхности трёхмерного объекта (X, Y, Z). Буквы «U», «V» обозначают оси двумерной текстуры, потому что «X», «Y» и «Z» уже используются для обозначения осей 3D-объекта в пространстве модели. Значения U и V обычно изменяются от 0 до 1.

буква *s* означает, что координаты во множественном числе'")" onmouseout="this.className='';" onmouseover="this.className='over';" style="color: #0000ff; text-decoration: underline;">s

Инициализируется как пустой массив, заполняется методом [.fromGeometry\(\)](#).

.groups

Инициализируется как пустой массив, заполняется методом [.fromGeometry\(\)](#).

- цели морфинга (трансформации)')." onmouseout="hide()">>.morphTarg

Инициализируется как пустой массив, заполняется методом [.fromGeometry\(\)](#).

Примечание переводчика: [морфинг](#) (англ. morphing, трансформация) — технология в компьютерной анимации, визуальный эффект, создающий впечатление плавной трансформации одного объекта в другой. Цели морфинга - это набор последовательных изменений объекта. Исходный объект, зачастую называется затравочным (seed object).

.skinWeights

Инициализируется как пустой массив, заполняется методом `.fromGeometry()`.

Примечание переводчика: морфинг.

.skinIndices

Инициализируется как пустой массив, заполняется методом [.fromGeometry\(\)](#).

от англ. слов bounding box

- ограничительная рамка')" onmouseout="hide()">.boundingBox

Ограничительная рамка для bufferGeometry, которую можно рассчитать с помощью метода [computeBoundingBox](#). Значением по умолчанию является null.

от англ. слов bounding sphere

- ограничительная сфера'" onmouseout="hide()">>.boundingSphere

Ограничительная сфера для bufferGeometry, которую можно рассчитать с помощью метода [computeBoundingSphere](#).

Значением по умолчанию является null.

- вершины, которые нужно обновить'" onmouseout="hide()">>.vertices

Значением по умолчанию является false.

- нормали, которые нужно обновить'" onmouseout="hide()">>.normals

Значением по умолчанию является false.

- цвета, которые нужно обновить'" onmouseout="hide()">>.colorsNeedUpdate

Значением по умолчанию является false.

- текстурные координаты, которые нужно обновить'" onmouseout="hide()">>.texcoordsNeedUpdate

Значением по умолчанию является false.

- группы, которые нужно обновить'" onmouseout="hide()">>.groupsNeedUpdate

Значением по умолчанию является false.

Методы

В данном классе возможно применение методов [EventDispatcher](#).

- вычисление ограничивающей рамки'" onmouseout="hide()">>.computeBoundingBox

Смотрите [Geometry.computeBoundingBox](#).

- вычисление ограничивающей сферы'" onmouseout="hide()">>.computeBoundingSphere

Смотрите [Geometry.computeBoundingSphere](#).

```
.computeGroups( geometry )
```

Вычисляет части геометрии с разными materialIndex. Смотрите [BufferGeometry.groups](#).

```
.dispose( )
```

Удаляет объект из памяти. Этот метод вызывается, если во время работы приложения нужно удалить directGeometry.

```
.fromGeometry( geometry )
```

Переход на экземпляр [Geometry](#) для конвертирования.

Исходники

[DirectGeometry.js на github.com](#)

EventDispatcher

События JavaScript для настраиваемых объектов.
[EventDispatcher на Гитхабе](#).

Пример

```
// Adding events to a custom object
// Добавляем события в пользовательский объект
var Car = function () {
    this.start = function () {
        this.dispatchEvent( { type: 'start', message: 'vroom vroom!' } );
    };
};

// Mixin the EventDispatcher.prototype with the custom object prototype
// Объединяем EventDispatcher.prototype с прототипом пользовательского объекта
Object.assign( Car.prototype, EventDispatcher.prototype );

// Using events with the custom object
// Использование событий с пользовательским объектом
var car = new Car();
car.addEventListener( 'start', function ( event ) {
    alert( event.message );
}
```

```
    } );  
  
    car.start();
```

Конструктор

[EventDispatcher\(\)](#)

Создает объект EventDispatcher.

Методы

[- добавить приемник событий'" onmouseout="hide\(\)">.addEventListe](#)

type - тип принимаемого события.

listener - функция, вызываемая при запуске события.

Добавляет приемник к типу события.

[- имеется приемник событий'" onmouseout="hide\(\)">hasEventListene](#)

type - тип принимаемого события.

listener - функция, вызываемая при запуске события.

Проверяет, добавлен ли приемник к типу события.

[- удалить приемник событий'" onmouseout="hide\(\)">removeEventList](#)

type - тип приемника удаляемого события.

listener - функция удаляемого приемника.

Удаляет приемник с типом события.

[dispatchEvent\(event \)](#)

event - запускаемое событие.

Запускает тип события.

Исходники

[EventDispatcher.js на github.com](#)

Face3

Треугольная грань, используемая в [Geometry](#). Границы создаются автоматически для всех стандартных типов геометрии, однако, если строится нестандартная геометрия, их придется создавать вручную.

Пример

```
var material = new THREE.MeshStandardMaterial( { color : 0x00cc00 } )

// create a triangular geometry
// создадим геометрию треугольника
var geometry = new THREE.Geometry();
geometry.vertices.push( new THREE.Vector3( -50, -50, 0 ) );
geometry.vertices.push( new THREE.Vector3( 50, -50, 0 ) );
geometry.vertices.push( new THREE.Vector3( 50, 50, 0 ) );

// create a new face using vertices 0, 1, 2
// создаем новую грань, используя вершины 0, 1, 2
var normal = new THREE.Vector3( 0, 1, 0 ); // optional (необязательно)
var color = new THREE.Color( 0xffaa00 ); // optional (необязательно)
var materialIndex = 0; // optional (необязательно)
var face = new THREE.Face3( 0, 1, 2, normal, color, materialIndex )

// add the face to the geometry's faces array
// добавляем грань к массиву граней геометрии
geometry.faces.push( face );

// the face normals and vertex normals can be calculated automatically
// если выше не предоставлено, то нормали граней и вершин могут вычисляться автоматически
geometry.computeFaceNormals();
geometry.computeVertexNormals();

scene.add( new THREE.Mesh( geometry, material ) );
```

Другие примеры:

[ubiquity / test](#)
[svg / sandbox](#)
[WebGL / exporter / obj](#)
[WebGL / shaders / vector](#)

Конструктор

```
Face3(  
    - ячейки сетки-арматуры, из которой состоят все грани геометрическо  
    - ячейки сетки-арматуры, из которой состоят все грани геометрическо  
    - ячейки сетки-арматуры, из которой состоят все грани геометрическо  
    normal - нормаль'" onmouseout="hide()">normal, color, materialIndex
```

а — индекс вершины А.

б — индекс вершины В.

с — индекс вершины С.

normal — нормаль грани или массив нормалей вершин.

color — цвет грани или массив цветов вершин.

materialIndex — это индекс массива материалов, связанного с данной гранью (дополнительный, необязательный аргумент).

Свойства

```
- ячейки сетки-арматуры, из которой состоят все грани геометрическо
```

Индекс вершины А.

```
- ячейки сетки-арматуры, из которой состоят все грани геометрическо
```

Индекс вершины Б.

```
- ячейки сетки-арматуры, из которой состоят все грани геометрическо
```

Индекс вершины С.

```
normal - нормаль'" onmouseout="hide()">.normal
```

Нормаль грани - вектор, представляющий направление данной грани (Face3). При автоматическом вычислении (используется метод [Geometry.computeFaceNormals](#)), это будет нормализованное [векторное произведение](#) двух сторон треугольника. Значением по умолчанию является (0, 0, 0).

color - цвет')" onmouseout="hide()">>.color

Цвет грани - при этом должно использоваться свойство материала [vertexColors](#), установленное как [THREE.FaceColors](#).

.vertexNormals

Массив из 3 [нормалей вершин](#).

.vertexColors

Массив из 3 цветов вершин - при этом должно использоваться свойство материала [vertexColors](#), установленное как [THREE.VertexColors](#).

.materialIndex

Индекс материала (показывает индекс в массиве материалов, связанном с гранью). Значение по умолчанию равно 0.

Методы

clone - клонировать')" onmouseout="hide()">>.clone()

Создает новый клон объекта Face3.

copy - копировать')" onmouseout="hide()">>.copy(face3)

Копирует в эту треугольную грань параметры другой треугольной грани (переданной в аргументе face3).

Исходники

[Face3.js на github.com](#)

Geometry

Основной класс для всех геометрий (но не для [BufferGeometries](#)).

Также его можно использовать напрямую для построения собственных и нестандартных геометрий.

С Geometry работать легче, чем с [BufferGeometries](#), так как она хранит атрибуты, такие как вершины, грани, цвета и тому подобное непосредственно у себя (а не в буферах), хотя она, как правило, работает медленней.

Пример

```
var geometry = new THREE.Geometry();

geometry.vertices.push(
    new THREE.Vector3( -10, 10, 0 ),
    new THREE.Vector3( -10, -10, 0 ),
    new THREE.Vector3( 10, -10, 0 )
);

geometry.faces.push( new THREE.Face3( 0, 1, 2 ) );

geometry.computeBoundingSphere();
```

Другие примеры:

[WebGL / geometry / minecraft](#),
[WebGL / geometry / minecraft / ao](#),
[WebGL / geometry / nurbs](#),
[WebGL / geometry / spline / editor](#),
[WebGL / interactive / cubes / gpu](#),
[WebGL / interactive / lines](#),
[WebGL / interactive / raycasting / points](#),
[WebGL / interactive / voxelpainter](#),
[WebGL / morphNormals](#).

Конструктор

`Geometry()`

Конструктор не принимает аргументов.

Свойства

[.boundingBox](#)

Ограничительная рамка для Geometry, которую можно вычислить

методом [computeBoundingBox](#). Значением по умолчанию является null.

[.boundingSphere](#)

Ограничительная сфера для Geometry, которую можно вычислить методом [computeBoundingSphere](#). Значением по умолчанию является null.

[.colors](#)

Массив значений [цвета](#) вершин, в соответствии с их числом и порядком.

Это свойство используется [точками](#) и [линией](#), а также любыми классами, производными от них, такие как [LineSegments](#) и различные [вспомогательные элементы](#). [Сетки](#) взамен этого свойства используют [Face3.vertexColors](#).

Чтобы задать обновление этого массива, нужно установить свойство [Geometry.colorsNeedUpdate](#) как true.

[.faces](#)

Массив [граней](#).

Массив граней описывает, как для формирования граней соединяется каждая вершина модели. Кроме того, в нем содержится информация о цвете и нормалях граней и вершин. Чтобы задать обновление этого массива, нужно установить свойство [Geometry.elementsNeedUpdate](#) как true.

[.faceVertexUvs](#)

Array of face UV layers, used for mapping textures onto the geometry. Each UV layer is an array of UVs matching the order and number of vertices in faces.

Чтобы задать обновление этого массива, нужно установить свойство [uvsNeedUpdate](#) как true.

[.id](#)

Уникальный номер для данного экземпляра геометрии.

.isGeometry

Используется для проверки, является ли данный класс и его производные геометриями. Значением по умолчанию является true.

Не изменяйте это свойство, так как оно используется для оптимизации внутри **three.js**.

.lineDistances

Массив содержит расстояния между вершинами для геометрий Line. Они нужны для правильной визуализации [LineSegments](#) / [LineDashedMaterial](#). Линейные расстояния могут быть сгенерированы автоматически методом [computeLineDistances](#).

.morphTargets

Массив из целей морфинга. Каждая цель морфинга является объектом Javascript:

```
{ name: "targetName", vertices: [ new THREE.Vector3(), ... ] }
```

Morph vertices match number and order of primary vertices.

.morphNormals

Массив нормалей морфинга. Нормали морфинга имеют структуру подобную целям морфинга, каждый набор нормалей является объектом Javascript:

```
morphNormal = { name: "NormalName", normals: [ new THREE.Vector3(
```

Смотрите пример [WebGL / morphNormals](#).

.name

Имя данной геометрии. Значением по умолчанию является пустая строка ("").

.skinWeights

При работе с [SkinnedMesh](#), каждая вершина может иметь до 4 [костей](#), влияющих на неё. Свойство skinWeights - это массив значений веса, который соответствует порядку вершин данной геометрии. Так например первый skinWeight будет соответствовать первой вершине геометрического элемента.

Поскольку каждая вершина может быть изменена 4 костями, для представления веса оболочки каждой вершины используется [четырехмерный вектор](#).

Значения вектора обычно должны быть в диапазоне между 0 и 1. Например при установке значения равным 0, преобразование кости не будет иметь никакого эффекта. Если установить значение равным 0.5, преобразование будет иметь эффект в 50%. При установке 1, будет влияние на 100%. Если существует только одна кость, связанная с конкретной вершиной, нужно побеспокоиться только о первом компоненте приданного вектора, остальные можно проигнорировать и установить равными 0.

.skinIndices

Точно также как у свойства skinWeights, значения свойства skinIndices соответствуют вершинам геометрического элемента. Каждая вершина может иметь до четырех вершин, связанных с ней. Поэтому, если посмотреть на первую вершину (vertex) и первый индекс оболочки (skinIndex), то они расскажут о kostях, связанный с данной вершиной. Например предположим, первая вершина имеет значение (10.05, 30.10, 12.12), значением первого индекса оболочки может быть (10, 2, 0, 0), а первый вес оболочки имеет значение (0.8, 0.2, 0, 0). In affect this would take the first vertex, and then the bone mesh.bones[10] and apply it 80% of the way. Then it would take the bone skeleton.bones[2] and apply it 20% of the way.

Последующие два значения имеют вес равный 0, так что они не будут оказывать никакого влияния.

Код подобного примера может выглядеть так:

```
// e.g. (например)
geometry.skinIndices[15] = new THREE.Vector4( 0, 5, 9, 0 );
geometry.skinWeights[15] = new THREE.Vector4( 0.2, 0.5, 0.3, 0 );

// corresponds with the following vertex
// соответствует следующей вершине
geometry.vertices[15];

// these bones will be used like so:
```

```
// эти кости будут использоваться вот так:  
skeleton.bones[0]; // weight of 0.2 (вес 0.2)  
skeleton.bones[5]; // weight of 0.5 (вес 0.5)  
skeleton.bones[9]; // weight of 0.3 (вес 0.3)  
skeleton.bones[10]; // weight of 0 (вес 0)  
.uuid
```

[UUID](#) данного экземпляра объекта. Вот статья Википедии о [UUID](#) на русском языке. Он присваивается автоматически и не должен редактироваться.

.vertices

Массив [вершин](#).

Массив вершин содержит положение каждой вершины геометрического элемента. To signal an update in this array, значение свойства [verticesNeedUpdate](#) должно быть установлено как true.

.verticesNeedUpdate

Значение свойства устанавливается как true, если массив вершин был обновлен. Set to *true* if the vertices array has been updated.

.elementsNeedUpdate

Set to *true* if the faces array has been updated.

.uvsNeedUpdate

Значение свойства устанавливается как true, если массив UV координат был обновлен. Set to *true* if the uvs array has been updated.

.normalsNeedUpdate

Значение свойства устанавливается как true, если массив нормалей был обновлен. Set to *true* if the normals array has been updated.

.colorsNeedUpdate

Set to *true* if the colors array or a face3 color has been updated.

.groupsNeedUpdate

Set to *true* if a face3 materialIndex has been updated.

.lineDistancesNeedUpdate

Значение свойства устанавливается как true, если массив linedistances был обновлен. Set to *true* if the linedistances array has been updated.

Методы

В данном классе возможно применение методов [EventDispatcher](#).

.applyMatrix(matrix - матрица)

Метод преобразует матрицу непосредственно в координаты вершин.

.center()

Метод центрирует геометрических элемент относительно ограничительной рамки.

.clone()

Создает новый клон Geometry.

Этот метод копирует только вершины, грани и UV-координаты. Он не копирует все остальные свойства геометрического элемента.

.computeBoundingBox()

Метод рассчитывает ограничительную рамку вокруг геометрического элемента, обновляя атрибут [boundingBox](#).

.computeBoundingSphere()

Метод рассчитывает ограничительную сферу вокруг геометрического элемента, обновляя атрибут [boundingSphere](#).

По умолчанию, ни ограничительные рамки, ни ограничительные сферы не рассчитываются. Они должны рассчитываться явно, в

противном случае значения этих свойств равно null.

.computeFaceNormals()

Метод вычисляет нормали граней.

.computeFlatVertexNormals()

Computes flat vertex normals. Sets the vertex normal of each vertex of each face to be the same as the face's normal.

.computeLineDistances()

Метод вычисляет расстояния между вершинами.

.computeMorphNormals()

Метод вычисляет нормали морфинга.

.computeVertexNormals(areaWeighted)

areaWeighted - если установлено значение true, то вклад каждой нормали грани к нормали вершины определяется по площади грани. Значением по умолчанию является true.

Метод рассчитывает нормали вершин путем усреднения нормалей граней.

copy - копировать'" onmouseout="hide()">.copy(geometry)

Метод копирует вершины, грани и UV-координаты в данную геометрию. Никакие другие свойства не копируются.

.dispose()

Метод удаляет объект из памяти.

Не забывайте вызывать этот метод при удалении геометрического элемента, так как он может привести к утечке памяти.

from buffer geometry - из буферной геометрии'" onmouseout="hide()

Метод конвертирует BufferGeometry в Geometry.

```
.lookAt ( vector )
```

| vector - A world vector to look at.

Rotates the geometry to face point in space. This is typically done as a one time operation but not during the render loop.

Use Object3D.lookAt for typical real-time mesh usage.

```
.merge ( geometry, matrix, materialIndexOffset )
```

Merge two geometries or geometry and geometry from object (using object's transform)

```
.mergeMesh( mesh )
```

Merge the mesh's geometry with this, also applying the mesh's transform.

```
.mergeVertices( )
```

Checks for duplicate vertices using hashmap.

Duplicated vertices are removed and faces' vertices are updated.

```
.normalize( )
```

Метод нормализует геометрический элемент.

Make the geometry centered and have a bounding sphere of radius 1.0.

```
rotate X - повернуть вокруг оси X'" onmouseout="hide()">.rotateX
```

Поворот геометрического элемента вокруг оси X. Обычно он выполняется как однократная операция, но не во время цикла визуализации.

Для стандартного поворота сетки (mesh) в реальном времени используйте метод [rotation](#).

```
rotate X - повернуть вокруг оси Y'" onmouseout="hide()">.rotateY
```

Поворот геометрического элемента вокруг оси Y. Обычно он выполняется как однократная операция, но не во время цикла

визуализации.

Для стандартного поворота сетки (mesh) в реальном времени используйте метод [rotation](#).

```
rotate X - повернуть вокруг оси Z'" onmouseout="hide\(\)">.rotateZ
```

Поворот геометрического элемента вокруг оси Z. Обычно он выполняется как однократная операция, но не во время цикла визуализации.

Для стандартного поворота сетки (mesh) в реальном времени используйте метод [rotation](#).

```
.setFromPoints ( points )
```

Sets the vertices for this Geometry from an array of points.

```
.sortFacesByMaterialIndex ( )
```

Sorts the faces array according to material index. For complex geometries with several materials, this can result in reduced draw calls and improved performance.

```
.scale ( x, y, z )
```

Scale the geometry data. This is typically done as a one time operation but not during the render loop. Use Object3D.scale for typical real-time mesh scaling.

```
.toJSON ( )
```

Convert the geometry to JSON format.

```
.translate ( x, y, z )
```

Translate the geometry. This is typically done as a one time operation but not during the render loop.

Use Object3D.position for typical real-time mesh translation.

Исходники

[Geometry.js на github.com](#)

[BufferAttribute →](#)

InstancedBufferAttribute

В отличие от слова «создание», применяется не к объекту, а к классу. То есть, говорят: (в виртуальной среде) создать экземпляр класса или, другими словами, инстанцировать класс.'") onmouseout="hide()"> версия [BufferAttribute](#) (то есть, это экземпляр класса BufferAttribute). An instanced version of BufferAttribute.

Конструктор

`InstancedBufferAttribute(array, itemSize, meshPerAttribute)`

Свойства

Смотрите наследуемые свойства в описании [BufferAttribute](#).

[.meshPerAttribute](#)

Значение по умолчанию равно 1.

- является инстанцированным буферным атрибутом'") onmouseout="hid

Значение по умолчанию равно true.

Методы

Смотрите наследуемые методы в описании [BufferAttribute](#).

Исходники

[InstancedBufferAttribute.js на github.com](#)

[BufferGeometry →](#)

InstancedBufferGeometry

В отличие от слова «создание», применяется не к объекту, а к классу.
То есть, говорят: (в виртуальной среде) создать экземпляр класса
или, другими словами, инстанцировать класс.'") onmouseout="hide()">
версия [BufferGeometry](#) (то есть, это экземпляр класса
BufferGeometry).

Конструктор

`InstancedBufferGeometry()`

Свойства

Смотрите наследуемые свойства в описании [BufferGeometry](#).

- максимальное число инстанцированных экземпляров'"") onmouseout="

Значение по умолчанию равно `undefined`, т.е. не определено.

- является инстанцированной буферной геометрией'"") onmouseout="hi

Значение по умолчанию равно `true`.

Методы

Смотрите наследуемые методы в описании [BufferGeometry](#).

`.addGroup(start, count, materialIndex)`

Исходники

[InstancedBufferGeometry.js на github.com](#)

[InterleavedBuffer](#) →

InstancedInterleavedBuffer

Инстанцированная версия [InterleavedBuffer](#).

Конструктор

`InstancedInterleavedBuffer(array, itemSize, meshPerAttribute)`

Свойства

Смотрите наследуемые свойства в описании [InterleavedBuffer](#).

[.meshPerAttribute](#)

Значение по умолчанию равно 1.

[.isInstancedInterleavedBuffer](#)

Значение по умолчанию равно true.

Методы

Смотрите наследуемые методы в описании [InterleavedBuffer](#).

Исходники

[InstancedInterleavedBuffer.js на github.com](#)

InterleavedBuffer

Конструктор

`InterleavedBuffer(array, stride)`

`InterleavedBuffer([page:TypedArray array, [page:Integer stride]`

Свойства

`[property:Array array`

`[property:Integer stride`

`[property:Integer count`

Gives the total number of elements in the array.

`[property:Boolean dynamic`

Значение по умолчанию равно false.

`[property:Object updateRange`

Object containing offset and count.

`[property:Number updateRange.offset`

Значение по умолчанию равно 0.

`[property:Number updateRange.count`

Значение по умолчанию равно -1.

[property:Integer version]

A version number, incremented every time the needsUpdate property is set to true.

[property:Integer isInterleavedBuffer]

Значение по умолчанию равно true.

[property:Integer needsUpdate]

Значение по умолчанию равно false. Setting this to true increments [page:InterleavedBuffer.version version].

Методы

[method:null setArray ([page:TypedArray array)

array - must be a Typed Array.

[method:InterleavedBuffer setDynamic] ([page:Boolean value]

Set [page:InterleavedBuffer.dynamic dynamic] to value.

[method:InterleavedBuffer copy](source)

Copy the array, count, stride and value of dynamic to this.

[method:InterleavedBuffer copyAt](index1, attribute, index2)

[method:InterleavedBuffer set](value, offset)

[method:InterleavedBuffer clone](index, x, y)

Исходники

[InterleavedBuffer.js на github.com](#)

InterleavedBufferAttribute

Конструктор

InterleavedBufferAttribute(interleavedBuffer, itemSize, offset, normalized)

[name]([page:InterleavedBuffer interleavedBuffer], [page:Integer itemSize], [page:Integer offset], [page:Boolean normalized])

Свойства

[property:InterleavedBuffer data]

The [page:InterleavedBuffer InterleavedBuffer] instance passed in the constructor.

[property:Integer itemSize]

[property:Integer offset]

[property:Boolean normalized]

Значение по умолчанию равно true.

[property:Boolean isInterleavedBufferAttribute]

Значение по умолчанию равно true.

Методы

[method:Integer count]()

The value of [page:InterleavedBufferAttribute.data data].count.

[method:Array array]()

The value of [page:InterleavedBufferAttribute.data data].array.

[method:null getX](index)

[method:null getY](index)

[method:null getZ](index)

[method:null getW](index)

[method:null setX](index, x)

[method:null setY](index, y)

[method:null setZ](index, z)

[method:null setXY](index, x, y)

[method:null setXYZ](index, x, y, z)

[method:null setXYZW](index, x, y, z, w)

Исходники

[InterleavedBufferAttribute.js на github.com](#)

Layers

Объект предоставляет битовую маску и методы допуска, используемые для управления видимостью [Object3D](#). An object providing a bit mask and accessor methods used to control an Object3D's visibility. A Layers object assigns an Object3D to 0 or more of

32 layers numbered 0 to 31.

This is used to control visibility - an object must share a layer with a camera to be visible when that camera's view is rendered.

All classes that inherit from Object3D have a .layers property which is an instance of this class. An object providing a bit mask and accessor method used to control an [page:Object3D]'s visibility. A [page:Layers] object assigns an [page:Object3D] to 0 or more of 32 layers numbered 0 to 31.

Конструктор

`Layers()`

Создает новый объект Layers, принадлежность которого установлена изначально на слой 0.

Свойства

`.mask`

Битовая маска, в которой хранится к каким из 32 слоев принадлежит этот объект в данный момент.

Методы

`.disable(layer)`

| `layer` - целое число от 0 до 31.

Метод удаляет принадлежность объекта к данному слою (указанному в `layer`).

`.enable(layer)`

| `layer` - целое число от 0 до 31.

Метод добавляет принадлежность объекта к данному слою (указанному в `layer`).

`set(layer)`

| `layer` - целое число от 0 до 31.

Метод устанавливает принадлежность объекта к указанному слою (в аргументе layer) и удаляет принадлежность ко всем другим слоям.

```
.test( layers )
```

layers - a Layers object Returns true if this and the passed layers object are members of the same set of layers.

```
.toggle( layer )
```

| layer - целое число от 0 до 31.

Метод переключает принадлежность к слою.

[method:Boolean test]([page:Integer layers])

| layers - a 32bit bit mask of layer numbers.

Returns true if *layers* and .mask have any bits set in common.

Исходники

[Layers.js на github.com](#)

Object3D

Object3D является базовым классом для большинства объектов *three.js* и предоставляет набор свойств и методов для управления объектами в трехмерном пространстве.

Обратите внимание, что имеется возможность для группировки объектов с помощью метода [.add\(object \)](#), который добавляет объект в качестве дочернего, но, тем не менее, для этого лучше использовать класс [Group](#).

Конструктор

```
object3D()
```

Конструктор не принимает аргументов.

Свойства

.castShadow

Будет ли объект представлен в теневой карте. Значением по умолчанию является false.

children - дети, здесь имеются ввиду дочерние объекты, т.е. объекты, являющиеся производными от объекта-родителя'" опто

Массив дочерних объектов. Для информации о группировке объектов вручную,смотрите [Group](#).

.frustumCulled

When this is set, it checks every frame if the object is in the frustum of the camera before rendering the object. Otherwise the object gets rendered every frame even if it isn't visible. Значением по умолчанию является true.

.id

только для чтения - Уникальное число для данного экземпляра объекта

.isObject

Используется для проверки, является ли этот класс или классы, производные от него, трехмерными объектами (Object3D).
Значением по умолчанию является true.

Это свойство нельзя изменять, так как оно используется для оптимизации внутри *three.js*.

layers - слои, уровни'" onmouseout="hide()">.layers

Принадлежность к какому-либо слою. Объект отображается только в том случае, если имеет хотя бы один слой, используемый совместно с [камерой](#).

.matrix

Матрица локальных преобразований.

`.matrixAutoUpdate`

When this is set, it calculates the matrix of position, (rotation or quaternion) and scale every frame and also recalculates the `matrixWorld` property. Default is `Object3D.DefaultMatrixAutoUpdate` (true).

`.matrixWorld`

The global transform of the object. If the `Object3D` has no parent, then it's identical to the local transform `.matrix`.

`.matrixWorldNeedsUpdate`

When this is set, it calculates the `matrixWorld` in that frame and resets this property to false. Default is false.

`.modelViewMatrix`

This is passed to the shader and used to calculate the position of the object.

`.name`

Optional name of the object (doesn't need to be unique). Default is an empty string.

`.normalMatrix`

This is passed to the shader and used to calculate lighting for the object. It is the transpose of the inverse of the upper left 3x3 sub-matrix of this object's `modelViewMatrix`.

The reason for this special matrix is that simply using the `modelViewMatrix` could result in a non-unit length of normals (on scaling) or in a non-perpendicular direction (on non-uniform scaling). On the other hand the translation part of the `modelViewMatrix` is not relevant for the calculation of normals. Thus a `Matrix3` is sufficient.

`.onAfterRender`

An optional callback that is executed immediately after the `Object3D` is rendered. This function is called with the following parameters: `renderer`, `scene`, `camera`, `geometry`, `material`, `group`.

`.onBeforeRender`

An optional callback that is executed immediately before the Object3D is rendered. This function is called with the following parameters: renderer, scene, camera, geometry, material, group.

`.parent`

Object's parent in the scene graph.

`.position`

A Vector3 representing the object's local position. Default is (0, 0, 0).

`.quaternion`

Object's local rotation as a Quaternion.

`.renderOrder`

This value allows the default rendering order of scene graph objects to be overridden although opaque and transparent objects remain sorted independently. Sorting is from lowest to highest renderOrder.
Значением по умолчанию является 0.

`- универсальный уникальный идентификатор'" onmouseout="hide()">>.`

[link:http://en.wikipedia.org/wiki/Universally_unique_identifier UUID] of this object instance. This gets automatically assigned, so this shouldn't be edited.

`[property:Euler rotation]`

Object's local rotation ([Euler angles](#)), in radians.

`[property:Vector3 scale]`

Object's local scale.

`[property:Vector3 up]`

Up direction. Default is THREE.Vector3(0, 1, 0).

`[property:Boolean visible]`

Object gets rendered if *true*. default – true

`[property:Boolean receiveShadow]`

Material gets baked in shadow receiving.

default – false

[property:Boolean frustumCulled]

When this is set, it checks every frame if the object is in the frustum of the camera. Otherwise the object gets drawn every frame even if it isn't visible.

default – true

[property:Object userData]

An object that can be used to store custom data about the Object3d. It should not hold references to functions as these will not be cloned.

Статические свойства

Static properties and methods are defined per class rather than per instance of that class. This means that changing Object3D.DefaultUp or Object3D.DefaultMatrixAutoUpdate will change the values of up and matrixAutoUpdate for every instance of Object3D (or derived classes) created after the change has been made (already created Object3Ds will not be affected).

.DefaultUp

The default up direction for objects, also used as the default position.

По умолчанию устанавливается как (0, 1, 0).

.DefaultMatrixAutoUpdate

The default setting for matrixAutoUpdate for newly created Object3Ds.

Методы

В данном классе возможно применение методов

[EventDispatcher](#).

.add(object, ...)

Метод добавляет заданный объект в качестве дочернего к этому
Посмотрите описание класса [Group](#) для получения информации (

.applyMatrix(
matrix - матрица)" onmouseout="hide()">matrix)

Применяет матричное преобразование к объекту и обновляет по-

[apply quaternion - применить кватернион'](#)" onmouseout="hide()">

Applies the rotation represented by the quaternion to the object.

[method:null translateX]([page:Float distance])

distance - Distance.

Translates object along x axis by distance.

[method:null translateY]([page:Float distance])

distance - Distance.

Translates object along y axis by distance.

[method:null translateZ]([page:Float distance])

distance - Distance.

Translates object along z axis by distance.

[method:null rotateX]([page:Float rad])

rad - the angle to rotate in radians.

Rotates the object around x axis in local space.

[method:null rotateY]([page:Float rad])

rad - the angle to rotate in radians.

Rotates the object around y axis in local space.

[method:null rotateZ]([page:Float rad])

rad - the angle to rotate in radians.

Rotates the object around z axis in local space.

[method:Vector3 localToWorld]([page:Vector3 vector])

vector - A local vector.

Updates the vector from local space to world space.

[method:Vector3 worldToLocal]([page:Vector3 vector])

vector - A world vector.

Updates the vector from world space to local space.

[method:null lookAt]([page:Vector3 vector])

vector - A world vector to look at.

Rotates object to face point in space.

[method:null remove]([page:Object3D object], ...)

object - An object.

Removes *object* as child of this object. An arbitrary number of objec

```
[method:null traverse]( [page:Function callback] )
```

callback - A function with as first argument an object3D object.

Executes the callback on this object and all descendants.

```
[method:null traverseVisible]( [page:Function callback] )
```

callback - A function with as first argument an object3D object.

Like traverse, but the callback will only be executed for visible objects

```
[method:null traverseAncestors]( [page:Function callback] )
```

callback - A function with as first argument an object3D object.

Executes the callback on all ancestors.

```
[method:null updateMatrix]()
```

Updates local transform.

```
[method:null updateMatrixWorld]( [page:Boolean force] )
```

Updates global transform of the object and its children.

```
[method:Object3D clone]( [page:Boolean recursive] )
```

recursive -- if true, descendants of the object are also cloned. Default is false.

Returns a clone of this object and optionally all descendants.

```
[method:Object3D getObjectByName]( [page:String name] )
```

name -- String to match to the children's Object3d.name property.

Searches through the object's children and returns the first with a matching name.

```
[method:Object3D getObjectById]( [page:Integer id] )
```

id -- Unique number of the object instance

Searches through the object's children and returns the first with a matching id.

```
[method:Vector3 getWorldPosition]( [page:Vector3 optionalTarget] )
```

optionalTarget — Optional target to set the result. Otherwise, a new Vector3 is created.

Returns a vector representing the position of the object in world space.

```
[method:Quaternion getWorldQuaternion]( [page:Quaternion optionalTarget] )
```

optionalTarget — Optional target to set the result. Otherwise, a new Quaternion is created.

Returns a quaternion representing the rotation of the object in world space.

```
[method:Euler getWorldRotation]( [page:Euler optionalTarget] )
```

optionalTarget — Optional target to set the result. Otherwise, a new Euler is created.

Returns the euler angles representing the rotation of the object in world space.

```
[method:Vector3 getWorldScale]( [page:Vector3 optionalTarget] )
```

optionalTarget — Optional target to set the result. Otherwise, a new Vector3 is created.

Returns a vector of the scaling factors applied to the object for each axis.

```
[method:Vector3 getWorldDirection]( [page:Vector3 optionalTarget] )
```

optionalTarget — Optional target to set the result. Otherwise, a new `` Returns a vector representing the direction of object's positive z-axis |

[method:Object3D translateOnAxis]([page:Vector3 axis], [page:Float distance]) -- A normalized vector in object space.

distance -- The distance to translate.

Translate an object by distance along an axis in object space. The axis

[method:Object3D rotateOnAxis]([page:Vector3 axis], [page:Float angle]) -- A normalized vector in object space.

angle -- The angle in radians.

Rotate an object along an axis in object space. The axis is assumed to

[method:Array raycast]([page:Raycaster raycaster], [page:Array intersections]) -- Abstract method to get intersections between a casted ray and this object.

[.updateMatrix\(\)](#)

Обновление локального преобразования. Update the local transformation matrix.

[.updateMatrixWorld \(force \)](#)

Обновление глобальных преобразований объекта и его дочерних

[.world_to_local - \(vector \)](#)

| vector - A world vector.

Метод обновляет вектор из world space to local space.

Исходники

[Object3D.js на github.com](#)

Raycaster

Примечание переводчика: Raycasting в переводе на русский - бросание луча (ray - луч, casting - бросание). ... читать далее

Данный класс предназначен для оказания помощи с [raycasting](#).

[Вот описание рейкастинга](#) в Википедии на русском языке.

Рейкастинг используется для выборки курсором мышки (вычисляет какие из объектов в трехмерном пространстве накрывает курсор мышки) среди других объектов. This class is designed to assist with raycasting. Raycasting is used for mouse

picking (working out what objects in the 3d space the mouse is over) amongst other things. Этот класс упрощает рейкастинг. Рейкастинг используется для выбора и многое другое. This class makes raycasting easier. Raycasting is used for picking and more.

Пример

```
var raycaster = new THREE.Raycaster();
var mouse = new THREE.Vector2();

function onMouseMove( event ) {

    // calculate mouse position in normalized device coordinates
    // (-1 to +1) for both components
    // вычисляем положение курсора мышки в нормализованной
    // системе координат (от -1 до +1) для обоих компонентов

    mouse.x = ( event.clientX / window.innerWidth ) * 2 - 1;
    mouse.y = - ( event.clientY / window.innerHeight ) * 2 + 1;

}

function render() {

    // update the picking ray with the camera and mouse position
    // обновляем луч выборки с позиций камеры и курсора мышки
    raycaster.setFromCamera( mouse, camera );

    // calculate objects intersecting the picking ray
    // вычисляем объекты, пересекающие луч выборки
    var intersects = raycaster.intersectObjects( scene.children );

    for ( var i = 0; i < intersects.length; i++ ) {

        intersects[ i ].object.material.color.set( 0xff0000 );

    }

    renderer.render( scene, camera );

}

window.addEventListener( 'mousemove', onMouseMove, false );

window.requestAnimationFrame(render);
```

[Больше примеров:](#)

[Raycasting to a Mesh](#),
[Raycasting to a Mesh in using an OrthographicCamera](#),
[Raycasting to a Mesh with BufferGeometry](#),
[Raycasting to a Line](#),
[Raycasting to Points](#),
[Terrain raycasting](#),
[Raycasting using an octree](#),
[Raycasting to paint voxels](#),
[Raycast to a Texture](#)

Конструктор

`Raycaster(origin, direction, near, far) {`

origin -- Начальный вектор, из которого берется луч.

direction — Направляющий вектор, который задает направление лучу. Должен быть нормализован.

near — Все возвращаемые результаты должны быть дальше near. near не может быть отрицательным. Значение по умолчанию равно 0.

far — Все возвращаемые результаты должны быть ближе far. far не может быть меньше near. Значение по умолчанию равно Infinity (бесконечность).

Данный метод создает новый объект raycaster.

Свойства

.far

Фактор дальности рейкастера. Это значение показывает какие объекты

.linePrecision

Фактор точности рейкастера при пересечении объектов [Line](#).

.near

Фактор близости рейкастера. Это значение показывает какие объекты

.params

Это объект со следующими свойствами:

```
{  
  Mesh: {},
```

```
Line: {},
LOD: {},
Points: { threshold: 1 },
Sprite: {}
}
```

.ray

Луч, используемый для рейкастинга.

Методы

set(origin, direction)

origin -- Начальный вектор, из которого берется луч.

direction -- Нормализованный вектор направления, который

Метод обновляет луч (ray) с новыми значениями origin и direct

setFromCamera(coords, camera)

coords — 2D coordinates of the mouse, in normalized device coordinates

camera -- камера, из которой должен исходить луч.

Метод обновляет луч (ray) с новыми координатами курсора мыши

intersectObject(object, recursive)

object — The object to check for intersection with the ray.

recursive -- If true, it also checks all descendants. Otherwise it only

Checks all intersection between the ray and the object with or without

[{ distance, point, face, faceIndex, indices, object }, ...]

distance -- расстояние между началом луча и точкой пересечения

[page:Vector3] point -- точка пересечения, in world coordinates

[page:Face3] face -- пересекаемая грань

faceIndex -- индекс пересекаемой грани

indices -- индексы вершин, включенных в пересекаемую грань

[page:Object3D] object -- пересекаемый объект

[page:Vector2] uv -- координаты U,V точки пересечения

When intersecting a [page:Mesh] with a [page:BufferGeometry], the *

Raycaster delegates to the [page:Object3D.raycast raycast] method

Note that for meshes, faces must be pointed towards the origin of the object's local coordinate system.

[method:Array intersectObjects]([page:Array objects], [page:Boolean recursive]) — The objects to check for intersection with the ray. If true, it also checks all descendants of the objects.

Checks all intersection between the ray and the objects with or without the descendants. Intersections are returned sorted by distance, closest first. Intersections are of the same form as those returned by [page:.intersectObject].

Исходники

[Raycaster.js на github.com](#)

Uniform

Uniform'ы являются глобальными переменными [GLSL](#). Они передаются в шейдерные программы.

Пример

При объявлении uniform из [ShaderMaterial](#), он объявляется по значению или по объекту.

```
uniforms: {
    time: { value: 1.0 },
    resolution: new THREE.Uniform(new THREE.Vector2())
}
```

Типы Uniform

Каждый uniform должен иметь свойство value. Тип этого value должен соответствовать типу переменной uniform в коде GLSL, как указано для примитивных типов GLSL в таблице ниже. The type of the value must correspond to the type of the uniform variable in the GLSL code as specified for the primitive GLSL types in the table below.

Поддерживаются также структуры и массивы uniform. Массивы примитивных типов GLSL должны либо указываться как массивы соответствующих объектов THREE, либо как двумерный массив, содержащий данные всех объектов. Uniform structures and arrays are also supported. GLSL arrays of primitive type must either be specified as

an array of the corresponding THREE objects or as a flat array containing the data of all the objects. Другими словами; примитивы GLSL в массивах не должны быть представлены массивами. Это правило не применяется транзитивно. In other words; GLSL primitives in arrays must not be represented by arrays. This rule does not apply transitively. Массив из массивов vec2, каждый из которых длиной в пять векторов, должен быть массивом массивов либо из пяти объектов Vector2, либо из десяти чисел. An array of vec2 arrays, each with a length of five vectors, must be an array of arrays, of either five Vector2 objects or ten numbers.

Типы Uniform

| ТИПЫ GLSL | ТИПЫ JAVASCRIPT |
|--------------|--------------------------------------|
| int | Number |
| float | Number |
| bool | Boolean |
| bool | [page:Number] |
| vec2 | THREE.Vector2 |
| vec2 | Float32Array(*) |
| vec2 | [page:Array Array (*)] |
| vec3 | [page:Vector3 THREE.Vector3] |
| vec3 | [page:Color THREE.Color] |
| vec3 | [page:Float32Array Float32Array] (*) |
| vec3 | [page:Array Array] (*) |
| vec4 | [page:Vector4 THREE.Vector4] |
| vec4 | [page:Quaternion THREE.Quaternion] |
| vec4 | [page:Float32Array Float32Array] (*) |
| vec4 | [page:Array Array] (*) |
| mat2 | [page:Float32Array Float32Array] (*) |
| mat2 | [page:Array Array] (*) |
| mat3 | [page:Matrix3 THREE.Matrix3] |
| mat3 | [page:Float32Array Float32Array] (*) |
| mat3 | [page:Array Array] (*) |
| mat4 | [page:Matrix3 THREE.Matrix4] |
| mat4 | [page:Float32Array Float32Array] (*) |
| mat4 | [page:Array Array] (*) |
| ivec2, bvec2 | [page:Float32Array Float32Array] (*) |
| ivec2, bvec2 | [page:Array Array (*)] |
| ivec3, bvec3 | [page:Int32Array Int32Array (*)] |
| ivec3, bvec3 | [page:Array Array (*)] |
| ivec4, bvec4 | [page:Int32Array Int32Array (*)] |
| ivec4, bvec4 | [page:Array Array (*)] |
| sampler2D | [page:Texture THREE.Texture] |

| samplerCube | [page:CubeTexture THREE.CubeTexture]
(*) Same for an (innermost) array (dimension) of the same GLSL type, содержащего компоненты всех векторов или матриц в массиве.

Конструктор

`Uniform(value)`

`value` -- Объект, содержащий значение для установки в uniform. Его тип должен быть одним из типов Uniform описанных выше.

Свойства

`.value`

Текущее значение uniform.

`.dynamic`

Sets whether this uniform is updated at each render call when used by a renderer. You must set this attribute by calling [page:.onUpdate].

Методы

`.clone()`

Возвращает клона данного uniform. Если свойство value uniform является объектом с методом clone(), то это используется, в противном случае, значение копируется по назначению. Значения массива являются общими для клонов Uniform.

В качестве примера использования данного метода посмотрите [WebGL deferred animation](#). Returns a clone of this uniform. If the uniform's value property is an Object with a clone() method, this is used, otherwise the value is copied by assignment. Array values are shared between cloned Uniforms. See WebGL deferred animation for an example of this method in use.

Исходники

[Uniform.js на github.com](#)

СПИСОК УСТАРЕВШИХ ЭЛЕМЕНТОВ API

Поскольку интерфейс `three.js` довольно быстро развивается, можно столкнуться с примерами, предлагающими использование элементов API, которые больше не являются частью ядра.

Ниже приведен список таких элементов, а также информация относительно их замены.

Звуковое сопровождение

Audio

`Audio.load` устарел, взамен используйте [`AudioLoader`](#).

AudioAnalyser

`AudioAnalyser.getData()` был переименован в
[`AudioAnalyser.getFrequencyData\(\)`](#).

BinaryTextureLoader

`BinaryTextureLoader` был переименован в [`DataTextureLoader`](#).

Буферные элементы

BufferAttribute

`BufferAttribute.length` был переименован в
[`BufferAttribute.count`](#).

DynamicBufferAttribute

`DynamicBufferAttribute` был удален. Вместо него используйте
[`BufferAttribute.setDynamic\(true \)`](#).

Int8Attribute

`Int8Attribute` был удален. Вместо него используйте
[`Int8BufferAttribute`](#).

Uint8Attribute

`Uint8Attribute` был удален. Вместо него используйте [`Uint8BufferAttribute`](#).

`Uint8ClampedAttribute`

`Uint8ClampedAttribute` был удален. Вместо него используйте [`Uint8ClampedBufferAttribute`](#).

`Int16Attribute`

`Int16Attribute` был удален. Вместо него используйте [`Int16BufferAttribute`](#).

`Uint16Attribute`

`Uint16Attribute` был удален. Вместо него используйте [`Uint16BufferAttribute`](#).

`Int32Attribute`

`Int32Attribute` был удален. Вместо него используйте [`Int32BufferAttribute`](#).

`Uint32Attribute`

`Uint32Attribute` был удален. Вместо него используйте [`Uint32BufferAttribute`](#).

`Float32Attribute`

`Float32Attribute` был удален. Вместо него используйте [`Float32BufferAttribute`](#).

`Float64Attribute`

`Float64Attribute` был удален. Вместо него используйте [`Float64BufferAttribute`](#).

Камеры

[`PerspectiveCamera`](#)

`PerspectiveCamera.setLens()` is deprecated. Use [page:`PerspectiveCamera.setFocalLength()`] and [page:`PerspectiveCamera.filmGauge()`] for a photographic setup.

Константы

[\[page:LineStrip\]](#)

[\[page:LinePieces\]](#)

LinePieces mode is no longer supported for [\[page:Line\]](#)s. Create a [\[page:LineSegments\]](#) instead.

Core

[\[page:EventDispatcher\]](#)

`EventDispatcher.apply` has been removed. Inherit or `Object.assign` the prototype to mix-in instead.

[\[page:Raycaster\]](#)

`Raycaster.params.PointCloud` has been renamed to [\[page:Raycaster.params.Points\]](#).

[\[page:Uniform\]](#)

`Uniform.dynamic` has been removed. Use `object.onBeforeRender()` instead.

`Uniform.onUpdate` has been removed. Use `object.onBeforeRender()` instead.

Extras

[\[page:ClosedSplineCurve3\]](#)

`ClosedSplineCurve3` has been deprecated. Use [\[page:CatmullRomCurve3\]](#) instead.

[\[page:SplineCurve3\]](#)

`SplineCurve3` has been deprecated. Use [\[page:CatmullRomCurve3\]](#) instead.

Geometry

[\[page:BufferGeometry\]](#)

`BufferGeometry.addIndex` has been renamed to [\[page:BufferGeometry.setIndex\]](#).

`BufferGeometry.addDrawCall` is now [\[page:BufferGeometry.addGroup\]](#).

`BufferGeometry.clearDrawCalls` is now
[page:BufferGeometry.clearGroups].

`BufferGeometry.computeTangents` has been removed.

`BufferGeometry.computeOffsets` has been removed.

`BufferGeometry.drawcalls` has been renamed to
[page:BufferGeometry.groups].

`BufferGeometry.offsets` has been renamed to
[page:BufferGeometry.groups].

[\[page:CubeGeometry\]](#)

`CubeGeometry` has been renamed to [page:BoxGeometry].

[\[page:Geometry\]](#)

`Geometry.computeTangents()` has been removed.

[\[page:GeometryUtils\]](#)

`GeometryUtils.merge` has been moved to [page:Geometry]. Use
[page:Geometry.merge](geometry2, matrix, materialIndexOffset)
instead.

`GeometryUtils.center` has been moved to [page:Geometry]. Use
[page:Geometry.center]() instead.

[\[page:Plane\]](#)

`Plane.isIntersectionLine]()` has been renamed to
[page:Plane.intersectsLine]().

Helpers

[\[page:BoundingBoxHelper\]](#)

`BoundingBoxHelper` has been deprecated. Use [page:BoxHelper]
instead.

[\[page:EdgesHelper\]](#)

`EdgesHelper` has been removed. Use [page:EdgesGeometry] instead.

[\[page:GridHelper\]](#)

GridHelper.setColors() has been deprecated, pass them in the constructor instead.

[\[page:WireframeHelper WireframeHelper\]](#)

WireframeHelper has been removed. Use [\[page:WireframeGeometry\]](#) instead.

Lights

[\[page:Light\]](#)

Light.onlyShadow has been removed.

Light.shadowCameraLeft is now [\[page:Light.shadow.camera.left\]](#).

Light.shadowCameraRight is now [\[page:Light.shadow.camera.right\]](#).

Light.shadowCameraTop is now [\[page:Light.shadow.camera.top\]](#).

Light.shadowCameraBottom is now [\[page:Light.shadow.camera.bottom\]](#).

Light.shadowCameraNear is now [\[page:Light.shadow.camera.near\]](#).

Light.shadowCameraFar is now [\[page:Light.shadow.camera.far\]](#).

Light.shadowCameraVisible has been removed. Use [\[page:CameraHelper\]](#) (light.shadow.camera) instead.

Light.shadowMapWidth is now [\[page:Light.shadow.mapSize.width\]](#).

Light.shadowMapHeight is now [\[page:Light.shadow.mapSize.height\]](#).

Loaders

[\[page:XHRLoader\]](#)

XHRLoader has been renamed to [\[page:FileLoader\]](#).

Maths

[\[page:Box2\]](#)

Box2.center has been renamed to [\[page:Box2.getCenter\]\(\)](#).

Box2.empty has been renamed to [page:Box2.isEmpty]().

Box2.isIntersectionBox has been renamed to [page:Box2.intersectsBox]().

Box2.size has been renamed to [page:Box2.getSize]().

[\[page:Box3\]](#)

Box3.center has been renamed to [page:Box3.getCenter]().

Box3.empty has been renamed to [page:Box3.isEmpty]().

Box3.isIntersectionBox has been renamed to [page:Box3.intersectsBox]().

Box3.isIntersectionSphere has been renamed to
[page:Box3.intersectsSphere]().

Box3.size has been renamed to [page:Box3.getSize]().

[\[page:Face4\]](#)

Face4 has been removed. Use [page:Face3] instead.

[\[page:Line3\]](#)

Line3.center has been renamed to [page:Line3.getCenter]().

[\[page:Math\]](#)

Math.random16() has been deprecated. Use Math.random() instead.

[\[page:Matrix3\]](#)

Matrix3.flattenToArrayOffset is deprecated. Use [page:Matrix3.toArray]() instead.

Matrix3.multiplyVector3 has been removed. Use vector.applyMatrix3(matrix) instead.

Matrix3.multiplyVector3Array has been renamed to
[page:Matrix3.applyToVector3Array](array).

`Matrix3.applyToBuffer` has been removed. Use `matrix.applyToBufferAttribute(attribute)` instead.

`Matrix3.applyToVector3Array` has been removed.

[\[page:Matrix4\]](#)

`Matrix4.flattenToArrayOffset()` is deprecated. Use [\[page:Matrix4.toArray\]\(\)](#) instead.

`Matrix4.extractPosition()` has been renamed to [\[page:Matrix4.copyPosition\]\(matrix \)](#).

`Matrix4.getPosition()` has been removed. Use [\[page:Vector3.setFromMatrixPosition\]\(matrix \)](#) instead.

`Matrix4.setRotationFromQuaternion()` has been renamed to [\[page:Matrix4.makeRotationFromQuaternion\]\(quaternion \)](#).

`Matrix4.multiplyVector3()` has been removed. Use `vector.applyMatrix4(matrix)` instead.

`Matrix4.multiplyVector4()` has been removed. Use `vector.applyMatrix4(matrix)` instead.

`Matrix4.multiplyVector3Array()` has been renamed to [\[page:Matrix4.applyToVector3Array\]\(array \)](#).

`Matrix4.rotateAxis()` has been removed. Use [\[page:Matrix4.transformDirection\]\(matrix \)](#) instead.

`Matrix4.crossVector()` has been removed. Use `vector.applyMatrix4(matrix)` instead.

`Matrix4.rotateX()` has been removed.

`Matrix4.rotateY()` has been removed.

`Matrix4.rotateZ()` has been removed.

`Matrix4.rotateByAxis()` has been removed.

`Matrix4.applyToBuffer()` has been removed. Use `matrix.applyToBufferAttribute()` instead.

`Matrix4.applyToVector3Array()` has been removed.

`Matrix4.makeFrustum()` has been removed. Use [\[page:Matrix4.makePerspective\]](#)(left, right, top, bottom, near, far) instead.

[\[page:Quaternion.multiplyVector3\]](#)

`Quaternion.multiplyVector3()` has been removed. Use `vector.applyQuaternion(quaternion)` instead.

[\[page:Ray\]](#)

`Ray.isIntersectionBox()` has been renamed to [\[page:Ray.intersectsBox\]\(\)](#).

`Ray.isIntersectionPlane` has been renamed to [\[page:Ray.intersectsPlane\]](#).

`Ray.isIntersectionSphere` has been renamed to [\[page:Ray.intersectsSphere\]](#).

[\[page:Vector2\]](#)

`Vector2.fromAttribute()` has been renamed to [\[page:Vector2.fromBufferAttribute\]\(\)](#).

[\[page:Vector3\]](#)

`Vector3.setEulerFromRotationMatrix()` has been removed. Use [\[page:Euler.setFromRotationMatrix\]\(\)](#) instead.

`Vector3.setEulerFromQuaternion()` has been removed. Use [\[page:Euler.setFromQuaternion\]\(\)](#) instead.

`Vector3.getPositionFromMatrix()` has been renamed to [\[page:Vector3.setFromMatrixPosition\]\(\)](#).

`Vector3.getScaleFromMatrix()` has been renamed to
[page:`Vector3.setFromMatrixScale`].

`Vector3.getColumnFromMatrix()` has been renamed to
[page:`Vector3.setFromMatrixColumn`]().

`Vector3.applyProjection()` has been removed. Use
[page:`Vector3.applyMatrix4`]() instead.

`Vector3.fromAttribute()` has been renamed to
[page:`Vector3.fromBufferAttribute`]().

[page:`Vector4`]

`Vector4.fromAttribute()` has been renamed to
[page:`Vector4.fromBufferAttribute`]().

[page:`Vertex`]

`Vertex` has been removed. Use [page:`Vector3`] instead.

[page:`Spline`]

`Spline` has been removed. Use [page:`CatmullRomCurve3`] instead.

Materials

[page:`Material`]

`Material.wrapAround` has been removed.

`Material.wrapRGB` has been removed.

[page:`MeshFaceMaterial`]

`MeshFaceMaterial` has been removed. Use [page:`MultiMaterial`] instead.

[page:`MeshPhongMaterial`]

`MeshPhongMaterial.metal` has been removed. Use
[page:`MeshStandardMaterial`] instead.

[page:`ParticleBasicMaterial`]

`ParticleBasicMaterial` has been renamed to [page:`PointsMaterial`].

[page:`ParticleSystemMaterial`]

`ParticleBasicMaterial` has been renamed to [page:`PointsMaterial`].

[\[page:PointCloudMaterial\]](#)

PointCloudMaterial has been renamed to [\[page:PointsMaterial\]](#).

[\[page:ShaderMaterial.derivatives\]](#)

ShaderMaterial.derivatives has been moved to
[\[page:ShaderMaterial.extensions.derivatives\]](#).

Objects

[\[page:LOD.objects\]](#)

LOD.objects has been renamed to [\[page:LOD.levels\]](#).

[\[page:Object3D\]](#)

Object3D.eulerOrder is now [\[page:Object3D.rotation.order\]](#).

Object3D.getChildByName() has been renamed to
[\[page:Object3D.getObjectName\]\(\)](#).

Object3D.renderDepth has been removed. Use
[\[page:Object3D.renderOrder\]](#) instead.

Object3D.translate() has been removed. Use
[\[page:Object3D.translateOnAxis\]\(\)](#) instead.

Object3D.useQuaternion has been removed. The library now uses
quaternions by default.

[\[page:Particle\]](#)

ParticleSystem has been renamed to [\[page:Sprite\]](#).

[\[page:ParticleSystem\]](#)

ParticleSystem has been renamed to [\[page:Points\]](#).

[\[page:PointCloud\]](#)

PointCloud has been renamed to [\[page:Points\]](#).

[\[page:Shape\]](#)

Shape.extrude has been removed. Use [\[page:ExtrudeGeometry\]](#) instead.

Shape.makeGeometry has been removed. Use [\[page:ShapeGeometry\]](#)
instead.

Renderer

[page:Projector]

CanvasRenderer был перенесен в
</examples/js/renderers/CanvasRenderer.js>.

[page:Projector]

Projector был перенесен в </examples/js/renderers/Projector.js>.

Projector.projectVector() is now [page:Vector.project]().

Projector.unprojectVector() is now [page:Vector.unproject]().

Projector::pickingRay() is now [page:Raycaster.setFromCamera]().

[page:WebGLProgram]

WebGLProgram.uniforms is now [page: WebGLProgram.getUniforms]().

WebGLProgram.attributes is now [page: WebGLProgram.getAttributes]().

[page:WebGLRenderer]

WebGLRenderer.supportsFloatTextures() is now
[page:WebGLRenderer.extensions.get]('OES_texture_float').

WebGLRenderer.supportsHalfFloatTextures() is now
[page:WebGLRenderer.extensions.get]('OES_texture_half_float').

WebGLRenderer.supportsStandardDerivatives() is now
[page:WebGLRenderer.extensions.get]('OES_standard_derivatives').

WebGLRenderer.supportsCompressedTextureS3TC() is now
[page:WebGLRenderer.extensions.get](
'WEBGL_compressed_texture_s3tc').

WebGLRenderer.supportsCompressedTexturePVRTC() is now
[page:WebGLRenderer.extensions.get](
'WEBGL_compressed_texture_pvr').

`WebGLRenderer.supportsBlendMinMax()` is now
[page:`WebGLRenderer.extensions.get`]('EXT_blend_minmax').

`WebGLRenderer.supportsVertexTextures()` is now
[page:`WebGLRenderer.capabilities.vertexTextures`].

`WebGLRenderer.supportsInstancedArrays()` is now
[page:`WebGLRenderer.extensions.get`]('ANGLE_instanced_arrays').

`WebGLRenderer.enableScissorTest()` is now
[page:`WebGLRenderer.setScissorTest`]().

`WebGLRenderer.initMaterial()` has been removed.

`WebGLRenderer.addPrePlugin()` has been removed.

`WebGLRenderer.addPostPlugin()` has been removed.

`WebGLRenderer.updateShadowMap()` has been removed.

`WebGLRenderer.setTexture` is deprecated, use
[page:`WebGLRenderer.setTexture2D`]() instead.

`WebGLRenderer.shadowMapEnabled` is now
[page:`WebGLRenderer.shadowMap.enabled`].

`WebGLRenderer.shadowMapType` is now
[page:`WebGLRenderer.shadowMap.type`].

`WebGLRenderer.shadowMapCullFace` is now
[page:`WebGLRenderer.shadowMap.cullFace`].

`WebGLRenderer.shadowMap.cullFace` is deprecated. Set
[page:`WebGLRenderer.shadowMap.renderReverseSided`] to true or false
instead.

[page:`WebGLRenderTarget`]

`WebGLRenderTarget.wrapS` is now

[page:WebGLRenderTarget.texture.wrapS].

WebGLRenderTarget.wrapT is now
[page:WebGLRenderTarget.texture.wrapT].

WebGLRenderTarget.magFilter is now
[page:WebGLRenderTarget.texture.magFilter].

WebGLRenderTarget.minFilter is now
[page:WebGLRenderTarget.texture.minFilter].

WebGLRenderTarget.anisotropy is now
[page:WebGLRenderTarget.texture.anisotropy].

WebGLRenderTarget.offset is now
[page:WebGLRenderTarget.texture.offset].

WebGLRenderTarget.repeat is now
[page:WebGLRenderTarget.texture.repeat].

WebGLRenderTarget.format is now
[page:WebGLRenderTarget.texture.format].

WebGLRenderTarget.type is now
[page:WebGLRenderTarget.texture.type].

WebGLRenderTarget.generateMipmaps is now
[page:WebGLRenderTarget.texture.generateMipmaps].

Текстуры

[page:ImageUtils

ImageUtils.loadTexture устарел. Вместо него используйте
[TextureLoader](#).

ImageUtils.loadTextureCube устарел. Вместо него используйте
[CubeTextureLoader](#).

ImageUtils.loadCompressedTexture был удален. Вместо него

используйте [DDSLoader](#).

`ImageUtils.loadCompressedTextureCube` был удален. Вместо него
используйте [DDSLoader](#).

Исходники

[Three.Legacy.js в этом справочнике](#)

[Three.Legacy.js на github.com](#)

ДОПОЛНЕНИЯ

CurveUtils

A class containing utility functions for curves.

Note that these are all linear functions so it is necessary to calculate separately for x, y (and z, w if present) components of a curve.

Методы

```
[method:Number interpolate]( p0, p1, p2, p3, t )
```

| t -- interpolation weight.

| p0, p1, p2, p3 -- the points defining the spline curve.

Used internally by [page:SplineCurve SplineCurve].

```
[method:Number tangentQuadraticBezier]( t, p0, p1, p2 )
```

| t -- the point at which to calculate the tangent.

| p0, p1, p2 -- the three points defining the quadratic Bezier curve.

Calculate the tangent at the point t on a quadratic Bezier curve given by the three points.

Used internally by [page:QuadraticBezierCurve QuadraticBezierCurve].

```
[method:Number tangentCubicBezier]( t, p0, p1, p2, p3 )
```

| t -- the point at which to calculate the tangent.

| p0, p1, p2, p3 -- the points defining the cubic Bezier curve.

Calculate the tangent at the point t on a cubic Bezier curve given by the four points.

Used internally by [page:CubicBezierCurve CubicBezierCurve].

```
[method:Number tangentSpline]( t, p0, p1, p2, p3 )
```

| t -- the point at which to calculate the tangent.

| p0, p1, p2, p3 -- the points defining the spline curve.

Calculate the tangent at the point t on a spline curve given by the four points.

Исходники

[link:[https://github.com/mrdoob/three.js/blob/master/src/\[path\].js](https://github.com/mrdoob/three.js/blob/master/src/[path].js)
src/[path].js]

SceneUtils

A class containing useful utility functions for scene manipulation.

Методы

```
[method:Group createMultiMaterialObject]( [page:Geometry geometry  
geometry -- The geometry for the set of materials.  
materials -- The materials for the object.
```

Creates a new Group that contains a new mesh for each material defined in materials. Beware that this is not the same as an array of materials which defines multiple materials for 1 mesh.

This is mostly useful for objects that need both a material and a wireframe implementation.

```
[method:null attach]( [page:Object3D child], [page:Object3D scene  
child -- The object to add to the parent  
scene -- The scene to detach the object on.  
parent -- The parent to attach the object from.
```

Attaches the object to the parent without the moving the object in the worldspace. Beware that to do this the matrixWorld needs to be updated, this can be done by calling the updateMatrixWorld method on the parent object.

```
[method:null detach]( [page:Object3D child], [page:Object3D parent  
child -- The object to remove from the parent.  
scene -- The scene to attach the object on.  
parent -- The parent to detach the object from.
```

Detaches the object from the parent and adds it back to the scene without moving in worldspace. Beware that to do this the matrixWorld needs to be updated, this can be done by calling the updateMatrixWorld method on the parent object.

Исходники

[link:[https://github.com/mrdoob/three.js/blob/master/src/\[path\].js](https://github.com/mrdoob/three.js/blob/master/src/[path].js)
src/[path].js]

ShapeUtils

Класс содержащий вспомогательные функции для форм. A class containing utility functions for shapes.

Заметьте, что это все линейные функции, так что их нужно рассчитывать по отдельности для компонентов вектора x, y (и z, w, если имеются). Note that these are all linear functions so it is necessary to calculate separately for x, y (and z, w if present) components of a vector.

Методы

[method:Number area](contour)

contour -- 2D polygon.

Calculate area of a (2D) contour polygon.

[method:Number b2](t, p0, p1, p2)

t -- number

p0, p1, p2 -- x, y, z or w components of a quadratic bezier curve.

Note that this is a linear function so it is necessary to calculate separately for x, y (and z for 3D curves) components of a curve.

Used internally by [page:QuadraticBezierCurve QuadraticBezierCurve], [page:QuadraticBezierCurve3 QuadraticBezierCurve3] and [page:Font Font].

[method:Number b3](t, p0, p1, p2, p3)

t -- number.

p0, p1, p2, p3 -- x, y or z components of a cubic bezier curve..

Note that this is a linear function so it is neccessary to calculate separately for x, y (and z for 3D curves) components of a curve.

Used internally by [page:CubicBezierCurve CubicBezierCurve], [page:CubicBezierCurve3 CubicBezierCurve3] and [page:Font Font].

[method:Boolean isClockwise](pts)

pts -- points defining a 2D polygon

Note that this is a linear function so it is neccessary to calculate separately for x, y components of a polygon.

Used internally by [page:Path Path], [page:ExtrudeGeometry ExtrudeGeometry] and [page:ShapeBufferGeometry ShapeBufferGeometry].

[method:null triangulate](contour, indices)

contour -- 2D polygon.

indices --

Used internally by [page:ExtrudeGeometry ExtrudeGeometry] and [page:ShapeBufferGeometry ShapeBufferGeometry] to calculate faces.

[method:null triangulateShape](contour, holes)

contour -- 2D polygon.

holes -- array of holes

Used internally by [page:ExtrudeGeometry ExtrudeGeometry] and [page:ShapeBufferGeometry ShapeBufferGeometry] to calculate faces in shapes with holes.

Исходники

[link:[https://github.com/mrdoob/three.js/blob/master/src/\[path\].js](https://github.com/mrdoob/three.js/blob/master/src/[path].js)
src/[path].js]

Анимация collada-файлов

[name]

This class animates an object based on an hierarchy. This hierarchy can be Object3ds or bones.

Конструктор

```
[name]([page:Object3d root], [page:String name])
```

root -- The mesh to animate.

name -- The name of the animation

Creates an animation for root. The animation data is gathered from AnimationHandler based on its name.

Свойства

```
[property:Object3d root]
```

The root object of the animation.

[property:Object data]

The data containing the animation

[property:Array hierarchy]

The objects that are influenced by the animation.

[property:number currentTime]

The time elapsed since the last start/restart of the animation.

[property:number timeScale]

How much to scale the speed of the animation. Defaults to 1.

[property:boolean.isPlaying]

Indicates whether the animation is playing. This shouldn't be adapted by user code.

[property:boolean isPaused]

Indicates whether the animation is paused. This shouldn't be adapted by user code.

[property:boolean loop]

Set to make the animation restart when the animation ends.

[property:number interpolationType]

The type to indicate how to interpolate between 2 data points.

Методы

[method:null play]([page:Number startTime])

Starts the animation at the startTime (in seconds) of the animation.

[method:null stop]()

Stops the animation.

[method:Boolean update]([page:Number deltaTimeMS])

deltaTimeMS -- The time of the between the previous frame and this frame in miliseconds.

Updates the animation in time. This shouldn't be called by user code. The animationHandler calls this method.

[method:array interpolateCatmullRom]([page:Array points], [page:

points -- The 4 control point to calculate CatMullRom

scale -- The scale between the previous key and the next key

Interpolates the point based on the key. Is used in update.

[method:Object getNextKeyWith]([page:String type], [page:Objec

type -- The animationtype for the key. Can be "pos", "rot" and "scl".
h -- The object of the hierarchy that contains the key
key -- The index of the next possible key.
Gets the next key. Is used in Update.

[method:Object getPrevKeyWith]([page:String type], [page:Object h], [page:Number key])

type -- The animationtype for the key. Can be "pos", "rot" and "scl".
h -- The object of the hierarchy that contains the key
key -- The index of the prev possible key.
Gets the previous key. Is used in Update.

Исходники

[link:[https://github.com/mrdoob/three.js/blob/master/examples/js/loaders/collada/\[name\].js](https://github.com/mrdoob/three.js/blob/master/examples/js/loaders/collada/[name].js)]

[name]

The AnimationHandler handles the initialisation of the Animation data and the animations itself. It keeps track of every animation and if it's active or not. It also update all animations which are active if its method *update* is called.

Конструктор

[name]()

The animationHandler can't be called as constructor.

Свойства

[property:number CATMULLROM]

Enum Value to indicate that the animation needs to be interpolated as CATMULLROM.

[property:number CATMULLROM_FORWARD]

Enum Value to indicate that the animation needs to be interpolated as CATMULLROM_FORWARD.

[property:number LINEAR]

Enum Value to indicate that the animation needs to be interpolated as LINEAR.

Методы

[method:null removeFromUpdate]([page:Animation animation])

animation -- The Animation to remove from the update.

Removes the animation from the update cycle. This gets called when the animation stops. This shouldn't be called by usercode.

[method:Object get]([page:String name])

name -- The name of the animationData
Gets the animationData from its library.

[method:null update]([page:Number deltaTimeMS])

deltaTimeMS -- Time elapsed since last update in milliseconds.
Updates all active animations with deltaTime.

[method:null parse]([page:Object root])

root -- object
Parses the object to get the hierarchy.

[method:null add]([page:object data])

data -- The animationData
Adds the animationData from its library.

[method:null addToUpdate]([page:Animation animation])

animation -- The Animation to add from the update.
Adds the animation from the update cycle. This gets called when the animation starts. This shouldn't be called by user code.

Исходники

[link:[https://github.com/mrdoob/three.js/blob/master/examples/js/loaders/collada/\[name\].js](https://github.com/mrdoob/three.js/blob/master/examples/js/loaders/collada/[name].js)]

[name]

Runs a keyframe animation as imported from the [page:ColladaLoader].

Examples

[example:webgl_loader_collada_keyframe loader / collada / keyframe]

Конструктор

[name]([page:Object data])

data -- An individual animation object from a the [page:ColladaLoader],
e.g. loadedColladaObj.animations[0]

Creates a new keyframe animation and initializes it to the first keyframes.

Свойства

[property:Object3D root]

The root object of the animation

[property:Object data]

The data containing the animation

[property:Array hierarchy]

The objects that are influenced by the animation.

[property:number currentTime]

The time elapsed since the last start/restart of the animation.

[property:number timeScale]

How much to scale the speed of the animation. Defaults to 1.

[property:boolean isPlaying]

Indicates whether the animation is playing. This shouldn't be adapted by

user code.

[property:boolean isPaused]

Indicates whether the animation is paused. This shouldn't be adapted by user code.

[property:boolean loop]

Set to make the animation restart when the animation ends.

Методы

[method:null play]([page:Number startTime])

Starts the animation at the startTime (in seconds) of the animation.

[method:null stop]()

Stops the animation.

[method:null update]([page:Float deltaTime])

deltaTime -- The change in time in seconds

Updates the keyframe animation

[method:Object getNextKeyWith]([page:String sid], [page:Integer h], [page:Integer key])

sid -- The string id

h -- The index in the heirarchy to use

key -- The index of which key to start from

Used internally to traverse the animation

[method:Object getPrevKeyWith]([page:String sid], [page:Integer h], [page:Integer key])

sid -- The string id

h -- The index in the heirarchy to use

key -- The index of which key to start from

Used internally to traverse the animation
[link:[https://github.com/mrdoob/three.js/blob/master/examples/js/loaders/cexamples/js/loaders/collada/\[name\].js](https://github.com/mrdoob/three.js/blob/master/examples/js/loaders/cexamples/js/loaders/collada/[name].js)]

Curve

An abstract base class for creating a curve object that contains methods for interpolation.

For an array of Curves see [page:CurvePath].

Examples

[example:webgl_geometry_extrude_splines geometry / extrude /

Конструктор

Curve()

This constructor creates a new [name].

Методы

[method:Vector getPoint](t)

Returns a vector for point t of the curve where t is between 0 and 1. Must be implemented in the extending class.

[method:Vector getPointAt](u)

Returns a vector for point at relative position in curve according to arc length

[method:Array getPoints](divisions)

Get sequence of points using getPoint(t)

[method:Array getSpacedPoints](divisions)

Get sequence of equi-spaced points using `getPointAt(u)`

[method:Float getLength]()

Get total curve arc length

[method:Array getLengths](divisions)

Get list of cumulative segment lengths

[method:null updateArcLengths]()

Update the cumulative segment distance cache

[method:Float getUtoTmapping](u, distance)

Given u (0 .. 1), get a t to find p . This gives you points which are equidistant

[method:Vector getTangent](t)

Returns a unit vector tangent at t . If the subclassed curve do not implement its tangent derivation, 2 points a small delta apart will be used to find its gradient which seems to give a reasonable approximation

[method:Vector getTangentAt](u)

Returns tangent at equidistant point u on the curve

[method:Object computeFrenetFrames](segments, closed)

Generates the Frenet Frames. Used in geometries like [page:TubeGeometry] or [page:ExtrudeGeometry].

Исходники

[link:[https://github.com/mrdoob/three.js/blob/master/src/\[path\].js](https://github.com/mrdoob/three.js/blob/master/src/[path].js)
src/[path].js]

[page:Curve] →

[name]

An abstract base class further extending [page:Curve]. A CurvePath is simply an array of connected curves, but retains the api of a curve.

Конструктор

[name]()

The constructor take no parameters.

Свойства

[property:array curves]

The array of [page:Curve]s

[property:array bends]

An array of [page:Curve]s used to transform and bend the curve using [page:CurvePath.getWrapPoints].

[property:boolean autoClose]

Whether or not to automatically close the path.

Методы

[method:Array getWrapPoints]([page:Array vertices], [page:CurvePath curve])

vertices -- An array of [page:Vector2]s to modify

curve -- An array of 2d [page:Curve]s

Modifies the array of vertices by warping it by the curve. The curve parameter also accepts objects with similar interfaces such as [page:CurvePath], [page:Path], [page:SplineCurve], etc. Returns the original vertices after modification.

[method:null addWrapPath]([page:Curve curve])

curve -- A [page:Curve] or object with a similar interface.
Pushes a curve onto the bends array.

[method:Geometry createGeometry]([page:Vector3 points])

points -- An array of [page:Vector3]s
Creates a geometry from points

[method:Geometry createPointsGeometry]([page:Integer division])

divisions -- How many segments to create with [page:Vector3]s. Defaults to 12.
Creates a [page:Geometry] object comprised of [page:Vector3]s

[method:Geometry createSpacedPointsGeometry]([page:Integer divisions])

divisions -- How many segments to create with [page:Vector3]s. Defaults to 12.
Creates a [page:Geometry] object comprised of [page:Vector3]s that are equidistant.

[method:null add]([page:Curve curve])

curve -- The [page:Curve curve] to add
Pushes a curve onto the curves array.

[method:null closePath]()

Adds a curve to close the path.

[method:Object getBoundingBox]()

Returns an object with the keys minX, minY, maxX, maxY, (if 3d: maxZ, minZ)

[method:Float getCurveLengths]()

Adds together the length of the curves

[method:Array getTransformedPoints]([page:Integer segments],

segments -- The number of segments to create using the getPoints()

bends -- (optional) An array of [page:Curve]s used to transform the points. Defaults to this.bends if blank.

Uses this CurvePath to generate a series of points transformed by the curves in the bends array. Returns an array of [page:Vector2]s.

[method:Array getTransformedSpacedPoints]([page:Integer seg

segments -- The number of segments to create using the getPoints()

bends -- (optional) Defaults to this.bends if blank. An array of [page:Curve]s used to transform the points.

Uses this CurvePath to generate a series equidistant points that are then transformed by the curves in the bends. Returns an array of [page:Vector2]s.

Исходники

[link:[https://github.com/mrdoob/three.js/blob/master/src/\[path\].js](https://github.com/mrdoob/three.js/blob/master/src/[path].js)
src/[path].js]

[page:CurvePath] →

[name]

A 2d path representation, comprising of points, lines, and cubes, similar to the html5 2d canvas api. It extends [page:CurvePath].

Конструктор

[name]([page:Array points])

points -- array of Vector2

Creates a Path from the points. The first vector defines the offset. After that the lines get defined.

Свойства

[property:array actions]

The possible actions that define the path.

Методы

[method:null fromPoints]([page:Array vector2s])

Adds to the Path from the points. The first vector defines the offset. After that the lines get defined.

[method:null moveTo]([page:Float x], [page:Float y])

This moves the offset to x and y

[method:null lineTo]([page:Float x], [page:Float y])

This creates a line from the offset to X and Y and updates the offset to

X and Y.

[method:null quadraticCurveTo]([page:Float cpX], [page:Float

This creates a quadratic curve from the offset to x and y with cpX and cpY as control point and updates the offset to x and y.

[method:null bezierCurveTo]([page:Float cp1X], [page:Float cp1Y], [page:Float x], [page:Float y])

This creates a bezier curve from the last offset to x and y with cp1X, cp1Y and cp2X, cp2Y as control points and updates the offset to x and y.

[method:null splineThru] ([page:Array points])

points - An array of [page:Vector2]s

Connects a new [page:SplineCurve] onto the path.

[method:null arc]([page:Float x], [page:Float y], [page:Float radius], [page:Float cx], [page:Float cy], [page:Float startAngle], [page:Float endAngle], [page:Boolean clockwise], [page:Boolean absolute])

x, y -- The center of the arc offset from the last call
radius -- The radius of the arc
startAngle -- The start angle in radians
endAngle -- The end angle in radians
clockwise -- Sweep the arc clockwise.
absolute -- Draw an arc absolutely positioned

Defaults to false

Draw an arc offset from the last call

[method:null absarc]([page:Float x], [page:Float y], [page:Float radius], [page:Float cx], [page:Float cy], [page:Float startAngle], [page:Float endAngle], [page:Boolean clockwise])

x, y -- The absolute center of the arc
radius -- The radius of the arc
startAngle -- The start angle in radians
endAngle -- The end angle in radians
clockwise -- Sweep the arc clockwise.
absolute -- Draw an arc absolutely positioned

Defaults to false

[method:null ellipse]([page:Float x], [page:Float y], [page:Float xRadius], [page:Float yRadius])

x, y -- The center of the ellipse offset from the last call
xRadius -- The radius of the ellipse in the x axis
yRadius -- The radius of the ellipse in the y axis

the y axis startAngle -- The start angle in radians endAngle -- The end angle in radians clockwise -- Sweep the ellipse clockwise. Defaults to false rotation -- The rotation angle of the ellipse in radians, counterclockwise from the positive X axis. Optional, defaults to 0
Draw an ellipse offset from the last call

[method:null absellipse]([page:Float x], [page:Float y], [page:F

x, y -- The absolute center of the ellipse xRadius -- The radius of the ellipse in the x axis yRadius -- The radius of the ellipse in the y axis startAngle -- The start angle in radians endAngle -- The end angle in radians clockwise -- Sweep the ellipse clockwise. Defaults to false rotation -- The rotation angle of the ellipse in radians, counterclockwise from the positive X axis. Optional, defaults to 0
Draw an ellipse absolutely positioned

[method:Array toShapes]([page:Boolean isCCW], [page:Boolean

isCCW -- Changes how solids and holes are generated
noHoles -- Whether or not to generate holes
Converts the Path into an array of Shapes. By default solid shapes are defined clockwise (CW) and holes are defined counterclockwise (CCW). If isCCW is set to true, then those are flipped. If the parameter noHoles is set to true then all paths are set as solid shapes and isCCW is ignored.

Исходники

[link:[https://github.com/mrdoob/three.js/blob/master/src/\[path\].js](https://github.com/mrdoob/three.js/blob/master/src/[path].js)
src/[path].js]

[page:Path] →

[name]

Defines an arbitrary 2d shape plane using paths with optional holes. It can be used with [page:ExtrudeGeometry], [page:ShapeGeometry], to

get points, or to get triangulated faces.

```
var heartShape = new THREE.Shape();

heartShape.moveTo( 25, 25 );
heartShape.bezierCurveTo( 25, 25, 20, 0, 0, 0 );
heartShape.bezierCurveTo( 30, 0, 30, 35, 30, 35 );
heartShape.bezierCurveTo( 30, 55, 10, 77, 25, 95 );
heartShape.bezierCurveTo( 60, 77, 80, 55, 80, 35 );
heartShape.bezierCurveTo( 80, 35, 80, 0, 50, 0 );
heartShape.bezierCurveTo( 35, 0, 25, 25, 25, 25 );

var extrudeSettings = { amount: 8, bevelEnabled: tr
var geometry = new THREE.ExtrudeGeometry( heartShap
var mesh = new THREE.Mesh( geometry, new THREE.Mesh
```

Examples

```
[example:webgl_geometry_shapes geometry / shapes ]  
[example:webgl_geometry_extrude_shapes geometry / extrude / shapes  
]  
[example:webgl_geometry_extrude_shapes2 geometry / extrude /  
shapes2 ]  
[example:webgl_particles_shapes particles / shapes ]
```

Конструктор

[name]()

Свойства

[property:array holes]

An array of [page:Path paths] that define the holes in the shape.

Методы

[method:ShapeGeometry makeGeometry]([page:Object options

options -- This is passed as the second argument to
[page:ShapeGeometry ShapeGeometry]
Convenience method to return ShapeGeometry

[method:Array extractAllPoints]([page:Integer divisions])

divisions -- The number of divisions to create on the shape
Get points of shape and holes (keypoints based on segments parameter)

[method:ExtrudeGeometry extrude]([page:Object options])

options -- This is passed as the second argument to
[page:ExtrudeGeometry ExtrudeGeometry]
Convenience method to return ExtrudeGeometry

[method:Object extractPoints]([page:Integer divisions])

divisions -- The number of divisions to create on the shape
Returns an object with a *shape* and *holes* property that each reference an array of [page:Vector2 Vector2s].

[method:Object extractAllSpacedPoints]([page:Integer divisions]

divisions -- The number of divisions to create on the shape
Returns an object with a *shape* and *holes* property that each reference an array of [page:Vector2 Vector2s]. The points will all be equally spaced along the shape.

[method:Array getPointsHoles]([page:Integer divisions])

divisions -- The number of divisions to create on the shape
Get an array of [page Vector2 Vector2s] that represent the holes in the shape.

[method:Array getSpacedPointsHoles]([page:Integer divisions])

divisions -- The number of divisions to create on the shape
Get an array of equally spaced [page Vector2 Vector2s] that represent the holes in the shape.

Исходники

[link:[https://github.com/mrdoob/three.js/blob/master/src/\[path\].js](https://github.com/mrdoob/three.js/blob/master/src/[path].js)
src/[path].js]

[page:EllipseCurve] →

ArcCurve

Alias for [page:EllipseCurve]

Исходники

[link:[https://github.com/mrdoob/three.js/blob/master/src/\[path\].js](https://github.com/mrdoob/three.js/blob/master/src/[path].js)
src/[path].js]

[page:Curve] →

CatmullRomCurve3

Create a smooth 3d spline curve from a series of points using the Catmull-Rom algorithm

Пример

```
//Create a closed wavy loop
var curve = new THREE.CatmullRomCurve3( [
    new THREE.Vector3( -10, 0, 10 ),
    new THREE.Vector3( -5, 5, 5 ),
    new THREE.Vector3( 0, 0, 0 ),
    new THREE.Vector3( 5, -5, 5 ),
    new THREE.Vector3( 10, 0, 10 )
] );

var geometry = new THREE.Geometry();
geometry.vertices = curve.getPoints( 50 );

var material = new THREE.LineBasicMaterial( { color : 0xff0000 } );
```

[example:webgl_geometry_extrude_splines geometry / extrude /

Конструктор

CatmullRomCurve3([page:Array points])

points – An array of [page:Vector3] points

Свойства

[property:Array points]

[property:Boolean closed] – curve loops back onto itself when true

[property:String type] - possible values are `centripetal` (default)

[property:float tension] - when type is `catmullrom`, defines catm

Методы

See [page:Curve] for inherited methods

Исходники

[link:[https://github.com/mrdoob/three.js/blob/master/src/\[path\].js](https://github.com/mrdoob/three.js/blob/master/src/[path].js)
src/[path].js]

[page:Curve] →

ClosedSplineCurve3

Create a smooth 3d spline curve from a series of points that loops back onto itself. THREE.ClosedSplineCurve3 has been deprecated. Please use THREE.CatmullRomCurve3

Пример

```
//Create a closed wavy loop
var curve = new THREE.ClosedSplineCurve3( [
    new THREE.Vector3( -10, 0, 10 ),
    new THREE.Vector3( -5, 5, 5 ),
    new THREE.Vector3( 0, 0, 0 ),
    new THREE.Vector3( 5, -5, 5 ),
    new THREE.Vector3( 10, 0, 10 )
] );

var geometry = new THREE.Geometry();
geometry.vertices = curve.getPoints( 50 );

var material = new THREE.LineBasicMaterial( { color : 0xff0000 } );
```

[example:webgl_geometry_extrude_splines geometry / extrude /

Конструктор

[name]([page:Array points])

points – An array of [page:Vector3] points

Свойства

[property:Array points]

Методы

See [page:Curve] for inherited methods

Исходники

[link:[https://github.com/mrdoob/three.js/blob/master/src/\[path\].js](https://github.com/mrdoob/three.js/blob/master/src/[path].js)
src/[path].js]

[page:Curve] →

CubicBezierCurve3

Create a smooth 2d [cubic bezier curve](#).

Пример

```
var curve = new THREE.CubicBezierCurve(  
    new THREE.Vector3( -10, 0, 0 ),  
    new THREE.Vector3( -5, 15, 0 ),  
    new THREE.Vector3( 20, 15, 0 ),  
    new THREE.Vector3( 10, 0, 0 )  
);  
  
var path = new THREE.Path( curve.getPoints( 50 ) );  
  
var geometry = path.createPointsGeometry( 50 );  
var material = new THREE.LineBasicMaterial( { color : 0xff0000 } );  
  
// Create the final Object3d to add to the scene  
var curveObject = new THREE.Line( geometry, material );
```

Конструктор

CubicBezierCurve3([page:Vector2 v0], [page:Vector2 v1], [page

[page:Vector2 v0] – The starting point

[page:Vector2 v1] – The first control point

[page:Vector2 v2] – The second control point

[page:Vector2 v3] – The ending point

Свойства

[property:Vector2 v0]

[property:Vector2 v1]

[property:Vector2 v2]

[property:Vector2 v3]

Методы

See [page:Curve] for inherited methods

Исходники

[link:[https://github.com/mrdoob/three.js/blob/master/src/\[path\].js](https://github.com/mrdoob/three.js/blob/master/src/[path].js)
src/[path].js]

[page:Curve] →

[name]

Create a smooth 3d [cubic bezier curve](#).

Пример

```
var curve = new THREE.CubicBezierCurve3(  
    new THREE.Vector3( -10, 0, 0 ),  
    new THREE.Vector3( -5, 15, 0 ),  
    new THREE.Vector3( 20, 15, 0 ),  
    new THREE.Vector3( 10, 0, 0 )  
);  
  
var geometry = new THREE.Geometry();  
geometry.vertices = curve.getPoints( 50 );  
  
var material = new THREE.LineBasicMaterial( { color : 0xff0000 } );  
  
// Create the final Object3d to add to the scene  
var curveObject = new THREE.Line( geometry, material );
```

Конструктор

[name]([page:Vector3 v0], [page:Vector3 v1], [page:Vector3 v2],

[page:Vector3 v0] – The starting point

[page:Vector3 v1] – The first control point

[page:Vector3 v2] – The second control point

[page:Vector3 v3] – The ending point

Свойства

[property:Vector3 v0]

[property:Vector3 v1]

[property:Vector3 v2]

[property:Vector3 v3]

Методы

See [page:Curve] for inherited methods

[Исходники](#)

[link:[https://github.com/mrdoob/three.js/blob/master/src/\[path\].js](https://github.com/mrdoob/three.js/blob/master/src/[path].js)
src/[path].js]

[page:Curve] →

[name]

Creates a 2d curve in the shape of an ellipse.

Пример

```
var curve = new THREE.EllipseCurve(
    0, 0, // ax, aY
    10, 10, // xRadius, yRadius
    0, 2 * Math.PI, // aStartAngle, aEndAngle
    false, // aClockwise
    0 // aRotation
);

var path = new THREE.Path( curve.getPoints( 50 ) );
var geometry = path.createPointsGeometry( 50 );
var material = new THREE.LineBasicMaterial( { color : 0xff0000 } );

// Create the final Object3d to add to the scene
var ellipse = new THREE.Line( geometry, material );
```

Конструктор

[name]([page:Float aX], [page:Float aY], [page:Float xRadius], [

aX – The X center of the ellipse

aY – The Y center of the ellipse

xRadius – The radius of the ellipse in the x direction

yRadius – The radius of the ellipse in the y direction

aStartAngle – The start angle of the curve in radians starting from the middle right side

aEndAngle – The end angle of the curve in radians starting from the middle right side

aClockwise – Whether the ellipse is clockwise

aRotation – The rotation angle of the ellipse in radians, counterclockwise from the positive X axis (optional)

Note: When going clockwise it's best to set the start angle to $(\text{Math.PI} * 2)$ and then work towards lower numbers.

Свойства

[property:Float aX]

[property:Float aY]

[property:Radians xRadius]

[property:Radians yRadius]

[property:Float aStartAngle]

[property:Float aEndAngle]

[property:Boolean aClockwise]

[property:Float aRotation]

Методы

See [page:Curve] for inherited methods

Исходники

[link:[https://github.com/mrdoob/three.js/blob/master/src/\[path\].js](https://github.com/mrdoob/three.js/blob/master/src/[path].js)
src/[path].js]

[page:Curve] →

[name]

A curve representing a 2d line segment

Конструктор

[name]([page:Vector2 v1], [page:Vector2 v2])

v1 – The start point

v2 - The end point

Свойства

[property:Vector2 v1]

[property:Vector2 v2]

Методы

See [page:Curve] for inherited methods

Исходники

[link:[https://github.com/mrdoob/three.js/blob/master/src/\[path\].js](https://github.com/mrdoob/three.js/blob/master/src/[path].js)
src/[path].js]

[page:Curve] →

[name]

A curve representing a 3d line segment

Конструктор

[name]([page:Vector3 v1], [page:Vector3 v2])

v1 – The start point

v2 - The end point

Свойства

[property:Vector3 v1]

[property:Vector3 v2]

Методы

See [page:Curve] for inherited methods

Исходники

[link:[https://github.com/mrdoob/three.js/blob/master/src/\[path\].js](https://github.com/mrdoob/three.js/blob/master/src/[path].js)
src/[path].js]

[page:Curve] →

[name]

Create a smooth 2d [quadratic bezier curve](#).

Пример

```
var curve = new THREE.QuadraticBezierCurve(  
    new THREE.Vector3( -10, 0, 0 ),  
    new THREE.Vector3( 20, 15, 0 ),  
    new THREE.Vector3( 10, 0, 0 )  
);  
  
var path = new THREE.Path( curve.getPoints( 50 ) );  
  
var geometry = path.createPointsGeometry( 50 );  
var material = new THREE.LineBasicMaterial( { color : 0xff0000 } );  
  
//Create the final Object3d to add to the scene  
var curveObject = new THREE.Line( geometry, material );
```

Конструктор

[name]([page:Vector2 v0], [page:Vector2 v1], [page:Vector2 v2])

[page:Vector2 v0] – The starting point

[page:Vector2 v1] – The middle control point

[page:Vector2 v2] – The ending point

Свойства

[property:Vector2 v0]

[property:Vector2 v1]

[property:Vector2 v2]

Методы

See [page:Curve] for inherited methods

Исходники

[link:[https://github.com/mrdoob/three.js/blob/master/src/\[path\].js](https://github.com/mrdoob/three.js/blob/master/src/[path].js)
src/[path].js]

[page:Curve] →

[name]

Create a smooth 3d [quadratic bezier curve](#).

Пример

```
var curve = new THREE.QuadraticBezierCurve3(  
    new THREE.Vector3( -10, 0, 0 ),  
    new THREE.Vector3( 20, 15, 0 ),  
    new THREE.Vector3( 10, 0, 0 )  
);  
  
var geometry = new THREE.Geometry();  
geometry.vertices = curve.getPoints( 50 );  
  
var material = new THREE.LineBasicMaterial( { color : 0xff0000 } );  
  
// Create the final Object3d to add to the scene  
var curveObject = new THREE.Line( geometry, material );
```

Конструктор

[name]([page:Vector3 v0], [page:Vector3 v1], [page:Vector3 v2])

[page:Vector3 v0] – The starting point

[page:Vector3 v1] – The middle control point

[page:Vector3 v2] – The ending point

Свойства

[property:Vector3 v0]

[property:Vector3 v1]

[property:Vector3 v2]

Методы

See [page:Curve] for inherited methods

Исходники

[link:[https://github.com/mrdoob/three.js/blob/master/src/\[path\].js](https://github.com/mrdoob/three.js/blob/master/src/[path].js)
src/[path].js]

[page:Curve] →

[name]

Create a smooth 2d spline curve from a series of points

Пример

```
// Create a sine-like wave
var curve = new THREE.SplineCurve( [
    new THREE.Vector2( -10, 0 ),
    new THREE.Vector2( -5, 5 ),
    new THREE.Vector2( 0, 0 ),
    new THREE.Vector2( 5, -5 ),
    new THREE.Vector2( 10, 0 )
] );

var path = new THREE.Path( curve.getPoints( 50 ) );

var geometry = path.createPointsGeometry( 50 );
var material = new THREE.LineBasicMaterial( { color : 0xff0000 } );

// Create the final Object3d to add to the scene
var splineObject = new THREE.Line( geometry, material );
```

Конструктор

[name]([page:Array points])

points – An array of [page:Vector2] points

Свойства

[property:Array points]

Методы

See [page:Curve] for inherited methods

Исходники

[link:[https://github.com/mrdoob/three.js/blob/master/src/\[path\].js](https://github.com/mrdoob/three.js/blob/master/src/[path].js)
src/[path].js]

[page:Curve] →

[name]

Create a smooth 3d spline curve from a series of points

Пример

```
//Create a closed bent a sine-like wave
var curve = new THREE.SplineCurve3( [
    new THREE.Vector3( -10, 0, 10 ),
    new THREE.Vector3( -5, 5, 5 ),
    new THREE.Vector3( 0, 0, 0 ),
    new THREE.Vector3( 5, -5, 5 ),
    new THREE.Vector3( 10, 0, 10 )
] );

var geometry = new THREE.Geometry();
geometry.vertices = curve.getPoints( 50 );

var material = new THREE.LineBasicMaterial( { color : 0xff0000 } );

//Create the final Object3d to add to the scene
var splineObject = new THREE.Line( geometry, material );
```

[example:webgl_geometry_extrude_splines geometry / extrude /

Конструктор

[name]([page:Array points])

points – An array of [page:Vector3] points

Свойства

[property:Array points]

Методы

See [page:Curve] for inherited methods

Исходники

[link:[https://github.com/mrdoob/three.js/blob/master/src/\[path\].js](https://github.com/mrdoob/three.js/blob/master/src/[path].js)
src/[path].js]

[page:Object3D] →

[name]

An 3D arrow Object.

Пример

```
var dir = new THREE.Vector3( 1, 0, 0 );
    var origin = new THREE.Vector3( 0, 0, 0 );
    var length = 1;
    var hex = 0xffff00;

    var arrowHelper = new THREE.ArrowHelper( dir, origin );
    scene.add( arrowHelper );
```

Конструктор

[name]([page:Vector3 dir], [page:Vector3 origin], [page:Number l

dir -- Vector3 -- direction from origin. Must be a unit vector.

origin -- Vector3

length -- scalar

hex -- hexadecimal value to define color ex:0xffff00

headLength -- The length of the head of the arrow

headWidth -- The length of the width of the arrow

This creates an arrow starting in origin in the direction dir for a certain length. It is also possible to change color.

Свойства

[property:Line line]

Contains the line part of the arrowHelper.

[property:Mesh cone]

Contains the cone part of the arrowHelper.

Методы

[method:null setColor]([page:Number hex])

hex -- The hexadicmal value of the color
Sets the color of the arrowHelper.

[method:null setLength]([page:Number length], [page:Number he

length -- The desired length
headLength -- The length of the head of the arrow
headWidth -- The length of the width of the arrow
Sets the length of the arrowhelper.

[method:null setDirection]([page:Vector3 dir])

dir -- The desired direction. Must be a unit vector.
Sets the direction of the arrowhelper.

Исходники

[link:[https://github.com/mrdoob/three.js/blob/master/src/\[path\].js](https://github.com/mrdoob/three.js/blob/master/src/[path].js)
src/[path].js]

[page:Line] →

[name]

An axis object to visualize the the 3 axes in a simple way.
The X axis is red. The Y axis is green. The Z axis is blue.

Пример

```
var axisHelper = new THREE.AxisHelper( 5 );
    scene.add( axisHelper );
```

Конструктор

[name]([page:Number size])

size -- Define the size of the line representing the axes.
Creates an axisHelper with lines of length size.

Исходники

[link:[https://github.com/mrdoob/three.js/blob/master/src/\[path\].js](https://github.com/mrdoob/three.js/blob/master/src/[path].js)
src/[path].js]

[page:Mesh] →

[name]

A helper object to show the world-axis-aligned bounding box for an object.

Пример

```
var hex = 0xff0000;

var sphereMaterial = new THREE.MeshLambertMaterial();
var sphere = new THREE.Mesh( new THREE.SphereGeometry(),
    sphereMaterial );
scene.add( sphere );

var bbox = new THREE.BoundingBoxHelper( sphere, hex );
bbox.update();
scene.add( bbox );
```

Note that this helper will create a wireframe [page:Mesh] object with a [page:BoxGeometry]; the resulting bounding box object will therefore have face diagonals. You may want to use [page:BoxHelper], which generates a [page:Line] object without face diagonals.

Конструктор

[name]([page:Object3D object], [page:Number hex])

object -- Object3D -- the object3D to show the world-axis-aligned boundingbox.

hex -- hexadecimal value to define color ex:0x888888

This creates an line object to the boundingbox.

Свойства

[property:Object3D object]

Contains the object3D to show the world-axis-aligned boundingbox.

[property:Box3 box]

Contains the bounding box of the object.

Методы

[method:null update]()

Updates the BoundingBoxHelper based on the object property.

Исходники

[link:[https://github.com/mrdoob/three.js/blob/master/src/\[path\].js](https://github.com/mrdoob/three.js/blob/master/src/[path].js)
src/[path].js]

[page:Line] →

[name]

Helper object to show a wireframe box (with no face diagonals) around an object

Пример

```
var sphere = new THREE.SphereGeometry();
            var object = new THREE.Mesh( sphere, new THREE.Mesh
var box = new THREE.BoxHelper( object );
scene.add( box );
```

Конструктор

[name]([page:Object3D object], [page:Color color])

object -- Object3D -- the object3D to show the world-axis-aligned boundingbox.

color -- The color of the helper. This can be a [page:Color], a hexadecimal value and an CSS-Color name. Default is 0xffff00
Creates a new wireframe box matching the size of the passed box.

Свойства

(none)

Методы

[method:null update]([page:Object3D object])

Updates the helper's geometry to match the dimensions of the [page:Geometry.boundingBox bounding box] of the passed object's geometry.

Исходники

[link:[https://github.com/mrdoob/three.js/blob/master/src/\[path\].js](https://github.com/mrdoob/three.js/blob/master/src/[path].js)
src/[path].js]

[page:Line] →

[name]

The camera Helper is an Object3D which helps visualizing what a camera contains in its frustum.

It visualizes the frustum with an line Geometry.

Конструктор

[name]([page:Camera camera])

camera -- The camera to visualize.

This create a new [Name] for the specified camera.

Свойства

[property:object pointMap]

This contains the points to vualize the cameraHelper

[property:Camera camera]

The camera to visualize.

Методы

[method:null update]()

Updates the helper based on the projectionMatrix of the camera.

Исходники

[link:[https://github.com/mrdoob/three.js/blob/master/src/\[path\].js](https://github.com/mrdoob/three.js/blob/master/src/[path].js)
src/[path].js]

[page:Object3D] →

[name]

Visualize a [page:DirectionalLight]'s effect on the scene

Конструктор

[name]([page:DirectionalLight light], [page:Number size])

light -- [page:DirectionalLight] -- Light to visualize

size -- dimensions of the plane

Creates a line and plane to visualize the light's position and direction

Свойства

[property:Line lightPlane]

Contains the line mesh showing the location of the directional light.

[property:DirectionalLight light]

Contains the directionalLight.

[property:Line targetLine]

Contains the line mesh that shows the direction of the light.

Методы

.[method:null update]()

Updates the helper to match the position and direction of the [page:.light].

Исходники

[link:[https://github.com/mrdoob/three.js/blob/master/src/\[path\].js](https://github.com/mrdoob/three.js/blob/master/src/[path].js)
src/[path].js]

[page:Line] →

[name]

Renders [page:ArrowHelper arrows] to visualize an object's [page:Face3 face] normals. Requires that the object's geometry be an instance of [page:Geometry] (does not work with [page:BufferGeometry]), and that face normals have been specified on all [page:Face3 faces] or calculated with [page:Geometry.computeFaceNormals computeFaceNormals].

Пример

```
geometry = new THREE.BoxGeometry( 10, 10, 10, 2, 2,
material = new THREE.MeshBasicMaterial( { color: 0x
object = new THREE.Mesh( geometry, material );

edges = new THREE.FaceNormalsHelper( object, 2, 0x0

scene.add( object );
scene.add( edges );
```

[example:webgl_helpers Example using various helpers]

Конструктор

[name]([page:Object3D object], [page:Number size], [page:Color color], [page:Number linewidth])

object -- object for which to render face normals

size -- size (length) of the arrows

color -- color of the arrows

linewidth -- width of the arrow lines

Свойства

[property:Object3D object]

The attached object

Методы

[method:null update]()

Updates the face normal preview based on movement of the object.

Исходники

[link:[https://github.com/mrdoob/three.js/blob/master/src/\[path\].js](https://github.com/mrdoob/three.js/blob/master/src/[path].js)
src/[path].js]

[page:Line] →

[name]

The GridHelper is an object to define grids. Grids are two-dimensional arrays of lines.

Пример

```
var size = 10;
      var step = 1;

      var gridHelper = new THREE.GridHelper( size, step )
scene.add( gridHelper );
```

[example:webgl_helpers Example using various helpers]

Конструктор

[name]([page:number size], [page:Number step], [page:Color colorCenterLine], [page:Color colorGrid])

size -- The size of the grid

step -- The size of the step between 2 lines

colorCenterLine -- The color of the centerline. This can be a [page:Color], a hexadecimal value and an CSS-Color name. Default is 0x444444

colorGrid -- The color of the lines of the grid. This can be a [page:Color], a hexadecimal value and an CSS-Color name. Default is 0x888888

Creates a new [name] of size 'size' and with steps of size 'step'. Colors are optional.

Исходники

[link:[https://github.com/mrdoob/three.js/blob/master/src/\[path\].js](https://github.com/mrdoob/three.js/blob/master/src/[path].js)]
src/[path].js [page:Object3D] →

[name]

Creates a visual aid for a [page:HemisphereLight HemisphereLight].

Конструктор

[name]([page:HemisphereLight light], [page:Number sphereSize

light -- The HemisphereLight.

sphereSize -- The size of the sphere that shows the location.

Creates an helper for the hemispherelight.

Свойства

[property:Mesh lightSphere]

The sphere mesh that shows the location of the hemispherelight.

[property:HemisphereLight light]

Contains the HemisphereLight.

Методы

[method:null update]()

Updates the helper to match the position and direction of the [page:.light].

Исходники

[link:[https://github.com/mrdoob/three.js/blob/master/src/\[path\].js](https://github.com/mrdoob/three.js/blob/master/src/[path].js)
src/[path].js]

[page:Mesh] →

[name]

This displays a helper object for a [page:PointLight]

Пример

```
var pointLight = new THREE.PointLight( 0xff0000, 1, 100 );
    pointLight.position.set( 10, 10, 10 );
    scene.add( pointLight );

    var sphereSize = 1;
    var pointLightHelper = new THREE.PointLightHelper(
        scene.add( pointLightHelper );
```

[example:webgl_helpers Example using various helpers]

Конструктор

[name]([page:PointLight light], [page:Float sphereSize])

light -- The [page:PointLight] to display.

sphereSize -- The size of the sphere helper

Свойства

[property:PointLight light]

The [page:PointLight] that is being represented.

Методы

[method:null update]()

Updates the light helper.

Исходники

[link:[https://github.com/mrdoob/three.js/blob/master/src/\[path\].js](https://github.com/mrdoob/three.js/blob/master/src/[path].js)
src/[path].js]

[page:Object3D] →

[name]

This displays a cylinder helper object for a [page:SpotLight]

Пример



[View in Examples](#)

Other Examples

[example:webgl_lights_spotlights lights / spotlights]

Code Example

```
var spotLight = new THREE.SpotLight( 0xffffff );
spotLight.position.set( 10, 10, 10 );
scene.add( spotLight );

var spotLightHelper = new THREE.SpotLightHelper( spotLight );
scene.add( spotLightHelper );
```

Конструктор

[name]([page:SpotLight light])

light -- The [page:SpotLight] to display

Свойства

[property:SpotLight light]

The [page:SpotLight]

Методы

[method:null update]()

Updates the light helper.

Исходники

[link:[https://github.com/mrdoob/three.js/blob/master/src/\[path\].js](https://github.com/mrdoob/three.js/blob/master/src/[path].js)
src/[path].js]

[page:Line] →

[name]

Renders [page:ArrowHelper arrows] to visualize an object's vertex normal vectors. Requires that normals have been specified in a [page:BufferAttribute custom attribute] or have been calculated using [page:Geometry.computeVertexNormals computeVertexNormals].

Пример

```
geometry = new THREE.BoxGeometry( 10, 10, 10, 2, 2, 2 );
material = new THREE.MeshBasicMaterial( { color: 0xff0000 } );
object = new THREE.Mesh( geometry, material );

edges = new THREE.VertexNormalsHelper( object, 2, 0x00ff00, 1 );

scene.add( object );
scene.add( edges );
```

[example:webgl_helpers Example using various helpers]

Конструктор

[name]([page:Object3D object], [page:Number size], [page:Color color], [page:Number linewidth])

object -- object for which to render vertex normals
size -- size (length) of the arrows
color -- color of the arrows
linewidth -- width of the arrow lines

Свойства

[property:Object3D object]

The attached object

Методы

[method:null update]()

Updates the vertex normal preview based on movement of the object.

Исходники

[link:[https://github.com/mrdoob/three.js/blob/master/src/\[path\].js](https://github.com/mrdoob/three.js/blob/master/src/[path].js)
src/[path].js]

[page:Object3D] →

[name]

base class for immediate rendering objects.

Конструктор

[name]()

This creates a new [name].

Методы

[method:null render]([page:Function renderCallback])

renderCallback -- A function to render the generated object.

This function needs to be overridden to start the creation of the object and should call renderCallback when finished.

Исходники

[link:[https://github.com/mrdoob/three.js/blob/master/src/\[path\].js](https://github.com/mrdoob/three.js/blob/master/src/[path].js)
src/[path].js]

[page:Mesh] →

[name]

A mesh that can blend together multiple animated morph targets.

Пример

```
[example:webgl_morphtargets_md2_control morphtargets / md2 / contro
```

Конструктор

```
[name]([page:Geometry geometry], [page:Material material])
```

geometry — An instance of [page:Geometry].

material — An instance of [page:Material] (optional).

Свойства

```
[property:object animationsMap]
```

An object of named animations as added by

[page:MorphBlendMesh.createAnimation].

[property:array animationsList]

The list of animations as added by
[page:MorphBlendMesh.createAnimation].

Методы

[method:null setAnimationWeight]([page:String name], [page:Float weight])

name -- The name of the animation

weight -- Weight of the animation, typically 0-1

Set the weight of how much this animation will apply to the overall morph.
0 is off, 1 is full weight.

[method:null setAnimationFPS]([page:String name], [page:Float fps])

name -- The name of the animation

fps -- The number of frames (morphTargets) per second

A frame is typically 1 morph target.

[method:null createAnimation]([page:String name], [page:Integer start], [page:Integer end], [page:Float fps])

name -- The name of the animation

start -- The starting frame (morph)

end -- The ending frame (morph)

fps -- How many frames (morphs) to play per second

Creates an animation object that gets added to both the
[page:MorphBlendMesh.animationsMap animationsMap] and
[page:MorphBlendMesh.animationsList animationsList].

Animation object:

startFrame -- Starting frame

endFrame -- Ending frame

length -- The number of frames

fps -- The frames per second

duration -- The length of the animation in seconds

lastFrame -- The previous frame that was played

currentFrame -- The current frame
active -- Whether or not the animation is being played
time -- The time in seconds of the animation
direction -- Which way to play the animation
weight -- The weight of the animation
directionBackwards -- Is playing backwards
mirroredLoop -- Loop back and forth

[method:null playAnimation]([page:String name])

name -- The name of the animation
Sets the animation to active and animation time to 0

[method:null update]([page:Float delta])

delta -- Time in seconds
Updates and plays the animation

[method:null autoCreateAnimations]([page:Float fps])

fps -- Frames per second
Goes through the geometry's morphTargets and generates animations based on the morphTargets' names. Names are of the form "walk_01", "walk_02", "walk_03", etc or "run001", "run002", "run003".

[method:null setAnimationDuration]([page:String name], [page:F

name -- The name of the animation
duration -- How long in seconds to play the animation
Updates the animation object with proper values to update the duration.

[method:null setAnimationDirectionForward]([page:String name])

name -- The name of the animation
Sets the animation to play forwards

[method:null setAnimationDirectionBackward]([page:String name])

name -- The name of the animation
Sets the animation to play backwards

[method:Float getAnimationDuration]([page:String name])

name -- The name of the animation
Returns the duration in seconds of the animation. Returns -1 if it can't be found.

[method:Float getAnimationTime]([page:String name])

name -- The name of the animation
Returns the current time position of the animation.

[method:null setAnimationTime]([page:String name], [page:Float

name -- The name of the animation
time -- The time in seconds
Sets the current time position of the animation

[method:null stopAnimation]([page:String name])

name -- The name of the animation
Stops the playback of the animation

Исходники

[link:[https://github.com/mrdoob/three.js/blob/master/src/\[path\].js](https://github.com/mrdoob/three.js/blob/master/src/[path].js)
src/[path].js]

ГЕОМЕТРИЧЕСКИЕ ЭЛЕМЕНТЫ

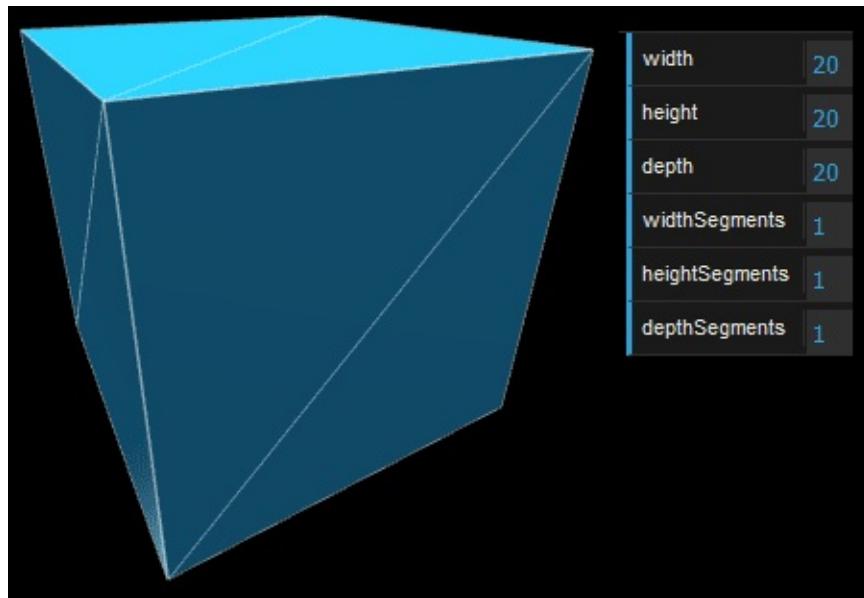
В этом разделе рассматриваются ...

Система координат ...

[BufferGeometry →](#)

BoxBufferGeometry

This is the [BufferGeometry](#) port of [BoxGeometry](#).



Куб с размером стороны равной 20 и числом сегментов по умолчанию, т.е. 1.

Пример

```
var geometry = new THREE.BoxBufferGeometry( 1, 1, 1 );
var material = new THREE.MeshBasicMaterial( {color: 0xffff00} );
var cube = new THREE.Mesh( geometry, material );
scene.add( cube );
```

Конструктор

```
BoxBufferGeometry( width, height, depth, widthSegments, heightSegments )
```

width — ширина куба, размер сторон по оси X. По умолчанию значение равно 1.

height — высота куба, т.е. размер сторон по оси Y. По умолчанию значение равно 1.

depth — глубина куба, т.е. размер сторон по оси Z. По умолчанию значение равно 1.

widthSegments — дополнительный, необязательный параметр. Количество частей, на которые разделяется сторона куба по ширине. По умолчанию значение равно 1.

heightSegments — дополнительный, необязательный параметр. Количество частей, на которые разделяется сторона куба по высоте. По умолчанию значение равно 1.

depthSegments — дополнительный, необязательный параметр. Количество частей, на которые разделяется сторона куба по глубине. По умолчанию значение равно 1.

Свойства

```
.parameters
```

Используем код вышеприведенного примера как основу:

```
geometry.parameters;  
// outputs an object {width: 1, height: 1, depth: 1, widthSegments: undefined, heightSegments: undefined}  
// параметры объекта {width: 1, height: 1, depth: 1, widthSegments: undefined, heightSegments: undefined}  
cube.geometry.parameters; // как указано выше  
cube.geometry.parameters.width; // === 1, т.е. ширина равна 1  
cube.geometry.parameters.widthSegments // === undefined, т.е. неопределено
```

Исходники

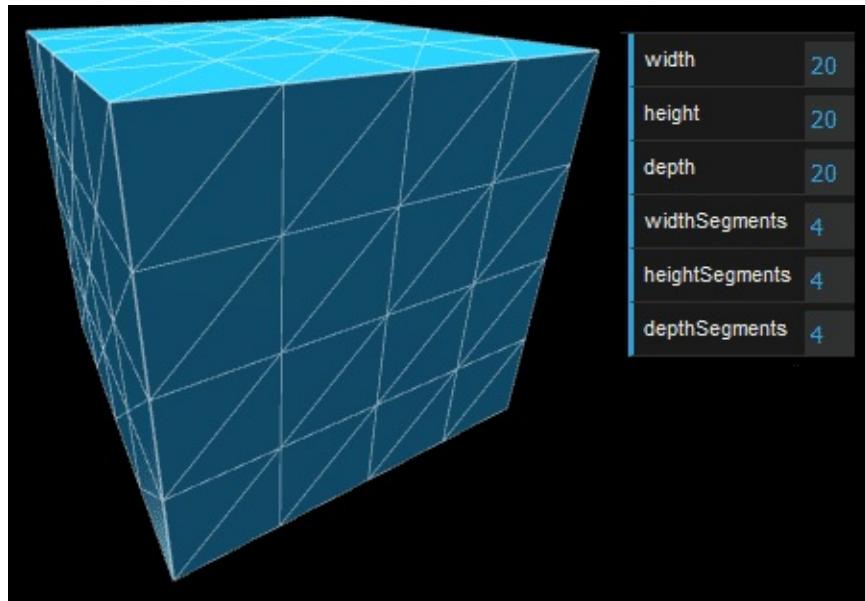
[BoxBufferGeometry.js на github.com](#)

[Geometry →](#)

BoxGeometry

BoxGeometry is the quadrilateral primitive geometry class. Обычно он используется для построения куба или параллелепипеда с

размерами, представленными аргументами конструктора `width` - ширина, `height` - высота, и `depth` - глубина.



Куб со стороной равной 20, разделенной на 4 сегмента.

Пример

```
var geometry = new THREE.BoxGeometry( 1, 1, 1 );
var material = new THREE.MeshBasicMaterial( {color: 0x00ff00} );
var cube = new THREE.Mesh( geometry, material );
scene.add( cube );
```

Конструктор

```
BoxGeometry( width, height, depth, widthSegments, heightSegments, d
```

`width` — ширина куба, размер сторон по оси X.

`height` — высота куба, т.е. размер сторон по оси Y.

`depth` — глубина куба, т.е. размер сторон по оси Z.

`widthSegments` — дополнительный, необязательный параметр.

Количество частей, на которые разделяется сторона куба по ширине. По умолчанию значение равно 1.

`heightSegments` — дополнительный, необязательный параметр.

Количество частей, на которые разделяется сторона куба по высоте. По умолчанию значение равно 1.

`depthSegments` — дополнительный, необязательный параметр.

Количество частей, на которые разделяется сторона куба по глубине. По умолчанию значение равно 1.

Свойства

.parameters

Using the above example code above as our basis:

```
geometry.parameters; // outputs an object {width: 1, height: 1, d  
cube.geometry.parameters; // as above  
cube.geometry.parameters.width; // === 1  
cube.geometry.parameters.widthSegments // === undefined.
```

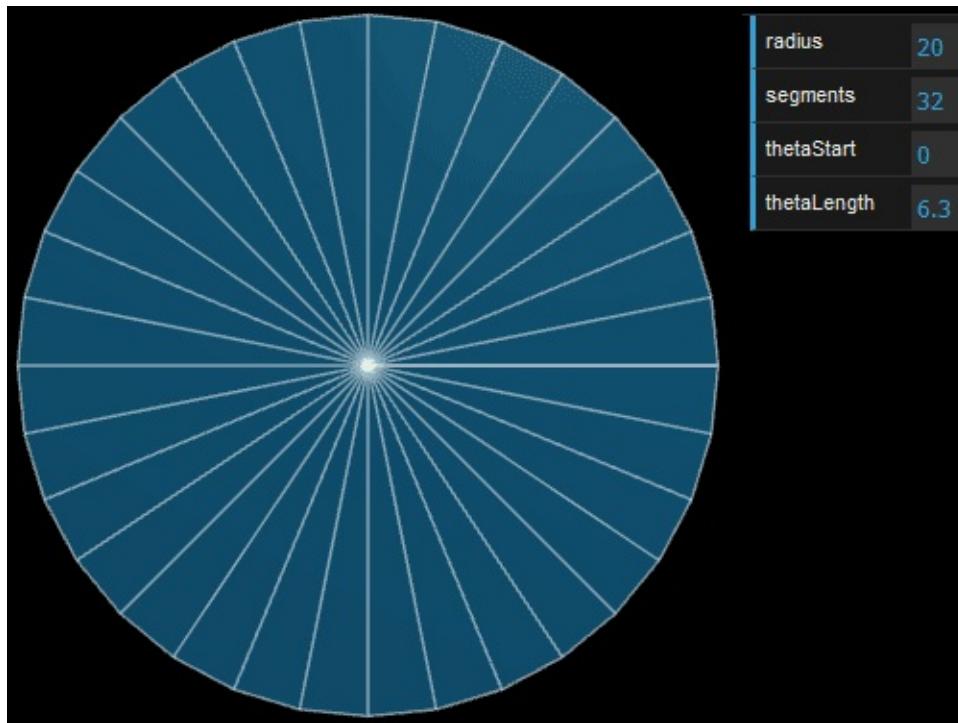
Исходники

[BoxGeometry.js на github.com](#)

[BufferGeometry →](#)

CircleBufferGeometry

Это порт [BufferGeometry](#) из [CircleGeometry](#).



Круг с радиусом 20 и 32 сегментами, остальные параметры по умолчанию.

Обратите внимание на более светлую границу между сегментами, справа.

Так обозначен *thetaStart* — начальный угол.

Пример

```
var geometry = new THREE.CircleBufferGeometry( 5, 32 );
var material = new THREE.MeshBasicMaterial( { color: 0xffff00 } );
var circle = new THREE.Mesh( geometry, material );
scene.add( circle );
```

Конструктор

`CircleBufferGeometry(radius, segments, thetaStart, thetaLength)`

`radius` — радиус круга, значение по умолчанию равно 50.

`segments` — число сегментов (треугольников), минимальное значение равно 3, значение по умолчанию равно 8.

`thetaStart` — начальный угол для первого сегмента, значение по умолчанию равно 0 (положение на 3 часа).

`thetaLength` — центральный угол (часто называемый theta) кругового сектора. Значение по умолчанию равно 2π , что составляет полный круг.

Исходники

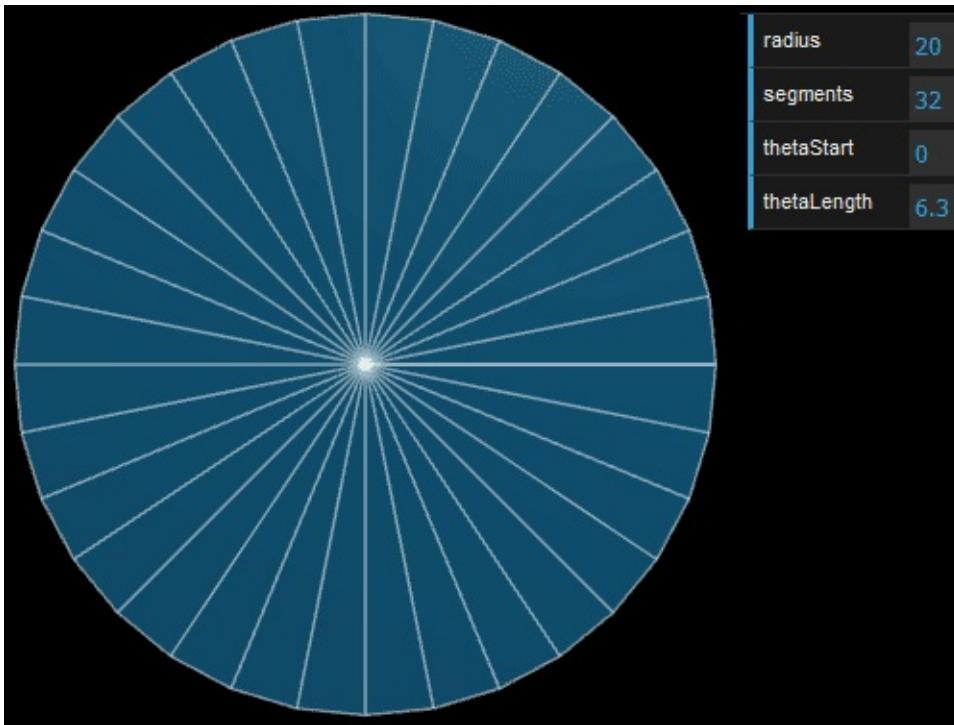
[CircleBufferGeometry.js в этом справочнике](#)

[CircleBufferGeometry.js на github.com](#)

[Geometry →](#)

CircleGeometry

CircleGeometry (геометрия круга) это простая форма евклидовой геометрии. Она построена из нескольких треугольных сегментов, расположенных вокруг центральной точки и is a simple shape of Euclidean geometry. It is constructed from a number of triangular segments that are oriented around a central point and extend as far out as a given radius. It is built counter-clockwise from a start angle and a given central angle. It can also be used to create regular polygons, where the number of segments determines the number of sides.



Круг с радиусом 20 и 32 сегментами, остальные параметры по умолчанию.

Обратите внимание на более светлую границу между сегментами, справа.

Так обозначен *thetaStart* — начальный угол.

Пример

```
var geometry = new THREE.CircleGeometry( 5, 32 );
var material = new THREE.MeshBasicMaterial( { color: 0xffff00 } );
var circle = new THREE.Mesh( geometry, material );
scene.add( circle );
```

Конструктор

`circleGeometry(radius, segments, thetaStart, thetaLength)`

`radius` — радиус круга, значение по умолчанию равно 50.

`segments` — число сегментов (треугольников), минимальное значение равно 3, значение по умолчанию равно 8.

`thetaStart` — начальный угол для первого сегмента, значение по умолчанию равно 0 (положение на 3 часа).

`thetaLength` — центральный угол (часто называемый *theta*) кругового сектора. Значение по умолчанию равно 2π , что составляет полный круг.

Исходники

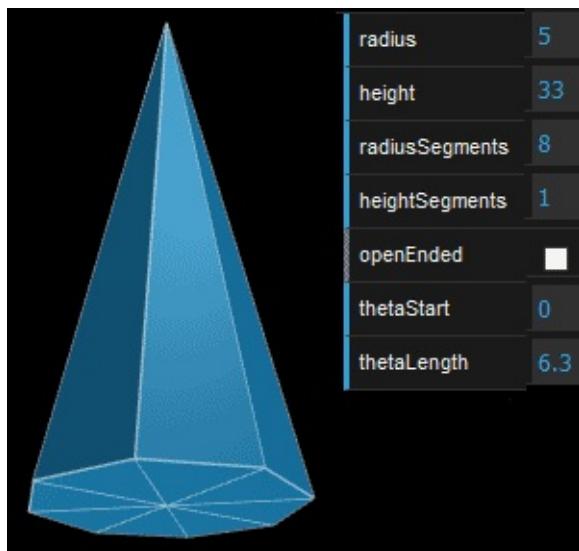
[CircleGeometry.js в этом справочнике](#)

[CircleGeometry.js на github.com](#)

[BufferGeometry](#) →

ConeBufferGeometry

This is the [page:BufferGeometry] port of [page:ConeGeometry].



Конус

Пример

```
var geometry = new THREE.ConeBufferGeometry( 5, 20, 32 );
var material = new THREE.MeshBasicMaterial( {color: 0xffff00} );
var cone = new THREE.Mesh( geometry, material );
scene.add( cone );
```

Конструктор

`ConeBufferGeometry(radius, height, radiusSegments, heightSegments,`

 | radius — радиус основания конуса, значение по умолчанию
 | равно 20.

 | height — высота конуса, значение по умолчанию равно 100.

 | radiusSegments — число сегментов-граней по окружности

конуса, значение по умолчанию равно 8.

`heightSegments` — число рядов граней по высоте конуса.

Значение по умолчанию равно 1.

`openEnded` — логическое значение, показывающее будет ли основание конуса открытым или закрытым. Значением по умолчанию является `false`, т.е. конус закрыт.

`thetaStart` — начальный угол первого сегмента в основании конуса, значение по умолчанию равно 0 (позиция на три часа).
`thetaLength` — центральный угол (зачастую называемый `theta` - тэта) сектора круга в основании конуса. По умолчанию равен 2π , что делает конус завершенным.

Исходники

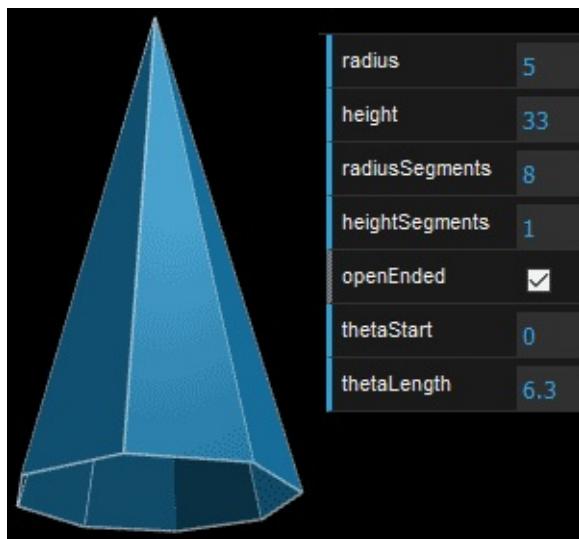
[ConeBufferGeometry.js в этом справочнике](#)

[ConeBufferGeometry.js на github.com](#)

[Geometry →](#)

ConeGeometry

Класс для построения конических фигур.



Конус

Пример

```
var geometry = new THREE.ConeGeometry( 5, 20, 32 );
var material = new THREE.MeshBasicMaterial( {color: 0xffff00} );
var cone = new THREE.Mesh( geometry, material );
scene.add( cone );
```

Конструктор

`ConeGeometry(radius, height, radiusSegments, heightSegments, openEnded)`

`radius` — радиус основания конуса, значение по умолчанию равно 20.

`height` — высота конуса, значение по умолчанию равно 100.

`radiusSegments` — число сегментов-граней по окружности конуса, значение по умолчанию равно 8.

`heightSegments` — число рядов граней по высоте конуса.

Значение по умолчанию равно 1.

`openEnded` — логическое значение, показывающее будет ли основание конуса открытым или закрытым. Значением по умолчанию является `false`, т.е. конус закрыт.

`thetaStart` — начальный угол первого сегмента в основании конуса, значение по умолчанию равно 0 (позиция на три часа).

`thetaLength` — центральный угол (зачастую называемый `theta`-тэта) сектора круга в основании конуса. По умолчанию равен 2π , что делает конус завершенным.

Исходники

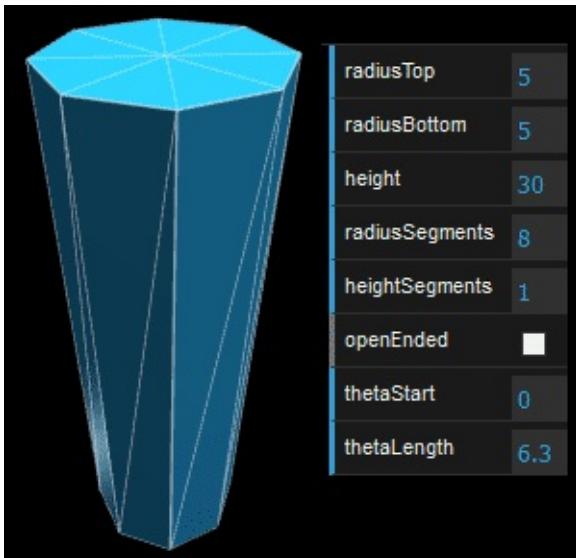
[ConeGeometry.js в этом справочнике](#)

[ConeGeometry.js на github.com](#)

[BufferGeometry →](#)

CylinderBufferGeometry

This is the [page:BufferGeometry] port of [page:CylinderGeometry].



Цилиндр с шестью гранями и закрытыми основаниями

Пример

```
var geometry = new THREE.CylinderBufferGeometry( 5, 5, 20, 32 );
var material = new THREE.MeshBasicMaterial( {color: 0xfffff00} );
var cylinder = new THREE.Mesh( geometry, material );
scene.add( cylinder );
```

Конструктор

`CylinderBufferGeometry(radiusTop, radiusBottom, height, radiusSegm`

`radiusTop` — радиус цилиндра по верху. Значение по умолчанию равно 20.

`radiusBottom` — радиус цилиндра по низу. Значение по умолчанию равно 20.

`height` — высота цилиндра. Значение по умолчанию равно 100.

`radiusSegments` — число сегментов-граней по окружности цилиндра. Значение по умолчанию равно 8.

`heightSegments` — число рядов граней по высоте цилиндра. Значение по умолчанию равно 1.

`openEnded` — логическое значение, показывающее будут ли основания цилиндра открытыми или закрытыми. Значением по умолчанию является `false`, т.е. цилиндр закрыт.

`thetaStart` — начальный угол первого сегмента в основании цилиндра, значение по умолчанию равно 0 (позиция на три часа).

`thetaLength` — центральный угол (зачастую называемый `theta` - тэта) сектора круга в основании цилиндра. По умолчанию равен 2π , что делает цилиндр завершенным.

Исходники

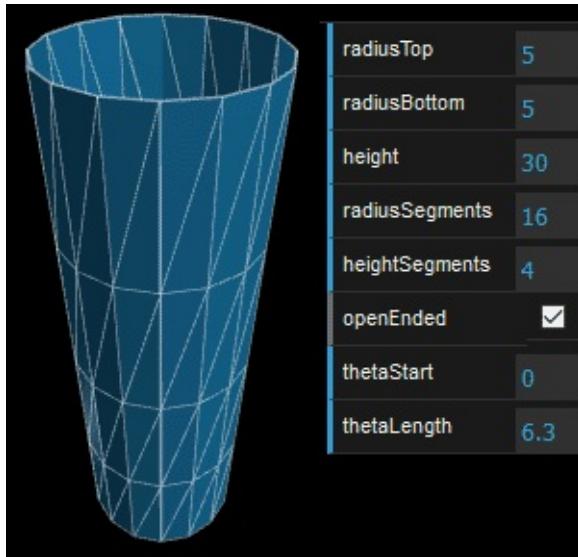
[CylinderBufferGeometry.js в этом справочнике](#)

[CylinderBufferGeometry.js на github.com](#)

[Geometry →](#)

CylinderGeometry

Класс для построения фигур цилиндрической формы.



Цилиндр с открытыми основаниями с 16 сегментами-гранями по окружности и четырьмя сегментами по высоте

Пример

```
var geometry = new THREE.CylinderGeometry( 5, 5, 20, 32 );
var material = new THREE.MeshBasicMaterial( {color: 0xffff00} );
var cylinder = new THREE.Mesh( geometry, material );
scene.add( cylinder );
```

Конструктор

```
CylinderGeometry( radiusTop, radiusBottom, height, radiusSegments,
```

`radiusTop` — радиус цилиндра по верху. Значение по умолчанию равно 20.

`radiusBottom` — радиус цилиндра по низу. Значение по умолчанию равно 20.

`height` — высота цилиндра. Значение по умолчанию равно 100.

`radiusSegments` — число сегментов-граней по окружности цилиндра. Значение по умолчанию равно 8.

`heightSegments` — число рядов граней по высоте цилиндра. Значение по умолчанию равно 1.

`openEnded` — логическое значение, показывающее будут ли основания цилиндра открытыми или закрытыми. Значением по умолчанию является `false`, т.е. цилиндр закрыт.

`thetaStart` — начальный угол первого сегмента в основании цилиндра, значение по умолчанию равно 0 (позиция на три часа).

`thetaLength` — центральный угол (зачастую называемый `theta` - тэта) сектора круга в основании цилиндра. По умолчанию равен 2π , что делает цилиндр завершенным.

Исходники

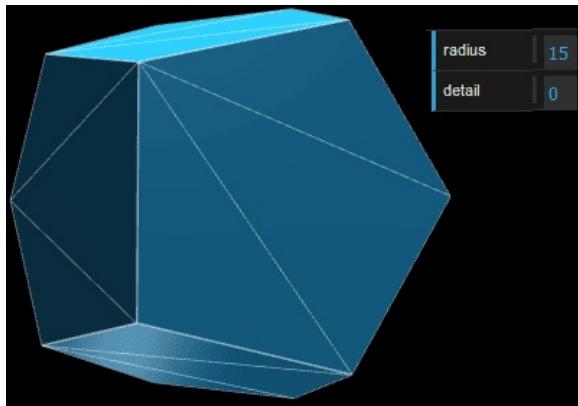
[CylinderGeometry.js в этом справочнике](#)

[CylinderGeometry.js на github.com](#)

[PolyhedronBufferGeometry](#) →

DodecahedronBufferGeometry

A class for generating a dodecahedron geometries.

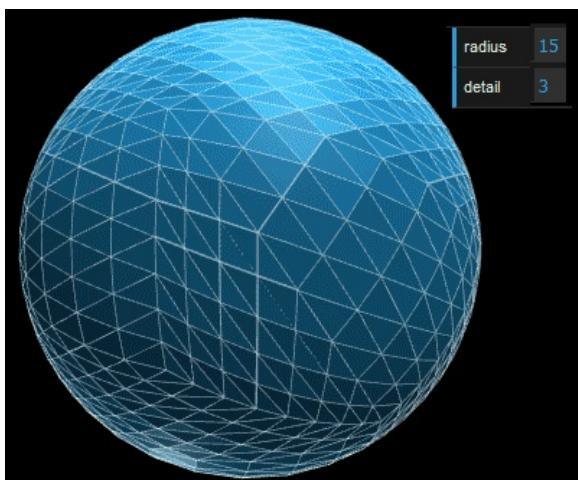


Додекаэдр

Конструктор

```
DodecahedronBufferGeometry( radius, detail )
```

`radius` — радиус додекаэдра. Значение по умолчанию равно 1.
`detail` — Значение по умолчанию равно 0. Установка здесь
значения большего 0, добавит вершин и это будет уже не
додекаэдр.



При установке значения аргумента `detail` равным 3,
получается вот такой «додекаэдр».

Свойства

```
.parameters
```

Объект со всеми параметрами, которые были использованы для построения этой геометрической фигуры.

Исходники

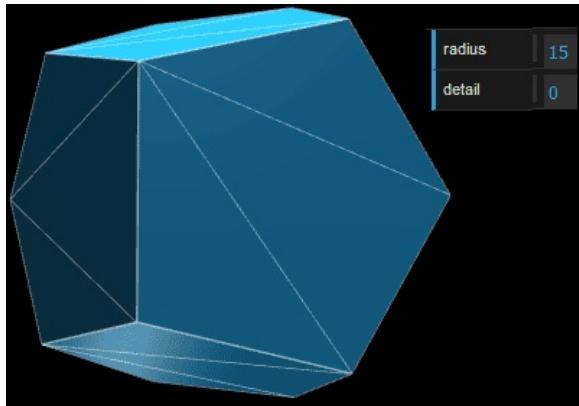
[DodecahedronBufferGeometry.js в этом справочнике](#)

[DodecahedronBufferGeometry.js на github.com](#)

[Geometry →](#)

DodecahedronGeometry

Класс для построения геометрических фигур в виде додекаэдра (двенадцатигранника).



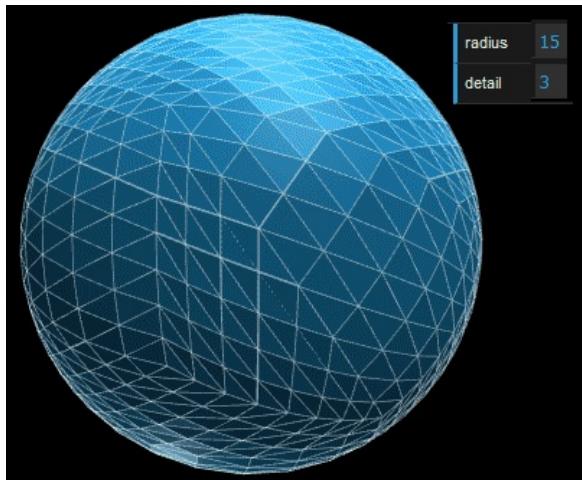
Додекаэдр

Конструктор

`DodecahedronGeometry(radius, detail)`

`radius` — радиус додекаэдра. Значение по умолчанию равно 1.

`detail` — Значение по умолчанию равно 0. Установка здесь значения большего 0, добавит вершин и это будет уже не додекаэдр.



При установке значения аргумента `detail` равным 3, получается вот такой «додекаэдр».

Свойства

.parameters

Объект со всеми параметрами, которые были использованы для построения этой геометрической фигуры.

Исходники

[DodecahedronGeometry.js в этом справочнике](#)

[DodecahedronGeometry.js на github.com](#)

[BufferGeometry →](#)

EdgesGeometry

Этот объект может использоваться в качестве вспомогательного для отображения кромок [геометрического](#) объекта.

Пример

```
var geometry = new THREE.BoxBufferGeometry( 100, 100, 100 );
var edges = new THREE.EdgesGeometry( geometry );
var line = new THREE.LineSegments( edges, new THREE.LineBasicMaterial( { color: 0x0000ff } ) );
scene.add( line );
```

Другие примеры:

helpers

Конструктор

`EdgesGeometry(geometry, thresholdAngle)`

`geometry` — любой геометрический объект.

`thresholdAngle` — кромка отображается только в том случае, если угол (в градусах) между нормалями соседних граней превышает данное значение. По умолчанию оно равно 1 градусу.

Исходники

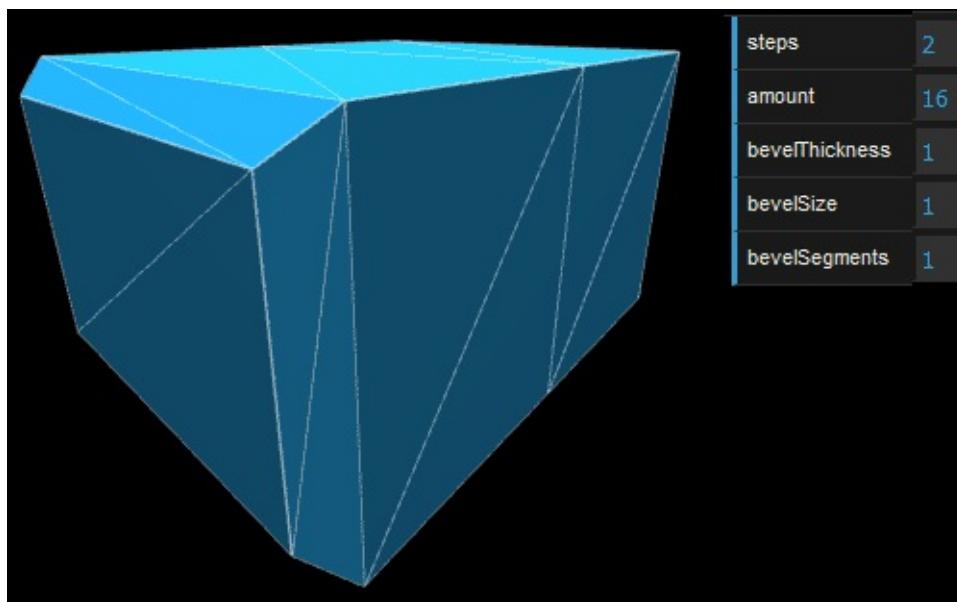
[EdgesGeometry.js в этом справочнике](#)

[EdgesGeometry.js на github.com](#)

[Geometry →](#)

ExtrudeGeometry

Creates extruded geometry from a path shape.



ExtrudeGeometry

Пример

```
var length = 12, width = 8;
```

```

var shape = new THREE.Shape();
shape.moveTo( 0,0 );
shape.lineTo( 0, width );
shape.lineTo( length, width );
shape.lineTo( length, 0 );
shape.lineTo( 0, 0 );

var extrudeSettings = {
    steps: 2,
    amount: 16,
    bevelEnabled: true,
    bevelThickness: 1,
    bevelSize: 1,
    bevelSegments: 1
};

var geometry = new THREE.ExtrudeGeometry( shape, data );
var material = new THREE.MeshBasicMaterial( { color: 0x00ff00 } );
var mesh = new THREE.Mesh( geometry, material ) ;
scene.add( mesh );

```

Конструктор

`ExtrudeGeometry([page:Array shapes], [page:Object options])`

`shapes` — форма или массив форм. Shape or an array of shapes.

`options` — объект, который может содержать следующие параметры:

`curveSegments` — int. number of points on the curves

`steps` — int. number of points used for subdividing segments of extrude spline

`amount` — int. Depth to extrude the shape

`bevelEnabled` — bool. turn on bevel

`bevelThickness` — float. how deep into the original shape bevel goes

`bevelSize` — float. how far from shape outline is bevel

`bevelSegments` — int. number of bevel layers

`extrudePath` — THREE.CurvePath. 3d spline path to extrude shape along. (creates Frames if frames aren't defined)

`frames` — THREE.TubeGeometry.FrenetFrames. containing arrays

of tangents, normals, binormals
material — int. material index for front and back faces
extrudeMaterial — int. material index for extrusion and beveled faces
UVGenerator — Object. object that provides UV generator functions

This object extrudes an 2D shape to an 3D geometry.

Свойства

Методы

```
[method:null addShapeList]([page:Array shapes], [page:Object opti
```

shapes — An Array of shapes to add.
options — Object that can contain the following parameters.
curveSegments — int. number of points on the curves
steps — int. number of points used for subdividing segments of extrusion
amount — int. Depth to extrude the shape
bevelEnabled — bool. turn on bevel
bevelThickness — float. how deep into the original shape bevel goes
bevelSize — float. how far from shape outline is bevel
bevelSegments — int. number of bevel layers
extrudePath — THREE.CurvePath. 3d spline path to extrude shape
frames — THREE.TubeGeometry.FrenetFrames. containing arrays of frames
material — int. material index for front and back faces
extrudeMaterial — int. material index for extrusion and beveled faces
UVGenerator — Object. object that provides UV generator functions

Adds the shapes to the list to extrude.

```
[method:null addShape]([page:Shape shape], [page:Object opti
```

shape — A shape to add.
options — Object that can contain the following parameters.
curveSegments — int. number of points on the curves
steps — int. number of points used for subdividing segments of extrusion
amount — int. Depth to extrude the shape

bevelEnabled — bool. turn on bevel
bevelThickness — float. how deep into the original shape bevel goes
bevelSize — float. how far from shape outline is bevel
bevelSegments — int. number of bevel layers
extrudePath — THREE.CurvePath. 3d spline path to extrude shape
frames — THREE.TubeGeometry.FrenetFrames. containing arrays of frames
material — int. material index for front and back faces
extrudeMaterial — int. material index for extrusion and beveled face
UVGenerator — Object. object that provides UV generator functions

Add the shape to the list to extrude.

Source

[link:[https://github.com/mrdoob/three.js/blob/master/src/\[path\].js](https://github.com/mrdoob/three.js/blob/master/src/[path].js)
src/[path].js] [page:PolyhedronBufferGeometry] →

IcosahedronBufferGeometry

A class for generating an icosahedron geometry.



Икосаэдр

Конструктор

IcosahedronBufferGeometry([page:Float radius], [page:Integer detail])

radius — Default is 1.

detail — Default is 0. Setting this to a value greater than 0 adds more vertices making it no longer an icosahedron. When detail is greater than

1, it's effectively a sphere.

Свойства

[property:Object parameters]

An object with all of the parameters that were used to generate the geometry.

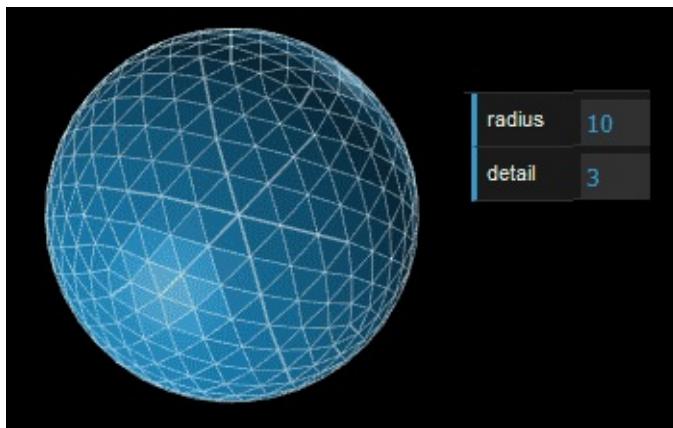
Source

[link:[https://github.com/mrdoob/three.js/blob/master/src/\[path\].js](https://github.com/mrdoob/three.js/blob/master/src/[path].js) src/[path].js]

[Geometry →](#)

IcosahedronGeometry

A class for generating an icosahedron geometry.



"Икосаэдр" со значением detail равном 3

Конструктор

IcosahedronGeometry([page:Float radius], [page:Integer detail])

radius — Default is 1.

detail — Default is 0. Setting this to a value greater than 0 adds more vertices making it no longer an icosahedron. When detail is greater than 1, it's effectively a sphere.

Свойства

[property:Object parameters]

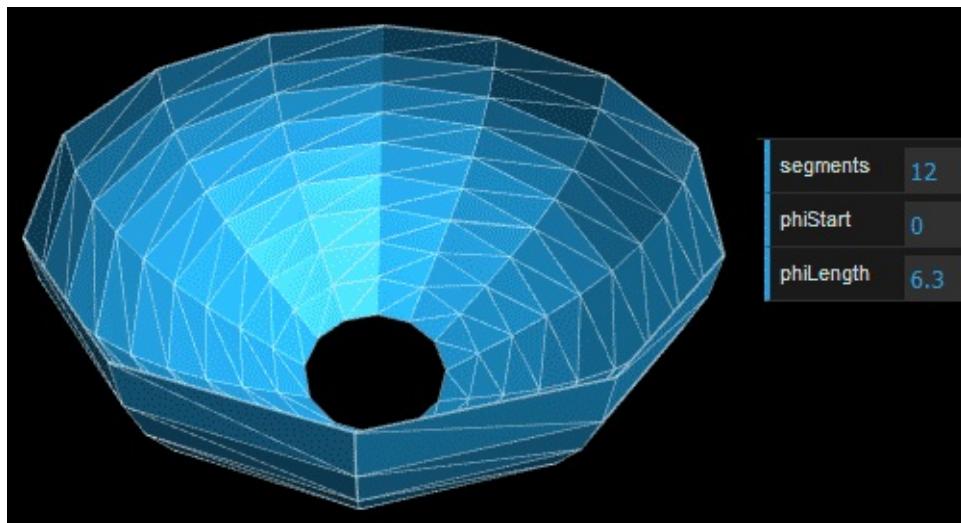
An object with all of the parameters that were used to generate the geometry.

Source

[link:https://github.com/mrdoob/three.js/blob/master/src/[path].js
src/[path].js] [BufferGeometry](#) →

LatheBufferGeometry

This is the [page:BufferGeometry] port of [page:LatheGeometry].



Lathe

Пример

```
var points = [];
for ( var i = 0; i < 10; i ++ ) {
    points.push( new THREE.Vector2( Math.sin( i * 0.2 ) * 10 + 5, ( i
} )
var geometry = new THREE.LatheBufferGeometry( points );
var material = new THREE.MeshBasicMaterial( { color: 0xffff00 } );
var lathe = new THREE.Mesh( geometry, material );
scene.add( lathe );
```

Конструктор

LatheBufferGeometry([page:Array points], [page:Integer segmer

points — Array of Vector2s. The x-coordinate of each point must be greater than zero.

segments — the number of circumference segments to generate. Default is 12.

phiStart — the starting angle in radians. Default is 0.

phiLength — the radian (0 to 2PI) range of the lathed section 2PI is a closed lathe, less than 2PI is a portion. Default is 2PI.

This creates a LatheBufferGeometry based on the parameters.

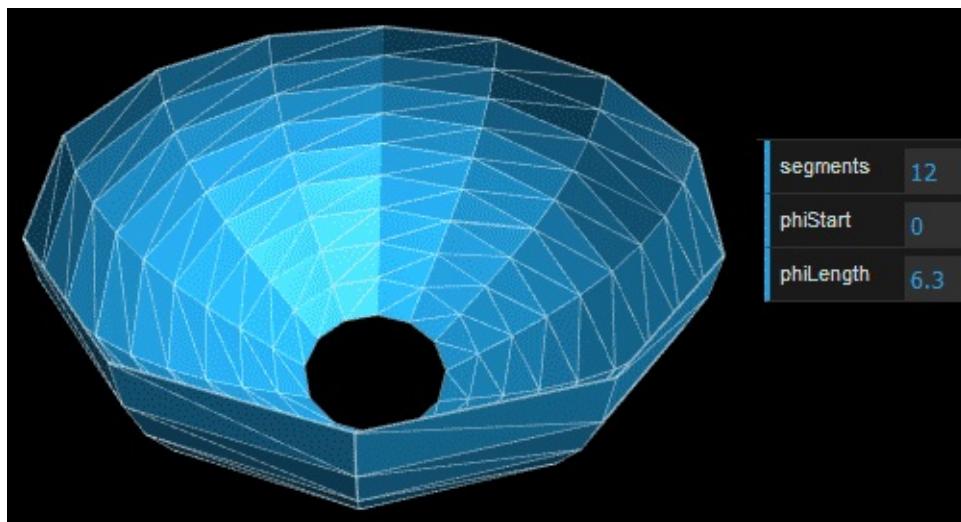
Source

[link:[https://github.com/mrdoob/three.js/blob/master/src/\[path\].js](https://github.com/mrdoob/three.js/blob/master/src/[path].js)
src/[path].js]

[Geometry →](#)

LatheGeometry

Class for generating meshes with axial symmetry. Possible uses include donuts, pipes, vases etc. The lathe rotate around the Y axis.



Lathe

Пример

```
var points = [];
for ( var i = 0; i < 10; i ++ ) {
  points.push( new THREE.Vector2( Math.sin( i * 0.2 ) * 10 + 5, ( i -
} )
var geometry = new THREE.LatheGeometry( points );
```

```
var material = new THREE.MeshBasicMaterial( { color: 0xffff00 } );
var lathe = new THREE.Mesh( geometry, material );
scene.add( lathe );
```

Конструктор

`LatheGeometry([page:Array points], [page:Integer segments], [page:F`

`points` — Array of Vector2s. The x-coordinate of each point must be greater than zero.

`segments` — the number of circumference segments to generate.

Default is 12.

`phiStart` — the starting angle in radians. Default is 0.

`phiLength` — the radian (0 to 2PI) range of the lathed section 2PI is a closed lathe, less than 2PI is a portion. Default is 2PI.

This creates a LatheBufferGeometry based on the parameters.

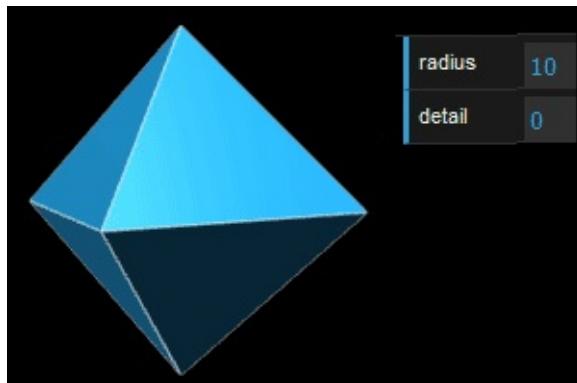
Source

[link:[https://github.com/mrdoob/three.js/blob/master/src/\[path\].js](https://github.com/mrdoob/three.js/blob/master/src/[path].js)
src/[path].js]

[page:PolyhedronBufferGeometry] →

OctahedronBufferGeometry

A class for generating an octahedron geometry.



Octahedron

Конструктор

`OctahedronBufferGeometry([page:Float radius], [page:Integer d`

`radius` — Radius of the octahedron. Default is 1.

`detail` — Default is 0. Setting this to a value greater than zero add vertices making it no longer an octahedron.

Свойства

[property:Object parameters]

An object with all of the parameters that were used to generate the geometry.

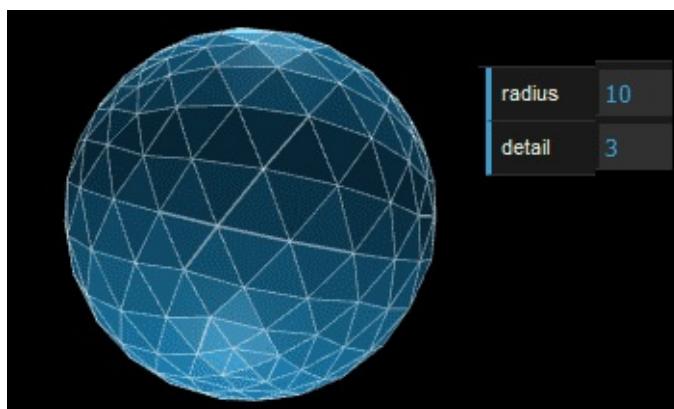
Исходники

[link:[https://github.com/mrdoob/three.js/blob/master/src/\[path\].js](https://github.com/mrdoob/three.js/blob/master/src/[path].js)
src/[path].js]

[Geometry →](#)

OctahedronGeometry

A class for generating an octahedron geometry.



Octahedron

Конструктор

OctahedronGeometry([page:Float radius], [page:Integer detail])

`radius` — Radius of the octahedron. Default is 1.

`detail` — Default is 0. Setting this to a value greater than zero add vertices making it no longer an octahedron.

Свойства

[property:Object parameters]

An object with all of the parameters that were used to generate the geometry.

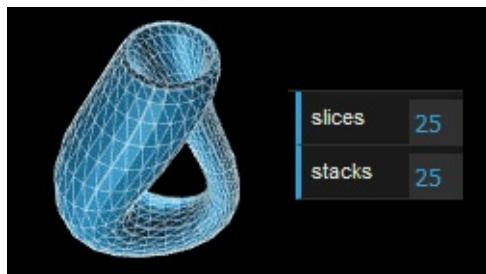
Исходники

[link:[https://github.com/mrdoob/three.js/blob/master/src/\[path\].js](https://github.com/mrdoob/three.js/blob/master/src/[path].js)
src/[path].js]

[BufferGeometry](#) →

ParametricBufferGeometry

Generate geometry representing a parametric surface.



Parametric

Пример

```
var geometry = new THREE.ParametricBufferGeometry( THREE.Parametric  
var material = new THREE.MeshBasicMaterial( { color: 0x00ff00 } );  
var cube = new THREE.Mesh( geometry, material );  
scene.add( cube );
```

Конструктор

ParametricBufferGeometry([page:Function func], [page:Integer s

func — A function that takes in a [page:Float u] and [page:Float v] value each between 0 and 1 and returns a [page:Vector3]

slices — The count of slices to use for the parametric function

stacks — The count of stacks to use for the parametric function

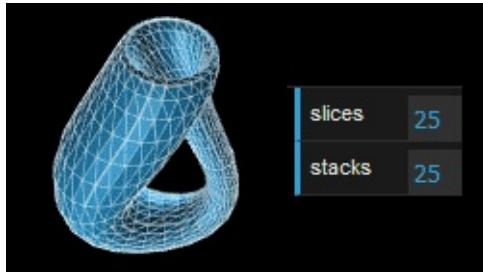
Исходники

[link:[https://github.com/mrdoob/three.js/blob/master/src/\[path\].js](https://github.com/mrdoob/three.js/blob/master/src/[path].js)
src/[path].js]

[Geometry →](#)

ParametricGeometry

Generate geometry representing a parametric surface.



Parametric

Пример

```
var geometry = new THREE.ParametricGeometry( THREE.ParametricGeometry  
var material = new THREE.MeshBasicMaterial( { color: 0x00ff00 } );  
var cube = new THREE.Mesh( geometry, material );  
scene.add( cube );
```

Конструктор

ParametricGeometry([page:Function func], [page:Integer slices],

func — A function that takes in a [page:Float u] and [page:Float v] value each between 0 and 1 and returns a [page:Vector3]

slices — The count of slices to use for the parametric function

stacks — The count of stacks to use for the parametric function

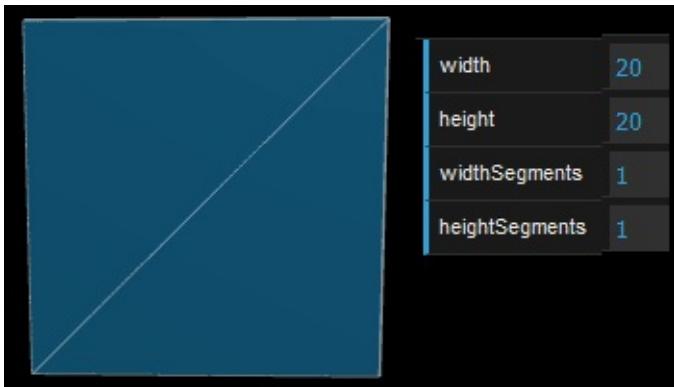
Исходники

[link:[https://github.com/mrdoob/three.js/blob/master/src/\[path\].js](https://github.com/mrdoob/three.js/blob/master/src/[path].js)
src/[path].js]

[BufferGeometry →](#)

PlaneBufferGeometry

This is the [page:BufferGeometry] port of [page:PlaneGeometry].



PlaneGeometry

Пример

```
var geometry = new THREE.PlaneBufferGeometry( 5, 20, 32 );
var material = new THREE.MeshBasicMaterial( {color: 0xffff00, side:
var plane = new THREE.Mesh( geometry, material );
scene.add( plane );
```

Конструктор

`PlaneBufferGeometry(width, height, widthSegments, heightSegments)`

`width` — ширина по оси X.

`height` — высота по оси Y.

`widthSegments` — количество сегментов (частей) по ширине
(дополнительный, необязательный параметр). По умолчанию
равен 1.

`heightSegments` — количество сегментов по высоте
(дополнительный, необязательный параметр). По умолчанию
равен 1.

Исходники

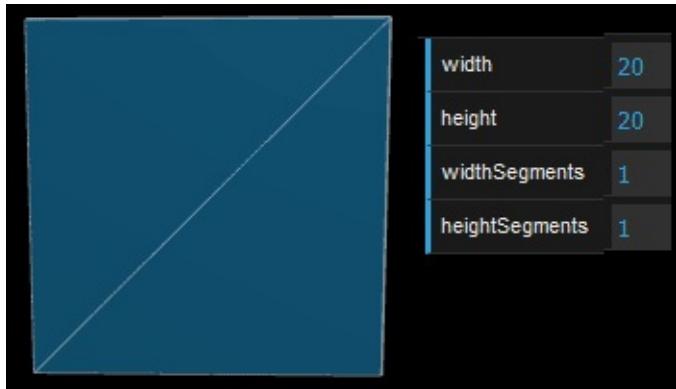
[PlaneBufferGeometry.js в этом справочнике](#)

[PlaneBufferGeometry.js на github.com](#)

[Geometry →](#)

PlaneGeometry

Класс для построения плоских геометрических фигур.



PlaneGeometry

Пример

```
var geometry = new THREE.PlaneGeometry( 5, 20, 32 );  
var material = new THREE.MeshBasicMaterial( {color: 0xffff00, side:  
var plane = new THREE.Mesh( geometry, material );  
scene.add( plane );
```

Конструктор

`PlaneBufferGeometry(width, height, widthSegments, heightSegments)`

`width` — ширина по оси X.

`height` — высота по оси Y.

`widthSegments` — количество сегментов (частей) по ширине
(дополнительный, необязательный параметр). По умолчанию
равен 1.

`heightSegments` — количество сегментов по высоте
(дополнительный, необязательный параметр). По умолчанию
равен 1.

Исходники

[PlaneGeometry.js в этом справочнике](#)

[PlaneGeometry.js на github.com](#)

[BufferGeometry →](#)

PolyhedronBufferGeometry

A polyhedron is a solid in three dimensions with flat faces. This class will take an array of vertices, project them onto a sphere, and then divide them up to the desired level of detail. This class is used by [page:DodecahedronBufferGeometry], [page:IcosahedronBufferGeometry], [page:OctahedronBufferGeometry], and [page:TetrahedronBufferGeometry] to generate their respective geometries.

Пример

```
var verticesOfCube = [
    -1,-1,-1,      1,-1,-1,      1, 1,-1,      -1, 1,-1,
    -1,-1, 1,      1,-1, 1,      1, 1, 1,      -1, 1, 1,
];
var indicesOfFaces = [
    2,1,0,      0,3,2,
    0,4,7,      7,3,0,
    0,1,5,      5,4,0,
    1,2,6,      6,5,1,
    2,3,7,      7,6,2,
    4,5,6,      6,7,4
];
var geometry = new THREE.PolyhedronBufferGeometry( verticesOfCube,
```

Конструктор

PolyhedronBufferGeometry(vertices, faces, radius, detail)

vertices — [page:Array] of points of the form [1,1,1, -1,-1,-1, ...]

faces — [page:Array] of indices that make up the faces of the form

[0,1,2, 2,3,0, ...]

radius — [page:Float] - The radius of the final shape

detail — [page:Integer] - How many levels to subdivide the geometry.

The more detail, the smoother the shape.

Свойства

[property:Object parameters]

An object with all of the parameters that were used to generate the geometry.

Исходники

[link:[https://github.com/mrdoob/three.js/blob/master/src/\[path\].js](https://github.com/mrdoob/three.js/blob/master/src/[path].js)
src/[path].js]

[Geometry →](#)

PolyhedronGeometry

A polyhedron is a solid in three dimensions with flat faces. This class will take an array of vertices, project them onto a sphere, and then divide them up to the desired level of detail.

Пример

```
var verticesOfCube = [
    -1, -1, -1,      1, -1, -1,      1,  1, -1,      -1,  1, -1,
    -1, -1,  1,      1, -1,  1,      1,  1,  1,      -1,  1,  1,
];
var indicesOfFaces = [
    2,1,0,      0,3,2,
    0,4,7,      7,3,0,
    0,1,5,      5,4,0,
    1,2,6,      6,5,1,
    2,3,7,      7,6,2,
    4,5,6,      6,7,4
];
var geometry = new THREE.PolyhedronGeometry( verticesOfCube, indice
```

Конструктор

`PolyhedronGeometry([page:Array vertices], [page:Array faces], [page vertices — [page:Array] of points of the form [1,1,1, -1,-1,-1, ...] faces — [page:Array] of indices that make up the faces of the form [0,1,2, 2,3,0, ...]`

`radius — [page:Float] - The radius of the final shape`

`detail — [page:Integer] - How many levels to subdivide the geometry. The more detail, the smoother the shape.`

Свойства

`[property:Object parameters]`

An object with all of the parameters that were used to generate the geometry.

Исходники

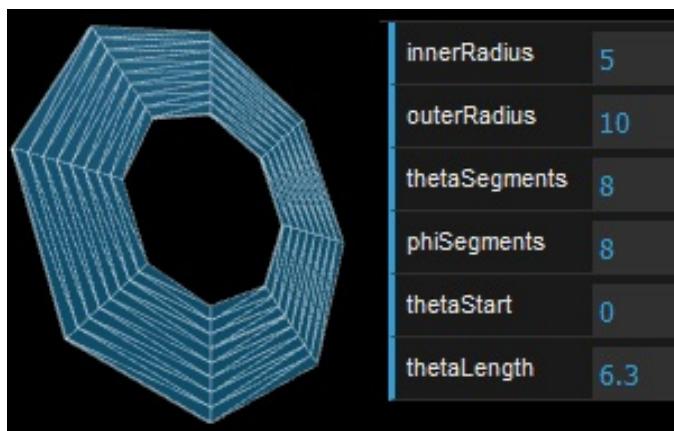
[PolyhedronGeometry.js в этом справочнике](#)

[PolyhedronGeometry.js на github.com](#)

[BufferGeometry →](#)

RingBufferGeometry

This is the [page:BufferGeometry] port of [page:RingGeometry].



Кольцо

Пример

```
var geometry = new THREE.RingBufferGeometry( 1, 5, 32 );
var material = new THREE.MeshBasicMaterial( { color: 0xffff00, side
var mesh = new THREE.Mesh( geometry, material );
scene.add( mesh );
```

Конструктор

`RingBufferGeometry(innerRadius, outerRadius, thetaSegments, phiSeg`

`innerRadius` — внутренний радиус кольца. Значение по умолчанию равно 20.

`outerRadius` — наружный радиус кольца. Значение по умолчанию равно 50.

`thetaSegments` — число сегментов по окружности кольца. Чем больше число сегментов, тем более круглым будет кольцо.

Минимальное значение равно 3. Значение по умолчанию равно 8.

`phiSegments` — число сегментов по радиусу кольца (в толщину).

Минимальное значение равно 1. Значение по умолчанию равно 8.

`thetaStart` — начальный угол. Значение по умолчанию равно 0.

`thetaLength` — центральный угол. Значение по умолчанию равно `Math.PI * 2`.

Исходники

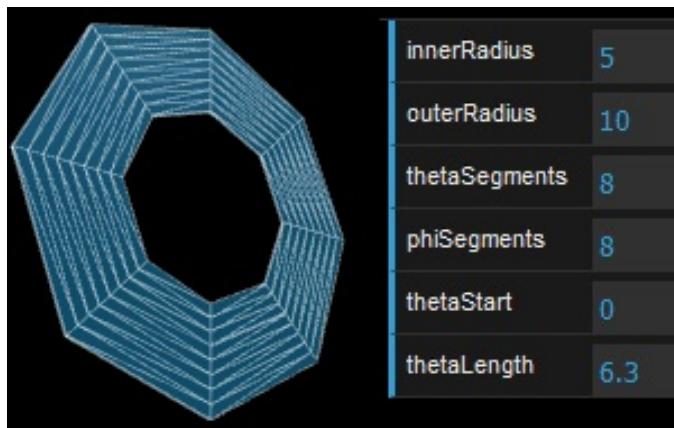
[RingBufferGeometry.js в этом справочнике](#)

[RingGeometry.js на github.com](#)

[Geometry →](#)

RingGeometry

Класс для построения двумерной кольцевой фигуры.



Кольцо

Пример

```
var geometry = new THREE.RingGeometry( 1, 5, 32 );
var material = new THREE.MeshBasicMaterial( { color: 0xffff00, side
var mesh = new THREE.Mesh( geometry, material );
scene.add( mesh );
```

Конструктор

```
RingGeometry( innerRadius, outerRadius, thetaSegments, phiSegments,
```

innerRadius — Default is 0, but it doesn't work right when innerRadius is set to 0.

outerRadius — Значение по умолчанию равно 50.

thetaSegments — Number of segments. A higher number means the ring will be more round. Minimum is 3. Значение по умолчанию равно 8.

phiSegments — Minimum is 1. Значение по умолчанию равно 8.

thetaStart — Начальный угол. Значение по умолчанию равно 0.

thetaLength — Центральный угол. Значение по умолчанию равно `Math.PI * 2`.

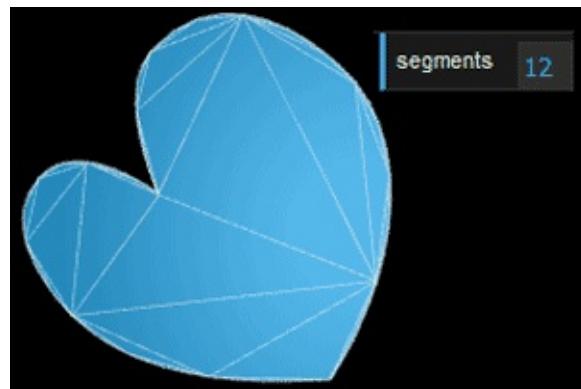
Исходники

[RingGeometry.js в этом справочнике](#)

[RingGeometry.js на github.com](#)

ShapeBufferGeometry

Creates a one-sided polygonal geometry from one or more path shapes.
Similar to [page:ExtrudeGeometry].



Геометрическая форма с числом сегментов по умолчанию, т.е. 12.

Пример

```
var length = 16, width = 12;  
  
var shape = new THREE.Shape();
```

```
shape.moveTo( 0, 0 );
shape.lineTo( 0, width );
shape.lineTo( length, width );
shape.lineTo( length, 0 );
shape.lineTo( 0, 0 );

var geometry = new THREE.ShapeGeometry( shape );
var material = new THREE.MeshBasicMaterial( { color: 0x00ff00 } );
var mesh = new THREE.Mesh( geometry, material ) ;
scene.add( mesh );
```

Конструктор

```
ShapeBufferGeometry([page:Array shapes], [page:Object options])
```

shapes — [page:Array] of shapes, or a single [page:Shape shape]

options — Optional options [page:Object object]

curveSegments - число сегментов в форме - значение по умолчанию равно 12

material - индекс материала из списка материалов

UVGenerator - A UV generator, defaults to

[page:ExtrudeGeometry]'s WorldUVGenerator

Методы

```
.addShapeList([page:Array shapes], [page:Object options]) [page:thi
```

shapes — [page:Array] of [page:Shape shapes]

options — See options in constructor

Adds a list of shapes to the geometry.

```
[method:null addShape]([page:Shape shape], [page:Object options])
```

shape — [page:Shape]

options — See options in constructor

Adds a single shape to the geometry

Исходники

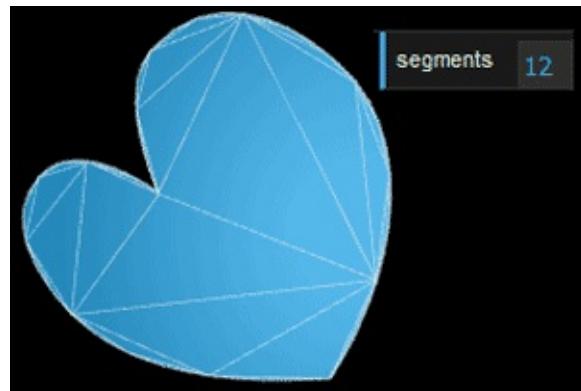
[link:[https://github.com/mrdoob/three.js/blob/master/src/\[path\].js](https://github.com/mrdoob/three.js/blob/master/src/[path].js)
src/[path].js]

[Geometry →](#)

ShapeGeometry

Creates a one-sided polygonal geometry from one or more path shapes.

Similar to [page:ExtrudeGeometry].



Геометрическая форма с числом сегментов по умолчанию, т.е. 12.

Пример

```
var length = 16, width = 12;

var shape = new THREE.Shape();
shape.moveTo( 0,0 );
shape.lineTo( 0, width );
shape.lineTo( length, width );
shape.lineTo( length, 0 );
shape.lineTo( 0, 0 );

var geometry = new THREE.ShapeGeometry( shape );
var material = new THREE.MeshBasicMaterial( { color: 0x00ff00 } );
var mesh = new THREE.Mesh( geometry, material ) ;
scene.add( mesh );
```

Конструктор

`ShapeGeometry([page:Array shapes], [page:Object options])`

`shapes` — [page:Array] of shapes, or a single [page:Shape shape]

`options` — Optional options [page:Object object]

`curveSegments` - число сегментов в форме - значение по умолчанию равно 12

`material` - индекс материала из списка материалов

`UVGenerator` - A UV generator, defaults to

[page:ExtrudeGeometry]'s WorldUVGenerator

Методы

`.addShapeList([page:Array shapes], [page:Object options]) [page:thi`

shapes — [page:Array] of [page:Shape shapes]

options — See options in constructor

Adds a list of shapes to the geometry.

```
[method:null addShape]([page:Shape shape], [page:Object options])
```

shape — [page:Shape]

options — See options in constructor

Adds a single shape to the geometry

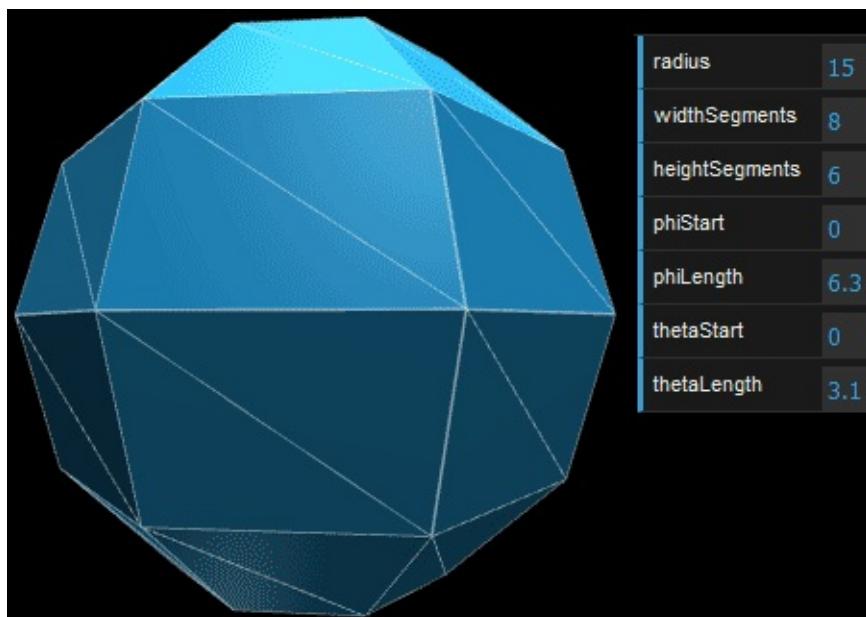
Исходники

[link:[https://github.com/mrdoob/three.js/blob/master/src/\[path\].js](https://github.com/mrdoob/three.js/blob/master/src/[path].js)
src/[path].js]

[BufferGeometry →](#)

SphereBufferGeometry

This is the [BufferGeometry](#) port of [SphereGeometry](#).



Сфера

Пример

```
var geometry = new THREE.SphereBufferGeometry( 5, 32, 32 );
var material = new THREE.MeshBasicMaterial( {color: 0xffff00} );
var sphere = new THREE.Mesh( geometry, material );
scene.add( sphere );
```

Конструктор

`SphereBufferGeometry(radius, widthSegments, heightSegments, phiStart)`

`radius` — радиус сферы, значение по умолчанию равно 50.

`widthSegments` — number of horizontal segments. Minimum value is 3, and the default is 8.

`heightSegments` — number of vertical segments. Minimum value is 2, and the default is 6.

`phiStart` — specify horizontal starting angle. Default is 0.

`phiLength` — specify horizontal sweep angle size. Default is `Math.PI * 2`.

`thetaStart` — specify vertical starting angle. Default is 0.

`thetaLength` — specify vertical sweep angle size. Default is `Math.PI`.

The geometry is created by sweeping and calculating vertexes around the Y axis (horizontal sweep) and the Z axis (vertical sweep). Thus, incomplete spheres (akin to '**sphere slices**') can be created through the use of different values of `phiStart`, `phiLength`, `thetaStart` and `thetaLength`, in order to define the points in which we start (or end) calculating those vertices.

Исходники

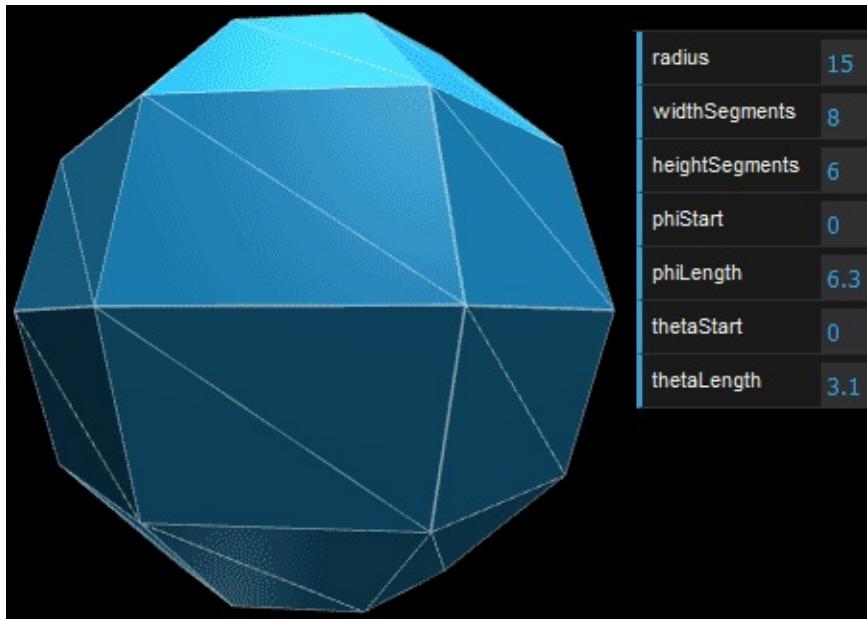
[SphereBufferGeometry.js в этом справочнике](#)

[SphereBufferGeometry.js на Гитхабе](#)

[Geometry →](#)

SphereGeometry (геометрия сферы)

Класс для генерации сферической геометрии.



Сфера

Пример

```
var geometry = new THREE.SphereGeometry( 5, 32, 32 );
var material = new THREE.MeshBasicMaterial( {color: 0xfffff00} );
var sphere = new THREE.Mesh( geometry, material );
scene.add( sphere );
```

Конструктор

`SphereGeometry(radius, widthSegments, heightSegments, phiStart, phiLength)`

`radius` — радиус сферы, значение по умолчанию равно 50.

`widthSegments` — количество горизонтальных сегментов.

Минимальное значение равно 3, значение по умолчанию 8.

`heightSegments` — количество вертикальных сегментов.

Минимальное значение равно 2, значение по умолчанию 6.

`phiStart` — устанавливает начальный угол по горизонтали

`specify horizontal starting angle`, значение по умолчанию равно 0.

`phiLength` — определяет размер угла ометания по горизонтали

`specify horizontal sweep angle size`, значение по умолчанию равно `Math.PI * 2`.

`thetaStart` — `specify vertical starting angle`, значение по умолчанию равно 0.

`thetaLength` — `specify vertical sweep angle size`, значение по умолчанию равно `Math.PI`.

Геометрия создается путем обхода и вычисления вершин вокруг оси Y (горизонтальный обход) и оси Z (вертикальный обход). Таким образом, неполные сферы (вроде '**ломтиков или кусков сферы**' - sphere slices) могут быть созданы через использование разных значений phiStart, phiLength, thetaStart и thetaLength, для определения точек, с которых начинается (или заканчивается) вычисление этих вершин. The geometry is created by sweeping and calculating vertexes around the Y axis (horizontal sweep) and the Z axis (vertical sweep). Thus, incomplete spheres (akin to '**sphere slices**') can be created through the use of different values of phiStart, phiLength, thetaStart and thetaLength, in order to define the points in which we start (or end) calculating those vertices.

Исходники

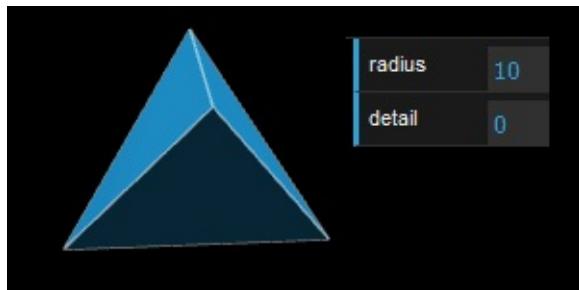
[SphereGeometry.js в этом справочнике](#)

[SphereGeometry.js на Гитхабе](#)

[PolyhedronBufferGeometry →](#)

TetrahedronBufferGeometry

Класс для создания геометрии тетраэдра (четырехгранника). A class for generating a tetrahedron geometries.



Tetrahedron

Конструктор

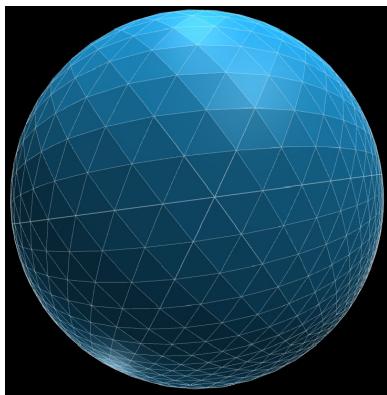
`TetrahedronBufferGeometry(radius, detail)`

`radius` — "Радиус" тетраэдра - скорее это длина ребра тетраэдра. По умолчанию равен 1.

`detail` — По умолчанию равен 0. Установка значения для этого параметра больше нуля добавит вершин и это будет уже не

тетраэдр.

Например, вот detail равен 5.



Свойства

.parameters

An object with all of the parameters that were used to generate the geometry.

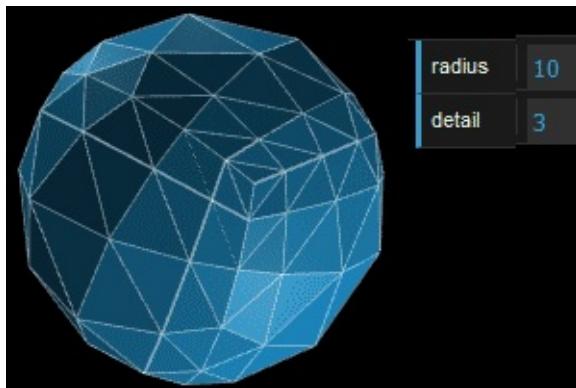
Исходники

[link:[https://github.com/mrdoob/three.js/blob/master/src/\[path\].js](https://github.com/mrdoob/three.js/blob/master/src/[path].js)
src/[path].js]

[Geometry →](#)

TetrahedronGeometry

A class for generating a tetrahedron geometries.



Tetrahedron

Конструктор

`TetrahedronGeometry([page:Float radius], [page:Integer detail])`

`radius` — Radius of the tetrahedron. Default is 1.

`detail` — Default is 0. Setting this to a value greater than 0 adds vertices making it no longer a tetrahedron.

Свойства

[property:Object `parameters`]

An object with all of the parameters that were used to generate the geometry.

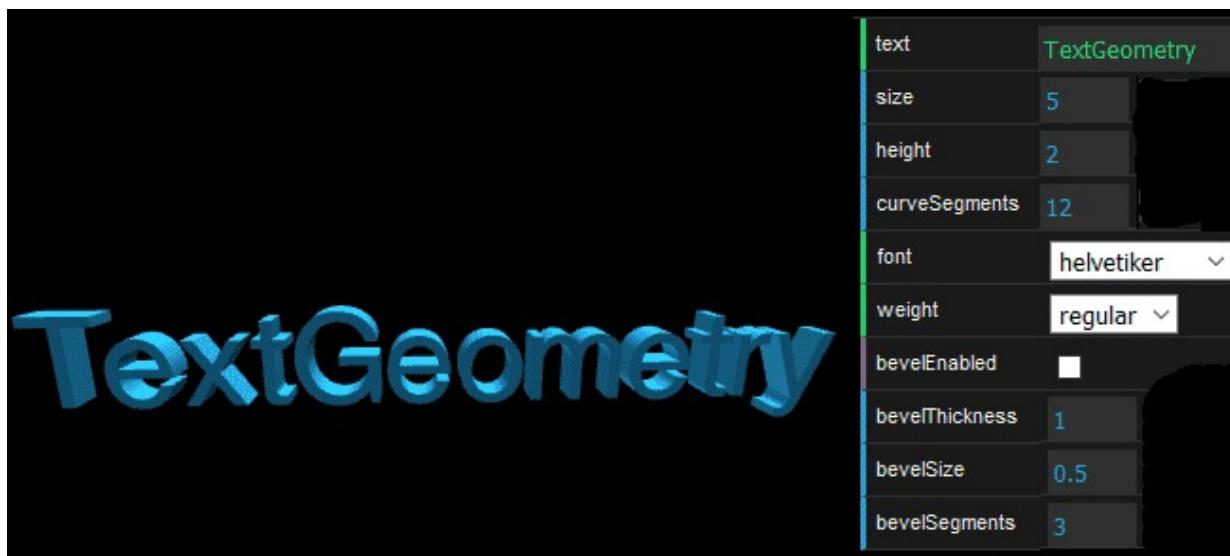
Исходники

[link:[https://github.com/mrdoob/three.js/blob/master/src/\[path\].js](https://github.com/mrdoob/three.js/blob/master/src/[path].js)
src/[path].js]

[page:ExtrudeBufferGeometry] →

TextBufferGeometry

A class for generating text as a single geometry. It is constructed by providing a string of text, and a hash of parameters consisting of a loaded [page:Font] and settings for the geometry's parent [page:ExtrudeBufferGeometry]. See the [page:Font], [page:FontLoader] and [page:Creating_Text] pages for additional details.



Text

Examples

```
[example:webgl_geometry_text geometry / text ]
[example:webgl_geometry_text2 geometry / text2 ]
var loader = new THREE.FontLoader();

loader.load( 'fonts/helvetiker_regular.typeface.json', function ( f

    var geometry = new THREE.TextBufferGeometry( 'Hello three.j
        font: font,
        size: 80,
        height: 5,
        curveSegments: 12,
        bevelEnabled: true,
        bevelThickness: 10,
        bevelSize: 8,
        bevelSegments: 5
    } );
} );
```

Конструктор

TextBufferGeometry([page:String text], [page:Object parameters])

text — The text that needs to be shown.

parameters — Object that can contains the following parameters.

- font — an instance of THREE.Font.
- size — Float. Size of the text. Default is 100.
- height — Float. Thickness to extrude text. Default is 50.
- curveSegments — Integer. Number of points on the curves. Default is 12.
- bevelEnabled — Boolean. Turn on bevel. Default is False.
- bevelThickness — Float. How deep into text bevel goes. Default is 10.
- bevelSize — Float. How far from text outline is bevel. Default is 8.
- bevelSegments — Integer. Number of bevel segments. Default is 3.

Available Fonts

TextGeometry uses [typeface.json](#) generated fonts. Some existing fonts can be found located in **/examples/fonts** and must be included in the page.

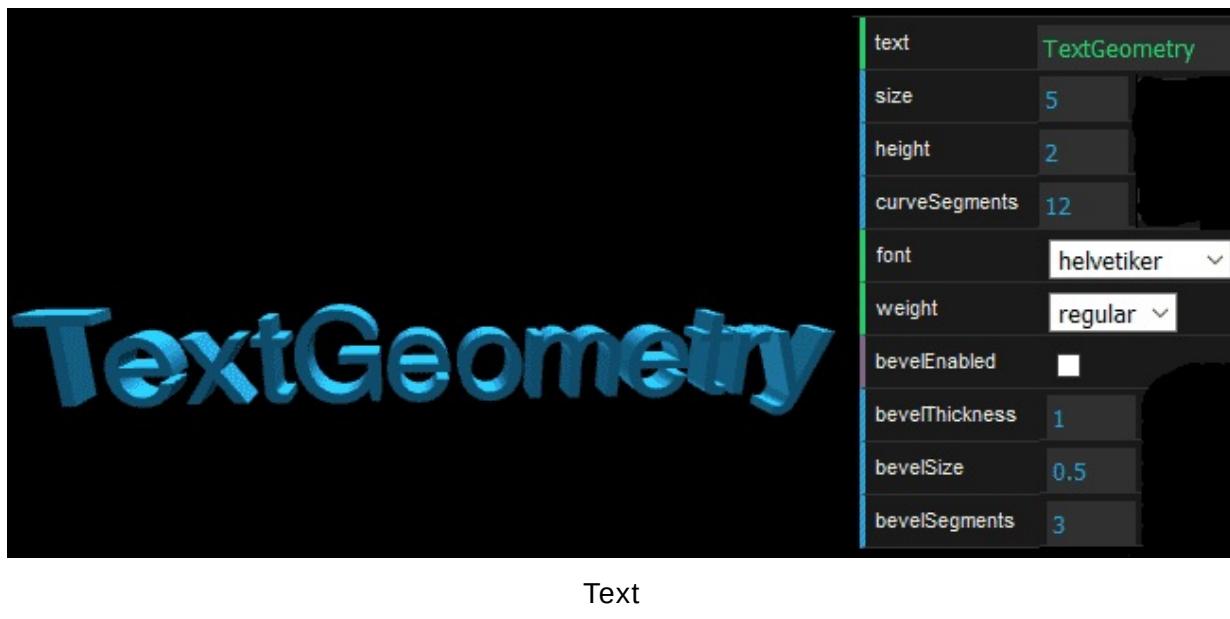
| Font | Weight | Style | File Path |
|-------------|--------|--------|---|
| helvetiker | normal | normal | /examples/fonts/helvetiker_regular.typeface.json |
| helvetiker | bold | normal | /examples/fonts/helvetiker_bold.typeface.json |
| optimer | normal | normal | /examples/fonts/optimer_regular.typeface.json |
| optimer | bold | normal | /examples/fonts/optimer_bold.typeface.json |
| gentilis | normal | normal | /examples/fonts/gentilis_regular.typeface.json |
| gentilis | bold | normal | /examples/fonts/gentilis_bold.typeface.json |
| droid sans | normal | normal | /examples/fonts/droid/droid_sans_regular.typeface.json |
| droid sans | bold | normal | /examples/fonts/droid/droid_sans_bold.typeface.json |
| droid serif | normal | normal | /examples/fonts/droid/droid_serif_regular.typeface.json |
| droid serif | bold | normal | /examples/fonts/droid/droid_serif_bold.typeface.json |

Source

[link:[https://github.com/mrdoob/three.js/blob/master/src/\[path\].js](https://github.com/mrdoob/three.js/blob/master/src/[path].js) src/[path].js] [page:ExtrudeGeometry] →

TextGeometry

This object creates a 3D object of text as a single object.



Text

Пример

[example:webgl_geometry_text geometry / text]
[example:webgl_geometry_text2 geometry / text2]

Конструктор

```
TextGeometry( text, parameters )
```

text — Текст, который нужно показать.

parameters — объект, который может содержать следующие параметры:

• font — THREE.Font.

• size — число с плавающей запятой. Size of the text.

• height — число с плавающей запятой. Thickness to extrude text. Значение по умолчанию равно 50.

- `curveSegments` — целое число. Number of points on the curves. Значение по умолчанию равно 12.
- `bevelEnabled` — логическое значение. Turn on bevel. Значением по умолчанию является `false`.
- `bevelThickness` — число с плавающей запятой. How deep into text bevel goes. Значение по умолчанию равно 10.
- `bevelSize` — число с плавающей запятой. How far from text outline is bevel. Значение по умолчанию равно 8.

Доступные шрифты

TextGeometry использует шрифты, сгенерированные [typeface.json](#).

Несколько имеющихся шрифтов можно найти в `/examples/fonts` и они должны быть включены в коде страницы.

| FONT ШРИФТ | WEIGHT ПЛОТНОСТЬ | STYLE СТИЛЬ | FILE PATH ПУТЬ К ФАЙЛУ |
|---------------|---------------------|----------------|---|
| helvetiker | normal | normal | /examples/fonts/helvetiker_regular.typeface.json |
| helvetiker | bold | normal | /examples/fonts/helvetiker_bold.typeface.json |
| optimer | normal | normal | /examples/fonts/optimer_regular.typeface.json |
| optimer | bold | normal | /examples/fonts/optimer_bold.typeface.json |
| gentilis | normal | normal | /examples/fonts/gentilis_regular.typeface.json |
| gentilis | bold | normal | /examples/fonts/gentilis_bold.typeface.json |
| droid sans | normal | normal | /examples/fonts/droid/droid_sans_regular.typeface.json |
| droid sans | bold | normal | /examples/fonts/droid/droid_sans_bold.typeface.json |
| droid serif | normal | normal | /examples/fonts/droid/droid_serif_regular.typeface.json |
| droid serif | bold | normal | /examples/fonts/droid/droid_serif_bold.typeface.json |

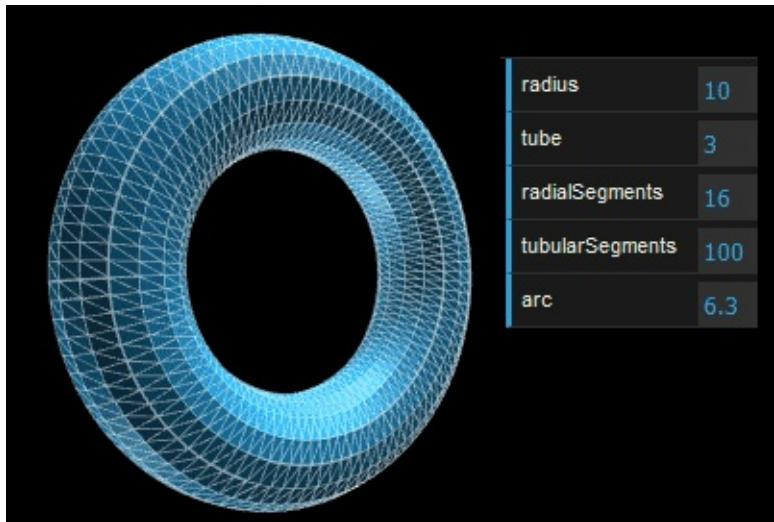
Исходники

[link:[https://github.com/mrdoob/three.js/blob/master/src/\[path\].js](https://github.com/mrdoob/three.js/blob/master/src/[path].js)
[src/\[path\].js](#)]

[BufferGeometry](#) →

TorusBufferGeometry

This is the [page:BufferGeometry] port of [page:TorusGeometry].



Top

Пример

```
var geometry = new THREE.TorusBufferGeometry( 10, 3, 16, 100 );
var material = new THREE.MeshBasicMaterial( { color: 0xffff00 } );
var torus = new THREE.Mesh( geometry, material );
scene.add( torus );
```

Конструктор

`TorusBufferGeometry(radius, tube, tubularSegments, radialSegments,`
radius — Default is 100.
tube — Diameter of the tube. Default is 40.
radialSegments — Default is 8
tubularSegments — Default is 6.
arc — Central angle. Default is `Math.PI * 2.`

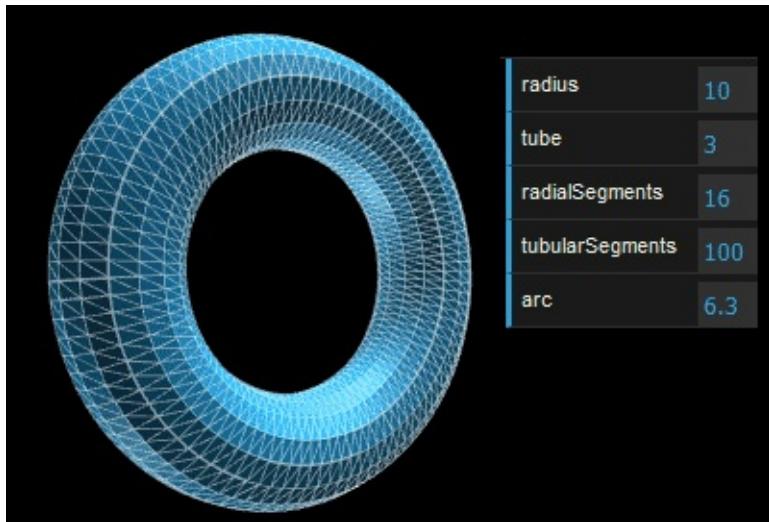
Исходники

[link:[https://github.com/mrdoob/three.js/blob/master/src/\[path\].js](https://github.com/mrdoob/three.js/blob/master/src/[path].js)
`src/[path].js`]

[Geometry →](#)

TorusGeometry

A class for generating torus geometries



Top

Пример

```
var geometry = new THREE.TorusGeometry( 10, 3, 16, 100 );
var material = new THREE.MeshBasicMaterial( { color: 0xffff00 } );
var torus = new THREE.Mesh( geometry, material );
scene.add( torus );
```

Конструктор

`TorusGeometry(radius, tube, tubularSegments, radialSegments, arc)`

`radius` — Default is 100.

`tube` — Diameter of the tube. Default is 40.

`radialSegments` — Default is 8

`tubularSegments` — Default is 6.

`arc` — Central angle. Default is `Math.PI * 2.`

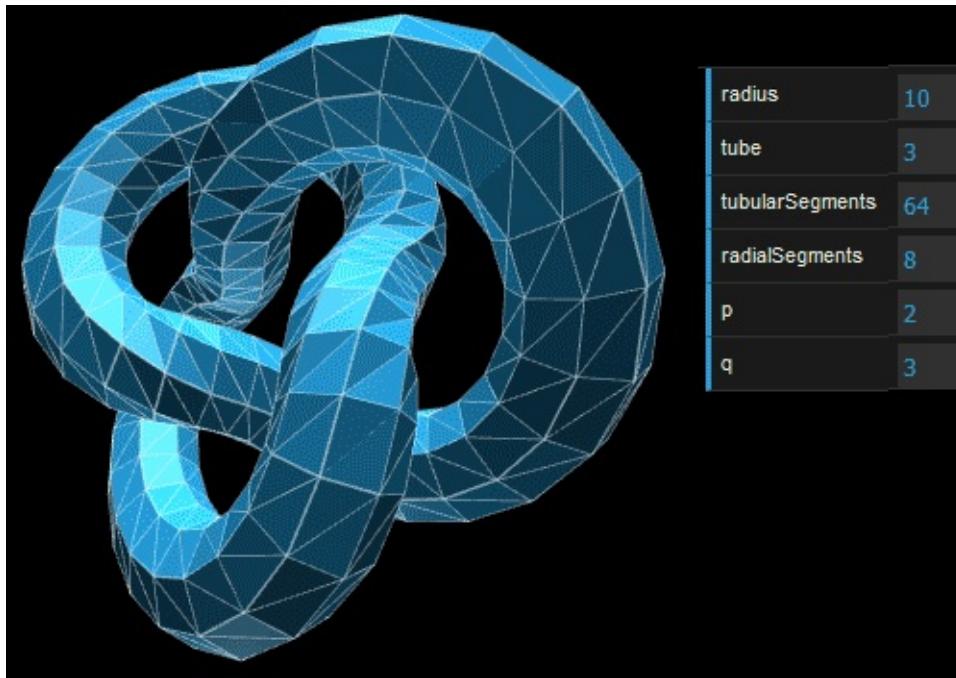
Исходники

[link:[https://github.com/mrdoob/three.js/blob/master/src/\[path\].js](https://github.com/mrdoob/three.js/blob/master/src/[path].js)
src/[path].js]

[BufferGeometry →](#)

TorusKnotBufferGeometry

This is the [page:BufferGeometry] port of [page:TorusKnotGeometry].



Кольцо с узлом

Пример

```
var geometry = new THREE.TorusKnotBufferGeometry( 10, 3, 100, 16 );
var material = new THREE.MeshBasicMaterial( { color: 0xffff00 } );
var torusKnot = new THREE.Mesh( geometry, material );
scene.add( torusKnot );
```

Конструктор

```
TorusKnotBufferGeometry( radius, tube, tubularSegments, radialSegme
```

- radius — Default is 100.
- tube — Diameter of the tube. Default is 40.
- tubularSegments — Default is 64.
- radialSegments — Default is 8.
- p — This value determines, how many times the geometry winds around its axis of rotational symmetry. Default is 2.
- q — This value determines, how many times the geometry winds around a circle in the interior of the torus. Default is 3.

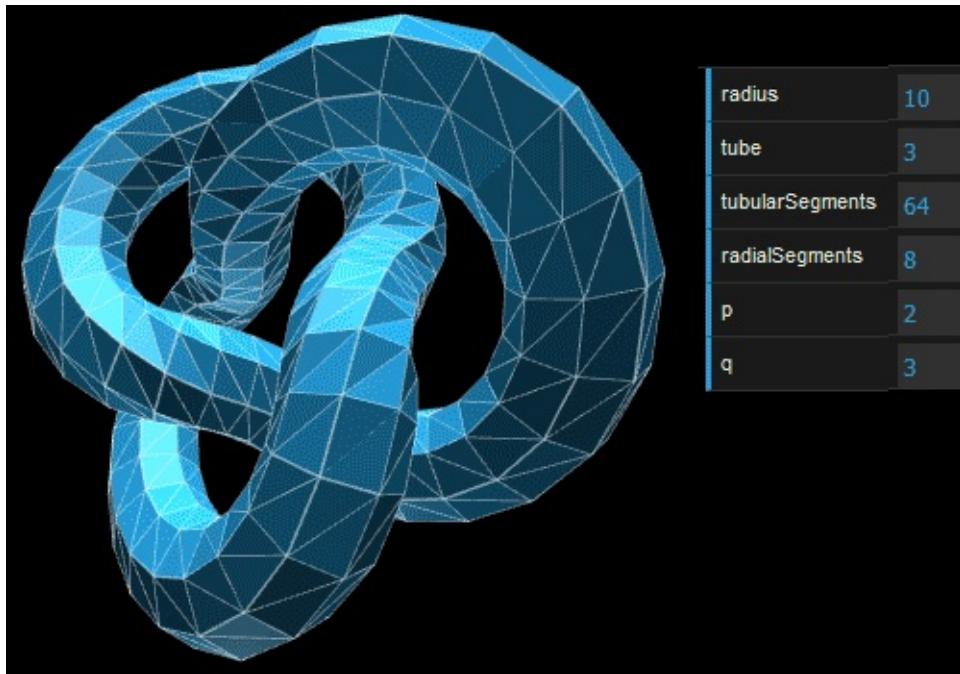
Исходники

[link:[https://github.com/mrdoob/three.js/blob/master/src/\[path\].js](https://github.com/mrdoob/three.js/blob/master/src/[path].js)
src/[path].js]

[Geometry →](#)

TorusKnotGeometry

Это геометрический объект в виде кольца с узлом, конкретная форма которого определяется парой [взаимно простых целых чисел](#), p и q . Если числа p и q не взаимно простые, получится Creates a torus knot, the particular shape of which is defined by a pair of coprime integers, p and q . If p and q are not coprime, the result will be a torus link.



Кольцо с узлом

Пример

```
var geometry = new THREE.TorusKnotGeometry( 10, 3, 100, 16 );
var material = new THREE.MeshBasicMaterial( { color: 0xffff00 } );
var torusKnot = new THREE.Mesh( geometry, material );
scene.add( torusKnot );
```

Конструктор

```
TorusKnotGeometry( radius, tube, tubularSegments, radialSegments, p
```

radius — радиус кольца (тора), значение по умолчанию равно 100.

tube — диаметр трубы, из которой сделано кольцо. Значение по умолчанию равно 40.

tubularSegments — число сегментов трубы, т.е на сколько

сегментов разделена труба по всей длине кольца. Значение по умолчанию равно 64.

`radialSegments` — число сегментов трубы по радиусу (т.е. при значении 3 - труба будет треугольной в сечении, при 4 - квадратной и т.д.) Значение по умолчанию равно 8.

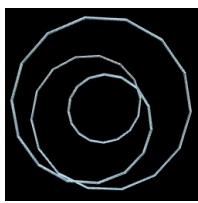
`p` — данное значение определяет, сколько оборотов делает труба вокруг центральной оси кольца. This value determines, how many times the geometry winds around its axis of rotational symmetry. Значение по умолчанию равно 2.



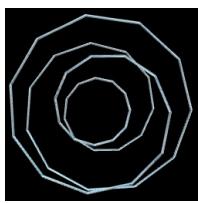
$p=1, q=1$



$p=2, q=1$



$p=3, q=1$

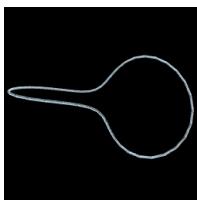


$p=4, q=1$

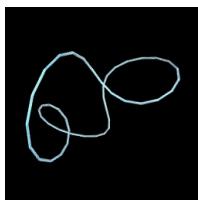
`q` — данное значение определяет, сколько оборотов сделает труба вдоль окружности кольца. This value determines, how many times the geometry winds around a circle in the interior of the torus. Значение по умолчанию равно 3.



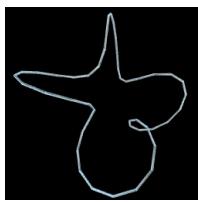
$p=1, q=1$



$p=1, q=2$



$p=1, q=3$



$p=1, q=4$

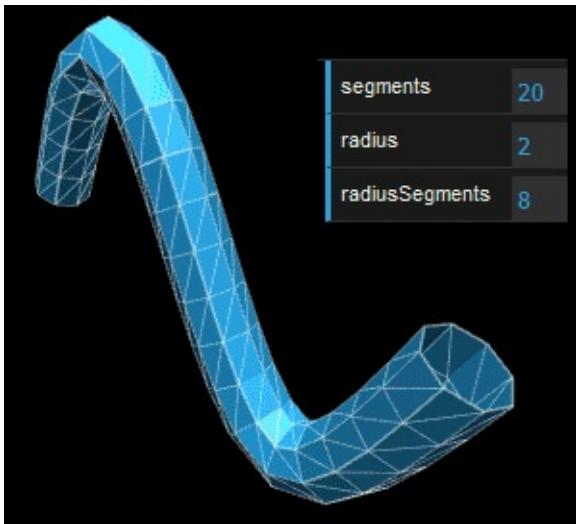
Исходники

[link:[https://github.com/mrdoob/three.js/blob/master/src/\[path\].js](https://github.com/mrdoob/three.js/blob/master/src/[path].js)
src/[path].js]

[BufferGeometry →](#)

TubeBufferGeometry

Creates a tube that extrudes along a 3d curve.



Труба

Пример

```
var CustomSinCurve = THREE.Curve.create(
    function ( scale ) { //custom curve constructor
        this.scale = ( scale === undefined ) ? 1 : scale;
    },
    function ( t ) { //getPoint: t is between 0-1
        var tx = t * 3 - 1.5;
        var ty = Math.sin( 2 * Math.PI * t );
        var tz = 0;
        return new THREE.Vector3( tx, ty, tz ).multiplyScal
    }
);

var path = new CustomSinCurve( 10 );
var geometry = new THREE.TubeBufferGeometry( path, 20, 2, 8, false
var material = new THREE.MeshBasicMaterial( { color: 0x00ff00 } );
var mesh = new THREE.Mesh( geometry, material );
scene.add( mesh );
```

Конструктор

`TubeBufferGeometry (path, tubularSegments, radius, radiusSegments,`
path — [page:Curve] - A path that inherits from the [page:Curve] base class

tubularSegments — [page:Integer] - The number of segments that make up the tube, default is 64

radius — [page:Float] - The radius of the tube, default is 1

radiusSegments — [page:Integer] - The number of segments that make

up the cross-section, default is 8
closed — [page:Boolean] Is the tube open or closed, default is false

Свойства

[property:Object parameters]

An object with all of the parameters that were used to generate the geometry.

[property:Array tangents]

An array of [page:Vector3] tangents

[property:Array normals]

An array of [page:Vector3] normals

[property:Array binormals]

An array of [page:Vector3] binormals

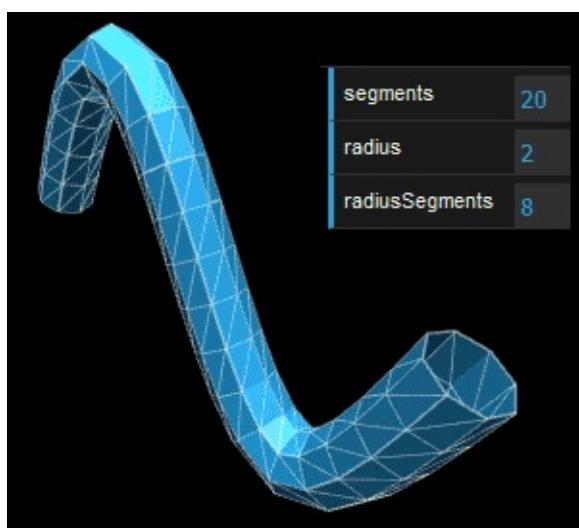
Исходники

[link:[https://github.com/mrdoob/three.js/blob/master/src/\[path\].js](https://github.com/mrdoob/three.js/blob/master/src/[path].js)
src/[path].js]

[Geometry →](#)

TubeGeometry

Creates a tube that extrudes along a 3d curve.



Труба

Пример

```
var CustomSinCurve = THREE.Curve.create(
```

```

        function ( scale ) { //custom curve constructor
            this.scale = ( scale === undefined ) ? 1 : scale;
        },
        function ( t ) { //getPoint: t is between 0-1
            var tx = t * 3 - 1.5;
            var ty = Math.sin( 2 * Math.PI * t );
            var tz = 0;
            return new THREE.Vector3( tx, ty, tz ).multiplyScalar( this.scale );
        }
    );

var path = new CustomSinCurve( 10 );
var geometry = new THREE.TubeGeometry( path, 20, 2, 8, false );
var material = new THREE.MeshBasicMaterial( { color: 0x00ff00 } );
var mesh = new THREE.Mesh( geometry, material );
scene.add( mesh );

```

Конструктор

[TubeGeometry\(path, tubularSegments, radius, radiusSegments, closed \)](#)

path — [page:Curve] - A path that inherits from the [page:Curve] base class

tubularSegments — [page:Integer] - The number of segments that make up the tube, default is 64

radius — [page:Float] - The radius of the tube, default is 1

radiusSegments — [page:Integer] - The number of segments that make up the cross-section, default is 8

closed — [page:Boolean] Is the tube open or closed, default is false

Свойства

[property:Object parameters]

An object with all of the parameters that were used to generate the geometry.

[property:Array tangents]

An array of [page:Vector3] tangents

[property:Array normals]

An array of [page:Vector3] normals

[property:Array binormals]

An array of [page:Vector3] binormals

Исходники

[TubeGeometry.js в этом справочнике](#)

[TubeGeometry.js на github.com](#)

[BufferGeometry →](#)

WireframeGeometry

Этот геометрический объект можно использовать в качестве вспомогательного для просмотра любого объекта [Geometry](#) в виде каркаса (wireframe).

Пример

```
var geometry = new THREE.SphereBufferGeometry( 100, 100, 100 );  
  
var wireframe = new THREE.WireframeGeometry( geometry );  
  
var line = new THREE.LineSegments( wireframe );  
line.material.depthTest = false;  
line.material.opacity = 0.25;  
line.material.transparent = true;  
  
scene.add( line );
```

[Другие примеры:](#)

[helpers](#)

Конструктор

`WireframeGeometry(geometry)`

| `geometry` — любой геометрический объект.

Исходники

[WireframeGeometry.js в этом справочнике](#)

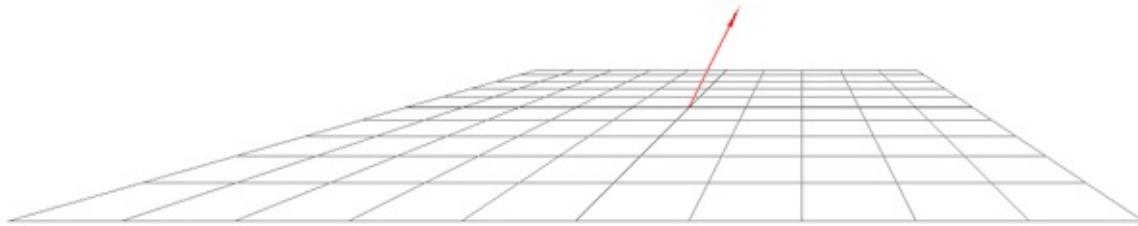
[WireframeGeometry.js на github.com](#)

ВСПОМОГАТЕЛЬНЫЕ ЭЛЕМЕНТЫ THREE.JS

ArrowHelper

[Object3D →](#)

Объект трехмерной (3D) стрелки для визуализации направлений.



Сетка и стрелка

Пример

```
var dir = new THREE.Vector3( 1, 2, 0 );

// normalize the direction vector (convert to vector of length 1)
// нормализуем вектор направления (конвертируем в вектор единичной
dir.normalize();

var origin = new THREE.Vector3( 0, 0, 0 );
var length = 1;
var hex = 0xffff00;

var arrowHelper = new THREE.ArrowHelper( dir, origin, length, hex )
scene.add( arrowHelper );
```

Другие примеры

[WebGL / geometries](#)
[WebGL / geometry / normals](#)
[WebGL / shadowmesh](#)

Конструктор

```
ArrowHelper(
  dir - сокращ. англ. слова direction - направление );" onmouseout="h
origin - начало, источник' );" onmouseout="hide()">origin, length, h
```

dir - сокращ. англ. слова direction - направление');" or
-- направление из origin (т.е из начальной точки стрелки).
Должен быть единичным вектором.

origin - начало, источник');" onmouseout="hide()">origir
-- точка из которой начинается стрелка.

length -- длина стрелки. Значением по умолчанию является 1.

hex -- шестнадцатиричное значение, определяющее цвет.
Значение по умолчанию равно 0xfffff00.

headLength -- длина наконечника стрелки. Значение по
умолчанию равно 0.2 * length.

headWidth -- ширина наконечника стрелки. Значение по
умолчанию равно 0.2 * headLength.

Свойства

Для информации об общих свойствах посмотрите базовый класс [Object3D](#).

.line

Содержит линейную часть arrowHelper.

.cone

Содержит коническую часть arrowHelper.

Методы

Для информации об общих методах посмотрите базовый класс [Object3D](#).

.setColor(hex)

hex -- шестнадцатиричное значение, определяющее цвет.

Устанавливает цвет arrowHelper.

.setLength(headLength, headWidth)

length -- желаемая длина стрелки.

headLength -- длина наконечника стрелки.

headWidth -- ширина наконечника стрелки.
Устанавливает длину arrowHelper.

```
.setDirection(  
dir - сокращ. англ. слова direction - направление');" onmouseout=  
| dir -- желаемое направление. Должен быть единичным  
вектором.
```

Устанавливает направление arrowHelper.

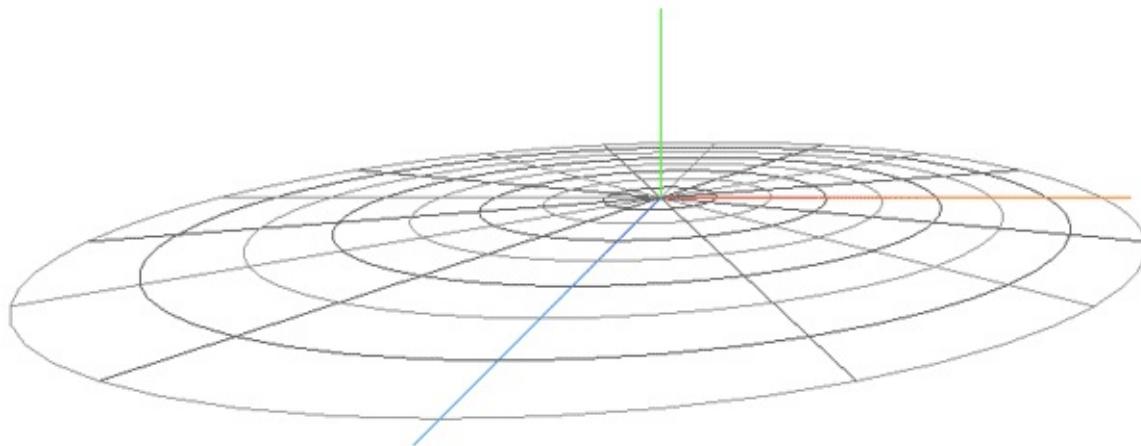
Исходники

[ArrowHelper.js на github.com](#)

[LineSegments](#) →

AxesHelper

Объект для визуализации трех осей координат простым способом.
Ось X красного цвета, ось Y - зеленая, ось Z - синяя.



Оси координат и полярная сетка

Пример

```
var axesHelper = new THREE.AxesHelper( 5 );  
scene.add( axesHelper );
```

Другие примеры

[WebGL / geometries](#)

[WebGL / geometries2](#)

[WebGL / geometry / convex](#)

[WebGL / geometry / spline / editor](#)

Конструктор

AxesHelper([size](#))

[size](#) -- длина линий, представляющих оси (дополнительный, необязательный параметр). Значение по умолчанию равно 1.

Свойства

Для информации об общих свойствах посмотрите базовый класс [LineSegments](#).

Методы

Для информации об общих методах посмотрите базовый класс [LineSegments](#).

Исходники

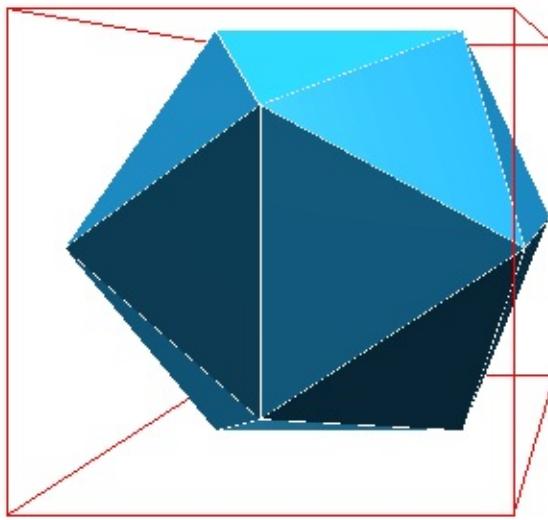
[AxesHelper.js на github.com](#)

[LineSegments →](#)

BoxHelper

Вспомогательный объект для отображения вокруг какого-либо объекта ограничительного каркаса из линий, ориентированного по осям мира.

Обратите внимание, для того чтобы это работало, объект должен иметь либо [Geometry](#), либо [BufferGeometry](#), так что со [спрайтами](#) он не будет работать.



BoxHelper вокруг икосаэдра

Пример

```
var sphere = new THREE.SphereGeometry();
var object = new THREE.Mesh( sphere, new THREE.MeshBasicMaterial( 0
var box = new THREE.BoxHelper( object, 0xffff00 );
scene.add( box );
```

Другие примеры

[WebGL / helpers](#)
[WebGL / loader / nrrd](#)
[advanced / buffergeometry / drawcalls](#)

Конструктор

`BoxHelper(object, color)`

object -- трехмерный (3D) объект, который нужно показать в ограничивающем его каркасе из линий, ориентированном по осям мира (дополнительный, необязательный параметр).

color -- шестнадцатиричное значение, определяющее цвет каркаса (сторон параллелипипеда) (дополнительный, необязательный параметр). Значение по умолчанию равно `0xffff00`.

Создает новый каркас, ограничивающий переданный объект. Внутри, для вычисления размеров, используется метод [`Box3.setFromObject`](#). Обратите внимание на то, что при этом в каркас включаются все

дочерние объекты.

Свойства

Для информации об общих свойствах посмотрите базовый класс [LineSegments](#).

Методы

Для информации об общих методах посмотрите базовый класс [LineSegments](#).

`.update()`

Обновляет геометрию вспомогательного объекта в соответствии с размерами объекта, включая любые дочерние объекты.

Смотрите описание метода [Box3.setFromObject](#).

`.setFromObject(object)`

`object` -- трехмерный (3D) объект, вокруг которого создается вспомогательный объект.

Обновляет каркас ограничивающего параллелипипеда для переданного объекта.

Исходники

[BoxHelper.js на github.com](#)

[LineSegments](#) →

CameraHelper

Этот вспомогательный объект для определения того, что попадает в объем усеченной пирамиды, который будет видим камере.

Он показывает эту область видимости камеры при помощи [LineSegments](#).

Пример

```
var camera = new THREE.PerspectiveCamera( 75, window.innerWidth / w  
var helper = new THREE.CameraHelper( camera );  
scene.add( helper );
```

Другие примеры

[WebGL / camera](#)

[WebGL / extrude / splines](#)

Конструктор

`CameraHelper(camera)`

| [camera](#) -- камера, область видимости которой, нужно показать.

Создает новый CameraHelper для указанной камеры.

Свойства

Для информации об общих свойствах посмотрите базовый класс [LineSegments](#).

[.camera](#)

Камера, область видимости которой нужно показать.

[.pointMap](#)

Здесь содержаться точки, используемые для визуализации области видимости камеры.

[.matrix](#)

Обратитесь к [camera.matrixWorld](#).

[.matrixAutoUpdate](#)

Смотрите описание свойства [Object3D.matrixAutoUpdate](#).

Установите здесь значение `false`, так как этот вспомогательный объект использует [matrixWorld](#) камеры.

Методы

Для информации об общих методах посмотрите базовый класс [LineSegments](#).

.update()

Метод обновляет вспомогательный объект, основываясь на `projectionMatrix` камеры.

Исходники

[CameraHelper.js на github.com](#)

[Object3D →](#)

DirectionalLightHelper

Вспомогательный объект для помощи с отображением на сцене эффекта `DirectionalLight`, т.е. направленного света.

Он состоит из плоскости и линии, представляющих положение и направление света.

Пример

```
var light = new THREE.DirectionalLight( 0xFFFFFF );
var helper = new THREE.DirectionalLightHelper( light, 5 );
scene.add( helper );
```

Конструктор

```
DirectionalLightHelper(  
  light - свет');" onmouseout="hide()">light, size, color )
```

`light - свет');` -- свет, который нужно показать.

`size` -- размер плоскости (дополнительный, необязательный параметр). Значение по умолчанию равно 1.

`color` -- если этот параметр не установлен, вспомогательный объект примет цвет освещения (дополнительный, необязательный параметр).

Свойства

Для информации об общих свойствах посмотрите базовый класс

Object3D.

[light_plane](#) - плоскость света');" onmouseout="hide()">>.lightPlane

Содержит сетку (mesh) линии, показывающую расположение направленного света.

[light](#) - свет');" onmouseout="hide()">>.light

Ссылка на отображаемый [directionalLight](#).

[.matrix](#)

Ссылка на [matrixWorld](#) освещения.

[.matrixAutoUpdate](#)

Смотрите описание [Object3D.matrixAutoUpdate](#). Установите здесь значение false, так как вспомогательный элемент использует [matrixWorld](#) освещения.

[.color](#)

Параметр цвета, переданный в конструктор. Значение по умолчанию неопределено. Если его не изменить, цвет вспомогательного объекта будет обновлен при следующем вызове обновления.

Методы

Для информации об общих методах посмотрите базовый класс [Object3D](#).

[.dispose\(\)](#)

Метод удаляет DirectionalLightHelper.

[.update\(\)](#)

Метод обновляет вспомогательный элемент в соответствии с позицией и направлением отображаемого [направленного света](#).

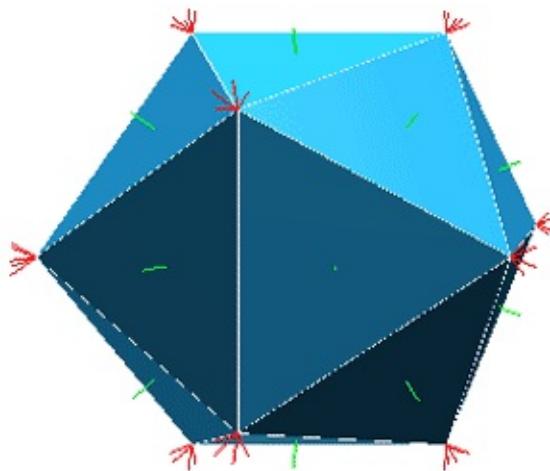
Исходники

[DirectionalLightHelper.js на github.com](#)

[LineSegments](#) →

FaceNormalsHelper

Показывает [стрелки](#) для визуализации нормалей [граней](#) объекта. Требуется, чтобы нормали граней были указаны для всех [граней](#) или были вычислены с помощью метода [computeFaceNormals](#). Обратите внимание, что этот вспомогательный объект работает только с объектами, чья геометрия является экземпляром [Geometry](#). Для [BufferGeometry](#) вместо него используйте [VertexNormalsHelper](#).



Нормали к вершинам (красные) и граням (зеленые)

Пример

```
geometry = new THREE.BoxGeometry( 10, 10, 10, 2, 2, 2 );
material = new THREE.MeshBasicMaterial( { color: 0xff0000 } );
box = new THREE.Mesh( geometry, material );

helper = new THREE.FaceNormalsHelper( box, 2, 0x00ff00, 1 );

scene.add( box );
scene.add( helper );
```

[Другие примеры](#)

[WebGL / helpers](#)

Конструктор

```
FaceNormalsHelper(  
  object - объект');" onmouseout="hide()">object, size, color, linewidth
```

object -- объект

для которого нужно отобразить нормали к граням.

size -- длина стрелок (дополнительный, необязательный параметр). Значение по умолчанию равно 1.

color -- шестнадцатиричное значение цвета стрелок (дополнительный, необязательный параметр). Значение по умолчанию равно 0xfffff00.

linewidth -- ширина линий стрелок (дополнительный, необязательный параметр). Значение по умолчанию равно 1.

Свойства

Для информации об общих свойствах посмотрите базовый класс [LineSegments](#).

.matrixAutoUpdate

Посмотрите описание [Object3D.matrixAutoUpdate](#). Установите данное свойство как false, так как этот вспомогательный элемент использует [matrixWorld](#) объекта.

.object

Объект, для которого должны быть показаны нормали граней.

.size

Длина стрелок, значение по умолчанию равно 1.

Методы

Для информации об общих методах посмотрите базовый класс [LineSegments](#).

.update()

Обновляет предпросмотр нормалей граней на основе перемещения объекта. Updates the face normal preview based on movement of the object.

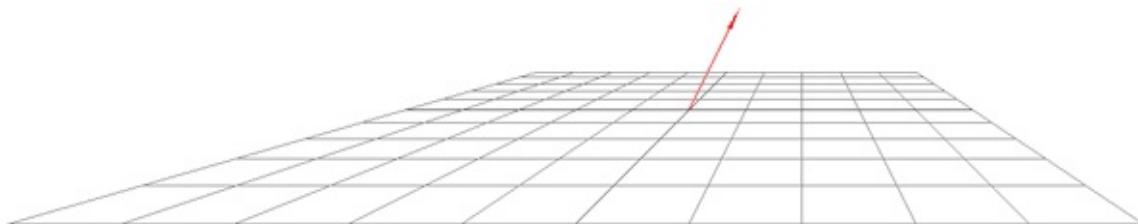
Исходники

[FaceNormalsHelper.js на github.com](#)

[Line →](#)

GridHelper

GridHelper - это объект для установки сеток. Сетка является двумерным массивом линий.



Сетка и стрелка

Пример

```
var size = 10;
var divisions = 10;

var gridHelper = new THREE.GridHelper( size, divisions );
scene.add( gridHelper );
```

Другие примеры

[Пример использования различных вспомогательных элементов](#)

Конструктор

```
GridHelper( size, divisions, colorCenterLine, colorGrid )
```

size -- размер стороны сетки. Значение по умолчанию равно 10.

divisions -- число ячеек на каждой стороне сетки. Значение по умолчанию равно 10.

colorCenterLine -- цвет осевой (центральной) линии. Это может

быть как шестнадцатиричное значение [цвета](#), так и название цвета в CSS. Значение по умолчанию равно 0x444444.

[colorGrid](#) -- цвет линий сетки. Это может быть как шестнадцатиричное значение [цвета](#), так и название цвета в CSS. Значение по умолчанию равно 0x888888.

Создает новый GridHelper размером size с divisions ячеек с каждой стороны. Цвета являются необязательными аргументами.

Исходники

[GridHelper.js на github.com](#)

[Object3D](#) →

HemisphereLightHelper

Создает видимый вспомогательный элемент, состоящий из сферической [сетки](#) для отображения зоны освещения от [HemisphereLight](#).

Пример

```
var light = new THREE.HemisphereLight( 0xffffbb, 0x080820, 1 );
var helper = new THREE.HemisphereLightHelper( light, 5 );
scene.add( helper );
```

Конструктор

[HemisphereLightHelper\(light, sphereSize, color \)](#)

[light](#) -- освещение, которое нужно визуализировать.

[sphereSize](#) -- Размер сетки, используемой для визуализации освещения.

[color](#) -- дополнительный, необязательный параметр, если он не установлен, вспомогательный элемент принимает цвет освещения.

Свойства

Для информации об общих свойствах посмотрите базовый класс

[Object3D](#).

.light

Reference to the HemisphereLight being visualized.

.matrix

Reference to the hemisphereLight's [page:Object3D.matrixWorld matrixWorld].

.matrixAutoUpdate

Смотрите свойство [Object3D.matrixAutoUpdate](#). Set to *false* here as the helper is using the hemisphereLight's [page:Object3D.matrixWorld matrixWorld].

.color

Параметр цвета, переданный в конструктор. По умолчанию значение равно undefined, т.е. неопределено. Если его изменить, цвет вспомогательного элемента будет обновлен при следующем вызове обновления.

Методы

Для информации об общих методах посмотрите базовый класс [Object3D](#).

.dispose()

Метод удаляет HemisphereLightHelper.

.update()

Updates the helper to match the position and direction of the [page:.light].

Исходники

[HemisphereLightHelper.js на github.com](#)

PointLightHelper

Создает видимый вспомогательный элемент, состоящий из сферической [сетки](#) для отображения зоны освещения от [PointLight](#) - точечного освещения.

Пример

```
var pointLight = new THREE.PointLight( 0xff0000, 1, 100 );
pointLight.position.set( 10, 10, 10 );
scene.add( pointLight );

var sphereSize = 1;
var pointLightHelper = new THREE.PointLightHelper( pointLight, sphereSize );
scene.add( pointLightHelper );
```

Другие примеры

[WebGL / helpers](#)

Конструктор

```
PointLightHelper(  
    light - свет');" onmouseout="hide()">light, sphereSize, color )
```

light - свет');" onmouseout="hide()">light -- освещение, которое нужно визуализировать.

sphereSize -- размер сферы вспомогательного элемента.

Значение по умолчанию равно 1.

color -- дополнительный, необязательный параметр, если он не установлен, вспомогательный элемент принимает цвет освещения.

Свойства

Для информации об общих свойствах посмотрите базовый класс [Mesh](#).

```
light - свет');" onmouseout="hide()">.light
```

Точечное освещение ([PointLight](#)), которое нужно визуализировать.

[.matrix](#)

Ссылка на [matrixWorld](#), отображаемого точечного освещения.

[.matrixAutoUpdate](#)

Смотрите описание [Object3D.matrixAutoUpdate](#). Установите здесь значение `false`, так как вспомогательный элемент использует [matrixWorld](#) точечного освещения.

[.color](#)

Параметр цвета, переданный в конструктор. По умолчанию значение равно `undefined`, т.е. неопределено. Если его изменить, цвет вспомогательного элемента будет обновлен при следующем вызове обновления.

Методы

Для информации об общих свойствах посмотрите базовый класс [Mesh](#).

[.dispose\(\)](#)

Метод удаляет PointLightHelper.

[.update\(\)](#)

Метод обновляет вспомогательный элемент в соответствии с позицией [.light](#).

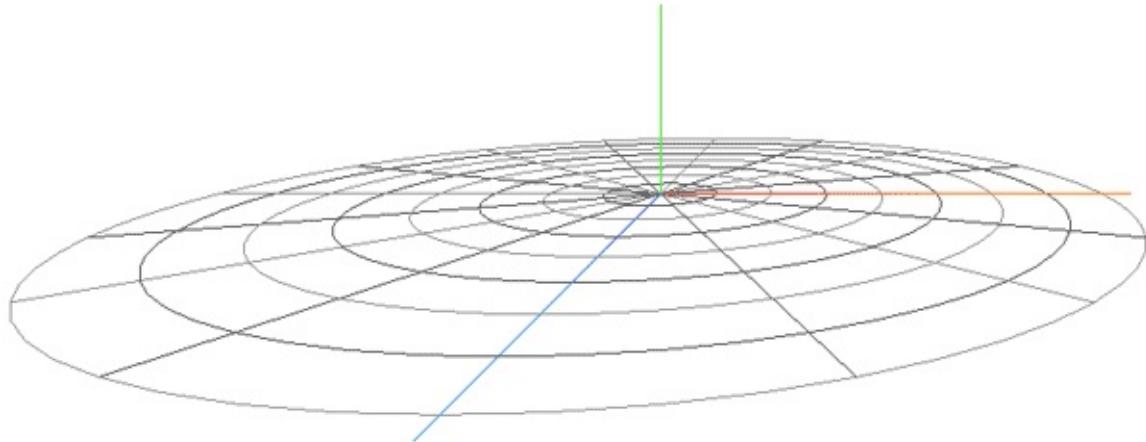
Исходники

[PointLightHelper.js на github.com](#)

[Line ↗](#)

PolarGridHelper

PolarGridHelper - это объект для установки сеток. Сетка является двумерным массивом линий.



Оси координат и полярная сетка

Пример

```
var radius = 10;
var radials = 16;
var circles = 8;
var divisions = 64;

var helper = new THREE.PolarGridHelper( radius, radials, circles, divisions );
scene.add( helper );
```

Другие примеры

[Пример использования разных вспомогательных элементов](#)

Конструктор

```
PolarGridHelper( radius, radials, circles, divisions, color1, color2 )
```

radius -- радиус полярной сетки. Он может быть любым положительным числом. Значение по умолчанию равно 10.

radials -- число радиальных линий. Оно может быть любым положительным целым числом. Значение по умолчанию равно 16.

circles -- число окружностей. Это может быть любое положительное целое число. Значение по умолчанию равно 8.

divisions -- число отрезков линии, из которых составлена каждая из окружностей. Это может быть любое

положительное целое число, равное 3 или большее. Значение по умолчанию равно 64.

Примечание переводчика: То есть, при значении divisions равному 3 окружности будут треугольниками, при 8 - восьмиугольниками. С увеличением значения divisions, окружности становятся все более "круглыми".

color1 -- Первый цвет, используемый для элементов сетки. Это может быть как шестнадцатиричное значение [цвета](#), так и название цвета в CSS. Значение по умолчанию равно 0x444444.

color2 -- Второй цвет, используемый для элементов сетки. Это может быть как шестнадцатиричное значение [цвета](#), так и название цвета в CSS. Значение по умолчанию равно 0x888888.

Создает новый PolarGridHelper с радиусом, определенным в параметре radius, числом лучей (радиалов), указанных в параметре radials, и числом окружностей, переданных в circles, где каждая окружность состоит из отрезков, число которых указано в divisions. Аргументы со значениями цвета являются дополнительными, необязательными.

Исходники

[PolarGridHelper.js на github.com](#)

[Object3D](#) →

RectAreaLightHelper

Создает видимый вспомогательный элемент для отображения зоны освещения от [RectAreaLight](#).

Пример

```
var light = new THREE.RectAreaLight( 0xffffbb, 1.0, 5, 5 );
var helper = new THREE.RectAreaLightHelper( light );
scene.add( helper );
```

Конструктор

`RectAreaLightHelper(light, color)`

| light -- освещение, которое нужно визуализировать.

`color` -- дополнительный, необязательный параметр, если он не установлен, вспомогательный элемент принимает цвет освещения.

Свойства

Для информации об общих свойствах посмотрите базовый класс [Object3D](#).

`.light`

Ссылка на [RectAreaLight](#), который нужно визуализировать.

`.color`

The color parameter passed in the constructor. Default is undefined. If this is changed, the helper's color will update the next time update is called.

Методы

Для информации об общих методах посмотрите базовый класс [Object3D](#).

`.dispose()`

Метод удаляет RectAreaLightHelper.

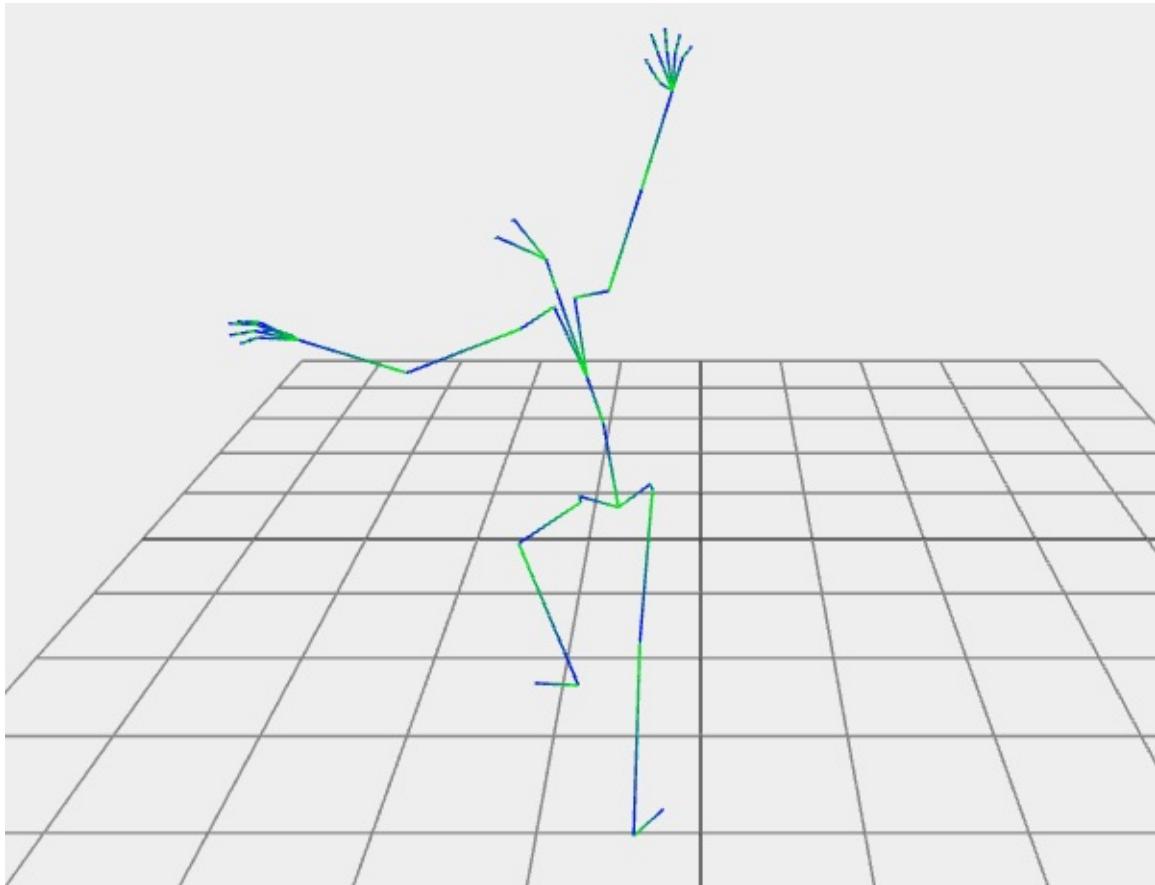
`[method:null update()]`

Updates the helper to match the position and direction of the [page:.light].

Исходники

[RectAreaLightHelper.js на github.com](#)

Вспомогательный объект для помощи с визуализацией объекта [Skeleton](#). Визуализация производится при помощи [LineBasicMaterial](#).



Вспомогательный элемент для скелета

Пример

```
var helper = new THREE.SkeletonHelper( mesh );
helper.material.linewidth = 3;
scene.add( helper );
```

Другие примеры

[animation / skinning / blending](#)
[animation / skinning / morph](#)
[loader / bvh](#)

Конструктор

`SkeletonHelper(object)`

| `object` -- может быть любым объектом, имеющим в качестве

вложенного объекта, массив [page:Bone Bone]s.
Например, [Skeleton](#) или [SkinnedMesh](#).

Свойства

[.bones](#)

Список костей, которые вспомогательный элемент представляет как [линии](#).

[.root](#)

Объект, переданный в конструктор.

Исходники

[SkeletonHelper.js на github.com](#)

[Object3D](#) →

SpotLightHelper

Показывает вспомогательный объект в виде конуса для [SpotLight](#).

Пример

```
var spotLight = new THREE.SpotLight( 0xffffff );
spotLight.position.set( 10, 10, 10 );
scene.add( spotLight );

var spotLightHelper = new THREE.SpotLightHelper( spotLight );
scene.add( spotLightHelper );
```

Другие примеры

[lights / spotlights](#)

Конструктор

```
SpotLightHelper(  
  light - освещение );" onmouseout="hide()">light, color )
```

`light - освещение');` `onmouseout="hide()">light` --

`SpotLight`, конус света которого нужно показать.

`color` -- дополнительный, необязательный параметр, если он не установлен, вспомогательный элемент принимает цвет освещения.

Свойства

Для информации об общих свойствах посмотрите базовый класс [Object3D](#).

`[property:LineSegments .cone`

`[page:LineSegments] used to visualize the light.`

`[property:SpotLight .light`

`Reference to the [page:SpotLight] being visualized.`

`[property:object .matrix`

`Reference to the spotLight's [page:Object3D.matrixWorld
matrixWorld].`

`[property:object .matrixAutoUpdate`

`See [page:Object3D.matrixAutoUpdate]. Set to *false* here as the
helper is using the spotLight's [page:Object3D.matrixWorld
matrixWorld].`

Методы

Для информации об общих методах посмотрите базовый класс [Object3D](#).

`.dispose()`

Метод удаляет SpotLightHelper.

`.update()`

Обновление вспомогательного элемента для прожекторного освещения.

Исходники

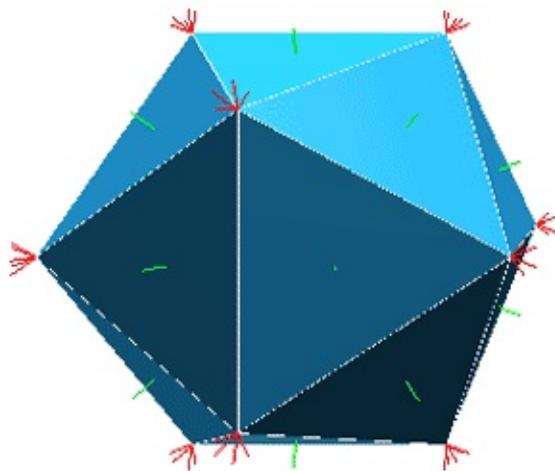
[SpotLightHelper.js на github.com](#)

[Line →](#)

VertexNormalsHelper

Отображает [стрелки](#) для визуализации векторов нормалей к вершинам объекта. Требуется, чтобы нормали были указаны в [custom attribute](#) или были вычислены с помощью метода [computeVertexNormals](#).

В отличие от [FaceNormalsHelper](#), данный класс работает с [BufferGeometry](#).



Нормали к вершинам (красные) и граням (зеленые)

Пример

```
var geometry = new THREE.BoxGeometry( 10, 10, 10, 2, 2, 2 );
var material = new THREE.MeshBasicMaterial( { color: 0xff0000 } );
var box = new THREE.Mesh( geometry, material );

var helper = new THREE.VertexNormalsHelper( box, 2, 0x00ff00, 1 );

scene.add( box );
scene.add( helper );
```

[Другие примеры](#)

[WebGL / helpers](#)

Конструктор

`VertexNormalsHelper(object, size, color, linewidth)`

[object](#) -- объект, для которого нужно показать нормали вершин.

[size](#) -- длина стрелок (дополнительный, необязательный параметр). Значение по умолчанию равно 1.

[color](#) -- шестнадцатиричное значение цвета стрелок. Значение по умолчанию равно 0xff0000.

[linewidth](#) -- ширина линий стрелок (дополнительный, необязательный параметр). Значение по умолчанию равно 1.

Свойства

Для информации об общих свойствах посмотрите базовый класс [LineSegments](#).

[.matrixAutoUpdate](#)

Смотрите [Object3D.matrixAutoUpdate](#). Установите его как false, так как данный вспомогательный элемент использует [matrixWorld](#) объекта.

[.object](#)

Объект, для которого нужно показать нормали вершин.

[.size](#)

Длина стрелок. Значение по умолчанию равно 1.

Методы

Для информации об общих методах посмотрите базовый класс [LineSegments](#).

[.update\(\)](#)

Обновляет представление нормалей вершин, основываясь на перемещении объекта.

Исходники

[VertexNormalsHelper.js на github.com](#)

ОСВЕЩЕНИЕ

В этой части рассматриваются различные виды освещения и образование тени, при некоторых способах освещения.

Для понимания процесса освещения в компьютерной графике посмотрите следующие статьи:

[learnopengl. Урок 2.2 — Основы освещения](#)

[Продолжаем изучение OpenGL: освещение по Фонгу](#)

[Затенение по Фонгу](#)

[Object3D](#) → [Light](#) →

[AmbientLight](#)

Фоновое освещение глобально и в равной степени освещает все объекты на сцене. Этот свет не может использоваться для создания теней, так как не имеет направления.

Пример

```
var light = new THREE.AmbientLight( 0x404040 ); // soft white light
scene.add( light );
```

Другие примеры:

[camera / orthographic](#)
[interactive / voxelpainter](#)
[materials](#)
[sandbox](#)
[animation / cloth](#)
[animation / skinning / blending](#)

Конструктор

```
AmbientLight( color, intensity )
```

[color](#) — числовое значение RGB компонентов цвета.

[intensity](#) — числовое значение силы/интенсивности света.

Конструктор создает окружающее (общее) освещение с заданными цветом и интенсивностью.

Свойства

Смотрите базовый класс [Light](#) для общих свойств.

[.castShadow](#)

В конструкторе значение этого свойства установлено как `undefined` - неопределено, поскольку фоновое освещение не может отбрасывать тени.

[.isAmbientLight](#)

Свойство используется для проверки, является ли данный и производные от него классы фоновым освещением. Значением по умолчанию является `true`.

Его нельзя изменять, так как оно используется внутри `three.js` для оптимизации.

Методы

Смотрите базовый класс [Light](#) для общих методов.

Исходники

[AmbientLight.js на github.com](#)

[Object3D](#) → [Light](#) →

[DirectionalLight](#)

Свет, который испускается в определенном направлении. Этот свет будет вести себя так, как будто он бесконечно удален, и лучи, полученные из него, все параллельны. Общий пример

использования - имитация дневного света; солнце достаточно далеко, так что расстояние до него можно считать бесконечным, а все лучи света, исходящие от него, параллельными.

Это освещение может отбрасывать тени - подробностисмотрите в статье [DirectionalLightShadow](#)

Примечание относительно положения, цели и вращения

Общим местом недопонимания с направленным освещением является то, что настройки поворота не оказывают влияния. Это происходит из-за того, что `DirectionalLight` в `three.js` равнозначен тому, что в других приложениях зачастую называется Target Direct Light - целенаправленный (адресно направленный) свет.

Это означает, что его направление вычисляется как вектор от места расположения источника освещения к месту, где находится цель (в отличие от Free Direct Light, у которого имеется только компонент вращения).

Примечание переводчика: То есть, если направление Free Direct Light определяется углом в трехмерном пространстве, то `DirectionalLight` направлен на конкретный объект - цель.

Это сделано для того, чтобы у данного освещения была возможность отбрасывать тени - теневой камере требуется позиция, от которой рассчитываются тени.

Смотрите ниже описание свойства [target](#) для подробностей по обновлению цели.

Пример

```
// White directional light at half intensity shining from the top.  
// Направленное освещение белого цвета в половину интенсивности, све  
  
var directionalLight = new THREE.DirectionalLight( 0xffffff, 0.5 );  
directionalLight.position.set( 0, 1, 0 );  
scene.add( directionalLight );
```

Другие примеры

[morphtargets / horse](#)

```
controls / fly
lights / test
cubes
effects / parallaxbarrier
effects / stereo
geometry / extrude / splines
materials / bumpmap
materials / cubemap / balls / reflection
```

Конструктор

```
DirectionalLight( hex, intensity )
```

[hex](#) — шестнадцатиричное значение цвета освещения (дополнительный, необязательный параметр). Значение по умолчанию равно 0xffffffff (белый).

[intensity](#) — числовое значение силы/интенсивности освещения (дополнительный, необязательный параметр). Значение по умолчанию равно 1.

Создает новый DirectionalLight

Свойства

Смотрите базовый класс [Light](#) для общих свойств.

```
.castShadow
```

Если установлено значение true, освещение будет отбрасывать динамические (движущиеся при перемещении объекта-цели) тени.

Предупреждение: Это довольно затратно и требует настройки для получения правильного отображения теней. Подробнее смотрите в описании [DirectionalLightShadow](#).

Значением по умолчанию является false.

```
.isDirectionalLight
```

Свойство используется для проверки, является ли данный класс и все производные от него классы, направленным освещением.

Значением по умолчанию является true.

Это свойство нельзя изменять, так как оно используется внутри **three.js** для оптимизации.

position - положение, позиция');" onmouseout="hide()">.position

Это свойство устанавливается равным

Object3D.DefaultUp (0, 1, 0), так что свет светит сверху вниз.

shadow - тень');" onmouseout="hide()">.shadow

Свойство используется в DirectionalLightShadow для расчета теней от данного освещения.

target - цель, мишень, назначение');" onmouseout="hide()">.target

Со своей позицией DirectionalLight направлен на позицию объекта-цели (target.position), которую и определяет данное свойство.

Позицией цели по умолчанию является (0, 0, 0).

Примечание: Чтобы изменить позицию объекта-цели на что-либо другое, кроме значения по умолчанию, он должен быть добавлен на сцену при помощи

```
scene.add( light.target );
```

Так что matrixWorld цели станет автоматически обновляться с каждым кадром.

Также возможно установить в качестве цели другой объект на сцене (что-нибудь со свойством position), вот так:

```
var targetObject = new THREE.Object3D();
scene.add(targetObject);
```

```
light.target = targetObject;
```

Теперь directionalLight будет отслеживать целевой объект.

Методы

Смотрите базовый класс Light для общих методов.

```
copy - копировать');" onmouseout="hide()">>.copy(  
source - источник');" onmouseout="hide()">>source )
```

Копирует значение всех свойств из source в данный объект DirectionalLight.

Исходники

[DirectionalLight.js на github.com](#)

[LightShadow →](#)

DirectionalLightShadow

Этот класс используется внутри для расчета теней от [DirectionalLight](#). В отличие от других классов тени, данный класс для вычисления теней использует [OrthographicCamera](#), а не [PerspectiveCamera](#). Это делается потому, что лучи света от [DirectionalLight](#) параллельны.

Пример

```
// Create a WebGLRenderer and turn on shadows in the renderer  
// создаем WebGLRenderer и включаем тени в визуализаторе  
var renderer = new THREE.WebGLRenderer();  
renderer.shadowMap.enabled = true;  
renderer.shadowMap.type = THREE.PCFSoftShadowMap; // default THREE.  
// (по умолчанию)  
  
// Create a DirectionalLight and turn on shadows for the light  
// создаем DirectionalLight и включаем тени от него  
var light = new THREE.DirectionalLight( 0xffffff, 1, 100 );  
light.position.set( 0, 1, 0 ); // default; light shining from top  
// (по умолчанию; свет светит сверху)  
light.castShadow = true; // default false (по умолчанию - fal  
scene.add( light );  
  
// Set up shadow properties for the light  
// установка свойств тени для этого освещения  
light.shadow.mapSize.width = 512; // default (значение по умолчани  
light.shadow.mapSize.height = 512; // default (значение по умолчани  
light.shadow.camera.near = 0.5; // default (значение по умолчани
```

```
light.shadow.camera.far = 500      // default (значение по умолчанию)

// Create a sphere that cast shadows (but does not receive them)
// создаем сферу, отбрасывающую тени (но не принимающую их)
var sphereGeometry = new THREE.SphereBufferGeometry( 5, 32, 32 );
var sphereMaterial = new THREE.MeshStandardMaterial( { color: 0xffff00 } );
var sphere = new THREE.Mesh( sphereGeometry, sphereMaterial );
sphere.castShadow = true;        // default is false (по умолчанию - false)
sphere.receiveShadow = false;    // default (значение по умолчанию)
scene.add( sphere );

// Create a plane that receives shadows (but does not cast them)
// Создаем плоскость, принимающую тени (но не отбрасывающую их)
var planeGeometry = new THREE.PlaneBufferGeometry( 20, 20, 32, 32 );
var planeMaterial = new THREE.MeshStandardMaterial( { color: 0x00ffff } );
var plane = new THREE.Mesh( planeGeometry, planeMaterial );
plane.receiveShadow = true;
scene.add( plane );

// Create a helper for the shadow camera (optional)
// создаем вспомогательный элемент для камеры теней (необязательно)
var helper = new THREE.CameraHelper( light.shadow.camera );
scene.add( helper );
```

Конструктор

`DirectionalLightShadow()`

Создает новый `DirectionalLightShadow`. Конструктор не предназначен для непосредственного вызова - он вызывается внутри `three.js`, посредством [DirectionalLight](#).

Свойства

Смотрите базовый класс [LightShadow](#) для общих свойств.

`camera - камера');` " onmouseout="hide()">.camera

Свойство представляет освещенную область виртуального мира. Она используется для создания карты глубины сцены; объекты, находящиеся позади других объектов, с точки зрения освещения

будут в тени.

По умолчанию это [OrthographicCamera](#) с параметрами [left](#) и [bottom](#) установленными как -5, [right](#) и [top](#) установленными как 5, параметр [near](#) (ближняя плоскость отсечения) установлен как 0.5, а параметр [far](#) (дальняя плоскость отсечения) установлен как 500.

Методы

Смотрите базовый класс [LightShadow](#) для общих методов.

Исходники

[DirectionalLightShadow.js на github.com](#)

[Object3D](#) → [Light](#) →

[HemisphereLight](#)

Источник света, расположенный непосредственно над сценой, с затуханием цвета от цвета неба до цвета земли.

Это освещение не может использоваться для отбрасывания теней.

Пример

```
var light = new THREE.HemisphereLight( 0xffffbb, 0x080820, 1 );
scene.add( light );
```

Другие примеры:

[lights / hemisphere](#)
[controls / pointerlock](#)
[decals](#)
[loader / collada / kinematics](#)
[materials / lightmap](#)
[shaders / ocean](#)

Конструктор

`HemisphereLight(skyColor, groundColor, intensity)`

[skyColor](#) — шестнадцатиричное значение цвета неба, по умолчанию равно `0xffffffff`. Дополнительный, необязательный аргумент.

[groundColor](#) — шестнадцатиричное значение цвета земли, по умолчанию равно `0xffffffff`. Дополнительный, необязательный аргумент.

[intensity](#) — числовое значение силы/интенсивности света, по умолчанию равно `1`. Дополнительный, необязательный аргумент.

Создает новый HemisphereLight.

Свойства

Смотрите базовый класс [Light](#) для общих свойств.

[.castShadow](#)

Это свойство в конструкторе устанавливается как `undefined`, так как освещение полусферой не может отбрасывать тени.

[.color](#)

Цвет освещения неба, какой передан в конструктор. Значение по умолчанию для нового [цвета](#) устанавливается как (`0xffffffff`) - белый.

[.groundColor](#)

Цвет освещения земли, какой передан в конструктор. Значение по умолчанию для нового [цвета](#) устанавливается как (`0xffffffff`) - белый.

[.isHemisphereLight](#)

Свойство используется для проверки, является ли данный класс и все производные от него классы, освещением полусфера. Значением по умолчанию является `true`.

Это свойство нельзя изменять, так как оно используется внутри `three.js` для оптимизации.

`position - позиция, положение');` `onmouseout="hide()">>.position`

Это свойство устанавливается равным

`Object3D.DefaultUp (0, 1, 0)`, так что свет светит сверху вниз.

Методы

Смотрите базовый класс `Light` для общих методов.

`copy - копировать');` `onmouseout="hide()">>.copy(
source - источник'); onmouseout="hide()">>source)`

Копирует значение параметров `color`, `intensity` и `groundColor` из `source` освещения в этот.

Исходники

[HemisphereLight.js на github.com](#)

[Object3D →](#)

Light

Абстрактный базовый класс освещения - все другие типы освещения наследуют описанные здесь свойства и методы.

Конструктор

`Light(color, intensity)`

`color` — шестнадцатиричное значение цвета освещения (дополнительный, необязательный параметр). Значением по умолчанию является `0xffffffff` (белый).

`intensity` — числовое значение силы/интенсивности света

(дополнительный, необязательный параметр). Значение по умолчанию равно 1.

Создает новый Light. Обратите внимание, что конструктор не предназначен для вызова напрямую (взамен используйте один из производных от него классов).

Свойства

Смотрите базовый класс [Object3D](#) для общих свойств.

[color](#) - цвет');

Цвет освещения. По умолчанию, если это не передается в конструкторе, новый Color устанавливается белым (0xffffffff).

[.intensity](#)

Интенсивность (сила) света.

В режиме, [правильном с точки зрения физики](#), произведение [color](#) * [intensity](#) интерпретируется как интенсивность (сила) света, измеряемая в канделях (candela).

Значение по умолчанию равно 1.0.

[.isLight](#)

Свойство используется для проверки, является ли данный класс и все производные от него классы, освещением. Значением по умолчанию является true.

Это свойство нельзя изменять, так как оно используется внутри ***three.js*** для оптимизации.

Методы

Смотрите базовый класс [Object3D](#) для общих методов.

[copy](#) - копировать');

```
source - исходник'); " onmouseout="hide()">source )
```

Метод копирует значение из source в данный объект Light.

```
.toJSON(
```

это цель, конечный пункт, предел, а метаданные - информация о другом, или данные, относящиеся к дополнительной информации о содержимом или объекте. Метаданные раскрывают сведения о признаках и свойствах, характеризующих какие-либо сущности, которые позволяют автоматически искать и управлять ими в больших информационных потоках.');" onmouseout="hide()">meta)

Возвращает данные Light в JSON-формате.

Исходники

[Light.js на github.com](#)

LightShadow

Этот класс используется для расчета теней внутри [PointLight](#), а также служит базовым классом для других классов теней.

Пример

```
// Create a WebGLRenderer and turn on shadows in the renderer
// Создаем WebGLRenderer и включаем в нем тени
var renderer = new THREE.WebGLRenderer();
renderer.shadowMap.enabled = true;
renderer.shadowMap.type = THREE.PCFSoftShadowMap; // default THREE.
                                                // по умолчанию Т

// Create a PointLight and turn on shadows for the light
// Создаем PointLight и включаем тени от него
var light = new THREE.PointLight( 0xffffff, 1, 100 );
light.position.set( 0, 10, 0 );
light.castShadow = true; // default false (значение по ум.
scene.add( light );

// Set up shadow properties for the light
// Устанавливаем свойства тени
light.shadow.mapSize.width = 512; // default (по умолчанию)
```

```

light.shadow.mapSize.height = 512; // default (по умолчанию)
light.shadow.camera.near = 0.5;    // default (по умолчанию)
light.shadow.camera.far = 500      // default (по умолчанию)

// Create a sphere that cast shadows (but does not receive them)
// Создаем сферу, отбрасывающую тени (но не принимающую их)
var sphereGeometry = new THREE.SphereBufferGeometry( 5, 32, 32 );
var sphereMaterial = new THREE.MeshStandardMaterial( { color: 0xff00ff } );
var sphere = new THREE.Mesh( sphereGeometry, sphereMaterial );
sphere.castShadow = true;        // default is false (по умолчанию равен)
sphere.receiveShadow = false;    // default (по умолчанию)
scene.add( sphere );

// Create a plane that receives shadows (but does not cast them)
// Создаем плоскость, которая принимает тени (но не отбрасывает их)
var planeGeometry = new THREE.PlaneBufferGeometry( 20, 20, 32, 32 );
var planeMaterial = new THREE.MeshStandardMaterial( { color: 0x00ffff } );
var plane = new THREE.Mesh( planeGeometry, planeMaterial );
plane.receiveShadow = true;
scene.add( plane );

// Create a helper for the shadow camera (optional)
// Создаем вспомогательный элемент для теневой камеры (необязательный)
var helper = new THREE.CameraHelper( light.shadow.camera );
scene.add( helper );

```

Конструктор

```

LightShadow(
  camera - камера');" onmouseout="hide()">camera )

```

camera - камера');" onmouseout="hide()">camera — The shadow's view of the world.

Конструктор создает новый LightShadow. Он не предназначен для вызова непосредственно - конструктор вызывается внутри [PointLight](#) или используется в качестве базового класса при построении теней от других источников света.

Свойства

[camera - камера'\);](#);" onmouseout="hide()">>.camera

The shadow's view of the world.

[.bias](#)

Отклонение от теневой карты, сколько можно добавить или отнять от нормализованной глубины при определении находится ли поверхность в тени.

Значение по умолчанию равно 0. Самые маленькие подстройки значения данного свойства (порядка 0.0001) могут помочь уменьшить артефакты в тенях.

[map - карта'\);](#);" onmouseout="hide()">>.map

Карта глубины создается при помощи внутренней камеры; расположение вне пределов пиксельной глубины находится в тени. The depth map generated using the internal camera; a location beyond a pixel's depth is in shadow. Вычисляется внутри класса, во время визуализации (рендеринга).

[map size - размер карты'\);](#);" onmouseout="hide()">>.mapSize

[Vector2](#) определяющий ширину и высоту теневой карты.

Более высокие значения дают более качественные тени за счет увеличения времени вычисления. Значения должны быть степенями двойки, вплоть до значения свойства

[WebGLRenderer.capabilities.maxTextureSize](#) данного устройства, причем ширина и высота могут быть неодинаковыми (так например, (512, 1024) вполне допустимо). Значением по умолчанию является (512, 512).

[matrix - матрица'\);](#);" onmouseout="hide()">>.matrix

Модель пространства теневой камеры, для расчета местоположения и глубины в теневой карте. Хранится в [Matrix4](#). Значение свойства вычисляется внутренне в процессе визуализации (рендеринга).

.radius

Установка значения больше 1 вызовет размытие краев тени. Более высокие значения будут вызывать нежелательный эффект полосатости в тенях - большие значения свойства mapSize позволяют использовать высокие значения данного свойства, прежде чем этот эффект станет видимым. Обратите внимание, что это свойство не оказывает никакого влияния, если свойство WebGLRenderer.shadowMap.type установлено как BasicShadowMap.

Методы

copy - копировать');" onmouseout="hide()">.copy(
source - исходник');" onmouseout="hide()">source)

Метод копирует значения всех свойств из source - исходник');" onmouseout="hide()">source в данный SpotLight.

clone - клонировать');" onmouseout="hide()">.clone()

Возвращает новый LightShadow с теми же свойствами что и у данного.

.toJSON()

Сериализует данный LightShadow.

Исходники

[LightShadow.js на github.com](#)

Object3D → Light →

[PointLight](#)

Свет, который излучается из одной точки во всех направлениях. Обычный случай использования такого освещения это повторение

освещения от простой лампочки (без светильника).
Это освещение может отбрасывать тени - подробности смотрите в
описании [LightShadow](#).

Пример

```
var light = new THREE.PointLight( 0xff0000, 1, 100 );
light.position.set( 50, 50, 50 );
scene.add( light );
```

Другие примеры:

[lights / pointlights](#)
[lights / pointlights](#)
[lights / pointlights2](#)
[animation](#)
[effects / anaglyph](#)
[geometry / text](#)
[lensflares](#)

Конструктор

```
PointLight( color, intensity, distance, decay )
```

color — шестнадцатиричное значение цвета освещения, по умолчанию 0xffffffff (белый). Дополнительный, необязательный аргумент.

intensity — числовое значение силы/интенсивности света. Значение по умолчанию равно 1. Дополнительный, необязательный аргумент.

distance - расстояние от источника света, при котором интенсивность света становится равной 0. When set to 0, then the light never stops. Значение по умолчанию равно 0.

decay - значение, на которое уменьшается интенсивность света с увеличением расстояния от источника.

Создает новый PointLight.

Свойства

Смотрите базовый класс [Light](#) для общих свойств.

.decay

Значение, на которое уменьшается интенсивность света с увеличением расстояния от источника.

В режиме, [правильном с точки зрения физики](#), `decay = 2` приводит к физически реалистичному уменьшению освещенности. Значение по умолчанию равно 1.

.distance

При ненулевом значении свет будет уменьшаться линейно от максимальной интенсивности (в расположении источника света) до нуля (на расстоянии от источника света, задаваемым данным свойством). Значение по умолчанию равно 0 . 0.

.isPointLight

Свойство используется для проверки, является ли данный класс и производные от него классы точечным освещением (PointLight). Значением по умолчанию является `true`.

Его нельзя изменять, так как оно используется внутри *three.js* для оптимизации.

.power

Сила света. (По-русски это будет [световой поток](#).)

В режиме, [правильном с точки зрения физики](#), световой поток измеряется в люменах. Значением по умолчанию является `4Math.PI`.

Он напрямую связан с интенсивностью ([intensity](#)) соотношением

`power = intensity * 4π`

и его изменение также приведет к изменению интенсивности.

shadow - тень');;" onmouseout="hide()".shadow

Для вычисления теней при данном освещении используется класс [LightShadow](#).

[Камерой LightShadow](#) устанавливается [камера с перспективной проекцией](#) и параметрами: `fov` равен 90 градусов, `aspect` равен

1, ближняя плоскость отсечения [near](#) равна 0.5 и дальняя плоскость отсечения [far](#) равна 500.

Методы

Смотрите базовый класс [Light](#) для общих методов.

```
copy - копировать';" onmouseout="hide\(\)">>.copy\(  
source - исходник';" onmouseout="hide\\(\\)">>source \\)
```

Метод копирует значения всех свойств из [source - исходник';" onmouseout="hide\(\)">>source](#) в данный PointLight.

Исходники

[PointLight.js на github.com](#)

[Object3D](#) → [Light](#) →

[RectAreaLight](#)

Этот свет излучается равномерно всей стороной прямоугольной плоскости. Он может быть использован для симуляции таких вещей как светящееся окно или освещение лентой (полосой).

Примечание: этот класс в настоящее время находится в активной разработке и пока ещё не готов к полноценному использованию (начиная с версии r83). Вернитесь сюда через месяц или два! И в тоже время не стесняйтесь попробовать его в работе.

Пример

```
var width = 2;  
var height = 10;  
var rectLight = new THREE.RectAreaLight( 0xffffffff, undefined, width );  
rectLight.intensity = 70.0;  
rectLight.position.set( 5, 5, 0 );  
scene.add( rectLight )
```

```
rectLightHelper = new THREE.RectAreaLightHelper( rectLight );
scene.add( rectLightHelper );
```

Другие примеры:

[lights / rectarealight](#)

Конструктор

```
RectAreaLight( color, intensity, width, height )
```

color — шестнадцатиричное значение цвета освещения, по умолчанию 0xffffffff (белый). Дополнительный, необязательный аргумент.

intensity — числовое значение силы/интенсивности света. Значение по умолчанию равно 1. Дополнительный, необязательный аргумент.

width - ширина источника света. Значение по умолчанию равно 10. Дополнительный, необязательный аргумент.

height - высота источника света. Значение по умолчанию равно 10. Дополнительный, необязательный аргумент.

Создает новый RectAreaLight.

Свойства

Смотрите базовый класс [Light](#) для общих свойств.

[castShadow](#)

Примечание: это свойство для этого типа освещения пока ещё не реализовано! (версия three.js r83).

[.decay](#)

Значение, на которое уменьшается интенсивность света с увеличением расстояния от источника.

В режиме, [правильном с точки зрения физики](#), decay = 2 приводит к физически реалистичному уменьшению освещенности. Значение по умолчанию равно 1.

Примечание: это свойство для этого типа освещения пока ещё не реализовано! (версия three.js r83).

.distance

При ненулевом значении свет будет уменьшаться линейно от максимальной интенсивности (в расположении источника света) до нуля (на расстоянии от источника света, задаваемым данным свойством). Значение по умолчанию равно 0.0.

Примечание: это свойство для этого типа освещения пока ещё не реализовано! (версия three.js r83).

освещением от прямоугольной области');;" onmouseout="hide()">>.isRe

Свойство используется для проверки, является ли данный класс и производные от него классы RectAreaLight. Значением по умолчанию является true.

Его нельзя изменять, так как оно используется внутри *three.js* для оптимизации.

position - позиция, положение');;" onmouseout="hide()">>.position

Это свойство устанавливается равным `Object3D.DefaultUp` (0, 1, 0), так что свет светит сверху вниз.

target - цель, назначение');;" onmouseout="hide()">>.target

Свет RectAreaLight направлен из своего местоположения (свойство position) в целевую позицию, определяемую данным свойством. По умолчанию target будет равен (0, 0, 0).

Примечание: чтобы изменить целевую позицию на что-либо другое кроме положения по умолчанию, её нужно добавить к сцене при помощи

```
scene.add( light.target );
```

Это делается для того, чтобы matrixWorld целевой позиции автоматически обновлялся с каждым кадром.

Также возможно установить целью другой объект на сцене (любой объект со свойством position), вот так:

```
var targetObject = new THREE.Object3D();
```

```
scene.add(targetObject);  
light.target = targetObject;
```

Теперь RectAreaLight будет отслеживать целевой объект.

Методы

Смотрите базовый класс [Light](#) для общих методов.

```
copy - копировать"); " onmouseout="hide()">.copy(  
source - исходник"); " onmouseout="hide()">source )
```

Метод копирует значения всех свойств из source - исходник"); " onmouseout="hide()">source в данный RectAreaLight.

Исходники

[RectAreaLight.js на github.com](#)

[Object3D](#) → [Light](#) →

[SpotLight](#)

Данный свет излучается из одной точки в одном направлении, вдоль конуса, расширяемого по мере удаления от источника света. Это освещение может отбрасывать тени - подробности смотрите в описании [SpotLightShadow](#).

Пример

```
// white spotlight shining from the side, casting shadow  
// белый прожектор светит со стороны, отбрасывает тень  
  
var spotLight = new THREE.SpotLight( 0xffffff );  
spotLight.position.set( 100, 1000, 100 );  
  
spotLight.castShadow = true;  
  
spotLight.shadow.mapSize.width = 1024;  
spotLight.shadow.mapSize.height = 1024;
```

```
spotLight.shadow.camera.near = 500;
spotLight.shadow.camera.far = 4000;
spotLight.shadow.camera.fov = 30;

scene.add( spotLight );
```

Другие примеры:

[lights / spotlights](#)
[interactive / cubes / gpu](#)
[interactive / draggablecubes](#)
[materials / bumpmap / skin](#)
[materials / cubemap / dynamic](#)
[loader / md2](#)
[shading / physical](#)
[shadowmap](#)
[shadowmap / performance](#)
[shadowmap / viewer](#)

Конструктор

```
SpotLight( color, intensity, distance, angle, penumbra, decay )
```

color — шестнадцатиричное значение цвета освещения, по умолчанию 0xffffffff (белый). Дополнительный, необязательный аргумент.

intensity — числовое значение силы/интенсивности света. Значение по умолчанию равно 1. Дополнительный, необязательный аргумент.

distance - максимальное расстояние от источника, где свет будет светить с интенсивностью, которая будет линейно уменьшаться по мере удаления от источника.

angle - максимальный угол рассеяния света по его направлению, верхней границей которого является угол $\text{Math.PI}/2$.

penumbra - процент от конуса света прожектора, в котором интенсивность света ослабляется из-за полутени. Принимает значения от нуля до единицы. Значение по умолчанию равно нулю.

decay - значение, на которое уменьшается интенсивность света с увеличением расстояния от источника.

Создает новый SpotLight.

Свойства

Смотрите базовый класс [Light](#) для общих свойств.

[.angle](#)

Maximum extent of the spotlight, in radians, from its direction. Should be no more than `Math.PI/2`. Значением по умолчанию является `Math.PI/3`.

[.castShadow](#)

Если задано значение `true`, то свет будет отбрасывать динамические тени.

Предупреждение: Это затратно по ресурсам и для правильного отображения теней требуется подстройка.

Значением по умолчанию является `false`.

[.decay](#)

Значение, на которое уменьшается интенсивность света с увеличением расстояния от источника.

В режиме, [правильном с точки зрения физики](#), `decay = 2` приводит к физически реалистичному уменьшению освещенности. Значение по умолчанию равно 1.

[.distance](#)

При ненулевом значении свет будет уменьшаться линейно от максимальной интенсивности (в расположении источника света) до нуля (на расстоянии от источника света, задаваемым данным свойством). Значение по умолчанию равно 0.0.

[.isSpotLight](#)

Свойство используется для проверки, является ли данный класс и производные от него классы прожекторным освещением.

Значением по умолчанию является `true`.

Его нельзя изменять, так как оно используется внутри `three.js` для оптимизации.

.penumbra

Процент от конуса света прожектора, в котором интенсивность света ослабляется из-за полутени. Принимает значения от 0 до 1. Значение по умолчанию равно 0 . 0.

position - положение, позиция');" onmouseout="hide()">.position

Значение свойства устанавливается равным Object3D.DefaultUp (0, 1, 0), так что свет светит сверху вниз.

.power

Сила света. (По-русски это будет световой поток.)

В режиме, правильном с точки зрения физики, световой поток измеряется в люменах. Значением по умолчанию является 4Math.PI.

Он напрямую связан с интенсивностью (intensity) соотношением

`power = intensity * π`

и его изменение также приведет к изменению интенсивности.

shadow - тень');" onmouseout="hide()">.shadow

Для вычисления теней при данном освещении используется класс SpotLightShadow.

target - цель, мишень, назначение');" onmouseout="hide()">.target

Со своей позицией Spotlight направлен на позицию объекта-цели (target.position), которую и определяет данное свойство. Позицией цели по умолчанию является (0, 0, 0).

Примечание: Чтобы изменить позицию объекта-цели на что-либо другое, кроме значения по умолчанию, он должен быть добавлен на сцену при помощи

`scene.add(light.target);`

Так что matrixWorld цели станет автоматически обновляться с каждым кадром.

Также возможно установить в качестве цели другой объект на сцене (что-нибудь со свойством [position](#)), вот так:

```
var targetObject = new THREE.Object3D();
scene.add(targetObject);

light.target = targetObject;
```

Теперь Spotlight будет отслеживать целевой объект.

Методы

Смотрите базовый класс [Light](#) для общих методов.

```
copy - копировать');" onmouseout="hide()">.copy(
source - исходник');" onmouseout="hide()">source )
```

Метод копирует значения всех свойств из [source - исходник](#);" onmouseout="hide()">source в данный [SpotLight](#).

Исходники

[SpotLight.js на github.com](#)

[LightShadow →](#)

SpotLightShadow

Данный класс используется для расчета теней внутри класса

[SpotLight](#).

Пример

```
// Create a WebGLRenderer and turn on shadows in the renderer
// создаем WebGLRenderer и включаем тени в визуализаторе
var renderer = new THREE.WebGLRenderer();
renderer.shadowMap.enabled = true;
renderer.shadowMap.type = THREE.PCFSoftShadowMap; // default THREE.
                                                // (по умолчанию

// Create a SpotLight and turn on shadows for the light
```

```

// создаем SpotLight и включаем тени от него
var light = new THREE.SpotLight( 0xffffffff );
light.castShadow = true; // default false (по умолчанию)
scene.add( light );

// Set up shadow properties for the light
// установка свойств тени для этого освещения
light.shadow.mapSize.width = 512; // default (значение по умолчанию)
light.shadow.mapSize.height = 512; // default (значение по умолчанию)
light.shadow.camera.near = 0.5; // default (значение по умолчанию)
light.shadow.camera.far = 500 // default (значение по умолчанию)

// Create a sphere that cast shadows (but does not receive them)
// создаем сферу, отбрасывающую тени (но не принимающую их)
var sphereGeometry = new THREE.SphereBufferGeometry( 5, 32, 32 );
var sphereMaterial = new THREE.MeshStandardMaterial( { color: 0xff00ff } );
var sphere = new THREE.Mesh( sphereGeometry, sphereMaterial );
sphere.castShadow = true; // default is false (по умолчанию - false)
sphere.receiveShadow = false; // default (значение по умолчанию)
scene.add( sphere );

// Create a plane that receives shadows (but does not cast them)
// Создаем плоскость, принимающую тени (но не отбрасывающую их)
var planeGeometry = new THREE.PlaneBufferGeometry( 20, 20, 32, 32 );
var planeMaterial = new THREE.MeshStandardMaterial( { color: 0x00ffff } );
var plane = new THREE.Mesh( planeGeometry, planeMaterial );
plane.receiveShadow = true;
scene.add( plane );

// Create a helper for the shadow camera (optional)
// создаем вспомогательный элемент для камеры теней (необязательно)
var helper = new THREE.CameraHelper( light.shadow.camera );
scene.add( helper );

```

Конструктор

Конструктор создает [PerspectiveCamera](#) для управления отображением теней в представляемом мире.

Свойства

Смотрите базовый класс [LightShadow](#) для общих свойств.

[camera - камера'\);](#);" onmouseout="hide()">>.camera

Представление освещения в мире. Свойство используется для построения карты глубины сцены; объекты позади других объектов исходя из распространения света будут находиться в тени.

По умолчанию это [камера с перспективной проекцией](#) с ближней плоскостью отсечения [near](#) равной 0.5. Свойство [fov](#) будет отслеживать свойство [angle](#) имеющегося [SpotLight](#) с помощью метода [update](#). Аналогично, свойство [aspect](#) будет отслеживать размеры сторон из [mapSize](#). Если установлено свойство [distance](#) освещения, дальняя плоскость отсечения [far](#) будет отслеживать его, в противном случае она по умолчанию будет установлена как 500.

[.isSpotLightShadow](#)

Свойство используется для проверки, является ли данный класс и производные от него классы тенью от прожекторного освещения. Значением по умолчанию является true.

Его нельзя изменять, так как оно используется внутри [three.js](#) для оптимизации.

Методы

Смотрите базовый класс [LightShadow](#) для общих методов.

[update - обновление' \);" onmouseout="hide\(\)">.update\(light \)](#)

Метод обновляет внутреннюю [камеру с перспективной проекцией](#) исходя из параметров освещения, переданного в light.

Исходники

[SpotLightShadow.js на github.com](#)

ЗАГРУЗЧИКИ

AnimationLoader

Класс для загрузки анимации в формате JSON. Для загрузки файлов, внутри класса, используется [FileLoader](#).

Пример

```
// instantiate a loader (создаем экземпляр загрузчика)
var loader = new THREE.AnimationLoader();

// load a resource (загрузка ресурса)
loader.load(
    // resource URL (URL-адрес ресурса)
    'animations/animation.js',
    // Function when resource is loaded
    // функция, вызываемая при загрузке ресурса
    function ( animation ) {
        // do something with the animation
        // здесь выполняется какая-нибудь анимация
    },
    // Function called when download progresses
    // функция, вызываемая при процессе загрузки
    function ( xhr ) {
        console.log( (xhr.loaded / xhr.total * 100) + '% loaded' );
    },
    // Function called when download errors
    // функция, вызываемая при ошибке загрузки
    function ( xhr ) {
        console.log( 'An error happened' );
    }
);
```

Конструктор

```
AnimationLoader( manager )
```

manager — The [loadingManager](#) for the loader to use. По умолчанию это [THREE.DefaultLoadingManager](#).

Создает новый AnimationLoader.

Свойства

.manager

The [page:LoadingManager loadingManager] the loader is using. По умолчанию это [page:DefaultLoadingManager].

Методы

.load(url, onLoad, onProgress, onError)

url — путь или URL к файлу. Также это может быть [Data URI](#).

onLoad — Функция, которая будет вызвана по завершении загрузки. Аргументом будет загруженная [анимация](#).

onProgress — Функция, которая будет вызываться пока идет процесс загрузки. Аргументом будет экземпляр XMLHttpRequest, что содержит байты .total и .loaded.

onError — Функция, которая будет вызвана в случае ошибки при загрузке.

Начинает загрузку с URL-адреса и передает загруженную анимацию функции, указанной в onLoad.

.parse(json, onLoad)

json — обязателен

onLoad — функция, вызываемая по завершении разбора (парсинга) JSON объекта.

Анализирует (разбирает) объект JSON и передает результаты в onLoad. Отдельные клипы объекта будут обработаны с помощью метода [AnimationClip.parse](#).

Исходники

[AnimationLoader.js в этом справочнике](#)

[AnimationLoader.js на github.com](#)

AudioLoader

Класс для загрузки [AudioBuffer](#). Для загрузки файлов, внутри класса, используется [FileLoader](#).

Пример

```
// instantiate a listener (создаем экземпляр слушателя)
```

```

var audioListener = new THREE.AudioListener();

// add the listener to the camera (добавляем слушателя к камере)
camera.add( audioListener );

// instantiate audio object (создаем экземпляр аудиообъекта)
var oceanAmbientSound = new THREE.Audio( audioListener );

// add the audio object to the scene (добавляем к сцене аудиообъект)
scene.add( oceanAmbientSound );

// instantiate a loader (создаем экземпляр загрузчика)
var loader = new THREE.AudioLoader();

// load a resource (загружаем ресурс)
loader.load(
    // resource URL (адрес ресурса)
    'audio/ambient_ocean.ogg',
    // Function when resource is loaded
    // функция, вызываемая после загрузки ресурса
    function ( audioBuffer ) {
        // set the audio object buffer to the loaded object
        // устанавливаем буфер аудиообъекта для загруженного объекта
        oceanAmbientSound.setBuffer( audioBuffer );

        // play the audio - воспроизводим аудио
        oceanAmbientSound.play();
    },
    // Function called when download progresses
    // функция, вызываемая во время выполнения загрузки
    function ( xhr ) {
        console.log( (xhr.loaded / xhr.total * 100) + '% loaded' );
    },
    // Function called when download errors
    // функция, вызываемая при ошибках загрузки
    function ( xhr ) {
        console.log( 'An error happened' );
    }
);

```

Конструктор

`AudioLoader(context, manager)`

context — [AudioContext](#) for the loader to use. Значением по умолчанию является [window.AudioContext](#).

manager — [loadingManager](#) for the loader to use. По умолчанию это [THREE.DefaultLoadingManager](#).

Создает новый AudioLoader.

Свойства

`.manager`

The loadingManager the loader is using. По умолчанию это [THREE.DefaultLoadingManager](#).

Методы

`load(url, onLoad, onProgress, onError)`

`url` — обязательный аргумент.

`onLoad` — Будет вызываться при завершении загрузки.

Аргументом должен быть загруженный текст ответа.

`onProgress` — Будет вызываться во время процесса загрузки.

Аргументом должен быть экземпляр XMLHttpRequest, который содержит байты `.total` и `.loaded`.

`onError` — будет вызываться в случае ошибок при загрузке.

Начинает загрузку из `url` и передает загруженный [AudioBuffer](#) в `onLoad`.

Исходники

[AudioLoader.js в этом справочнике](#)

[AudioLoader.js на github.com](#)

BufferGeometryLoader

Загрузчик для загрузки [BufferGeometry](#). Для загрузки файлов, внутри класса, используется [FileLoader](#).

Пример

[WebGL / geometry / colors / lookuptable](#)

```
// instantiate a loader
var loader = new THREE.BufferGeometryLoader();

// load a resource
loader.load(
    // resource URL
    'models/json/pressure.json',
    // Function when resource is loaded
    function ( geometry ) {
```

```
        var material = new THREE.MeshLambertMaterial( { col
        var object = new THREE.Mesh( geometry, material );
        scene.add( object );

    },
    // Function called when download progresses
    function ( xhr ) {
        console.log( (xhr.loaded / xhr.total * 100) + '% lo
    },
    // Function called when download errors
    function ( xhr ) {
        console.log( 'An error happened' );
    }
);
```

Конструктор

`BufferGeometryLoader([page:LoadingManager manager])`

[page:LoadingManager manager] — The [page:LoadingManager loadingManager] for the loader to use. Default is [page:LoadingManager THREE.DefaultLoadingManager].

Creates a new BufferGeometryLoader.

Свойства

Методы

`[method:null load]([page:String url], [page:Function onLoad], [pag`

[page:String url] — required

[page:Function onLoad] — Will be called when load completes. The argument will be the loaded [page:BufferGeometry].

[page:Function onProgress] — Will be called while load progresses. The argument will be the XMLHttpRequest instance, that contain .

[page:Integer total] and .[page:Integer loaded] bytes.

[page:Function onError] — Will be called when load errors.

Begin loading from url and call onLoad with the parsed response content.

`[method:BufferGeometry parse]([page:Object json])`

[page:Object json] — The **JSON** structure to parse.

Parse a **JSON** structure and return a [page:BufferGeometry].

Исходники

[BufferGeometryLoader.js в этом справочнике](#)

[BufferGeometryLoader.js на github.com](#)

Cache

Простая система кэширования, используемая внутри [FileLoader](#).

Примеры

```
WebGL / geometry / text / earcut
WebGL / interactive / instances / gpu
WebGL / loader / ttf
```

Применение

Чтобы включить кэширование во всех загрузчиках, использующих [FileLoader](#), установите

```
THREE.Cache.enabled = true
```

Свойства

[.enabled](#)

Свойство определяет, будет ли включено кэширование.

Значением по умолчанию является `false`.

[.files](#)

Объект, удерживающий кэшированные файлы. An [page:Object object] that holds cached files.

Методы

[.add\(key, file \)](#)

`key` — the [page:String key] to reference the cached file by.

[page:Object file] — файл, который нужно кэшировать.

Adds a cache entry with a key to reference the file. If this key already holds a file, it is overwritten.

[.get\(key \)](#)

`key` — строковый ключ.

Метод получает значение [page:String key]. Если ключа не существует, возвращается значение `undefined`, т.е. неопределено.

[.remove\(key \)](#)

`key` — строковый ключ, связанный с кэшированным файлом.

Удаляет кэшированный файл, связанный с данным ключом.

[.clear\(\)](#)

Удаляет все значения из кэша.

Исходники

[Cache.js в этом справочнике](#)

[Cache.js на github.com](#)

CompressedTextureLoader

Abstract base class for block based textures loader (dds, pvr, ...). This uses the [page:FileLoader] internally for loading files.

Примеры

See the

[link:<https://github.com/mrdoob/three.js/blob/master/examples/js/loaders/CDDSLoader>] and

[link:<https://github.com/mrdoob/three.js/blob/master/examples/js/loaders/FPVRLoader>] for examples of derived classes.

Конструктор

CompressedTextureLoader([page:LoadingManager manager])

[page:LoadingManager manager] — The [page:LoadingManager loadingManager] for the loader to use. Default is [page:LoadingManager THREE.DefaultLoadingManager].

Создает новый CompressedTextureLoader.

Свойства

[property:LoadingManager manager]

The [page:LoadingManager loadingManager] the loader is using. Default is [page:DefaultLoadingManager].

[property:String path]

The base path from which files will be loaded. See [page:.setPath]. Default is *undefined*.

Методы

[method:null load]([page:String url], [page:Function onLoad], [page:Function onProgress], [page:Function onError])

[page:String url] — the path or URL to the file. This can also be a [link:[Data URI](https://developer.mozilla.org/en-US/docs/Web/HTTP/Basics_of_HTTP/Data_URIs)].

[page:Function onLoad] — Will be called when load completes. The argument will be the loaded texture.

[page:Function onProgress] — Will be called while load progresses. The argument will be the XMLHttpRequest instance, which contains .

[page:Integer total] and .[page:Integer loaded] bytes.

[page:Function onError] — Will be called when load errors.

Begin loading from url and pass the loaded texture to onLoad.

[method:FileLoader setPath]([page:String path])

Set the base path or URL from which to load files. This can be useful if you are loading many textures from the same directory.

Исходники

[CompressedTextureLoader.js в этом справочнике](#)

[CompressedTextureLoader.js на github.com](#)

CubeTextureLoader

Class for loading a [page:CubeTexture CubeTexture]. This uses the [page:ImageLoader] internally for loading files.

Пример

[example:webgl_materials_cubemap materials / cubemap]

[example:webgl_materials_cubemap_balls_reflection materials / cubemap / balls / reflection]

[example:webgl_materials_cubemap_balls_refraction materials / cubemap / balls / refraction]

[example:webgl_materials_cubemap_dynamic materials / cubemap / dynamic]

[example:webgl_materials_cubemap_dynamic2 materials / cubemap / dynamic2]

[example:webgl_materials_cubemap_refraction materials / cubemap / refraction]

```
var scene = new THREE.Scene();
scene.background = new THREE.CubeTextureLoader()
    .setPath( 'textures/cubeMaps/' )
    .load( [
        '1.png',
        '2.png',
        '3.png',
        '4.png',
        '5.png',
        '6.png'
    ] );
```

Конструктор

CubeTextureLoader([page:LoadingManager manager])

[page:LoadingManager manager] — The [page:LoadingManager loadingManager] for the loader to use. Default is [page:LoadingManager THREE.DefaultLoadingManager].

Создает новый CubeTextureLoader.

Свойства

[property:String crossOrigin]

If set, assigns the [link:https://developer.mozilla.org/en-US/docs/Web/HTML/CORS_settings_attributes crossOrigin] attribute of the image to the value of *crossOrigin*, prior to starting the load. Default is *undefined*.

[property:LoadingManager manager]

The [page:LoadingManager loadingManager] the loader is using. Default is [page:DefaultLoadingManager].

[property:String path]

The base path from which files will be loaded. See [page:.setPath]. Default is *undefined*.

Методы

[method:null load]([page:String urls], [page:Function onLoad], [page:Function onProgress], [page:Function onError])

[page:String urls] — array of 6 urls to images, one for each side of the CubeTexture. These can also be [link:https://developer.mozilla.org/en-US/docs/Web/HTTP/Basics_of_HTTP/Data_URIs Data URIs].

[page:Function onLoad] — Will be called when load completes. The argument will be the loaded [page:Texture texture].

[page:Function onProgress] — Will be called while load progresses. The argument will be the XMLHttpRequest instance, which contains .

[page:Integer total] and .[page:Integer loaded] bytes.

[page:Function onError] — Will be called when load errors.

Begin loading from url and pass the loaded [page:Texture texture] to onLoad.

[method:null setCrossOrigin]([page:String value])

Set the [page:.crossOrigin] attribute.

[method:FileLoader setPath]([page:String path])

Set the base path or URL from which to load files. This can be useful if you are loading many textures from the same directory.

Исходники

[CubeTextureLoader.js в этом справочнике](#)

[CubeTextureLoader.js на github.com](#)

DataTextureLoader

Abstract base class to load generic binary textures formats (rgbe, hdr, ...). This uses the [page:FileLoader] internally for loading files, and creates a new [page:DataTexture].

Примеры

See the

[link:<https://github.com/mrdoob/three.js/blob/master/examples/js/loaders/RGBELoader>] for an example of a derived class.

Конструктор

DataTextureLoader([page:LoadingManager manager])

[page:LoadingManager manager] — The [page:LoadingManager loadingManager] for the loader to use. Default is [page:LoadingManager THREE.DefaultLoadingManager].

Создание нового DataTextureLoader.

Свойства

[property:LoadingManager manager]

The [page:LoadingManager loadingManager] the loader is using. Default is [page:DefaultLoadingManager].

Методы

[method:null load]([page:String url], [page:Function onLoad], [page:Function onProgress], [page:Function onError])

[page:String url] — the path or URL to the file. This can also be a [link:[Data URI](https://developer.mozilla.org/en-US/docs/Web/HTTP/Basics_of_HTTP/Data_URIs)].

[page:Function onLoad] — Will be called when load completes. The argument will be the loaded texture.

[page:Function onProgress] — Will be called while load progresses. The argument will be the XMLHttpRequest instance, which contains .

[page:Integer total] and .[page:Integer loaded] bytes.

[page:Function onError] — Will be called when load errors.

Begin loading from url and pass the loaded texture to onLoad.

Исходники

[DataTextureLoader.js в этом справочнике](#)

[DataTextureLoader.js на github.com](#)

FileLoader

A low level class for loading resources with XMLHttpRequest, used internally by most loaders.

Конструктор

```
FileLoader( [page:LoadingManager manager] )
```

[page:LoadingManager manager] — The [page:LoadingManager loadingManager] for the loader to use. Default is [page:LoadingManager THREE.DefaultLoadingManager].

Creates a new FileLoader.

Свойства

```
[property:Cache cache]
```

A [page:Cache cache] instance that hold the response from each request made through this loader, so each file is requested once.

```
[property:String responseType]
```

Can be set to change the response type.

Методы

```
[method:null load]( [page:String url], [page:Function onLoad], [pag
```

[page:String url] — required

[page:Function onLoad] — Will be called when load completes. The argument will be the loaded text response.

[page:Function onProgress] — Will be called while load progresses. The argument will be the XMLHttpRequest instance, that contain .

[page:Integer total] and .[page:Integer loaded] bytes.

[page:Function onError] — Will be called when load errors.

Begin loading from url and return the [page:String text] response that will contain the data.

```
[method:null setCrossOrigin]( [page:String value] )
```

[page:String value] — The crossOrigin string to implement CORS for loading the url from a different domain that allows CORS.

```
[method:null setResponseType]( [page:String value] )
```

[page:String value] — the empty string (default), "arraybuffer", "blob", "document", "json", or "text".

Пример

[example:webgl_morphtargets_human]

Исходники

[FileLoader.js в этом справочнике](#)

[FileLoader.js на github.com](#)

FontLoader

Class for loading a font in JSON format. Returns a [page:Font Font], which is an array of [page:Shape Shape]s representing the font. This uses the [page:FileLoader] internally for loading files.

You can convert fonts online using [link:<https://gero3.github.io/facetype.js/facetype.js>]

Examples

```
[example:webgl_geometry_text_shapes geometry / text / shapes ]
[example:webgl_geometry_text geometry / text ]
[example:webgl_geometry_text_earcut geometry / text / earcut]
[example:webgl_geometry_text_pnltri geometry / text / pnltri]

var loader = new THREE.FontLoader();
var font = loader.load(
    // resource URL
    'fonts/helvetiker_bold.typeface.json'
    // Function when resource is loaded
    function ( font ) {
        // do something with the font
        scene.add( font );
    },
    // Function called when download progresses
    function ( xhr ) {
        console.log( (xhr.loaded / xhr.total * 100) + '% lo
    },
    // Function called when download errors
    function ( xhr ) {
        console.log( 'An error happened' );
    }
);
```

Constructor

`FontLoader([page:LoadingManager manager])`

[page:LoadingManager manager] — The [page:LoadingManager loadingManager] for the loader to use. Default is [page:LoadingManager THREE.DefaultLoadingManager].

Creates a new FontLoader.

Properties

[property:LoadingManager manager]

The [page:LoadingManager loadingManager] the loader is using. Default is [page:DefaultLoadingManager].

Methods

[method:null load]([page:String url], [page:Function onLoad], [pa

[page:String url] — the path or URL to the file. This can also be a [link:[Data URI](https://developer.mozilla.org/en-US/docs/Web/HTTP/Basics_of_HTTP/Data_URIs)].

[page:Function onLoad] — Will be called when load completes. The argument will be the loaded [page:Texture texture].

[page:Function onProgress] — Will be called while load progresses. The argument will be the XMLHttpRequest instance, which contains .

[page:Integer total] and .[page:Integer loaded] bytes.

[page:Function onError] — Will be called when load errors.

Begin loading from url and pass the loaded [page:Texture texture] to onLoad.

[method:Font parse]([page:Object json])

[page:Object json] — The **JSON** structure to parse.

Parse a **JSON** structure and return a [page:Font].

Исходники

[FontLoader.js в этом справочнике](#)

[FontLoader.js на github.com](#)

ImageLoader

Загрузчик для загрузки [page:Image].

Конструктор

ImageLoader([page:LoadingManager manager])

[page:LoadingManager manager] — The [page:LoadingManager loadingManager] for the loader to use. Default is

[page:LoadingManager THREE.DefaultLoadingManager].
Создает новый ImageLoader.

Свойства

[property:String crossOrigin]

The crossOrigin string to implement CORS for loading the url from a different domain that allows CORS.

Методы

[method:null load]([page:String url], [page:Function onLoad], [pag

[page:String url] — required

[page:Function onLoad] — Will be called when load completes. The argument will be the loaded [page:Image image].

[page:Function onProgress] — Will be called while load progresses. The argument will be the progress event.

[page:Function onError] — Will be called when load errors.

Begin loading from url and return the [page:Image image] object that will contain the data.

[method:null setCrossOrigin]([page:String value])

[page:String value] — The crossOrigin string.

The crossOrigin string to implement CORS for loading the url from a different domain that allows CORS.

Пример

```
// instantiate a loader
var loader = new THREE.ImageLoader();

// load a image resource
loader.load(
    // resource URL
    'textures/skyboxsun25degtest.png',
    // Function when resource is loaded
    function ( image ) {
        // do something with it

        // like drawing a part of it on a canvas
        var canvas = document.createElement( 'canvas' );
        var context = canvas.getContext( '2d' );
        context.drawImage( image, 100, 100 );

    },
    // Function called when download progresses
    function ( xhr ) {
```

```
        console.log( (xhr.loaded / xhr.total * 100) + '% loaded' );
    },
    // Function called when download errors
    function ( xhr ) {
        console.log( 'An error happened' );
    }
);
```

[example:webgl_shaders_ocean]

Исходники

[ImageLoader.js в этом справочнике](#)

[ImageLoader.js на github.com](#)

JSONLoader

Загрузчик для загрузки объектов в JSON формате.

Конструктор

```
JSONLoader( [page:LoadingManager manager] )
```

[page:LoadingManager manager] — The [page:LoadingManager loadingManager] for the loader to use. Default is [page:LoadingManager THREE.DefaultLoadingManager].

Создает новый JSONLoader.

Методы

```
[method:null load]( [page:String url], [page:Function onLoad], [pag
```

[page:String url] — required.

[page:Function onLoad] — Will be called when load completes. The argument will be the loaded text response.

[page:Function onProgress] — Will be called while load progresses. The argument will be the XMLHttpRequest instance, that contains .

[page:Integer total] and .[page:Integer loaded] bytes.

[page:Function onError] — Will be called when load errors.

Begin loading from url and pass the **JSON** to onLoad.

```
[method:null setTexturePath]( [page:String texturePath] )
```

[page:String texturePath] — Base path for textures.

Set the base path for textures.

```
[method:Object3D parse]( [page:Object json], [page:String texturePa
```

[page:String json] — JSON object to parse.
[page:String texturePath] — Base path for textures.

Parse a **JSON** structure and return an [page:object] containing the parsed [page:Geometry geometry] and [page:Array materials].

Пример

```
// instantiate a loader
var loader = new THREE.JSONLoader();

// load a resource
loader.load(
    // resource URL
    'models/animated/monster/monster.js',
    // Function when resource is loaded
    function ( geometry, materials ) {
        var material = new THREE.MultiMaterial( materials )
        var object = new THREE.Mesh( geometry, material );
        scene.add( object );
    }
);
```

[example:webgl_loader_json_blender]

Исходники

[JSONLoader.js в этом справочнике](#)
[JSONLoader.js на github.com](#)

Loader

Base class for implementing loaders.

Конструктор

```
Loader()
```

Creates a new Loader. This should be called as base class.

Свойства

```
[property:Function onLoadStart]
```

Will be called when load starts.

The default is a function with empty body.

```
[property:Function onLoadProgress]
```

Will be called while load progresses.

The default is a function with empty body.

```
[property:Function onLoadComplete]
```

Will be called when load completes.

The default is a function with empty body.

```
[property:string crossOrigin]
```

The crossOrigin string to implement CORS for loading the url from a different domain that allows CORS.

Методы

```
[method:Material createMaterial]( [page:Object m], [page:string tex
```

[page:Object m] — The parameters to create the material.

[page:String texturePath] — The base path of the textures.

Creates the Material based on the parameters m.

```
[method:Array initMaterials]( [page:Array materials], [page:string
```

[page:Array materials] — an array of parameters to create materials.

[page:String texturePath] — The base path of the textures.

Creates an array of [page:Material] based on the array of parameters m.

The index of the parameters decide the correct index of the materials.

```
[method:String extractUrlBase]( [page:string url] )
```

[page:String url] — The url to extract the base url from.

Extract the base from the URL.

Исходники

[Loader.js в этом справочнике](#)

[Loader.js на github.com](#)

MaterialLoader

A loader for loading a [page:Material] in JSON format.

Конструктор

```
MaterialLoader( [page:LoadingManager manager] )
```

[page:LoadingManager manager] — The [page:LoadingManager loadingManager] for the loader to use. Default is [page:LoadingManager THREE.DefaultLoadingManager].

Создает новый MaterialLoader.

Методы

[method:null load]([page:String url], [page:Function onLoad], [page:String url] — required

[page:Function onLoad] — Will be called when load completes. The argument will be the loaded [page:Material].

[page:Function onProgress] — Will be called while load progresses. The argument will be the progress event.

[page:Function onError] — Will be called when load errors.

Begin loading from url and return the [page:Material] object that will contain the data.

[method:Material parse]([page:Object json])

[page:Object json] — The json object containing the parameters of the Material.

Parse a **JSON** structure and create a new [page:Material] of the type [page:String json.type] with parameters defined in the json object.

Пример

```
// instantiate a loader
var loader = new THREE.MaterialLoader();

// load a resource
loader.load(
    // resource URL
    'path/to/material.json',
    // Function when resource is loaded
    function ( material ) {
        object.material = material;
    },
    // Function called when download progresses
    function ( xhr ) {
        console.log( (xhr.loaded / xhr.total * 100) + '% loaded' );
    },
    // Function called when download errors
    function ( xhr ) {
        console.log( 'An error happened' );
    }
);
```

Source

[link:[https://github.com/mrdoob/three.js/blob/master/src/\[path\].js](https://github.com/mrdoob/three.js/blob/master/src/[path].js)
src/[path].js]

ObjectLoader

A loader for loading a JSON resource. Unlike the [page:JSONLoader], this one make use of the **.type** attributes of objects to map them to their original classes.

Конструктор

```
objectLoader( [page:LoadingManager manager] )
```

[page:LoadingManager manager] — The [page:LoadingManager loadingManager] for the loader to use. Default is [page:LoadingManager THREE.DefaultLoadingManager].

Создает новый ObjectLoader.

Свойства

Методы

```
[method:null load]( [page:String url], [page:Function onLoad], [pag
```

[page:String url] — required

[page:Function onLoad] — Will be called when load completes. The argument will be the loaded [page:Object3D object].

[page:Function onProgress] — Will be called while load progresses. The argument will be the XMLHttpRequest instance, that contain .

[page:Integer total] and [.page:Integer loaded] bytes.

[page:Function onError] — Will be called when load errors.

Begin loading from url and call onLoad with the parsed response content.

```
[method:Object3D parse]( [page:Object json] )
```

[page:Object json] — required. The JSON source to parse

Parse a **JSON** content and return a threejs object.

```
[method:null setCrossOrigin]( [page:String value] )
```

[page:String value] — The crossOrigin string to implement CORS for loading the url from a different domain that allows CORS.

Пример

```
// instantiate a loader (создаем экземпляр загрузчика)
var loader = new THREE.ObjectLoader();

// assuming we loaded a JSON structure from elsewhere
// предполагаем, что структура JSON загружена из другого места
var object = loader.parse( a_json_object );

scene.add( object );

[example:webgl_loader_msgpack]
```

Исходники

[link:[https://github.com/mrdoob/three.js/blob/master/src/\[path\].js](https://github.com/mrdoob/three.js/blob/master/src/[path].js)
src/[path].js]

TextureLoader

Класс для загрузки [page:Texture texture].

Конструктор

TextureLoader([page:LoadingManager manager])

[page:LoadingManager manager] — The [page:LoadingManager loadingManager] for the loader to use. Default is [page:LoadingManager THREE.DefaultLoadingManager].

Создает новый TextureLoader.

Свойства

[property:String crossOrigin]

default — *null*.

If set, assigns the *crossOrigin* attribute of the image to the value of *crossOrigin*, prior to starting the load.

Методы

[method:null load]([page:String url], [page:Function onLoad], [pag

[page:String url] — required

[page:Function onLoad] — Will be called when load completes. The argument will be the loaded [page:Texture texture].

[page:Function onProgress] — Will be called while load progresses. The argument will be the XMLHttpRequest instance, that contain .

[page:Integer total] and .[page:Integer loaded] bytes.

[page:Function onError] — Will be called when load errors.

Begin loading from url and pass the loaded [page:Texture texture] to onLoad.

Пример

```
// instantiate a loader (создаем экземпляр загрузчика)
var loader = new THREE.TextureLoader();

// load a resource (загружаем ресурс)
loader.load(
    // resource URL (URL-адрес ресурса)
```

```
'textures/land_ocean_ice_cloud_2048.jpg',
// Function when resource is loaded
// функция, вызываемая после загрузки ресурса
function ( texture ) {
    // do something with the texture
    var material = new THREE.MeshBasicMaterial( {
        map: texture
    } );
},
// Function called when download progresses
// функция, вызываемая в процессе загрузки
function ( xhr ) {
    console.log( (xhr.loaded / xhr.total * 100) + '% loaded' );
},
// Function called when download errors
// функция, вызываемая в случае ошибок загрузки
function ( xhr ) {
    console.log( 'An error happened' );
}
);
```

[example:canvas_geometry_earth]

Исходники

[link:[https://github.com/mrdoob/three.js/blob/master/src/\[path\].js](https://github.com/mrdoob/three.js/blob/master/src/[path].js)
src/[path].js]

Менеджеры загрузки

DefaultLoadingManager

A global instance of the [page:LoadingManager LoadingManager], used by most loaders when no custom manager has been specified.

This will be sufficient for most purposes, however there may be times when you desire separate loading managers for say, textures and models.

Пример

[example:webgl_loader_scene WebGL / loader / scene]
You can optionally set the [page:LoadingManager.onStart onStart], [page:LoadingManager.onLoad onLoad], [page:LoadingManager.onProgress onProgress], [page:LoadingManager.onStart onError] functions for the manager. These will then apply to any loaders using the DefaultLoadingManager.
Note that these shouldn't be confused with the similarly named functions of individual loaders, as they are intended for displaying information about the overall status of loading, rather than dealing with the data that has been loaded.

```
THREE.DefaultLoadingManager.onStart = function ( url, itemsLoaded,
    console.log( 'Started loading file: ' + url + '\nLoaded ' +
};

THREE.DefaultLoadingManager.onLoad = function ( ) {
    console.log( 'Loading Complete!' );
};

THREE.DefaultLoadingManager.onProgress = function ( url, itemsLoade
    console.log( 'Loading file: ' + url + '\nLoaded ' + itemsL
};

THREE.DefaultLoadingManager.onError = function ( url ) {
    console.log( 'There was an error loading ' + url );
};
```

Свойства

See the [page:LoadingManager LoadingManager] page for details of properties.

Методы

See the [page:LoadingManager LoadingManager] page for details of methods.

Исходники

[link:[## LoadingManager](https://github.com/mrdoob/three.js/blob/master/src/loaders>LoadingM
src/loaders>LoadingManager.js]</p></div><div data-bbox=)

Handles and keeps track of loaded and pending data. A default global instance of this class is created and used by loaders if not supplied manually - see [page:DefaultLoadingManager].

In general that should be sufficient, however there are times when it can be useful to have separate loaders - for example if you want to show separate loading bars for objects and textures.

Пример

```
[example:webgl_loader_babylon WebGL / loader / babylon]
[example:webgl_loader_fbx WebGL / loader / fbx]
[example:webgl_loader_obj WebGL / loader / obj]
[example:webgl_materials_reflectivity WebGL / materials / reflectivity]
[example:webgl_postprocessing_outline WebGL / postprocessing / outline]
[example:webgl_terrain_dynamic WebGL / terrain / dynamic]

var manager = new THREE.LoadingManager();
manager.onStart = function ( url, itemsLoaded, itemsTotal ) {
    console.log( 'Started loading file: ' + url + '.\nLoaded ' +
};

manager.onLoad = function ( ) {
    console.log( 'Loading complete!' );
};

manager.onProgress = function ( url, itemsLoaded, itemsTotal ) {
    console.log( 'Loading file: ' + url + '.\nLoaded ' + itemsL
};

manager.onError = function ( url ) {
    console.log( 'There was an error loading ' + url );
};
```

```
var loader = new THREE.OBJLoader( manager );
loader.load( 'file.obj', function ( object ) {
    //
} );
```

Конструктор

`LoadingManager([page:Function onLoad], [page:Function onProgress], [page:Function onLoad] — (optional) this function will be called when all loaders are done.`

`[page:Function onProgress] — (optional) this function will be called when an item is complete.`

`[page:Function onError] — (optional) this function will be called a loader encounters errors.`

Создает новый LoadingManager.

Свойства

`[property:Function onStart]`

This function will be called when loading starts. The arguments are:

`[page:String url] — The url of the item just loaded.`

`[page:Integer itemsLoaded] — the number of items already loaded so far.`

`[page:Integer itemsTotal] — the total amount of items to be loaded.`

By default this is undefined.

`[property:Function onLoad]`

This function will be called when all loading is completed. By default this is undefined, unless passed in the constructor.

`[property:Function onProgress]`

This function will be called when an item is complete. The arguments are:

`[page:String url] — The url of the item just loaded.`

`[page:Integer itemsLoaded] — the number of items already loaded so far.`

`[page:Integer itemsTotal] — the total amount of items to be loaded.`

By default this is undefined, unless passed in the constructor.

`[property:Function onError]`

This function will be called when any item errors, with the argument:
[page:String url] — The url of the item that errored.

By default this is undefined, unless passed in the constructor.

Методы

Note: The following methods are designed to be called internally by loaders. You shouldn't call them directly.

[method:null itemStart]([page:String url])

[page:String url] — the url to load

This should be called by any loader used by the manager when the loader starts loading an url.

[method:null itemEnd]([page:String url])

[page:String url] — the loaded url

This should be called by any loader used by the manager when the loader ended loading an url.

[method:null itemError]([page:String url])

[page:String url] — the loaded url

This should be called by any loader used by the manager when the loader errors loading an url.

Source

[link:<https://github.com/mrdoob/three.js/blob/master/src/loaders>LoadingManager.js>]

МАТЕРИАЛЫ

LineBasicMaterial

[Material →](#)

Материал для рисования геометрических фигур в каркасном стиле обычными линиями.

Пример

```
var material = new THREE.LineBasicMaterial( {  
    color: 0xffffffff,  
    linewidth: 1,  
    linecap: 'round', // ignored by WebGLRenderer (игнорируется WebGL  
    linejoin: 'round' // ignored by WebGLRenderer (игнорируется WebGL  
} );
```

Другие примеры

[WebGL / buffergeometry / drawcalls](#)
[WebGL / buffergeometry / lines](#)
[WebGL / buffergeometry / lines / indexed](#)
[WebGL / decals](#)
[WebGL / geometry / nurbs](#)
[WebGL / geometry / shapes](#)
[WebGL / geometry / spline / editor](#)
[WebGL / interactive / buffergeometry](#)
[WebGL / interactive / voxelpainter](#)
[WebGL / lines / colors](#)
[WebGL / lines / cubes](#)
[WebGL / lines / dashed](#)
[WebGL / lines / sphere](#)
[WebGL / lines / splines](#)
[WebGL / materials](#)
[WebGL / physics / rope](#)

Конструктор

`LineBasicMaterial(parameters)`

parameters - объект с одним или несколькими свойствами, определяющими внешний вид материала (дополнительный, необязательный параметр). Здесь можно передать любое

свойство материала (включая любое свойство унаследованное от [Material](#)).

Исключением является свойство [color](#), значение которого можно передать как шестнадцатиричную строку и по умолчанию равно `0xffffffff` (белый цвет). Метод [Color.set\(color \)](#) вызывается внутренне.

Свойства

Для информации об общих свойствах смотрите базовый класс [Material](#).

[.color](#)

[Цвет](#) материала, по умолчанию устанавливается как белый (`0xffffffff`).

[.isLineBasicMaterial](#)

Используется для проверки, является ли данный объект, или производные от него, материалом обычных линий. По умолчанию значение равно `true`.

Не нужно его изменять, так как внутри *three.js* оно используется для оптимизации.

[.lights](#)

Свойство определяет, будет ли освещение влиять на материал. Значением по умолчанию является `false`.

[.linewidth](#)

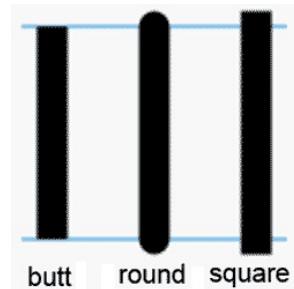
Свойство управляет толщиной линии. Значение по умолчанию `1`.

[.linecap](#)

Определяет внешний вид концов линии. Возможные значения: `butt` (торец), `round` (круглый) и `square` (квадратный). Значение по умолчанию `round` - круглый.

Данное свойство соответствует свойству [linecap в 2D Canvas](#) и игнорируется [WebGL](#) визуализатором (рендерером). Вот эта [статья о linecap](#) на русском языке.

Примечание переводчика: Вот нарисованы три линии с разными значениями `linecap`. Для наглядности добавлены две направляющие. Каждая линия будет начинаться и заканчиваться этими направляющими.
Левая линия будет использовать для `linecap` значение `butt`. Она не будет выходить за направляющие. Средняя линия будет нарисована со значением `round`. За направляющие будут выходить полукруги с диаметром равным толщине линии. Правая линия будет использовать значение `square`. Она будет выходить за направляющие на поля с шириной равной толщине линии и высотой равной половине толщины.



[.linejoin](#)

Определяет внешний вид соединений линии. Возможные значения: `round` - круглый, `bevel` (скос) и `miter` ([митра](#)). Значение по умолчанию `round` - круглый.

Данное свойство соответствует свойству [lineJoin в 2D Canvas](#) и игнорируется [WebGL](#) визуализатором (рендерером). Вот эта [статья о lineJoin на русском языке](#).

Примечание переводчика: `miter` соединяет линии в одной точке, расширяя для этого их границы и заполняя пространство ромбами.
`round` скругляет углы засчет добавления сектора с центром в точке пересечения линий и радиусом равным толщине линии.
`bevel` "срезает" угол, рисуя треугольник с вершинами в точке пересечения линий и крайних точках каждой линии.



Методы

Общие методы смотрите в описании базового класса [Material](#).

Исходники

[LineBasicMaterial.js в этом справочнике](#)

[LineBasicMaterial.js на github.com](#)

[Material](#) →

LineDashedMaterial

Материал для рисования геометрических фигур в каркасном стиле пунктирными линиями.

Пример

```
var material = new THREE.LineDashedMaterial( {  
    color: 0xffffffff,  
    linewidth: 1,  
    scale: 1,  
    dashSize: 3,  
    gapSize: 1,  
} );
```

Другие примеры

[WebGL / lines / dashed](#)
[Canvas / lines /dashed](#)

Конструктор

`LineDashedMaterial(parameters)`

parameters - объект с одним или несколькими свойствами, определяющими внешний вид материала (дополнительный, необязательный параметр). Здесь можно передать любое свойство материала (включая любое свойство унаследованное от [Material](#)).

Исключением является свойство `color`, значение которого можно передать как шестнадцатиричную строку и по умолчанию равно `0xffffffff` (белый цвет). Метод `Color.set(color)` вызывается внутренне.

Свойства

Для информации об общих свойствахсмотрите базовый класс [Material](#).

.color

Цвет материала, по умолчанию устанавливается как белый (`0xffffffff`).

.dashSize

Размер штриха, по умолчанию равен 3. Это сумма размеров самого штриха и зазора между штрихами.

.gapSize

Размер зазора (расстояния) между штрихами, по умолчанию равен 1.

.isLineDashedMaterial

Используется для проверки, является ли данный объект, или производные от него, материалом штриховых линий. По умолчанию значение равно `true`.

Не нужно его изменять, так как внутри `three.js` оно используется для оптимизации.

.lights

Свойство определяет, будет ли освещение влиять на материал. Значением по умолчанию является `false`.

.linewidth

Управляет толщиной линии. Значение по умолчанию 1.

.scale

The scale штриховой части линии.

Методы

Общие методы смотрите в описании базового класса [Material](#).

Исходники

[LineDashedMaterial.js в этом справочнике](#)

[LineDashedMaterial.js на github.com](#)

Material

Абстрактный, базовый класс для материалов.

Материалы описывают внешний вид [объектов](#). Они определены (в основном) независимыми от визуализатора (рендерера), так что не придется переписывать материалы, если вы решили использовать другой визуализатор.

Перечисленные ниже свойства и методы наследуются всеми другими типами материала (хотя и могут иметь разные значения по умолчанию).

Конструктор

`Material()`

Конструктор создает общий (универсальный) материал.

Свойства

[`.alphaTest`](#)

Устанавливает значение альфа-канала (определяет прозрачность), которое будет использоваться при прохождении проверки состояния альфа-канала. Если непрозрачность (свойство [opacity](#)) будет меньше заданного здесь значения, материал не будет отображаться.

Значение по умолчанию 0.

[`.blendDst`](#)

Целевое смещивание. Значением по умолчанию является [OneMinusSrcAlphaFactor](#). Все возможные значения смотрите в описании констант [целевых факторов смещивания](#).

Для того, чтобы это свойство как-то повлияло на внешний вид, свойство [blending](#) должно быть установлено как [CustomBlending](#).

[`.blendDstAlpha`](#)

Прозрачность для [blendDst](#). Значением по умолчанию является `null`.

[`.blendEquation`](#)

Уравнение смешивания, используемое при применении смешивания. Значением по умолчанию является [AddEquation](#). Все возможные значения смотрите в описании констант [уравнений смешивания](#).

Для того, чтобы это свойство как-то повлияло на внешний вид, свойство [blending](#) должно быть установлено как [CustomBlending](#).

[.blendEquationAlpha](#)

Прозрачность для [blendEquation](#). Значением по умолчанию является null.

[.blending](#)

Свойство определяет какое смешивание будет использоваться при отображении объектов с этим материалом. При использовании настраиваемых свойств [blendSrc](#), [blendDst](#) или [blendEquation](#), оно должно быть установлено как [CustomBlending](#).

Все возможные значения смотрите в описании констант [режима смешивания](#). Значением по умолчанию является [NormalBlending](#).

[.blendSrc](#)

Исходное смешивание. Значением по умолчанию является [SrcAlphaFactor](#). Все возможные значения смотрите в описании констант [режима смешивания](#).

Для того, чтобы это свойство как-то повлияло на внешний вид, свойство [blending](#) должно быть установлено как [CustomBlending](#).

[.blendSrcAlpha](#)

Прозрачность для [blendSrc](#). Значением по умолчанию является null.

[.clipIntersection](#)

Изменяет поведение плоскостей отсечения так, что обрезается только их пересечение, а не их соединение. Changes the behavior

of clipping planes so that only their intersection is clipped, rather than their union. Значение по умолчанию `false`.

.clippingPlanes

User-defined clipping planes specified as THREE.Plane objects in world space. These planes apply to the objects this material is attached to. Points in space whose signed distance to the plane is negative are clipped (not rendered). Смотрите пример [WebGL / clipping /intersection](#). Значением по умолчанию является `null`.

.clipShadows

Свойство определяет, обрезать ли тени в соответствии с плоскостями отсечения, указанными на этом материале. Defines whether to clip shadows according to the clipping planes specified on this material. Значение по умолчанию `false`.

.colorWrite

Будет ли отображаться цвет материала. Это свойство можно использовать в соединении со свойством сетки (mesh) [renderOrder](#) для создания невидимых объектов, закрывающих другие объекты. Whether to render the material's color. This can be used in conjunction with a mesh's `renderOrder` property to create invisible objects that occlude other objects. Значение по умолчанию `true`.

.customDepthMaterial

Custom depth material to be used by this material when rendering to the depth map. Для правильного отображения теней при направленном ([DirectionalLight](#)) или прожекторном ([SpotLight](#)) освещении, требуется указывать `customDepthMaterial`, если было:

- a. изменение позиций вершин в шейдере вершин,
- b. использование карты смещения,
- c. использование карты прозрачности (alpha map) с проверкой прозрачности ([alphaTest](#)), или
- d. использовалась прозрачная текстура с проверкой

прозрачности ([alphaTest](#)).

Значением по умолчанию является `undefined`.

[.customDistanceMaterial](#)

Тоже самое, что и `customDepthMaterial`, но используется с точечным освещением ([PointLight](#)). Значением по умолчанию является `undefined`.

[.defines](#)

Custom defines to be injected into the shader. These are passed in form of an object literal, with key/value pairs. {
MY_CUSTOM_DEFINE: " ", PI2: `Math.PI * 2` }. The pairs are defined in both vertex and fragment shaders. Значением по умолчанию является `undefined`.

[.depthFunc](#)

Which depth function to use. Default is `LessEqualDepth`. See the depth mode constants for all possible values.

[.depthTest](#)

Whether to have depth test enabled when rendering this material.
Значение по умолчанию `true`.

[.depthWrite](#)

Whether rendering this material has any effect on the depth buffer.
Значение по умолчанию `true`.
When drawing 2D overlays it can be useful to disable the depth writing in order to layer several things together without creating z-index artifacts.

[.dithering](#)

Whether to apply dithering to the color to remove the appearance of banding. Значение по умолчанию `false`.

[.flatShading](#)

Define whether the material is rendered with flat shading. Значение

по умолчанию `false`.

[.fog](#)

Свойство определяет, будет ли материал подвержен влиянию тумана. Значение по умолчанию `true`.

[.id](#)

Уникальное число-идентификатор для данного экземпляра материала.

[.isMaterial](#)

Используется для проверки, является ли данный объект, или производные от него, материалами. По умолчанию значение равно `true`.

Не нужно его изменять, так как внутри ***three.js*** оно используется для оптимизации.

[.lights](#)

Свойство определяет, будет ли освещение влиять на материал. Значение по умолчанию `true`.

[.name](#)

Дополнительное, необязательное название объекта (необязательно быть уникальным). По умолчанию это пустая строка ("").

[.needsUpdate](#)

Specifies that the material needs to be updated at the WebGL level. Set it to `true` if you made changes that need to be reflected in WebGL. При создании экземпляра нового материала, данное свойство автоматически устанавливается как `true`.

[.opacity](#)

Число с плавающей запятой (float) в диапазоне `0.0 - 1.0`, показывающее насколько прозрачен материал. Значение `0.0` указывает на полную прозрачность, а `1.0` - на полную

непрозрачность. Значение по умолчанию равно 1.0. Если свойство [transparent](#) материала не установлено как `true`, материал будет оставаться полностью непрозрачным и данное значение будет влиять только на его цвет.

[.overdraw](#)

Amount of triangle expansion at draw time. This is a workaround for cases when gaps appear between triangles when using CanvasRenderer. 0.5 tends to give good results across browsers. Значение по умолчанию 0.

[.polygonOffset](#)

Свойство устанавливает, будет ли использоваться смещение полигонов. Значение по умолчанию `false`. Это свойство соответствует свойству WebGL `GL_POLYGON_OFFSET_FILL`.

[.polygonOffsetFactor](#)

Устанавливает [коэффициент масштабирования](#) смещения полигонов. Значение по умолчанию 0.

[.polygonOffsetUnits](#)

Устанавливает блоки смещения полигонов. Значение по умолчанию 0.

[.precision](#)

Свойство переопределяет точность по умолчанию визуализатора для данного материала. Может быть `highp`, `mediump` или `lowp`. Значение по умолчанию равно `null`.

[.premultipliedAlpha](#)

Whether to premultiply the alpha (transparency) value. Посмотрите отличия в примере [WebGL / Materials / Transparency](#). Значение по умолчанию `false`.

[.side](#)

Свойство определяет сторону грани, которая будет отображаться,

- передняя, задняя или обе. Значение по умолчанию [THREE.FrontSide](#). Остальные варианты это [THREE.BackSide](#) и [THREE.DoubleSide](#).

[.transparent](#)

Определяет будет ли данный материал прозрачным или нет. Данное свойство влияет на процесс визуализации, так как прозрачные объекты нуждаются в специальной обработке и отображаются после непрозрачных объектов.

Если данное свойство установлено как true, степень прозрачности материала определяется настройкой значения свойства [opacity](#).

Значение по умолчанию false.

[.type](#)

Значением этого свойства является строка Material. Она не должна изменяться и может использоваться для поиска на сцене всех объектов данного типа.

[.uuid](#)

[UUID](#) данного экземпляра материала. Он присваивается автоматически и не должен быть изменен.

Вот [статья Википедии про UUID](#) на русском языке.

[.vertexColors](#)

Свойство определяет, как окрашиваются вершины, путем задания способа заполнения атрибута цвета. Возможными значениями являются [THREE.NoColors](#), [THREE.FaceColors](#) и [THREE.VertexColors](#). Значение по умолчанию THREE.NoColors.

[.visible](#)

Свойство определяет, будет ли данный материал видимым. Значение по умолчанию true.

[.userData](#)

Этот объект может использоваться для хранения

пользовательских данных о материале. В нем нельзя хранить ссылки на функции, так как они не будут клонироваться.

Методы

В этом классе доступны методы [EventDispatcher](#).

[.clone](#) ()

Возвращает новый материал с теми же параметрами как у данного материала.

[.copy](#) ([material](#))

Метод копирует параметры из переданного материала (параметр [material](#)) в данный материал.

[.dispose](#) ()

This disposes the material. Textures of a material don't get disposed. These needs to be disposed by [Texture](#).

[.setValues](#) ([values](#))

| [values](#) -- контейнер с параметрами.

Метод устанавливает свойства на основе значений, переданных в аргументе.

[.toJSON](#) ([meta](#))

| [meta](#) -- объект, содержащий [метаданные](#) для материала, вроде текстур или изображений.

Конвертирует материал в формат [JSON](#) для [three.js](#).

[.update](#) ()

Метод вызывает для материала

[.dispatchEvent](#) ({ type: '[update](#)' }).

Исходники

[Material.js в этом справочнике](#)

[Material.js на github.com](#)

MeshBasicMaterial

Материал для рисования геометрических элементов в A material for drawing geometries in a simple shaded (flat or wireframe) way.

На этот материал не влияет освещение. This material is not affected by lights. A material for drawing geometries in a simple shaded (flat or wireframe) way.

По умолчанию будет отображаться в виде плоских многоугольников. Чтобы нарисовать сетку в виде каркаса, просто установите свойство [wireframe](#) как true.

Конструктор

```
MeshBasicMaterial( parameters )
```

[parameters](#) - объект с одним или несколькими свойствами, определяющими внешний вид материала (дополнительный, необязательный параметр). Здесь можно передать любое свойство материала (включая любое свойство унаследованное от [Material](#)).

Исключением является свойство [color](#), значение которого можно передать как шестнадцатиричную строку и по умолчанию равно 0xffffffff (белый цвет). Метод [Color.set\(color \)](#) вызывается внутренне.

Свойства

Для информации об общих свойствахсмотрите базовый класс [Material](#).

```
.alphaMap
```

Альфа-карта представляет собой текстуру в оттенках серого, которая управляет непрозрачностью по всей поверхности (черный цвет: полностью прозрачный; белый: полностью непрозрачный). Значением по умолчанию является null.

Используется только цвет текстуры, альфа-канал (если он

существует) игнорируется. Для текстур [RGB](#) и [RGBA](#) визуализатор [WebGL](#) будет использовать канал зеленого цвета For RGB and RGBA textures, the WebGL renderer will use the green channel when sampling this texture due to the extra bit of precision provided for green in DXT-compressed and uncompressed RGB 565 formats. из-за дополнительного бита точности, предоставляемого зеленым каналом, в сжатом DXT и несжатом RGB 565 форматах. Текстуры только с каналом яркости и яркости/прозрачности (альфа-канал) также будут работать как и ожидалось. Luminance-only and luminance/alpha textures will also still work as expected.

[.aoMap](#)

Канал красного цвета используется в данной текстуре в качестве ambient occlusion map. Значением по умолчанию является null. The aoMap requires a second set of UVs, and consequently will ignore the repeat and offset Texture properties.

Статья в русской Википедии о [эффекте перекрытия](#) (модели затенения) при фоновом освещении. А [это](#) в английской.

[.aoMapIntensity](#)

Интенсивность затенения при общем (фоновом) освещении. Значение по умолчанию равно 1. Нулевое значение означает отсутствие эффекта перекрытия (затенения).

[.color](#)

[Цвет](#) материала, по умолчанию установлен как белый (0xffffffff - white).

[.combine](#)

How to combine the result of the surface's color with the environment map, if any.

Возможными значениями являются [THREE.Multiply](#) (значение по умолчанию), [THREE.MixOperation](#) и [THREE.AddOperation](#). Если выбрано смешивание (THREE.MixOperation), то свойство [reflectivity](#) будет использовано для смешивания двух цветов. If mix is chosen, the [reflectivity](#) is used to blend between the two

colors.

.isMeshBasicMaterial

Используется для проверки, является ли данный объект, или производные от него, основными материалами сетки. По умолчанию значение равно true.

Не нужно его изменять, так как внутри **three.js** оно используется для оптимизации.

.envMap

Карта окружения (environment map). Значением по умолчанию является null.

.lightMap

Карта освещения. Значением по умолчанию является null. Свойство lightMap требует второго набора UV координат и, следовательно, будет игнорировать свойства repeat и offset текстуры.

.lightMapIntensity

Intensity of the baked light. Значение по умолчанию равно 1.

.lights

Свойство определяет, будет ли материал зависеть от освещения. Значением по умолчанию является false.

.map

Карта цвета. Значением по умолчанию является null.

.morphTargets

Define whether the material uses morphTargets. Значением по умолчанию является false.

.reflectivity

How much the environment map affects the surface; also see .combine. The default value is 1 and the valid range is between 0 (no

reflections) and 1 (full reflections).

.refractionRatio

The index of refraction (IOR) of air (approximately 1) divided by the index of refraction of the material. It is used with environment mapping modes THREE.CubeRefractionMapping and THREE.EquirectangularRefractionMapping. The refraction ratio should not exceed 1. Значение по умолчанию равно 0.98.

.skinning

Define whether the material uses skinning. Значением по умолчанию является false.

.specularMap

Карта отражений используемая материалом. Значением по умолчанию является null.

.wireframe

Свойство позволяет отобразить геометрическую фигуру в виде каркаса. Значением по умолчанию является false (т.е. каркас не виден и визуализация происходит при помощи плоских многоугольников).

.wireframeLinecap

Определяет внешний вид концов линии. Возможные значения: butt (торец), round (круглый) и square (квадратный). Значение по умолчанию round - круглый.

Данное свойство соответствует свойству [linecap в 2D Canvas](#) и игнорируется [WebGL](#) визуализатором (рендерером). Вот эта [статья о linecap](#) на русском языке.

.wireframeLinejoin

Определяет внешний вид соединений линии. Возможные значения: round - круглый, bevel (скос) и miter ([митра](#)). Значение по умолчанию round - круглый.

Данное свойство соответствует свойству [lineJoin в 2D Canvas](#) и

игнорируется [WebGL](#) визуализатором (рендерером). Вот эта [статья о lineJoin](#) на русском языке.

[.wireframeLinewidth](#)

Управляет толщиной линии каркаса. Значение по умолчанию равно 1.

Методы

Для информации об общих методах смотрите базовый класс [Material](#).

Исходники

[MeshBasicMaterial.js в этом справочнике](#)

[MeshBasicMaterial.js на github.com](#)

[Material](#) →

MeshDepthMaterial

Материал для рисования геометрических элементов фигуры в глубину. Глубина зависит от ближней и дальней плоскостей отсечения камеры. A material for drawing geometry by depth. Depth is based off of the camera near and far plane. White is nearest, black is farthest.

Конструктор

`MeshDepthMaterial(parameters)`

[parameters](#) - объект с одним или несколькими свойствами, определяющими внешний вид материала (дополнительный, необязательный параметр). Здесь можно передать любое свойство материала (включая любое свойство унаследованное от [Material](#)).

Свойства

Для информации об общих свойствах смотрите базовый класс

Material.

`.alphaMap`

The alpha map is a grayscale texture that controls the opacity across the surface (black: fully transparent; white: fully opaque). Default is null.

Only the color of the texture is used, ignoring the alpha channel if one exists. For RGB and RGBA textures, the WebGL renderer will use the green channel when sampling this texture due to the extra bit of precision provided for green in DXT-compressed and uncompressed RGB 565 formats. Luminance-only and luminance/alpha textures will also still work as expected.

`.depthPacking`

Encoding for depth packing. Default is BasicDepthPacking.

`.displacementMap`

The displacement map affects the position of the mesh's vertices. Unlike other maps which only affect the light and shade of the material the displaced vertices can cast shadows, block other objects, and otherwise act as real geometry. The displacement texture is an image where the value of each pixel (white being the highest) is mapped against, and repositions, the vertices of the mesh.

`.displacementScale`

How much the displacement map affects the mesh (where black is no displacement, and white is maximum displacement). Without a displacement map set, this value is not applied. Значение по умолчанию равно 1.

`.displacementBias`

The offset of the displacement map's values on the mesh's vertices. Without a displacement map set, this value is not applied. Значение по умолчанию равно 0.

`.fog`

Свойство определяет, будет ли материал подвержен влиянию тумана. Значением по умолчанию является `false`.

[.isMeshDepthMaterial](#)

Используется для проверки, является ли данный объект, или производные от него, материалами сетки с прорисовкой глубины. По умолчанию значение равно `true`.

Не нужно его изменять, так как внутри `three.js` оно используется для оптимизации.

[.lights](#)

Свойство определяет, будет ли освещение влиять на материал. Значением по умолчанию является `false`.

[.map](#)

Карта цвета. Значением по умолчанию является `null`.

[.morphTargets](#)

Свойство определяет, будет ли материал использовать `morphTargets`. Значением по умолчанию является `false`.

[.skinning](#)

Свойство определяет, будет ли материал использовать `skinning`. Значением по умолчанию является `false`.

[.wireframe](#)

Свойство позволяет отобразить геометрическую фигуру в виде каркаса. Значением по умолчанию является `false` (т.е. каркас не виден и объект отображен плавно затушеванным).

[.wireframeLineWidth](#)

Управляет толщиной линии каркаса. Значение по умолчанию равно 1.

Методы

Для информации об общих методах смотрите базовый класс

Material.

Исходники

[MeshDepthMaterial.js в этом справочнике](#)

[MeshDepthMaterial.js на github.com](#)

[Material →](#)

MeshLambertMaterial

Материал для матовых поверхностей, без A material for non-shiny surfaces, without specular highlights.

The material uses a non-physically based Lambertian model for calculating reflectance. This can simulate some surfaces (such as untreated wood or stone) well, but cannot simulate shiny surfaces with specular highlights (such as varnished wood).

Shading is calculated using a Gouraud shading model. This calculates shading per vertex (i.e. in the vertex shader) and interpolates the results over the polygon's faces.

Due to the simplicity of the reflectance and illumination models, performance will be greater when using this material over the MeshPhongMaterial, MeshStandardMaterial or MeshPhysicalMaterial, at the cost of some graphical accuracy.

Конструктор

`MeshLambertMaterial(parameters)`

parameters - объект с одним или несколькими свойствами, определяющими внешний вид материала (дополнительный, необязательный параметр). Здесь можно передать любое свойство материала (включая любое свойство унаследованное от [Material](#)).

Исключением является свойство `color`, значение которого можно передать как шестнадцатиричную строку и по умолчанию равно `0xffffffff` (белый цвет). Метод `Color.set(color)` вызывается внутренне.

Свойства

Для информации об общих свойствах смотрите базовый класс [Material](#).

[property:Color color]

Diffuse color of the material. Default is white.

[property:Texture map]

Set color texture map. Default is null.

[property:Texture lightMap]

Set light map. Default is null. The lightMap requires a second set of UVs.

[property:Float lightMapIntensity]

TODO

[property:Texture aoMap]

Set ambient occlusion map. Default is null. The aoMap requires a second set of UVs.

[property:Float aoMapIntensity]

TODO

[property:Color emissive]

Emissive (light) color of the material, essentially a solid color unaffected by other lighting. Default is black.

[property:Texture emissiveMap]

Set emissive (glow) map. Default is null. The emissive map color is modulated by the emissive color and the emissive intensity. If you have an emissive map, be sure to set the emissive color to something other than black.

[property:Float emissiveIntensity]

Intensity of the emissive light. Modulates the emissive color. Default is 1.

[property:Texture specularMap]

Since this material does not have a specular component, the specular value affects only how much of the environment map affects the surface. Default is null.

[property:Texture alphaMap]

The alpha map is a grayscale texture that controls the opacity across

the surface (black: fully transparent; white: fully opaque). Default is null.

Only the color of the texture is used, ignoring the alpha channel if one exists. For RGB and RGBA textures, the [page:WebGLRenderer WebGL] renderer will use the green channel when sampling this texture due to the extra bit of precision provided for green in DXT-compressed and uncompressed RGB 565 formats. Luminance-only and luminance/alpha textures will also still work as expected.

```
[property:TextureCube envMap]
```

Set env map. Default is null.

```
[property:Integer combine]
```

How to combine the result of the surface's color with the environment map, if any.

Options are [page:Textures THREE.Multiply] (default), [page:Textures THREE.MixOperation], [page:Textures THREE.AddOperation]. If mix is chosen, the reflectivity is used to blend between the two colors.

```
[property:Float reflectivity]
```

How much the environment map affects the surface; also see "combine".

```
[property:Float refractionRatio]
```

The index of refraction for an environment map using [page:Textures THREE.CubeRefractionMapping]. Default is *0.98*.

```
[property:Boolean fog]
```

Define whether the material color is affected by global fog settings. Default is *true*.

This setting might not have any effect when used with certain renderers. For example, it is ignored with the [page:CanvasRenderer Canvas] renderer, but does work with the [page:WebGLRenderer WebGL] renderer.

```
[property:Boolean wireframe]
```

Whether the triangles' edges are displayed instead of surfaces.

Default is *false*.

```
[property:Float wireframeLineWidth]
```

Line thickness for wireframe mode. Default is *1.0*.

Due to limitations in the [ANGLE layer](#), on Windows platforms linewidth will always be 1 regardless of the set value.

[property:String wireframeLinecap]

Define appearance of line ends. Possible values are "butt", "round" and "square". Default is 'round'.

This setting might not have any effect when used with certain renderers. For example, it is ignored with the [page:WebGLRenderer WebGL] renderer, but does work with the [page:CanvasRenderer Canvas] renderer.

[property:String wireframeLinejoin]

Define appearance of line joints. Possible values are "round", "bevel" and "miter". Default is 'round'.

This setting might not have any effect when used with certain renderers. For example, it is ignored with the [page:WebGLRenderer WebGL] renderer, but does work with the [page:CanvasRenderer Canvas] renderer.

[property:Integer vertexColors]

Define how the vertices gets colored. Possible values are THREE.NoColors, THREE.FaceColors and THREE.VertexColors. Default is THREE.NoColors.

This setting might not have any effect when used with certain renderers. For example, it is ignored with the [page:CanvasRenderer Canvas] renderer, but does work with the [page:WebGLRenderer WebGL] renderer.

[property:Boolean skinning]

Define whether the material uses skinning. Default is *false*.

[property:Boolean morphTargets]

Define whether the material uses morphTargets. Default is *false*.

[property:boolean morphNormals]

Defines whether the material uses morphNormals. Set as true to pass morphNormal attributes from the [page:Geometry] to the shader. Default is *false*.

Методы

Для информации об общих методах смотрите базовый класс [Material](#).

Исходники

[MeshLambertMaterial.js в этом справочнике](#)
[MeshLambertMaterial.js на github.com](#)

[Material →](#)

MeshNormalMaterial

Материал, который сопоставляет векторы нормалей в RGB цвета. A material that maps the normal vectors to RGB colors.

Конструктор

`MeshNormalMaterial(parameters)`

[parameters](#) - объект с одним или несколькими свойствами, определяющими внешний вид материала (дополнительный, необязательный параметр). Здесь можно передать любое свойство материала (включая любое свойство унаследованное от [Material](#)).

Свойства

Для информации об общих свойствах смотрите базовый класс [Material](#).

[.fog](#)

Свойство устанавливает, будет или нет материал подвержен влиянию тумана. По умолчанию значение равно `false`.

[.isMeshNormalMaterial](#)

Используется для проверки, является ли данный объект, или производные от него, материалом нормалей сетки. По умолчанию значение равно `true`.

Не нужно его изменять, так как внутри [three.js](#) оно используется для оптимизации.

[.lights](#)

Свойство устанавливает, будет или нет освещение влиять на материал. По умолчанию значение равно `false`.

.morphTargets

Define whether the material uses morphTargets. По умолчанию значение равно false.

.wireframe

Свойство позволяет отобразить геометрическую фигуру в виде каркаса. Значением по умолчанию является false (т.е. каркас не виден и объект ровно закрашен).

.wireframeLinewidth

Свойство управляет толщиной линий каркаса. Значение по умолчанию равно 1.

Из-за ограничений в [слое ANGLE](#) в операционных системах на основе Windows ширина линии всегда будет равна 1, независимо от установленного значения.

Примечание переводчика: ANGLE (Almost Native Graphics Layer Engine - практически встроенный движок графического слоя, а перевод самой аббревиатуры ANGLE - угол) это графический движок слоя абстракций, разработанный компанией Google, с открытым исходным кодом под лицензией BSD. Этот API в основном предназначен для обеспечения высокопроизводительной совместимости OpenGL с компьютерами Windows и веб-браузерами вроде Chromium, путем перевода вызовов OpenGL в Direct3D, который гораздо лучше поддерживается драйверами. Для ANGLE существует два внутренних визуализатора: самый старый использует Direct3D 9.0c, в то время как новый использует Direct3D 11.

из статьи англ. Википедии о [ANGLE](#), перевод мой.

Методы

Для информации об общих методах смотрите базовый класс [Material](#).

Исходники

[MeshNormalMaterial.js в этом справочнике](#)
[MeshNormalMaterial.js на github.com](#)

MeshPhongMaterial

A material for shiny surfaces, evaluated per pixel.

Конструктор

```
MeshPhongMaterial( [page:Object parameters] )
```

parameters -- an object with one or more of the material's properties defining the material's appearance.

color — geometry color in hexadecimal. Default is 0xffffffff.

specular — Set specular color. Default is 0x111111 .

shininess — Set shininess Default is 30.

map — Set texture map. Default is null.

lightMap — Set light map. Default is null.

lightMapIntensity — Set light map intensity. Default is 1.

aoMap — Set ao map. Default is null.

aoMapIntensity — Set ao map intensity. Default is 1.

emissive - Set emissive color. Default is 0x000000.

emissiveMap — Set emissive map. Default is null.

emissiveIntensity — Set emissive map intensity. Default is 1.

bumpMap — Set bump map. Default is null.

bumpScale — Set bump map scale. Default is 1.

normalMap — Set normal map. Default is null.

normalScale — Set normal map scale. Default is (1, 1).

displacementMap — Set displacement map. Default is null.

displacementScale — Set displacement scale. Default is 1.

displacementBias — Set displacement offset. Default is 0.

specularMap — Set specular map. Default is null.

alphaMap — Set alpha map. Default is null.

envMap — Set env map. Default is null.

combine — Set combine operation. Default is THREE.MultiplyOperation.

reflectivity — Set reflectivity. Default is 1.

refractionRatio — Set refraction ratio. Default is 0.98.

fog — Define whether the material color is affected by global fog settings. Default is true.

shading — Define shading type. Default is THREE.SmoothShading.

wireframe — render geometry as wireframe. Default is false.

wireframeLineWidth — Line thickness. Default is 1.

`wireframeLinecap` — Define appearance of line ends. Default is 'round'.
`wireframeLinejoin` — Define appearance of line joints. Default is 'round'.
`vertexColors` — Define how the vertices gets colored. Default is `THREE.NoColors`.

`skinning` — Define whether the material uses skinning. Default is false.
`morphTargets` — Define whether the material uses morphTargets. Default is false.

`morphNormals` — Define whether the material uses morphNormals.
Default is false.

Example:

```
materials.push( new THREE.MeshPhongMaterial( { color: 0xddddddd,  
specular: 0x009900, shininess: 30, shading: THREE.FlatShading } ) );
```

Свойства

See the base [page:Material] class for common properties.

[property:Color `color`]

Diffuse color of the material. Default is white.

[property:Color `specular`]

Specular color of the material, i.e., how shiny the material is and the color of its shine. Setting this the same color as the diffuse value (times some intensity) makes the material more metallic-looking; setting this to some gray makes the material look more plastic. Default is dark gray.

[property:Float `shininess`]

How shiny the specular highlight is; a higher value gives a sharper highlight. Default is *30*.

[property:Texture `map`]

Set color texture map. Default is null. The texture map color is modulated by the diffuse color.

[property:Texture `lightMap`]

Set light map. Default is null. The lightMap requires a second set of UVs.

[property:Float `lightMapIntensity`]

TODO

[property:Texture `aoMap`]

Set ambient occlusion map. Default is null. The aoMap requires a second set of UVs.

[property:Float `aoMapIntensity`]

TODO

[property:Color emissive]

Emissive (light) color of the material, essentially a solid color unaffected by other lighting. Default is black.

[property:Texture emissiveMap]

Set emissive (glow) map. Default is null. The emissive map color is modulated by the emissive color and the emissive intensity. If you have an emissive map, be sure to set the emissive color to something other than black.

[property:Float emissiveIntensity]

Intensity of the emissive light. Modulates the emissive color. Default is 1.

[property:Texture bumpMap]

The texture to create a bump map. The black and white values map to the perceived depth in relation to the lights. Bump doesn't actually affect the geometry of the object, only the lighting. If a normal map is defined this will be ignored.

[property:Float bumpScale]

How much the bump map affects the material. Typical ranges are 0-1.

Default is 1.

[property:Texture normalMap]

The texture to create a normal map. The RGB values affect the surface normal for each pixel fragment and change the way the color is lit.

Normal maps do not change the actual shape of the surface, only the lighting.

[property:Vector2 normalScale]

How much the normal map affects the material. Typical ranges are 0-1.

Default is (1,1).

[property:Texture displacementMap]

The displacement map affects the position of the mesh's vertices. Unlike other maps which only affect the light and shade of the material the displaced vertices can cast shadows, block other objects, and otherwise act as real geometry. The displacement texture is an image where the value of each pixel (white being the highest) is mapped against, and repositions, the vertices of the mesh.

[property:Float displacementScale]

How much the displacement map affects the mesh (where black is no displacement, and white is maximum displacement). Without a

displacement map set, this value is not applied. Default is 1.

[property:Float displacementBias]

The offset of the displacement map's values on the mesh's vertices.

Without a displacement map set, this value is not applied. Default is 0.

[property:Texture specularMap]

The specular map value affects both how much the specular surface highlight contributes and how much of the environment map affects the surface. Default is null.

[property:Texture alphaMap]

The alpha map is a grayscale texture that controls the opacity across the surface (black: fully transparent; white: fully opaque). Default is null.

Only the color of the texture is used, ignoring the alpha channel if one exists. For RGB and RGBA textures, the [page:WebGLRenderer WebGL] renderer will use the green channel when sampling this texture due to the extra bit of precision provided for green in DXT-compressed and uncompressed RGB 565 formats. Luminance-only and luminance/alpha textures will also still work as expected.

[property:TextureCube envMap]

Set env map. Default is null.

[property:Integer combine]

How to combine the result of the surface's color with the environment map, if any.

Options are [page:Textures THREE.MultiplyOperation] (default), [page:Textures THREE.MixOperation], [page:Textures THREE.AddOperation]. If mix is chosen, the reflectivity is used to blend between the two colors.

[property:Float reflectivity]

How much the environment map affects the surface; also see "combine".

[property:Float refractionRatio]

The index of refraction for an environment map using [page:Textures THREE.CubeRefractionMapping]. Default is *0.98*.

[property:Boolean fog]

Define whether the material color is affected by global fog settings.

Default is *true*.

This setting might not have any effect when used with certain renderers.

For example, it is ignored with the [page:CanvasRenderer Canvas]

renderer, but does work with the [page:WebGLRenderer WebGL] renderer.

[property:Integer shading]

How the triangles of a curved surface are rendered: as a smooth surface, as flat separate facets, or no shading at all.

Options are [page:Materials THREE.SmoothShading] (default), [page:Materials THREE.FlatShading].

[property:Boolean wireframe]

Whether the triangles' edges are displayed instead of surfaces. Default is *false*.

[property:Float wireframeLineWidth]

Line thickness for wireframe mode. Default is *1.0*.

Due to limitations in the [ANGLE layer](#), on Windows platforms linewidth will always be 1 regardless of the set value.

[property:String wireframeLinecap]

Define appearance of line ends. Possible values are "butt", "round" and "square". Default is 'round'.

This setting might not have any effect when used with certain renderers. For example, it is ignored with the [page:WebGLRenderer WebGL] renderer, but does work with the [page:CanvasRenderer Canvas] renderer.

[property:String wireframeLinejoin]

Define appearance of line joints. Possible values are "round", "bevel" and "miter". Default is 'round'.

This setting might not have any effect when used with certain renderers. For example, it is ignored with the [page:WebGLRenderer WebGL] renderer, but does work with the [page:CanvasRenderer Canvas] renderer.

[property:Integer vertexColors]

Define how the vertices gets colored. Possible values are THREE.NoColors, THREE.FaceColors and THREE.VertexColors. Default is THREE.NoColors.

This setting might not have any effect when used with certain renderers. For example, it is ignored with the [page:CanvasRenderer Canvas] renderer, but does work with the [page:WebGLRenderer WebGL] renderer.

[property:Boolean] skinning]

Define whether the material uses skinning. Default is *false*.

[property:Boolean] morphTargets]

Define whether the material uses morphTargets. Default is *false*.

[property:boolean] morphNormals]

Defines whether the material uses morphNormals. Set as true to pass morphNormal attributes from the [page:Geometry] to the shader. Default is *false*.

Методы

Исходники

[MeshPhongMaterial.js в этом справочнике](#)

[MeshPhongMaterial.js на github.com](#)

[Material](#) → [MeshStandardMaterial](#) →

MeshPhysicalMaterial

Расширение [MeshStandardMaterial](#), позволяющее увеличить регулировку способности к отражению. reflectivity.

Обратите внимание, что при использовании этого материала, для достижения наилучших результатов, всегда следует указывать карту окружения. Note that for best results you should always specify an environment map when using this material.

Примеры

[webgl / materials / variations / physical](#)

[webgl / materials / reflectivity](#)

Конструктор

`MeshPhysicalMaterial(parameters)`

parameters - объект с одним или несколькими свойствами, определяющими внешний вид материала (дополнительный, необязательный параметр). Здесь можно передать любое

свойство материала (включая любое свойство унаследованное от [Material](#) и [MeshStandardMaterial](#)).

Исключением является свойство [color](#), значение которого можно передать как шестнадцатиричную строку и по умолчанию равно `0xffffffff` (белый цвет). Метод [Color.set\(color \)](#) вызывается внутренне.

Свойства

Для информации об общих свойствах смотрите базовые классы [Material](#) и [MeshStandardMaterial](#).

[.clearCoat](#)

ClearCoat level, from 0.0 to 1.0. Значение по умолчанию равно 0.0.

[.clearCoatRoughness](#)

How rough the clearCoat appears, from 0.0 to 1.0. Значение по умолчанию равно 0.0.

[.isMeshPhysicalMaterial](#)

Используется для проверки, является ли данный объект, или производные от него, физическим материалом сетки. По умолчанию значение равно `true`.

Не нужно его изменять, так как внутри [three.js](#) оно используется для оптимизации.

[.defines](#)

Объект в виде:

```
{ 'PHYSICAL': '' };
```

Он используется [WebGLRenderer](#)'ом для выбора шейдеров.

[.reflectivity](#)

Степень отражения, возможны значения от 0.0 до 1.0. Значение по умолчанию равно 0.5.

Свойство моделирует способность к отражению света неметаллических материалов. Оно не работает если свойство

[metalness](#) равно 1.0.

Методы

Для информации об общих методах смотрите базовые классы [Material](#) и [MeshStandardMaterial](#).

Исходники

[MeshPhysicalMaterial.js в этом справочнике](#)

[MeshPhysicalMaterial.js на github.com](#)

[Material](#) →

MeshStandardMaterial

Стандартный, физически обоснованный материал, using Metallic-Roughness workflow.

Physically based rendering (PBR) has recently become the standard in many 3D applications, such as Unity, Unreal and 3D Studio Max. This approach differs from older approaches in that instead of using approximations for the way in which light interacts with a surface, a physically correct model is used. The idea is that, instead of tweaking materials to look good under specific lighting, a material can be created that will react 'correctly' under all lighting scenarios. In practice this gives a more accurate and realistic looking result than the MeshLambertMaterial or MeshPhongMaterial, at the cost of being somewhat more computationally expensive. Shading is calculated in the same way as for the MeshPhongMaterial, using a Phong shading model. This calculates shading per pixel (i.e. in the fragment shader, AKA pixel shader) which gives more accurate results than the Gouraud model used by MeshLambertMaterial, at the cost of some performance. Note that for best results you should always specify an environment map when using this material. For a non-technical introduction to the concept of PBR and how to set up a PBR material, check out these articles by the people at marmoset:

- [Basic Theory of Physically Based Rendering](#)

- [Physically Based Rendering and You Can Too](#)

Технические подробности подхода, используемого в *three.js* (и большинстве других систем PBR), можно найти в [этой статье](#) (pdf) от диснеевской студии анимации, написанную Брентом Барли (Brent Burley).

Конструктор

```
MeshStandardMaterial( parameters )
```

[parameters](#) - объект с одним или несколькими свойствами, определяющими внешний вид материала (дополнительный, необязательный параметр). Здесь можно передать любое свойство материала (включая любое свойство унаследованное от [Material](#)).

Исключением является свойство [color](#), значение которого можно передать как шестнадцатиричную строку и по умолчанию равно `0xffffffff` (белый цвет). Метод [Color.set\(color \)](#) вызывается внутренне.

Свойства

Для информации об общих свойствахсмотрите базовый класс [Material](#).

```
.alphaMap
```

Альфа-карта представляет собой текстуру в оттенках серого, которая управляет непрозрачностью по всей поверхности (черный цвет: полностью прозрачный; белый: полностью непрозрачный). Значением по умолчанию является `null`. Используется только цвет текстуры, альфа-канал (если он существует) игнорируется. Для текстур RGB и RGBA визуализатор WebGL будет использовать канал зеленого цвета Only the color of the texture is used, ignoring the alpha channel if one exists. For RGB and RGBA textures, the WebGL renderer will use the green channel when sampling this texture due to the extra bit of precision provided for green in DXT-compressed and uncompressed RGB 565 formats. Luminance-only and luminance/alpha textures will

also still work as expected.

.aoMap

Канал красного цвета используется в данной структуре в качестве The red channel of this texture is used as the ambient occlusion map. Значением по умолчанию является null. The aoMap requires a second set of UVs, and consequently will ignore the repeat and offset Texture properties.

.aoMapIntensity

Intensity of the ambient occlusion effect. Значение по умолчанию равно 1. Zero is no occlusion effect.

.bumpMap

The texture to create a bump map. The black and white values map to the perceived depth in relation to the lights. Bump doesn't actually affect the geometry of the object, only the lighting. If a normal map is defined this will be ignored.

.bumpScale

How much the bump map affects the material. Typical ranges are 0-1. Значение по умолчанию равно 1.

.color

Color of the material, by default set to white (0xffffffff).

.defines

An object of the form:

```
{ 'STANDARD': '' };
```

This is used by the WebGLRenderer for selecting shaders.

.displacementMap

The displacement map affects the position of the mesh's vertices. Unlike other maps which only affect the light and shade of the material the displaced vertices can cast shadows, block other objects, and otherwise act as real geometry. The displacement texture is an

image where the value of each pixel (white being the highest) is mapped against, and repositions, the vertices of the mesh.

.displacementScale

How much the displacement map affects the mesh (where black is no displacement, and white is maximum displacement). Without a displacement map set, this value is not applied. Значение по умолчанию равно 1.

.displacementBias

The offset of the displacement map's values on the mesh's vertices. Without a displacement map set, this value is not applied. Значение по умолчанию равно 0.

.emissive

Emissive (light) color of the material, essentially a solid color unaffected by other lighting. Default is black.

.emissiveMap

Set emissive (glow) map. Значением по умолчанию является null. The emissive map color is modulated by the emissive color and the emissive intensity. If you have an emissive map, be sure to set the emissive color to something other than black.

.emissiveIntensity

Intensity of the emissive light. Modulates the emissive color. Значение по умолчанию равно 1.

.envMap

The environment map. Значением по умолчанию является null. Note that in order for the material roughness property to correctly blur out the environment map, the shader must have access to mipmaps of the env texture. TextureCubes created with default settings are correctly configured; if adjusting texture parameters manually, ensure minFilter is set to one of the MipMap options, and that mip maps have not been otherwise forcibly disabled.

.envMapIntensity

Scales the effect of the environment map by multiplying its color.

.isMeshStandardMaterial

Used to check whether this or derived classes are mesh standard materials. Значением по умолчанию является `true`.

You should not change this, as it used internally for optimisation.

.lightMap

The light map. Значением по умолчанию является `null`. The lightMap requires a second set of UVs, and consequently will ignore the repeat and offset Texture properties.

.lightMapIntensity

Intensity of the baked light. Значение по умолчанию равно 1.

.map

The color map. Значением по умолчанию является `null`. The texture map color is modulated by the diffuse `.color`.

[.metalness](#)

Свойство определяет, насколько материал похож на металл. Для неметаллических материалов, вроде дерева или камня, используется значение 0.0, для металлических - используется 1.0, остальные материалы (обычно) в промежутке между этими значениями. Значение по умолчанию равно 0.5. Значение между 0.0 и 1.0 может использоваться для передачи вида ржавого металла. Если также представлено свойство `metalnessMap`, оба значения перемножаются.

[.metalnessMap](#)

Канал синего цвета данной текстуры используется для передачи изменения металличности (свойство `metalness`) этого материала.

[.morphNormals](#)

Defines whether the material uses morphNormals. Set as true to pass morphNormal attributes from the Geometry to the shader. Значением по умолчанию является false.

.morphTargets

Define whether the material uses morphTargets. Значением по умолчанию является false.

.normalMap

The texture to create a normal map. The RGB values affect the surface normal for each pixel fragment and change the way the color is lit. Normal maps do not change the actual shape of the surface, only the lighting.

.normalScale

How much the normal map affects the material. Typical ranges are 0-1. Default is a Vector2 set to (1,1).

.refractionRatio

The index of refraction (IOR) of air (approximately 1) divided by the index of refraction of the material. It is used with environment mapping modes THREE.CubeRefractionMapping and THREE.EquirectangularRefractionMapping. The refraction ratio should not exceed 1. Значение по умолчанию равно 0.98.

.roughness

How rough the material appears. 0.0 means a smooth mirror reflection, 1.0 means fully diffuse. Значение по умолчанию равно 0.5. If roughnessMap is also provided, both values are multiplied.

.roughnessMap

The green channel of this texture is used to alter the roughness of the material.

.skinning

Define whether the material uses skinning. Значением по

умолчанию является `false`.

[.wireframe](#)

Свойство позволяет отобразить геометрическую фигуру в виде каркаса. Значением по умолчанию является `false` (т.е. каркас не виден и визуализация происходит при помощи плоских многоугольников - полигонов).

[.wireframeLinecap](#)

Свойство определяет внешний вид концов линий. Возможные значения: "butt" (торец), "round" (круглый) и "square" (прямоугольный). Значением по умолчанию является `round`.

Данное свойство соответствует свойству [lineCap в 2D Canvas](#) и оно игнорируется WebGL визуализатором (рендерером). Вот эта [статья о lineCap](#) на русском языке.

[.wireframeLinejoin](#)

Свойство определяет внешний вид соединений линий.

Возможные значения: "round" (округление), "bevel" (скос) и "miter" (митра). Значением по умолчанию является `round`.

Данное свойство соответствует свойству [lineJoin в 2D Canvas](#) и игнорируется WebGL визуализатором (рендерером). Вот эта [статья о lineJoin](#) на русском языке.

[.wireframeLineWidth](#)

Свойство управляет толщиной линий каркаса. Значение по умолчанию равно 1.

Из-за ограничений в [слое ANGLE](#) в операционных системах на основе Windows ширина линии всегда будет равна 1, независимо от установленного значения.

Примечание переводчика: ANGLE (Almost Native Graphics Layer Engine - практически встроенный движок графического слоя, а перевод самой аббревиатуры ANGLE - угол) это графический движок слоя абстракций, разработанный компанией Google, с открытым исходным кодом под лицензией BSD. Этот API в основном предназначен для обеспечения высокопроизводительной совместимости OpenGL с компьютерами Windows и веб-браузерами вроде Chromium, путем перевода вызовов OpenGL в Direct3D, который гораздо лучше поддерживается драйверами. Для ANGLE существует два внутренних визуализатора: самый старый использует Direct3D 9.0c, в

то время как новый использует Direct3D 11.

из статьи англ. Википедии о [ANGLE](#), перевод мой.

Методы

Для информации об общих методах смотрите базовый класс [Material](#).

Исходники

[MeshStandardMaterial.js в этом справочнике](#)
[MeshStandardMaterial.js на github.com](#)

[Material](#) → [MeshPhongMaterial](#) →

MeshToonMaterial

Расширение [MeshPhongMaterial](#) с with toon shading.

Пример

[webgl / materials / variations / toon](#)

Конструктор

`MeshToonMaterial(parameters)`

[parameters](#) - объект с одним или несколькими свойствами, определяющими внешний вид материала (дополнительный, необязательный параметр). Здесь можно передать любое свойство материала (включая любое свойство унаследованное от [Material](#) и [MeshStandardMaterial](#)).

Исключением является свойство [color](#), значение которого можно передать как шестнадцатиричную строку и по умолчанию равно `0xffffffff` (белый цвет). Метод [Color.set\(color \)](#) вызывается внутренне.

Свойства

Для информации об общих свойствах смотрите базовые классы [Material](#) и [MeshPhongMaterial](#).

.gradientMap

Карта градиента для Gradient map for the toon shading. По умолчанию значение равно null.

.isMeshToonMaterial

Используется для проверки, является ли данный объект, или производные от него, тонированным материалом сетки. По умолчанию значение равно true.

Не нужно его изменять, так как внутри **three.js** оно используется для оптимизации.

.defines

Объект в виде:

```
{ 'TOON': '' };
```

Он используется [WebGLRenderer](#)'ом для выбора шейдеров.

Методы

Для информации об общих методах смотрите базовые классы [Material](#) и [MeshPhongMaterial](#).

Исходники

[MeshToonMaterial.js в этом справочнике](#)

[MeshToonMaterial.js на github.com](#)

[Material →](#)

PointsMaterial

Материал, по умолчанию используемый классом [Points](#).

Пример

```
// This will add a starfield to the background of a scene
// Этот код добавит звездное поле на задний план сцены
var starsGeometry = new THREE.Geometry();

for ( var i = 0; i < 10000; i ++ ) {
    var star = new THREE.Vector3();
```

```

star.x = THREE.Math.randFloatSpread( 2000 );
star.y = THREE.Math.randFloatSpread( 2000 );
star.z = THREE.Math.randFloatSpread( 2000 );

starsGeometry.vertices.push( star );
}

var starsMaterial = new THREE.PointsMaterial( { color: 0x888888 } );
var starField = new THREE.Points( starsGeometry, starsMaterial );

scene.add( starField );

```

Другие примеры

[misc / controls / fly](#)
[webgl / buffergeometry / drawcalls](#)
[webgl / BufferGeometry / points](#)
[webgl / BufferGeometry / points / interleaved](#)
[webgl / camera](#)
[webgl / geometry / convex](#)
[webgl / geometry / shapes](#)
[webgl / interactive / raycasting / points](#)
[webgl / multiple / elements / text](#)
[webgl / points / billboards](#)
[webgl / points / billboards / colors](#)
[webgl / points / dynamic](#)
[webgl / points / random](#)
[webgl / points / sprites](#)
[webgl / trails](#)

Конструктор

`PointsMaterial(parameters)`

`parameters` - объект с одним или несколькими свойствами, определяющими внешний вид материала (дополнительный, необязательный параметр). Здесь можно передать любое свойство материала (включая любое свойство унаследованное от [Material](#)).

Исключением является свойство `color`, значение которого можно передать как шестнадцатиричную строку и по умолчанию равно `0xffffffff` (белый цвет). Метод `Color.set(color)` вызывается внутренне.

Свойства

Для информации об общих свойствах смотрите базовый класс [Material](#).

.color

[Цвет](#) материала, по умолчанию устанавливается как белый (0xffffffff).

.isPointsMaterial

Используется для проверки, является ли данный объект, или производные от него, материалом точек. По умолчанию значение равно `true`.

Не нужно его изменять, так как внутри *three.js* оно используется для оптимизации.

.lights

Свойство определяет, будет ли освещение влиять на материал. Значением по умолчанию является `false`.

.map

Свойство устанавливает цвет точек используя данные из [Texture](#).

.size

Свойство устанавливает размер точек. Значение по умолчанию равно `1.0`.

.sizeAttenuation

Свойство определяет, будет ли размер точек уменьшаться с расстоянием. Значением по умолчанию является `true`.

Методы

Для информации об общих методах смотрите базовый класс [Material](#).

Исходники

[PointsMaterial.js в этом справочнике](#)

[PointsMaterial.js на github.com](#)

[ShaderMaterial](#) →

RawShaderMaterial

Данный класс работает также как класс [ShaderMaterial](#), за исключением того, что определения встроенных атрибутов и uniform не добавляются к коду шейдера GLSL автоматически. except that definitions of built-in uniforms and attributes are not automatically prepended to the GLSL shader code.

Пример

```
var material = new THREE.RawShaderMaterial( {  
  
    uniforms: {  
        time: { value: 1.0 }  
    },  
    vertexShader: document.getElementById( 'vertexShader' ).textContent,  
    fragmentShader: document.getElementById( 'fragmentShader' ).textContent  
} );
```

Другие примеры

[WebGL / buffergeometry / rawshader](#)
[WebGL / buffergeometry / instancing / billboards](#)
[WebGL / buffergeometry / instancing / dynamic](#)
[WebGL / buffergeometry / instancing / interleaved / dynamic](#)
[WebGL / buffergeometry / instancing](#)
[WebGL / interactive / instances / gpu](#)
[WebGL / raymarching / reflect](#)

Конструктор

```
RawShaderMaterial( parameters )
```

parameters - объект с одним или несколькими свойствами, определяющими внешний вид материала (дополнительный, необязательный параметр). Здесь можно передать любое свойство материала (включая любое свойство унаследованное от [Material](#) и [ShaderMaterial](#)).

Свойства

Для информации об общих свойствах смотрите базовые классы [Material](#) и [ShaderMaterial](#).

[.isRawShaderMaterial](#)

Используется для проверки, является ли данный объект, или производные от него, материалами необработанных шейдеров. По умолчанию значение равно `true`.

Не нужно его изменять, так как внутри `three.js` оно используется для оптимизации.

Методы

Для информации об общих методах смотрите базовые классы [Material](#) и [ShaderMaterial](#).

Исходники

[RawShaderMaterial.js в этом справочнике](#)

[RawShaderMaterial.js на github.com](#)

[Material](#) →

ShaderMaterial

Материал отображаемый с помощью собственных (заказных) шейдеров. Шейдер - это небольшая программа, написанная на языке [GLSL](#) для запуска на [GPU](#). Можно использовать заказной шейдер, если требуется:

- реализовать эффект не включенный ни в один из встроенных [материалов](#)
- для повышения производительности объединить несколько объектов в одной [Geometry](#) или [BufferGeometry](#)

При использовании `ShaderMaterial` нужно учитывать следующие замечания:

- A `*ShaderMaterial*` will only be rendered properly by

[page:WebGLRenderer], since the GLSL code in the *vertexShader* and *fragmentShader* properties must be compiled and run on the GPU using WebGL.

- Начиная с версии THREE r72, больше не поддерживается прямое назначение атрибутов в ShaderMaterial. А [page:BufferGeometry] instance (instead of a [page:Geometry] instance) must be used instead, using [page:BufferAttribute] instances to define custom attributes.
- Начиная с версии THREE r77, экземпляры [WebGLRenderTarget](#) или [WebGLRenderTargetCube](#) не должны больше использоваться как uniforms. Вместо этого нужно воспользоваться их свойством [texture](#).
- Встроенные атрибуты и uniforms передаются шейдерам вместе с кодом. Если не требуется, чтобы [WebGLProgram](#) добавляла что-либо к коду шейдера, вместо данного класса можно использовать [RawShaderMaterial](#).

Пример

```
var material = new THREE.ShaderMaterial( {  
  
    uniforms: {  
  
        time: { value: 1.0 },  
        resolution: { value: new THREE.Vector2() }  
  
    },  
  
    vertexShader: document.getElementById( 'vertexShader' ).textContent,  
    fragmentShader: document.getElementById( 'fragmentShader' ).textContent  
} );
```

Другие примеры

[... показать](#)

Шейдеры вершин и фрагментов

Для любого материала можно задать два разных вида шейдеров:

- Вначале запускается **вершинный шейдер** (vertex shader); он принимает атрибуты, вычисляет/управляет положение каждой отдельной вершиной и передает дополнительные данные

шейдеру фрагмента.

- Вторым номером запускается **шейдер фрагмента** (fragment shader); он устанавливает цвет каждого отдельного фрагмента (пикселя) отображаемого на экране.

В шейдерах существует три типа переменных: униформы (uniforms), атрибуты (attributes) и вариации (varyings):

- Униформы (uniforms) - это переменные, что имеют однообразное значение для всех вершин; освещение, туман и карты тени - вот примеры данных, которые будут храниться в униформах. Uniforms can be accessed by both the vertex shader and the fragment shader.
- Атрибуты (attributes) - это переменные, связанные с каждой вершиной - например, положение вершины, нормаль к грани и цвет вершины - это все примеры данных, которые будут хранится в атрибутах. Атрибуты могут быть доступны **только** внутри вершинного шейдера.
- Вариации (varyings) - это переменные, которые передаются из вершинного шейдера в шейдер фрагмента. Для каждого фрагмента значение каждой вариации будет плавно интерполироваться из значений смежных вершин.

Обратите внимание, что **внутри** собственно самого шейдера, униформы и атрибуты действуют как константы; можно только изменять их значения передачей различных значений в буферы из JavaScript кода. Note that **within** the shader itself, uniforms and attributes act like constants; you can only modify their values by passing different values to the buffers from your JavaScript code.

Встроенные атрибуты и униформы

По умолчанию [WebGLRenderer](#) предоставляет шейдерам множество атрибутов и униформов; определения этих переменных добавляются в код `fragmentShader` и `vertexShader` с помощью [WebGLProgram](#) при компиляции шейдера; отдельно объявлять их самих не требуется. Подробнее об этих переменных смотрите в описании [WebGLProgram](#).

Некоторые из этих униформ или атрибутов (например связанные с

освещением, туманом и т.д.) требуют установки свойств материала, чтобы [WebGLRenderer](#) мог скопировать соответствующие значения для GPU - не забудьте установить эти флаги, если хотите в своем собственном шейдере использовать данные функции.

Если не требуется, чтобы [WebGLProgram](#) добавляла что-либо к коду шейдера, вместо данного класса можно использовать [RawShaderMaterial](#).

Настраиваемые атрибуты и униформы Custom attributes and uniforms
Оба типа переменных (и настраиваемые атрибуты, и униформы), должны быть объявлены в GLSL коде шейдера (внутри vertexShader и/или fragmentShader). Both custom attributes and uniforms must be declared in your GLSL shader code (within *vertexShader* and/or *fragmentShader*). Custom uniforms must be defined in both the *uniforms* property of your *ShaderMaterial*, whereas any custom attributes must be defined via [page:BufferAttribute] instances. Note that *varying*s only need to be declared within the shader code (not within the material).

To declare a custom attribute, please reference the [page:BufferGeometry] page for an overview, and the [page:BufferAttribute] page for a detailed look at the *BufferAttribute* API.

При создании собственных атрибутов имейте ввиду, что каждый типизированный массив, для хранения данных вашего атрибута, должен быть кратным размеру вашего типа данных. Например, если типом атрибута будет [THREE.Vector3](#) и в вашей [BufferGeometry](#) имеется 3000 вершин, то величина создаваемого типизированного массива должна иметь длину $3000 * 3$ или 9000 (т.е. одно значение на каждый элемент). When creating your attributes, each typed array that you create to hold your attribute's data must be a multiple of your data type's size. For example, if your attribute is a [page:Vector3 THREE.Vector3] type, and you have 3000 vertices in your [page:BufferGeometry], your typed array value must be created with a length of $3000 * 3$, or 9000 (one value per-component). Для справки ниже приведена таблица размеров для каждого типа данных:

Размеры атрибутов

| ТИП GLSL GLSL TYPE | ТИП JAVASCRIPT JAVASCRIPT TYPE | РАЗМЕР SIZE |
|-----------------------|-----------------------------------|----------------|
| float | Number | 1 |
| vec2 | THREE.Vector2 | 2 |
| vec3 | THREE.Vector3 | 3 |
| vec3 | THREE.Color | 3 |
| vec4 | THREE.Vector4 | 4 |

Note that attribute buffers are **not** refreshed automatically when their values change. To update custom attributes, set the *needsUpdate* flag to true on the [page:BufferAttribute] of the geometry (see [page:BufferGeometry] for further details).

To declare a custom [page:Uniform], use the *uniforms* property:

```
uniforms: {  
    time: { value: 1.0 },  
    resolution: { value: new THREE.Vector2() }  
}
```

Конструктор

```
ShaderMaterial( parameters )
```

[parameters](#) - объект с одним или несколькими свойствами, определяющими внешний вид материала (дополнительный, необязательный параметр). Здесь можно передать любое свойство материала (включая любое свойство унаследованное от [Material](#)).

Свойства

Для информации об общих свойствахсмотрите базовый класс [Material](#).

```
.clipping
```

Свойство определяет, будет ли данный материал поддерживать обрезку; Defines whether this material supports clipping; true to let the renderer pass the clippingPlanes uniform. Значением по умолчанию является false.

```
.defaultAttributeValues
```

When the rendered geometry doesn't include these attributes but the material does, these default values will be passed to the shaders. This avoids errors when buffer data is missing.

```
this.defaultAttributeValues = {  
    'color': [ 1, 1, 1 ],  
    'uv': [ 0, 0 ],  
    'uv2': [ 0, 0 ]  
};
```

.defines

Defines custom constants using `*#define*` directives within the GLSL code for both the vertex shader and the fragment shader; each key/value pair yields another directive:

```
defines: {  
    FOO: 15,  
    BAR: true  
}
```

yields the lines

```
#define FOO 15  
#define BAR true
```

in the GLSL code.

.extensions

Объект со следующими свойствами:

```
this.extensions = {  
    derivatives: false, // set to use derivatives (установить при и  
    fragDepth: false, // set to use fragment depth values (установи  
    drawBuffers: false, // set to use draw buffers (установить при  
    shaderTextureLOD: false // set to use shader texture LOD (устан  
};
```

.fog

Define whether the material color is affected by global fog settings; true to pass fog uniforms to the shader. По умолчанию значение равно false.

.fragmentShader

Fragment shader GLSL code. This is the actual code for the shader. In the example above, the vertexShader and fragmentShader code is

extracted from the DOM; it could be passed as a string directly or loaded via AJAX instead.

[.index0AttributeName](#)

If set, this calls gl.bindAttribLocation to bind a generic vertex index to an attribute variable. По умолчанию значение равно undefined.

[.isShaderMaterial](#)

Используется для проверки, является ли данный объект, или производные от него, материалами шейдера. По умолчанию значение равно true.

Не нужно его изменять, так как внутри **three.js** оно используется для оптимизации.

[.lights](#)

Defines whether this material uses lighting; true to pass uniform data related to lighting to this shader. По умолчанию значение равно false.

[.linewidth](#)

Controls wireframe thickness. По умолчанию значение равно 1. Due to limitations in the ANGLE layer, on Windows platforms linewidth will always be 1 regardless of the set value.

[.morphTargets](#)

Defines whether the material uses morphTargets; true morphTarget attributes to this shader

[.morphNormals](#)

Defines whether the material uses morphNormals. Set as true to pass morphNormal attributes from the Geometry to the shader. По умолчанию значение равно false.

[.program](#)

The compiled shader program associated with this material, generated by WebGLRenderer. You should not need to access this property.

[.flatShading](#)

Define whether the material is rendered with flat shading. Default is false.

.skinning

Define whether the material uses skinning; true to pass skinning attributes to the shader. Default is false.

.uniforms

An object of the form:

```
{ "uniform1": { value: 1.0 }, "uniform2": { value: 2 } }
```

specifying the uniforms to be passed to the shader code; keys are uniform names, values are definitions of the form

```
{ value: 1.0 }
```

where value is the value of the uniform. Names must match the name of the uniform, as defined in the GLSL code. Note that uniforms are refreshed on every frame, so updating the value of the uniform will immediately update the value available to the GLSL code.

.vertexColors

Свойство определяет, как окрашиваются вершины, путем задания способа заполнения атрибута цвета. Define how the vertices are colored, by defining how the colors attribute gets populated.

Возможными значениями являются [THREE.NoColors](#), [THREE.FaceColors](#) и [THREE.VertexColors](#). Значение по умолчанию THREE.NoColors.

.vertexShader

Код GLSL вершинного шейдера. Для шейдера он является действующим кодом. В примере выше, код вершинного шейдера (vertexShader) и шейдера фрагмента (fragmentShader) извлекается из [DOM](#); вместо этого, он мог быть передан напрямую, в виде строки или загружен через [AJAX](#).

.wireframe

Отображать ли геометрию как каркас Render geometry as wireframe (применяется GL_LINES вместо GL_TRIANGLES). Значением по умолчанию является false (т.е. каркас не виден и визуализация происходит при помощи плоских многоугольников).

.wireframeLinewidth

Свойство управляет толщиной линий каркаса. Значением по умолчанию является 1.

Из-за ограничений в [слое ANGLE](#) в операционных системах на основе Windows ширина линии всегда будет равна 1, независимо от установленного значения.

Примечание переводчика: ANGLE (Almost Native Graphics Layer Engine - практически встроенный движок графического слоя, а перевод самой аббревиатуры ANGLE - угол) это графический движок слоя абстракций, разработанный компанией Google, с открытым исходным кодом под лицензией BSD. Этот API в основном предназначен для обеспечения высокопроизводительной совместимости OpenGL с компьютерами Windows и веб-браузерами вроде Chromium, путем перевода вызовов OpenGL в Direct3D, который гораздо лучше поддерживается драйверами. Для ANGLE существует два внутренних визуализатора: самый старый использует Direct3D 9.0c, в то время как новый использует Direct3D 11.

из статьи англ. Википедии о [ANGLE](#), перевод мой.

Методы

Для информации об общих методах смотрите базовый класс [Material](#).

.clone() this

Generates a shallow copy of this material. Обратите внимание на то, что vertexShader и fragmentShader копируются **по ссылке**, as are the definitions of the *attributes*; this means that clones of the material will share the same compiled [page:WebGLProgram]. А вот униформы копируются **по значению**, что позволяет иметь различные наборы униформ для разных копий материала. However, the *uniforms* are copied **by value**, which allows you to have different sets of uniforms for different copies of the material.

Исходники

[ShaderMaterial.js в этом справочнике](#)
[ShaderMaterial.js на github.com](#)

[Material](#) → [ShaderMaterial](#) →

ShadowMaterial

Данный материал может принимать тени, но в остальном он полностью прозрачен (На русский язык shadow material можно перевести как теневой материал, или материал для тени).

Пример

```
var planeGeometry = new THREE.PlaneGeometry( 2000, 2000 );
planeGeometry.rotateX( - Math.PI / 2 );

var planeMaterial = new THREE.ShadowMaterial();
planeMaterial.opacity = 0.2;

var plane = new THREE.Mesh( planeGeometry, planeMaterial );
plane.position.y = -200;
plane.receiveShadow = true;
scene.add( plane );
```

Другие примеры

[geometry](#) / [spline](#) / [editor](#)

Конструктор

`ShadowMaterial(parameters)`

`parameters` - объект с одним или несколькими свойствами, определяющими внешний вид материала (дополнительный, необязательный параметр). Здесь можно передать любое свойство материала (включая любое свойство унаследованное от [Material](#) и [ShaderMaterial](#)).

Свойства

Для информации об общих свойствах смотрите базовые классы [Material](#) и [ShaderMaterial](#).

`.isShadowMaterial`

Используется для проверки, является ли данный объект, или производные от него, теневыми материалами. По умолчанию

значение равно true.

Не нужно его изменять, так как внутри ***three.js*** оно используется для оптимизации.

[.lights](#)

Свойство устанавливает, будет или нет освещение влиять на материал. Значением по умолчанию является true.

[.transparent](#)

Определяет будет ли данный материал прозрачным или нет. Значением по умолчанию является true.

Методы

Для информации об общих методах смотрите базовые классы [Material](#) и [ShaderMaterial](#).

Исходники

[ShadowMaterial.js в этом справочнике](#)
[ShadowMaterial.js на github.com](#)

[Material](#) →

SpriteMaterial

Материал для использования со [Sprite](#).

Пример

```
var spriteMap = new THREE.TextureLoader().load( 'textures/sprite.png' );
var spriteMaterial = new THREE.SpriteMaterial( { map: spriteMap, color: 0x0000ff, size: 200 } );
var sprite = new THREE.Sprite( spriteMaterial );
sprite.scale.set(200, 200, 1);
scene.add( sprite );
```

Другие примеры

```
WebGL / sprites
misc / ubiquity / test
misc / ubiquity / test2
software / sandbox
svg / sandbox
webgl / materials / cubemap / dynamic
```

Конструктор

```
SpriteMaterial( parameters )
```

[parameters](#) - объект с одним или несколькими свойствами, определяющими внешний вид материала (дополнительный, необязательный параметр). Здесь можно передать любое свойство материала (включая любое свойство унаследованное от [Material](#)).

Исключением является свойство [color](#), значение которого можно передать как шестнадцатиричную строку и по умолчанию равно `0xffffffff` (белый цвет). Метод [Color.set\(color \)](#) вызывается внутренне.

Свойства

Для информации об общих свойствах смотрите базовый класс [Material](#).

[.color](#)

[Цвет](#) материала, по умолчанию устанавливается как белый цвет (`0xffffffff`). Карта текстуры ([map](#)) перемножается с этим цветом.

[.fog](#)

Свойство устанавливает, будет или нет туман на сцене влиять на материал. Значением по умолчанию является `false`.

[.lights](#)

Свойство устанавливает, будет или нет освещение влиять на материал. Значением по умолчанию является `false`.

[.map](#)

Карта текстуры. Значением по умолчанию является null.

[.rotation](#)

Угол поворота спрайта, в радианах. Значением по умолчанию является 0.

Методы

Для информации об общих методах смотрите базовый класс [Material](#).

Исходники

[SpriteMaterial.js в этом справочнике](#)

[SpriteMaterial.js на github.com](#)

МАТЕМАТИКА

В этом разделе рассматриваются некоторые объекты и функции, имеющие отношение к математике.

Box2

Представляет собой ограниченный прямоугольник в двумерном пространстве.

Конструктор

```
Box2(  
    сокращ. от англ. слова minimum - минимум' );" onmouseout="hide()">mi  
    сокращ. от англ. слова maximum - максимум' );" onmouseout="hide()">m
```

min - двумерный вектор ([Vector2](#)), представляющий нижнюю границу (x, y) прямоугольника, дополнительный, необязательный параметр. Значением по умолчанию является (+Infinity, +Infinity).

max - двумерный вектор ([Vector2](#)), представляющий верхнюю границу (x, y) прямоугольника, дополнительный, необязательный параметр. Значением по умолчанию является (-Infinity, -Infinity).

Создает прямоугольник, ограниченный по min и max.

Свойства

```
    сокращ. от англ. слова minimum - минимум' );" onmouseout="hide()">
```

Двумерный вектор ([Vector2](#)), представляющий нижнюю границу (x, y) прямоугольника. Значением по умолчанию является (+Infinity, +Infinity).

сокращ. от англ. слова maximum - максимум');" onmouseout="hide()"

Двумерный вектор ([Vector2](#)), представляющий верхнюю границу (x, y) прямоугольника. Значением по умолчанию является (-Infinity, -Infinity).

Методы

clamp point - зажать (закрепить) точку');" onmouseout="hide()">>.clamp
point - точка');" onmouseout="hide()">>point,
optional target - дополнительная цель');" onmouseout="hide()">>opt

point - точка)" onmouseout="hide()">>point -
двумерный вектор ([Vector2](#)), который нужно закрепить.

optional target - дополнительная цель)" onmouseout="hide()">>target
— если этот параметр указан, то результат будет скопирован в данный двумерный вектор ([Vector2](#)), в противном случае будет создан новый [Vector2](#).

Дополнительный, необязательный параметр.

Метод [зажимает точку](#) внутри границ заданного прямоугольника.

Примечание переводчика: Вот перевод вышеприведенной статьи из английской Википедии

В компьютерной графике, зажим (clamping) — это процесс ограничения (фиксации) местоположения в какой-либо области. В отличие от упаковки ([wrapping](#)'а), зажим (clamping) просто перемещает точку к ближайшему доступному значению.

В общем случае, зажим (clamping) используется для ограничения значения в заданном диапазоне. Например, в OpenGL `glClearColor` принимает значение `GLclampf`, которым является число `g1` с плавающей запятой, «зажатое» в диапазоне $[0, 1]$.

Одно из многих применений зажима (clamping'a) в компьютерной графике является размещение некоей детали в полигоне — например, отверстие от пули в стене. Также зажим может использоваться для создания различных эффектов.

clone - клонировать)" onmouseout="hide()">>.clone()

Возвращает новый Box2 с теми же самыми значениями `min` и `max` как у этого.

```
.containsBox(  
box - коробка, ящик, клетка'" onmouseout="hide()">box )
```

`box - коробка, ящик, клетка'" onmouseout="hide()">box` - прямоугольник
`box - коробка, ящик, клетка'" onmouseout="hide()">Box` для проверки на включение.

Возвращает значение `true`, если в данный прямоугольник включен целиком прямоугольник, указанный в параметре `box`. Если данный прямоугольник и указанный в параметре `box` одинаковы, эта функция также возвратит значение `true`.

```
.containsPoint(  
point - точка'" onmouseout="hide()">point )
```

`point - точка'" onmouseout="hide()">point` - двумерный вектор ([Vector2](#)) для проверки на включение.
Возвращает значение `true`, если точка, указанная в параметре `point - точка'" onmouseout="hide()">point`, лежит внутри или на границах данного прямоугольника.

```
.copy( box )
```

Копирует значения `min` и `max` из прямоугольника указанного в `box` в данный прямоугольник.

```
.distanceToPoint( point )
```

`point` - двумерный вектор ([Vector2](#)) до которого измеряется расстояние.
Метод возвращает расстояние от любого края данного прямоугольника до указанной точки. Если точка (`point`) лежит внутри данного прямоугольника, то расстояние будет равно 0.

```
.equals( box )
```

`box` - прямоугольник для сравнения с данным прямоугольником.
Возвращает значение `true`, если данный прямоугольник и прямоугольник `box` имеют одни и те же нижние и верхние границы.

```
.expandByPoint( point )
```

| point - вектор ([Vector2](#)), который должен быть включен в этот прямоугольник.

Расширяет границы данного прямоугольника для включения точки [point](#).

```
.expandByScalar ( scalar )
```

| scalar - Distance to expand the box by.

Expands each dimension of the box by scalar. If negative, the dimensions of the box will be contracted.

```
.expandByVector ( vector )
```

| vector - Vector2 to expand the box by.

Expands this box equilaterally by vector. The width of this box will be expanded by the x component of vector in both directions. The height of this box will be expanded by the y component of vector in both directions.

```
.getCenter ( optionalTarget )
```

| optionalTarget — если этот параметр указан, то результат будет скопирован в данный двумерный вектор ([Vector2](#)), в противном случае будет создан новый [Vector2](#).

| Дополнительный, необязательный параметр.

Returns the center point of the box as a [Vector2](#).

```
.getParameter ( point, optionalTarget )
```

| point - [Vector2](#).

| optionalTarget — если этот параметр указан, то результат будет скопирован в данный двумерный вектор ([Vector2](#)), в противном случае будет создан новый [Vector2](#).

| Дополнительный, необязательный параметр.

Returns a point as a proportion of this box's width and height.

```
.getSize( optionalTarget )
```

| optionalTarget — если этот параметр указан, то результат

будет скопирован в данный двумерный вектор (Vector2), в противном случае будет создан новый Vector2.
Дополнительный, необязательный параметр.

Возвращает ширину и высоту данного прямоугольника.

.intersect(box)

box - Box to intersect with.

Returns the intersection of this and box, setting the upper bound of this box to the lesser of the two boxes' upper bounds and the lower bound of this box to the greater of the two boxes' lower bounds.

.intersectsBox(box)

box - прямоугольник для проверки на наличие пересечения.

Определяет, пересекается ли данный прямоугольник с прямоугольником box.

.isEmpty()

Метод возвращает значение true, если данный прямоугольник содержит нулевые точки внутри своих границ if this box includes zero points within its bounds.

Обратите внимание, что прямоугольник с одинаковыми нижними и верхними границами все же включает одну точку, которая принадлежит одновременно обоим границам Returns true if this box includes zero points within its bounds. Note that a box with equal lower and upper bounds still includes one point, the one both bounds share.

.makeEmpty()

Метод делает данный прямоугольник пустым.

.set(min, max)

min - обязательный аргумент. (required) Vector2 representing the lower (x, y) boundary of the box.

max - обязательный аргумент. (required) Vector2 representing the lower upper (x, y) boundary of the box.

Sets the lower and upper (x, y) boundaries of this box.

```
.setFromCenterAndSize( center, size )
```

| center - Desired center position of the box (Vector2).

| size - Desired x and y dimensions of the box (Vector2).

Centers this box on center and sets this box's width and height to the values specified in size.

```
.setFromPoints( points )
```

| points - массив из двумерных векторов ([Vector2](#)), которые будет содержать получающийся прямоугольник.

Устанавливает верхние и нижние границы данного прямоугольника так, что в него будут входить все точки, указанные в points.

```
.translate ( offset )
```

| offset - направление и расстояние смещения.

Добавляет смещение к обоим, верхней и нижней, границам данного прямоугольника, фактически перемещая этот прямоугольник Adds offset to both the upper and lower bounds of this box, effectively moving this box offset units in 2D space.

```
.union ( box )
```

| box - прямоугольник, который будет объединен с данным прямоугольником.

Метод объединяет данный прямоугольник с прямоугольником, представленным в параметре box, устанавливая верхнюю границу данного прямоугольника выше верхних границ обоих прямоугольников, а нижнюю границу этого прямоугольника ниже нижних границ обоих прямоугольников. Unions this box with box, setting the upper bound of this box to the greater of the two boxes' upper bounds and the lower bound of this box to the lesser of the two boxes' lower bounds.

Исходники

[Box2.js в этом справочнике](#)

[Box2.js на Гитхабе](#)

Box3

Представляет собой прямоугольный параллелепипед или куб в 3-мерном пространстве. Основное назначение данного класса - представление [прямоугольного параллелепипеда со сторонами, параллельными осям координат](#) (в англоязычном варианте это [Minimum Bounding Boxes](#)), для ограничения некоторых геометрических объектов в пространстве.

Конструктор

`Box3(min, max)`

[min](#) - трехмерный вектор ([Vector3](#)), представляющий нижнюю границу (x, y, z) прямоугольника, дополнительный, необязательный параметр. Значением по умолчанию является (+Infinity, +Infinity, +Infinity).

[max](#) - трехмерный вектор ([Vector3](#)), представляющий верхнюю границу (x, y, z) прямоугольника, дополнительный, необязательный параметр. Значением по умолчанию является (-Infinity, -Infinity, -Infinity).

Создает Box3, с границами по min и max.

Свойства

`.isBox3`

Свойство используется для проверки, является ли данный и производные от него классы Box3-ами. Значением по умолчанию является true.

Его нельзя изменять, так как оно используется внутри *three.js* для оптимизации.

`.min`

Трехмерный вектор ([Vector3](#)), представляющий нижнюю границу

(x , y , z) Box3. Значением по умолчанию является
($+\infty$, $+\infty$, $+\infty$).

.max

Трехмерный вектор ([Vector3](#)), представляющий верхнюю границу
(x , y , z) Box3. Значением по умолчанию является
($-\infty$, $-\infty$, $-\infty$).

Методы

.applyMatrix4(matrix)

| matrix - The Matrix4 to apply

Метод преобразует данный Box3 Transforms this Box3 with the supplied matrix.

.clampPoint (point, optionalTarget)

| point - трехмерный вектор ([Vector2](#)), который нужно зафиксировать.

| optionalTarget — если этот параметр указан, то результат будет скопирован в данный трехмерный вектор ([Vector3](#)), в противном случае будет создан новый [Vector3](#).

| Дополнительный, необязательный параметр.

Метод [зажимает точку](#) внутри границ заданного прямоугольника.

Примечание переводчика: Вот перевод вышеупомянутой статьи из английской Википедии

В компьютерной графике, зажим (clamping) - это процесс ограничения (фиксации) местоположения в какой-либо области. В отличие от упаковки ([wrapping](#)'a), зажим (clamping) просто перемещает точку к ближайшему доступному значению.

В общем случае, зажим (clamping) используется для ограничения значения в заданном диапазоне. Например, в OpenGL `glClearColor` принимает значение `GLclampf`, которым является число `gl` с плавающей запятой, «зажатое» в диапазоне $[0, 1]$.

Одно из многих применений зажима (clamping'a) в компьютерной графике является размещение некоей детали в полигоне - например, отверстие от пули в стене. Также зажим может использоваться для создания различных эффектов.

.clone()

Возвращает новый Box3 с теми же самыми значениями min и max

как у этого.

.containsBox (box)

| box - параллелепипед [Box2](#) для проверки на включение.
Возвращает значение true, если в данный параллелепипед включен целиком параллелепипед, указанный в параметре box.
Если данный параллелепипед и указанный в параметре box одинаковы, эта функция также возвратит значение true.

.containsPoint (point)

| point - трехмерный вектор ([Vector3](#)) для проверки на включение.
Возвращает значение true, если точка, указанная в параметре point, лежит внутри или на границах данного параллелепипеда.

.copy (box)

Копирует значения min и max из параллелепипеда указанного в box в данный параллелепипед.

.distanceToPoint(point)

| point - трехмерный вектор ([Vector3](#)) до которого измеряется расстояние.
Метод возвращает расстояние от любого края данного параллелепипеда до указанной точки. Если точка (point) лежит внутри данного параллелепипеда, то расстояние будет равно 0.

.equals(box)

| box - параллелепипед для сравнения с данным параллелепипедом.
Возвращает значение true, если данный параллелепипед и параллелепипед box имеют одни и теже нижние и верхние границы.

.expandByObject(object)

| object - [Object3D](#), под который нужно расширить параллелепипед.

Расширяет границы данного параллелепипеда, чтобы включить объект с его дочерними объектами с учетом их глобальных преобразований. Expands the boundaries of this box to include object and its children, accounting for the object's, and children's, world transforms.

```
.expandByPoint( point )
```

point - вектор ([Vector3](#)), который должен быть включен в этот параллелепипед.

Расширяет границы данного параллелепипеда для включения точки [point](#).

```
.expandByScalar( scalar )
```

scalar - Distance to expand the box by.

Expands each dimension of the box by scalar. If negative, the dimensions of the box will be contracted.

```
.expandByVector( vector )
```

vector - Vector3 to expand the box by.

Expands this box equilaterally by vector. The width of this box will be expanded by the x component of vector in both directions. The height of this box will be expanded by the y component of vector in both directions. The depth of this box will be expanded by the z component of vector in both directions.

```
.getBoundingSphere( optionalTarget )
```

optionalTarget — (optional) if specified, the result will be copied into this Sphere, otherwise a new Sphere will be created.

Gets a Sphere that bounds the box.

```
.getCenter( optionalTarget )
```

optionalTarget — если этот параметр указан, то результат будет скопирован в данный трехмерный вектор ([Vector3](#)), в противном случае будет создан новый [Vector3](#).
Дополнительный, необязательный параметр.

Метод возвращает центральную точку параллелепипеда в виде

трехмерного вектора Vector3.

`.getParameter(point, optionalTarget)`

`point` — трехмерный вектор ([Vector3](#)).

`optionalTarget` — если этот параметр указан, то результат будет скопирован в данный трехмерный вектор (Vector3), в противном случае будет создан новый Vector3.

Дополнительный, необязательный параметр.

Returns a point as a proportion of this box's width and height.

`.getSize(optionalTarget)`

`optionalTarget` — если этот параметр указан, то результат будет скопирован в данный двумерный вектор (Vector3), в противном случае будет создан новый Vector3.

Дополнительный, необязательный параметр.

Возвращает ширину и высоту данного прямоугольника.

`[method:Box3 set([page:Vector3 min, [page:Vector3 max)`

`min` -- Lower (x, y, z) boundary of the box.

`max` -- Upper (x, y, z) boundary of the box.

Sets the lower and upper (x, y, z) boundaries of this box.

`[method:Boolean intersectsBox([page:Box3 box) [page:Box3 this`

`box` -- Box to check for intersection against.

Determines whether or not this box intersects *box*.

`[method:Box3 setFromObject([page:Object3D object)`

`object` -- [page:Object3D to compute the bounding box for.

Computes the world-axis-aligned bounding box of an object (including its children), accounting for both the object's, and childrens', world transforms

`[method:Box3 intersect([page:Box3 box)`

`box` -- Box to intersect with.

Returns the intersection of this and *box*, setting the upper bound of

this box to the lesser
of the two boxes' upper bounds and the lower bound of this box to the
greater of the two boxes'
lower bounds.

[method:Boolean isEmpty()]

Returns true if this box includes zero points within its bounds.
Note that a box with equal lower and upper bounds still includes one
point, the one both bounds share.

[method:Box3 makeEmpty()]

Makes this box empty.

[method:Vector3 center([page:Vector3 optionalTarget)

optionalTarget -- If specified, the result will be copied here.

Returns the center point of this box.

[method:Box3 setFromCenterAndSize([page:Vector3 center, [page:Ve

center -- Desired center position of the box.

size -- Desired x, y and z dimensions of the box.

Centers this box on *center* and sets this box's width, height and
depth to the values specified in *size*.

.setFromPoints(points)

points - массив из трехмерных векторов ([Vector3](#)), которые
будет содержать получающийся прямоугольник.

Устанавливает верхние и нижние границы данного
параллелепипеда так, что в него будут входить все точки,
указанные в points.

.translate (offset)

offset - направление и расстояние смещения.

Добавляет смещение к обоим, верхней и нижней, границам
данного прямоугольника, фактически перемещая этот
прямоугольник Adds offset to both the upper and lower bounds of
this box, effectively moving this box offset units in 2D space.

.union ([box](#))

[box](#) - прямоугольный параллелепипед, который будет объединен с данным прямоугольным параллелепипедом. Метод объединяет данный прямоугольник с прямоугольником, представленным в параметре [box](#), устанавливая верхнюю границу данного параллелепипеда по наибольшей из верхних границ обоих параллелепипедов, а нижнюю границу этого параллелепипеда по наименьшей из нижних границ обоих параллелепипедов.

Исходники

[Box3.js в этом справочнике](#)

[Box3.js на Гитхабе](#)

Color

Класс представляющий цвет.

Пример

Класс Color может быть инициализирован любым из следующих способов:

```
// empty constructor - will default white
// пустой конструктор - по умолчанию цвет будет белым
var color = new THREE.Color();

// Hexadecimal color (recommended)
// шестнадцатиричное значение цвета (рекомендуется)
var color = new THREE.Color( 0xff0000 );

// RGB string
// строковое RGB значение
var color = new THREE.Color("rgb(255, 0, 0)");
var color = new THREE.Color("rgb(100%, 0%, 0%)");

// X11 color name - all 140 color names are supported.
```

```

// Note the lack of CamelCase in the name
// название цвета X11 - поддерживаются все 140 названий цвета.
// обратите внимание, в названии цвета отсутствует CamelCase - верб
var color = new THREE.Color( 'skyblue' );

// HSL string
// строковое значение HSL
var color = new THREE.Color("hsl(0, 100%, 50%)");

// Separate RGB values between 0 and 1
// отдельные значения RGB от 0 до 1
var color = new THREE.Color( 1, 0, 0 );

```

Конструктор

`Color(r, g, b)`

r - (дополнительный, необязательный параметр) если указаны аргументы g и b, то данный аргумент определяет красный компонент цвета. Если их нет, это может быть либо шестнадцатиричное значение цвета (рекомендуется) (у автора здесь [hexadecimal triplet](#)), либо строковое значение цвета в стиле CSS, либо другой экземпляр Color. Вот [статья Википедии о цвете в HTML](#) на русском языке.

g - (дополнительный, необязательный параметр), если указан, то определяет зеленый компонент цвета.

b - (дополнительный, необязательный параметр), если указан, то определяет синий компонент цвета.

Цвет по умолчанию - белый (white).

Если указаны все аргументы, то r будет красным компонентом, g - зеленым, а b будет синим компонентом цвета.

Когда установлен только аргумент r, то:

- это может быть шестнадцатиричное значение цвета (рекомендуется к применению).
- это может быть другой экземпляр цвета.
- это может быть строковое значение стиля CSS, например:

```

'rgb(250, 0, 0)'
'rgb(100%,0%,0%)'
'hsl(0, 100%, 50%)'
'#ff0000'
'#f00'

```

```
'red'
```

Свойства

.isColor

Свойство используется для проверки, является ли данный и производные от него классы цветом. Значением по умолчанию является true.

Его нельзя изменять, так как оно используется внутри three.js для оптимизации.

.r

Значение красного канала цвета, от 0 до 1. По умолчанию равно 1.

.g

Значение зеленого канала цвета, от 0 до 1. По умолчанию равно 1.

.b

Значение синего канала цвета, от 0 до 1. По умолчанию равно 1.

Методы

.add(color)

Метод выполняет сложение RGB значений цвета, переданного в параметре color, со RGB значениями данного цвета.

.addColors(color1, color2)

Sets this color's RGB values to the sum of the RGB values of color1 and color2.

.addScalar(s)

Adds s to the RGB values of this color.

`.clone()`

Метод возвращает новый экземпляр класса Color с теми же самыми значениями `r`, `g` и `b` как у данного.

`.convertGammaToLinear()`

Метод конвертирует данный цвет из гамма-пространства в линейное возведением в квадрат значений `r`, `g` и `b`. Converts this color from gamma to linear space by squaring the values of r, g and b).

`.convertLinearToGamma()`

Метод конвертирует данный цвет из линейного в гамма-пространство с помощью извлечения квадратного корня из значений `r`, `g` и `b`. Converts this color from linear to gamma space by taking the square root of r, g and b).

`.copy(color)`

Метод копирует параметры `r`, `g` и `b` цвета, переданного в аргументе `color`, в данный цвет.

`.copyGammaToLinear(color, gammaFactor)`

`color` — Color to copy.

`gammaFactor` - (optional). Значение по умолчанию равно 2.0.

Copies the given color into this color while converting it from gamma to linear space by taking r, g and b to the power of gammaFactor.

`.copyLinearToGamma(color, gammaFactor)`

`color` — Color to copy.

`gammaFactor` - (optional). Значение по умолчанию равно 2.0.

Copies the given color into this color while converting it from linear to gamma space by taking r, g and b to the power of 1 / gammaFactor.

`.equals(color)`

Метод сравнивает значения RGB, переданные в аргументе `color` с такими же значениями данного объекта. Если они одинаковы, возвращается значение `true`, в противном случае возвращается значение `false`.

`.fromArray(array, offset)`

`array` - массив чисел с плавающей запятой в форме
[`.r`, `.g`, `.b`].

`offset` - дополнительное смещение в массиве.

Метод устанавливает компоненты данного цвета основываясь на массиве в формате [`.r`, `.g`, `.b`].

`.getHex()`

Возвращает шестнадцатиричное значение данного цвета.

`.getHexString()`

Метод возвращает шестнадцатиричное значение данного цвета в виде строки (например, 'FFFFFF').

`.getHSL(optionalTarget)`

`optionalTarget` — (optional) if specified, adds h, s and l keys to the object (if not already present) and sets the results there.
Otherwise, a new Object will be created.

Конвертирует значения `r`, `g` и `b` данного цвета в формат [HSL](#) и возвращает объект в виде { `h:0`, `s:0`, `l:0` }.

Вот статья Википедии о [HSL](#) на русском языке

`.getStyle()`

Возвращает значение данного цвета в виде строки CSS стиля.

Пример: 'rgb(255,0,0)'.

`.lerp(color, alpha)`

`color` - цвет, с которым производится сближение.

`alpha` - коэффициент интерполяции в закрытом интервале [0, 1].

Метод выполняет операцию линейного интерполирования между значениями RGB данного цвета и RGB значениям цвета, переданного в аргументе `color`. Аргумент `alpha` можно рассматривать как соотношение между двумя цветами, где 0.0 означает данный цвет, а 1.0 - цвет, переданный в первом

аргументе.

`.multiply(color)`

Умножает значения RGB данного цвета на значения RGB цвета, переданного в аргументе [color](#).

`.multiplyScalar(s)`

Метод умножает значения RGB данного цвета на число, переданное в аргументе [s](#).

`.offsetHSL(h, s, l)`

Метод добавляет значения [h](#), [s](#), и [l](#) к значениям данного цвета. Внутри же метода, вначале конвертируются значения [r](#), [g](#) и [b](#) в HSL, добавляются значения [h](#), [s](#), и [l](#), а затем цвет конвертируется обратно в RGB.

`.set(value)`

| [value](#) - значение для установки в данный цвет.

Более подробную информацию о том, какое значение может быть указано в аргументе [value](#), см. в разделе [Конструктор](#). В зависимости от введенного типа, выполнение передается методам [copy](#), [setStyle](#), или [setHex](#).

`.setHex(hex)`

| [hex](#) — значение цвета в шестнадцатиричном формате (у автора здесь [hexadecimal triplet](#)). Вот [статья Википедии о цвете в HTML](#) на русском языке.

Метод устанавливает значение данного цвета из шестнадцатиричного значения, переданного в параметре [hex](#).

`.setHSL(h, s, l)`

| [h](#) — значение оттенка цвета в диапазоне от 0.0 до 1.0.

| [s](#) — значение насыщенности цвета в диапазоне от 0.0 до 1.0.

| [l](#) — значение яркости цвета в диапазоне от 0.0 до 1.0.

Метод устанавливает данный цвет из значений HSL.

`.setRGB(r, g, b)`

`r` — значение красного канала в диапазоне от 0.0 до 1.0.

`g` — значение зеленого канала в диапазоне от 0.0 до 1.0.

`b` — значение синего канала в диапазоне от 0.0 до 1.0.

Метод устанавливает данный цвет из значений RGB.

`.setScalar(scalar)`

`scalar` — значение в диапазоне между 0.0 и 1.0.

Метод устанавливает все три компонента цвета равными значению, переданному в параметре `scalar`.

`.setStyle(style)`

`style` — строковое значение в стиле CSS в качестве значения цвета.

Метод устанавливает данный цвет из строкового значения в стиле CSS. Например, "rgb(250, 0, 0)", "rgb(100%, 0%, 0%)", "hsl(0, 100%, 50%)", "#ff0000", "#f00" или "red" (или любое [название цвета из цветовой схемы X11](#) - поддерживаются все 140 названий цвета). Вот эта страничка Википедии о [цвете из схемы X11](#) на русском языке.

Также принимаются полупрозрачные цвета, такие как "rgba(255, 0, 0, 0.5)" и "hsla(0, 100%, 50%, 0.5)", но значение альфа-канала (канал прозрачности) отбрасывается. Обратите внимание, что названия цвета в схеме X11, состоящие из нескольких слов, вроде Dark Orange (темно-оранжевый), становятся строковым значением 'darkorange' (все буквы строчные).

`.sub(color)`

Метод вычитает из RGB компонентов данного цвета компоненты RGB цвета, переданного в параметре `color`. Если в результате получается отрицательное значение компонента, этот компонент устанавливается равным нулю.

`.toArray(array, offset)`

`array` - дополнительный массив для хранения значений цвета.

| **offset** - дополнительное смещение в массиве.
Метод возвращает массив в виде [r, g, b].

Исходники

[Color.js в этом справочнике](#)
[Color.js на Гитхабе](#)

Cylindrical

Координаты точки в [цилиндрической системе координат](#).
[Цилиндрическая система координат](#) на русском языке.

Конструктор

`Cylindrical(radius, theta, y)`

radius - расстояние от начала координат до точки в плоскости x-z. Значение по умолчанию равно 1.0.

theta - угол против часовой стрелки в плоскости x-z, измеренный в радианах от положительной оси z. Значение по умолчанию равно 0.

y - высота над плоскостью x-z. Значение по умолчанию равно 0.

Свойства

`.radius`

`.theta`

`.y`

Методы

`.clone()`

Метод возвращает новый cylindrical с теми же самыми свойствами `radius`, `theta` и `y` как у данного.

`.copy(other)`

Метод копирует значения свойств `radius`, `theta` и `y` Cylindrical, переданного в аргументе `other`, в данный cylindrical.

`.set(.radius, .theta, .y)`

Устанавливает значения свойств `radius`, `theta` и `y` данного Cylindrical.

`.setFromVector3(vec3)`

Устанавливает значения свойств `radius`, `theta` и `y` данного Cylindrical из трехмерного вектора, переданного в аргументе `vec3`. Значение свойства `radius` устанавливается как длина вектора от начала координат, измеряемая вдоль плоскости x - z, тогда как значение свойства `theta` устанавливается по его направлению на плоскости x - z, а значение свойства `y` задается из у-компоненты вектора.

Исходники

[Cylindrical.js в этом справочнике](#)

[Cylindrical.js на Гитхабе](#)

Euler

Класс, представляющий [углы Эйлера](#). Вот статья про [углы Эйлера](#) в русскоязычной Википедии.

Углы Эйлера описывают изменения при вращении объекта по его различным осям с определенными значениями для каждой оси и с указанной последовательностью вращения по осям.

Пример

```
var a = new THREE.Euler( 0, 1, 1.57, 'XYZ' );
var b = new THREE.Vector3( 1, 0, 1 );
b.applyEuler(a);
```

Конструктор

`Euler(x, y, z, order)`

- x -- угол по оси x в радианах. Дополнительный, необязательный параметр, значение по умолчанию равно 0.
- y -- угол по оси y в радианах. Дополнительный, необязательный параметр, значение по умолчанию равно 0.
- z -- угол по оси z в радианах. Дополнительный, необязательный параметр, значение по умолчанию равно 0.
- order -- строковое значение, представляющее порядок применения вращения. Дополнительный, необязательный параметр, значением по умолчанию является XYZ (символы должны быть в верхнем регистре).

Свойства

`.isEuler`

Свойство используется для проверки, является ли данный класс и производные от него классы Eulers. Значением по умолчанию является true.

Не изменяйте это свойство, так как оно используется для оптимизации внутри *three.js*.

`.order`

Порядок (последовательность) применения вращения по осям. Значением по умолчанию является XYZ, которое означает, что вначале объект будет поворачиваться вокруг своей оси X, затем вокруг своей оси Y и наконец, вокруг оси Z. Другие возможные значения: YZX, ZXY, XZY, YXZ и ZYX. Символы должны быть в верхнем регистре.

three.js использует собственные углы Тейта-Брайана. Это значит

что вращения выполняются относительно локальной системы координат. То есть, Three.js uses intrinsic Tait-Bryan angles. This means that rotations are performed with respect to the local coordinate system. That is, for order 'XYZ', the rotation is first around the local-X axis (which is the same as the world-X axis), then around local-Y (which may now be different from the world Y-axis), then local-Z (which may be different from the world Z-axis).

Если order изменен, будет вызван метод [onChangeCallback](#).

.x

Текущее значение составляющей угла по оси x.

Если значение изменено, будет вызван метод [onChangeCallback](#).

.y

Текущее значение составляющей угла по оси y.

Если значение изменено, будет вызван метод [onChangeCallback](#).

.z

Текущее значение составляющей угла по оси z.

Если значение изменено, будет вызван метод [onChangeCallback](#).

Методы

.copy (euler)

Копирует значение из Copies value of euler to this euler.

.clone ()

Возвращает новый Euler с теми же самыми параметрами, как у этого.

.equals (euler)

Checks for strict equality of this euler and euler.

.fromArray (array)

array длиной 3 или 4. Дополнительный, необязательный 4-ый

аргумент соответствует [order](#).

Assigns this euler's x angle to array[0].

Assigns this euler's y angle to array[1].

Assigns this euler's z angle to array[2].

Optionally assigns this euler's order to array[3].

```
.onChange( onChangeCallback )
```

onChangeCallback - set the value of the onChangeCallback() function.

```
.onChangeCallback( )
```

By default this is an empty function, however it can be set via onChange(). It gets called after changing the x, y, z or order properties, and also after calling most setter functions (see those for details).

```
.reorder( newOrder )
```

Resets the euler angle with a new order by creating a quaternion from this euler angle and then setting this euler angle with the quaternion and the new order.

Предупреждение:WARNING: this discards revolution information.

```
.set( x, y, z, order )
```

x - the angle of the x axis in radians.

y - the angle of the y axis in radians.

z - the angle of the z axis in radians.

order - (optional) a string representing the order that the rotations are applied.

Sets the angles of this euler transform and optionally the order and then call onChangeCallback().

```
.setFromRotationMatrix( m, order, update )
```

m - a Matrix4 of which the upper 3x3 of matrix is a pure rotation matrix (i.e. unscaled).

order - (optional) a string representing the order that the rotations are applied.

update - (optional) whether to call onChangeCallback() after applying the matrix.

Sets the angles of this euler transform from a pure rotation matrix based on the orientation specified by order.

```
.setFromQuaternion( q, order, update )
```

q - a normalized quaternion.

order - (optional) a string representing the order that the rotations are applied.

update - (optional) whether to call onChangeCallback() after applying the matrix.

Sets the angles of this euler transform from a normalized quaternion based on the orientation specified by order.

```
.setFromVector3( vector, order )
```

vector - Vector3.

order - (optional) a string representing the order that the rotations are applied.

Set the x, y and z, and optionally update the order.

onChangeCallback() is called after these changes are made.

```
.toArray( array, offset )
```

array - (optional) array to store the euler in.

offset - (optional) offset in the array.

Возвращает массив в виде [[x](#), [y](#), [z](#), [order](#)].

```
.toVector3\( \)
```

Returns the Euler's x, y and z properties as a Vector3.

Исходники

[Euler.js в этом справочнике](#)

[Euler.js на Гитхабе](#)

Frustum

[Frustums](#) are used to determine what is inside the camera's field of view. They help speed up the rendering process.

Frustums are used to determine what is inside the camera's field of view. They help speed up the rendering process - object which lie outside a camera's frustum can safely be excluded from rendering.

Статья о [усеченной пирамиде](#) в русскоязычной Википедии.

Этот класс в основном предназначен для внутреннего использования визуализатором (рендерером) для расчета усеченных пирамид камеры и теневой камеры. This class is mainly intended for use internally by a renderer for calculating a camera or shadowCamera's frustum.

Конструктор

`Frustum(p0, p1, p2, p3, p4, p5)`

`p0` -- дополнительный, необязательный параметр. Значением по умолчанию является новая [плоскость](#).

`p1` -- дополнительный, необязательный параметр. Значением по умолчанию является новая [плоскость](#).

`p2` -- дополнительный, необязательный параметр. Значением по умолчанию является новая [плоскость](#).

`p3` -- дополнительный, необязательный параметр. Значением по умолчанию является новая [плоскость](#).

`p4` -- дополнительный, необязательный параметр. Значением по умолчанию является новая [плоскость](#).

`p5` -- дополнительный, необязательный параметр. Значением по умолчанию является новая [плоскость](#).

Создает новый Frustum.

Свойства

`.planes`

Массив из 6 [плоскостей](#).

Методы

```
.clone( )
```

Return a new Frustum with the same parameters as this one.

```
.containsPoint( point )
```

| point - Vector3 to test.

Checks to see if the frustum contains the point.

```
.copy( frustum )
```

| frustum - The frustum to copy

Copies the properties of the passed frustum into this one.

```
.intersectsBox( box )
```

| box - Box3 to check for intersection.

Return true if box intersects with this frustum.

```
.intersectsObject( object )
```

Checks whether the object's bounding sphere is intersecting the Frustum. Note that the object must have a Geometry or BufferGeometry so that the bounding sphere can be calculated.

```
.intersectsSphere( sphere )
```

| sphere - Sphere to check for intersection.

Return true if sphere intersects with this frustum.

```
.intersectsSprite( sprite )
```

Checks whether the sprite is intersecting the Frustum.

```
.set ( p0, p1, p2, p3, p4, p5 )
```

Sets the current frustum from the passed planes. No plane order is implicitly implied.

```
.setFromMatrix ( matrix )
```

| matrix - Matrix4 used to set the planes

This is used by the WebGLRenderer to set up the Frustum from a Camera's projectionMatrix and matrixWorldInverse.

Исходники

[Frustum.js в этом справочнике](#)

[Frustum.js на Гитхабе](#)

Interpolant

Abstract base class of interpolants over parametric samples.

The parameter domain is one dimensional, typically the time or a path along a curve defined by the data.

The sample values can have any dimensionality and derived classes may apply special interpretations to the data.

This class provides the interval seek in a Template Method, deferring the actual interpolation to derived classes.

Time complexity is $*O(1)*$ for linear access crossing at most two points and $*O(\log N)*$ for random access, where $*N*$ is the number of positions.

References: [link:<http://www.odesign.com/template-method-pattern.html>
<http://www.odesign.com/template-method-pattern.html>

Конструктор

`Interpolant(parameterPositions, sampleValues, sampleSize, resultBu`

`parameterPositions -- array of positions`

`sampleValues -- array of samples`

`sampleSize -- number of samples`

`resultBuffer -- buffer to store the interpolation results.`

Примечание: Этот конструктор не предназначен для вызова напрямую.

Свойства

[.parameterPositions](#)

[.resultBuffer](#)

[.sampleValues](#)

[.settings](#)

Optional, subclass-specific settings structure.

[.valueSize](#)

Методы

[.evaluate\(t \)](#)

Вычисляет интерполяцию на позицию t. Evaluate the interpolant at position t.

Исходники

[Interpolant.js в этом справочнике](#)

[Interpolant.js на Гитхабе](#)

[Interpolant →](#)

CubicInterpolant

Пример

```
var interpolant = new THREE.[name](
  new Float32Array( 2 ),
  new Float32Array( 2 ),
  1,
  new Float32Array( 1 )
);

interpolant.evaluate( 0.5 );
```

Конструктор

```
CubicInterpolant( parameterPositions, sampleValues, sampleSize, res
  parameterPositions -- array of positions
  sampleValues -- array of samples
  sampleSize -- number of samples
  resultBuffer -- buffer to store the interpolation results.
```

Свойства

```
[property:null parameterPositions
```

```
[property:null resultBuffer
```

```
[property:null sampleValues
```

```
[property:Object settings
```

```
[property:null valueSize
```

Методы

```
[method:null evaluate( [page:Number t )
```

Evaluate the interpolant at position *t*.

Исходники

[CubicInterpolant.js в этом справочнике](#)

[CubicInterpolant.js на Гитхабе](#)

DiscreteInterpolant

Пример

```
var interpolant = new THREE.DiscreteInterpolant(  
    new Float32Array( 2 ),  
    new Float32Array( 2 ),  
    1,  
    new Float32Array( 1 )  
);  
  
interpolant.evaluate( 0.5 );
```

Конструктор

```
DiscreteInterpolant( parameterPositions, sampleValues, sampleSize,  
    parameterPositions -- array of positions  
    sampleValues -- array of samples  
    sampleSize -- number of samples  
    resultBuffer -- buffer to store the interpolation results.
```

Свойства

[.parameterPositions](#)

[.resultBuffer](#)

[.sampleValues](#)

[.settings](#)

[.valueSize](#)

Методы

[.evaluate\(t \)](#)

Evaluate the interpolant at position t.

Исходники

[DiscreteInterpolant.js в этом справочнике](#)

[DiscreteInterpolant.js на Гитхабе](#)

[Interpolant →](#)

LinearInterpolant

Пример

```
var interpolant = new THREE.LinearInterpolant(  
    new Float32Array( 2 ),  
    new Float32Array( 2 ),  
    1,  
    new Float32Array( 1 )  
);  
  
interpolant.evaluate( 0.5 );
```

Конструктор

```
LinearInterpolant( parameterPositions, sampleValues, sampleSize, re  
    parameterPositions -- array of positions  
    sampleValues -- array of samples  
    sampleSize -- number of samples  
    resultBuffer -- buffer to store the interpolation results.
```

Свойства

[.parameterPositions](#)

[.resultBuffer](#)

[.sampleValues](#)

[.settings](#)

[.valueSize](#)

Методы

[.evaluate\(t \)](#)

Evaluate the interpolant at position t.

Исходники

[LinearInterpolant.js в этом справочнике](#)

[LinearInterpolant.js на Гитхабе](#)

[Interpolant →](#)

QuaternionLinearInterpolant

Пример

```
var interpolant = new THREE.QuaternionLinearInterpolant(  
    new Float32Array( 2 ),  
    new Float32Array( 2 ),  
    1,  
    new Float32Array( 1 )  
);  
  
interpolant.evaluate( 0.5 );
```

Конструктор

```
QuaternionLinearInterpolant( parameterPositions, sampleValues, samp
```

parameterPositions -- array of positions

sampleValues -- array of samples

sampleSize -- number of samples

resultBuffer -- buffer to store the interpolation results.

Свойства

[.parameterPositions](#)

[.resultBuffer](#)

[.sampleValues](#)

[.settings](#)

[.valueSize](#)

Методы

[.evaluate\(t \)](#)

Evaluate the interpolant at position t.

Исходники

[QuaternionLinearInterpolant.js в этом справочнике](#)

[QuaternionLinearInterpolant.js на Гитхабе](#)

Line3

Отрезок геометрической линии, представленный начальной (start) и конечной (end) точками.

Конструктор

```
Line3( start, end )
```

start -- начало отрезка линии. Значением по умолчанию является (0, 0, 0).

end -- конец отрезка линии. Значением по умолчанию является (0, 0, 0).

Создает новую Line3.

Свойства

```
.start
```

Трехмерный вектор ([Vector3](#)), представляющий начальную точку линии.

```
.end
```

Трехмерный вектор ([Vector3](#)), представляющий конечную точку линии.

Методы

```
.applyMatrix4( matrix )
```

Метод применяет матричное преобразование к отрезку линии.

```
.at( t, optionalTarget )
```

t - используйте значения в диапазоне от 0 до 1 для возврата позиции вдоль отрезка линии.

optionalTarget - дополнительный, необязательный аргумент.

Если указан, то результат метода будет скопирован в предоставленный [Vector3](#), в противном случае будет создан новый [Vector3](#).

Метод возвращает вектор в определенной позиции вдоль линии. При $t = 0$ возвращается начальный вектор, а при $t = 1$ - конечный вектор.

`.clone()`

Возвращает новую `Line3` с теми же самыми начальным (`start`) и конечным (`end`) векторами, что и у данного.

`.closestPointToPoint(point, clampToLine, optionalTarget)`

`point` - return the closest point on the line to this point.

`clampToLine` - whether to clamp the returned value to the line segment.

`optionalTarget` - дополнительный, необязательный аргумент.

Если указан, то результат метода будет скопирован в предоставленный `Vector3`, в противном случае будет создан новый `Vector3`.

Метод возвращает точку на линии, ближайшую к точке, указанной в аргументе `point`. Если аргумент `clampToLine` указан как `true`, то возвращаемое значение будет прижато к отрезку линии.

Returns the closest point on the line. If `clampToLine` is `true`, then the returned value will be clamped to the line segment.

`.closestPointToPointParameter(point, clampToLine)`

`point` - the point for which to return a point parameter.

`clampToLine` - Whether to clamp the result to the range [0, 1].

Returns a point parameter based on the closest point as projected on the line segment. If `clampToLine` is `true`, then the returned value will be between 0 and 1.

`.copy(line)`

Метод копирует начальный (`start`) и конечный (`end`) векторы линии, переданной в аргументе `line`, в данную линию.

`.delta(optionalTarget)`

`optionalTarget` - дополнительный, необязательный аргумент.

Если указан, то результат метода будет скопирован в

предоставленный [Vector3](#), в противном случае будет создан новый [Vector3](#).

Метод возвращает дельта-вектор отрезка линии (т.е. конечный вектор - [end](#) минус начальный вектор - [start](#)).

[.distance\(\)](#)

Метод возвращает [евклидово расстояние](#) (расстояние по прямой) между начальной ([start](#)) и конечной ([end](#)) точками линии.

Вот статья русскоязычной Википедии о [Эвклидовой метрике](#).

[.distanceSq\(\)](#)

Метод возвращает квадрат [евклидова расстояния](#) (расстояние по прямой) между начальным ([start](#)) и конечным ([end](#)) векторами линии.

Вот статья русскоязычной Википедии о [Эвклидовой метрике](#).

[.equals\(line \)](#)

[line](#) - линия ([Line3](#)), которая будет сравниваться с данной линией.

Метод возвращает значение `true`, если обе (и начальная - [start](#), и конечная - [end](#)) точки равны.

[.getCenter\(optionalTarget \)](#)

[optionalTarget](#) - дополнительный, необязательный аргумент.

Если указан, то результат метода будет скопирован в предоставленный [Vector3](#), в противном случае будет создан новый [Vector3](#).

Метод возвращает центр отрезка линии.

[.set\(.start, .end \)](#)

[.start](#) - вектор, который задает начальную ([start](#)) точку линии.

[.end](#) - вектор, который задает конечную ([end](#)) точку линии.

Метод устанавливает значения начальной и конечной точек линии, копируя их из представленных векторов.

Исходники

[Line3.js в этом справочнике](#)

[Line3.js на Гитхабе](#)

Math

Объект с некоторыми вспомогательными математическими функциями.

Функции

[`.clamp\(value, min, max \)`](#)

`value` — значение, которое нужно зафиксировать.

`min` — минимальное значение.

`max` — максимальное значение.

«Зажимает» (фиксирует) значение, указанное в параметре `value` в диапазоне значений, заданных в аргументах `min` и `max`.

[`.degToRad \(degrees \)`](#)

Конвертирует градусы в радианы.

[`.euclideanModulo \(n, m \)`](#)

Вычисляет модуль (т.е. наибольший общий делитель - НОД) по алгоритму Евклида из `m % n`, то есть:

`((n % m) + m) % m`

Примечание переводчика: Символ `%` в JavaScript это оператор взятия остатка от деления.

[`.generateUUID \(\)`](#)

Метод создает [UUID \(универсальный уникальный идентификатор\)](#).

[`.isPowerOfTwo \(n \)`](#)

Возвращает значение true, если n является степенью числа 2.

[lerp\(x, y, t \)](#)

x -- начальная точка.

y -- конечная точка.

t -- коэффициент интерполяции в заданном интервале [0, 1].

Возвращает [линейно интерполированное](#) значение для двух известных точек (x и y), основываясь на заданном интервале (t) - при t = 0 будет возвращен x, а при t = 1 будет возвращен y.
Статья из русскоязычной Википедии о [линейной интерполяции](#).

[.mapLinear\(x, a1, a2, b1, b2 \)](#)

x — значение для отображения.

a1 — минимальное значение для диапазона A.

a2 — максимальное значение для диапазона A.

b1 — минимальное значение для диапазона B.

b2 — максимальное значение для диапазона B.

[Линейное отображение](#) x из диапазона [a1, a2] в диапазон [b1, b2].

А вот статья из англоязычной Википедии о [Linear mapping](#).

[.nearestPowerOfTwo \(n \)](#)

Возвращает ближайшую степень числа 2 для заданного числа n.

[.nextPowerOfTwo \(n \)](#)

Возвращает ближайшую степень числа 2 для заданного числа n, но которая больше числа n.

[.radToDeg \(radians \)](#)

Конвертирует радианы в градусы.

[.randFloat \(low, high \)](#)

Возвращает случайное число с плавающей запятой в интервале чисел от low (нижняя граница) до high (верхняя граница интервала).

.randFloatSpread (range)

Возвращает случайное число с плавающей запятой в интервале чисел от - range/2 до range/2.

.randInt (low, high)

Возвращает случайное целое число в интервале от low (нижняя граница) до high (верхняя граница интервала).

.smoothstep(x, min, max)

x -- значение для вычисления, в зависимости от его позиции между значениями min и max.

min -- любое значение x меньше значения min будет равно 0.

max -- любое значение x больше значения max будет равно 1.

Возвращает значение в диапазоне между 0 и 1, представляющее Returns a value between 0-1 that represents the percentage that x has moved between min and max, but smoothed or slowed down the closer X is to the min and max.

Подробностисмотрите в статье Википедии [Smoothstep](#)

[method:Float .smootherstep([page:Float x, [page:Float min, [pag

x -- The value to evaluate based on its position between min and max.

min -- любое значение x меньше значения min будет равно 0.

max -- любое значение x больше значения max будет равно 1.

Возвращает значение в диапазоне от 0 до 1. Функция работает также как smoothstep, но более плавно. but more smooth. Эта функция - вариант функции smoothstep, имеющий A variation on smoothstep that has zero 1st and 2nd order derivatives at x=0 and x=1.

Исходники

[Math.js в этом справочнике](#)

[Math.js на Гитхабе](#)

Matrix3

Класс, представляющий [матрицу](#) 3×3 . Вот статья русскоязычной Википедии о [матрицах](#).

Пример

```
var m = new Matrix3();
```

Примечание о упорядочивании по рядам и колонкам

Метод [set\(\)](#) принимает аргументы, [упорядоченные по рядам](#), в то время как внутри они хранятся в массиве [элементов](#), упорядоченном по колонкам.

Это значит, что вызов

```
m.set( 11, 12, 13,
       21, 22, 23,
       31, 32, 33 );
```

приведет в результате к массиву элементов, содержащему:

```
m.elements = [ 11, 21, 31,
                 12, 22, 32,
                 13, 23, 33 ];
```

и внутри все вычисления выполняются с использованием упорядочивания по колонкам. Однако, поскольку фактически порядок размещения элементов в матрице, с точки зрения математики, не имеет никакого значения, а большинство людей привыкли представлять матрицы упорядоченными по строкам, то в документации *three.js* матрицы также показываются упорядоченными по строкам. Просто имейте ввиду, чтобы понять смысл расчетов, при чтении исходного кода нужно переставлять ([транспонировать](#), в англоязычном варианте - [transpose](#)) любые матрицы, представленные здесь. То есть, нужно каждую строчку исходной матрицы записать в виде столбца в том же порядке.

Конструктор

```
Matrix3( )
```

Создает и инициализирует Matrix3 как [единичную матрицу](#) 3×3 .
Вот статья русскоязычной Википедии о [единичной матрице](#).

Свойства

.elements

Список значений матрицы, упорядоченный по колонкам.

.isMatrix3

Свойство используется для проверки, является ли данный и производные от него классы матрицами 3×3 . Значением по умолчанию является `true`.

Его нельзя изменять, так как оно используется внутри *three.js* для оптимизации.

Методы

.applyToBufferAttribute(attribute)

attribute - атрибут из чисел с плавающей запятой, представляющий трехмерные векторы.

Умножает (применяет) данную матрицу на каждый трехмерный вектор, переданный в attribute.

.clone()

Создает новую матрицу Matrix3 с точно такими же элементами как в данной.

.copy(m)

Метод копирует элементы матрицы, указанной в параметре m, в данную матрицу.

.determinant()

Метод вычисляет и возвращает определитель (determinant) данной матрицы.

.equals(m)

Возвращает значение `true`, если данная матрица и матрица,

указанная в параметре `m` равны.

`.fromArray(array, offset)`

`array` - массив, из которогочитываются элементы.

`offset` - смещение в массиве (дополнительный, необязательный аргумент). Значением по умолчанию является 0.

Устанавливает элементы данной матрицы на основе массива (параметр `array`, упорядоченного по колонкам).

`.getInverse(m, throwOnDegenerate)`

`m` - матрица, которая берется для инвертирования.

`throwOnDegenerate` - если установлено значение `true`, то при вырожденной матрице (не обратимой) будет выброшено сообщение об ошибке. Дополнительный, необязательный аргумент.

Метод устанавливает данную матрицу обратной (`inverse`) по отношению к матрице, переданной в параметре `m`, при помощи аналитического метода. Если аргумент `throwOnDegenerate` не установлен, а матрица не обратима (т.е. обратной матрицы не существует), то данная матрица устанавливается как единичная матрица 3×3 . Англоязычный вариант единичной матрицы называется identity matrix.

`.getNormalMatrix(m)`

`m` - четырехмерная матрица Matrix4.

Устанавливает данную матрицу как верхнюю левую часть 3×3 нормальной матрицы (`normal matrix`) из переданной в аргументе `m` четырехмерной матрицы. Нормальная матрица - это инверсия
The normal matrix is the inverse transpose of the matrix `m`.

`.identity()`

Метод сбрасывает данную матрицу в единичную матрицу 3×3 :

```
1, 0, 0
0, 1, 0
0, 0, 1
```

```
.multiply( m )
```

Производится последующее умножение данной матрицы с матрицей, переданной в аргументе m.

```
.multiplyMatrices( a, b )
```

Устанавливает данную матрицу как результат действия a \times b.

```
.multiplyScalar( s )
```

Метод перемножает каждый компонент данной матрицы на скалярное значение, переданное в аргументе s.

```
.premultiply( m )
```

Производится предварительное умножение данной матрицы с матрицей, переданной в аргументе m.

```
.set( n11, n12, n13, n21, n22, n23, n31, n32, n33 )
```

n11 -- значение, которое требуется поместить в 1 ряду и 1 колонке.

n12 -- значение, которое требуется поместить в 1 ряду и 2 колонке.

...

...

n32 -- значение, которое требуется поместить в 3 ряду и 2 колонке.

n33 -- значение, которое требуется поместить в 3 ряду и 3 колонке.

Метод устанавливает значения матрицы 3×3 в заданной последовательности, упорядоченной по строкам. Sets the 3×3 matrix values to the given row-major sequence of values.

```
.setFromMatrix4( m )
```

Устанавливает эту матрицу как верхнюю матрицу 3×3 из четырехмерной матрицы (Matrix4), представленной в аргументе m.

```
.setUvTransform( tx, ty, sx, sy, rotation, cx, cy )
```

tx - offset x
ty - offset y
sx - repeat x
sy - repeat y
rotation - угол поворота (в радианах).
cx - center x of rotation
cy - center y of rotation

Sets the UV transform matrix from offset, repeat, rotation, and center.

[.toArray\(array, offset \)](#)

array - массив для хранения результирующего вектора (дополнительный, необязательный параметр). Если параметр не задан, будет создан новый массив.
offset - смещение в массиве, где будет размещен результат (дополнительный, необязательный параметр).

Записывает элементы данной матрицы в массив, упорядочивая их по колонкам.

[.transpose\(\)](#)

Транспонирует данную матрицу на месте. Вот англоязычный вариант статьи о транспонировании.

[.transposeIntoArray\(array \)](#)

array - массив для хранения результирующего вектора.
Транспонирует данную матрицу в предоставленный массив и возвращает саму себя без изменений.

Исходники

[Matrix3.js в этом справочнике](#)

[Matrix3.js на Гитхабе](#)

Matrix4

Класс, представляющий [матрицу](#) 4×4 (англоязычный вариант [matrix](#)). Наиболее распространенным использование матрицы 4×4 в трехмерной компьютерной графике является [матрица перехода](#) (в англоязычном варианте [Transformation Matrix](#)). Чтобы иметь представление о матрицах перехода (матрицах трансформаций), используемых в WebGL, ознакомьтесь с [этим руководством](#). Вот эта же [статья на русском языке](#).

При помощи перемножения с матрицей, становятся возможными преобразования трехмерного вектора, представляющего точку в трехмерном пространстве, такие как перенос, вращение, сдвиг, масштабирование, отражение, ортогональное или перспективное проецирование и так далее. Это называется применением матрицы к вектору.

Каждый [трехмерный объект \(Object3D\)](#) имеет три, взаимосвязанных матрицы has three associated Matrix4s:

- [Object3D.matrix](#): здесь хранится локальное изменение объекта.
- [Object3D.matrixWorld](#): глобальное или "мировое" изменение объекта. Это преобразование объекта относительно его родителя. Если у объекта нет родителя, то здесь все идентично с локальным изменением.
- [Object3D.modelViewMatrix](#): здесь представлено преобразование объекта относительно системы координат камеры. Матрица `modelViewMatrix` объекта - это матрица `matrixWorld` предварительно перемноженная с матрицей `matrixWorldInverse` камеры.

[Камеры](#) имеют две дополнительные матрицы Matrix4:

- [Camera.matrixWorldInverse](#): матрица просмотра - инверсия матрицы "мира" (`matrixWorld`) камеры.
- [Camera.projectionMatrix](#): представляет сведения о том, как проецировать сцену на пространство клипа.

Примечание: [Object3D.normalMatrix](#) является не Matrix4, а [Matrix3](#).

Примечание о упорядочивании по рядам и колонкам

Метод [set\(\)](#) принимает аргументы, [упорядоченные по рядам](#), в то время как внутри они хранятся в массиве [элементов](#), упорядоченном по колонкам.

Это значит, что вызов

```
var m = new Matrix4();

m.set( 11, 12, 13, 14,
       21, 22, 23, 24,
       31, 32, 33, 34,
       41, 42, 43, 44 );
```

приведет к массиву элементов, содержащему:

```
m.elements = [ 11, 21, 31, 41,
                12, 22, 32, 42,
                13, 23, 33, 43,
                14, 24, 34, 44 ];
```

и внутри все вычисления выполняются с использованием упорядочивания по колонкам. Однако, поскольку фактически порядок размещения элементов в матрице, с точки зрения математики, не имеет никакого значения, а большинство людей привыкли представлять матрицы упорядоченными по строкам, то в документации *three.js* матрицы также показываются упорядоченными по строкам. Просто имейте ввиду, чтобы понять смысл расчетов, при чтении исходного кода нужно переставлять ([транспонировать](#), в англоязычном варианте - [transpose](#)) любые матрицы, представленные здесь. То есть, нужно каждую строчку исходной матрицы записать в виде столбца в том же порядке.

Конструктор

`Matrix4()`

Создает и инициализирует Matrix4 как [единичную матрицу](#) 4×4 (в англоязычном варианте это [identity matrix](#)).

Свойства

`.elements`

Список значений матрицы, [упорядоченный по колонкам](#).

`.isMatrix4`

Свойство используется для проверки, является ли данный и производные от него классы матрицами 4×4 . Значением по

умолчанию является true.

Его нельзя изменять, так как оно используется внутри **three.js** для оптимизации.

Методы

`.applyToBufferAttribute(attribute)`

`attribute` - атрибут из чисел с плавающей запятой, представляющий трехмерные векторы.

Умножает (применяет) данную матрицу на каждый трехмерный вектор, переданный в `attribute`.

`.clone()`

Создает новую матрицу Matrix4 с точно такими же элементами как в данной.

`.compose(position, quaternion, scale)`

Метод устанавливает в данную матрицу изменения, составленные из положения (параметр `position`), кватерниона (параметр `quaternion`) и масштаба (параметр `scale`). При этом внутри метода сначала вызывается метод `makeRotationFromQuaternion(quaternion)`, затем метод `scale(scale)` и наконец метод `setPosition(position)`.

`.copy(m)`

Метод копирует элементы матрицы, указанной в параметре `m`, в данную матрицу.

`.copyPosition(m)`

Копирует компонент перемещения (трансляции) из матрицы, переданной в параметре `m`, в компонент перемещения данной матрицы.

`.decompose(position, quaternion, scale)`

Метод разлагает данную матрицу на составные части -

положение ([position](#)), кватернион ([quaternion](#)) и масштаб ([scale](#)).

[.determinant\(\)](#)

Метод вычисляет и возвращает [определитель \(determinant\)](#) данной матрицы.

Выполняется на основе метода, изложенного [здесь](#).

[.equals\(m \)](#)

Возвращает значение `true`, если данная матрица и матрица, указанная в параметре `m` равны.

[.extractBasis\(xAxis, yAxis, zAxis \)](#)

Метод извлекает [базис \(basis\)](#), т.е. основные элементы матрицы, по трем представленным осевым векторам. Если задана такая матрица:

```
a, b, c, d,  
e, f, g, h,  
i, j, k, l,  
m, n, o, p
```

то [xAxis](#), [yAxis](#), [zAxis](#) будут установлены как:

```
xAxis = (a, e, i)  
yAxis = (b, f, j)  
zAxis = (c, g, k)
```

[.extractRotation\(m \)](#)

Метод извлекает из предоставленной матрицы `m` компонент вращения и устанавливает его как компонент вращения данной матрицы.

[.fromArray\(array, offset \)](#)

[array](#) - массив, из которогочитываются элементы.

[offset](#) - смещение в массиве (дополнительный, необязательный аргумент). Значением по умолчанию является 0.

Устанавливает элементы данной матрицы на основе массива (параметр [array](#)), [упорядоченного по колонкам](#).

`.getInverse(m, throwOnDegenerate)`

m - матрица, которая берется для инвертирования.
throwOnDegenerate - если установлено значение `true`, то при вырожденной матрице (необратимой) будет выброшено сообщение об ошибке. Дополнительный, необязательный аргумент.

Метод устанавливает данную матрицу обратной (`inverse`) по отношению к матрице, переданной в параметре m, при помощи способа, описанного здесь. Если аргумент throwOnDegenerate не установлен, а матрица необратима (т.е. обратной матрицы не существует), то данная матрица устанавливается как единичная матрица 4×4 .

`.getMaxScaleOnAxis()`

Метод получает максимальное значение масштаба по трем осям.

`.identity()`

Метод сбрасывает данную матрицу в единичную матрицу (англоязычный вариант - identity matrix).

`.lookAt(eye, center, up)`

Метод создает матрицу вращения, наблюдающую из позиции, переданной в параметре eye, в направлении координат, переданных в параметре center, ориентируясь по вектору, переданному в up.

`.makeRotationAxis(axis, theta)`

axis — оси вращения, должны быть нормализованы.
theta — угол поворота в радианах.

Метод устанавливает данную матрицу как изменение вращения вокруг axis на угол theta в радианах. Это немного спорная, но математически верная альтернатива вращению посредством кватернионов (Quaternion). Посмотрите обсуждение этой темы здесь.

```
.makeBasis( xAxis, yAxis, zAxis )
```

Set this to the basis matrix consisting of the three provided basis vectors:

```
xAxis.x, yAxis.x, zAxis.x, 0,  
xAxis.y, yAxis.y, zAxis.y, 0,  
xAxis.z, yAxis.z, zAxis.z, 0,  
0, 0, 0, 1
```

```
.makePerspective( left, right, top, bottom, near, far )
```

Метод создает матрицу перспективной проекции. Она используется внутри **three.js** методом PerspectiveCamera.updateProjectionMatrix.

```
.makeOrthographic( left, right, top, bottom, near, far )
```

Метод создает матрицу ортогональной проекции. Она используется внутри **three.js** методом OrthographicCamera.updateProjectionMatrix.

```
.makeRotationFromEuler( euler )
```

Устанавливает компонент вращения (левую верхнюю матрицу 33) данной матрицы как поворот, определенный представленными углами Эйлера (параметр euler). Оставшаяся часть матрицы устанавливается как единичная. В зависимости от свойства order угла Эйлера, возможны шесть вариантов. Полный список смотрите на этой странице. Вот матрица поворота на русском языке и отдельно rotation matrix на английском.

```
.makeRotationFromQuaternion( q )
```

Устанавливает компонент вращения данной матрицы как поворот, определенный кватернионом (параметр q), как описано здесь. Оставшаяся часть матрицы устанавливается как единичная. Так при заданном $q = w + xi + yj + zk$, результирующей матрицей будет:

$$\begin{matrix} 1-2y^2-2z^2 & 2xy-2zw & 2xz+2yw & 0 \\ 2xy+2zw & 1-2x^2-2z^2 & 2yz-2xw & 0 \\ 2xz-2yw & 2yz+2xw & 1-2x^2-2y^2 & 0 \\ 0 & 0 & 0 & 1 \end{matrix}$$

.makeRotationX(theta)

| theta — угол поворота в радианах.

Метод устанавливает данную матрицу как изменение поворота вокруг оси X на угол, указанный в параметре theta (θ), в радианах. В результате получится матрица:

| | | | |
|---|----------------|-----------------|---|
| 1 | 0 | 0 | 0 |
| 0 | $\cos(\theta)$ | $-\sin(\theta)$ | 0 |
| 0 | $\sin(\theta)$ | $\cos(\theta)$ | 0 |
| 0 | 0 | 0 | 1 |

.makeRotationY(theta)

| theta — угол поворота в радианах.

Метод устанавливает данную матрицу как изменение поворота вокруг оси Y на угол, указанный в параметре theta (θ), в радианах. В результате получится матрица:

| | | | |
|-----------------|---|----------------|---|
| $\cos(\theta)$ | 0 | $\sin(\theta)$ | 0 |
| 0 | 1 | 0 | 0 |
| $-\sin(\theta)$ | 0 | $\cos(\theta)$ | 0 |
| 0 | 0 | 0 | 1 |

.makeRotationZ(theta)

| theta — угол поворота в радианах.

Метод устанавливает данную матрицу как изменение поворота вокруг оси Z на угол, указанный в параметре theta (θ), в радианах. В результате получится матрица:

| | | | |
|----------------|-----------------|---|---|
| $\cos(\theta)$ | $-\sin(\theta)$ | 0 | 0 |
| $\sin(\theta)$ | $\cos(\theta)$ | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 1 |

.makeScale(x, y, z)

| x - величина масштабирования по оси X.

| y - величина масштабирования по оси Y.

| z - величина масштабирования по оси Z.

Устанавливает данную матрицу в измененном масштабе:

| | | | |
|----|----|----|----|
| x, | 0, | 0, | 0, |
| 0, | y, | 0, | 0, |

```
0, 0, z, 0,  
0, 0, 0, 1
```

`.makeShear(x, y, z)`

| `x` - величина сдвига по оси X.

| `y` - величина сдвига по оси Y.

| `z` - величина сдвига по оси Z.

Устанавливает данную матрицу с изменением на величину сдвига:

```
1, y, z, 0,  
x, 1, z, 0,  
x, y, 1, 0,  
0, 0, 0, 1
```

`.makeTranslation(x, y, z)`

| `x` - the amount to translate in the X axis.

| `y` - the amount to translate in the Y axis.

| `z` - the amount to translate in the Z axis.

Sets this matrix as a translation transform:

```
1, 0, 0, x,  
0, 1, 0, y,  
0, 0, 1, z,  
0, 0, 0, 1
```

`.multiply(m)`

Post-multiplies this matrix by m.

`.multiplyMatrices(a, b)`

Sets this matrix to a x b.

`.multiplyScalar(s)`

Multiplies every component of the matrix by a scalar value s.

`.premultiply(m)`

Pre-multiplies this matrix by m.

`.scale(v)`

Multiples the columns of this matrix by vector v.

```
.set( n11, n12, n13, n14, n21, n22, n23, n24, n31, n32, n33, n34,
```

Метод устанавливает все элементы данной матрицы Set the elements of this matrix to the supplied row-major values n11, n12, ... n44. Sets all fields of this matrix to the supplied row-major values n11..n44.

```
.setPosition( v )
```

Sets the position component for this matrix from vector v, without affecting the rest of the matrix - i.e. if the matrix is currently:

```
a, b, c, d,  
e, f, g, h,  
i, j, k, l,  
m, n, o, p
```

This becomes:

```
a, b, c, v.x,  
e, f, g, v.y,  
i, j, k, v.z,  
m, n, o, p
```

```
.toArray( array, offset )
```

array - (optional) array to store the resulting vector in.

offset - (optional) offset in the array at which to put the result.

Writes the elements of this matrix to an array in column-major format.

```
.transpose( )
```

Transposes this matrix.

Исходники

[Matrix4.js в этом справочнике](#)

[Matrix4.js на Гитхабе](#)

Plane

Двумерная поверхность, расширяемая до бесконечности в

трехмерном пространстве, представленная в [Hessian normal form](#)
[нормальным вектором](#) единичной длины и константой.

Конструктор

`Plane(normal, constant)`

[normal](#) - трехмерный вектор ([Vector3](#)) единичной длины, определяющий нормаль к плоскости. Дополнительный, необязательный аргумент. Значением по умолчанию является `(1, 0, 0)`.

[constant](#) - the signed distance from the origin to the plane.

Дополнительный, необязательный параметр, значение по умолчанию равно 0.

Свойства

[.normal](#)

[.constant](#)

Методы

[.applyMatrix4\(matrix, optionalNormalMatrix \)](#)

matrix - the Matrix4 to apply.

optionalNormalMatrix - (optional) pre-computed normal Matrix3 of the Matrix4 being applied.

Apply a Matrix4 to the plane. The matrix must be an affine, homogeneous transform. If supplying an optionalNormalMatrix, it can be created like so:

```
var optionalNormalMatrix = new THREE.Matrix3().getNormalMatrix( m
```

[.clone\(\)](#)

Возвращает новую плоскость с такими же [нормалью](#) и [константой](#), как у этой.

[.coplanarPoint\(optionalTarget \)](#)

optionalTarget - (optional) if specified, the result will be copied into this Vector3, otherwise a new Vector3 will be created.

Returns a Vector3 coplanar to the plane, by calculating the projection of the normal vector at the origin onto the plane.

```
.copy( plane )
```

Copies the values of the passed plane's normal and constant properties to this plane.

```
.distanceToPoint( point )
```

Возвращает расстояние (со знаком, т.е. положительное или отрицательное) от данной точки до этой плоскости.

```
.distanceToSphere( sphere )
```

Возвращает расстояние (со знаком, т.е. положительное или отрицательное) от данной сферы до этой плоскости. Returns the signed distance from the sphere to the plane.

```
.equals( plane )
```

Checks to see if two planes are equal (their normal and constant properties match).

```
.intersectLine( line, optionalTarget ) or undefined
```

line - the Line3 to check for intersection.

optionalTarget - (optional) if specified, the result will be copied into this Vector3, otherwise a new Vector3 will be created.

Returns the intersection point of the passed line and the plane.

Returns undefined if the line does not intersect. Returns the line's starting point if the line is coplanar with the plane.

```
.intersectsBox( box )
```

box - the Box3 to check for intersection.

Determines whether or not this plane intersects box.

```
.intersectsLine( line )
```

line - the Line3 to check for intersection.

Tests whether a line segment intersects with (passes through) the plane.

```
.intersectSphere( sphere )
```

| sphere - the Sphere to check for intersection.

Determines whether or not this plane intersects sphere.

```
.negate( )
```

Negates both the normal vector and the constant.

```
.normalize( )
```

Normalizes the normal vector, and adjusts the constant value accordingly.

```
.projectPoint( point, optionalTarget )
```

| point - the Vector3 to project onto the plane.

| optionalTarget - (optional) if specified, the result will be copied into this Vector3, otherwise a new Vector3 will be created.

Projects a point onto the plane.

```
.set( normal, constant )
```

| normal - a unit length Vector3 defining the normal of the plane.

| constant - the signed distance from the origin to the plane. Default is 0.

Sets the plane's normal and constant properties.

```
.setComponents( x, y, z, w )
```

| x - x value of the unit length normal vector.

| y - y value of the unit length normal vector.

| z - z value of the unit length normal vector.

| w - the value of the plane's constant property.

Set the individual components that define the plane.

```
.setFromCoplanarPoints( a, b, c )
```

| a - первая точка на плоскости.

| b - вторая точка на плоскости.

| c - третья точка на плоскости.

Определяет плоскость на основе трех предоставленных точек.

Defines the plane based on the 3 provided points. The winding order is assumed to be counter-clockwise, and determines the direction of the normal.

```
.setFromNormalAndCoplanarPoint ( normal, point ) this
```

 | normal - a unit length Vector3 defining the normal of the plane.

 | point - Vector3

Устанавливает свойства плоскости Sets the plane's properties as defined by a normal and an arbitrary coplanar point.

```
.translate ( offset )
```

 | offset - the amount to move the plane by.

Translates the plane by the distance defined by the offset vector. Note that this only affects the plane constant and will not affect the normal vector.

Исходники

[Plane.js в этом справочнике](#)

[Plane.js на Гитхабе](#)

Quaternion

Реализация [кватерниона](#). Вот [статья Википедии о кватернионе](#) на русском языке.

Кватернионы используются при расчетах [вращающихся элементов](#) (вот [эта статья](#) Википедии на русском), при этом, среди других преимуществ, не встречаясь с ужасной проблемой [шарнирного замка](#). Вот статья Википедии о [шарнирном замке](#) на русском языке.

Пример

```
var quaternion = new THREE.Quaternion();
quaternion.setFromAxisAngle( new THREE.Vector3( 0, 1, 0 ), Math.PI

var vector = new THREE.Vector3( 1, 0, 0 );
```

```
vector.applyQuaternion( quaternion );
```

Конструктор

```
Quaternion( x, y, z, w )
```

- | x - координата x.
- | y - координата y
- | z - координата z
- | w - координата w

Свойства

.x

Изменение этого свойства приведет к вызову метода [onChangeCallback](#).

.y

Изменение этого свойства приведет к вызову метода [onChangeCallback](#).

.z

Изменение этого свойства приведет к вызову метода [onChangeCallback](#).

.w

Изменение этого свойства приведет к вызову метода [onChangeCallback](#).

Методы

.clone ()

Создает новый Quaternion с такими же свойствами x, y, z и w как у данного.

.conjugate ()

Возвращает [конъюгат](#) вращения к данному кватерниону.

Конъюгат кватерниона представляет собой тоже самое вращение в обратном направлении вокруг осей вращения.

`.copy (q)`

Метод копирует свойства `x`, `y`, `z` и `w` кватерниона, указанного в параметре `q`, в данный кватернион.

`.equals (v)`

`v` - кватернион, с которым будет сравниваться данный кватернион.

Метод сравнивает свойства `x`, `y`, `z` и `w` кватерниона, указанного в параметре `v`, с аналогичными свойствами данного кватерниона, чтобы определить представлено ли в них одно и то же вращение.

`.dot (v)`

Метод вычисляет [скалярное произведение](#) (в англ. варианте - [dot product](#)) данного кватерниона и кватерниона, представленного в аргументе `v`.

`.fromArray (array, offset)`

`array` - массив в формате (`x`, `y`, `z`, `w`), используемый для построения кватерниона.

`offset` - смещение в массиве (дополнительный, необязательный параметр).

Устанавливает свойства `x`, `y`, `z` и `w` данного кватерниона из массива.

`.inverse()`

Метод инвертирует данный кватернион - вычисляет [конъюгат](#) и затем [нормализует](#) результат.

`.length()`

Метод вычисляет [евклидово расстояние \(Euclidean length\)](#), т.е. длину прямой линии данного кватерниона, рассматриваемого в качестве четырехмерного вектора.

`.lengthSq()`

Метод вычисляет квадрат [евклидова расстояния \(Euclidean length\)](#), т.е. квадрат длины прямой линии данного кватерниона, рассматриваемого в качестве четырехмерного вектора. Его удобно применять при сравнении длин двух кватернионов, поскольку это несколько более эффективный расчет чем в методе [length](#).

`.normalize ()`

Метод [нормализует](#) данный кватернион - то есть, вычисляет кватернион, выполняющий такое же вращение, как и данный, но имеет длину равную 1.

Вот статья Википедии о [нормализованном, т.е. единичном векторе](#) на русском языке.

`.multiply (q)`

Метод умножает данный кватернион на кватернион, указанный в аргументе [q](#).

`.multiplyQuaternions (a, b)`

Устанавливает этот кватернион равным [a × b](#).

Умножение производится в соответствии с методом, приведенным [здесь](#).

`.onChange (onChangeCallback)`

Устанавливает метод [onChangeCallback](#).

`.onChangeCallback ()`

Данная функция вызывается всякий раз, когда происходит одно из следующих событий:

- Изменились свойства [x](#), [y](#), [z](#) или [w](#).
- Был вызов функций [set](#), [copy](#), [clone](#), [setFromAxisAngle](#), [setFromRotationMatrix](#), [conjugate](#), [normalize](#), [multiplyQuaternions](#), [slerp](#) или [fromArray](#).
- Вызывалась функция [setFromEuler](#) со своим аргументом [update](#), установленным как `true`.

По умолчанию это "пустая" функция, однако при необходимости вы можете ее изменить, воспользовавшись методом [onChange\(onChangeCallback \)](#).

[.premultiply\(q \)](#)

Метод предварительно умножает данный кватернион с заданным в аргументе q.

[.slerp\(qb, t \)](#)

[.qb](#) -- другой кватернион вращения.

[t](#) -- коэффициент интерполяции (interpolation factor) в закрытом (замкнутом) интервале (closed interval) [0, 1].

Метод обрабатывает сферическую линейную интерполяцию между кватернионами. Аргумент t представляетHandles the spherical linear interpolation between quaternions. *t* represents the amount of rotation between this quaternion (where *t* is 0) and quaternionB (where *t* is 1). This quaternion is set to the result.

Также посмотрите статическую версию метода slerp ниже.

```
// rotate a mesh towards a target quaternion  
// поворачиваем сетку (mesh) к целевому кватерниону  
mesh.quaternion.slerp( endQuaternion, 0.01 );
```

Примечание переводчика: Вот статья английской Википедии о [сферической линейной интерполяции](#). А здесь статья gamedev.ru о [интерполяции кватернионов](#) на русском языке.

[set\(x, y, z, w \)](#)

Устанавливает свойства x, y, z, w данного кватерниона.

[.setFromAxisAngle\(axis, angle \)](#)

Устанавливает этот кватернион из вращения, заданного осью и углом. Sets this quaternion from rotation specified by axis and angle.

Adapted from

[link:<http://www.euclideanspace.com/maths/geometry/rotations/conversions/quaternionFromEuler/>]

Предполагается, что axis нормализованы, а angle в радианах.

[.setFromEuler\(euler \)](#)

Sets this quaternion from rotation specified by Euler angle.

.setFromRotationMatrix(m)

Sets this quaternion from rotation component of *m*.

Adapted from

[link:[http://www.euclideanspace.com/mathematics/geometry/rotations/conversi](http://www.euclideanspace.com/mathematics/geometry/rotations/conversions/matrix/quaternion.htm)

[method:Quaternion .setFromUnitVectors([page:Vector3 vFrom, [pag

Sets this quaternion to the rotation required to rotate direction vector *vFrom* to direction vector *vTo*.

Adapted from [link:<http://lolengine.net/blog/2013/09/18/beautiful-maths-quaternion-from-vectors>.

vFrom and *vTo* are assumed to be normalized.

.toArray(array, offset)

array -- дополнительный, необязательный массив для хранения кватерниона. Если этот параметр не указан, будет создан новый массив.

offset -- дополнительное, необязательное смещение в выходном массиве. (optional) if specified, the result will be copied into this Array.

Метод возвращает числовые элементы данного кватерниона как массив в формате [x, y, z, w].

Статические методы

Статические методы (в противоположность методам для отдельных экземпляров) разработаны для вызова непосредственно от класса, а не от отдельного экземпляра. Поэтому для использования статической версии метода, вызов делается так:

```
THREE.Quaternion.slerp( qStart, qEnd, qTarget, t );
```

Для сравнения, чтобы вызвать "обычный" или "экземплярный" метод slerp нужно сделать следующее:

```
// instantiate a quaternion with default values
// создаем экземпляр кватерниона со значениями по умолчанию
var q = new THREE.Quaternion();

// call the instanced slerp method (вызов "экземплярного" метода slerp)
q.slerp( qb, t )
```

```
.slerp( qStart, qEnd, qTarget, t )
```

qStart -- начальный кватернион (когда t равно 0).

qEnd -- конечный кватернион (когда t равно 1).

qTarget -- целевой кватернион, который устанавливается в результате операции.

t -- коэффициент интерполяции в замкнутом интервале [0, 1].

В отличие от обычного метода, статическая версия `slerp` в результате операции сферического линейного интерполирования устанавливает целевой кватернион.

```
// Code setup (код настройки)
var startQuaternion = new THREE.Quaternion().set( 0, 0, 0, 1 );
var endQuaternion = new THREE.Quaternion().set( 1, 1, 1, 1 ).no
var t = 0;

// Update a mesh's rotation in the loop (обновление вращения се
t = ( t + 0.01 ) % 1; // constant angular momentum (постоянный
THREE.Quaternion.slerp( startQuaternion, endQuaternion, mesh.qu
```

```
.slerpFlat( dst, dstOffset, src0, src0ffset0, src1, src0ffset1, t )
```

dst -- выходной массив.

dstOffset -- смещение в выходном массиве.

src0 -- исходный массив начального кватерниона.

src0ffset0 -- смещение в массиве src0.

src1 -- исходный массив целевого кватерниона.

src0ffset1 -- смещение в массиве src1.

t -- коэффициент интерполяции в замкнутом интервале [0, 1].

Работает также как и статический метод `slerp`, описанный выше, но оперирует непосредственно с "плоскими" массивами чисел.

Исходники

[Quaternion.js в этом справочнике](#)

[Quaternion.js на Гитхабе](#)

Ray

Луч (ray), который испускается из источника в определенном направлении. Данный класс используется [RayCaster'ом](#) для оказания помощи с [raycasting](#). [Вот описание рейкастинга](#) в Википедии на русском языке. Рейкастинг используется для выборки курсором мыши (вычисляет какие из объектов в трехмерном пространстве накрывает курсор мыши) среди других объектов.

Конструктор

`Ray(origin, direction)`

[origin](#) -- источник луча ([Ray](#)). Дополнительный, необязательный параметр, значением по умолчанию является трехмерный вектор ([Vector3](#)) с координатами (0, 0, 0).
[direction](#) -- Vector3 The direction of the Ray. This must be normalized (with [Vector3.normalize](#)) for the methods to operate properly.
Default is a Vector3 at (0, 0, 0).

Создает новый Ray.

Свойства

[.origin](#)

Источник луча ([Ray](#)). Значением по умолчанию является трехмерный вектор ([Vector3](#)) с координатами (0, 0, 0).

[.direction](#)

Направление луча ([Ray](#)). Для правильной работы методов, этот вектор должен быть нормализован (с помощью метода [normalize](#)).

Методы

[.applyMatrix4\(\[matrix4\]\(#\) \)](#)

[matrix4](#) -- [page:Matrix4 The [page:Matrix4 to transform this [page:Ray by.
Transform this [page:Ray by the [page:Matrix4.

```
.at( t, optionalTarget )
```

| t - расстояние вдоль луча ([Ray](#)) the distance along the Ray to retrieve a position for.

| optionalTarget - дополнительный, необязательный аргумент.

| Если указан, то результат метода будет скопирован в предоставленный [Vector3](#), в противном случае будет создан новый [Vector3](#).

Метод получает трехмерный вектор ([Vector3](#)) that is a given distance along this [page:Ray].

```
.clone( )
```

Создает новый луч (Ray) со свойствами [origin](#) и [direction](#) идентичными свойствам данного луча.

```
.closestPointToPoint( point, optionalTarget )
```

| point - the point to get the closest approach to.

| optionalTarget - дополнительный, необязательный аргумент.

| Если указан, то результат метода будет скопирован в предоставленный [Vector3](#), в противном случае будет создан новый [Vector3](#).

Get the point along this Ray that is closest to the Vector3 provided.

```
.copy( ray )
```

Copies the origin and direction properties of ray into this ray.

```
.distanceSqToPoint( point )
```

| point - the Vector3 to compute a distance to.

Get the squared distance of the closest approach between the Ray and the Vector3.

```
.distanceSqToSegment( v0, v1, optionalPointOnRay, optionalPointOn
```

| v0 - начальная точка отрезка линии.

| v1 - конечная точка отрезка линии.

| optionalPointOnRay - если этот параметр представлен, он принимается как точка данного луча ([Ray](#)), ближайшая к

данному отрезку линии. Дополнительный, необязательный параметр.

[optionalPointOnSegment](#) - если этот параметр представлен, он принимается как точка данного отрезка линии, ближайшая к данному лучу ([Ray](#)). Дополнительный, необязательный параметр.

Метод получает квадрат расстояния между данным лучом ([Ray](#)) и отрезком линии.

```
.distanceToPlane( plane )
```

| plane - the Plane to get the distance to.

Get the distance from origin to the Plane, or null if the Ray doesn't intersect the Plane.

```
.distanceToPoint( point )
```

| point - Vector3 The Vector3 to compute a distance to.

Get the distance of the closest approach between the Ray and the point.

```
.equals( ray )
```

| ray - the Ray to compare to.

Returns true if this and the other ray have equal offset and direction.

```
.intersectBox( box, optionalTarget )
```

| box - the Box3 to intersect with.

[optionalTarget](#) - дополнительный, необязательный аргумент.

Если указан, то результат метода будет скопирован в предоставленный [Vector3](#), в противном случае будет создан новый [Vector3](#).

Intersect this Ray with a Box3, returning the intersection point or null if there is no intersection.

```
.intersectPlane( plane, optionalTarget )
```

| plane - the Plane to intersect with.

[optionalTarget](#) - дополнительный, необязательный аргумент.

Если указан, то результат метода будет скопирован в предоставленный [Vector3](#), в противном случае будет

| создан новый [Vector3](#).

Intersect this Ray with a Plane, returning the intersection point or null if there is no intersection.

`.intersectSphere(sphere, optionalTarget)`

| sphere - the Sphere to intersect with.

[optionalTarget](#) - дополнительный, необязательный аргумент.

| Если указан, то результат метода будет скопирован в предоставленный [Vector3](#), в противном случае будет создан новый [Vector3](#).

Intersect this Ray with a Sphere, returning the intersection point or null if there is no intersection.

`.intersectTriangle(a, b, c, backfaceCulling, optionalTarget)`

| a, b, c - The Vector3 points making up the triangle.

| backfaceCulling - whether to use backface culling.

[optionalTarget](#) - дополнительный, необязательный аргумент.

| Если указан, то результат метода будет скопирован в предоставленный [Vector3](#), в противном случае будет создан новый [Vector3](#).

Intersect this Ray with a triangle, returning the intersection point or null if there is no intersection.

`.intersectsBox(box)`

| box - the Box3 to intersect with.

Return true if this Ray intersects with the Box3.

`.intersectsPlane(plane)`

| plane - the Plane to intersect with.

Return true if this Ray intersects with the Plane.

`.intersectsSphere(sphere)`

| sphere - the Sphere to intersect with.

Return true if this Ray intersects with the Sphere.

`.lookAt(v)`

| `v` - The Vector3 to look at.
Adjusts the direction of the ray to point at the vector in world coordinates.

`.recast(t)`

| `t` - The distance along the Ray to interpolate.
Shift the origin of this Ray along its direction by the distance given.

`.set(origin, direction)`

| `origin` - the origin of the Ray.
`direction` - the direction of the Ray. This must be normalized (with `Vector3.normalize`) for the methods to operate properly.
Метод копирует параметры свойств, переданных в аргументах `origin` и `direction`, в данный луч (Ray). Copy the parameters to the `origin` and `direction` properties of this ray.

Исходники

[Ray.js в этом справочнике](#)

[Ray.js на Гитхабе](#)

Sphere

Сфера, определяемая центром и радиусом.

Конструктор

`Sphere(center, radius)`

| `center` -- центр сферы. Значением по умолчанию является `Vector3` с координатами (0, 0, 0).
`radius` -- радиус сферы. Значение по умолчанию равно 0.

Создает новую сферу.

Свойства

[center](#)

Трехмерный вектор, определяющий центр сферы. Значением по умолчанию является (0, 0, 0).

[.radius](#)

Радиус сферы. Значение по умолчанию равно 0.

Методы

```
[method:Sphere applyMatrix4( [page:Matrix4 matrix) [page:Sphere  
| matrix -- [page:Matrix4
```

Transforms this sphere with the provided [page:Matrix4].

```
[method:Vector3 clampPoint( [page:Vector3 point, [page:Vector3 op  
| point -- [page:Vector3 The point to clamp
```

```
| optionalTarget -- [page:Vector3 The optional target point to return  
Clamps a point within the sphere. If the point is outside the sphere,  
it will clamp it to the closest point on the edge of the sphere.
```

```
[method:Sphere clone()
```

Provides a new copy of the sphere.

```
.equals( [page:Sphere sphere) )
```

```
| sphere -- [page:Sphere
```

Checks to see if the two spheres' centers and radii are equal.

```
[method:Float distanceToPoint( [page:Vector3 point) )
```

```
| point -- [page:Vector3
```

Returns the closest distance from the boundary of the sphere to the point. If the sphere contains the point, the distance will be negative.

```
[method:Box3 getBoundingBox( [page:Box optionalTarget) )
```

```
| optionalTarget -- [page:Box
```

Returns a bounding box for the sphere, optionally setting a provided box target.

```
[method:Boolean containsPoint( [page:Vector3 point )
```

```
| point -- [page:Vector3
```

Checks to see if the sphere contains the provided point inclusive of the edge of the sphere.

```
[method:Sphere copy( [page:Sphere sphere )
```

```
| sphere -- [page:Sphere to copy
```

Copies the values of the passed sphere to this sphere.

```
[method:Boolean intersectsSphere( [page:Sphere sphere )
```

```
| sphere -- [page:Sphere
```

Checks to see if two spheres intersect.

```
.empty( )
```

Checks to see if the sphere is empty (the radius set to 0).

```
.set( center, radius )
```

```
| center -- центр сферы.
```

```
| radius -- радиус сферы.
```

Метод устанавливает свойства center и radius для этой сферы.

```
.setFromPoints( points, optionalCenter )
```

```
| points -- массив позиций трехмерных векторов (Vector3).
```

```
| optionalCenter -- дополнительная позиция (Vector3) центра сферы.
```

Метод вычисляет минимальную ограничивающую сферу для массива точек (переданного в параметре points). Если параметр optionalCenter задан, то он будет использоваться в качестве центра сферы. В противном случае, вычисляется центр ограничивающей рамки, выровненной по осям и охватывающей весь массив точек (points).

```
.translate( offset )
```

Метод переносит центр сферы на представленный вектор смещения (параметр offset).

Исходники

[Sphere.js в этом справочнике](#)
[Sphere.js на Гитхабе](#)

Spherical

Координаты точки в [сферической системе координат](#).
[Сферическая система координат](#) на русском языке.

Конструктор

`Spherical(radius, phi, theta)`

radius - радиус или the radius, or the

[link:https://en.wikipedia.org/wiki/Euclidean_distance] Euclidean distance (straight-line distance) from the point to the origin.

Значение по умолчанию равно 1.0.

phi - полярный угол от оси у (верх). Значение по умолчанию равно 0.

Примечание переводчика: На русском языке принято называть этот угол зенитным.

theta - экваториальный угол вокруг оси у (верх). Значение по умолчанию равно 0.

Примечание переводчика: На русском языке принято называть этот угол азимутальным.

Полюсы (phi) находятся на положительной и отрицательной сторонах оси у. Экватор (theta) начинается с положительной стороны оси z.

Свойства

`.radius`

`.phi`

.theta

Методы

[method:Spherical .clone()

Returns a new plane with the same [page:.radius radius, [page:.phi phi and [page:.theta theta properties as this one.

[method:Spherical .copy([page:Spherical s)

Copies the values of the passed Spherical's [page:.radius radius, [page:.phi phi and [page:.theta theta properties to this spherical.

[method:Spherical .makeSafe()

Restricts the polar angle page:.phi phi to be between 0.000001 and pi - 0.000001.

[method:Spherical .set([page:Float radius, [page:Float phi, [pag

Sets values of this spherical's [page:.radius radius, [page:.phi phi and [page:.theta theta properties.

[method:Spherical .setFromVector3([page:Vector3 vec3)

Sets values of this spherical's [page:.radius radius, [page:.phi phi and [page:.theta theta properties from the [page:Vector3 Vector3.

The [page:.radius radius is set the vector's [page:Vector3.length, while the [page:.phi phi and [page:.theta theta properties are set from its direction.

Исходники

[Spherical.js в этом справочнике](#)

[Spherical.js на Гитхабе](#)

Triangle

Треугольник - геометрическая фигура, определяемая тремя трехмерными векторами ([Vector3](#)), представляющими три его угла.

Конструктор

`Triangle(a, b, c)`

- `a` - первый угол треугольника. Значением по умолчанию является трехмерный вектор $(0, 0, 0)$.
- `b` - второй угол треугольника. Значением по умолчанию является трехмерный вектор $(0, 0, 0)$.
- `c` - последний угол треугольника. Значением по умолчанию является трехмерный вектор $(0, 0, 0)$.

Создает новый Triangle.

Свойства

`.a`

Первый угол треугольника. Значением по умолчанию является трехмерный вектор $(0, 0, 0)$.

`.b`

Второй угол треугольника. Значением по умолчанию является трехмерный вектор $(0, 0, 0)$.

`.c`

Последний угол треугольника. Значением по умолчанию является трехмерный вектор $(0, 0, 0)$.

Методы

`.area()`

Метод возвращает площадь треугольника.

```
.barycoordFromPoint( point, optionalTarget )
```

`point` - трехмерный вектор ([Vector3](#)).

`optionalTarget` - если этот параметр указан, результат будет скопирован в заданный трехмерный вектор ([Vector3](#)), в противном случае будет создан новый трехмерный вектор. Дополнительный, необязательный параметр.

Метод возвращает [барицентрические координаты](#) ([barycentric coordinate](#)) из данного вектора.

[Изображение барицентрических координат.](#)

```
.clone( )
```

Метод возвращает новый треугольник с точно такими же свойствами `a`, `b` и `c` как и у данного.

```
.closestPointToPoint( point, optionalTarget )
```

`point` - Vector3

`optionalTarget` - если этот параметр указан, результат будет скопирован в заданный трехмерный вектор ([Vector3](#)), в противном случае будет создан новый трехмерный вектор. Дополнительный, необязательный параметр.

Returns the closest point on the triangle to point.

```
.containsPoint( point )
```

`point` - Vector3 to check. Returns true if the passed vector is within the triangle.

```
.copy( triangle )
```

Copies the values of the passed triangle's `a`, `b` and `c` properties to this triangle.

```
.equals( triangle )
```

Returns true if the two triangles have identical `a`, `b` and `c` properties.

```
.midpoint( optionalTarget )
```

`optionalTarget` - если этот параметр указан, результат будет скопирован в заданный трехмерный вектор ([Vector3](#)), в

противном случае будет создан новый трехмерный вектор.
Дополнительный, необязательный параметр.
Вычисляет [серединный треугольник](#) (англоязычный вариант - [midpoint triangle](#)) данного треугольника.

`.normal(optionalTarget)`

[optionalTarget](#) - если этот параметр указан, результат будет скопирован в заданный трехмерный вектор ([Vector3](#)), в противном случае будет создан новый трехмерный вектор.
Дополнительный, необязательный параметр.

Метод вычисляет [вектор нормали](#) () треугольника.

`.plane(optionalTarget)`

[optionalTarget](#) - если этот параметр указан, результат будет скопирован в заданную плоскость ([Plane](#)), в противном случае будет создана новая [Plane](#). Дополнительный, необязательный параметр.

Метод создает [плоскость](#) на основе данного треугольника.

`.set(a, b, c)`

Устанавливает свойства [a](#), [b](#) и [c](#) данного треугольника по переданным [трехмерным векторам](#) (аргументы `a`, `b` и `c`).

`.setFromPointsAndIndices(points, i0, i1, i2)`

[points](#) - массив трехмерных векторов ([Vector3](#)).
[i0](#) - целочисленный индекс.
[i1](#) - целочисленный индекс.
[i2](#) - целочисленный индекс.

Метод устанавливает векторы треугольника по векторам предоставленного массива (аргумент [points](#)).

Исходники

[Triangle.js в этом справочнике](#)

[Triangle.js на Гитхабе](#)

Vector2

Класс, представляющий [двумерный \(2D\) вектор](#). (Вот [статья Википедии о векторном пространстве](#) на русском языке.) Вектор 2D - это упорядоченная пара чисел (помеченных как x и y), которая может использоваться для представления ряда вещей, таких как:

- Точка в двумерном пространстве (т.е. позиция на плоскости).
- Направление и расстояние по плоскости. В [three.js](#) расстояние всегда будет [Эвклидовым расстоянием](#) (т.е. расстоянием по прямой линии) от точки начала координат $(0, 0)$ до координат заданной точки (x, y) , направление также измеряется от $(0, 0)$ к (x, y) .
- Любая произвольная упорядоченная пара чисел.

Имеются и другие вещи, которые можно представить как 2D-вектор, однако это его наиболее распространенное использование в [three.js](#).

Пример

```
var a = new THREE.Vector2( 0, 1 );  
  
// no arguments; will be initialised to (0, 0)  
// без аргументов вектор будет инициализирован как (0, 0)  
var b = new THREE.Vector2( );  
  
var d = a.distanceTo( b );
```

Конструктор

```
Vector2( x, y )
```

| [x](#) - значение x вектора, по умолчанию равно 0.

| [y](#) - значение y вектора, по умолчанию равно 0.

Создает новый Vector2.

Свойства

.isVector2

Свойство используется для проверки, является ли данный класс и производные от него классы двумерными векторами (Vector2). Значением по умолчанию является true.

Не изменяйте это свойство, так как оно используется для оптимизации внутри *three.js*.

.height

Псевдоним (алиас) для y.

.width

Псевдоним (алиас) для x.

x

y

Методы

.add(v)

Метод добавляет к данному вектору вектор v.

.addScalar(s)

Метод добавляет скалярное значение s к значениям данного вектора x и y.

.addScaledVector(v, s)

Добавляет к данному вектору множество из v и s.

.addVectors(a, b)

Устанавливает данный вектор как a + b.

.angle()

Метод вычисляет угол (в радианах) данного вектора

относительно положительной оси x.

`.applyMatrix3(m)`

Multiplies this vector (with an implicit 1 as the 3rd component) by m.

`.ceil()`

Компоненты x и у данного вектора округляются вверх до ближайшего целого значения.

`.clamp(min, max)`

`min` - минимальные значения x и y.

`max` - максимальные значения x и y.

Если значение x или y данного вектора больше значения x или y вектора, указанного в параметре max, оно заменяется этим соответствующим значением.

Если значение x или y данного вектора меньше значения x или y вектора, указанного в параметре min, оно заменяется этим соответствующим значением.

`.clampLength(min, max)`

`min` - минимальное значение длины, на котором она будет зафиксирована (зажата).

`max` - максимальное значение длины, на котором она будет зафиксирована (зажата).

Если длина данного вектора больше значения, указанного в параметре max, она заменяется на значение max.

Если длина данного вектора меньше значения, указанного в параметре min, она заменяется на значение min.

`.clampScalar(min, max)`

`min` - минимальное значение компонентов, на котором они будут зафиксированы (зажаты).

`max` - максимальное значение компонентов, на котором они будут зафиксированы (зажаты).

Если значение x или y данного вектора больше значения, указанного в параметре max, оно заменяется на значение max.

Если значение x или y данного вектора меньше значения, указанного в параметре min , оно заменяется на значение min .

`.clone()`

Возвращает новый Vector2 с такими же значениями x и y , что и у данного вектора.

`.copy(v)`

Копирует значения свойств x и y из вектора ([Vector2](#)), переданного в параметре v , в данный вектор ([Vector2](#)).

`.distanceTo (v)`

Computes the distance from this vector to v .

`.manhattanDistanceTo (v)`

Computes the Manhattan distance from this vector to v .

`.distanceToSquared (v)`

Computes the squared distance from this vector to v . If you are just comparing the distance with another distance, you should compare the distance squared instead as it is slightly more efficient to calculate.

`.divide (v)`

Divides this vector by v .

`.divideScalar (s)`

Divides this vector by scalar s . Sets vector to $(0, 0)$ if $s = 0$.

`.dot (v)`

Calculates the dot product of this vector and v .

`.equals (v)`

Checks for strict equality of this vector and v .

`.floor ()`

The components of the vector are rounded down to the nearest integer value.

```
.fromArray ( array, offset )
```

array - the source array.

offset - (optional) offset into the array. Default is 0.

Sets this vector's x value to be array[offset] and y value to be array[offset + 1].

```
.fromBufferAttribute ( attribute, index )
```

attribute - the source attribute.

index - index in the attribute.

Sets this vector's x and y values from the attribute.

```
.getComponent ( index )
```

index - 0 or 1.

If index equals 0 returns the x value. If index equals 1 returns the y value.

```
.length ()
```

Computes the Euclidean length (straight-line length) from (0, 0) to (x, y).

```
.manhattanLength ()
```

Computes the Manhattan length of this vector.

```
.lengthSq ()
```

Computes the square of the Euclidean length (straight-line length) from (0, 0) to (x, y). If you are comparing the lengths of vectors, you should compare the length squared instead as it is slightly more efficient to calculate.

```
.lerp ( v, alpha )
```

v - Vector2 to interpolate towards.

alpha - interpolation factor in the closed interval [0, 1].

Linearly interpolates between this vector and v, where alpha is the

distance along the line - alpha = 0 will be this vector, and alpha = 1 will be v.

```
.lerpVectors ( v1, v2, alpha )
```

v1 - the starting Vector2.

v2 - Vector2 to interpolate towards.

alpha - interpolation factor in the closed interval [0, 1]. Sets this vector to be the vector linearly interpolated between v1 and v2 where alpha is the distance along the line connecting the two vectors - alpha = 0 will be v1, and alpha = 1 will be v2.

```
.negate ()
```

Inverts this vector - i.e. sets x = -x and y = -y.

```
.normalize ()
```

Converts this vector to a unit vector - that is, sets it equal to the vector with the same direction as this one, but length 1.

```
.max ( v )
```

If this vector's x or y value is less than v's x or y value, replace that value with the corresponding max value.

```
.min ( v )
```

If this vector's x or y value is greater than v's x or y value, replace that value with the corresponding min value.

```
.multiply ( v )
```

Multiplies this vector by v.

```
.multiplyScalar ( s )
```

Multiplies this vector by scalar s.

```
.rotateAround ( center, angle )
```

center - the point around which to rotate.

angle - the angle to rotate, in radians.

Rotates the vector around center by angle radians.

```
.round()
```

The components of the vector are rounded to the nearest integer value.

```
.roundToZero()
```

The components of the vector are rounded towards zero (up if negative, down if positive) to an integer value.

```
.set( x, y )
```

Sets the x and y components of this vector.

```
.setComponent( index, value )
```

index - 0 or 1.

value - Float

If index equals 0 set x to value. If index equals 1 set y to value

```
.setLength( l )
```

Sets this vector to the vector with the same direction as this one, but length l.

```
.setScalar( scalar )
```

Sets the x and y values of this vector both equal to scalar.

```
.setX( x )
```

Replaces this vector's x value with x.

```
.setY( y )
```

Replaces this vector's y value with y.

```
.sub( v )
```

Метод вычитает из данного вектора вектор v.

```
.subScalar( s )
```

Вычитает скаляр s из компонентов x и y данного вектора.

```
.subVectors( a, b )
```

Устанавливает данный вектор как `a - b`.

`.toArray(array, offset)`

`array` - дополнительный, необязательный массив для хранения вектора. Если он не предоставлен, будет создан новый массив.

`offset` - дополнительное, необязательное смещение в массиве.

Возвращает массив в виде [x, y] или копирует x и y в предоставленный (`array`) массив.

Исходники

[Vector2.js в этом справочнике](#)

[Vector2.js на Гитхабе](#)

Vector3

Класс, представляющий [трехмерный \(3D\) вектор](#). (Вот [статья Википедии о векторном пространстве](#) на русском языке.) Вектор 3D - это упорядоченная тройка (триплет) чисел (помеченных как x, y и z), которая может использоваться для представления ряда вещей, таких как:

- Точка в трехмерном пространстве.
- Направление и расстояние в трехмерном пространстве. В `three.js` расстояние всегда будет [Эвклидовым расстоянием](#) (т.е. расстоянием по прямой линии) от точки начала координат (0, 0, 0) до координат заданной точки (x, y, z), направление также измеряется от (0, 0, 0) к (x, y, z).
- Любая произвольная упорядоченная тройка (триплет) чисел.

Имеются и другие вещи, которые можно представить как 3D-вектор (такие как векторы импульса и т. д.), однако это его наиболее распространенное использование в `three.js`.

Пример

```
var a = new THREE.Vector3( 0, 1, 0 );  
  
// no arguments; will be initialised to (0, 0, 0)  
// без аргументов вектор будет инициализирован как (0, 0, 0)  
var b = new THREE.Vector3( );  
  
var d = a.distanceTo( b );
```

Конструктор

`Vector3 (x, y, z)`

x - значение x вектора, по умолчанию равно 0.

y - значение y вектора, по умолчанию равно 0.

z - значение z вектора, по умолчанию равно 0.

Создает новый трехмерный вектор Vector3.

Свойства

[.isVector3](#)

Свойство используется для проверки, является ли данный класс и производные от него классы трехмерными векторами (Vector3). Значением по умолчанию является `true`.

Не изменяйте это свойство, так как оно используется для оптимизации внутри ***three.js***.

[.x](#)

[.y](#)

[.z](#)

Методы

[.add\(v \)](#)

Метод добавляет к данному вектору вектор `v`.

`.addScalar(s)`

Метод добавляет скалярное значение `s` к значениям данного вектора `x`, `y` и `z`.

`.addScaledVector(v, s)`

Добавляет к данному вектору множество из `v` и `s`.

`.addVectors(a, b)`

Устанавливает данный вектор как `a` + `b`.

`.angleTo(v)`

Метод возвращает угол в радианах между данным вектором и вектором, указанным в параметре `v`.

`.applyAxisAngle(axis, angle)`

`axis` - нормализованный вектор `Vector3`.

`angle` - угол в радианах.

Метод применяет к данному вектору поворот, указанный осями (параметр `axis`) и углом (параметр `angle`).

`.applyEuler(euler)`

Applies euler transform to this vector by converting the Euler object to a Quaternion and applying.

`.applyMatrix3(m)`

Метод умножает данный вектор на матрицу, указанную в параметре `m`.

`.applyMatrix4(m)`

Перемножает данный вектор (с подразумеваемой 1 в четвертом измерении) с матрицей `m` и делит на перспективу. Multiplies this vector (with an implicit 1 in the 4th dimension) and `m`, and divides by perspective.

```
.applyQuaternion( quaternion )
```

Applies a Quaternion transform to this vector.

```
.ceil( )
```

Компоненты вектора x, y и z округляются вверх до ближайшего целочисленного значения.

```
.clamp( min, max )
```

min - минимальные значения x, y и z в желаемом диапазоне.

max - максимальные значения x, y и z в желаемом диапазоне.

Если значение x, y или z данного вектора больше значения x, y или z вектора, указанного в параметре max, оно заменяется этим соответствующим значением.

Если значение x, y или z данного вектора меньше значения x, y или z вектора, указанного в параметре min, оно заменяется этим соответствующим значением.

```
.clampLength( min, max )
```

min - минимальное значение длины, на котором она будет зафиксирована (зажата).

max - максимальное значение длины, на котором она будет зафиксирована (зажата).

Если длина данного вектора больше значения, указанного в параметре max, она заменяется на значение max.

Если длина данного вектора меньше значения, указанного в параметре min, она заменяется на значение min.

```
.clampScalar( min, max )
```

min - минимальное значение компонентов, на котором они будут зафиксированы (зажаты).

max - максимальное значение компонентов, на котором они будут зафиксированы (зажаты).

Если значение x, y или z данного вектора больше значения, указанного в параметре max, оно заменяется на значение max.

Если значение x, y или z данного вектора меньше значения, указанного в параметре min, оно заменяется на значение min.

`.clone()`

Возвращает новый Vector3 с такими же значениями `x`, `y` и `z`, что и у данного вектора.

`.copy(v)`

Копирует значения свойств `x`, `y` и `z` из вектора, переданного в параметре `v`, в данный трехмерный вектор.

`.cross(v)`

Метод устанавливает данный вектор как [векторное произведение](#) (в англоязычном варианте это [cross product](#)) самого себя и вектора, указанного в параметре `v`.

`.crossVectors(a, b)`

Устанавливает данный вектор как [векторное произведение](#) (в англоязычном варианте это [cross product](#)) векторов, указанных в параметрах `a` и `b`.

`.distanceTo(v)`

Метод вычисляет расстояние от данного вектора до вектора, указанного в параметре `v`.

`.distanceToManhattan(v)`

Метод вычисляет [манхэттенское расстояние](#) ([Manhattan length](#)) от данного вектора до вектора, указанного в параметре `v`.

`.distanceToSquared(v)`

Вычисляет квадрат расстояния от данного вектора до вектора, указанного в параметре `v`. При простом сравнении одного расстояния с другим взамен следует сравнивать именно квадраты расстояний, так как это вычисление несколько более эффективно.

Примечание переводчика: Вычисление расстояния методом `distanceTo` это вычисление квадратного корня из результата, полученного методом `distanceToSquared`.

`.divide(v)`

Метод делит данный вектор на вектор, указанный в параметре `v`.

`.divideScalar(s)`

Делит данный вектор на скалярное значение `s`. Если значение скаляра равно нулю (`s = 0`), вектор устанавливается как $(0, 0)$.

`.dot(v)`

Метод вычисляет скалярное произведение (dot product) данного вектора и вектора, переданного в параметре `v`.

`.equals(v)`

Метод проверяет точное равенство данного вектора и вектора, переданного в параметре `v`.

`.floor()`

Этим методом компоненты вектора округляются вниз, до ближайшего целочисленного значения.

`.fromArray (array, offset)`

array - исходный массив в виде $[x, y, z, w]$.
offset - смещение массива (дополнительный, необязательный параметр). Значение по умолчанию равно 0.

Метод устанавливает для данного вектора значение `x` равное `array[offset + 0]`, значение `y` равное `array[offset + 1]` и значение `z` равное `array[offset + 2]`.

`.fromBufferAttribute(attribute, index)`

attribute - исходный атрибут.
index - индекс в атрибуте.

Устанавливает значения `x`, `y` и `z` данного вектора из атрибута, указанного в параметре `attribute`.

`.getComponent(index)`

| [index](#) -- целочисленное значение, может быть 0, 1 или 2.

Если индекс равен 0, возвращается значение [x](#).

Если индекс равен 1, возвращается значение [y](#).

Если индекс равен 2, возвращается значение [z](#).

[.length\(\)](#)

Вычисляет [евклидову длину \(Euclidean length\)](#), т.е. расстояние по прямой линии, от точки начала координат (0, 0, 0) до точки с заданными координатами (x, y, z).

[.lengthManhattan\(\)](#)

Метод вычисляет [манхэттенское расстояние \(Manhattan length\)](#) данного вектора.

[.lengthSq\(\)](#)

Метод вычисляет квадрат [евклидовой длины \(Euclidean length\)](#), т.е. квадрат расстояния по прямой линии, от точки начала координат (0, 0, 0) до точки с заданными координатами (x, y, z). При сравнении длин векторов, замен сравнения длин, следует сравнивать квадраты длины, так как для расчета это несколько более эффективно.

Примечание переводчика: Вычисление длины вектора методом [length](#) это вычисление квадратного корня из результата, полученного методом [lengthSq](#).

[.lerp\(v, alpha \)](#)

[v](#) - [трехмерный вектор \(Vector3\)](#), до которого будет проводится операция интерполяции.

[alpha](#) - коэффициент интерполяции в закрытом интервале [0, 1].

Метод выполняет операцию линейного интерполяирования между данным вектором и вектором, переданным в параметре [v](#) с коэффициентом интерполяции [alpha](#), причем [alpha](#) можно рассматривать как расстояние вдоль линии соединения этих двух векторов - при [alpha](#) = 0, он будет равен данному вектору, а при [alpha](#) = 1 - вектору [v](#).

.lerpVectors(v1, v2, alpha)

v1 - начальный трехмерный вектор (Vector3).

v2 - трехмерный вектор (Vector3), до которого будет проводится операция интерполяции.

alpha - коэффициент интерполяции в закрытом интервале [0, 1].

Метод устанавливает данный вектор как линейно интерполированный вектор между векторами v1 и v2, и где alpha - это расстояние вдоль линии соединения этих двух векторов - при alpha = 0, он будет равен вектору v1, при alpha = 1 - вектору v2.

.max(v)

Если значение x, y или z данного вектора меньше значения x, y или z вектора, переданного в параметре v, то это значение заменяется соответствующим значением вектора v.

.min(v)

Если значение x, y или z данного вектора больше значения x, y или z вектора, переданного в параметре v, то это значение заменяется соответствующим значением вектора v.

.multiply(v)

Метод умножает данный вектор на вектор, указанный в параметре v.

.multiplyScalar(s)

Метод умножает данный вектор на скаляр, переданный в параметре s.

.multiplyVectors(a, b)

Устанавливает данный вектор равным a×b.

.negate()

Метод инвертирует данный вектор, т.е. устанавливает x = -x, y = -y и z = -z.

.normalize()

Метод преобразовывает данный вектор в [единичный вектор \(unit vector\)](#) - то есть, направление его остается тем же самым, а длина вектора ([length](#)) устанавливается равной 1.

.project(camera)

| [camera](#) — камера, используемая в проекции.

Метод [проецирует](#) вектор на камеру. Projects the vector with the camera.

.projectOnPlane(planeNormal)

| [planeNormal](#) - A vector representing a plane normal.

Метод [проецирует](#) данный вектор на плоскость путем вычитания из данного вектора проекции данного вектора на нормаль к этой плоскости.

.projectOnVector(Vector3)

Метод [проецирует](#) данный вектор на другой вектор (параметр [Vector3](#)).

.reflect(normal)

| [normal](#) - нормаль (перпендикуляр) к плоскости отражения.

Метод отражает вектор от плоскости, перпендикулярной (ортогональной) к нормали (параметр [normal](#)). Предполагается что нормаль имеет единичную длину.

.round()

Метод округляет компоненты вектора до ближайшего целочисленного значения.

.roundToZero()

Метод округляет компоненты вектора к нулю (если значение отрицательно - происходит округление вверх, если положительное - вниз) до целочисленного значения.

```
.set( x, y, z )
```

Устанавливает компоненты x, y и z данного вектора.

```
.setComponent( index, value )
```

index -- целочисленное значение, может быть 0, 1 или 2.

value -- число с плавающей запятой.

Если индекс равен 0, устанавливается значение x.

Если индекс равен 1, устанавливается значение y.

Если индекс равен 2, устанавливается значение z.

```
.setFromCylindrical( c )
```

Sets this vector from the cylindrical coordinates c.

```
.setFromMatrixColumn( matrix, index )
```

Sets this vector's x, y and z equal to the column of the matrix specified by the index.

```
.setFromMatrixPosition( m )
```

Sets this vector to the position elements of the transformation matrix m.

```
.setFromMatrixScale( m )
```

Sets this vector to the scale elements of the transformation matrix m.

```
.setFromSpherical( s )
```

Sets this vector from the spherical coordinates s.

```
.setLength( l )
```

Метод устанавливает данный вектор как вектор с тем же самым направлением, но длиной (length) равной l.

```
.setScalar( scalar )
```

Устанавливает все значения x, y и z данного вектора равными скалярному значению, указанному в параметре scalar.

```
.setX( x )
```

Метод заменяет значение x данного вектора на значение, переданное в параметре x.

.setY(y)

Метод заменяет значение y данного вектора на значение, переданное в параметре y.

.setZ(z)

Метод заменяет значение z данного вектора на значение, переданное в параметре z.

.sub (v)

Метод вычитает из данного вектора вектор, указанный в параметре v.

.subScalar (s)

Метод вычитает из компонентов x, y и z данного вектора скалярное значение, указанное в параметре s.

.subVectors(a, b)

Метод устанавливает данный вектор как разницу векторов a - b.

.toArray(array, offset)

array - дополнительный, необязательный массив для хранения вектора. Если он не предоставлен, создается новый массив.

offset - дополнительное, необязательное смещение в массиве.

Возвращает массив в виде [x, y, z], или копирует x, y и z в предоставленный (array) массив.

.transformDirection (m)

Метод изменяет данный вектор на матрицу (набор 3×3 вверху слева в матрице, переданной в параметре m) и затем нормализует результат.

`.unproject (camera)`

| `camera` — камера, используемая в проекции.

Отменяет [проецирование](#) вектора на матрицу проекции камеры.
Unprojects the vector with the camera's projection matrix.

Исходники

[Vector3.js в этом справочнике](#)

[Vector3.js на Гитхабе](#)

Vector4

Класс, представляющий [четырехмерный \(4D\) вектор](#). (Вот [статья Википедии о векторном пространстве](#) на русском языке.) Вектор 4D - это упорядоченная четверка (квадруплет) чисел (помеченных как `x`, `y`, `z` и `w`), которая может использоваться для представления ряда вещей, таких как:

- Точка в четырехмерном пространстве.
- Направление и расстояние в четырехмерном пространстве. В `three.js` расстояние всегда будет [Эвклидовым расстоянием](#) (т.е. расстоянием по прямой линии) от точки начала координат $(0, 0, 0, 0)$ до координат заданной точки (x, y, z, w) , направление также измеряется от $(0, 0, 0, 0)$ к (x, y, z, w) .
- Любая произвольная упорядоченная четверка (квадруплет) чисел.

Имеются и другие вещи, которые можно представить как 4D-вектор, однако это его наиболее распространенное использование в `three.js`.

Пример

```
var a = new THREE.Vector4( 0, 1, 0, 0 );
// no arguments; will be initialised to (0, 0, 0, 1)
// без аргументов вектор будет инициализирован как (0, 0, 0, 1)
var b = new THREE.Vector4( );
```

```
var d = a.dot( b );
```

Конструктор

```
Vector4 ( x, y, z, w )
```

- | x - значение x вектора, по умолчанию равно 0.
- | y - значение y вектора, по умолчанию равно 0.
- | z - значение z вектора, по умолчанию равно 0.
- | w - значение w вектора, по умолчанию равно 1.

Создает новый Vector4.

Свойства

.isVector4

Свойство используется для проверки, является ли данный класс и производные от него классы четырехмерными векторами (Vector4). Значением по умолчанию является true.

Не изменяйте это свойство, так как оно используется для оптимизации внутри **three.js**.

.x

.y

.z

.w

Методы

.add(v)

Метод добавляет к данному вектору вектор v.

.addScalar(s)

Метод добавляет скалярное значение s к значениям данного вектора x, y, z и w.

.addScaledVector(v, s)

Добавляет к данному вектору множество из v и s.

.addVectors(a, b)

Устанавливает данный вектор как a + b.

.applyMatrix4(m)

Метод умножает данный вектор с матрицей 4×4, заданной в аргументе m.

.ceil()

Компоненты вектора x, y, z и w округляются вверх до ближайшего целочисленного значения.

.clamp(min, max)

min - минимальные значения x, y, z и w в желаемом диапазоне.

max - максимальные значения x, y, z и w в желаемом диапазоне.

Если значение x, y, z или w данного вектора больше значения x, y, z или w вектора, указанного в параметре max, оно заменяется этим соответствующим значением.

Если значение x, y, z или w данного вектора меньше значения x, y, z или w вектора, указанного в параметре min, оно заменяется этим соответствующим значением.

.clampLength(min, max)

min - минимальное значение длины, на котором она будет зафиксирована (зажата).

max - максимальное значение длины, на котором она будет зафиксирована (зажата).

Если длина данного вектора больше значения, указанного в параметре `max`, она заменяется на значение `max`.

Если длина данного вектора меньше значения, указанного в параметре `min`, она заменяется на значение `min`.

`.clampScalar(min, max)`

`min` - минимальное значение компонентов, на котором они будут зафиксированы (зажаты).

`max` - максимальное значение компонентов, на котором они будут зафиксированы (зажаты).

Если значение `x`, `y`, `z` или `w` данного вектора больше значения, указанного в параметре `max`, оно заменяется на значение `max`.

Если значение `x`, `y`, `z` или `w` данного вектора меньше значения, указанного в параметре `min`, оно заменяется на значение `min`.

`.clone()`

Возвращает новый `Vector4` с такими же значениями `x`, `y`, `z` и `w`, что и у данного вектора.

`.copy(v)`

Копирует значения свойств `x`, `y`, `z` и `w` из вектора, переданного в параметре `v`, в данный четырехмерный вектор.

`.divideScalar(s)`

Делит данный вектор на скалярное значение `s`. Если значение скаляра равно нулю (`s = 0`), вектор устанавливается как `(0, 0)`.

`.dot(v)`

Метод вычисляет [скалярное произведение \(dot product\)](#) данного вектора и вектора, переданного в параметре `v`.

`.equals(v)`

Метод проверяет точное равенство данного вектора и вектора, переданного в параметре `v`.

`.floor()`

Этим методом компоненты вектора округляются вниз, до ближайшего целочисленного значения.

`.fromArray(array, offset)`

`array` - исходный массив в виде [x, y, z, w].
`offset` - смещение массива (дополнительный, необязательный параметр). Значение по умолчанию равно 0.

Метод устанавливает для данного вектора значение `x` равное `array[offset + 0]`, значение `y` равное `array[offset + 1]`, значение `z` равное `array[offset + 2]` и значение `w` равное `array[offset + 3]`.

`.fromBufferAttribute(attribute, index)`

`attribute` - исходный атрибут.
`index` - индекс в атрибуте.

Устанавливает значения `x`, `y`, `z` и `w` данного вектора из атрибута, указанного в параметре `attribute`.

`.getComponent(index)`

`index` -- целочисленное значение, может быть 0, 1, 2, или 3.

Если индекс равен 0, возвращается значение `x`.

Если индекс равен 1, возвращается значение `y`.

Если индекс равен 2, возвращается значение `z`.

Если индекс равен 3, возвращается значение `w`.

`.length()`

Вычисляет [евклидову длину \(Euclidean length\)](#), т.е. расстояние по прямой линии, от точки начала координат (0, 0, 0, 0) до точки с заданными координатами (x, y, z, w).

`.lengthManhattan()`

Метод вычисляет [манхэттенское расстояние \(Manhattan length\)](#) данного вектора.

`.lengthSq()`

Метод вычисляет квадрат [евклидовой длины \(Euclidean length\)](#), т.е. квадрат расстояния по прямой линии, от точки начала координат (0, 0, 0, 0) до точки с заданными координатами (x, y, z, w). При сравнении длин векторов, замен сравнения длин, следует сравнивать квадраты длины, так как для расчета это несколько более эффективно.

Примечание переводчика: Вычисление длины вектора методом [length](#) это вычисление квадратного корня из результата, полученного методом [lengthSq](#).

[.lerp\(v, alpha \)](#)

[v](#) - [четырехмерный вектор \(Vector4\)](#), до которого будет проводится операция интерполяции.

[alpha](#) - коэффициент интерполяции в закрытом интервале [0, 1].

Метод выполняет операцию линейного интерполяции между данным вектором и вектором, переданным в параметре [v](#) с коэффициентом интерполяции [alpha](#), причем [alpha](#) можно рассматривать как расстояние вдоль линии соединения этих двух векторов - при [alpha](#) = 0, он будет равен данному вектору, а при [alpha](#) = 1 - вектору [v](#).

[.lerpVectors\(v1, v2, alpha \)](#)

[v1](#) - начальный [четырехмерный вектор \(Vector4\)](#).

[v2](#) - [четырехмерный вектор \(Vector4\)](#), до которого будет проводится операция интерполяции.

[alpha](#) - коэффициент интерполяции в закрытом интервале [0, 1].

Метод устанавливает данный вектор как линейно интерполированный вектор между векторами [v1](#) и [v2](#), и где [alpha](#) - это расстояние вдоль линии соединения этих двух векторов - при [alpha](#) = 0, он будет равен вектору [v1](#), при [alpha](#) = 1 - вектору [v2](#).

[.max\(v \)](#)

Если значение x, y, z или w данного вектора меньше значения x, y, z или w вектора, переданного в параметре [v](#), то это значение

заменяется соответствующим значением вектора [v](#).

[.min\(v \)](#)

Если значение x, y, z или w данного вектора больше значения x, y, z или w вектора, переданного в параметре [v](#), то это значение заменяется соответствующим значением вектора [v](#).

[.multiplyScalar\(s \)](#)

Метод умножает данный вектор на скаляр, переданный в параметре [s](#).

[.negate\(\)](#)

Метод инвертирует данный вектор, т.е. устанавливает x = -x, y = -y, z = -z и w = -w.

[.normalize\(\)](#)

Метод преобразовывает данный вектор в [единичный вектор \(unit vector\)](#) - то есть, направление его остается тем же самым, а длина вектора ([length](#)) устанавливается равной 1.

[.round\(\)](#)

Метод округляет компоненты вектора до ближайшего целочисленного значения.

[.roundToZero\(\)](#)

Метод округляет компоненты вектора к нулю (если значение отрицательно - происходит округление вверх, если положительное - вниз) до целочисленного значения.

[.set \(x, y, z, w \)](#)

Устанавливает компоненты [x](#), [y](#), [z](#) и [w](#) данного вектора.

[.setAxisAngleFromQuaternion\(q \)](#)

| [q](#) - нормализованный [кватернион](#).

Метод устанавливает значения компонентов [x](#), [y](#), [z](#) данного вектора по осям кватерниона, а значение компонента [w](#) по углу.

`.setAxisAngleFromRotationMatrix(m)`

m - матрица 4×4, в которой матрица 3×3, вверху слева, является простой матрицей вращения.

Метод устанавливает значения x, y, z по осям вращения, а значение w по углу.

`.setComponent(index, value)`

index -- целочисленное значение, может быть 0, 1, 2, или 3.

value -- число с плавающей запятой.

Если индекс равен 0, устанавливается значение x.

Если индекс равен 1, устанавливается значение y.

Если индекс равен 2, устанавливается значение z.

Если индекс равен 3, устанавливается значение w.

`.setLength (l)`

Метод устанавливает данный вектор как вектор с тем же самым направлением, но длиной (length) равной l.

`.setScalar(scalar)`

Устанавливает все значения x, y, z и w данного вектора равными скалярному значению, указанному в параметре scalar.

`.setX(x)`

Метод заменяет значение x данного вектора на значение, переданное в параметре x.

`.setY(y)`

Метод заменяет значение y данного вектора на значение, переданное в параметре y.

`.setZ(z)`

Метод заменяет значение z данного вектора на значение, переданное в параметре z.

.setW(w)

Метод заменяет значение w данного вектора на значение, переданное в параметре w.

.sub (v)

Метод вычитает из данного вектора вектор, указанный в параметре v.

.subScalar (s)

Метод вычитает из компонентов x, y, z и w данного вектора скалярное значение, указанное в параметре s.

.subVectors(a, b)

Метод устанавливает данный вектор как разницу векторов a - b.

.toArray(array, offset)

array - дополнительный, необязательный массив для хранения вектора. Если он не предоставлен, создается новый массив.

offset - дополнительное, необязательное смещение в массиве.

Возвращает массив в виде [x, y, z, w], или копирует x, y, z и w в предоставленный (array) массив.

Исходники

[Vector4.js на Гитхабе](#)

ОБЪЕКТЫ

Bone

[Object3D →](#)

Кость (bone) является частью скелета ([Skeleton](#)). Скелет, в свою очередь, используется "сеткой, покрытой кожей" ([SkinnedMesh](#)). Кости практически одинаковы с пустым [Object3D](#).

Пример

```
var root = new THREE.Bone();
var child = new THREE.Bone();

root.add( child );
child.position.y = 5;
```

Конструктор

`Bone(skin)`

`skin` — (дополнительный, необязательный аргумент)

[SkinnedMesh](#) которой принадлежит кость.

Свойства

`.skin`

An optional reference to the [page:[SkinnedMesh](#)].

Методы

`.clone()`

Возвращает клона данного объекта Bone и его потомков.

Исходники

[Bone.js в этом справочнике](#)

[Bone.js на github.com](#)

[Object3D →](#)

Group

Данный объект (группа) практически идентичен [Object3D](#). Его назначение сделать работу с группой объектов синтаксически яснее.

Пример

```
var geometry = new THREE.BoxBufferGeometry( 1, 1, 1 );
var material = new THREE.MeshBasicMaterial( {color: 0x00ff00} );

var cubeA = new THREE.Mesh( geometry, material );
cubeA.position.set( 100, 100, 0 );

var cubeB = new THREE.Mesh( geometry, material );
cubeB.position.set( -100, -100, 0 );

// create a group and add the two cubes
// These cubes can now be rotated / scaled etc as a group
// создаем группу и добавляем два куба
// Эти кубы теперь можно вращать/масштабировать и т.д. как группу
var group = new THREE.Group();
group.add( cubeA );
group.add( cubeB )

scene.add( group );
```

Конструктор

`Group()`

Свойства

Свойства одинаковы со свойствами [Object3D](#), за исключением:

`.type`

Строка 'Group'. Она не должна изменяться.

Методы

Методы одинаковы с методами [Object3D](#).

Исходники

[Group.js в этом справочнике](#)

[Group.js на github.com](#)

[Object3D](#) →

Line

Непрерывная линия.

Пример

```
var material = new THREE.LineBasicMaterial({
    color: 0x0000ff
});
```

```
var geometry = new THREE.Geometry();
geometry.vertices.push(
    new THREE.Vector3( -10, 0, 0 ),
    new THREE.Vector3( 0, 10, 0 ),
    new THREE.Vector3( 10, 0, 0 )
);

var line = new THREE.Line( geometry, material );
scene.add( line );
```

Конструктор

`Line([page:Geometry geometry], [page:Material material])`

`geometry` — Vertices representing the line segment(s).

`material` — Material for the line. Default is [page:LineBasicMaterial LineBasicMaterial].

If no material is supplied, a randomized line material will be created and assigned to the object.

Свойства

`[property:Geometry geometry]`

Vertices representing the line segment(s).

`[property:Material material]`

Material for the line.

Методы

`[method:Array raycast]([page:Raycaster raycaster], [page:Array int`

Get intersections between a casted ray and this Line.

[page:Raycaster.intersectObject] will call this method.

`[method:Line clone]()`

Returns a clone of this Line object and its descendants.

Исходники

[link:[https://github.com/mrdoob/three.js/blob/master/src/\[path\].js](https://github.com/mrdoob/three.js/blob/master/src/[path].js)

`src/[path].js`

[Object3D](#) → [Line](#) →

LineLoop

A continuous line that connects back to the start.

[Line](#) →

LineSegments

A series of lines drawn between pairs of vertices.

Конструктор

```
LineSegments( [page:Geometry geometry], [page:Material material] )
```

geometry — Pair(s) of vertices representing each line segment(s).

material — Material for the line. Default is [page:LineBasicMaterial LineBasicMaterial].

If no material is supplied, a randomized line material will be created and assigned to the object.

Свойства

```
[property:Geometry geometry]
```

Pair(s) of vertices representing the line segment(s).

```
[property:Material material]
```

Material for the line.

Методы

```
[method:Array raycast]( [page:Raycaster raycaster], [page:Array int
```

Get intersections between a casted ray and this Line.

[page:Raycaster.intersectObject] will call this method.

```
[method:LineSegments clone]()
```

Returns a clone of this LineSegments object and its descendants.

Source

[link:[https://github.com/mrdoob/three.js/blob/master/src/\[path\].js](https://github.com/mrdoob/three.js/blob/master/src/[path].js)
src/[path].js]

[Object3D →](#)

LOD

Level of Detail - Show meshes with more or less geometry based on distance.

Пример

[example:webgl_lod LOD]

```
var lod = new THREE.LOD();  
  
//Create 5 levels of spheres
```

```
        for( var i = 0; i < 5; i++ ) {

            var geometry = new THREE.IcosahedronGeometry();
            var material = new THREE.MeshBasicMaterial();
            var mesh = new THREE.Mesh( geometry, material );

            lod.addLevel( mesh, i * 50 );
        }
    }
```

Конструктор

`LOD()`

Свойства

`[property:array levels]`

An array of [page:Object level] objects

level is an object with two properties.

`distance` -- The distance at which to display this level of detail

`object` -- The Object3D which will be displayed

Методы

`[method:null addLevel]([page:Object3D mesh], [page:Float distance])`

`mesh` -- The Object3D to display

`distance` -- The distance at which to display this level of detail

Adds a mesh that will display at a certain distance and greater. Typically the further away the distance, the lower the detail on the mesh.

`[method:Object3D getObjectOfTypeForDistance]([page:Float distance])`

Get a reference to the first [page:Object3D] (mesh) that is greater than supplied distance.

`[method:null update]([page:Camera camera])`

`camera` -- The current camera

Update the visibility of the level of detail based on the distance from the camera.

`[method:LOD clone]()`

Returns a clone of this LOD object and its associated distance specific objects.

Исходники

[link:[https://github.com/mrdoob/three.js/blob/master/src/\[path\].js](https://github.com/mrdoob/three.js/blob/master/src/[path].js)
src/[path].js]

Mesh

Base class for Mesh objects, such as [page:MorphAnimMesh] and [page:SkinnedMesh].

Пример

```
var geometry = new THREE.BoxGeometry( 1, 1, 1 ); var material =
```

Конструктор

```
[name]( [page:Geometry geometry], [page:Material material] )
```

geometry — An instance of [page:Geometry].

material — An instance of [page:Material] (optional).

Свойства

```
[property:Geometry geometry]
```

An instance of [page:Geometry], defining the object's structure.

```
[property:Material material]
```

An instance of [page:Material], defining the object's appearance. Default is a [page:MeshBasicMaterial] with wireframe mode enabled and randomised colour.

```
[property:Array morphTargetInfluences]
```

An array of weights typically from 0-1 that specify how much of the morph is applied. Undefined by default, but reset to a blank array by [page:Mesh.updateMorphTargets updateMorphTargets].

```
[property:Array morphTargetDictionary]
```

A dictionary of morphTargets based on the morphTarget.name property. Undefined by default, but rebuilt [page:Mesh.updateMorphTargets updateMorphTargets].

```
[property:Integer drawMode]
```

Determines how the mesh triangles are constructed from the vertices. See the draw mode [page:DrawModes constants] for all possible values. Default is [page:DrawModes TrianglesDrawMode].

Методы

```
[method:null updateMorphTargets]()
```

Updates the morphtargets to have no influence on the object. Resets the [page:Mesh.morphTargetInfluences morphTargetInfluences],

[page:Mesh.morphTargetDictionary morphTargetDictionary], and [page:Mesh.morphTargetBase morphTargetBase] properties.

[method:Array raycast]([page:Raycaster raycaster], [page:Array int Get intersections between a casted ray and this mesh. [page:Raycaster.intersectObject] will call this method.

[method:Mesh clone]() Returns a clone of this Mesh object and its descendants.

Исходники

[link:[https://github.com/mrdoob/three.js/blob/master/src/\[path\].js](https://github.com/mrdoob/three.js/blob/master/src/[path].js) src/[path].js]

[Object3D](#) →

Points

A class for displaying points. For example, if using the [page:WebGLRenderer], the points are displayed using GL_POINTS.

Конструктор

Points([page:Geometry geometry], [page:Material material])
geometry — An instance of [page:Geometry].
material — An instance of [page:Material] (optional).

Свойства

[property:Geometry geometry]
An instance of [page:Geometry], where each vertex designates the position the points.

[property:Material material]
An instance of [page:Material], defining the object's appearance. Default is a [page:PointsMaterial] with randomised colour.

Методы

[method:Array raycast]([page:Raycaster raycaster], [page:Array int Get intersections between a casted ray and this Points. [page:Raycaster.intersectObject] will call this method.

[method:Points clone]() Returns a clone of this Points object and its descendants.

Исходники

[link:[https://github.com/mrdoob/three.js/blob/master/src/\[path\].js](https://github.com/mrdoob/three.js/blob/master/src/[path].js)
src/[path].js]

Skeleton

Use an array of [page:Bone bones] to create a skeleton that can be used by a [page:SkinnedMesh]. WebGL only.

Пример

```
// Create a simple "arm"

var bones = [];

var shoulder = new THREE.Bone();
var elbow = new THREE.Bone();
var hand = new THREE.Bone();

shoulder.add( elbow );
elbow.add( hand );

bones.push( shoulder );
bones.push( elbow );
bones.push( hand );

shoulder.position.y = -5;
elbow.position.y = 0;
hand.position.y = 5;

var armSkeleton = new THREE.Skeleton( bones );

// See THREE.SkinnedMesh for an example of usage with a mesh
```

Конструктор

`Skeleton([page:Array bones], [page:Array boneInverses], [page:Bool useVertexTexture])`

bones — The array of [page:bone bones]
boneInverses — (optional) An array of [page:Matrix4 Matrix4s]
useVertexTexture — (optional) Whether or not to use a vertex texture in the shader.

The constructor automatically sets up all of the properties below.

Свойства

`[property:Array bones]`

The array of [page:bone bones]

[property:Boolean useVertexTexture]

Whether or not to use a vertex texture in the shader, set in the constructor. Not all devices support floating point pixel textures. If this option is set then the bone matrices will be packed into a texture and sent to the shader. This method allows a much larger set of bones to be used. Otherwise the vertex shader will use uniforms, which do not allow for as many bones to be used. The exact numbers vary between devices.

[property:Array boneInverses]

An array of [page:Matrix4 Matrix4s] that represent the inverse of the matrixWorld of the individual bones.

[property:Integer boneTextureWidth]

The width of the vertex data texture.

[property:Integer boneTextureHeight]

The height of the vertex data texture.

[property:Float32Array boneMatrices]

The array buffer holding the bone data when using a vertex texture.

[property:DataTexture boneTexture]

The [page:DataTexture] holding the bone data when using a vertex texture.

Методы

[method:null calculateInverses]()

Generates the boneInverses.

[method:null pose]()

Returns the skeleton to the base pose.

[method:null update]()

Updates the [page:Float32Array boneMatrices] and [page:DataTexture boneTexture] after changing the bones. This is called automatically by the [page:WebGLRenderer] if the skeleton is used with a [page:SkinnedMesh].

[method:Skeleton clone]()

Returns a clone of this Skeleton object.

Исходники

[link:[https://github.com/mrdoob/three.js/blob/master/src/\[path\].js](https://github.com/mrdoob/three.js/blob/master/src/[path].js)
src/[path].js]

SkinnedMesh

A mesh that has a [page:Skeleton] with [page:Bone bones] that can then be used to animate the vertices of the geometry.

Пример

```
var geometry = new THREE.CylinderGeometry( 5, 5, 5, 5, 15, 5, 30 );

//Create the skin indices and skin weights
for ( var i = 0; i < geometry.vertices.length; i ++ ) {

    // Imaginary functions to calculate the indices and weights
    // This part will need to be changed depending your skeleton and
    var skinIndex = calculateSkinIndex( geometry.vertices, i );
    var skinWeight = calculateSkinWeight( geometry.vertices, i );

    // Ease between each bone
    geometry.skinIndices.push( new THREE.Vector4( skinIndex, skinIndex,
    geometry.skinWeights.push( new THREE.Vector4( 1 - skinWeight, skinWeight, skinWeight, skinWeight ) );

}

var mesh = THREE.SkinnedMesh( geometry, material );

// See example from THREE.Skeleton for the armSkeleton
var rootBone = armSkeleton.bones[ 0 ];
mesh.add( rootBone );

// Bind the skeleton to the mesh
mesh.bind( armSkeleton );

// Move the bones and manipulate the model
armSkeleton.bones[ 0 ].rotation.x = -0.1;
armSkeleton.bones[ 1 ].rotation.x = 0.2;
```

Конструктор

`SkinnedMesh([page:Geometry geometry], [page:Material material], [p
geometry — An instance of [page:Geometry].
[page:Geometry.skinIndices] and [page:Geometry.skinWeights]
should be set.
material — An instance of [page:Material] (optional).
useVertexTexture -- Defines whether a vertex texture can be used`

(optional).

Свойства

[property:array bones]

This contains the array of bones for this mesh. These should be set in the constructor.

[property:Matrix4 identityMatrix]

This is an identityMatrix to calculate the bones matrices from.

[property:boolean useVertexTexture]

The boolean defines whether a vertex texture is used to calculate the bones. This boolean shouldn't be changed after constructor.

[property:array boneMatrices]

This array of matrices contains the matrices of the bones. These get calculated in the constructor.

[property:string bindMode]

Either "attached" or "detached". "attached" uses the [page:SkinnedMesh.matrixWorld] property for the base transform matrix of the bones. "detached" uses the [page:SkinnedMesh.bindMatrix].

[property:Matrix4 bindMatrix]

The base matrix that is used for the bound bone transforms.

[property:Matrix4 inverseBindMatrix]

The inverse of the bindMatrix.

Методы

[method:null bind]([page:Skeleton skeleton], [page:Matrix4 bindMat
skeleton — [page:Skeleton]

bindMatrix — [page:Matrix4] that represents the base transform of the
skeleton

Bind a skeleton to the skinned mesh. The bindMatrix gets saved to
.bindMatrix property and the .bindMatrixInverse gets calculated.

[method:null normalizeSkinWeights]()

Normalizes the [page:Geometry.skinWeights] vectors. Does not affect
[page:BufferGeometry].

[method:null pose]()

This method sets the skinned mesh in the rest pose.

[method:Bone addBone]([page:Bone bone])

bone — This is the bone that needs to be added. (optional)

This method adds the bone to the skinned mesh when it is provided. It creates a new bone and adds that when no bone is given.

[method:SkinnedMesh clone]()

Returns a clone of this SkinnedMesh object and its descendants.

Исходники

[link:[https://github.com/mrdoob/three.js/blob/master/src/\[path\].js](https://github.com/mrdoob/three.js/blob/master/src/[path].js)
src/[path].js]

[Object3D →](#)

Sprite

Спрайт это плоскость на трехмерной сцене, которая всегда стоит перед камерой.

Спрайты не отбрасывают тени, так что установка castShadow = true не даст никакого эффекта.

Пример

```
var map = new THREE.TextureLoader().load( "sprite.png" );
var material = new THREE.SpriteMaterial( { map: map, color: 0xffff
var sprite = new THREE.Sprite( material );
scene.add( sprite );
```

Конструктор

Sprite([material](#))

material — экземпляр [Material](#) (дополнительный, необязательный аргумент).

Создает новый объект Sprite с указанным материалом material.

Свойства

[property:SpriteMaterial material]

An instance of [page:Material], defining the object's appearance.

Значение по умолчанию равно [page:SpriteMaterial], который является белой плоскостью.

Методы

[method:Sprite clone]()

Возвращает клон данного объекта Sprite и его потомков.

Исходники

[link:[https://github.com/mrdoob/three.js/blob/master/src/\[path\].js](https://github.com/mrdoob/three.js/blob/master/src/[path].js)
src/[path].js]

src/[path].js

ВИЗУАЛИЗАТОРЫ

WebGLRenderer

The WebGL renderer displays your beautifully crafted scenes using [WebGL](#). Вот [статья Википедии о WebGL](#) на русском языке.

Конструктор

```
WebGLRenderer( parameters )
```

parameters - объект со свойствами, определяющими поведение визуализатора (рендерера) (дополнительный, необязательный параметр).

Конструктор также допускает полное отсутствие параметров. Во всех подобных случаях, им будут приниматься соответствующие значения по умолчанию.

Допустимы следующие параметры:

canvas — [холст](#), на который визуализатор выводит результаты своей работы. Он соответствует свойству [domElement](#), указанному ниже. Если он не был передан здесь, то будет создан новый элемент canvas.

context — этот параметр можно использовать для присоединения визуализатора (рендерера) к существующему [RenderingContext](#). Значением по умолчанию является null.

Вот статья о [RenderingContext](#) на русском языке.

precision — точность шейдера. Может быть `highp` (похоже сокращ. от англ. слов `high precision` - высокая точность), `mediump` (от слов `medium precision` - средняя точность) или `lowp` (от слов `low precision` - низкая точность). Значением по умолчанию является `highp`, если поддерживается устройством. Смотрите примечания "Things to Avoid" в статье [WebGL best practices](#). На русском языке это будет раздел "Чего следует избегать" в статье [Лучшие практики WebGL](#).

alpha — параметр определяет, будет ли холст (canvas)

содержать буфер альфа-канала (канал прозрачности) или нет. Значением по умолчанию является `false`.

[premultipliedAlpha](#) — будет ли визуализатор (рендерер) предполагать, что цвета `whether the renderer will assume that colors have premultiplied alpha`. Значением по умолчанию является `true`.

Примечание переводчика: предварительное перемножение альфа-канала (premultiplied alpha) - вариант вычисления растрового изображения или альфа-смешивания, в котором предполагается, что RGB значения цвета уже умножены на значения альфа-канала, для снижения вычислений во время альфа-смешивания; применяется операция смешивания:

`dst = dst * (1 - alpha) + src;` допускающая соединение альфа-смешивание с дополнительными эффектами смешивания. A variation of a bitmap image or alpha blending calculation in which the RGB color values are assumed to be already multiplied by an alpha channel, to reduce computations during Alpha blending; uses the blend operation: `dst = dst * (1 - alpha) + src;` capable of mixing alpha blending with additive blending effects

[antialias](#) — параметр определяет, будет ли выполняться сглаживание (antialiasing). Значением по умолчанию является `false`.

[stencil](#) — будет ли в буфере рисования находиться [буфер трафарета](#) не меньше 8 бит. Значением по умолчанию является `true`.

Примечание переводчика: вот перевод начала вышеприведенной статьи о буфере трафарета. ... [читать далее](#)

[preserveDrawingBuffer](#) — `whether to preserve the buffers until manually cleared or overwritten.` Значением по умолчанию является `false`.

[depth](#) — `whether the drawing buffer has a depth buffer of at least 16 bits.` Значением по умолчанию является `true`.

[logarithmicDepthBuffer](#) — параметр определяет, будет ли использоваться логарифмический буфер глубины. Его использование может понадобиться, если имеются очень большие различия в масштабе при работе в одной сцене. Значением по умолчанию является `false`. Смотрите пример [camera / logarithmicdepthbuffer](#).

Свойства

.autoClear

Свойство определяет, будет ли визуализатор (рендерер) автоматически очищать свой выход перед визуализацией (рендерингом).

.autoClearColor

Свойство определяет, должен ли визуализатор (рендерер) очищать буфер цвета, если значение свойства [autoClear](#) установлено как true. Значением по умолчанию является true.

.autoClearDepth

Свойство определяет, должен ли визуализатор (рендерер) очищать буфер глубины, если значение свойства [autoClear](#) установлено как true. Значением по умолчанию является true.

.autoClearStencil

Свойство определяет, должен ли визуализатор (рендерер) очищать буфер трафарета, если значение свойства [autoClear](#) установлено как true. Значением по умолчанию является true.

.capabilities

Объект, содержащий сведения о возможностях текущего [RenderingContext](#). Вот [статья Википедии о RenderingContext](#) на русском языке.

[.floatFragmentTextures](#) : поддерживает ли контекст расширение [OES_texture_float](#). По данным [WebGLStats](#), по состоянию на февраль 2016 года, его поддерживают свыше 95% устройств of WebGL enabled devices support this.

[.floatVertexTextures](#) : true if .floatFragmentTextures and .vertexTextures are both true.

[.getMaxAnisotropy\(\)](#) :смотрите ниже метод [getMaxAnisotropy](#).

[.getMaxPrecision\(\)](#) :смотрите ниже метод [getMaxPrecision](#).

[.logarithmicDepthBuffer](#) : true if the .logarithmicDepthBuffer was set to true in the constructor and the context supports the EXT_frag_depth extension. According to WebGLStats, as of

February 2016 around 66% of WebGL enabled devices support this.

[.maxAttributes](#) : The value of gl.MAX_VERTEX_ATTRIBS.

[.maxCubemapSize](#) : The value of gl.MAX_CUBE_MAP_TEXTURE_SIZE. Maximum height * width of cube map textures that a shader can use.

[.maxFragmentUniforms](#) : The value of gl.MAX_FRAGMENT_UNIFORM_VECTORS. The number of uniforms that can be used by a fragment shader.

[.maxTextureSize](#) : The value of gl.MAX_TEXTURE_SIZE. Maximum height * width of a texture that a shader use.

[.maxTextures](#) : The value of gl.MAX_TEXTURE_IMAGE_UNITS. The maximum number of textures that can be used by a shader.

[.maxVaryings](#) : The value of gl.MAX_VARYING_VECTORS. The number of varying vectors that can be used by shaders.

[.maxVertexTextures](#) : The value of gl.MAX_VERTEX_TEXTURE_IMAGE_UNITS. The number of textures that can be used in a vertex shader.

[.maxVertexUniforms](#) : The value of gl.MAX_VERTEX_UNIFORM_VECTORS. The maximum number of uniforms that can be used in a vertex shader.

[.precision](#) : The shader precision currently being used by the renderer.

[.vertexTextures](#) : true if .maxVertexTextures is greater than 0 (i.e. vertex textures can be used).

[.clippingPlanes](#)

User-defined clipping planes specified as THREE.Plane objects in world space. These planes apply globally. Points in space whose dot product with the plane is negative are cut away. Значением по умолчанию является [].

[.context](#)

The renderer obtains a RenderingContext context from its domElement by default, using HTMLCanvasElement.getContext().

You can create this manually, however it must correspond to the domElement in order to render to the screen.

.domElement

А [page:Canvas where the renderer draws its output.

Он автоматически создается визуализатором (рендерером) в конструкторе (если уже не предоставлен); нужно просто добавить его на страницу, вот так:

```
document.body.appendChild( renderer.domElement );
```

.extensions

Оболочка для метода [extensions.get](#), используется для проверки поддержки различных расширений WebGL.

.gammaFactor

[Гамма-коррекция](#), т.е. коррекция яркости или цветовых компонентов цифрового изображения или видео. Значение по умолчанию равно 2.

.gammaInput

Если свойство установлено, то предполагается, что все текстуры и цвета предварительно перемножены на значение gammaFactor. Значением по умолчанию является false.

.gammaOutput

Если свойство установлено то предполагается, что все текстуры и цвета должны выводиться предварительно перемноженными на значение gammaFactor. Значением по умолчанию является false.

.info

Объект с рядом статических сведений о памяти видеокарты и процессе визуализации. Полезно при отладке или просто из любопытства. Объект содержит следующие поля:

- memory (память):
 - geometries (геометрические элементы)

- textures (текстуры)
 - render (визуализация):
 - calls (вызовы)
 - vertices (вершины)
 - faces (границы)
 - points (точки)
 - programs (программы)

.localClippingEnabled

Свойство определяет, Defines whether the renderer respects object-level clipping planes. Значением по умолчанию является false.

.maxMorphNormals

Значение по умолчанию равно 4. Default is 4. The maximum number of MorphNormals allowed in a shader. Keep in mind that the standard materials only allow 4 MorphNormals.

.maxMorphTargets

Значение по умолчанию равно 8. Максимальное число целей морфинга (MorphTargets), допускаемых в шейдере. Имейте ввиду, что стандартные материалы допускают только 8 целей морфинга. Default is 8. The maximum number of MorphTargets allowed in a shader. Keep in mind that the standard materials only allow 8 MorphTargets.

.physicallyCorrectLights

Свойство определяет, использовать ли режим освещения, правильный с точки зрения физики. Значением по умолчанию является `false`. Смотрите пример [lights / physical](#).

.properties

Used internally by the renderer to keep track of various sub object properties.

.renderLists

Used internally to handle ordering of scene object rendering.

.shadowMap

This contains the reference to the shadow map, if used. #

```
.shadowMap.enabled
```

If set, use shadow maps in the scene. Default is false. #

```
.shadowMap.type
```

Defines shadow map type (unfiltered, percentage close filtering, percentage close filtering with bilinear filtering in shader) Options are THREE.BasicShadowMap, THREE.PCFShadowMap (default), THREE.PCFSOFTShadowMap. See WebGLRenderer constants for details. #

```
.shadowMap.renderReverseSided
```

Whether to render the opposite side as specified by the material into the shadow map. When disabled, an appropriate shadow.bias must be set on the light source for surfaces that can both cast and receive shadows at the same time to render correctly. Default is true. #

```
.shadowMap.renderSingleSided
```

Whether to treat materials specified as double-sided as if they were specified as front-sided when rendering the shadow map. When disabled, an appropriate shadow.bias must be set on the light source for surfaces that can both cast and receive shadows at the same time to render correctly. Default is true. #

```
.sortObjects
```

Defines whether the renderer should sort objects. Default is true.

Note: Sorting is used to attempt to properly render objects that have some degree of transparency. By definition, sorting objects may not work in all cases. Depending on the needs of application, it may be necessary to turn off sorting and use other methods to deal with transparency rendering e.g. manually determining each object's rendering order. #

```
.state
```

Объект содержит функции для установки разных свойств состояния [WebGLRenderer.context](#).

```
.toneMapping
```

Значением по умолчанию является [LinearToneMapping](#). Другие вариантысмотрите в описании [констант WebGLRenderer'a](#).

```
.toneMappingExposure
```

Exposure level of tone mapping. Значение по умолчанию равно 1.

[.toneMappingWhitePoint](#)

Tone mapping white point. Значение по умолчанию равно 1.

Методы

[.allocTextureUnit](#)

Метод пытается выделить блок текстуры для использования шейдером. При попытке выделения блоков текстуры больше, чем поддерживается GPU, будет предупреждение. В основном этот метод используется внутренне. Смотрите описание свойства [capabilities.maxTextures](#).

[.clear \(color, depth, stencil \)](#)

Tells the renderer to clear its color, depth or stencil drawing buffer(s). This method initializes the color buffer to the current clear color value. По умолчанию аргументы равны true.

[. clearColor \(\)](#)

Метод очищает буфер цвета. Равнозначен вызову метода [.clear\(true, false, false \)](#).

[.clearDepth \(\)](#)

Метод очищает буфер глубины. Равнозначен вызову метода [.clear\(false, true, false \)](#).

[.clearStencil \(\)](#)

Метод очищает буфер трафарета. Равнозначен вызову метода [.clear\(false, false, true \)](#).

[.clearTarget \(renderTarget, color, depth, stencil \)](#)

renderTarget -- целевой буфер визуализации, который нужно очистить.

color -- если аргумент установлен, то буфер цвета очищается.

depth -- если аргумент установлен, то буфер глубины

очищается.

[stencil](#) -- если аргумент установлен, то буфер трафарета очищается.

Этот метод очищает целевой буфер визуализации. Чтобы сделать это, он активирует целевой буфер визуализации.

`.compile (scene, camera)`

Compiles all materials in the scene with the camera. This is useful to precompile shaders before the first rendering.

`.dispose ()`

Dispose of the current rendering context.

`.extensions.get (extensionName)`

Метод используется для проверки поддержки разных расширений и, если таковая имеется, возвращает объект с подробной информацией о расширении. Он может проверять следующие расширения:

WEBGL_depth_texture
EXT_texture_filter_anisotropic
WEBGL_compressed_texture_s3tc
WEBGL_compressed_texture_pvrtc
WEBGL_compressed_texture_etc1

`.forceContextLoss ()`

Simulate loss of the WebGL context. This requires support for the WEBGL_lose_context extensions. According to WebGLStats, as of February 2016 90% of WebGL enabled devices support this.

`.getClearAlpha ()`

Returns a float with the current clear alpha. Диапазон значений от 0 до 1.

`.getClearColor ()`

Returns a THREE.Color instance with the current clear color.

```
.getContext ( )
```

Метод возвращает текущий контекст WebGL.

```
.getContextAttributes ( )
```

Returns an object that describes the attributes set on the WebGL context when it was created.

```
.getCurrentRenderTarget ( )
```

Returns the current RenderTarget, if any.

```
.getMaxAnisotropy ( )
```

This returns the anisotropy level of the textures.

```
.getDrawingBufferSize ( )
```

Returns an object containing the width and height of the renderer's drawing buffer, in pixels.

```
.getPixelRatio ( )
```

Returns current device pixel ratio used.

```
.getPrecision ( )
```

This gets the precision used by the shaders. It returns "highp", "mediump" or "lowp".

```
.getSize ( )
```

Returns an object containing the width and height of the renderer's output canvas, in pixels.

```
.resetGLState ( )
```

Reset the GL state to default. Called internally if the WebGL context is lost.

```
.readRenderTargetPixels ( renderTarget, x, y, width, height, buff
```

Reads the pixel data from the renderTarget into the buffer you pass in.

Buffer should be a Javascript Uint8Array instantiated with new Uint8Array(renderTargetWidth * renderTargetWidth * 4) to account

for size and color information. This is a wrapper around gl.readPixels. See the interactive / cubes / gpu example.

```
.render ( scene, camera, renderTarget, forceClear )
```

Render a scene using a camera. The render is done to the renderTarget (if specified) or to the canvas as usual. If forceClear is true, the depth, stencil and color buffers will be cleared before rendering even if the renderer's autoClear property is false. Even with forceClear set to true you can prevent certain buffers being cleared by setting either the autoClearColor, autoClearStencil or autoClearDepth properties to false.

```
.renderBufferDirect ( camera, lights, fog, material, geometryGrou
```

Render a buffer geometry group using the camera and with the specified material.

```
.renderBufferImmediate ( object, program, shading )
```

object - an instance of Object3D
program - an instance of shaderProgram
shading - an instance of Material
Render an immediate buffer. Gets called by renderImmediateObject.

```
.setClearAlpha ( alpha )
```

Sets the clear alpha. Valid input is a float between 0.0 and 1.0.

```
.setClearColor ( color, alpha )
```

Sets the clear color and opacity.

```
.setFaceCulling ( cullFace, frontFace )
```

See WebGLRenderer constants for all possible values for cullFace and frontFace. Used for setting the gl.frontFace and gl.cullFace states in the GPU, thus enabling/disabling face culling when rendering. If cullFace is set to CullFaceNone, culling will be disabled.

```
.setPixelRatio ( value )
```

Sets device pixel ratio. This is usually used for HiDPI device to prevent bluring output canvas.

```
.setRenderTarget ( renderTarget )
```

renderTarget -- The renderTarget that needs to be activated (optional). This method sets the active rendertarget. If the parameter is omitted the canvas is set as the active rendertarget.

```
.setScissor ( x, y, width, height )
```

Sets the scissor area from (x, y) to (x + width, y + height)

```
.setScissorTest ( boolean )
```

Enable or disable the scissor test. When this is enabled, only the pixels within the defined scissor area will be affected by further renderer actions.

```
.supportsVertexTextures( )
```

Возвращает логическое значение true, если контекст поддерживает текстуры вершин. Этот метод устарел Return a Boolean true if the context supports vertex textures. This has been deprecated in favour of capabilities.vertexTextures.

```
.setSize( width, height, updateStyle )
```

Изменяет размеры выходного холста (canvas) к (width, height)
Resizes the output canvas to (width, height) with device pixel ratio taken into account, а также устанавливает область просмотра (viewport) в соответствии с этим размером, начиная с (0, 0). Установка параметра updateStyle как true добавляет adds explicit pixel units to the output canvas style.

```
.setTexture2D( texture, slot )
```

texture -- текстура, которую нужно установить.

slot -- число, показывающее какой слот должен использоваться текстурой.

This method sets the correct texture to the correct slot for the WebGL shader. The slot number can be found as a value of the uniform of the sampler.

Примечание: Этот метод заменяет устаревший метод

```
.setTexture.
```

```
.setTextureCube( cubeTexture, slot )
```

cubeTexture -- The cubeTexture that needs to be set.

slot -- число, показывающее какой слот должен

использоваться текстурой. The number indicating which slot should be used by the texture.

Этот метод устанавливает правильную текстуру в правильный слот для шейдера WebGL. Номер слота можно найти как значение This method sets the correct texture to the correct slot for the WebGL shader. The slot number can be found as a value of the uniform of the sampler.

```
.setViewport( x, y, width, height )
```

Метод устанавливает область просмотра (viewport) визуализации от (x, y) до (x + width, y + height).

Исходники

[WebGLRenderer.js в этом справочнике](#)

[WebGLRenderer.js на github.com](#)

WebGLRenderTarget

[Целевой буфер визуализации](#) - это буфер, где видеокарта рисует пиксели сцены, представленной на заднем плане. Он используется в разных эффектах, вроде применения постобработки к отображаемому изображению перед показом его на экране.

Примечание переводчика: вот перевод вышеприведенной статьи о целевом буфере визуализации. ... читать далее

Конструктор

```
WebGLRenderTarget( width, height, options )
```

width -- ширина renderTarget.

height -- высота renderTarget.

options - дополнительный, необязательный объект, который содержит параметры текстуры для автоматической генерации

целевой текстуры и логические значения буферов глубины/трафаретов (depthBuffer/stencilBuffer). Пояснения к параметрам текстуры смотрите в статье [Texture](#).

Возможны следующие опции:

`wraps` - константа, значением по умолчанию является

`THREE.ClampToEdgeWrapping`.

`wrapT` - константа, значением по умолчанию является

`THREE.ClampToEdgeWrapping`.

`magFilter` - константа, значением по умолчанию является

`THREE.LinearFilter`.

`minFilter` - константа, значением по умолчанию является

`THREE.LinearFilter`.

`format` - константа, значением по умолчанию является

`THREE.RGBAFormat`.

`type` - константа, значением по умолчанию является

`THREE.UnsignedByteType`.

`anisotropy` — значением по умолчанию является 1. Смотрите

описание свойства [Texture.anisotropy](#).

`encoding` - константа, значением по умолчанию является

`THREE.LinearEncoding`.

`depthBuffer` — логическое значение, по умолчанию равно `true`.

Установите его как `false`, если буфер глубины не нужен.

`stencilBuffer` — логическое значение, по умолчанию равно

`true`. Установите его как `false`, если буфер трафарета не нужен.

Создает новый буфер целевой визуализации (`WebGLRenderTarget`) с определенными шириной и высотой.

Свойства

`.uuid`

Уникальное число для данного экземпляра целевого буфера визуализации.

`.width`

Ширина целевого буфера визуализации.

[.height](#)

Высота целевого буфера визуализации.

[.scissor](#)

Прямоугольная область внутри окна просмотра целевого буфера визуализации. Фрагменты, находящиеся за пределами этой области, будут отбрасываться.

[.scissorTest](#)

Свойство показывает, активна или нет Indicates whether the scissor test is active or not.

[.viewport](#)

Область (окно) просмотра данного целевого буфера визуализации.

[\[property:Texture .texture\]](#)

This texture instance holds the rendered pixels. Use it as input for further processing.

[.depthBuffer](#)

Свойство определяет, представлять ли буфер глубины. Значением по умолчанию является true.

[.stencilBuffer](#)

Свойство определяет, представлять ли буфер трафаретов. Значением по умолчанию является true.

[depthTexture](#)

Если это свойство установлено, глубина сцены будет отображаться по данной текстуре. Значением по умолчанию является null.

Методы

`.setSize(width, height)`

Устанавливает размеры данного целевого буфера визуализации.

`.clone()`

Создает копию данного целевого буфера визуализации.

`.copy(source)`

Перенимает настройки заданного (в параметре `source`) целевого буфера визуализации.

`.dispose()`

Метод отправляет Dispatches a dispose event.

Для данного класса доступны методы [EventDispatcher](#).

Исходники

[WebGLRenderTarget.js в этом справочнике](#)

[WebGLRenderTarget.js на Гитхабе](#)

[WebGLRenderTarget](#) →

WebGLRenderTargetCube

Данный буфер целевой визуализации используется [CubeCamera](#)'ой в качестве своего [WebGLRenderTarget](#).

Примеры

Примеры смотрите в описании [CubeCamera](#).

Конструктор

`WebGLRenderTargetCube(width, height, options)`

`width` -- ширина renderTarget.

`height` -- высота renderTarget.

`options` - дополнительный, необязательный объект, который содержит параметры текстуры для автоматической генерации

целевой текстуры и логические значения буферов глубины/трафаретов (depthBuffer/stencilBuffer). Пояснения к параметрам текстуры смотрите в статье [Texture](#).

Возможны следующие опции:

`wraps` - константа, значением по умолчанию является

[THREE.ClampToEdgeWrapping](#).

`wrapT` - константа, значением по умолчанию является

[THREE.ClampToEdgeWrapping](#).

`magFilter` - константа, значением по умолчанию является

[THREE.LinearFilter](#).

`minFilter` - константа, значением по умолчанию является

[THREE.LinearFilter](#).

`format` - константа, значением по умолчанию является

[THREE.RGBAFormat](#).

`type` - константа, значением по умолчанию является

[THREE.UnsignedByteType](#).

`anisotropy` — значением по умолчанию является 1. Смотрите

описание свойства [Texture.anisotropy](#).

`encoding` - константа, значением по умолчанию является

[THREE.LinearEncoding](#).

`depthBuffer` — логическое значение, по умолчанию равно true.

Установите его как false, если буфер глубины не нужен.

`stencilBuffer` — логическое значение, по умолчанию равно

true. Установите его как false, если буфер трафарета не нужен.

Создает новый WebGLRenderTargetCube.

Свойства

Для ознакомления с унаследованными свойствами смотрите

[WebGLRenderTarget](#).

[`.activeCubeFace`](#)

Свойство activeCubeFace соответствует стороне куба (PX 0, NX 1, PY 2, NY 3, PZ 4, NZ 5) and is used and set internally by the [CubeCamera](#).

Методы

Для ознакомления с унаследованными методамисмотрите
[WebGLRenderTarget](#).

Исходники

[WebGLRenderTargetCube.js в этом справочнике](#)

[WebGLRenderTargetCube.js на Гитхабе](#)

Шейдеры

ShaderChunk

Блоки шейдеров для библиотеки WebLG Shader

[Свойства](#)

[Методы](#)

[Исходники](#)

[ShaderChunk.js в этом справочнике](#)

[ShaderChunk.js на github.com](#)

ShaderLib

Библиотека Webgl Shader для *three.js*

[Свойства](#)

[Методы](#)

[Исходники](#)

[ShaderLib.js в этом справочнике](#)

[ShaderLib.js на github.com](#)

UniformsLib

Библиотека однообразных переменных (uniform) для общеспользуемых webgl шейдеров. Uniforms library for shared webgl shaders

[Свойства](#)

[Методы](#)

[Исходники](#)

[UniformsLib.js в этом справочнике](#)

[UniformsLib.js на github.com](#)

UniformsUtils

Uniform Utilities. Поддерживает слияние и клонирование однообразных переменных (uniform).

[Свойства](#)

[Методы](#)

[Исходники](#)

[UniformsUtils.js в этом справочнике](#)

[UniformsUtils.js на github.com](#)

СЦЕНЫ

Fog

Данный класс содержит параметры, определяющие линейный туман (fog), т.е. плотность тумана линейно возрастает с увеличением расстояния.

Конструктор

```
Fog( hex, near, far )
```

Параметр hex передается в конструктор [Color](#) для установки свойства color - цвет. Значением параметра hex может быть шестнадцатиричное целое число или строка CSS-стиля.

Свойства

.name

Значением по умолчанию является пустая строка ("").

.color

Цвет тумана. Например, если установить его черным, дальние объекты будут отображаться черными.

.near

Минимальное расстояние, с которого начинается применение тумана. На объекты, размещенные от активной камеры ближе расстояния, указанного в near, туман не повлияет.

Значение по умолчанию равно 1.

.far

Максимальное расстояние, на котором прекращается расчет и применение тумана. На объекты, размещенные дальше от

активной камеры, чем указано в `far`, туман не повлияет.
Значение по умолчанию равно 1000.

Методы

[.clone\(\)](#)

Возвращает новый экземпляр тумана с теми же параметрами как у данного.

[.toJSON\(\)](#)

Возвращает данные тумана в формате JSON.

Исходники

[Fog.js на github.com](https://github.com)

FogExp2

Данный класс содержит параметры, определяющие экспоненциальный туман (`fog`), т.е. плотность тумана с увеличением расстояния, возрастает экспоненциально (в геометрической прогрессии).

Конструктор

`FogExp2(hex, density)`

Параметр `hex` передается в конструктор [Color](#) для установки свойства `color` - цвет. Значением параметра `hex` может быть шестнадцатиричное целое число или строка CSS-стиля.

Свойства

[.name](#)

Значением по умолчанию является пустая строка ("").

.color

Цвет тумана. Например, если установить его черным, дальние объекты будут отображаться черными.

.density

Определяет, как быстро будет нарастать плотность тумана. Значение по умолчанию равно 0.00025.

Методы

.clone()

Возвращает копию (клон) этого тумана.

to JSON - сокр. от англ. слов to JavaScript Object Notation,
— то есть, в текстовый формат описания объекта,
основанный на JavaScript');" onmouseout="hide()">>.toJSON()

Возвращает данные тумана FogExp2 в формате JSON.

Исходники

[FogExp2.js на github.com](#)

[Object3D](#) →

Scene

Сцены позволяют устанавливать все, как и где должно быть показано с помощью ***three.js***. Это то место, где размещаются объекты, освещение и камеры.

Конструктор

Scene()

Создает новый объект места действия (новую сцену).

Свойства

.fog

Экземпляр [тумана](#), определяющий тип тумана, что будет влиять на все, что отображается на сцене. Значением по умолчанию является null.

.overrideMaterial

Если значение не равно null, метод принудительно будет показывать все вещи на сцене с этим материалом. Значение по умолчанию равно null.

.autoUpdate

Значением по умолчанию является true. Если установлено, то визуализатор (рендерер) проверяет на каждом кадре, не требуются ли обновления матриц сцены и всех объектов на ней. Если это не так, то необходимо сохранять все матрицы сцены самостоятельно.

.background

Значение, не равное null, задает фон, который используется при визуализации сцены и который всегда отображается первым. Может устанавливаться как значение [цвета](#), которое определяет чистый (однотонный) цвет, как текстура ([Texture](#)), покрывающая холст, или как текстура куба ([CubeTexture](#)). Значение по умолчанию равно null.

Методы

— то есть, в текстовый формат описания объекта, основанный на JavaScript');" onmouseout="hide()">>.toJSON

Возвращает данные сцены в формате JSON.

Исходники

[Scene.js на github.com](#)

ТЕКСТУРЫ

Текстура — растровое изображение, накладываемое на поверхность полигональной модели для придания ей цвета, окраски или иллюзии рельефа.

[... читать далее](#)

[Texture →](#)

CanvasTexture

Создает текстуру из элемента canvas.

Это фактически тоже самое, что и базовый класс [Texture](#), за исключением того, что свойство [needsUpdate](#) сразу же устанавливается как true.

Конструктор

`CanvasTexture(canvas, mapping, wrapS, wrapT, magFilter, minFilter,`

[canvas](#) -- HTML элемент canvas, из которого загружается текстура.

[page:Constant mapping -- How the image is applied to the object. An object type of [page:Textures THREE.UVMapping. Другие варианты смотрите в [константах картирования](#).

[page:Constant wrapS -- The default is [page:Textures THREE.ClampToEdgeWrapping. See [page:Textures wrap mode constants for other choices.

[page:Constant wrapT -- The default is [page:Textures THREE.ClampToEdgeWrapping. See [page:Textures wrap mode constants for other choices.

[page:Constant magFilter -- How the texture is sampled when a texel covers more than one pixel. The default is [page:Textures THREE.LinearFilter. See [page:Textures magnification filter constants for other choices.

[page:Constant minFilter -- How the texture is sampled when a texel covers less than one pixel. The default is [page:Textures THREE.LinearMipMapLinearFilter. See [page:Textures minification filter constants for other choices.

[page:Constant format -- The format used in the texture. See [page:Textures format constants for other choices.]

[page:Constant type -- Default is [page:Textures THREE.UnsignedByteType]. See [page:Textures type constants for other choices.]

[page:Number anisotropy -- The number of samples taken along the axis through the pixel that has the highest density of texels. By default, this value is 1. A higher value gives a less blurry result than a basic mipmap, at the cost of more texture samples being used.

Use [page:WebGLrenderer.getMaxAnisotropy()](#WebGLrenderer.getMaxAnisotropy) to find the maximum valid anisotropy value for the GPU; this value is usually a power of 2.

Свойства

Общие свойства смотрите в описании базового класса [Texture](#).

.needsUpdate

Значением по умолчанию является `true`. Это необходимо для загрузки данных холста (canvas).

Методы

Общие методы смотрите в описании базового класса [Texture](#).

Исходники

[CanvasTexture.js в этом справочнике](#)

[CanvasTexture.js на github.com](#)

[Texture →](#)

CompressedTexture

Создает текстуру, основываясь на данных в сжатом виде.

Конструктор

`CompressedTexture(mipmaps, width, height, format, type, mapping, w`

`mipmaps` - массив тірмар должен содержать объекты с данными, шириной и высотой. `mipmaps` должны быть правильного формата и типа.

<https://ru.wikipedia.org/wiki/MIP-%D1%82%D0%B5%D0%BA%D1%81%D1%82%D1%83%D1%80>
тірмар - множественное отображение (последовательность текстур одного и того же изображения с уменьшающимся разрешением по мере удаления отображаемого объекта от наблюдателя)

`width` - ширина наибольшего тірмар

`height` - высота наибольшего тірмар

`format` - формат, используемый в `mipmaps`

`type` - тип, используемый в `mipmaps`

`mapping` -- How the image is applied to the object. An object type of `THREE.UVMapping` is the default, where the U,V coordinates are used to apply the map, and a single texture is expected. The other types are `THREE.CubeReflectionMapping`, for cube maps used as a reflection map; `THREE.CubeRefractionMapping`, refraction mapping; and `THREE.SphericalReflectionMapping`, a spherical reflection map projection.

`wraps` -- The default is `THREE.ClampToEdgeWrapping`, where the edge is clamped to the outer edge texels. The other two choices are `THREE.RepeatWrapping` and `THREE.MirroredRepeatWrapping`.

`wrapT` -- The default is `THREE.ClampToEdgeWrapping`, where the edge is clamped to the outer edge texels. The other two choices are `THREE.RepeatWrapping` and `THREE.MirroredRepeatWrapping`.

`magFilter` -- How the texture is sampled when a texel covers more than one pixel. The default is `THREE.LinearFilter`, which takes the four closest texels and bilinearly interpolates among them. The other option is `THREE.NearestFilter`, which uses the value of the closest texels.

`minFilter` -- How the texture is sampled when a texel covers less than one pixel. The default is `THREE.LinearMipMapLinearFilter`, which uses mipmapping and a trilinear filter. Other choices are

`THREE.NearestFilter`, `THREE.NearestMipMapNearestFilter`, `THREE.NearestMipMapLinearFilter`, `THREE.LinearFilter`, and `THREE.LinearMipMapNearestFilter`. These vary whether the nearest texel or nearest four texels are retrieved on the nearest mipmap or nearest two mipmaps. Interpolation occurs among the samples retrieved.

`anisotropy` -- The number of samples taken along the axis through the pixel that has the highest density of texels. By default, this value is 1. A higher value gives a less blurry result than a basic mipmap, at the cost of more texture samples being used. Use `renderer.getMaxAnisotropy()` to find the maximum valid anisotropy value for the GPU; this value is usually a power of 2. This creates a texture from compressed data. This is mostly used in `ImageUtils.loadCompressedTexture`.

Свойства

Общие свойства смотрите в описании базового класса [Texture](#).

[.flipy](#)

Значение по умолчанию равно `false`. Flipping textures does not work for compressed textures.

[.generateMipmaps](#)

Значение по умолчанию равно `false`. Mipmaps can't be generated for compressed textures.

Методы

Общие методы смотрите в описании базового класса [Texture](#).

Исходники

[CompressedTexture.js в этом справочнике](#)

[CompressedTexture.js на github.com](#)

[Texture](#) ↗

CubeTexture

Создает кубическую текстуру, состоящую из шести изображений.

Пример

```
var loader = new THREE.CubeTextureLoader();
loader.setPath( 'textures/cube/pisa/' );

var textureCube = loader.load( [
    'px.png', 'nx.png',
    'py.png', 'ny.png',
    'pz.png', 'nz.png'
] );

var material = new THREE.MeshBasicMaterial( { color: 0xffffffff, envM
```

Конструктор

```
CubeTexture( images, mapping, wrapS, wrapT, magFilter, minFilter, f
```

По функциональности и применению CubeTexture практически равна [Texture](#). Отличие состоит только в том, что изображения представляют собой массив из 6 изображений, а не одно изображение, и and the mapping options are [page:Textures THREE.CubeReflectionMapping (default) or [page:Textures THREE.CubeRefractionMapping.

Свойства

Общие свойства смотрите в описании базового класса [Texture](#).

Методы

Общие методы смотрите в описании базового класса [Texture](#).

Исходники

[CubeTexture.js в этом справочнике](#)

[CubeTexture.js на github.com](#)

[Texture](#) →

DataTexture

Создает текстуру непосредственно из необработанных данных (raw data), ширины и высоты.

Конструктор

```
DataTexture( data, width, height, format, type, mapping, wrapS, wrapT )
```

The data argument must be an ArrayBuffer or a typed array view. Further parameters correspond to the properties inherited from [page:Texture], where both magFilter and minFilter default to THREE.NearestFilter. The properties flipY and generateMipmaps are initially set to false.

The interpretation of the data depends on type and format: If the type is THREE.UnsignedByteType, a Uint8Array will be useful for addressing the texel data. If the format is THREE.RGBAFormat, data needs four values for one texel; Red, Green, Blue and Alpha (typically the opacity).

Similarly, THREE.RGBFormat specifies a format where only three values are used for each texel.

For the packed types, THREE.UnsignedShort4444Type, THREE.UnsignedShort5551Type or THREE.UnsignedShort565Type, all color components of one texel can be addressed as bitfields within an integer element of a Uint16Array.

In order to use the types THREE.FloatType and THREE.HalfFloatType, the WebGL implementation must support the respective extensions OES_texture_float and OES_texture_half_float. In order to use THREE.LinearFilter for component-wise, bilinear interpolation of the texels based on these types, the WebGL extensions OES_texture_float_linear or OES_texture_half_float_linear must also be present.

Свойства

```
[property:Image image]
```

Overridden with a record type holding data, width and height.

Методы

Исходники

[DataTexture.js в этом справочнике](#)

[DataTexture.js на github.com](#)

[Texture ↗](#)

DepthTexture

Creates a texture for use as a Depth Texture. Require support for the [WEBGL_depth_texture](#) extension. According to [WebGLStats](#), as of February 2016 around 85% of WebGL enabled devices support this.

Пример

```
[example:webgl_depth_texture depth / texture]
```

Конструктор

```
DepthTexture( [page:Number width, [page:Number height, [page:Consta
[page:Number width -- width of the texture.
[page:Number height -- height of the texture.
[page:Constant type -- Default is [page:Textures
    THREE.UnsignedShortType. See [page:Textures type constants for
    other choices.
[page:Constant mapping -- See [page:Textures type constants for
    details.
[page:Constant wrapS -- The default is [page:Textures
    THREE.ClampToEdgeWrapping. See [page:Textures wrap mode
    constants for other choices.
[page:Constant wrapT -- The default is [page:Textures
    THREE.ClampToEdgeWrapping. See [page:Textures wrap mode
    constants for other choices.
[page:Constant magFilter -- How the texture is sampled when a texel
    covers more than one pixel. The default is [page:Textures
    THREE.NearestFilter. See [page:Textures magnification filter
    constants for other choices.
[page:Constant minFilter -- How the texture is sampled when a texel
    covers less than one pixel. The default is [page:Textures
    THREE.NearestFilter. See [page:Textures minification filter
```

constants for other choices.

[page:Number anisotropy -- The number of samples taken along the axis through the pixel that has the highest density of texels. By default, this value is 1. A higher value gives a less blurry result than a basic mipmap, at the cost of more texture samples being used. Use [page:WebGLrenderer.getMaxAnisotropy() to find the maximum valid anisotropy value for the GPU; this value is usually a power of 2.]

[page:Constant format -- must be either [page:Textures DepthFormat (default) or [page:Textures DepthStencilFormat. See [page:Textures format constants for details.]

Свойства

Общие методы смотрите в описании базового класса [Texture](#).

Свойства, приведенные ниже, также являются частью класса [Texture](#), но имеют здесь другие значения по умолчанию. - the following are also part of the texture class, but have different defaults here.

[page:Texture.format .format

Either [page:Textures DepthFormat (default) or [page:Textures DepthStencilFormat. See [page:Textures format constants for details.]

[page:Texture.type .type

Default is [page:Textures THREE.UnsignedShortType. See [page:Textures format constants for details.]

[page:Texture.magFilter .magFilter

How the texture is sampled when a texel covers more than one pixel. The default is [page:Textures THREE.NearestFilter. See [page:Textures magnification filter constants for other choices.]

[page:Texture.minFilter .minFilter

How the texture is sampled when a texel covers less than one pixel. The default is [page:Textures THREE.NearestFilter. See [page:Textures magnification filter constants for other choices.]

[page:Texture.flipY .flipY

Depth textures do not need to be flipped so this is ***false*** by default.

[page:Texture.generateMipmaps .generateMipmaps]

Depth textures do not use mipmaps.

Методы

Общие методы смотрите в описании базового класса [Texture](#).

Исходники

[DepthTexture.js в этом справочнике](#)

[DepthTexture.js на github.com](#)

Texture

Создает текстуру для применения к поверхности или в качестве карт отражения или преломления.

Пример

```
// load a texture, set wrap mode to repeat
// загрузка текстуры, установка режима оболочки для повторения
var texture = new THREE.TextureLoader().load( "textures/water.jpg"
texture.wrapS = THREE.RepeatWrapping;
texture.wrapT = THREE.RepeatWrapping;
texture.repeat.set( 4, 4 );
```

Конструктор

`Texture(image, mapping, wrapS, wrapT, magFilter, minFilter, format)`

Свойства

[.id](#)

Уникальное число для данного экземпляра текстуры (только для чтения).

.image

An Image object, typically created using the ImageUtils or [page:ImageLoader ImageLoader classes. The Image object can include an image (e.g., PNG, JPG, GIF, DDS), video (e.g., MP4, OGG/OGV), or set of six images for a cube map. To use video as a texture you need to have a playing HTML5 video element as a source for your texture image and continuously update this texture as long as video is playing.

.mapping

How the image is applied to the object. An object type of THREE.UVMapping is the default, where the U,V coordinates are used to apply the map, and a single texture is expected. The other types are THREE.CubeReflectionMapping, for cube maps used as a reflection map; THREE.CubeRefractionMapping, refraction mapping; and THREE.SphericalReflectionMapping, a spherical reflection map projection.

.wraps

The default is THREE.ClampToEdgeWrapping, where the edge is clamped to the outer edge texels. The other two choices are THREE.RepeatWrapping and THREE.MirroredRepeatWrapping.

.wrapT

The default is THREE.ClampToEdgeWrapping, where the edge is clamped to the outer edge texels. The other two choices are THREE.RepeatWrapping and THREE.MirroredRepeatWrapping.

Примечание: tiling of images in textures only functions if image dimensions are powers of two (2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, ...) in terms of pixels. Individual dimensions need not be equal, but each must be a power of two. Это ограничение от WebGL, а не Three.js.

.magFilter

How the texture is sampled when a texel covers more than one pixel.

The default is THREE.LinearFilter, which takes the four closest texels and bilinearly interpolates among them. The other option is THREE.NearestFilter, which uses the value of the closest texel.

.minFilter

How the texture is sampled when a texel covers less than one pixel.

The default is THREE.LinearMipMapLinearFilter, which uses mipmapping and a trilinear filter. Other choices are

THREE.NearestFilter, THREE.NearestMipMapNearestFilter, THREE.NearestMipMapLinearFilter, THREE.LinearFilter, and THREE.LinearMipMapNearestFilter. These vary whether the nearest texel or nearest four texels are retrieved on the nearest mipmap or nearest two mipmaps. Interpolation occurs among the samples retrieved.

.format

The default is THREE.RGBAFormat for the texture. Other formats are: THREE.AlphaFormat, THREE.RGBFormat,

THREE.LuminanceFormat, and THREE.LuminanceAlphaFormat.

There are also compressed texture formats, if the S3TC extension is supported: THREE.RGB_S3TC_DXT1_Format,

THREE.RGBA_S3TC_DXT1_Format,

THREE.RGBA_S3TC_DXT3_Format, and

THREE.RGBA_S3TC_DXT5_Format.

.type

The default is THREE.UnsignedByteType. Other valid types (as

WebGL allows) are THREE.ByteType, THREE.ShortType,

THREE.UnsignedShortType, THREE.IntType,

THREE.UnsignedIntType, THREE.FloatType,

THREE.UnsignedShort4444Type, THREE.UnsignedShort5551Type,

and THREE.UnsignedShort565Type.

.anisotropy

The number of samples taken along the axis through the pixel that has the highest density of texels. By default, this value is 1. A higher value gives a less blurry result than a basic mipmap, at the cost of

more texture samples being used. Use `renderer.getMaxAnisotropy()` to find the maximum valid anisotropy value for the GPU; this value is usually a power of 2.

[.needsUpdate](#)

If a texture is changed after creation, set this flag to true so that the texture is properly set up. Particularly important for setting the wrap mode.

[.repeat](#)

How many times the texture is repeated across the surface, in each direction U and V.

[.offset](#)

How much a single repetition of the texture is offset from the beginning, in each direction U and V. Typical range is 0.0 to 1.0.

[.name](#)

Given name of the texture, empty string by default.

[.generateMipmaps](#)

Whether to generate mipmaps (if possible) for a texture. True by default.

[.flipy](#)

True by default. Flips the image's Y axis to match the WebGL texture coordinate space.

[.mipmaps](#)

Array of user-specified mipmaps (optional).

[.unpackAlignment](#)

4 by default. Specifies the alignment requirements for the start of each pixel row in memory. The allowable values are 1 (byte-alignment), 2 (rows aligned to even-numbered bytes), 4 (word-alignment), and 8 (rows start on double-word boundaries). See

[glPixelStorei](#) for more information.

.premultiplyAlpha

False by default, which is the norm for PNG images. Set to true if the RGB values have been stored premultiplied by alpha.

.encoding

Set to THREE.LinearEncoding by default, but supports sRGB, RGBE, RGBM, RGBD, LogLuv and Gamma corrected encodings.

IMPORTANT: If this value is changed on a texture after the material has been used, it is necessary to trigger a Material.needsUpdate for this value to be realized in the shader.

.onUpdate

A callback function, called when the texture is updated (e.g., when needsUpdate has been set to true and then the texture is used).

Методы

В этом классе доступны методы [EventDispatcher](#).

.clone (texture)

Make copy of the texture. Note this is not a "deep copy", the image is shared.

.dispose ()

Call EventDispatcher.dispatchEvent with a 'dispose' event type.

.toJSON (meta)

meta -- объект, содержащий метаданные (дополнительный, необязательный параметр).

Convert the material to three.js JSON format.

.transformUV (uv)

Transform the uv based on the value of this texture's .repeat, .offset, .wrapS, .wrapT and .flipY properties.

Исходники

[Texture.js в этом справочнике](#)

[Texture.js на github.com](#)

[Texture →](#)

VideoTexture Creates a texture for use with a video texture.
This is almost the same as the base [page:Texture Texture class]

Пример

```
//assuming you have created a HTML video element with id="video"
var video = document.getElementById( 'video' );

var texture = new THREE.VideoTexture( video );
texture.minFilter = THREE.LinearFilter;
texture.magFilter = THREE.LinearFilter;
texture.format = THREE.RGBFormat;
```

[Другие примеры](#)

[WebGL / materials / video](#)

Конструктор

`VideoTexture(video, mapping, wrapS, wrapT, magFilter, minFilter, f)`

video -- видеоэлемент для использования в качестве текстуры.

mapping -- How the image is applied to the object. An object type of [page:Textures THREE.UVMapping]. See [page:Textures mapping constants for other choices].

wrapS -- The default is [page:Textures

THREE.ClampToEdgeWrapping]. See [page:Textures wrap mode constants for other choices].

wrapT -- The default is [page:Textures

THREE.ClampToEdgeWrapping]. See [page:Textures wrap mode constants for other choices].

magFilter -- способ выбора текстуры, если тексель занимает больше одного пикселя. Значением по умолчанию является THREE.LinearFilter. Другие возможные варианты смотрите в

[константах фильтров увеличения](#).

[`minFilter`](#) -- способ выбора текстуры, если тексель занимает менее одного пикселя. Значением по умолчанию является [`THREE.LinearMipMapLinearFilter`](#). Другие возможные варианты смотрите в [константах фильтров уменьшения](#).

[`format`](#) -- формат используемый в текстуре. Возможные варианты смотрите в [константах форматов текстур](#).

[`type`](#) -- значением по умолчанию является [`THREE.UnsignedByteType`](#). Другие возможные варианты смотрите в [константах типов текстур](#).

[`anisotropy`](#) -- The number of samples taken along the axis through the pixel that has the highest density of texels. По умолчанию это значение равно 1. A higher value gives a less blurry result than a basic mipmap, at the cost of more texture samples being used. Используйте метод [`renderer.getMaxAnisotropy\(\)`](#), чтобы найти максимальное допустимое значение анизотропии для графического процессора (GPU); это значение обычно является степенью двойки.

Свойства

Общие свойства смотрите в описании базового класса [Texture](#).

[`.needsUpdate`](#)

Это свойство не нужно устанавливать вручную, поскольку оно автоматически обрабатывается методом [`update`](#).

Методы

Общие методы смотрите в описании базового класса [Texture](#).

[`.update\(\)`](#)

Этот метод вызывается автоматически и устанавливает свойство [`needsUpdate`](#) как `true` каждый раз, когда доступен новый кадр (frame).

Исходники

[VideoTexture.js в этом справочнике](#)

[VideoTexture.js на github.com](#)

ПРИМЕРЫ

Контролы

OrbitControls

Орбитальный контрол (элемент управления) позволяет камере вращаться вокруг цели.

Для его использования, как и всех файлов в директории /examples, нужно будет отдельно включить этот файл в HTML-код.

Пример

```
var renderer = new THREE.WebGLRenderer();
renderer.setSize( window.innerWidth, window.innerHeight );
document.body.appendChild( renderer.domElement );

var scene = new THREE.Scene();

var camera = new THREE.PerspectiveCamera( 45, window.innerWidth /
var controls = new THREE.OrbitControls( camera );

// controls.update() must be called after any manual changes to the
// после любых ручных изменений состояния камеры должен быть вызван
camera.position.set( 0, 20, 100 );
controls.update();

function animate() {

    requestAnimationFrame( animate );

    // required if controls.enableDamping or controls.autoRotate are
    // если controls.enableDamping или controls.autoRotate установлен
    controls.update();

    renderer.render( scene, camera );
}
```

Другие примеры:

[misc / controls / orbit](#)

Конструктор

```
OrbitControls(  
    object - объект'" onmouseout="hide()">object, domElement )
```

`object - объект'" onmouseout="hide()">object` - камера, которой нужно управлять (обязательный аргумент).

`domElement` - элемент HTML, используемый для прослушивателей (приемников) событий (дополнительный, необязательный аргумент). По умолчанию это весь документ целиком, однако, если нужно чтобы элементы управления (контролы) работали от какого-либо определенного элемента (например, <canvas> - холст), укажите его в этом аргументе.

Статья о [DOM](#) в Википедии.

Свойства

[.autoRotate](#)

Для автоматического вращения вокруг цели установите значение данного свойства как `true`.

Обратите внимание, если это свойство включено, в цикле анимации нужно вызывать метод [update](#).

[.autoRotateSpeed](#)

Свойство определяет скорость вращения камеры вокруг цели, если свойство [autoRotate](#) установлено как `true`. Значение по умолчанию равно `2.0`, что равно 30 секундам на один оборот при 60 кадрах в секунду.

Обратите внимание, если свойство [autoRotate](#) включено, в цикле анимации нужно вызывать метод [update](#).

[.dampingFactor](#)

The damping inertia used if `.enableDamping` is set to `true`.

Обратите внимание, для работы этого свойства в цикле

анимации нужно вызывать метод [update](#).

`.domElement`

The `HTMLDOMELEMENT` used to listen for mouse / touch events. This must be passed in the constructor; changing it here will not set up new event listeners.

Default is the whole document.

`.enabled`

Whether or not the controls are enabled.

`.enableDamping`

Set to true to enable damping (inertia), which can be used to give a sense of weight to the controls. Default is false.

Note that if this is enabled, you must call `.update()` in your animation loop.

`.enableKeys`

Enable or disable the use of keyboard controls.

`.enablePan`

Enable or disable camera panning. Default is true.

`.enableRotate`

Enable or disable horizontal and vertical rotation of the camera.

Default is true.

Note that it is possible to disable a single axis by setting the min and max of the polar angle or azimuth angle to the same value, which will cause the vertical or horizontal rotation to be fixed at that value.

`.enableZoom`

Enable or disable zooming (dollying) of the camera.

`.keyPanSpeed`

How fast to pan the camera when the keyboard is used. Default is 7.0 pixels per keypress.

.keys

This object contains references to the keycodes for controlling camera panning. Default is the 4 arrow keys.

```
controls.keys = {  
    LEFT: 37, //left arrow  
    UP: 38, // up arrow  
    RIGHT: 39, // right arrow  
    BOTTOM: 40 // down arrow  
}
```

See this page for a full list of keycodes.

.maxAzimuthAngle

How far you can orbit horizontally, upper limit. Range is - Math.PI to Math.PI (or Infinity for no limit) and default is Infinity;

.maxDistance

How far you can dolly out (PerspectiveCamera only). Default is Infinity.

.maxPolarAngle

How far you can orbit vertically, upper limit. Range is 0 to Math.PI radians, and default is Math.PI.

.maxZoom

How far you can zoom out (OrthographicCamera only). Default is Infinity.

.minAzimuthAngle

How far you can orbit horizontally, lower limit. Range is - Math.PI to Math.PI (or - Infinity for no limit) and default is - Infinity;

.minDistance

How far you can dolly in (PerspectiveCamera only). Default is 0.

.minPolarAngle

How far you can orbit vertically, lower limit. Range is 0 to Math.PI radians, and default is 0.

.minZoom

How far you can zoom in (OrthographicCamera only). Default is 0.

.mouseButtons

This object contains references to the mouse buttons used for the controls.

```
controls.mouseButtons = {  
    ORBIT: THREE.MOUSE.LEFT,  
    ZOOM: THREE.MOUSE.MIDDLE,  
    PAN: THREE.MOUSE.RIGHT  
}
```

.object

The camera (or other object) that is being controlled.

.position0

Used internally by the .saveState and .reset methods.

.rotateSpeed

Speed of rotation. Default is 1.

.target0

Used internally by the .saveState and .reset methods.

.target

The focus point of the controls, the .object orbits around this. It can be updated manually at any point to change the focus of the controls.

.zoom0

Used internally by the .saveState and .reset methods.

.zoomSpeed

Speed of zooming / dollying. Default is 1.

Методы

.dispose()

Remove all the event listeners.

```
.getAzimuthalAngle( )
```

Get the current horizontal rotation, in radians.

```
.getPolarAngle( )
```

Get the current vertical rotation, in radians.

```
.reset( )
```

Reset the controls to their state from either the last time the .saveState was called, or the initial state.

```
.saveState( )
```

Save the current state of the controls. This can later be recovered with .reset.

```
.update( )
```

Update the controls. Must be called after any manual changes to the camera's transform, or in the update loop if .autoRotate or .enableDamping are set.

Исходники

[OrbitControls.js на github.com](#)

Геометрические элементы

ConvexBufferGeometry

[BufferGeometry →](#)

ConvexBufferGeometry can be used to generate a convex hull for a given array of 3D points. The average time complexity for this task is considered to be $O(n \log(n))$.

Пример

```
var geometry = new THREE.ConvexBufferGeometry( points );
var material = new THREE.MeshBasicMaterial( {color: 0x00ff00} );
var mesh = new THREE.Mesh( geometry, material );
scene.add( mesh );
```

Другие примеры:

[geometry / convex](#)

Конструктор

`ConvexBufferGeometry(points)`

points — массив [трехмерных векторов](#), that the resulting convex hull will contain.

Исходники

[ConvexGeometry.js на github.com](#)

[Geometry →](#)

ConvexGeometry

ConvexGeometry can be used to generate a convex hull for a given array of 3D points. The average time complexity for this task is considered to be $O(n \log(n))$.

Пример

```
var geometry = new THREE.ConvexGeometry( points );
var material = new THREE.MeshBasicMaterial( {color: 0x00ff00} );
var mesh = new THREE.Mesh( geometry, material );
scene.add( mesh );
```

Другие примеры:

[geometry / convex](#)

Конструктор

`ConvexGeometry(points)`

`points` — Array of [page:Vector3 Vector3s that the resulting convex hull will contain.

Исходники

[ConvexGeometry.js на github.com](#)

[BufferGeometry →](#)

DecalGeometry

DecalGeometry can be used to create a decal mesh that serves different kinds of purposes e.g. adding unique details to models, performing dynamic visual environmental changes or covering seams.

Пример

```
var geometry = new THREE.DecalGeometry( mesh, position, orientation );
var material = new THREE.MeshBasicMaterial( { color: 0x00ff00 } );
var mesh = new THREE.Mesh( geometry, material );
scene.add( mesh );
```

Другие примеры:

[decals](#)

Конструктор

`DecalGeometry(mesh, position, orientation, size)`

`mesh` — Any mesh object.

`position` — Position of the decal projector.

`orientation` — Orientation of the decal projector.

`size` — Size of the decal projector.

Исходники

[DecalGeometry.js на github.com](#)

Загрузчики

BabylonLoader

Загрузчик для загрузки ресурсов в формате **.babylon**. Файловый формат **babylon** используется JavaScript'овским фреймворком [Babylon.js](#).

Пример

```
// instantiate a loader (создаем экземпляр загрузчика)
var loader = new THREE.BabylonLoader();

// load a Babylon resource (загружаем Babylon ресурс)
loader.load(
    // resource URL (URL-адрес ресурса)
    'models/babylon/skull.babylon',
    // Function when resource is loaded (функция при загрузке р
    function ( object ) {
        scene.add( object );
    },
    // Function called when download progresses
    // функция, вызываемая в процессе загрузки
    function ( xhr ) {
        console.log( (xhr.loaded / xhr.total * 100) + '% lo
    },
    // Function called when download errors
    // функция, вызываемая при ошибках загрузки
    function ( xhr ) {
        console.log( 'An error happened' );
    }
);
```

Другие примеры:

[webgl loader babylon](#)

Конструктор

```
BabylonLoader(
    manager - менеджер, управляющий'" onmouseout="hide()">manager )
```

[manager - менеджер, управляющий'" onmouseout="hide\(\)">manager](#)

— The [page:LoadingManager loadingManager for the loader to use. Default is [page:LoadingManager THREE.DefaultLoadingManager.

Создает новый BabylonLoader.

Свойства

Методы

[.load\(\[page:String url, \[page:Function onLoad, \[page:Function onPr](#)

[- единообразный локатор \(определитель местонахождения\) ресурса. ' \)" onmouseout="hide\(\)">url](#) — строка, содержащая путь/URL-адрес babylon-файла (обязательный аргумент).

[page:function onLoad — Will be called when load completes. The argument will be the loaded [page:Object3D.

[page:function onProgress — Will be called while load progresses. The argument will be the XMLHttpRequest instance, which contains .

[page:Integer total and .[page:Integer loaded bytes.

[page:function onError — Will be called when load errors.

Begin loading from url and call onLoad with the parsed response content.

[\[method:Object3D parse\(\[page:Object json \)](#)

[page:Object json — The **JSON** structure to parse.

Parse a **JSON** structure and return an [page:Object3D object or a [page:Scene scene.

Found objects are converted to [page:Mesh with a [page:BufferGeometry and a default [page:MeshPhongMaterial.

Lights are parsed accordingly.

Исходники

[BabylonLoader.js на github.com](#)

[page:Loader →

GLTFLoader

A loader for loading glTF 2.0 resource.

glTF (GL Transmission Format) is an open format specification for efficient delivery and loading of 3D content. Assets may be provided either in JSON (.gltf) or binary (.glb) format. External files store textures (.jpg, .png, ...) and additional binary data (.bin). A glTF asset may deliver one or more scenes, including meshes, materials, textures, skins, skeletons, morph targets, animations, lights, and/or cameras.

Расширения

GLTFLoader supports the following glTF extensions:

KHR_materials_pbrSpecularGlossiness
KHR_lights (experimental)

Example

```
// Instantiate a loader
var loader = new THREE.GLTFLoader();

// Load a glTF resource
loader.load(
    // resource URL
    'models/gltf/duck/duck.gltf',
    // called when the resource is loaded
    function ( gltf ) {

        scene.add( gltf.scene );

        gltf.animations; // Array
        gltf.scene; // THREE.Scene
        gltf.scenes; // Array
        gltf.cameras; // Array

    },
    // called when loading is in progresses
    function ( xhr ) {

        console.log( ( xhr.loaded / xhr.total * 100

    },
    // called when loading has errors
```

```
        function ( error ) {
            console.log( 'An error happened' );
        }
    );
}
```

[Другие примеры:](#)

[webgl / loader / gltf](#)

Совместимость с браузером

GLTFLoader relies on ES6 Promises, which are not supported in IE11. To use the loader in IE11, you must include a polyfill providing a Promise replacement.

Конструктор

`GLTFLoader(manager)`

manager — The loadingManager for the loader to use. Default is THREE.DefaultLoadingManager.

Создает новый GLTFLoader.

Свойства

Методы

.load (url, onLoad, onProgress, onError) url — A string containing the path/URL of the .gltf or .glb file. onLoad — A function to be called after the loading is successfully completed. The function receives the loaded JSON response returned from parse. onProgress — (optional) A function to be called while the loading is in progress. The argument will be the XMLHttpRequest instance, that contains .total and .loaded bytes. onError — (optional) A function to be called if an error occurs during loading. The function receives error as an argument. Begin loading from url and call the callback function with the parsed response content. # .setPath (path) path — Base path for loading additional resources e.g. textures and .bin data. Set the base path for additional resources. # .setCrossOrigin (value) value — The crossOrigin string to implement CORS for loading the url from a different domain that allows CORS. # .parse (data, path,

onLoad, onError) data — glTF asset to parse, as an ArrayBuffer or JSON string. path — The base path from which to find subsequent glTF resources such as textures and .bin data files. onLoad — A function to be called when parse completes. onError — (optional) A function to be called if an error occurs during parsing. The function receives error as an argument. Parse a glTF-based ArrayBuffer or JSON String and fire onLoad callback when complete. The argument to onLoad will be an object that contains loaded parts: .scene, .scenes, .cameras, and .animations.

Исходники

[GLTFLoader.js на github.com](#)

MTLLoader

A loader for loading an **.mtl** resource, used internally by [page:OBJMTLLoader and [page:UTF8Loader. A loader for loading an .mtl resource, used internally by OBJLoader and UTF8Loader. The Material Template Library format (MTL) or .MTL File Format is a companion file format to .OBJ that describes surface shading (material) properties of objects within one or more .OBJ files.

Конструктор

MTLLoader([page:LoadingManager loadingManager])

[page:LoadingManager loadingManager — LoadingManager to use. Defaults to [page:DefaultLoadingManager DefaultLoadingManager Creates a new [name].

Методы

[method:null load([page:String url, [page:Function onLoad, [pag

[page:String url — required

[page:Function onLoad — Will be called when load completes. The argument will be the loaded [page:MTLLoaderMaterialCreator

`MTLLoader.MaterialCreator` instance.

[page:Function `onProgress` — Will be called while load progresses. The argument will be the XMLHttpRequest instance, which contains .

[page:Integer `total` and .[page:Integer `loaded` bytes.

[page:Function `onError` — Will be called when load errors.

Begin loading from url and return the loaded material.

[method:null `setPath([page:String path)`

[page:String `path` — required

Set base path for resolving references. If set this path will be prepended to each loaded and found reference.

[method:null `setTexturePath([page:String path)`

[page:String `path` — required

Set base path for resolving texture references. If set this path will be prepended found texture reference. If not set and `setPath` is, it will be used as texture base path.

[method:null `setCrossOrigin([page:boolean useCrossOrigin)`

[page:boolean `useCrossOrigin` — required

Set to true if you need to load textures from a different origin.

[method:null `setMaterialOptions([page:Object options)`

[page:Object `options` — required

- `side`: Which side to apply the material. THREE.FrontSide (default), THREE.BackSide, THREE.DoubleSide
- `wrap`: What type of wrapping to apply for textures. THREE.RepeatWrapping (default), THREE.ClampToEdgeWrapping, THREE.MirroredRepeatWrapping
- `normalizeRGB`: RGBs need to be normalized to 0-1 from 0-255.
Default: false, assumed to be already normalized
- `ignoreZeroRGBs`: Ignore values of RGBs (Ka,Kd,Ks) that are all 0's.
Default: false

Set of options on how to construct the materials

[method:MTLLoaderMaterialCreator parse([page:String text)

[page:String text — The textual *mtl* structure to parse.

Parse a *mtl* text structure and return a [page:MTLLoaderMaterialCreator instance.

Исходники

[MTLLoader.js на github.com](#)

OBJLoader

A loader for loading an *.obj* resource.

Пример

```
// instantiate a loader
var loader = new THREE.OBJLoader();

// load a resource
loader.load(
    // resource URL
    'models/monster.obj',
    // Function when resource is loaded
    function ( object ) {
        scene.add( object )
    }
);
```

[example:webgl_loader_obj]

Конструктор

OBJLoader([page:LoadingManager manager)

[page:LoadingManager manager — The [page:LoadingManager loadingManager for the loader to use. Default is [page:LoadingManager THREE.DefaultLoadingManager.

Creates a new [name.

[Свойства](#)

[Методы](#)

[method:null load([page:String url, [page:Function onLoad, [pag

[page:String url — required

[page:Function onLoad — Will be called when load completes. The argument will be the loaded [page:Object3D].

[page:Function onProgress — Will be called while load progresses. The argument will be the XMLHttpRequest instance, which contains .

[page:Integer total and .[page:Integer loaded bytes.

[page:Function onError — Will be called when load errors.

Begin loading from url and call onLoad with the parsed response content.

[method:Object3D parse([page:String text)

[page:String text — The textual **obj** structure to parse.

Returns an [page:Object3D. It contains the parsed meshes as [page:Mesh and lines as [page:LineSegments.

All geometry is created as [page:BufferGeometry. Default materials are created as [page:MeshPhongMaterial.

If an **obj** object or group uses multiple materials while declaring faces geometry groups and an array of materials are used.

[Исходники](#)

[OBJLoader.js на github.com](#)

OBJLoader2

A loader for loading a **.obj** resource.

The OBJ file format is a simple data-format that represents 3D geometry in a human readable format as, the position of each vertex, the UV position of each texture coordinate vertex, vertex normals, and the faces that make each polygon defined as a list of vertices, and texture vertices.

[Пример](#)

```
// instantiate the loader
var loader = new THREE.OBJLoader2();

// function called on successful load
var intergrateIntoScene = function ( object ) {
    scene.add( object );
};

// load a resource from provided URL
loader.load( 'obj/female02/female02.obj', intergrat
```

[example:webgl_loader_obj2

Конструктор

OBJLoader2([page:LoadingManager manager])

[page:LoadingManager manager — The [page:LoadingManager loadingManager for the loader to use. Default is [page:LoadingManager THREE.DefaultLoadingManager.

Use [name to load OBJ data from files or to parse OBJ data from arraybuffer or text.

Свойства

Методы

[method:null load([page:String url, [page:Function onLoad, [pag

[page:String url — URL of the file to load

[page:Function onLoad — Called after loading was successfully completed. The argument will be the loaded [page:Object3D.

[page:Function onProgress — Called to report progress of loading. The argument will be the XMLHttpRequest instance, which contains .

[page:Integer total and .[page:Integer loaded bytes.

[page:Function onError Called after an error occurred during loading.

[page:boolean useArrayBuffer — Set this to false to force string based parsing

Use this convenient method to load an OBJ file at the given URL. Per default the fileLoader uses an arraybuffer

[method:Object3D parse([page:ArrayBuffer arrayBuffer)

[page:ArrayBuffer arrayBuffer — OBJ data as Uint8Array

Default parse function: Parses OBJ file content stored in arrayBuffer and returns the [page:Object3D sceneGraphNode.

[method:Object3D parseText([page:String test)

[page:String text — OBJ data as string

Legacy parse function: Parses OBJ file content stored in string and returns the [page:Object3D sceneGraphNode.

[method:null setMaterials (Array of [page:Material materials)

Array of [page:Material materials — Array of [page:Material Materials from MTLLoader

Set materials loaded by MTLLoader or any other supplier of an Array of [page:Material Materials.

[method:null setPath ([page:String path)

[page:String path — The basePath

Base path to use.

[method:null setSceneGraphNode ([page:Object3D sceneGraphNode)

[page:Object3D sceneGraphNode — Scenegraph object where meshes will be attached

Set the node where the loaded objects will be attached.

[method:null setDebug([page:Boolean parserDebug, [page:Boolean meshCreatorDebug)

[page:Boolean parserDebug — Internal Parser will produce debug output

[page:Boolean meshCreatorDebug — Internal MeshCreator will produce debug output

Allows to set debug mode for the parser and the meshCreator.

[Исходники](#)

[OBJLoader2.js на github.com](#)

LoaderSupport

Supporting classes for file loaders and web worker based loaders.

Sub-Classes

```
LoaderSupport.Builder  
LoaderSupport.LoadedMeshUserOverride  
LoaderSupport.WorkerSupport  
LoaderSupport.WorkerRunnerRefImpl  
LoaderSupport.WorkerDirector  
LoaderSupport.PrepData  
LoaderSupport.LoaderBase  
LoaderSupport.Callbacks  
LoaderSupport.Validator  
LoaderSupport.ConsoleLogger
```

Пример

webgl_loader_obj2_meshspray - Example using
LoaderSupport.LoaderWorkerDirector and
LoaderSupport.LoaderWorkerSupport.

Builder

Конструктор

Builder() Builds one or many Mesh from one raw set of Arraybuffers, materialGroup descriptions and further parameters. Supports vertex, vertexColor, normal, uv and index buffers.

Методы

```
# .setMaterials ( Array of materials ) Array of materials - Array of  
Materials Set materials loaded by any supplier of an Array of Materials. #  
.processPayload ( Object payload ) payload - Raw Mesh or Material  
descriptions. Delegates processing of the payload (mesh building or  
material update) to the corresponding functions (BW-compatibility). #  
.buildMeshes ( Object meshPayload ) meshPayload - Raw mesh  
description (buffers, params, materials) used to build one to many
```

meshes. Builds one or multiple meshes from the data described in the payload (buffers, params, material info). # .updateMaterials (Object materialPayload) materialPayload - Material update instructions Updates the materials with contained material objects (sync) or from alteration instructions (async). # .getMaterialsJSON () Returns the mapping object of material name and corresponding jsonified material. # .getMaterials () Returns the mapping object of material name and corresponding material.

LoadedMeshUserOverride

Конструктор

LoadedMeshUserOverride(disregardMesh, bufferGeometry)

disregardMesh - Tell implementation to completely disregard this mesh
alteredMesh - Tell implementation that mesh(es) have been altered or added Object to return by callback onMeshAlter. Used to disregard a certain mesh or to return one to many meshes.

Методы

.addMesh (mesh) mesh - Mesh Add a mesh created within callback. # .isDisregardMesh () Answers if mesh shall be disregarded completely. # .providesAlteredMeshes () Answers if new mesh(es) were created.

WorkerSupport

Конструктор

WorkerSupport(logger) logger - logger to be used This class provides means to transform existing parser code into a web worker. It defines a simple communication protocol which allows to configure the worker and receive raw mesh data during execution.

Методы

.validate (functionCodeBuilder, Array of libLocations, libPath, runnerImpl) functionCodeBuilder - Function that is invoked with funcBuildObject and funcBuildSingleton that allows stringification of objects and singletons. Array of libLocations - URL of libraries that shall be added to worker code relative to libPath. libPath - Base path used for loading libraries. runnerImpl - The default worker parser wrapper

implementation (communication and execution). An extended class could be passed here. Validate the status of worker code and the derived worker. # .setTerminateRequested (terminateRequested)
terminateRequested - True or false. Request termination of worker once parser is finished. # .setCallbacks (builder, onLoad) builder - The builder function. Default is LoaderSupport.Builder. onLoad - The function that is called when parsing is complete. Specify functions that should be build when new raw mesh data becomes available and when the parser is finished. # .run (payload) payload - Raw mesh description (buffers, params, materials) used to build one to many meshes. Runs the parser with the provided configuration.

[WorkerRunnerRefImpl](#)

[Конструктор](#)

`WorkerRunnerRefImpl()` Default implementation of the WorkerRunner responsible for creation and configuration of the parser within the worker.

[Методы](#)

.applyProperties (parser, params) parser - The parser instance
params - The parameter object Applies values from parameter object via set functions or via direct assignment. # .run (payload) payload - Raw mesh description (buffers, params, materials) used to build one to many meshes. Configures the Parser implementation according the supplied configuration object.

[WorkerDirector](#)

[Конструктор](#)

`WorkerDirector(classDef, logger)` classDef - Class definition to be used for construction
logger - logger to be used Orchestrate loading of multiple OBJ files/data from an instruction queue with a configurable amount of workers (1-16). - Workflow:
- prepareWorkers - enqueueForRun - processQueue - tearDown

[Методы](#)

.prepareWorkers (globalCallbacks, maxQueueSize, maxWebWorkers)
globalCallbacks - Register global callbacks used by all web workers

maxQueueSize - Set the maximum size of the instruction queue (1-1024)
maxWebWorkers - Set the maximum amount of workers (1-16) Create or destroy workers according limits. Set the name and register callbacks for dynamically created web workers. # .enqueueForRun (runParams)
runParams Store run instructions in internal instructionQueue. # .processQueue () Process the instructionQueue until it is depleted. # .tearDown (callbackOnFinishedProcessing)
callbackOnFinishedProcessing - Function called once all workers finished processing. Terminate all workers. # .getMaxQueueSize () Returns the maximum length of the instruction queue. # .getMaxWebWorkers () Returns the maximum number of workers. # .isRunning () Returns if any workers are running. # .setCrossOrigin (crossOrigin) crossOrigin - CORS value Sets the CORS string to be used.

PrepData

Конструктор

PrepData(modelName) modelName - Overall name of the model Configuration instructions to be used by run method.

Методы

.setStreamMeshesTo (streamMeshesTo) streamMeshesTo - Object already attached to scenegraph where new meshes will be attached to Set the node where the loaded objects will be attached directly. # .setMaterialPerSmoothingGroup (materialPerSmoothingGroup) materialPerSmoothingGroup Tells whether a material shall be created per smoothing group. # .setUseIndices (useIndices) useIndices - Default is false Instructs loaders to create indexed BufferGeometry. # .setDisregardNormals (disregardNormals) disregardNormals Tells whether normals should be completely disregarded and regenerated. # .getCallbacks () Returns all callbacks as LoaderSupport.Callbacks. # .setCrossOrigin (crossOrigin) crossOrigin - CORS value Sets the CORS string to be used. # .addResource (resource) resource Add a resource description. # .setUseAsync (useAsync) useAsync If true uses async loading with worker, if false loads data synchronously.

LoaderBase

Конструктор

LoaderBase(manager, logger) manager - The loadingManager for the loader to use. Default is THREE.DefaultLoadingManager. logger - logger to be used Base class to be used by loaders.

Методы

```
# .getLogger () Returns LoaderSupport.ConsoleLogger. #
.set modelName ( modelName ) modelName Set the name of the model.
# .setPath ( path ) path - URL The URL of the base path. #
.setStreamMeshesTo ( streamMeshesTo ) streamMeshesTo - Object already attached to scenegraph where new meshes will be attached to Set the node where the loaded objects will be attached directly. #
.setMaterials ( Array of materials ) Array of materials - Array of Materials Set materials loaded by MTLLoader or any other supplier of an Array of Materials. # .setUseIndices ( useIndices ) useIndices Instructs loaders to create indexed BufferGeometry. # .setDisregardNormals ( disregardNormals ) disregardNormals Tells whether normals should be completely disregarded and regenerated. # .onProgress ( type, text, numericalValue ) type - The type of event text - Textual description of the event numericalValue - Numerical value describing the progress Announce feedback which is give to the registered LoaderSupport.Callbacks.
```

Callbacks

Конструктор

Callbacks() Callbacks utilized by loaders and builder.

Методы

```
# .setCallbackOnProgress ( callbackOnProgress ) callbackOnProgress - Callback function for described functionality Register callback function that is invoked by internal function "announceProgress" to print feedback.
# .setCallbackOnMeshAlter ( callbackOnMeshAlter )
callbackOnMeshAlter - Callback function for described functionality Register callback function that is called every time a mesh was loaded. Use LoadedMeshUserOverride for alteration instructions (geometry, material or disregard mesh). # .setCallbackOnLoad ( callbackOnLoad )
callbackOnLoad - Callback function for described functionality Register
```

callback function that is called once loading of the complete OBJ file is completed. # .setCallbackOnLoadMaterials (callbackOnLoadMaterials)
callbackOnLoadMaterials - Callback function for described functionality
Register callback function that is called when materials have been loaded.

Validator

Constructor

Validator() Validation functions.

Методы

.isValid (input) input - Can be anything If given input is null or undefined, false is returned otherwise true. # .verifyInput (input, defaultValue) input - Can be anything defaultValue - Can be anything If given input is null or undefined, the defaultValue is returned otherwise the given input.

ConsoleLogger

Конструктор

ConsoleLogger(enabled, debug) enabled - Tell if logger is enabled.
debug - Toggle debug logging. Logging wrapper for console.

Методы

.setDebug (debug) debug - True or False Enable or disable debug logging. # .isDebug () Returns if is enabled and debug. # .setEnabled (enabled) enabled - True or False Enable or disable info, debug and time logging. # .isEnabled () Returns if is enabled. # .logDebug (message) message - Message to log Log a debug message if enabled and debug is set. # .logInfo (message) message - Message to log Log an info message if enabled. # .logWarn (message) message - Message to log Log a warn message (always). # .logError (message) message - Message to log Log an error message (always). # .logTimeStart (id) id - Time identification Start time measurement with provided id. # .logTimeEnd (id) id - Time identification Stop time measurement started with provided id.

Исходники

[LoaderSupport.js на github.com](#)

PCDLoader

A loader for **PCD** files. Loads ascii and binary. Compressed binary files are not supported.

Пример

```
// instantiate a loader
var loader = new THREE.PCDLoader();

// load a resource
loader.load(
    // resource URL
    'pointcloud.pcd' ,
    // Function when resource is loaded
    function ( mesh ) {
        scene.add( mesh );
    }
);
```

[example:webgl_loader_pcd

Конструктор

PCDLoader([page:LoadingManager manager])

[page:LoadingManager manager — The [page:LoadingManager loadingManager for the loader to use. Default is [page:LoadingManager THREE.DefaultLoadingManager.

Creates a new [name].

Свойства

[page:Boolean littleEndian

Default value is true.

Методы

[method:null load([page:String url, [page:Function onLoad, [page:Function onProgress, [page:Function onError

[page:String url — required

[page:Function onLoad — Will be called when load completes. The argument will be the loaded [page:Object3D].

[page:Function onProgress — Will be called while load progresses. The argument will be the XMLHttpRequest instance, which contains .

[page:Integer total and .[page:Integer loaded bytes.

[page:Function onError — Will be called when load errors.

Begin loading from url and call onLoad with the parsed response content.

[method:Object3D parse([page:Arraybuffer data,[page:String url

[page:Arraybuffer data — The binary structure to parse.

[page:String url — The file name or file url.

Parse an **pcd** binary structure and return an [page:Object3D].

The object is converted to [page:Points with a [page:BufferGeometry and a [page:PointsMaterial].

Исходники

[PCDLoader.js на github.com](#)

PDBLoader

A loader for loading a **.pdb** resource.

The [Protein Data Bank file format](#) is a textual file format describing the three-dimensional structures of molecules.

Пример

```
// instantiate a loader
var loader = new THREE.PDBLoader();

// load a PDB resource
loader.load(
    // resource URL
    'models/molecules/caffeine.pdb',
    // Function when resource is loaded
    function ( geometryAtoms, geometryBonds, js
```

```
        console.log( 'This molecule has ' +
    },
    // Function called when download progresses
    function ( xhr ) {
        console.log( (xhr.loaded / xhr.tota
    },
    // Function called when download errors
    function ( xhr ) {
        console.log( 'An error happened' );
    }
);
```

[example:webgl_loader_pdb

Конструктор

PDBLoader([page:LoadingManager manager])

[page:LoadingManager manager — The [page:LoadingManager loadingManager for the loader to use. Default is [page:LoadingManager THREE.DefaultLoadingManager.

Creates a new PDBLoader.

Свойства

Методы

[method:null load([page:String url, [page:Function onLoad, [pag

[page:String url — required. URL to the **.pdb** file

[page:Function onLoad — Will be called when load completes. The arguments will be an [page:BufferGeometry geometryAtoms, [page:BufferGeometry geometryBonds and the [page:Object JSON structure.

[page:Function onProgress — Will be called while load progresses. The argument will be the XMLHttpRequest instance, which contains .

[page:Integer total and .[page:Integer loaded bytes.

[page:Function onError — Will be called when load errors.

Begin loading from url and call onLoad with the parsed response content.

[method:Object parsePDB([page:String text)

[page:String text — The textual **pdb** structure to parse.
Parse a **pdb** text and return a **JSON** structure.

[method:null **createModel([page:Object json, [page:Function cal**

[page:Object json — The **(JSON) pdb** structure to parse.
[page:Function callback — Will be called when parse completes, with
three arguments: [page:BufferGeometry geometryAtoms,
[page:BufferGeometry geometryBonds and the original [page:Object
json.

Parse a **(JSON) pdb** structure and return two [page:BufferGeometry: one
for atoms, one for bonds.

Исходники

[PDBLoader.js на github.com](#)

SVGLoader

Загрузчик для загрузки ресурса в формате **.svg**.

Конструктор

`SVGLoader(manager)`

manager — The [page:LoadingManager loadingManager for the
loader to use. Default is [page:LoadingManager
THREE.DefaultLoadingManager.

Создает новый SVGLoader.

Свойства

Методы

`.load(url, onLoad, onProgress, onError)`

url — обязательный параметр

onLoad — функция, которая должна быть вызвана по
завершении загрузки. Её аргументом должен быть

загруженный [page:SVGDocument].

onProgress — функция, которая будет вызвана во время хода процесса загрузки. Аргументом этой функции должен быть экземпляр XMLHttpRequest, which contains .[page:Integer total and .[page:Integer loaded bytes].

onError — функция, которая будет вызвана при ошибках загрузки.

Begin loading from url and call onLoad with the response content.

Исходники

[SVGLoader.js на github.com](#)

TGALoader

Class for loading a **.tga** [page:DataTexture texture].

Пример

```
// instantiate a loader
var loader = new THREE.TGALoader();

// load a resource
var texture = loader.load(
    // resource URL
    'textures/crate_grey8.tga'
    // Function when resource is loaded
    function ( texture ) {
        console.log( 'Texture is loaded' );
    },
    // Function called when download progresses
    function ( xhr ) {
        console.log( (xhr.loaded / xhr.total) * 100 + '%' );
    },
    // Function called when download errors
    function ( xhr ) {
        console.log( 'An error happened' );
    }
);

var material = new THREE.MeshPhongMaterial( {
```

```
        color: 0xffffffff,  
        map: texture  
    } );
```

[example:webgl_materials_texture_tga

Конструктор

TGALoader([page:LoadingManager manager])

[page:LoadingManager manager — The [page:LoadingManager loadingManager for the loader to use. Default is [page:LoadingManager THREE.DefaultLoadingManager.

Creates a new TGALoader.

Методы

[method:DataTexture load([page:String url, [page:Function onLoad, [page:Function onProgress, [page:Function onError])]

[page:String url — required

[page:Function onLoad — Will be called when load completes. The argument will be the loaded [page:DataTexture.

[page:Function onProgress — Will be called while load progresses. The argument will be the XMLHttpRequest instance, which contains .

[page:Integer total and .[page:Integer loaded bytes.

[page:Function onError — Will be called when load errors.

Begin loading from url and pass the loaded [page:DataTexture texture to onLoad. The [page:DataTexture texture is also directly returned for immediate use (but may not be fully loaded).

Исходники

[TGALoader.js на github.com](#)

PRWMLoader

A loader for loading a .prwm resource. Packed Raw WebGL Model is an open-source binary file format for nD geometries specifically designed for

JavaScript and WebGL with a strong focus on fast parsing (from 1ms to 0.1ms in Chrome 59 on a MBP Late 2013). The parsing of PRWM file is especially fast when the endianness of the file is the same as the endianness of the client platform. More information on this [here](#).

Пример

```
// instantiate a loader
var loader = new THREE.PRWMLoader(); // load a
resource
loader.load( // resource URL 'models/nefertiti.le.prwm', // called
when resource is loaded
function ( bufferGeometry ) {
    var object = new
    THREE.Mesh( bufferGeometry, new THREE.MeshNormalMaterial() );
    scene.add( object );
}, // called when loading is in progresses
function ( xhr ) {
    console.log( ( xhr.loaded / xhr.total * 100 ) + '% loaded' );
}, // called when loading has errors
function ( error ) {
    console.log( 'An error happened' );
} );
webgl_loader_prwm
```

Конструктор

PRWMLoader(manager)
manager — The loadingManager for the loader to use. Default is THREE.DefaultLoadingManager. Creates a new PRWMLoader.

Свойства

Методы

.load (url, onLoad, onProgress, onError)
url — A string containing the path/URL of the .prwm file. Any * character in the URL will be automatically replaced by le or be depending on the platform endianness.
onLoad — (optional) A function to be called after the loading is successfully completed. The function receives the loaded BufferGeometry as an argument.
onProgress — (optional) A function to be called while the loading is in progress. The function receives a XMLHttpRequest instance, which contains total and loaded bytes.
onError — (optional) A function to be called if an error occurs during loading. The function receives error as an argument.
Begin loading from url and call onLoad with the parsed response content.
.parse (arrayBuffer)
arrayBuffer — ArrayBuffer containing the prwm data. Parse a prwm file passed as an ArrayBuffer and directly return an instance of

`BufferGeometry`.`PRWMLoader.isBigEndianPlatform()` Return true if the endianness of the platform is Big Endian, false otherwise.

Исходники

[PRWMLoader.js на github.com](#)

Additional notes

This loader is additionally available on npm as `three-prwm-loader`.

Объекты

LensFlare

[Mesh →](#)

Создает имитацию блика объектива камеры, зависящего от направления освещения. Creates a simulated lens flare that tracks a light.

Примечание: Для работы LensFlare у [WebGLRenderer](#) аргумент alpha должен быть установлен как true.

Пример

[example:webgl_lensesflares lensflares]

Конструктор

```
LensFlare( [page:Texture texture], [page:Float size], [page:Float d  
texture — THREE.Texture (optional)  
size — size in pixels (-1 = use texture.width)  
distance — (0-1) from light source (0 = at light source)  
blending — [page:Materials Blending Mode] - Defaults to  
THREE.NormalBlending  
color — The color of the lens flare.
```

Automatically adds a lens flare to the lensFlares array if a texture is set.

Свойства

[property:array lensFlares]

The array of flares as set by [page:LensFlare.add]

[property:Vector3 positionScreen]

The position of the lens flare on the screen.

[property:Function customUpdateCallback]

A custom update callback

Методы

[method:null add]([page:Texture texture], [page:Float size], [page

Adds a lens flare. See the constructor for details on the parameters.

[method:null updateLensFlares]()

Updates the lens flare based on the [page:LensFlare.positionScreen positionScreen] property.

[method:Lens Flare clone]()

Returns a clone of this LensFlare object and its descendants.

Исходники

[link:[https://github.com/mrdoob/three.js/blob/master/src/\[path\].js](https://github.com/mrdoob/three.js/blob/master/src/[path].js)
src/[path].js]

Экспортеры

GLTFExporter

An exporter for glTF 2.0.

glTF (GL Transmission Format) is an open format specification for efficient delivery and loading of 3D content. Assets may be provided either in JSON (.gltf) or binary (.glb) format. External files store textures (.jpg, .png, ...) and additional binary data (.bin). A glTF asset may deliver one or more scenes, including meshes, materials, textures, skins, skeletons, morph targets, animations, lights, and/or cameras.

Пример

```
// Instantiate a exporter var exporter = new THREE.GLTFExporter(  
defaultOptions ); // Parse the input and generate the glTF output  
exporter.parse( scene, function ( gltf ) { console.log( gltf );  
downloadJSON( gltf ); }, options ); misc_exporter_gltf
```

Конструктор

GLTFExporter() Creates a new GLTFExporter.

Методы

.parse (input, onCompleted, options) input — Scenes or objects to export. Valid options: Export scenes exporter.parse(scene1, ...) exporter.parse([scene1, scene2], ...) Export objects (It will create a new Scene to hold all the objects) exporter.parse(object1, ...) exporter.parse([object1, object2], ...) Mix scenes and objects (It will export the scenes as usual but it will create a new scene to hold all the single objects). exporter.parse([scene1, object1, object2, scene2], ...) onCompleted — Will be called when the export completes. The argument will be the generated glTF JSON or binary ArrayBuffer. options - Export options trs - bool. Export position, rotation and scale instead of matrix per node. Default is false onlyVisible - bool. Export only visible objects. Default is true. truncateDrawRange - bool. Export just the attributes within the drawRange, if defined, instead of exporting the whole array. Default is true. binary - bool. Export in binary (.glb) format, returning an ArrayBuffer.

Default is false. embedImages - bool. Export with images embedded into the glTF asset. Default is true. animations - Array. List of animations to be included in the export. Generates a .gltf (JSON) or .glb (binary) output from the input (Scene or Objects)

[Исходники](#)

[examples/js/exporters/GLTFExporter.js](#)

Плагины

CombinedCamera

[Camera →](#)

A general purpose camera, for setting FOV, Lens Focal Length, and switching between perspective and orthographic views easily. Use this only if you do not wish to manage both an Orthographic and Perspective Camera.

Примеры

```
[example:canvas_camera_orthographic2 camera / orthographic2
```

```
//Create combined camera - создаем комбинированную камеру  
camera = new THREE.CombinedCamera( window.innerWidth / 2, window.in
```

Конструктор

```
CombinedCamera(width, height, fov, near, far, orthoNear, orthoFar)
```

`width` — Camera frustum width.

`height` — Camera frustum height.

`fov` — Camera frustum vertical field of view in perspective view.

`near` — Camera frustum near plane in perspective view.

`far` — Camera frustum far plane in perspective view.

`orthoNear` — Camera frustum near plane in orthographic view.

`orthoFar` — Camera frustum far plane in orthographic view.

Creates a CombinedCamera. This initializes 2 cameras, an OrthographicCamera and a PerspectiveCamera. The default is the perspective Camera.

Свойства

```
.fov
```

Gets or sets the camera frustum vertical field of view in perspective view.

```
.left
```

Gets or sets the camera frustum left plane in orthographic view.

```
.right
```

Gets or sets the camera frustum right plane in orthographic view.

```
.top
```

Gets or sets the camera frustum top plane in orthographic view.

```
.bottom
```

Gets or sets the camera frustum bottom plane in orthographic view.

[.zoom](#)

Gets or sets the zoom factor of the camera.

[.near](#)

Gets camera frustum near plane.

[.far](#)

Gets camera frustum far plane.

[property:Matrix4 projectionMatrix]

This is the matrix which contains the projection.

[property:OrthographicCamera camera0]

Gets or sets the internal OrthographicCamera used as camera.

[property:PerspectiveCamera cameraP]

Gets or sets the internal PerspectiveCamera used as camera.

[property:boolean inOrthographicMode]

Gets whether the combinedCamera is in Orthographic Mode.

[property:boolean inPerspectiveMode]

Gets whether the combinedCamera is in Perspective Mode.

Методы

[method:null setFov([page:Number fov])

fov — Camera frustum vertical field of view in perspective view.

sets the camera frustum vertical field of view in perspective view.

[method:null setZoom([page:Number zoom])

zoom — The zoom factor.

Sets the zoomfactor.

[method:null setLens([page:number focalLength, [page:Number filmGau

focalLength — The focal length of a lens is defined as the distance from the optical center of a lens (or, the secondary principal point for a complex lens like a camera lens) to the focal point (sensor) when the lens is focused on an object at infinity.

filmGauge — the size of the frame in mm. (default is *35*)

Sets the fov based on lens data.

[method:null toFrontView()])

Sets the camera to view the front of the target.

[method:null toBackView()])

Sets the camera to view the back of the target.

```
[method:null toLeftView()]
```

Sets the camera to view the left of the target.

```
[method:null toRightView()]
```

Sets the camera to view the right of the target.

```
[method:null toTopView()]
```

Sets the camera to view the top.

```
[method:null toBottomView()]
```

Sets the camera to view the bottom.

```
[method:null setSize([page:Number width, [page:Number height])]
```

width — The width of the orthographic view.

height — The height of the orthographic view.

Sets the size of the orthographic view.

```
[method:null toOrthographic()]
```

Change the camera to orthographic view.

```
[method:null toPerspective()]
```

Change the camera to Perspective view.

```
[method:null updateProjectionMatrix()]
```

Updates the ProjectionMatrix.

Source

[CombinedCamera.js в этом справочнике](#)

[CombinedCamera.js на github.com](#)

Lut

Represents a lookup table for colormaps. It is used to determine the color values from a range of data values.

Пример

```
var lut = new THREE.Lut( "rainbow", 512 );
var data = [0, 10.1, 4.2, 3.4, 63, 28;
lut.setMax(63);
color = lut.getColor(10);
```

Конструктор

```
Lut( colormap, numberOfColors )
```

colormap — optional argument that sets a colormap from predefined

colormaps. Available colormaps are : "rainbow", "cooltowarm", "blackbody".

numberOfColors — optional argument that sets the number of colors used to represent the data array.

Свойства

```
[property:Float minV
```

The minimum value to be represented with the lookup table. Default is 0.

```
[property:Float maxV
```

The maximum value to be represented with the lookup table. Default is 1.

```
. [legend
```

The legend of the lookup table.

Методы

```
[method:null copy( [page:Lut lut] ) [page:Lut this
```

color — Lut to copy.

Copies given lut.

```
.setLegendOn [parameters
```

parameters — { layout: value, position: { x: value, y: value, z: value }, dimensions: { width: value, height: value } } layout — Horizontal or vertical layout. Default is vertical.

position — The position x,y,z of the legend.

dimensions — The dimensions (width and height) of the legend.

Sets this Lut with the legend on.

```
.setLegendOff
```

Sets this Lut with the legend off.

```
.setLegendLabels [parameters, callback
```

parameters — { fontsize: value, fontface: value, title: value, um: value, ticks: value, decimal: value, notation: value } fontsize — Font size to be used for labels.

fontface — Font type to be used for labels.

title — The title of the legend.

um — The unit of measurements of the legend.

ticks — The number of ticks to be displayed.

decimal — The number of decimals to be used for legend values.

notation — Legend notation: standard (default) or scientific.

callback — An optional callback to be used to format the legend

labels.

Sets the labels of the legend of this Lut.

```
[method:Lut setminV( [page:Float minV )
```

minV — The minimum value to be represented with the lookup table.

Sets this Lut with the minimum value to be represented.

```
[method:Lut setmaxV( [page:Float maxV )
```

maxV — The maximum value to be represented with the lookup table.

Sets this Lut with the maximum value to be represented.

```
[method:Lut changeNumberOfColors( [page:Float numberOfColors )
```

numberOfColors — The number of colors to be used to represent the data array.

Sets this Lut with the number of colors to be used.

```
[method:Lut changeColorMap( [page:Float colorMap )
```

colorMap — The name of the color map to be used to represent the data array.

Sets this Lut with the colormap to be used.

```
[method:Lut addColorMap( colorMapName, arrayOfColors )
```

Insert a new color map into the set of available color maps.

```
[method:Lut getColor( value ) [page:Lut this
```

value — the data value to be displayed as a color.

Returns a Three.Color.

Исходники

[\[link:https://github.com/mrdoob/three.js/blob/master/examples/js/math/\[path\].js\]](https://github.com/mrdoob/three.js/blob/master/examples/js/math/[path].js)

[Material →](#)

SpriteCanvasMaterial

Create a material that can draw custom sprites using a 2d canvas.

Конструктор

```
SpriteCanvasMaterial( [page:Object parameters )
```

parameters is an object that can be used to set up the default properties

color — the color of the sprite

program — the program used to draw the sprite

Свойства

```
[property:Color color]
```

The color of the sprite. The material will set up the color for the context before calling the material's program.

Методы

```
[method:null program([page:CanvasRenderingContext2D context, [page:  
context — The canvas context  
color — The color of the sprite
```

Define a program that will use the context to draw the sprite.

Исходники

```
[link:https://github.com/mrdoob/three.js/blob/master/examples/js/renderers/  
examples/js/renderers/CanvasRenderer.js
```

QuickHull

Face

Represents a section bounded by a specific amount of half-edges. The current implementation assumes that a face always consist of three edges.

Конструктор

[name()]

Свойства

[property:Vector3 normal]

The normal vector of the face. Default is a [page:Vector3 at (0, 0, 0)].

[property:Vector3 midpoint]

The midpoint or centroid of the face. Default is a [page:Vector3 at (0, 0, 0)].

[property:Float area]

The area of the face. Default is 0.

[property:Float constant]

Signed distance from face to the origin. Default is 0.

[property:VertexNode outside]

Reference to a vertex in a vertex list this face can see. Default is null.

[property:Integer mark]

Marks if a face is visible or deleted. Default is 'Visible'.

[property:HalfEdge edge

Reference to the base edge of a face. To retrieve all edges, you can use the 'next' reference of the current edge. Default is null.

Методы

[method:Face create([page:VertexNode a, [page:VertexNode b,

[page:VertexNode a - First vertex of the face.

[page:VertexNode b - Second vertex of the face.

[page:VertexNode c - Third vertex of the face.

Creates a face.

[method:HalfEdge getEdge([page:Integer i)

[page:Integer i - The index of the edge.

Returns an edge by the given index.

[method:Face compute ()

Computes all properties of the face.

[method:Float distanceToPoint([page:Vector3 point)

[page:Vector3 point - Any point in 3D space.

Returns the signed distance from a given point to the plane representation of this face.

Source

[link:<https://github.com/mrdoob/three.js/blob/master/examples/js/QuickHull.js>]

HalfEdge

The basis for a half-edge data structure, also known as doubly connected edge list (DCEL).

Конструктор

HalfEdge([page:VertexNode vertex, [page:Face face)]

[page:VertexNode vertex - [page:VertexNode A reference to its destination vertex.

[page:Face face - [page:Face A reference to its face.

Свойства

[property:VertexNode vertex

Reference to the destination vertex. The origin vertex can be obtained by querying the destination of its twin, or of the previous half-edge. Default is undefined.

[property:HalfEdge prev

Reference to the previous half-edge of the same face. Default is null.

[property:HalfEdge next

Reference to the next half-edge of the same face. Default is null.

[property:HalfEdge twin

Reference to the twin half-edge to reach the opposite face. Default is null.

[property:Face face]

Each half-edge bounds a single face and thus has a reference to that face. Default is undefined.

Методы

[method:VertexNode head()]

Returns the destination vertex.

[method:VertexNode tail()]

Returns the origin vertex.

[method:Float length()]

Returns the [link:https://en.wikipedia.org/wiki/Euclidean_distance] Euclidean length (straight-line length) of the edge.

[method:Float lengthSquared()]

Returns the square of the [link:https://en.wikipedia.org/wiki/Euclidean_distance] Euclidean length (straight-line length) of the edge.

[method:HalfEdge setTwin([page:HalfEdge edge)

[page:HalfEdge edge - Any half-edge.

Sets the twin edge of this half-edge. It also ensures that the twin reference of the given half-edge is correctly set.

Source

[link:<https://github.com/mrdoob/three.js/blob/master/examples/js/QuickHull.js>]

QuickHull

General information about the Quickhull algorithm: Dirk Gregorius. March 2014, Game Developers Conference:

[link:http://media.steampowered.com/apps/valve/2014/DirkGregorius_Imp Implementing QuickHull.

Конструктор

QuickHull()

Свойства

[property:Float tolerance]

The epsilon value that is used for internal comparative operations. The calculation of this value depends on the size of the geometry. Default is -1.

[property:Array faces]

The generated faces of the convex hull. Default is an empty array.

[property:Array newFaces]

This array holds the faces that are generated within a single iteration. Default is an empty array.

[property:VertexList assigned]

This [page:VertexList vertex list holds all vertices that are assigned to a

face. Default is an empty vertex list.

[property:VertexList unassigned]

This [page:VertexList vertex list holds all vertices that are not assigned to a face. Default is an empty vertex list.

[property:Array vertices]

The internal representation of the given geometry data (an array of [page:VertexNode vertices).

Методы

[method:QuickHull setFromPoints([page:Array points))

[page:Array points - Array of [page:Vector3 Vector3s that the resulting convex hull will contain.

Computes to convex hull for the given array of points.

[method:QuickHull setFromObject([page:Object3D object))

[page:Object3D object - [page:Object3D to compute the convex hull of.

Computes the convex hull of an [page:Object3D (including its children), accounting for the world transforms of both the object and its childrens.

[method:QuickHull makeEmpty()

Makes this convex hull empty.

[method:QuickHull addVertexToFace([page:VertexNode vertex,

[page:VertexNodeNode vertex - The vetex to add.

[page:Face face - The target face.

Adds a vertex to the 'assigned' list of vertices and assigns it to the given face.

[method:QuickHull removeVertexFromFace([page:VertexNode \

[page:VertexNode vertex - The vertex to remove.

[page:Face face - The target face.

Removes a vertex from the 'assigned' list of vertices and from the given face. It also makes sure that the link from 'face' to the first vertex it sees in 'assigned' is linked correctly after the removal.

[method:VertexNode removeAllVerticesFromFace([page:Face f;

[page:Face face - The given face.

Removes all the visible vertices that a given face is able to see which are stored in the 'assigned' vertex list.

[method:QuickHull deleteFaceVertices([page:Face face, [page:I

[page:Face face - The given face.

[page:Face absorbingFace - An optional face that tries to absorb the vertices of the first face.

Removes all the visible vertices that 'face' is able to see.

- If 'absorbingFace' doesn't exist, then all the removed vertices will be added to the 'unassigned' vertex list.
- If 'absorbingFace' exists, then this method will assign all the vertices of 'face' that can see 'absorbingFace'.
- If a vertex cannot see 'absorbingFace', it's added to the 'unassigned' vertex list.

[method:QuickHull resolveUnassignedPoints([page:Array newF

[page:Face newFaces - An array of new faces.

Reassigns as many vertices as possible from the unassigned list to the new faces.

[method:Object computeExtremes()]

Computes the extremes values (min/max vectors) which will be used to compute the initial hull.

[method:QuickHull computeInitialHull()]

Computes the initial simplex assigning to its faces all the points that are candidates to form part of the hull.

[method:QuickHull reindexFaces()]

Removes inactive (e.g. deleted) faces from the internal face list.

[method:VertexNode nextVertexToAdd()]

Finds the next vertex to create faces with the current hull.

- Let the initial face be the first face existing in the 'assigned' vertex list.
- If a face doesn't exist then return since there're no vertices left.
- Otherwise for each vertex that face sees find the one furthest away from it.

[method:QuickHull computeHorizon([page:Vector3 eyePoint, [pa

[page:Vector3 eyePoint - The 3D-coordinates of a point.

[page:HalfEdge crossEdge - The edge used to jump to the current face.

[page:Face face - The current face being tested.

[page:Array horizon - The edges that form part of the horizon in CCW

order.

Computes a chain of half edges in CCW order called the 'horizon'. For an edge to be part of the horizon it must join a face that can see 'eyePoint' and a face that cannot see 'eyePoint'.

[method:HalfEdge addAdjoiningFace([page:VertexNode eyeVer

[page:VertexNode eyeVertex - The vertex that is added to the hull.

[page:HalfEdge horizonEdge - A single edge of the horizon.

Creates a face with the vertices 'eyeVertex.point', 'horizonEdge.tail' and 'horizonEdge.head' in CCW order. All the half edges are created in CCW order thus the face is always pointing outside the hull

[method:QuickHull addNewFaces([page:VertexNode eyeVertex,

[page:VertexNode eyeVertex - The vertex that is added to the hull.

[page:HalfEdge horizon - An array of half-edges that form the horizon.

Adds 'horizon.length' faces to the hull, each face will be linked with the horizon opposite face and the face on the left/right.

[method:QuickHull addVertexToHull([page:VertexNode eyeVerte

[page:VertexNode eyeVertex - The vertex that is added to the hull.

Adds a vertex to the hull with the following algorithm

- Compute the 'horizon' which is a chain of half edges. For an edge to belong to this group it must be the edge connecting a face that can see 'eyeVertex' and a face which cannot see 'eyeVertex'.
- All the faces that can see 'eyeVertex' have its visible vertices removed from the assigned vertex list.
- A new set of faces is created with each edge of the 'horizon' and 'eyeVertex'. Each face is connected with the opposite horizon face

- and the face on the left/right.
- The vertices removed from all the visible faces are assigned to the new faces if possible.

[method:QuickHull cleanup()]

Cleans up internal properties after computing the convex hull.

[method:QuickHull compute()]

Starts the execution of the quick hull algorithm.

Source

[link:<https://github.com/mrdoob/three.js/blob/master/examples/js/QuickHull.js>]

VertexNode

A vertex as a double linked list node.

Конструктор

VertexNode([page:Vector3 point])

[page:Vector3 point] - [page:Vector3] A point (x, y, z) in 3D space.

Свойства

[property:Vector3 point]

A point (x, y, z) in 3D space. Default is undefined.

[property:VertexNode prev]

Reference to the previous vertex in the double linked list. Default is null.

[property:VertexNode next]

Reference to the next vertex in the double linked list. Default is null.

[property:Face face]

Reference to the face that is able to see this vertex. Default is undefined.

Source

[link:<https://github.com/mrdoob/three.js/blob/master/examples/js/QuickHull.js>]

VertexList

A doubly linked list of vertices.

Конструктор

VertexList()

Свойства

[property:VertexNode head]

Reference to the first vertex of the linked list. Default is null.

[property:VertexNode tail]

Reference to the last vertex of the linked list. Default is null.

Методы

[method:VertexNode first()]

Returns the head reference.

[method:VertexNode last()]

Returns the tail reference.

[method:VertexList clear()]

Clears the linked list.

[method:VertexList insertBefore([page:Vertex target, [page:Vertex vertex])

[page:Vertex target - The target vertex. It's assumed that this vertex belongs to the linked list.

[page:Vertex vertex - The vertex to insert.

Inserts a vertex **before** a target vertex.

[method:VertexList insertAfter([page:Vertex target, [page:Vertex vertex])

[page:Vertex target - The target vertex. It's assumed that this vertex belongs to the linked list.

[page:Vertex vertex - The vertex to insert.

Inserts a vertex **after** a target vertex.

[method:VertexList append([page:Vertex vertex))

[page:Vertex vertex - The vertex to append.

Appends a vertex to the end of the linked list.

[method:VertexList appendChain([page:Vertex vertex))

[page:Vertex vertex - The head vertex of a chain of vertices.

Appends a chain of vertices where the given vertex is the head.

[method:VertexList remove([page:Vertex vertex))

[page:Vertex vertex - The vertex to remove.

Removes a vertex from the linked list.

[method:VertexList removeSubList([page:Vertex a, [page:Vertex b))

[page:Vertex a - The head of the sublist.

[page:Vertex b - The tail of the sublist.

Removes a sublist of vertices from the linked list.

[method:Boolean isEmpty()

Returns true if the linked list is empty.

Source

[link:<https://github.com/mrdoob/three.js/blob/master/examples/js/QuickHull.js>]

Визуализаторы

CanvasRenderer

The Canvas renderer displays your beautifully crafted scenes **not** using WebGL, but draws it using the (slower) [Canvas 2D Context API](#).

NOTE: The Canvas renderer has been deprecated and is no longer part of the three.js core. If you still need to use it you can find it here: [link:[https://github.com/mrdoob/three.js/blob/master/examples/js/\[path.js](https://github.com/mrdoob/three.js/blob/master/examples/js/[path.js)]

This renderer can be a nice fallback from [page:WebGLRenderer for simple scenes:

```
function webglAvailable() {
    try {
        var canvas = document.createElement('canvas');
        return !(window.WebGLRenderingContext.prototype.getContext =
            canvas.getContext ||
            canvas.createContext);
    } catch (e) {
        return false;
    }
}

if ( webglAvailable() ) {
    renderer = new THREE.WebGLRenderer();
} else {
    renderer = new THREE.CanvasRenderer();
}
```

Note: both WebGLRenderer and CanvasRenderer are embedded in the web page using an HTML5 <canvas> tag. The "Canvas" in

CanvasRenderer means it uses Canvas 2D instead of WebGL.

Don't confuse either CanvasRenderer with the SoftwareRenderer example, which simulates a screen buffer in a Javascript array.

Конструктор

[name([page:object parameters)]

parameters is an optional object with properties defining the renderer's behaviour. The constructor also accepts no parameters at all. In all cases, it will assume sane defaults when parameters are missing.
canvas — A [page:Canvas where the renderer draws its output.

Свойства

[property:Object info]

An object with a series of statistical information about the graphics board memory and the rendering process. Useful for debugging or just for the sake of curiosity. The object contains the following fields:

- render:
 - vertices
 - faces

[property:DOMElement domElement]

A [page:Canvas where the renderer draws its output. This is automatically created by the renderer in the constructor (if not provided already); you just need to add it to your page.

[property:Boolean autoClear]

Defines whether the renderer should automatically clear its output before rendering.

[property:Boolean sortObjects]

Defines whether the renderer should sort objects. Default is true.

Note: Sorting is used to attempt to properly render objects that have some degree of transparency. By definition, sorting objects may not work in all cases. Depending on the needs of application, it may be necessary to turn off sorting and use other methods to deal with transparency rendering e.g. manually determining the object rendering order.

[property:boolean sortElements]

Defines whether the renderer should sort the face of each object. Default is true.

Методы

[method:null render([page:Scene scene, [page:Camera camera])

scene -- The scene to render.

camera -- the camera to view the scene.

Render a scene using a camera.

[method:null clear())

Tells the renderer to clear its color drawing buffer with the clearColor.

[method:null setClearColor([page:Color color, [page:number alp)]

color -- The color to clear the canvas with.

alpha -- The alpha channel to clear the canvas with.

This set the clearColor and the clearAlpha.

[method:null setSize([page:Number width, [page:Number height)]

width -- The width of the drawing canvas.

height -- The height of the drawing canvas.

This set the size of the drawing canvas and if updateStyle is set, then the css of the canvas is updated too.

[method:null setClearColorHex([page:number hex, [page:numbe)]

hex -- The the hexadecimal value of the color to clear the canvas with.

alpha -- The alpha channel to clear the canvas with.

This set the clearColor and the clearAlpha.

[method:number getClearColorHex()]

Returns the [page:number hex color.

[method:number getClearAlpha()]

Returns the alpha value.

Empty Methods to Maintain Compatibility with [page]:

```
[method:null clearColor()  
[method:null clearDepth()  
[method:null clearStencil()  
[method:null setFaceCulling()  
[method:null supportsVertexTextures()  
[method:number getMaxAnisotropy() - returns 1
```

Source

[link:[https://github.com/mrdoob/three.js/blob/master/examples/js/\[path.js](https://github.com/mrdoob/three.js/blob/master/examples/js/[path.js)
examples/js/[path.js

СПРАВОЧНИК РАЗРАБОТЧИКА

Полифиллы

«Полифилл» (англ. «polyfill» - poly - много, fill - наполнять, заполнять)
... читать далее

В *three.js* включены полифиллы для следующих функций и констант.

[Number.EPSILON](#)

Разница между единицей и наименьшим значением, большим единицы, которую можно представить в виде числа.

[Справка MDN \(Mozilla Developer Network - сеть разработчиков Mozilla\)](#),

[то же и там же, но на русском языке](#).

[Math.sign\(x \)](#)

Если аргумент является положительным числом, отрицательным числом, положительным нулем или отрицательным нулем, функция возвратит соответственно 1, -1, 0 или -0. В противном случае возвращается значение [NaN](#).

[Справка MDN \(Mozilla Developer Network - сеть разработчиков Mozilla\)](#),

[то же и там же, но на русском языке](#).

[Function.prototype.name\(x \)](#)

Возвращает имя функции или (до внедрения ES6) пустую строку для безымянных функций.

[Справка MDN \(Mozilla Developer Network - сеть разработчиков Mozilla\)](#),

[то же и там же, но на русском языке](#).

[page:Object.assign Object.assign([page:Object target, [page:Obj

Метод [Object.assign\(\)](#) используется для копирования значений

всех имеющихся перечислимых свойств одного или нескольких исходных объектов в целевой объект (target). Метод возвратит целевой объект.

[Справка MDN \(Mozilla Developer Network - сеть разработчиков Mozilla\),](#)
[то же и там же, но на русском языке.](#)

Исходники

[Polyfills.js на github.com](#)

WebGLRenderer

WebGLProgram

Constructor for the GLSL program sent to vertex and fragment shaders, including default uniforms and attributes.

Встроенные униформы и атрибуты Built-in uniforms and attributes

Vertex shader (unconditional):

```
// = object.matrixWorld
uniform mat4 modelMatrix;

// = camera.matrixWorldInverse * object.matrixWorld
uniform mat4 modelViewMatrix;

// = camera.projectionMatrix
uniform mat4 projectionMatrix;

// = camera.matrixWorldInverse
uniform mat4 viewMatrix;

// = inverse transpose of modelViewMatrix
uniform mat3 normalMatrix;

// = camera position in world space
uniform vec3 cameraPosition;

// default vertex attributes provided by Geometry and BufferGeometry
attribute vec3 position;
attribute vec3 normal;
attribute vec2 uv;
attribute vec2 uv2;
```

Note that you can therefore calculate the position of a vertex in the vertex shader by:

```
gl_Position = projectionMatrix * modelViewMatrix * vec4( position,
```

or alternatively

```
gl_Position = projectionMatrix * viewMatrix * modelMatrix * vec4( p
```

Vertex shader (conditional):

```

#define USE_COLOR
    // vertex color attribute
    attribute vec3 color;
#endif

#define USE_MORPHTARGETS

    attribute vec3 morphTarget0;
    attribute vec3 morphTarget1;
    attribute vec3 morphTarget2;
    attribute vec3 morphTarget3;

#define USE_MORPHNORMALS

    attribute vec3 morphNormal0;
    attribute vec3 morphNormal1;
    attribute vec3 morphNormal2;
    attribute vec3 morphNormal3;

#else

    attribute vec3 morphTarget4;
    attribute vec3 morphTarget5;
    attribute vec3 morphTarget6;
    attribute vec3 morphTarget7;

#endif
#endif

#define USE_SKINNING
    attribute vec4 skinIndex;
    attribute vec4 skinWeight;
#endif

```

Fragment shader:

```

uniform mat4 viewMatrix;
uniform vec3 cameraPosition;

```

Конструктор

`WebGLProgram(renderer, [page:Object code, [page:Material material,`

Параметрысмотрите в описании [WebGLRenderer](#).

Свойства

```
id
code
usedTimes
program
[property:WebGLShader vertexShader
[property:WebGLShader fragmentShader
```

Методы

```
getUniforms\(\)
```

Returns a name-value mapping of all active uniform locations.

```
getAttributes\(\)
```

Returns a name-value mapping of all active vertex attribute locations.

Исходники

[link:[https://github.com/mrdoob/three.js/blob/master/src/\[path\].js](https://github.com/mrdoob/three.js/blob/master/src/[path].js)]

WebGLShader

A lower level function to compile either a vertex or fragment shader.

Пример

```
var gl = renderer.context;

var glVertexShader = new THREE.WebGLShader( gl, gl.VERTEX_SHADER, v
var glFragmentShader = new THREE.WebGLShader( gl, gl.FRAGMENT_SHADE

var program = gl.createProgram();

gl.attachShader( program, glVertexShader );
gl.attachShader( program, glFragmentShader );
```

```
gl.linkProgram( program );
```

Функция

```
[page:WebGLShader objects([page:WebGLContext gl, [page:WebGLenum type -- текущий контекст WebGL  
type -- тип WebGL, либо gl.VERTEX_SHADER, либо gl.FRAGMENT_SHADER  
source -- исходный код для шейдера
```

This will compile an individual shader, but won't link it to be a complete [page:WebGLProgram]. Note: this is a function so the new operator should not be used.

Исходники

```
[link:https://github.com/mrdoob/three.js/blob/master/src/\[path\].js  
src/[path].js]
```

WebGLState

Методы

```
[method:null enable( [page:integer id, [page:boolean boolean )
```

TODO

```
[method:null disable]( [page:integer id], [page:boolean boolean])
```

TODO

```
[method:null setDepthTest]( [page:boolean depthTest] )
```

depthTest -- The boolean to decide if depth of a fragment needs to be tested, based on depthTest, whether or not the depth data needs to be tested against the depth buffer.

```
[method:null setDepthWrite]( [page:boolean depthWrite] )
```

depthWrite -- The boolean to decide if depth of a fragment needs to be written, based on depthWrite, whether or not the depth data needs to be written in the depth buffer.

```
[method:null setBlending]( [page:number blending], [page:number b  
blending -- A number indicating the blending mode. Possible  
value are THREE.NoBlending, THREE.NormalBlending,  
THREE.AdditiveBlending, THREE.SubtractiveBlending,  
THREE.MultiplyBlending or THREE.CustomBlending  
blendEquation -- When blending is THREE.CustomBlending,  
then you can set the blendEquation. Possible values are  
THREE.AddEquation, THREE.SubtractEquation or  
THREE.ReverseSubtractEquation.  
blendSrc -- When blending is THREE.CustomBlending, then you  
can set the blendSrc. Possible values are THREE.ZeroFactor,  
THREE.OneFactor, THREE.SrcColorFactor,  
THREE.OneMinusSrcColorFactor, THREE.SrcAlphaFactor,  
THREE.OneMinusSrcAlphaFactor, THREE.DstAlphaFactor,  
THREE.OneMinusDstAlphaFactor,  
THREE.DstColorFactor, THREE.OneMinusDstColorFactor or  
THREE.SrcAlphaSaturateFactor  
blendDst -- When blending is THREE.CustomBlending, then you  
can set the blendDst. Possible values are THREE.ZeroFactor,  
THREE.OneFactor, THREE.SrcColorFactor,  
THREE.OneMinusSrcColorFactor, THREE.SrcAlphaFactor,  
THREE.OneMinusSrcAlphaFactor, THREE.DstAlphaFactor,  
THREE.OneMinusDstAlphaFactor,  
THREE.DstColorFactor, THREE.OneMinusDstColorFactor or  
THREE.SrcAlphaSaturateFactor
```

This method sets the correct blending.

Исходники

[link:[https://github.com/mrdoob/three.js/blob/master/src/\[path\].js](https://github.com/mrdoob/three.js/blob/master/src/[path].js)
src/[path].js]

Плагины WebGLRenderer

LensFlarePlugin

The Webglrenderer plugin class that allows lensflares to be rendered in the WebGLRenderer. Этот плагин автоматически загружен в WebGLrenderer.

Конструктор

```
LensFlarePlugin()
```

Создает новый LensFlarePlugin.

Методы

```
.render( scene, camera, viewportWidth, viewportHeight )
```

scene -- The scene to render.

camera -- The camera to render.

viewportWidth -- ширина окна просмотра The width of the viewpor

viewportHeight -- высота окна просмотра The height of the viewpo

Renders the lensflares defined in the scene. This gets automatically called as post render function to draw the lensflares.

Исходники

[LensFlarePlugin.js на github.com](#)

SpritePlugin

The Webglrenderer plugin class that allows Sprites to be rendered in the WebGLRenderer. This plugin is automatically loaded in the WebGLrenderer.

Конструктор

`SpritePlugin()`

Создает новый SpritePlugin.

Методы

`.render(scene, camera)`

 | `scene` -- The scene to render.

 | `camera` -- The camera to render.

Renders the sprites defined in the scene. This gets automatically called as post-render function to draw the lensflares.

Исходники

[SpritePlugin.js на github.com](#)