Alpha Engine

Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

AEGfxTexture	
AEGfxVertexList	
AELineSegment2	
AEMtx33	
AEVec2	

Alpha Engine

Public Attributes | List of all members

AEGfxTexture Struct Reference

`#include <AEGraphics.h>`

Public Attributes

AEGfxSurface *	**mpSurface**
s8	**mpName** [256]

Detailed Description

Definition at line **82** of file **AEGraphics.h**.

Member Data Documentation

s8 AEGfxTexture::mpName[256]

Definition at line 85 of file AEGraphics.h.

AEGfxSurface* AEGfxTexture::mpSurface

Definition at line 84 of file AEGraphics.h.

The documentation for this struct was generated from the following file:

- AEGraphics.h

Public Attributes | List of all members

AEGfxVertexList Struct Reference

`#include <AEGraphics.h>`

Public Attributes

AEGfxVertexBuffer *	mpVtxBuffer
u32	vtxNum

Detailed Description

Definition at line **72** of file **AEGraphics.h**.

Member Data Documentation

AEGfxVertexBuffer* AEGfxVertexList::mpVtxBuffer

Definition at line 74 of file AEGraphics.h.

u32 AEGfxVertexList::vtxNum

Definition at line 75 of file AEGraphics.h.

The documentation for this struct was generated from the following file:

- AEGraphics.h

Alpha Engine

Public Attributes | List of all members

AELineSegment2 Struct Reference

`#include <AELineSegment2.h>`

Public Attributes

AEVec2	**mP0** Point on the line. More...
AEVec2	**mP1** Point on the line. More...
AEVec2	**mN** Line's normal. More...
float	**mNdotP0** To avoid computing it every time it's needed. More...

Detailed Description

Definition at line **22** of file **AELineSegment2.h**.

Member Data Documentation

AEVec2 AELineSegment2::mN

Line's normal.

Definition at line 26 of file AELineSegment2.h.

float AELineSegment2::mNdotP0

To avoid computing it every time it's needed.

Definition at line 27 of file AELineSegment2.h.

AEVec2 AELineSegment2::mP0

Point on the line.

Definition at line 24 of file AELineSegment2.h.

AEVec2 AELineSegment2::mP1

Point on the line.

Definition at line 25 of file AELineSegment2.h.

The documentation for this struct was generated from the following file:

- AELineSegment2.h

Alpha Engine

Public Attributes | List of all members

AEMtx33 Struct Reference

`#include <AEMtx33.h>`

Public Attributes

f32 m [3][3]

Detailed Description

Matrix is stored in column major format, ie. the translation term is in the right most column.

m[0][0] m[0][1] m[0][2]
m[1][0] m[1][1] m[1][2]
m[2][0] m[2][1] m[2][2]

Definition at line **36** of file **AEMtx33.h**.

Member Data Documentation

f32 AEMtx33::m[3][3]

Definition at line 38 of file AEMtx33.h.

The documentation for this struct was generated from the following file:

- AEMtx33.h

Alpha Engine

Public Attributes | List of all members

AEVec2 Struct Reference

```
#include <AEVec2.h>
```

Public Attributes

f32	x
f32	y

Detailed Description

Definition at line **26** of file **AEVec2.h**.

Member Data Documentation

f32 AEVec2::x

Definition at line 28 of file AEVec2.h.

f32 AEVec2::y

Definition at line 28 of file AEVec2.h.

The documentation for this struct was generated from the following file:

- AEVec2.h

Alpha Engine

Here is a list of all class members with links to the classes they belong to:

- m : **AEMtx33**
- mN : **AELineSegment2**
- mNdotP0 : **AELineSegment2**
- mP0 : **AELineSegment2**
- mP1 : **AELineSegment2**
- mpName : **AEGfxTexture**
- mpSurface : **AEGfxTexture**
- mpVtxBuffer : **AEGfxVertexList**
- vtxNum : **AEGfxVertexList**
- x : **AEVec2**
- y : **AEVec2**

Alpha Engine

- m : **AEMtx33**
- mN : **AELineSegment2**
- mNdotP0 : **AELineSegment2**
- mP0 : **AELineSegment2**
- mP1 : **AELineSegment2**
- mpName : **AEGfxTexture**
- mpSurface : **AEGfxTexture**
- mpVtxBuffer : **AEGfxVertexList**
- vtxNum : **AEGfxVertexList**
- x : **AEVec2**
- y : **AEVec2**

Alpha Engine

File List

Here is a list of all files with brief descriptions:

AEEngine.h	Header file for the Alpha Engine
AEExport.h	Header file for the library settings
AEFrameRateController.h	Header file for the frame rate controller
AEGameStateMgr.h	Header file for the game state manager
AEGraphics.h	Header file for the graphics library
AEInput.h	Header file for the input library
AELineSegment2.h	Header file for the 2D line segment library
AEMath.h	Header file for the math library
AEMtx33.h	Header file for the 3x3 matrix library
AESystem.h	Header file for the system library
AETypes.h	Header file for the typedefs
AEUtil.h	Header file for the utility library
AEVec2.h	Header file for the 2D vector library

Alpha Engine

Macros

AEEngine.h File Reference

Header file for the Alpha Engine. More...

```
#include <stdlib.h> #include <stdio.h>
#include <string.h>
#include <assert.h>
#include <windows.h>
#include "AEExport.h"
#include "AETypes.h"
#include "AEMath.h"
#include "AEUtil.h"
#include "AEFrameRateController.h"
#include "AESystem.h"
#include "AEGraphics.h"
#include "AEInput.h"
#include "AEGameStateMgr.h"
```

Go to the source code of this file.

Macros

#define **AE_ASSERT**(x)

#define **AE_ASSERT_MESG**(x,...)

#define **AE_ASSERT_PARM**(x)

#define **AE_ASSERT_ALLOC**(x)

#define **AE_WARNING**(x)

#define **AE_WARNING_MESG**(x,...)

#define **AE_WARNING_PARM**(x)

#define **AE_FATAL_ERROR**(...)

Detailed Description

Header file for the Alpha Engine.

Project: Alpha Engine

Author
Sun Tjen Fam

Date
January 31, 2008

The Alpha Engine consist of the following header files:

- **AEEngine.h**
- **AETypes.h**
- **AEMath.h**
  - **AEVec2.h**
  - **AEMtx33.h**
  - **AELineSegment2.h**
- **AEUtil.h**
- **AEFrameRateController.h**
- **AESystem.h**
- AEGraphcs.h
- **AEInput.h**
- **AEGameStateMgr.h**

Definition in file **AEEngine.h**.

Macro Definition Documentation

#define AE_ASSERT (x)

Value:

```
{
     \
    if((x) == 0)
     \
    {
     \
                PRINT("AE_ASSERT: %s\nLine:
    %d\nFunc: %s\nFile: %s\n",         \
            #x, __LINE__, __FUNCTION__, __FILE__);
    \
        exit(1);
    \
    }
     \
}
```

Definition at line **52** of file **AEEngine.h**.

#define AE_ASSERT_ALLOC (x)

Value:

```
{
     \
    if((x) == 0)
     \
    {
     \
                PRINT("AE_ASSERT_ALLOC: %s\nLine:
    %d\nFunc: %s\nFile: %s\n",  \
```

```
            #x, __LINE__, __FUNCTION__, __FILE__);
    \
        exit(1);
    \
    }
    \
}
```

Definition at line **90** of file **AEEngine.h**.

#define AE_ASSERT_MESG (x,
...
)

Value:

```
{
    \
    if((x) == 0)
    \
    {
    \
                PRINT("AE_ASSERT_MESG: %s\nLine:
    %d\nFunc: %s\nFile: %s\n",   \
            #x, __LINE__, __FUNCTION__, __FILE__);
    \
                PRINT("Mesg: "__VA_ARGS__);
    \
                PRINT("\n");
    \
        exit(1);
    \
    }
    \
}
```

Definition at line **64** of file **AEEngine.h**.

#define AE_ASSERT_PARM (x)

Value:

```
{
     \
    if((x) == 0)
     \
    {
     \
                 PRINT("AE_ASSERT_PARM: %s\nLine:
    %d\nFunc: %s\nFile: %s\n",    \
            #x, __LINE__, __FUNCTION__, __FILE__);
    \
        exit(1);
    \
    }
     \
}
```

Definition at line **78** of file **AEEngine.h**.

#define AE_FATAL_ERROR (...)

Value:

```
{                                                  \
        PRINT("AE_FATAL_ERROR: "__VA_ARGS__);   \
    exit(1);                                       \
}
```

Definition at line **158** of file **AEEngine.h**.

#define AE_WARNING (x)

Value:

```
{
     \
    if((x) == 0)
```

```
    \
    {
    \
                PRINT("AE_WARNING: %s\nLine:
    %d\nFunc: %s\nFile: %s\n",   \
            #x, __LINE__, __FUNCTION__, __FILE__);
    \
    }
    \
}
```

Definition at line **103** of file **AEEngine.h**.

#define AE_WARNING_MESG (x,
...
)

Value:

```
{
    \
    if((x) == 0)
    \
    {
    \
                PRINT("AE_WARNING_MESG: %s\nLine:
    %d\nFunc: %s\nFile: %s\n",  \
            #x, __LINE__, __FUNCTION__, __FILE__);
    \
                PRINT("Mesg: "__VA_ARGS__);
    \
                PRINT("\n");
    \
    }
    \
}
```

Definition at line **114** of file **AEEngine.h**.

#define AE_WARNING_PARM (x)

Value:

```
{
     \
    if((x) == 0)
     \
    {
     \
                 PRINT("AE_WARNING_PARM: %s\nLine:
    %d\nFunc: %s\nFile: %s\n",  \
            #x, __LINE__, __FUNCTION__, __FILE__);
     \
    }
     \
}
```

Definition at line **127** of file **AEEngine.h**.

Alpha Engine

Macros

AEExport.h File Reference

Header file for the library settings. More...

Go to the source code of this file.

Macros

#define	**EPSILON** 0.00001f
#define	**PI** 3.1415926f
#define	**HALF_PI** (**PI** * 0.5f)
#define	**TWO_PI** (**PI** * 2.0f)
#define	**EXPORT_WINDOWS** 1
#define	**EXPORT_ANDROID** 0
#define	**PRINT**(...) printf(__VA_ARGS__)
#define	**PRINT_INFO**(...)
#define	**AE_API** __declspec(dllexport)
#define	**DECLARE_FUNCTION_FOR_ANDROID**(functionName)
#define	**IMPLEMENT_FUNCTION_FOR_ANDROID**(functionName)
#define	**DECLARE_FUNCTION_FOR_ANDROID_2_INT**(functionNam paramType1, paramType2)
#define	**IMPLEMENT_FUNCTION_FOR_ANDROID_2_INT**(functionNa paramType1, paramType2)

Detailed Description

Header file for the library settings.

Project: Alpha Engine

Author
Antoine Abi Chacra

Date
March 21, 2013

Copyright

Definition in file **AEExport.h**.

Macro Definition Documentation

#define AE_API __declspec(dllexport)

Definition at line **53** of file **AEExport.h**.

#define DECLARE_FUNCTION_FOR_ANDROID (functionName)

Definition at line **59** of file **AEExport.h**.

#define DECLARE_FUNCTION_FOR_ANDROID_2_INT (functionName, paramType1, paramType2)

Definition at line **62** of file **AEExport.h**.

#define EPSILON 0.00001f

Definition at line **21** of file **AEExport.h**.

#define EXPORT_ANDROID 0

Definition at line **33** of file **AEExport.h**.

#define EXPORT_WINDOWS 1

Definition at line 32 of file AEExport.h.

#define HALF_PI (PI * 0.5f)

Definition at line 28 of file AEExport.h.

#define IMPLEMENT_FUNCTION_FOR_ANDROID (functionName)

Definition at line 60 of file AEExport.h.

#define IMPLEMENT_FUNCTION_FOR_ANDROID_2_INT (functionName, paramType1, paramType2)

Definition at line 63 of file AEExport.h.

#define PI 3.1415926f

Definition at line 25 of file AEExport.h.

#define PRINT (...) printf(__VA_ARGS__)

Definition at line 40 of file AEExport.h.

#define PRINT_INFO (...)

Value:

```
PRINT("File: %s\nLine: %d\nFunc: %s\n\n",
```

```
    \

    __FILE__,__LINE__, __FUNCTION__);
```

Definition at line **46** of file **AEExport.h**.

#define TWO_PI (PI * 2.0f)

Definition at line **29** of file **AEExport.h**.

Alpha Engine

Functions

AEFrameRateController.h File Reference

Header file for the frame rate controller. More...

Go to the source code of this file.

Functions

void **AEFrameRateControllerInit** (**u32** FrameRateMax)
Initialize the frame rate controller. More...

void **AEFrameRateControllerReset** ()
Reset the frame rate controller. More...

void **AEFrameRateControllerStart** ()
Tell the frame rate controller that a new frame is starting. More...

void **AEFrameRateControllerEnd** ()
Tell the frame rate controller that the current frame is ending. More...

f64 **AEFrameRateControllerGetFrameTime** ()
Get the time for the previous frame, in seconds. More...

u32 **AEFrameRateControllerGetFrameCount** ()
Get the total number of frames elapsed. More...

Detailed Description

Header file for the frame rate controller.

Project: Alpha Engine

Author
Sun Tjen Fam

Date
April 26, 2007

Copyright

Definition in file **AEFrameRateController.h**.

Function Documentation

void AEFrameRateControllerEnd ( )

Tell the frame rate controller that the current frame is ending.

Warning
This function is already called in AESysFrameEnd.

Return values
void No return

u32 AEFrameRateControllerGetFrameCount ( )

Get the total number of frames elapsed.

Return values
f64 Return the total number of frames elapsed.

f64 AEFrameRateControllerGetFrameTime ( )

Get the time for the previous frame, in seconds.

Return values
f64 Return the time for the previous frame, in seconds.

bool AEFrameRateControllerInit (u32 FrameRateMax)

Initialize the frame rate controller.

Warning

This function is already called in AESysInit.

Parameters

[in] **FrameRateMax** The maximum number of frames per second.

Return values

bool Returns true

void AEFrameRateControllerReset ( )

Reset the frame rate controller.

Warning

This function is already called in AESysReset.

Return values

void No return

void AEFrameRateControllerStart ( )

Tell the frame rate controller that a new frame is starting.

Warning

This function is already called in AESysFrameStart.

Return values

void No return

Alpha Engine

Macros | Functions | Variables

AEGameStateMgr.h File Reference

Header file for the game state manager. More...

Go to the source code of this file.

Macros

#define **AE_GS_RESTART** 0xFFFFFFFE
Special GameState: Restart. More...

#define **AE_GS_QUIT** 0xFFFFFFFF
Special GameState: Quit. More...

Functions

void	**AEGameStateMgrAdd** (**u32** gameStateIdx, void(*pLoad)(), void(*pInit)(), void(*pUpdate)(), void(*pDraw)(), void(*pFree)(), void(*pUnload)()) Create a new GameState and add it to GameStateManager. More...
void	**AEGameStateMgrInit** (**u32** gameStateInit) Initialize the GameStateManager. More...
void	**AEGameStateMgrUpdate** () Update the GameStateManager. More...

Variables

u32 **gAEGameStateInit**
Initial GameState. More...

u32 **gAEGameStateCurr**
Current GameState. More...

u32 **gAEGameStatePrev**
Previous GameState. More...

u32 **gAEGameStateNext**
Next GameState. More...

void(* **AEGameStateLoad**)(void)
Function pointer for load. More...

void(* **AEGameStateInit**)(void)
Function pointer for init. More...

void(* **AEGameStateUpdate**)(void)
Function pointer for update. More...

void(* **AEGameStateDraw**)(void)
Function pointer for draw. More...

void(* **AEGameStateFree**)(void)
Function pointer for free. More...

void(* **AEGameStateUnload**)(void)
Function pointer for unload. More...

Detailed Description

Header file for the game state manager.

Project: Alpha Engine

Author
Sun Tjen Fam

Date
October 10, 2007

Copyright

Definition in file **AEGameStateMgr.h**.

Macro Definition Documentation

#define AE_GS_QUIT 0xFFFFFFFF

Special GameState: Quit.

Definition at line **24** of file **AEGameStateMgr.h**.

#define AE_GS_RESTART 0xFFFFFFFE

Special GameState: Restart.

Definition at line **23** of file **AEGameStateMgr.h**.

Function Documentation

```
void AEGameStateMgrAdd ( u32        gameStateIdx,
                         void(*)()  pLoad,
                         void(*)()  pInit,
                         void(*)()  pUpdate,
                         void(*)()  pDraw,
                         void(*)()  pFree,
                         void(*)()  pUnload
                       )
```

Create a new GameState and add it to GameStateManager.

The new GameState is identified as gameStateIdx and will contain the 6 function pointers passed in.

Parameters

[in]	**gameStateIdx**	ID of the newly created GameState.
[in]	**pLoad**	Function pointer to the load function of the new GameState.
[in]	**pInit**	Function pointer to the init function of the new GameState.
[in]	**pUpdate**	Function pointer to the update function of the new GameState.
[in]	**pDraw**	Function pointer to the draw function of the new GameState.
[in]	**pFree**	Function pointer to the free function of the new GameState.
[in]	**pUnload**	Function pointer to the unload function of the new GameState.

Return values

void No return.

void AEGameStateMgrInit (u32 gameStateInit)

Initialize the GameStateManager.

Set the initial GameState and start the GameStateManager with that GameState.

Warning
This function should be called AFTER all the GameStates have been added to the GameStateManager.

Parameters
`[in]` **gameStateInit** ID of the initial GameState.

Return values
void No return.

void AEGameStateMgrUpdate ( )

Update the GameStateManager.

Call this function when the GameState changes to set the function pointers for the new state.

Return values
void No return.

Variable Documentation

void(* AEGameStateDraw)(void)

Function pointer for draw.

void(* AEGameStateFree)(void)

Function pointer for free.

void(* AEGameStateInit)(void)

Function pointer for init.

void(* AEGameStateLoad)(void)

Function pointer for load.

void(* AEGameStateUnload)(void)

Function pointer for unload.

void(* AEGameStateUpdate)(void)

Function pointer for update.

u32 gAEGameStateCurr

Current GameState.

u32 gAEGameStateInit

Initial GameState.

u32 gAEGameStateNext

Next GameState.

u32 gAEGameStatePrev

Previous GameState.

Alpha Engine

Classes | Typedefs | Enumerations | Functions

AEGraphics.h File Reference

Header file for the graphics library. More...

Go to the source code of this file.

Classes

struct	**AEGfxVertexList**
struct	**AEGfxTexture**

Typedefs

typedef struct **AEGfxVertexBuffer**	**AEGfxVertexBuffer**
typedef struct **AEGfxVertexList**	**AEGfxVertexList**
typedef struct **AEGfxSurface**	**AEGfxSurface**
typedef struct **AEGfxTexture**	**AEGfxTexture**

Enumerations

enum	**AEGfxRenderMode** { **AE_GFX_RM_NONE**, **AE_GFX_RM_COLOR**, **AE_GFX_RM_TEXTURE**, **AE_GFX_RM_NUM** }
enum	**AEGfxBlendMode** { **AE_GFX_BM_NONE** = 0, **AE_GFX_BM_BLEND**, **AE_GFX_BM_ADD**, **AE_GFX_BM_NUM** }
enum	**AEGfxTextureMode** { **AE_GFX_TM_PRECISE** = 0, **AE_GFX_TM_AVERAGE** }
enum	**AEGfxMeshDrawMode** { **AE_GFX_MDM_POINTS** = 0, **AE_GFX_MDM_LINES**, **AE_GFX_MDM_LINES_STRIP**, **AE_GFX_MDM_TRIANGLES**, **AE_GFX_MDM_NUM** }

Functions

s32	**AEGfxInit** (**s32** Width, **s32** Height) Initialize AEGraphics. More...
	DECLARE_FUNCTION_FOR_ANDROID_2_INT (**AEGfxInit**, **s32** Width, **s32** Height)
void	**AEGfxReset** () Reset AEGraphics. More...
void	**AEGfxExit** () Free AEGraphics. More...
void	**AEGfxStart** () Tell AEGraphics that a new frame is starting. More...
	DECLARE_FUNCTION_FOR_ANDROID (**AEGfxStart**)
void	**AEGfxEnd** () Tell AEGraphics that the current frame is ending. More...
	DECLARE_FUNCTION_FOR_ANDROID (**AEGfxEnd**)
void	**AEGfxSetBackgroundColor** (**f32** Red, **f32** Green, **f32** Blue) Set the background colour to selected RGB values. More...
void	**AEGfxSetRenderMode** (**AEGfxRenderMode** RenderMode) Set the render mode. More...

void	**AEGfxSetBlendMode** (**AEGfxBlendMode** BlendMode) Set the blend mode. More...
f32	**AEGfxGetWinMinX** (void) Get the minimum X world coordinate. More...
f32	**AEGfxGetWinMaxX** (void) Get the maximum X world coordinate. More...
f32	**AEGfxGetWinMinY** (void) Get the minimum Y world coordinate. More...
f32	**AEGfxGetWinMaxY** (void) Get the maximum Y world coordinate. More...
void	**AEGfxSetCamPosition** (**f32** X, **f32** Y) Set the camera's X and Y position. More...
void	**AEGfxGetCamPosition** (**f32** *pX, **f32** *pY) Get the camera's X and Y position. More...
void	**AEGfxSetPosition** (**f32** X, **f32** Y) Set the position (translation) to render from. More...
void	**AEGfxSetTransform** (**f32** pTransform[3][3]) Set the new global transformation matrix. More...
void	**AEGfxSetTransform3D** (**f32** pTransform[4][4]) Set the new global transformation matrix. More...
void	**AEGfxSetTransparency** (**f32** Alpha) Set the new global transparency value. More...

void	**AEGfxSetBlendColor** (**f32** Red, **f32** Green, **f32** Blue, **f32** Alpha) Sets a color (RGBA) that will be used to blend with the original material. More...
void	**AEGfxSetTintColor** (float Red, float Green, float Blue, float Alpha) Sets a color (RGBA) that will be used to tint the original material. More...
void	**AEGfxMeshStart** () Instruct AEGraphics to start creating a new mesh. More...
void	**AEGfxTriAdd** (**f32** x0, **f32** y0, **u32** c0, **f32** tu0, **f32** tv0, **f32** x1, **f32** y1, **u32** c1, **f32** tu1, **f32** tv1, **f32** x2, **f32** y2, **u32** c2, **f32** tu2, **f32** tv2) Add a new triangle to the mesh. More...
void	**AEGfxVertexAdd** (**f32** x0, **f32** y0, **u32** c0, **f32** tu0, **f32** tv0) Add a new point to the mesh. More...
AEGfxVertexList *	**AEGfxMeshEnd** () Instruct AEGraphics to end creating of new mesh. More...
void	**AEGfxMeshDraw** (**AEGfxVertexList** *pVertexList, **AEGfxMeshDrawMode** MeshDrawMode) Draw the mesh on screen. More...
void	**AEGfxMeshFree** (**AEGfxVertexList** *pVertexList) Free the mesh. More...

AEGfxTexture *	**AEGfxTextureLoad** (const **s8** *pFileName) Load a texture file into memory. More...
void	**AEGfxTextureSet** (**AEGfxTexture** *pTexture, **f32** offset_x, **f32** offset_y) Set a texture to be used for rendering. More...
void	**AEGfxTextureUnload** (**AEGfxTexture** *pTexture) Unload a texture file from memory. More...
AEGfxTexture *	**AEGfxTextureLoadFromMemory** (**u8** *pColors, **u32** Width, **u32** Height) Load a texture from data in memory. More...
void	**AEGfxSaveTextureToFile** (**AEGfxTexture** *pTexture, **s8** *pFileName) Save a texture from memory to file. More...
void	**AEGfxSetTextureMode** (**AEGfxTextureMode** TextureMode) Set the texture mode. More...
void	**AEGfxPoint** (**f32** x0, **f32** y0, **f32** z0, **u32** c0) Draw a point on screen. More...
void	**AEGfxLine** (**f32** x0, **f32** y0, **f32** z0, **f32** r0, **f32** g0, **f32** b0, **f32** a0, **f32** x1, **f32** y1, **f32** z1, **f32** r1, **f32** g1, **f32** b1, **f32** a1) Draw a line on screen. More...
void	**AEGfxTri** (**f32** x0, **f32** y0, **f32** z0, **u32** c0, **f32** x1, **f32** y1, **f32** z1, **u32** c1, **f32** x2, **f32** y2, **f32** z2, **u32** c2) Draw a triangle on screen. More...
	AEGfxQuad (**f32** x0, **f32** y0, **f32** z0, **u32** c0, **f32**

void	x1, **f32** y1, **f32** z1, **u32** c1, **f32** x2, **f32** y2, **f32** z2, **u32** c2, **f32** x3, **f32** y3, **f32** z3, **u32** c3) Draw a rectangle on screen. More...
void	**AEGfxBox** (**f32** x0, **f32** y0, **f32** z0, **f32** sizeX, **f32** sizeY, **f32** sizeZ, **u32** c0, **u32** c1) Draw a 3D box on screen. More...
void	**AEGfxSphere** (**f32** x0, **f32** y0, **f32** z0, **f32** radius, **u32** c0, **u32** c1, **u32** division) Draw a 3D sphere on screen. More...
void	**AEGfxCone** (**f32** x0, **f32** y0, **f32** z0, **f32** x1, **f32** y1, **f32** z1, **f32** radius, **u32** c0, **u32** c1, **u32** division) Draw a 3D cone on screen. More...
void	**AEGfxAxis** (**f32** scale) Draw a 3D axis on screen. More...
u32	**AEGfxColInterp** (**u32** c0, **u32** c1, **f32** t) Get the interpolation between two colours (c0 and c1) at time interval t. More...
void	**AEGfxPrint** (**s32** x, **s32** y, **u32** color, **s8** *pStr) Render text onto the screen. More...

Detailed Description

Header file for the graphics library.

Project: Alpha Engine

Author
Sun Tjen Fam

Date
January 30, 2008

Copyright

Definition in file **AEGraphics.h**.

Typedef Documentation

typedef struct AEGfxSurface AEGfxSurface

Definition at line **80** of file **AEGraphics.h**.

typedef struct AEGfxTexture AEGfxTexture

typedef struct AEGfxVertexBuffer AEGfxVertexBuffer

Definition at line **70** of file **AEGraphics.h**.

typedef struct AEGfxVertexList AEGfxVertexList

Enumeration Type Documentation

enum AEGfxBlendMode

AEEngine blend option

Enumerator	
AE_GFX_BM_NONE	No blending.
AE_GFX_BM_BLEND	Color blending.
AE_GFX_BM_ADD	Additive blending.
AE_GFX_BM_NUM	

Definition at line **37** of file **AEGraphics.h**.

enum AEGfxMeshDrawMode

Enumerator	
AE_GFX_MDM_POINTS	
AE_GFX_MDM_LINES	
AE_GFX_MDM_LINES_STRIP	
AE_GFX_MDM_TRIANGLES	
AE_GFX_MDM_NUM	

Definition at line **56** of file **AEGraphics.h**.

enum AEGfxRenderMode

AEEngine render option

Enumerator	

AE_GFX_RM_NONE	No render.
AE_GFX_RM_COLOR	Color rendering.
AE_GFX_RM_TEXTURE	Texture rendering.
AE_GFX_RM_NUM	

Definition at line **24** of file **AEGraphics.h**.

enum AEGfxTextureMode

Enumerator	
AE_GFX_TM_PRECISE	
AE_GFX_TM_AVERAGE	

Definition at line **48** of file **AEGraphics.h**.

Function Documentation

void AEGfxAxis (f32 scale)

Draw a 3D axis on screen.

The axis is rendered on the center of the screen with size scale.

Warning

Do not call this function when AEGraphics is building mesh.
Function may be slow. Use with caution.

Parameters

`[in]` **scale** Size of the axis

Return values

void No return.

void AEGfxBox (f32 x0,
f32 y0,
f32 z0,
f32 sizeX,
f32 sizeY,
f32 sizeZ,
u32 c0,
u32 c1
)

Draw a 3D box on screen.

The box is centered around the point (x0, y0, z0) and sizeX, sizeY and sizeZ are the width, height and depth. c0 and c1 are the box color. Default values are white and grey.

Warning

Do not call this function when AEGraphics is building mesh.

Function may be slow. Use with caution.

Parameters

[in] **x0** X coordinate of the center of the box
[in] **y0** Y coordinate of the center of the box
[in] **z0** Z coordinate of the center of the box
[in] **sizeX** Width of the box
[in] **sizeY** Height of the box
[in] **sizeZ** Depth of the box
[in] **c0** First color of the box.
[in] **c1** Second color of the box.

Return values

void No return.

```
u32 AEGfxColInterp ( u32  c0,
                     u32  c1,
                     f32  t
                   )
```

Get the interpolation between two colours (c0 and c1) at time interval t.

Warning

t should be between 0.0f to 1.0f.

Parameters

[in] **c0** The first colour.
[in] **c1** The second colour.
[in] **t** The time interval to calculate the interpolation at. At t = 0.0f, the result will be c0. At t = 1.0f, the result will be c1.

Return values

void No return.

```
void AEGfxCone ( f32  x0,
                 f32  y0,
                 f32  z0,
                 f32  x1,
                 f32  y1,
                 f32  z1,
                 f32  radius,
                 u32  c0,
                 u32  c1,
                 u32  division
               )
```

Draw a 3D cone on screen.

The base of the cone is centered around the point (x0, y0, z0) with size radius. The tip of the cone is at point(x1, y1, z1). c0 and c1 are the sphere color. Default colors are white and grey. Division is the resolution of the cone, or how many parts the cone consists of. Default value is 8.

Warning

This function is not implemented.

Do not call this function when AEGraphics is building mesh.

Function may be slow. Use with caution.

Parameters

[in]	**x0**	X coordinate of the center of the base of cone
[in]	**y0**	Y coordinate of the center of the base of cone
[in]	**z0**	Z coordinate of the tip of cone
[in]	**x1**	X coordinate of the tip of cone
[in]	**y1**	Y coordinate of the tip of cone
[in]	**z1**	Z coordinate of the tip of cone
[in]	**radius**	Radius of the cone

[in] **c0** First color of the cone.
[in] **c1** Second color of the cone.
[in] **division** Number of parts to divide the cone by.

Return values
void No return.

void AEGfxEnd ( )

Tell AEGraphics that the current frame is ending.

Warning
This function is already called in AESysFrameEnd.

Return values
void No return.

void AEGfxExit ( )

Free AEGraphics.

Warning
This function is already called in AESysExit.

Return values
void No return.

void AEGfxGetCamPosition (f32 * pX, f32 * pY)

Get the camera's X and Y position.

Parameters

[out] **pX** Pointer to f32 to store the X position in.
[out] **pY** Pointer to f32 to store the Y position in.

Return values
void No return.

f32 AEGfxGetWinMaxX (void)

Get the maximum X world coordinate.

Return values
f32 Returns the maximum X world coordinate.

f32 AEGfxGetWinMaxY (void)

Get the maximum Y world coordinate.

Return values
f32 Returns the maximum Y world coordinate.

f32 AEGfxGetWinMinX (void)

Get the minimum X world coordinate.

Return values
f32 Returns the minimum X world coordinate.

f32 AEGfxGetWinMinY (void)

Get the minimum Y world coordinate.

Return values
f32 Returns the minimum Y world coordinate.

```
s32 AEGfxInit ( s32  Width,
                s32  Height
              )
```

Initialize AEGraphics.

Warning
This function is already called in AESysInit.

Parameters

[in] **Width** Set the width of the window.
[in] **Height** Set the height of the window.

Return values

s32 Return 1 if initialization is successful. Else return 0.

```
void AEGfxLine ( f32  x0,
                 f32  y0,
                 f32  z0,
                 f32  r0,
                 f32  g0,
                 f32  b0,
                 f32  a0,
                 f32  x1,
                 f32  y1,
                 f32  z1,
                 f32  r1,
                 f32  g1,
                 f32  b1,
                 f32  a1
               )
```

Draw a line on screen.

The line is defined by 2 points: pt0 and pt1.

Warning

Do not call this function when AEGraphics is building mesh.

Function may be slow. Use with caution.

Parameters

- [in] **x0** X coordinate of pt0
- [in] **y0** Y coordinate of pt0
- [in] **z0** Z coordinate of pt0
- [in] **r0** Percentage of red of pt0. Range from 0.0f to 1.0f.
- [in] **g0** Percentage of green of pt0. Range from 0.0f to 1.0f.
- [in] **b0** Percentage of blue of pt0. Range from 0.0f to 1.0f.
- [in] **a0** Percentage of alpha (transparency) of pt0. Range from 0.0f (clear) to 1.0f (opaque).
- [in] **x1** X coordinate of pt1
- [in] **y1** Y coordinate of pt1
- [in] **z1** Z coordinate of pt1
- [in] **r1** Percentage of red of pt1. Range from 0.0f to 1.0f.
- [in] **g1** Percentage of green of pt1. Range from 0.0f to 1.0f.
- [in] **b1** Percentage of blue of pt1. Range from 0.0f to 1.0f.
- [in] **a1** Percentage of alpha (transparency) of pt1. Range from 0.0f (clear) to 1.0f (opaque).

Return values

void No return.

void AEGfxMeshDraw (AEGfxVertexList * pVertexList,
AEGfxMeshDrawMode MeshDrawMode
)

Draw the mesh on screen.

Render the mesh onto the screen using the MeshDrawMode stated.

Parameters

[in] **pVertexList** Pointer to the **AEGfxVertexList** to be rendered.

[in] **MeshDrawMode** The AEGfxMeshDrawMode to use for rendering the mesh.

Return values

void No return.

AEGfxVertexList * AEGfxMeshEnd ( )

Instruct AEGraphics to end creating of new mesh.

Call this function once after all the points have been added to the mesh.

Return values

AEGfxVertexList* Returns a **AEGfxVertexList** pointer to the new mesh.

void AEGfxMeshFree (AEGfxVertexList * pVertexList)

Free the mesh.

Release the mesh from AEGraphics memory. The mesh is destroyed and can no longer be used.

Parameters

[in] **pVertexList** Pointer to the **AEGfxVertexList** to be free.

Return values

void No return.

void AEGfxMeshStart ( )

Instruct AEGraphics to start creating a new mesh.

Call this function once before adding any points to the mesh.

Return values

void No return.

```
void AEGfxPoint ( f32  x0,
                  f32  y0,
                  f32  z0,
                  u32  c0
                )
```

Draw a point on screen.

The point is defined by x0, y0, z0 with color c0.

Warning

Do not call this function when AEGraphics is building mesh.

Function may be slow. Use with caution.

Parameters

[in] **x0** X coordinate of point

[in] **y0** Y coordinate of point

[in] **z0** Z coordinate of point

[in] **c0** Color value of point

Return values

void No return.

```
void AEGfxPrint ( s32  x,
                  s32  y,
                  u32  color,
                  s8 * pStr
                )
```

Render text onto the screen.

Render a null-terminated string of text (pStr) onto the screen starting at the point(x, y) using the color set.

Parameters

[in]	**x**	X coordinate to start rendering the text from. This is given in terms of actual windows coordinates.
[in]	**y**	X coordinate to start rendering the text from. This is given in terms of actual windows coordinates.
[in]	**color**	The color to render the text with.
[in]	**pStr**	Pointer to a NULL-terminated string containing the text to be rendered onto the screen.

Return values

void No return.

```
void AEGfxQuad ( f32  x0,
                 f32  y0,
                 f32  z0,
                 u32  c0,
                 f32  x1,
                 f32  y1,
                 f32  z1,
                 u32  c1,
                 f32  x2,
                 f32  y2,
                 f32  z2,
                 u32  c2,
                 f32  x3,
                 f32  y3,
                 f32  z3,
                 u32  c3
               )
```

Draw a rectangle on screen.

The rectangle is defined by 4 points: pt0, pt1, pt2 and pt3. The points are listed in a counter-clockwise order.

Warning

Do not call this function when AEGraphics is building mesh.

Function may be slow. Use with caution.

Parameters

[in] **x0** X coordinate of pt0
[in] **y0** Y coordinate of pt0
[in] **z0** Z coordinate of pt0
[in] **c0** Color value of pt0
[in] **x1** X coordinate of pt1
[in] **y1** Y coordinate of pt1
[in] **z1** Z coordinate of pt1
[in] **c1** Color value of pt1
[in] **x2** X coordinate of pt2
[in] **y2** Y coordinate of pt2
[in] **z2** Z coordinate of pt2
[in] **c2** Color value of pt2
[in] **x3** X coordinate of pt3
[in] **y3** Y coordinate of pt3
[in] **z3** Z coordinate of pt3
[in] **c3** Color value of pt3

Return values

void No return.

void AEGfxReset ( )

Reset AEGraphics.

Warning

This function is already called in AESysReset.

Return values

void No return.

void AEGfxSaveTextureToFile (AEGfxTexture * pTexture,
s8 * pFileName
)

Save a texture from memory to file.

Parameters

[in] **pTexture** Pointer to **AEGfxTexture** to be saved.
[in] **pFileName** Pointer to a null-terminated string containing the relative path of the file.

Return values

void No return.

void AEGfxSetBackgroundColor (f32 Red,
f32 Green,
f32 Blue
)

Set the background colour to selected RGB values.

Parameters

[in] **Red** Percentage of red. Range from 0.0f to 1.0f.
[in] **Green** Percentage of green. Range from 0.0f to 1.0f.
[in] **Blue** Percentage of blue. Range from 0.0f to 1.0f.

Return values

void No return.

void AEGfxSetBlendColor (f32 Red,
f32 Green,
f32 Blue,

f32 Alpha
)

Sets a color (RGBA) that will be used to blend with the original material.

Parameters

[in] **Red** Percentage of red. Range from 0.0f to 1.0f.
[in] **Green** Percentage of green. Range from 0.0f to 1.0f.
[in] **Blue** Percentage of blue. Range from 0.0f to 1.0f.
[in] **Alpha** Percentage of alpha (transparency). Range from 0.0f (clear) to 1.0f (opaque).

Return values

void No return.

void AEGfxSetBlendMode (AEGfxBlendMode BlendMode)

Set the blend mode.

Parameters

[in] **BlendMode** The AEGfxBlendMode to set.

Return values

void No return.

void AEGfxSetCamPosition (f32 X, f32 Y)

Set the camera's X and Y position.

Parameters

[in] **X** X position to set camera to.
[in] **Y** Y position to set camera to.

Return values
void No return.

void AEGfxSetPosition (f32 X, f32 Y)

Set the position (translation) to render from.

Parameters
[in] **X** X position to render from.
[out] **Y** Y position to render from.

Return values
void No return.

void AEGfxSetRenderMode (AEGfxRenderMode RenderMode)

Set the render mode.

Warning
This function should be called at least once per frame, before any rendering is done.

Parameters
[in] **RenderMode** The AEGfxRenderMode to set.

Return values
void No return.

void AEGfxSetTextureMode (AEGfxTextureMode TextureMode)

Set the texture mode.

Parameters

[in] **TextureMode** The AEGfxTextureMode to set.

Return values

void No return.

---

void AEGfxSetTintColor (float Red,
float Green,
float Blue,
float Alpha
)

Sets a color (RGBA) that will be used to tint the original material.

Parameters

[in] **Red** Percentage of red. Range from 0.0f to 1.0f.

[in] **Green** Percentage of green. Range from 0.0f to 1.0f.

[in] **Blue** Percentage of blue. Range from 0.0f to 1.0f.

[in] **Alpha** Percentage of alpha (transparency). Range from 0.0f (clear) to 1.0f (opaque).

Return values

void No return.

---

void AEGfxSetTransform (f32 pTransform[3][3])

Set the new global transformation matrix.

The new matrix will be applied to all object drawn after this function was called.

Parameters

[in] **pTransform** Pointer to a 3x3 matrix to set with. User may pass in a **AEMtx33.m**.

Return values

void No return.

void AEGfxSetTransform3D (f32 pTransform[4][4])

Set the new global transformation matrix.

The new matrix will be applied to all object drawn after this function was called.

Parameters

[in] **pTransform** Pointer to a 4x4 matrix to set with.

Return values

void No return.

void AEGfxSetTransparency (f32 Alpha)

Set the new global transparency value.

The new transparency value will be applied to all object drawn after this function was called.

Parameters

[in] **Alpha** Percentage of alpha (transparency). Range from 0.0f (clear) to 1.0f (opaque).

Return values

void No return.

void AEGfxSphere (f32 x0, f32 y0, f32 z0, f32 radius, u32 c0,

```
    u32 c1,
    u32 division
)
```

Draw a 3D sphere on screen.

The sphere is centered around the point (x0, y0, z0) with size radius. c0 and c1 are the sphere color. Default colors are white and grey. Division is the resolution, or how many parts the sphere consists of. Default value is 8.

Warning

Do not call this function when AEGraphics is building mesh.

Function may be slow. Use with caution.

Parameters

[in]	**x0**	X coordinate of the center of the sphere
[in]	**y0**	Y coordinate of the center of the sphere
[in]	**z0**	Z coordinate of the center of the sphere
[in]	**radius**	Radius of the sphere
[in]	**c0**	First color of the sphere.
[in]	**c1**	Second color of the sphere.
[in]	**division**	Number of parts to divide the sphere by.

Return values

void No return.

void AEGfxStart ( )

Tell AEGraphics that a new frame is starting.

Warning

This function is already called in AESysFrameStart.

Return values

void No return.

AEGfxTexture * AEGfxTextureLoad (const s8 * pFileName)

Load a texture file into memory.

Parameters

[in] **pFileName** Pointer to a null-terminated string containing the relative path of the file to be loaded.

Return values

AEGfxTexture* Returns a **AEGfxTexture** pointer to the loaded texture.

AEGfxTexture * AEGfxTextureLoadFromMemory (u8 * pColors,
u32 Width,
u32 Height
)

Load a texture from data in memory.

Parameters

[in] **pColors** Pointer to an array containing the data.

[in] **Width** The width of the texture.

[in] **Height** The height of the texture.

Return values

AEGfxTexture* Returns a **AEGfxTexture** pointer to the loaded texture.

void AEGfxTextureSet (AEGfxTexture * pTexture,
f32 offset_x,
f32 offset_y
)

Set a texture to be used for rendering.

Parameters

[in] **pTexture** Pointer to the **AEGfxTexture** to be used. Set to null if not using texture.

[in] **offset_x** X offset for the texture. Range from 0.0f to 1.0f.

[in] **offset_y** Y offset for the texture. Range from 0.0f to 1.0f.

Return values

void No return.

---

void AEGfxTextureUnload (AEGfxTexture * pTexture)

Unload a texture file from memory.

Parameters

[in] **pTexture** Pointer to **AEGfxTexture** to be unloaded.

Return values

void No return.

---

void AEGfxTri (f32 x0,
f32 y0,
f32 z0,
u32 c0,
f32 x1,
f32 y1,
f32 z1,
u32 c1,
f32 x2,
f32 y2,
f32 z2,
u32 c2

)

Draw a triangle on screen.

The triangle is defined by 3 points: pt0, pt1 and pt2. The points are listed in a counter-clockwise order.

Warning

Do not call this function when AEGraphics is building mesh.

Function may be slow. Use with caution.

Parameters

[in] **x0** X coordinate of pt0
[in] **y0** Y coordinate of pt0
[in] **z0** Z coordinate of pt0
[in] **c0** Color value of pt0
[in] **x1** X coordinate of pt1
[in] **y1** Y coordinate of pt1
[in] **z1** Z coordinate of pt1
[in] **c1** Color value of pt1
[in] **x2** X coordinate of pt2
[in] **y2** Y coordinate of pt2
[in] **z2** Z coordinate of pt2
[in] **c2** Color value of pt2

Return values

void No return.

```
void AEGfxTriAdd ( f32   x0,
                   f32   y0,
                   u32   c0,
                   f32   tu0,
                   f32   tv0,
                   f32   x1,
                   f32   y1,
```

```
        u32 c1,
        f32 tu1,
        f32 tv1,
        f32 x2,
        f32 y2,
        u32 c2,
        f32 tu2,
        f32 tv2
    )
```

Add a new triangle to the mesh.

The triangle is defined by 3 points: pt0, pt1 and pt2. The points are listed in a counter-clockwise order. The Z-value of the points are set to default 0.

Parameters

- [in] **x0** X coordinate of pt0.
- [in] **y0** Y coordinate of pt0.
- [in] **c0** Color value of pt0.
- [in] **tu0** Texture translation of pt0.
- [in] **tv0** Texture translation of pt0.
- [in] **x1** X coordinate of pt1.
- [in] **y1** Y coordinate of pt1.
- [in] **c1** Color value of pt1.
- [in] **tu1** Texture translation of pt1.
- [in] **tv1** Texture translation of pt1.
- [in] **x2** X coordinate of pt2.
- [in] **y2** Y coordinate of pt2.
- [in] **c2** Color value of pt2.
- [in] **tu2** Texture translation of pt2.
- [in] **tv2** Texture translation of pt2.

Return values

void No return.

void AEGfxVertexAdd (f32 x0,
f32 y0,
u32 c0,
f32 tu0,
f32 tv0
)

Add a new point to the mesh.

The Z-value of the point are set to default 0.

Parameters

[in] **x0** X coordinate of the point.
[in] **y0** Y coordinate of the point.
[in] **c0** Color value of the point.
[in] **tu0** Texture translation of the point.
[in] **tv0** Texture translation of the point.

Return values

void No return.

DECLARE_FUNCTION_FOR_ANDROID (AEGfxStart)

DECLARE_FUNCTION_FOR_ANDROID (AEGfxEnd)

DECLARE_FUNCTION_FOR_ANDROID_2_INT (AEGfxInit ,
s32 Width,
s32 Height
)

Alpha Engine

Macros | Functions

AEInput.h File Reference

Header file for the input library. More...

Go to the source code of this file.

Macros

#define **AEVK_LBUTTON** VK_LBUTTON
Left mouse button. More...

#define **AEVK_RBUTTON** VK_RBUTTON
Right mouse button. More...

#define **AEVK_MBUTTON** VK_MBUTTON
Middle mouse button. More...

#define **AEVK_BACK** VK_BACK
Backspace. More...

#define **AEVK_TAB** VK_TAB

#define **AEVK_RETURN** VK_RETURN

#define **AEVK_LSHIFT** VK_LSHIFT

#define **AEVK_RSHIFT** VK_RSHIFT

#define **AEVK_LCTRL** VK_LCONTROL

#define **AEVK_RCTRL** VK_RCONTROL

#define **AEVK_LALT** VK_LMENU

#define **AEVK_RALT** VK_RMENU

#define **AEVK_PRINTSCREEN** VK_SNAPSHOT

#define **AEVK_SCROLLLOCK** VK_SCROLL

#define **AEVK_PAUSE** VK_PAUSE

```
#define AEVK_CAPSLOCK VK_CAPITAL
#define AEVK_ESCAPE VK_ESCAPE
#define AEVK_SPACE VK_SPACE
#define AEVK_PAGEUP VK_PRIOR
#define AEVK_PAGEDOWN VK_NEXT
#define AEVK_END VK_END
#define AEVK_HOME VK_HOME
#define AEVK_LEFT VK_LEFT
#define AEVK_UP VK_UP
#define AEVK_RIGHT VK_RIGHT
#define AEVK_DOWN VK_DOWN
#define AEVK_INSERT VK_INSERT
#define AEVK_DELETE VK_DELETE
#define AEVK_0 0x30
#define AEVK_1 0x31
#define AEVK_2 0x32
#define AEVK_3 0x33
#define AEVK_4 0x34
#define AEVK_5 0x35
```

```
#define AEVK_6 0x36
#define AEVK_7 0x37
#define AEVK_8 0x38
#define AEVK_9 0x39
#define AEVK_A 0x41
#define AEVK_B 0x42
#define AEVK_C 0x43
#define AEVK_D 0x44
#define AEVK_E 0x45
#define AEVK_F 0x46
#define AEVK_G 0x47
#define AEVK_H 0x48
#define AEVK_I 0x49
#define AEVK_J 0x4A
#define AEVK_K 0x4B
#define AEVK_L 0x4C
#define AEVK_M 0x4D
#define AEVK_N 0x4E
```

```
#define AEVK_O 0x4F
#define AEVK_P 0x50
#define AEVK_Q 0x51
#define AEVK_R 0x52
#define AEVK_S 0x53
#define AEVK_T 0x54
#define AEVK_U 0x55
#define AEVK_V 0x56
#define AEVK_W 0x57
#define AEVK_X 0x58
#define AEVK_Y 0x59
#define AEVK_Z 0x5A
#define AEVK_NUMPAD0 VK_NUMPAD0
#define AEVK_NUMPAD1 VK_NUMPAD1
#define AEVK_NUMPAD2 VK_NUMPAD2
#define AEVK_NUMPAD3 VK_NUMPAD3
#define AEVK_NUMPAD4 VK_NUMPAD4
#define AEVK_NUMPAD5 VK_NUMPAD5
#define AEVK_NUMPAD6 VK_NUMPAD6
```

```
#define AEVK_NUMPAD7 VK_NUMPAD7

#define AEVK_NUMPAD8 VK_NUMPAD8
#define AEVK_NUMPAD9 VK_NUMPAD9
#define AEVK_NUM_MULTIPLY VK_MULTIPLY
#define AEVK_NUM_PLUS VK_ADD
#define AEVK_NUM_MINUS VK_SUBTRACT
#define AEVK_NUM_PERIOD VK_DECIMAL
#define AEVK_NUM_DIVIDE VK_DIVIDE
#define AEVK_NUMLOCK VK_NUMLOCK
#define AEVK_F1 VK_F1
#define AEVK_F2 VK_F2
#define AEVK_F3 VK_F3
#define AEVK_F4 VK_F4
#define AEVK_F5 VK_F5
#define AEVK_F6 VK_F6
#define AEVK_F7 VK_F7
#define AEVK_F8 VK_F8
#define AEVK_F9 VK_F9
```

#define **AEVK_F10** VK_F10

#define **AEVK_F11** VK_F11

#define **AEVK_F12** VK_F12

#define **AEVK_SEMICOLON** VK_OEM_1
; : More...

#define **AEVK_SLASH** VK_OEM_2
/ ? More...

#define **AEVK_BACKQUOTE** VK_OEM_3
` ~ More...

#define **AEVK_LBRACKET** VK_OEM_4
[{ More...

#define **AEVK_BACKSLASH** VK_OEM_5
\ | More...

#define **AEVK_RBRACKET** VK_OEM_6
] } More...

#define **AEVK_QUOTE** VK_OEM_7
' " More...

#define **AEVK_EQUAL** VK_OEM_PLUS
= + More...

#define **AEVK_MINUS** VK_OEM_MINUS

#define **AEVK_PERIOD** VK_OEM_PERIOD
. > More...

#define **AEVK_COMMA** VK_OEM_COMMA
, < More...

Functions

s32 **AEInputInit** ()
Initialize AEInput. More...

void **AEInputReset** ()
Reset AEInput. More...

void **AEInputUpdate** ()
Update AEInput. More...

void **AEInputExit** ()
Free AEInput. More...

u8 **AEInputCheckCurr** (**u8** key)
Check if a keyboard button is being pressed. More...

u8 **AEInputCheckPrev** (**u8** key)
Check if a keyboard button was pressed recently. More...

u8 **AEInputCheckTriggered** (**u8** key)
Check if a keyboard button was triggered. More...

u8 **AEInputCheckReleased** (**u8** key)
Check if a keyboard button was released. More...

void **AEInputGetCursorPosition** (**s32** *pX, **s32** *pY)
Get the current position of the cursor, in screen coordinates. More...

void **AEInputGetCursorPositionDelta** (**s32** *pDeltaX, **s32** *pDeltaY)
Get the change in position of the cursor since the last update, in screen coordinates. More...

void **AEInputShowCursor** (**s32** Show)

Set if mouse cursor is showned. More...

Detailed Description

Header file for the input library.

Project: Alpha Engine

Author
Sun Tjen Fam

Date
January 31, 2008

Definition in file **AEInput.h**.

Macro Definition Documentation

#define AEVK_0 0x30

Definition at line **59** of file **AEInput.h**.

#define AEVK_1 0x31

Definition at line **60** of file **AEInput.h**.

#define AEVK_2 0x32

Definition at line **61** of file **AEInput.h**.

#define AEVK_3 0x33

Definition at line **62** of file **AEInput.h**.

#define AEVK_4 0x34

Definition at line **63** of file **AEInput.h**.

#define AEVK_5 0x35

Definition at line **64** of file **AEInput.h**.

#define AEVK_6 0x36

Definition at line 65 of file AEInput.h.

#define AEVK_7 0x37

Definition at line 66 of file AEInput.h.

#define AEVK_8 0x38

Definition at line 67 of file AEInput.h.

#define AEVK_9 0x39

Definition at line 68 of file AEInput.h.

#define AEVK_A 0x41

Definition at line 70 of file AEInput.h.

#define AEVK_B 0x42

Definition at line 71 of file AEInput.h.

#define AEVK_BACK VK_BACK

Backspace.

Definition at line 31 of file AEInput.h.

#define AEVK_BACKQUOTE VK_OEM_3

` ~

Definition at line 130 of file AEInput.h.

#define AEVK_BACKSLASH VK_OEM_5

\ |

Definition at line 132 of file AEInput.h.

#define AEVK_C 0x43

Definition at line 72 of file AEInput.h.

#define AEVK_CAPSLOCK VK_CAPITAL

Definition at line 43 of file AEInput.h.

#define AEVK_COMMA VK_OEM_COMMA

, <

Definition at line 139 of file AEInput.h.

#define AEVK_D 0x44

Definition at line 73 of file AEInput.h.

#define AEVK_DELETE VK_DELETE

Definition at line **57** of file **AEInput.h**.

#define AEVK_DOWN VK_DOWN

Definition at line **55** of file **AEInput.h**.

#define AEVK_E 0x45

Definition at line **74** of file **AEInput.h**.

#define AEVK_END VK_END

Definition at line **50** of file **AEInput.h**.

#define AEVK_EQUAL VK_OEM_PLUS

= +

Definition at line **136** of file **AEInput.h**.

#define AEVK_ESCAPE VK_ESCAPE

Definition at line **45** of file **AEInput.h**.

#define AEVK_F 0x46

Definition at line **75** of file **AEInput.h**.

#define AEVK_F1 VK_F1

Definition at line **115** of file **AEInput.h**.

#define AEVK_F10 VK_F10

Definition at line **124** of file **AEInput.h**.

#define AEVK_F11 VK_F11

Definition at line **125** of file **AEInput.h**.

#define AEVK_F12 VK_F12

Definition at line **126** of file **AEInput.h**.

#define AEVK_F2 VK_F2

Definition at line **116** of file **AEInput.h**.

#define AEVK_F3 VK_F3

Definition at line **117** of file **AEInput.h**.

#define AEVK_F4 VK_F4

Definition at line **118** of file **AEInput.h**.

#define AEVK_F5 VK_F5

Definition at line **119** of file **AEInput.h**.

#define AEVK_F6 VK_F6

Definition at line **120** of file **AEInput.h**.

#define AEVK_F7 VK_F7

Definition at line **121** of file **AEInput.h**.

#define AEVK_F8 VK_F8

Definition at line **122** of file **AEInput.h**.

#define AEVK_F9 VK_F9

Definition at line **123** of file **AEInput.h**.

#define AEVK_G 0x47

Definition at line **76** of file **AEInput.h**.

#define AEVK_H 0x48

Definition at line **77** of file **AEInput.h**.

#define AEVK_HOME VK_HOME

Definition at line **51** of file **AEInput.h**.

#define AEVK_I 0x49

Definition at line 78 of file AEInput.h.

#define AEVK_INSERT VK_INSERT

Definition at line 56 of file AEInput.h.

#define AEVK_J 0x4A

Definition at line 79 of file AEInput.h.

#define AEVK_K 0x4B

Definition at line 80 of file AEInput.h.

#define AEVK_L 0x4C

Definition at line 81 of file AEInput.h.

#define AEVK_LALT VK_LMENU

Definition at line 38 of file AEInput.h.

#define AEVK_LBRACKET VK_OEM_4

[{

Definition at line 131 of file AEInput.h.

#define AEVK_LBUTTON VK_LBUTTON

Left mouse button.

Definition at line **27** of file **AEInput.h**.

#define AEVK_LCTRL VK_LCONTROL

Definition at line **36** of file **AEInput.h**.

#define AEVK_LEFT VK_LEFT

Definition at line **52** of file **AEInput.h**.

#define AEVK_LSHIFT VK_LSHIFT

Definition at line **34** of file **AEInput.h**.

#define AEVK_M 0x4D

Definition at line **82** of file **AEInput.h**.

#define AEVK_MBUTTON VK_MBUTTON

Middle mouse button.

Definition at line **29** of file **AEInput.h**.

#define AEVK_MINUS VK_OEM_MINUS

- _

Definition at line 137 of file AEInput.h.

#define AEVK_N 0x4E

Definition at line 83 of file AEInput.h.

#define AEVK_NUM_DIVIDE VK_DIVIDE

Definition at line 112 of file AEInput.h.

#define AEVK_NUM_MINUS VK_SUBTRACT

Definition at line 110 of file AEInput.h.

#define AEVK_NUM_MULTIPLY VK_MULTIPLY

Definition at line 108 of file AEInput.h.

#define AEVK_NUM_PERIOD VK_DECIMAL

Definition at line 111 of file AEInput.h.

#define AEVK_NUM_PLUS VK_ADD

Definition at line 109 of file AEInput.h.

#define AEVK_NUMLOCK VK_NUMLOCK

Definition at line 113 of file AEInput.h.

#define AEVK_NUMPAD0 VK_NUMPAD0

Definition at line **97** of file **AEInput.h**.

#define AEVK_NUMPAD1 VK_NUMPAD1

Definition at line **98** of file **AEInput.h**.

#define AEVK_NUMPAD2 VK_NUMPAD2

Definition at line **99** of file **AEInput.h**.

#define AEVK_NUMPAD3 VK_NUMPAD3

Definition at line **100** of file **AEInput.h**.

#define AEVK_NUMPAD4 VK_NUMPAD4

Definition at line **101** of file **AEInput.h**.

#define AEVK_NUMPAD5 VK_NUMPAD5

Definition at line **102** of file **AEInput.h**.

#define AEVK_NUMPAD6 VK_NUMPAD6

Definition at line **103** of file **AEInput.h**.

#define AEVK_NUMPAD7 VK_NUMPAD7

Definition at line **104** of file **AEInput.h**.

#define AEVK_NUMPAD8 VK_NUMPAD8

Definition at line **105** of file **AEInput.h**.

#define AEVK_NUMPAD9 VK_NUMPAD9

Definition at line **106** of file **AEInput.h**.

#define AEVK_O 0x4F

Definition at line **84** of file **AEInput.h**.

#define AEVK_P 0x50

Definition at line **85** of file **AEInput.h**.

#define AEVK_PAGEDOWN VK_NEXT

Definition at line **49** of file **AEInput.h**.

#define AEVK_PAGEUP VK_PRIOR

Definition at line **48** of file **AEInput.h**.

#define AEVK_PAUSE VK_PAUSE

Definition at line **42** of file **AEInput.h**.

#define AEVK_PERIOD VK_OEM_PERIOD

. >

Definition at line **138** of file **AEInput.h**.

#define AEVK_PRINTSCREEN VK_SNAPSHOT

Definition at line **40** of file **AEInput.h**.

#define AEVK_Q 0x51

Definition at line **86** of file **AEInput.h**.

#define AEVK_QUOTE VK_OEM_7

' "

Definition at line **134** of file **AEInput.h**.

#define AEVK_R 0x52

Definition at line **87** of file **AEInput.h**.

#define AEVK_RALT VK_RMENU

Definition at line **39** of file **AEInput.h**.

#define AEVK_RBRACKET VK_OEM_6

] }

Definition at line **133** of file **AEInput.h**.

#define AEVK_RBUTTON VK_RBUTTON

Right mouse button.

Definition at line **28** of file **AEInput.h**.

#define AEVK_RCTRL VK_RCONTROL

Definition at line **37** of file **AEInput.h**.

#define AEVK_RETURN VK_RETURN

Definition at line **33** of file **AEInput.h**.

#define AEVK_RIGHT VK_RIGHT

Definition at line **54** of file **AEInput.h**.

#define AEVK_RSHIFT VK_RSHIFT

Definition at line **35** of file **AEInput.h**.

#define AEVK_S 0x53

Definition at line **88** of file **AEInput.h**.

#define AEVK_SCROLLLOCK VK_SCROLL

Definition at line 41 of file AEInput.h.

#define AEVK_SEMICOLON VK_OEM_1

; :

Definition at line 128 of file AEInput.h.

#define AEVK_SLASH VK_OEM_2

/ ?

Definition at line 129 of file AEInput.h.

#define AEVK_SPACE VK_SPACE

Definition at line 47 of file AEInput.h.

#define AEVK_T 0x54

Definition at line 89 of file AEInput.h.

#define AEVK_TAB VK_TAB

Definition at line 32 of file AEInput.h.

#define AEVK_U 0x55

Definition at line 90 of file AEInput.h.

#define AEVK_UP VK_UP

Definition at line **53** of file **AEInput.h**.

#define AEVK_V 0x56

Definition at line **91** of file **AEInput.h**.

#define AEVK_W 0x57

Definition at line **92** of file **AEInput.h**.

#define AEVK_X 0x58

Definition at line **93** of file **AEInput.h**.

#define AEVK_Y 0x59

Definition at line **94** of file **AEInput.h**.

#define AEVK_Z 0x5A

Definition at line **95** of file **AEInput.h**.

Function Documentation

u8 AEInputCheckCurr (u8 key)

Check if a keyboard button is being pressed.

Check if keyboard button given by key is currently being pressed.

Parameters

[in] **key** Value corresponding to the keyboard button that is being checked for.

Return values

u8 Return true if keyboard button is being pressed. Else return false.

u8 AEInputCheckPrev (u8 key)

Check if a keyboard button was pressed recently.

Check if keyboard button given by key was pressed in the previous update.

Parameters

[in] **key** Value corresponding to the keyboard button that is being checked for.

Return values

u8 Return true if keyboard button was pressed. Else return false.

u8 AEInputCheckReleased (u8 key)

Check if a keyboard button was released.

Check if the keyboard button given by key was recently released.

Parameters

[in] **key** Value corresponding to the keyboard button that is being checked for.

Return values

u8 Return true if keyboard button was pressed. Else return false.

u8 AEInputCheckTriggered (u8 key)

Check if a keyboard button was triggered.

Check if the keyboard button given by key was recently triggerd (pressed and then quickly released).

Parameters

[in] **key** Value corresponding to the keyboard button that is being checked for.

Return values

u8 Return true if keyboard button was pressed. Else return false.

void AEInputExit ( )

Free AEInput.

Warning

This function is already called in AESysExit.

Return values

void No return.

void AEInputGetCursorPosition (s32 * pX,
s32 * pY
)

Get the current position of the cursor, in screen coordinates.

Parameters

[out] **pX** Pointer to s32 where the X value will be stored.
[out] **pY** Pointer to s32 where the Y value will be stored.

Return values

void No return.

void AEInputGetCursorPositionDelta (s32 * pDeltaX,
s32 * pDeltaY
)

Get the change in position of the cursor since the last update, in screen coordinates.

Parameters

[out] **pDeltaX** Pointer to s32 where the X value will be stored.
[out] **pDeltaY** Pointer to s32 where the Y value will be stored.

Return values

void No return.

s32 AEInputInit ( )

Initialize AEInput.

Warning

This function is already called in AESysInit.

Return values

s32 Return 1 if initialization is successful. Else return 0.

void AEInputReset ( )

Reset AEInput.

Warning
This function is already called in AESysReset.

Return values
void No return.

void AEInputShowCursor (s32 Show)

Set if mouse cursor is showned.

Setting true will allow mouse cursor to be rendered on screen.
Setting false will hide the mouse cursor.

Parameters
[in] **Show** Set if mouse cursor is showned or not.

Return values
void No return.

void AEInputUpdate ( )

Update AEInput.

Warning
This function is already called in AESysUpdate.

Return values
void No return.

Alpha Engine

Classes | Typedefs | Functions

AELineSegment2.h File Reference

Header file for the 2D line segment library. More...

Go to the source code of this file.

Classes

struct	**AELineSegment2**

Typedefs

typedef struct **AELineSegment2** **AELineSegment2**

Functions

s32	**AEBuildLineSegment2** (**AELineSegment2** *pLS, **AEVec2** *pPt0, **AEVec2** *pPt1) Initalize pLS using pPt0 and pPt1. Also computes the normal (unit vector) and the dot product of the normal with one of the points. More...

Detailed Description

Header file for the 2D line segment library.

Project: Alpha Engine

Author
Antoine Abi Chacra

Date
March 19, 2012

Definition in file **AELineSegment2.h**.

Typedef Documentation

typedef struct AELineSegment2 AELineSegment2

Function Documentation

```
s32 AEBuildLineSegment2 ( AELineSegment2 *  pLS,
                          AEVec2 *          pPt0,
                          AEVec2 *          pPt1
                        )
```

Initalize pLS using pPt0 and pPt1. Also computes the normal (unit vector) and the dot product of the normal with one of the points.

Warning

The distance between pPt0 and pPt1 should be greater than EPSILON.

Parameters

`[out]` **pLS** Pointer to **AELineSegment2** to be set.
`[in]` **pPt0** Pointer to **AEVec2** for input.
`[in]` **pPt1** Pointer to **AEVec2** for input.

Return values

s32 Returns 1 if pLS is successfully intialized. Else returns 0.

Alpha Engine

Functions

AEMath.h File Reference

Header file for the math library. More...

```
#include "AEVec2.h" #include "AEMtx33.h"
#include "AELineSegment2.h"
#include <float.h>
#include "math.h"
```

Go to the source code of this file.

Macros

#define **AESinDeg**(x) **AESin**(**AEDegToRad**(x))
Macros for the trigo functions that take input in degree.
More...

#define **AECosDeg**(x) **AECos**(**AEDegToRad**(x))
Macros for the trigo functions that take input in degree.
More...

#define **AETanDeg**(x) **AETan**(**AEDegToRad**(x))
Macros for the trigo functions that take input in degree.
More...

#define **AEASinDeg**(x) **AERadToDeg**(**AEASin**(x))
Macros for the trigo functions that take input in degree.
More...

#define **AEACosDeg**(x) **AERadToDeg**(**AEACos**(x))
Macros for the trigo functions that take input in degree.
More...

#define **AEATanDeg**(x) **AERadToDeg**(**AEATan**(x))
Macros for the trigo functions that take input in degree.
More...

Functions

f32 **AEDegToRad** (**f32** x)
Convert an angle from Degree to Radians. More...

f32 **AERadToDeg** (**f32** x)
Convert an angle from Radians to Degree. More...

f32 **AESin** (**f32** x)
Calculate the Sine value of an angle. More...

f32 **AECos** (**f32** x)
Calculate the Cosine value of an angle. More...

f32 **AETan** (**f32** x)
Calculate the Tangent value of an angle. More...

f32 **AEASin** (**f32** x)
Calculate the ArcSine value of an angle. More...

f32 **AEACos** (**f32** x)
Calculate the ArcCosine value of an angle. More...

f32 **AEATan** (**f32** x)
Calculate the ArcTangent value of an angle. More...

u32 **AEIsPowOf2** (**u32** x)
Check if x is a power of 2. More...

u32 **AENextPowOf2** (**u32** x)
Calculate the next power of 2 that is greater than x. More...

u32 **AELogBase2** (**u32** x)
Calculate the log2 of x. More...

f32 **AEClamp** (**f32** X, **f32** Min, **f32** Max)
Clamp x to between x0 and x1. More...

f32 **AEWrap** (**f32** x, **f32** x0, **f32** x1)
Wraparound for x with respect to range (x0 to x1). More...

f32 **AEMin** (**f32** x, **f32** y)
Find which of the 2 value is lower. More...

f32 **AEMax** (**f32** x, **f32** y)
Find which of the 2 value is higher. More...

s32 **AEInRange** (**f32** x, **f32** x0, **f32** x1)
Find if x is in the range (x0 to x1), inclusive. More...

f32 **AECalcDistPointToCircle** (**AEVec2** *pPos, **AEVec2** *pCtr, **f32** radius)
Calculate the shortest distance from a point to the edge of a circle. More...

f32 **AECalcDistPointToRect** (**AEVec2** *pPos, **AEVec2** *pRect, **f32** sizeX, **f32** sizeY)
Calculate the shortest distance from a point to the edge of a rectangle. More...

f32 **AECalcDistPointToLineSeg** (**AEVec2** *pPos, **AEVec2** *pLine0, **AEVec2** *pLine1)
Calculate the shortest distance from a point to a line segment. More...

f32 **AECalcDistPointToConvexPoly** (**AEVec2** *pPos, **AEVec2** *pVtx, **u32** vtxNum)
Calculate the shortest distance from a point to the edge of a convex polygon. More...

f32 **AECalcDistCircleToCircle** (**AEVec2** *pCtr0, **f32** radius0, **AEVec2** *pCtr1, **f32** radius1)

Calculate the shortest distance between the edges of two circles. More...

f32 **AECalcDistCircleToRect** (**AEVec2** *pCtr, **f32** radius, **AEVec2** *pRect, **f32** sizeX, **f32** sizeY)
Calculate the shortest distance between the edges of a circle and a rectangle. More...

f32 **AECalcDistRectToRect** (**AEVec2** *pRect0, **f32** sizeX0, **f32** sizeY0, **AEVec2** *pRect1, **f32** sizeX1, **f32** sizeY1, **AEVec2** *pNormal)
Calculate the shortest distance between the edges of two rectangles. More...

s32 **AETestPointToCircle** (**AEVec2** *pPos, **AEVec2** *pCtr, **f32** radius)
Test if a point is inside a circle. More...

s32 **AETestPointToRect** (**AEVec2** *pPos, **AEVec2** *pRect, **f32** sizeX, **f32** sizeY)
Test if a point is inside a rectangle. More...

s32 **AETestCircleToCircle** (**AEVec2** *pCtr0, **f32** radius0, **AEVec2** *pCtr1, **f32** radius1)
Test for collision between two circles. More...

s32 **AETestCircleToRect** (**AEVec2** *pCtr, **f32** radius, **AEVec2** *pRect, **f32** sizeX, **f32** sizeY)
Test for collision between a circle and a rectangle. More...

s32 **AETestRectToRect** (**AEVec2** *pRect0, **f32** sizeX0, **f32** sizeY0, **AEVec2** *pRect1, **f32** sizeX1, **f32** sizeY1)
Test for collision between two rectangles. More...

f32 **AEStaticPointToStaticLineSegment** (**AEVec2** *pPos, **AELineSegment2** *pLine)
Calculate the shortest distance from a point to a line. More...

f32 **AEAnimatedPointToStaticLineSegment** (**AEVec2** *pStart, **AEVec2** *pEnd, **AELineSegment2** *pLine, **AEVec2** *pInter)

Calculate the collision between a moving point with a line. More...

f32 **AEAnimatedCircleToStaticLineSegment** (**AEVec2** *pStart, **AEVec2** *pEnd, **f32** radius, **AELineSegment2** *pLine, **AEVec2** *pInter)
Calculate the collision between a moving circle with a line. More...

f32 **AEReflectAnimatedPointOnStaticLineSegment** (**AEVec2** *pStart, **AEVec2** *pEnd, **AELineSegment2** *pLine, **AEVec2** *pInter, **AEVec2** *pReflect)
Calculate the collision between a moving point with a line and the reflected path of the point. More...

f32 **AEReflectAnimatedCircleOnStaticLineSegment** (**AEVec2** *pStart, **AEVec2** *pEnd, **f32** radius, **AELineSegment2** *pLine, **AEVec2** *pInter, **AEVec2** *pReflect)
Calculate the collision between a moving circle with a line and the reflected path of the circle. More...

f32 **AEAnimatedPointToStaticCircle** (**AEVec2** *pStart, **AEVec2** *pEnd, **AEVec2** *pCtr, **f32** radius, **AEVec2** *pInter)
Calculate the collision between a moving point with a circle. More...

f32 **AEReflectAnimatedPointOnStaticCircle** (**AEVec2** *pStart, **AEVec2** *pEnd, **AEVec2** *pCtr, **f32** radius, **AEVec2** *pInter, **AEVec2** *pReflect)
Calculate the collision between a moving point with a circle and the reflected path of the point. More...

f32 **AEAnimatedCircleToStaticCircle** (**AEVec2** *pCtr0s, **AEVec2** *pCtr0e, **f32** radius0, **AEVec2** *pCtr1, **f32** radius1, **AEVec2**

	*pInter) Calculate the collision between a moving circle with a static circle. More...
f32	**AEReflectAnimatedCircleOnStaticCircle** (**AEVec2** *pCtr0s, **AEVec2** *pCtr0e, **f32** radius0, **AEVec2** *pCtr1, **f32** radius1, **AEVec2** *pInter, **AEVec2** *pReflect) Calculate the collision between a moving circle with a static circle and the reflected path of moving circle. More...

Detailed Description

Header file for the math library.

Project: Alpha Engine

Author
Sun Tjen Fam

Date
January 31, 2008

Copyright

Definition in file **AEMath.h**.

Macro Definition Documentation

#define AEACosDeg (x) AERadToDeg(AEACos(x))

Macros for the trigo functions that take input in degree.

Definition at line 169 of file AEMath.h.

#define AEASinDeg (x) AERadToDeg(AEASin(x))

Macros for the trigo functions that take input in degree.

Definition at line 168 of file AEMath.h.

#define AEATanDeg (x) AERadToDeg(AEATan(x))

Macros for the trigo functions that take input in degree.

Definition at line 170 of file AEMath.h.

#define AECosDeg (x) AECos(AEDegToRad(x))

Macros for the trigo functions that take input in degree.

Definition at line 166 of file AEMath.h.

#define AESinDeg (x) AESin(AEDegToRad(x))

Macros for the trigo functions that take input in degree.

Definition at line **165** of file **AEMath.h**.

#define AETanDeg (x) AETan(AEDegToRad(x))

Macros for the trigo functions that take input in degree.

Definition at line **167** of file **AEMath.h**.

Function Documentation

f32 AEACos (f32 x)

Calculate the ArcCosine value of an angle.

Parameters

[in] **x** The angle in Radians.

Return values

f32 The value of acos(x).

f32 AEAnimatedCircleToStaticCircle (AEVec2 * pCtr0s, AEVec2 * pCtr0e, f32 radius0, AEVec2 * pCtr1, f32 radius1, AEVec2 * pInter)

Calculate the collision between a moving circle with a static circle.

Given the start position (pCtr0s) and end position (pCtr0e) of moving circle0 of size (radius0) and the center of static circle1 (pCtr1) of size (radius1), calculate the time and point of intersection.

Warning

Radius of circles should be a non-negative value.

Parameters

[in]	**pCtr0s**	Pointer to **AEVec2** containing start pos of circle0.
[in]	**pCtr0e**	Pointer to **AEVec2** containing end pos of the

circle0.

[in]	**radius0**	The radius of circle0.
[in]	**pCtr1**	Pointer to **AEVec2** containing the center of circle1.
[in]	**radius1**	The radius of the circle1.
[out]	**pInter**	Pointer to **AEVec2** for storing the point of intersection. Will not be used if there is no collision.

Return values

f32 Returns the time of collision, if there is one. Else return -1.0f.

---

f32
AEAnimatedCircleToStaticLineSegment (**AEVec2 *** **pStar**
AEVec2 * **pEnd**
f32 **radiu**
AELineSegment2 * **pLine**
AEVec2 * **pInte**
)

Calculate the collision between a moving circle with a line.

Given the start position (pStart) and end position (pEnd) of a moving circle of size (radius) and a line (pLine), calculate the time and point of intersection.

Warning

Line is assumed to stretch to infinity.

Radius of circle should be a non-negative value.

Parameters

[in]	**pStart**	Pointer to **AEVec2** containing start pos of the circle.
[in]	**pEnd**	Pointer to **AEVec2** containing end pos of the circle.
[in]	**radius**	The radius of the circle.
[in]	**pLine**	Pointer to **AELineSegment2** containing the line.

[out] **pInter** Pointer to **AEVec2** for storing the point of intersection. Will not be used if there is no collision.

Return values

f32 Returns the time of collision, if there is one. Else return -1.0f.

```
f32 AEAnimatedPointToStaticCircle ( AEVec2 *  pStart,
                                    AEVec2 *  pEnd,
                                    AEVec2 *  pCtr,
                                    f32       radius,
                                    AEVec2 *  pInter
                                  )
```

Calculate the collision between a moving point with a circle.

Given the start position (pStart) and end position (pEnd) of a moving point and the center of a circle (pCtr) of size (radius), calculate the time and point of intersection.

Warning

Radius of circle should be a non-negative value.

Parameters

[in] **pStart** Pointer to **AEVec2** containing start pos of the point.

[in] **pEnd** Pointer to **AEVec2** containing end pos of the point.

[in] **pCtr** Pointer to **AEVec2** containing the center of the circle.

[in] **radius** The radius of the circle.

[out] **pInter** Pointer to **AEVec2** for storing the point of intersection. Will not be used if there is no collision.

Return values

f32 Returns the time of collision, if there is one. Else return -1.0f.

```
f32
AEAnimatedPointToStaticLineSegment ( AEVec2 *          pStart
                                     AEVec2 *          pEnd,
                                     AELineSegment2 *  pLine
                                     AEVec2 *          pInter
                                   )
```

Calculate the collision between a moving point with a line.

Given the start position (pStart) and end position (pEnd) of a moving point and a line (pLine), calculate the time and point of intersection.

Warning
Line is assumed to stretch to infinity.

Parameters

[in]	**pStart**	Pointer to **AEVec2** containing start pos of the point.
[in]	**pEnd**	Pointer to **AEVec2** containing end pos of the point.
[in]	**pLine**	Pointer to **AELineSegment2** containing the line.
[out]	**pInter**	Pointer to **AEVec2** for storing the point of intersection. Will not be used if there is no collision.

Return values

f32 Returns the time of collision, if there is one. Else return -1.0f.

f32 AEASin (f32 x)

Calculate the ArcSine value of an angle.

Parameters

[in] **x** The angle in Radians.

Return values

f32 The value of asin(x).

f32 AEATan (f32 x)

Calculate the ArcTangent value of an angle.

Parameters

[in] **x** The angle in Radians.

Return values

f32 The value of atan(x).

f32 AECalcDistCircleToCircle (AEVec2 * pCtr0, f32 radius0, AEVec2 * pCtr1, f32 radius1)

Calculate the shortest distance between the edges of two circles.

Given the center of circle0 (pCtr0) of size (radius0) and the center of circle1 (pCtr1) of size (radius1), calculate the shortest distance between their edges.

Warning

radius of circles should be a non-negative value.

Parameters

[in] **pCtr0** Pointer to **AEVec2** containing the center of circle0.

[in] **radius0** The radius of circle0.

[in] **pCtr1** Pointer to **AEVec2** containing the center of circle1.

[in] **radius1** The radius of circle1.

Return values

f32 Returns the shortest distance between the edge of both circles. This value will be negative if the circles are

overlapping.

```
f32 AECalcDistCircleToRect ( AEVec2 *  pCtr,
                             f32       radius,
                             AEVec2 *  pRect,
                             f32       sizeX,
                             f32       sizeY
                           )
```

Calculate the shortest distance between the edges of a circle and a rectangle.

Given the center of a circle (pCtr) of size (radius) and the center of a rect (pRect) of width (sizeX) and height (sizeY), calculate the shortest distance between their edges.

Warning

radius of circles should be a non-negative value.

The width and height of the rect should be non-negative values.

Function will only work if the rect is not rotated, i.e. the sides of the rect are parallel to the axis.

Parameters

[in]	**pCtr**	Pointer to **AEVec2** containing the center of the circle.
[in]	**radius**	The radius of circle.
[in]	**pRect**	Pointer to **AEVec2** containing the center of the rect.
[in]	**sizeX**	The width of the rect.
[in]	**sizeY**	The height of the rect.

Return values

f32 Returns the shortest distance between the edges of the circle and the rect. This value will be negative if the circle and rect are overlapping.

```
f32 AECalcDistPointToCircle ( AEVec2 *  pPos,
                              AEVec2 *  pCtr,
                              f32       radius
                            )
```

Calculate the shortest distance from a point to the edge of a circle.

Given a point (pPos) and the center of a circle (pCtr) of size (radius), calculate the shortest distance from the point to the circumference of the circle.

Warning

Radius of circle should be a non-negative value.

Parameters

[in]	**pPos**	Pointer to **AEVec2** containing the positon of the point.
[in]	**pCtr**	Pointer to **AEVec2** containing the position of the center of the circle
[in]	**radius**	The radius of the circle.

Return values

f32 Returns the shortest distance from the point to the edge of the circle. This value will be negative if the point is inside the circle.

```
f32 AECalcDistPointToConvexPoly ( AEVec2 *  pPos,
                                  AEVec2 *  pVtx,
                                  u32       vtxNum
                                )
```

Calculate the shortest distance from a point to the edge of a convex polygon.

Given a point (pPos) and an array of vertices (pVtx) of size (vtxNum) making up a convex polygon, calculate the shortest distance from the point to the edge of the polygon.

Warning
Function not implemented. Do not use. Currently returns -1.0f.

Parameters

[in] **pPos** Pointer to **AEVec2** containing the position of the point.

[in] **pVtx** Pointer to an array of **AEVec2** containing the vertices of the polygon.

[in] **vtxNum** The number of vertices in the polygon

Return values

f32 Returns the shortest distance from the point to the edge of the polygon. This value will be negative if the point is inside the polygon.

---

f32 AECalcDistPointToLineSeg (AEVec2 * pPos,
AEVec2 * pLine0,
AEVec2 * pLine1
)

Calculate the shortest distance from a point to a line segment.

Given a point (pPos) and start (pLine0) and end (pLine1) of a line segment, calculate the shortest distance from the point to the line segment.

Parameters

[in] **pPos** Pointer to **AEVec2** containing the position of the point.

[in] **pLine0** Pointer to **AEVec2** containing the start of the line segment.

[in] **pLine1** Pointer to **AEVec2** containing the end of the line segment.

Return values

f32 Returns the shortest distance from the point to the line segment.

```
f32 AECalcDistPointToRect ( AEVec2 *  pPos,
                            AEVec2 *  pRect,
                            f32       sizeX,
                            f32       sizeY
                          )
```

Calculate the shortest distance from a point to the edge of a rectangle.

Given a point (pPos) and the center of a rect (pRect) of width (sizeX) and height (sizeY), calculate the shortest distance from the point to the perimeter of the rect.

Warning

The width and height of the rect should be non-negative values.

Function will only work if the rect is not rotated, i.e. the sides of the rect are parallel to the axis.

Parameters

- `[in]` **pPos** Pointer to **AEVec2** containing the position of the point.
- `[in]` **pRect** Pointer to **AEVec2** containing the position of the center of the rect.
- `[in]` **sizeX** Width of the rect, i.e. the size of the rect along the x-axis.
- `[in]` **sizeY** Height of the rect, i.e. the size of the rect along the y-axis.

Return values

f32 Returns the shortest distance from the point to the edge of the rect. This value will be negative if the point is inside the rect.

```
f32 AECalcDistRectToRect ( AEVec2 *  pRect0,
                           f32       sizeX0,
```

```
    f32         sizeY0,
    AEVec2 *    pRect1,
    f32         sizeX1,
    f32         sizeY1,
    AEVec2 *    pNormal
)
```

Calculate the shortest distance between the edges of two rectangles.

Given the center of rect0 (pRect0) of width (sizeX0) and height (sizeY0) and the center of rect1 (pRect1) of width (sizeX1) and height (sizeY1), calculate the shortest distance between their edges.

Warning

The width and height of the rect should be non-negative values.

Function will only work if the rect is not rotated, i.e. the sides of the rect are parallel to the axis.

Parameters

[in]	**pRect0**	Pointer to **AEVec2** containing the center of rect0.
[in]	**sizeX0**	The width of rect0.
[in]	**sizeY0**	The height of rect0.
[in]	**pRect1**	Pointer to **AEVec2** containing the center of rect1.
[in]	**sizeX1**	The width of rect1.
[in]	**sizeY1**	The height of rect1.
[out]	**pNormal**	Pointer to **AEVec2** where the direction from rect1 to rect0 is stored. May be left null if not needed.

Return values

f32 Returns the shortest distance between the edges of both rects. This value will be negative if they are overlapping.

```
f32 AEClamp ( f32  x,
              f32  x0,
              f32  x1
            )
```

Clamp x to between x0 and x1.

If x is lower than the minimum value (x0), return x0. If x is higher than the maximum value (x1), return x1. Else return x.

Parameters

[in] **x** The input value.
[in] **x0** The minimum value.
[in] **x1** The maximum value.

Return values

f32 The clamped value of x.

f32 AECos (f32 x)

Calculate the Cosine value of an angle.

Parameters

[in] **x** The angle in Radians.

Return values

f32 The value of cos(x).

f32 AEDegToRad (f32 x)

Convert an angle from Degree to Radians.

Parameters

[in] **x** The angle in Degree to be converted.

Return values

f32 The angle in Radians after conversion.

---

s32 AEInRange (f32 x,
f32 x0,
f32 x1
)

Find if x is in the range (x0 to x1), inclusive.

If x is more than or equal to x0 OR If x is less than or equal to x1, return true. Else return false.

Parameters

[in] **x** The input value.
[in] **x0** The lower bound of range.
[in] **x1** The upper bound of range.

Return values

s32 Returns true if x is between x0 and x1, inclusive. Else return false.

---

u32 AEIsPowOf2 (u32 x)

Check if x is a power of 2.

Powers of 2 are values which can be presented as 2 ^ N, where N is any non-negative integer. Examples of powers of 2 are:

- 1 (2 ^ 0)
- 2 (2 ^ 1)
- 4 (2 ^ 2)
- 256 (2 ^ 8)
- 1024 (2 ^ 10)

Parameters

[in] **x** The value to be checked.

Return values

u32 Returns true (non-zero value) if x is a power of 2. Else return false (zero).

---

u32 AELogBase2 (u32 x)

Calculate the log2 of x.

Calculate the log2 of x, rounded to the lower integer.

Parameters

[in] **x** The input value.

Return values

u32 The log2 of x, rounded to lower integer.

---

f32 AEMax (f32 x,
f32 y
)

Find which of the 2 value is higher.

If x is higher than y, return x. If y is higher than x, return y.

Parameters

[in] **x** The first input value.
[in] **y** The second input value.

Return values

f32 The higher of the 2 value.

---

f32 AEMin (f32 x,
f32 y

)

Find which of the 2 value is lower.

If x is lower than y, return x. If y is lower than x, return y.

Parameters

[in] **x** The first input value.
[in] **y** The second input value.

Return values

f32 The lower of the 2 value.

u32 AENextPowOf2 (u32 x)

Calculate the next power of 2 that is greater than x.

Powers of 2 are values which can be presented as 2 ^ N, where N is any non-negative integer. Examples of powers of 2 are:

- 1 (2 ^ 0)
- 2 (2 ^ 1)
- 4 (2 ^ 2)
- 256 (2 ^ 8)
- 1024 (2 ^ 10)

Parameters

[in] **x** The input value.

Return values

u32 Returns the next power of 2 greater than x.

f32 AERadToDeg (f32 x)

Convert an angle from Radians to Degree.

Parameters

[in] **x** The angle in Radians to be converted.

Return values

f32 The angle in Degree after conversion.

```
f32 AEReflectAnimatedCircleOnStaticCircle ( AEVec2 *  pCtr0s,
                                            AEVec2 *  pCtr0e,
                                            f32       radius0,
                                            AEVec2 *  pCtr1,
                                            f32       radius1,
                                            AEVec2 *  pInter,
                                            AEVec2 *  pReflect
                                          )
```

Calculate the collision between a moving circle with a static circle and the reflected path of moving circle.

Given the start position (pCtr0s) and end position (pCtr0e) of moving circle0 of size (radius0) and the center of static circle1 (pCtr1) of size (radius1), calculate the time, point of intersection and reflected path of circle0.

Warning

Radius of circles should be a non-negative value.

Parameters

[in]	**pCtr0s**	Pointer to **AEVec2** containing start pos of circle0.
[in]	**pCtr0e**	Pointer to **AEVec2** containing end pos of the circle0.
[in]	**radius0**	The radius of circle0.
[in]	**pCtr1**	Pointer to **AEVec2** containing the center of circle1.
[in]	**radius1**	The radius of the circle1.
[out]	**pInter**	Pointer to **AEVec2** for storing the point of

intersection. Will not be used if there is no collision.

[out] **pReflect** Pointer to **AEVec2** for storing the reflected path. Will not be used if there is no collision.

Return values

f32 Returns the time of collision, if there is one. Else return -1.0f.

---

f32
**AEReflectAnimatedCircleOnStaticLineSegment (AEVec2 *
AEVec2 *
f32
AELineSegment2
AEVec2 *
AEVec2 *
)**

Calculate the collision between a moving circle with a line and the reflec the circle.

Given the start position (pStart) and end position (pEnd) of a moving cir (radius) and a line (pLine), calculate the time, point of intersection and r path.

Warning

Line is assumed to stretch to infinity.

Radius of circle should be a non-negative value.

Parameters

[in] **pStart** Pointer to **AEVec2** containing start pos of the circl

[in] **pEnd** Pointer to **AEVec2** containing end pos of the circl

[in] **radius** The radius of the circle.

[in] **pLine** Pointer to **AELineSegment2** containing the line.

[out] **pInter** Pointer to **AEVec2** for storing the point of intersec not be used if there is no collision.

[out] **pReflect** Pointer to **AEVec2** for storing the reflected path. V
used if there is no collision.

Return values

f32 Returns the time of collision, if there is one. Else return -1.0f.

---

f32 AEReflectAnimatedPointOnStaticCircle (**AEVec2 * pStart,**
AEVec2 * pEnd,
AEVec2 * pCtr,
f32 radius,
AEVec2 * pInter,
AEVec2 * pReflect
)

Calculate the collision between a moving point with a circle and the reflected path of the point.

Given the start position (pStart) and end position (pEnd) of a moving point and the center of a circle (pCtr) of size (radius), calculate the time, point of intersection and reflected path.

Warning

Radius of circle should be a non-negative value.

Parameters

[in]	**pStart**	Pointer to **AEVec2** containing start pos of the point.
[in]	**pEnd**	Pointer to **AEVec2** containing end pos of the point.
[in]	**pCtr**	Pointer to **AEVec2** containing the center of the circle.
[in]	**radius**	The radius of the circle.
[out]	**pInter**	Pointer to **AEVec2** for storing the point of intersection. Will not be used if there is no collision.
[out]	**pReflect**	Pointer to **AEVec2** for storing the reflected path. Will not be used if there is no collision.

Return values

f32 Returns the time of collision, if there is one. Else return -1.0f.

f32
AEReflectAnimatedPointOnStaticLineSegment (**AEVec2** ***** **AEVec2** ***** **AELineSegment2** **AEVec2** ***** **AEVec2** ***** **)**

Calculate the collision between a moving point with a line and the reflec of the point.

Given the start position (pStart) and end position (pEnd) of a moving pc line (pLine), calculate the time, point of intersection and reflected path.

Warning

Line is assumed to stretch to infinity.

Parameters

[in]	**pStart**	Pointer to **AEVec2** containing start pos of the poir
[in]	**pEnd**	Pointer to **AEVec2** containing end pos of the poin
[in]	**pLine**	Pointer to **AELineSegment2** containing the line.
[out]	**pInter**	Pointer to **AEVec2** for storing the point of intersec not be used if there is no collision.
[out]	**pReflect**	Pointer to **AEVec2** for storing the reflected path. V used if there is no collision.

Return values

f32 Returns the time of collision, if there is one. Else return -1.0f.

f32 AESin (**f32** **x** **)**

Calculate the Sine value of an angle.

Parameters

[in] **x** The angle in Radians.

Return values

f32 The value of sin(x).

---

f32
AEStaticPointToStaticLineSegment (**AEVec2** * **pPos,**
AELineSegment2 * **pLine**
)

Calculate the shortest distance from a point to a line.

Given a point (pPos) and a line (pLine), calculate the shortest distance from the point to the line.

Warning

Line is assumed to stretch to infinity.

Parameters

[in] **pPos** Pointer to **AEVec2** containing the point.

[in] **pLine** Pointer to **AELineSegment2** containing the line.

Return values

f32 Returns the distance from the point to the line. Negative if the point is in the line's inside half plane. Positive if the point is in the line's outside half plane. Otherwise zero if the point is on the line.

---

f32 AETan (f32 x)

Calculate the Tangent value of an angle.

Parameters

[in] **x** The angle in Radians.

Return values

f32 The value of tan(x).

```
s32 AETestCircleToCircle ( AEVec2 *  pCtr0,
                           f32       radius0,
                           AEVec2 *  pCtr1,
                           f32       radius1
                         )
```

Test for collision between two circles.

Given the center of circle0 (pCtr0) of size (radius0) and the center of circle1 (pCtr1) of size (radius1), check if they are colliding.

Warning

Radius of circle should be a non-negative value.

Parameters

[in] **pCtr0** Pointer to **AEVec2** containing the center of circle0.

[in] **radius0** The radius of circle0.

[in] **pCtr1** Pointer to **AEVec2** containing the center of circle1.

[in] **radius1** The radius of circle1.

Return values

s32 Returns true if the circles are colliding. Else return false.

```
s32 AETestCircleToRect ( AEVec2 *  pCtr,
                         f32       radius,
                         AEVec2 *  pRect,
                         f32       sizeX,
                         f32       sizeY
```

)

Test for collision between a circle and a rectangle.

Given the center of a circle (pCtr) of size (radius) and the center of a rect (pRect) of width (sizeX) and height (sizeY), check if they are colliding.

Warning

radius of circles should be a non-negative value.

The width and height of the rect should be non-negative values.

Function will only work if the rect is not rotated, i.e. the sides of the rect are parallel to the axis.

Parameters

[in]	**pCtr**	Pointer to **AEVec2** containing the center of the circle.
[in]	**radius**	The radius of circle.
[in]	**pRect**	Pointer to **AEVec2** containing the center of the rect.
[in]	**sizeX**	The width of the rect.
[in]	**sizeY**	The height of the rect.

Return values

s32 Returns true if the circle and the rect are colliding. Else return false.

```
s32 AETestPointToCircle ( AEVec2 *  pPos,
                          AEVec2 *  pCtr,
                          f32       radius
                        )
```

Test if a point is inside a circle.

Given a point (pPos) and the center of a circle (pCtr) of size (radius), check if the point is in the circle.

Warning

Radius of circle should be a non-negative value.

Parameters

[in]	**pPos**	Pointer to **AEVec2** containing the position of the point.
[in]	**pCtr**	Pointer to **AEVec2** containing the center of the circle.
[in]	**radius**	The radius of circle0.

Return values

s32	Returns true if the point is inside the circle. Else return false.

```
s32 AETestPointToRect ( AEVec2 *  pPos,
                        AEVec2 *  pRect,
                        f32       sizeX,
                        f32       sizeY
                      )
```

Test if a point is inside a rectangle.

Given a point (pPos) and the center of a rect (pRect) of width (sizeX) and height (sizeY), check if the point is in the rect.

Warning

The width and height of the rect should be non-negative values.

Function will only work if the rect is not rotated, i.e. the sides of the rect are parallel to the axis.

Parameters

[in]	**pPos**	Pointer to **AEVec2** containing the position of the point.
[in]	**pRect**	Pointer to **AEVec2** containing the center of the rect.
[in]	**sizeX**	The width of the rect.
[in]	**sizeY**	The height of the rect.

Return values

s32 Returns true if the point is inside the rect. Else return false.

```
s32 AETestRectToRect ( AEVec2 *  pRect0,
                       f32       sizeX0,
                       f32       sizeY0,
                       AEVec2 *  pRect1,
                       f32       sizeX1,
                       f32       sizeY1
                     )
```

Test for collision between two rectangles.

Given the center of rect0 (pRect0) of width (sizeX0) and height (sizeY0) and the center of rect1 (pRect1) of width (sizeX1) and height (sizeY1), check if they are colliding.

Warning

The width and height of the rect should be non-negative values.

Function will only work if the rect is not rotated, i.e. the sides of the rect are parallel to the axis.

Parameters

- `[in]` **pRect0** Pointer to **AEVec2** containing the center of rect0.
- `[in]` **sizeX0** The width of rect0.
- `[in]` **sizeY0** The height of rect0.
- `[in]` **pRect1** Pointer to **AEVec2** containing the center of rect1.
- `[in]` **sizeX1** The width of rect1.
- `[in]` **sizeY1** The height of rect1.

Return values

s32 Returns true if the rects are colliding. Else return false.

```
f32 AEWrap ( f32  x,
             f32  x0,
```

```
        f32 x1
    )
```

Wraparound for x with respect to range (x0 to x1).

If x is lesser than x0, return (x + range). If x is higher than x1, return (x - range).

Warning

Wraparound does not work if x is lesser than (x0 - range) or if x is greater than (x1 + range).

Parameters

[in] **x** The input value.
[in] **x0** The lower bound of range
[in] **x1** The upper bound of range.

Return values

f32 The wraparound value of x with respect to range.

Alpha Engine

Classes | Macros | Typedefs | Functions

AEMtx33.h File Reference

Header file for the 3x3 matrix library. More...

Go to the source code of this file.

Classes

struct	**AEMtx33**

Macros

#define **AEMtx33RowCol**(pMtx, row, col) (pMtx)->m[(row)][(col)]

Typedefs

typedef struct **AEMtx33** **AEMtx33**

Functions

void **AEMtx33Identity** (**AEMtx33** *pResult)
Set pResult to identity matrix. More...

void **AEMtx33Transpose** (**AEMtx33** *pResult, **AEMtx33** *pMtx)
Set pResult to the transpose of pMtx. More...

f32 **AEMtx33Inverse** (**AEMtx33** *pResult, **AEMtx33** *pMtx)
Set pResult to the inverse of pMtx. More...

void **AEMtx33InvTranspose** (**AEMtx33** *pResult, **AEMtx33** *pMtx)
Set pResult to the transpose of the inverse of pMtx. More...

void **AEMtx33Concat** (**AEMtx33** *pResult, **AEMtx33** *pMtx0, **AEMtx33** *pMtx1)
Set pResult to the multiplication of pMtx0 with pMtx1. More...

void **AEMtx33Orthogonalize** (**AEMtx33** *pResult, **AEMtx33** *pMtx)
Set pResult to the orthogonalization of pMtx. More...

f32 **AEMtx33Determinant** (**AEMtx33** *pMtx)
Calculate the determinant of pMtx. More...

void **AEMtx33SetCol** (**AEMtx33** *pResult, **u32** col, **AEVec2** *pVec)
Set the first and second element of the selected column of pResult to the x and y values of pVec respectively. The last element of the column will be set automatically. More...

void **AEMtx33SetRow** (**AEMtx33** *pResult, **u32** row, **AEVec2** *pVec)
Set the first and second element of the selected row of pResult to the x and y values of pVec respectively. The last element of the row will be set automatically. More...

void **AEMtx33GetCol** (**AEVec2** *pResult, **AEMtx33** *pMtx, **u32** col)
Set the x and y values of pResult with the first and second

element of the selected column of pMtx respectively. More...

void **AEMtx33GetRow** (**AEVec2** *pResult, **AEMtx33** *pMtx, **u32** row)

void **AEMtx33Trans** (**AEMtx33** *pResult, **f32** x, **f32** y)
Set pResult to the translation matrix of x and y. More...

void **AEMtx33TransApply** (**AEMtx33** *pResult, **AEMtx33** *pMtx, **f32** x, **f32** y)
Set pResult to the multiplication of translation matrix of x and y with pMtx. More...

void **AEMtx33Scale** (**AEMtx33** *pResult, **f32** x, **f32** y)
Set pResult to the scaling matrix of x and y. More...

void **AEMtx33ScaleApply** (**AEMtx33** *pResult, **AEMtx33** *pMtx, **f32** x, **f32** y)
Set pResult to the multiplication of scaling matrix of x and y with pMtx. More...

void **AEMtx33Rot** (**AEMtx33** *pResult, **f32** angle)
Set pResult to the rotation matrix of angle in radians rotating counter-clockwise. More...

void **AEMtx33RotDeg** (**AEMtx33** *pResult, **f32** angle)
Set pResult to the rotation matrix of angle in degrees rotating counter-clockwise. More...

void **AEMtx33MultVec** (**AEVec2** *pResult, **AEMtx33** *pMtx, **AEVec2** *pVec)
Set pResult to the multiplication of pMtx with pVec. More...

void **AEMtx33MultVecArray** (**AEVec2** *pResult, **AEMtx33** *pMtx, **AEVec2** *pVec, **u32** count)
Set an array of vectors (pResult) to the multiplication of pMtx with an array of vectors (pVec) of size (count). More...

void	**AEMtx33MultVecSR** (**AEVec2** *pResult, **AEMtx33** *pMtx, **AEVec2** *pVec)
void	**AEMtx33MultVecArraySR** (**AEVec2** *pResult, **AEMtx33** *pMtx, **AEVec2** *pVec, **u32** count)

Detailed Description

Header file for the 3x3 matrix library.

Project: Alpha Engine

Author
Sun Tjen Fam

Date
January 31, 2008

Copyright

Definition in file **AEMtx33.h**.

Macro Definition Documentation

```
#define AEMtx33RowCol (  pMtx,
                         row,
                         col
                      )    (pMtx)->m[(row)][(col)]
```

Definition at line **23** of file **AEMtx33.h**.

Typedef Documentation

typedef struct AEMtx33 AEMtx33

Matrix is stored in column major format, ie. the translation term is in the right most column.

m[0][0] m[0][1] m[0][2]
m[1][0] m[1][1] m[1][2]
m[2][0] m[2][1] m[2][2]

Function Documentation

void AEMtx33Concat (AEMtx33 * pResult,
AEMtx33 * pMtx0,
AEMtx33 * pMtx1
)

Set pResult to the multiplication of pMtx0 with pMtx1.

Parameters

`[out]` **pResult** Pointer to **AEMtx33** to be set.
`[in]` **pMtx0** Pointer to **AEMtx33** for input.
`[in]` **pMtx1** Pointer to **AEMtx33** for input.

Return values

void No return.

f32 AEMtx33Determinant (AEMtx33 * pMtx)

Calculate the determinant of pMtx.

Parameters

`[in]` **pMtx** Pointer to **AEMtx33** for input.

Return values

f32 Returns the determinant of pMtx.

void AEMtx33GetCol (AEVec2 * pResult,
AEMtx33 * pMtx,
u32 col
)

Set the x and y values of pResult with the first and second element of the selected column of pMtx respectively.

Parameters

[out] **pResult** Pointer to **AEVec2** to be set.
[in] **pMtx** Pointer to **AEMtx33** for input.
[in] **col** The selected column of pResult.

Return values

void No return.

---

void AEMtx33GetRow (AEVec2 * pResult,
AEMtx33 * pMtx,
u32 row
)

---

void AEMtx33Identity (AEMtx33 * pResult)

Set pResult to identity matrix.

Parameters

[out] **pResult** Pointer to **AEMtx33** to be set.

Return values

void No return.

---

f32 AEMtx33Inverse (AEMtx33 * pResult,
AEMtx33 * pMtx
)

Set pResult to the inverse of pMtx.

Parameters

[out] **pResult** Pointer to **AEMtx33** to be set.

[in] **pMtx** Pointer to **AEMtx33** for input.

Return values

f32 Returns the determinant of pMtx.

---

void AEMtx33InvTranspose (AEMtx33 * pResult,
AEMtx33 * pMtx
)

Set pResult to the transpose of the inverse of pMtx.

Parameters

[out] **pResult** Pointer to **AEMtx33** to be set.
[in] **pMtx** Pointer to **AEMtx33** for input.

Return values

void No return.

---

void AEMtx33MultVec (AEVec2 * pResult,
AEMtx33 * pMtx,
AEVec2 * pVec
)

Set pResult to the multiplication of pMtx with pVec.

Parameters

[out] **pResult** Pointer to **AEMtx33** to be set.
[in] **pMtx** Pointer to **AEMtx33** for input.
[in] **pVec** Pointer to **AEVec2** for input.

Return values

void No return.

---

void AEMtx33MultVecArray (AEVec2 * pResult,

```
                                    AEMtx33 *  pMtx,
                                    AEVec2 *   pVec,
                                    u32        count
                                )
```

Set an array of vectors (pResult) to the multiplication of pMtx with an array of vectors (pVec) of size (count).

Warning
Size of pResult should be not less than count.

Parameters

[out]	**pResult**	Pointer to an array of **AEMtx33** to be set.
[in]	**pMtx**	Pointer to **AEMtx33** for input.
[in]	**pVec**	Pointer to an array of **AEVec2** for input.
[in]	**count**	Number of elements in pVec to be multiplied.

Return values
void No return.

```
void AEMtx33MultVecArraySR ( AEVec2 *   pResult,
                             AEMtx33 *  pMtx,
                             AEVec2 *   pVec,
                             u32        count
                           )
```

```
void AEMtx33MultVecSR ( AEVec2 *   pResult,
                        AEMtx33 *  pMtx,
                        AEVec2 *   pVec
                      )
```

```
void AEMtx33Orthogonalize ( AEMtx33 *  pResult,
                            AEMtx33 *  pMtx
                          )
```

Set pResult to the orthogonalization of pMtx.

Warning
Function not implemented.

Parameters

[out]	**pResult**	Pointer to **AEMtx33** to be set.
[in]	**pMtx**	Pointer to **AEMtx33** for input.

Return values

void No return.

---

void AEMtx33Rot (AEMtx33 * pResult,
f32 angle
)

Set pResult to the rotation matrix of angle in radians rotating counter-clockwise.

Parameters

[out]	**pResult**	Pointer to **AEMtx33** to be set.
[in]	**angle**	Angle in radians rotating counter-clockwise.

Return values

void No return.

---

void AEMtx33RotDeg (AEMtx33 * pResult,
f32 angle
)

Set pResult to the rotation matrix of angle in degrees rotating counter-clockwise.

Parameters

[out]	**pResult**	Pointer to **AEMtx33** to be set.

[in] **angle** Angle in degress rotating counter-clockwise.

Return values

void No return.

---

void AEMtx33Scale (AEMtx33 * pResult,
f32 x,
f32 y
)

Set pResult to the scaling matrix of x and y.

Parameters

[out] **pResult** Pointer to **AEMtx33** to be set.
[in] **x** Scaling along the x-axis.
[in] **y** Scaling along the y-axis.

Return values

void No return.

---

void AEMtx33ScaleApply (AEMtx33 * pResult,
AEMtx33 * pMtx,
f32 x,
f32 y
)

Set pResult to the multiplication of scaling matrix of x and y with pMtx.

Parameters

[out] **pResult** Pointer to **AEMtx33** to be set.
[in] **pMtx** Pointer to **AEMtx33** for input.
[in] **x** Scaling along the x-axis.
[in] **y** Scaling along the y-axis.

Return values

void No return.

---

void AEMtx33SetCol (AEMtx33 * pResult,
u32 col,
AEVec2 * pVec
)

Set the first and second element of the selected column of pResult to the x and y values of pVec respectively. The last element of the column will be set automatically.

Parameters

[in,out]	**pResult**	Pointer to **AEMtx33** to be set.
[in]	**col**	The selected column of pResult.
[in]	**pVec**	Pointer to **AEVec2** for input.

Return values

void No return.

---

void AEMtx33SetRow (AEMtx33 * pResult,
u32 row,
AEVec2 * pVec
)

Set the first and second element of the selected row of pResult to the x and y values of pVec respectively. The last element of the row will be set automatically.

Parameters

[in,out]	**pResult**	Pointer to **AEMtx33** to be set.
[in]	**row**	The selected row of pResult.
[in]	**pVec**	Pointer to **AEVec2** for input.

Return values

void No return.

---

void AEMtx33Trans (AEMtx33 * pResult,
f32 x,
f32 y
)

Set pResult to the translation matrix of x and y.

Parameters

[out]	**pResult**	Pointer to **AEMtx33** to be set.
[in]	**x**	Translation along the x-axis.
[in]	**y**	Translation along the y-axis.

Return values

void No return.

---

void AEMtx33TransApply (AEMtx33 * pResult,
AEMtx33 * pMtx,
f32 x,
f32 y
)

Set pResult to the multiplication of translation matrix of x and y with pMtx.

Parameters

[out]	**pResult**	Pointer to **AEMtx33** to be set.
[in]	**pMtx**	Pointer to **AEMtx33** for input.
[in]	**x**	Translation along the x-axis.
[in]	**y**	Translation along the y-axis.

Return values

void No return.

void AEMtx33Transpose (AEMtx33 * pResult,
AEMtx33 * pMtx
)

Set pResult to the transpose of pMtx.

Parameters

[out] **pResult** Pointer to **AEMtx33** to be set.
[in] **pMtx** Pointer to **AEMtx33** for input.

Return values

void No return.

Alpha Engine

Functions | Variables

AESystem.h File Reference

Header file for the system library. More...

Go to the source code of this file.

Functions

s32	**AESysInit** (HINSTANCE hAppInstance, **s32** show, **s32** WinWidth, **s32** WinHeight, **s32** CreateConsole, **u32** FrameRateMax, LRESULT(*pWinCallBack)(HWND hWnd, UINT msg, WPARAM wp, LPARAM lp)) Initialize the Alpha Engine. More...
void	**AESysReset** () Reset the Alpha Engine. More...
void	**AESysExit** () Free the Alpha Engine. More...
void	**AESysFrameStart** () Start of frame. More...
void	**AESysFrameEnd** () End of frame. More...
HWND	**AESysGetWindowHandle** () Get the handle to the window. More...
void	**AESysSetWindowTitle** (const **s8** *pTitle) Set the title of the window. More...
s32	**AESysDoesWindowExist** () Check if the window has been closed. More...

Variables

HINSTANCE	**ghAESysAppInstance**
HWND	**gAESysWindowHandle**
WNDCLASS	**winClass**
const **s8** *	**gpAESysWinTitle**
const **s8** *	**gpAESysWinClassName**
s32	**gAESysAppActive**

Detailed Description

Header file for the system library.

Project: Alpha Engine

Author
Sun Tjen Fam

Date
January 31, 2007

Definition in file **AESystem.h**.

Function Documentation

s32 AESysDoesWindowExist ( )

Check if the window has been closed.

Return values

s32 Return 1 if the window exist. Else return 0.

void AESysExit ( )

Free the Alpha Engine.

Call this function once at the end of the program to free the system.

Return values

void No return.

void AESysFrameEnd ( )

End of frame.

Call this function once at the end of the frame to inform the system the current frame is ending.

Return values

void No return.

void AESysFrameStart ( )

Start of frame.

Call this function once at the start of the frame to inform the system a new frame is starting.

Return values

void No return.

HWND AESysGetWindowHandle ( )

Get the handle to the window.

Return values

HWND Returns the handle to the window.

```
s32
AESysInit ( HINSTANCE
            s32
            s32
            s32
            s32
            u32
            LRESULT(*)(HWND hWnd, UINT msg, WPARAM wp, LP
          )
```

Initialize the Alpha Engine.

Call this function once at the start of the program to initialize the system

Parameters

[in]	**hAppInstance**	Handle to the current instance of the applica
[in]	**show**	Set how the the window is to be shown.
[in]	**WinWidth**	Set the width of the window.
[in]	**WinHeight**	Set the height of the window.
[in]	**CreateConsole**	Input 1 if a console window should be create
[in]	**FrameRateMax**	Set the maximum frames per second.
[in]	**pWinCallBack**	Pointer to a callback function. May be left nu

Return values

s32 Return 1 if initialization is successful. Else return 0.

void AESysReset ( )

Reset the Alpha Engine.

Call this function when changing gamestates to reset the system.

Return values

void No return.

void AESysSetWindowTitle (const s8 * pTitle)

Set the title of the window.

Parameters

[in] **pTitle** Pointer to a null-terminated string to set as the title of the window.

Return values

void No return.

Variable Documentation

s32 gAESysAppActive

HWND gAESysWindowHandle

HINSTANCE ghAESysAppInstance

const s8* gpAESysWinClassName

const s8* gpAESysWinTitle

WNDCLASS winClass

Alpha Engine

Typedefs

AETypes.h File Reference

Header file for the typedefs. More...

Go to the source code of this file.

Typedefs

typedef char	**s8**
typedef unsigned char	**u8**
typedef signed short	**s16**
typedef unsigned short	**u16**
typedef signed int	**s32**
typedef unsigned int	**u32**
typedef signed long long	**s64**
typedef unsigned long long	**u64**
typedef float	**f32**
typedef double	**f64**

Detailed Description

Header file for the typedefs.

Project: Alpha Engine

Author
Antoine Abi Chacra

Date
March 21, 2013

Copyright

Definition in file **AETypes.h**.

Typedef Documentation

typedef float f32

Definition at line 31 of file AETypes.h.

typedef double f64

Definition at line 32 of file AETypes.h.

typedef signed short s16

Definition at line 25 of file AETypes.h.

typedef signed int s32

Definition at line 27 of file AETypes.h.

typedef signed long long s64

Definition at line 29 of file AETypes.h.

typedef char s8

Definition at line 23 of file AETypes.h.

typedef unsigned short u16

Definition at line **26** of file **AETypes.h**.

typedef unsigned int u32

Definition at line **28** of file **AETypes.h**.

typedef unsigned long long u64

Definition at line **30** of file **AETypes.h**.

typedef unsigned char u8

Definition at line **24** of file **AETypes.h**.

Alpha Engine

Macros | Functions

AEUtil.h File Reference

Header file for the utility library. More...

Go to the source code of this file.

Macros

#define **isZero**(x) ((x < **EPSILON**) && (x > -**EPSILON**))

#define **isEqual**(x, y) (((x >= y) ? (x-y) : (y-x)) < **EPSILON**)

Functions

f64 **AEGetTime** (**f64** *pTime)
Get the current time in seconds. More...

f32 **AERandFloat** ()
Get a random real number between 0.0f to 1.0f. More...

s8 * **ReadFromFile** (const **s8** *fileName)
Read file into memory. More...

Detailed Description

Header file for the utility library.

Project: Alpha Engine

Author
Sun Tjen Fam

Date
February 02, 2008

Copyright

Definition in file **AEUtil.h**.

Macro Definition Documentation

```
#define isEqual (   x,
                    y
                )     (((x >= y) ? (x-y) : (y-x)) < EPSILON)
```

Definition at line **56** of file **AEUtil.h**.

```
#define isZero (   x )    ((x < EPSILON) && (x > -EPSILON))
```

Definition at line **54** of file **AEUtil.h**.

Function Documentation

f64 AEGetTime (f64 * pTime)

Get the current time in seconds.

Parameters

[out] **pTime** Pointer to f64 to store the current time. May be left null if not needed.

Return values

f64 Returns the current time in seconds.

f32 AERandFloat ( )

Get a random real number between 0.0f to 1.0f.

Return values

f32 Returns a random real number between 0.0f to 1.0f.

s8 * ReadFromFile (const s8 * fileName)

Read file into memory.

Open a file named filename, allocate sufficient storage and read the file into memory.

Warning

User must delete the memory allocated on their own.

Parameters

[in] **fileName** Pointer to the name of the file.

Return values

s8* Pointer to the start of the memory containing the file.

Alpha Engine

Classes | Typedefs | Functions

AEVec2.h File Reference

Header file for the 2D vector library. More...

```
#include "AEExport.h" #include "AETypes.h"
```

Go to the source code of this file.

Classes

struct	**AEVec2**

Typedefs

typedef struct **AEVec2** **AEVec2**

Functions

s32 **AEVec2Test** (**s32** i1, **s32** i2)

void **AEVec2Zero** (**AEVec2** *pResult)
Set pResult to zero. More...

void **AEVec2Set** (**AEVec2** *pResult, **f32** x, **f32** y)
Set pResult to (x, y). More...

void **AEVec2Neg** (**AEVec2** *pResult, **AEVec2** *pVec0)
Set pResult to the negative of pVec0. More...

void **AEVec2Add** (**AEVec2** *pResult, **AEVec2** *pVec0, **AEVec2** *pVec1)
Set pResult to (pVec0 + pVec1). More...

void **AEVec2Sub** (**AEVec2** *pResult, **AEVec2** *pVec0, **AEVec2** *pVec1)
Set pResult to (pVec0 - pVec1). More...

void **AEVec2Normalize** (**AEVec2** *pResult, **AEVec2** *pVec0)
Set pResult to the normalized of pVec0. More...

void **AEVec2Scale** (**AEVec2** *pResult, **AEVec2** *pVec0, **f32** s)
Set pResult to (pVec0 * s). More...

void **AEVec2ScaleAdd** (**AEVec2** *pResult, **AEVec2** *pVec0, **AEVec2** *pVec1, **f32** s)
Set pResult to ((pVec0 + pVec1) * s). More...

void **AEVec2ScaleSub** (**AEVec2** *pResult, **AEVec2** *pVec0, **AEVec2** *pVec1, **f32** s)
Set pResult to ((pVec0 - pVec1) * s). More...

AEVec2Project (**AEVec2** *pResult, **AEVec2** *pVec0, **AEVec2**

void *pVec1)
Set pResult to the projection of pVec1 onto pVec0. More...

void **AEVec2ProjectPerp** (**AEVec2** *pResult, **AEVec2** *pVec0, **AEVec2** *pVec1)
Set pResult to the perpendicular projection (rejection) of pVec0 onto pVec1. More...

void **AEVec2Lerp** (**AEVec2** *pResult, **AEVec2** *pVec0, **AEVec2** *pVec1, **f32** t)
Set pResult to the linear interpolation between pVec0 and pVec1 at time interval t. More...

f32 **AEVec2Length** (**AEVec2** *pVec0)
Calculate the length of pVec0. More...

f32 **AEVec2SquareLength** (**AEVec2** *pVec0)
Calculate the squared length of pVec0. More...

f32 **AEVec2Distance** (**AEVec2** *pVec0, **AEVec2** *pVec1)
Calculate the distance between 2 points. More...

f32 **AEVec2SquareDistance** (**AEVec2** *pVec0, **AEVec2** *pVec1)
Calculate the squared distance between 2 points. More...

f32 **AEVec2DotProduct** (**AEVec2** *pVec0, **AEVec2** *pVec1)
Calculate the dot product of 2 vectors. More...

f32 **AEVec2CrossProductMag** (**AEVec2** *pVec0, **AEVec2** *pVec1)
Calculate the magnitude of the cross product of 2 vectors. More...

void **AEVec2FromAngle** (**AEVec2** *pResult, **f32** angle)
Set pResult to unit vector of angle. More...

Detailed Description

Header file for the 2D vector library.

Project: Alpha Engine

Author
Sun Tjen Fam

Date
January 31, 2008

Copyright

Definition in file **AEVec2.h**.

Typedef Documentation

typedef struct AEVec2 AEVec2

Function Documentation

void AEVec2Add (AEVec2 * pResult,
AEVec2 * pVec0,
AEVec2 * pVec1
)

Set pResult to (pVec0 + pVec1).

Parameters

`[out]` **pResult** Pointer to **AEVec2** to be set.
`[in]` **pVec0** Pointer to **AEVec2** for input.
`[in]` **pVec1** Pointer to **AEVec2** for input.

Return values

void No return.

f32 AEVec2CrossProductMag (AEVec2 * pVec0,
AEVec2 * pVec1
)

Calculate the magnitude of the cross product of 2 vectors.

Parameters

`[in]` **pVec0** Pointer to **AEVec2** of first vector.
`[in]` **pVec1** Pointer to **AEVec2** of second vector.

Return values

f32 Returns the magnitude of the cross product of pVec0 and pVec1.

```
f32 AEVec2Distance ( AEVec2 *  pVec0,
                     AEVec2 *  pVec1
                   )
```

Calculate the distance between 2 points.

Parameters

[in] **pVec0** Pointer to **AEVec2** of first point.
[in] **pVec1** Pointer to **AEVec2** of second point.

Return values

f32 Returns the distance between pVec0 and pVec1.

```
f32 AEVec2DotProduct ( AEVec2 *  pVec0,
                       AEVec2 *  pVec1
                     )
```

Calculate the dot product of 2 vectors.

Parameters

[in] **pVec0** Pointer to **AEVec2** of first vector.
[in] **pVec1** Pointer to **AEVec2** of second vector.

Return values

f32 Returns the dot product of pVec0 and pVec1.

```
void AEVec2FromAngle ( AEVec2 *  pResult,
                       f32       angle
                     )
```

Set pResult to unit vector of angle.

Parameters

[out] **pResult** Pointer to **AEVec2** to be set.

[in] **angle** Angle of the unit vector.

Return values

void No return.

---

f32 AEVec2Length (AEVec2 * pVec0)

Calculate the length of pVec0.

Parameters

[in] **pVec0** Pointer to **AEVec2** for input

Return values

f32 Returns the length of pVec0.

---

void AEVec2Lerp (AEVec2 * pResult, AEVec2 * pVec0, AEVec2 * pVec1, f32 t)

Set pResult to the linear interpolation between pVec0 and pVec1 at time interval t.

Warning

t should be between 0.0f to 1.0f.

Parameters

[out] **pResult** Pointer to **AEVec2** to be set.

[in] **pVec0** Pointer to **AEVec2** for input.

[in] **pVec1** Pointer to **AEVec2** for input.

[in] **t** The time interval to calculate the linear interpolation at. At t = 0.0f, the result will be pVec0. At t = 1.0f, the result will be pVec1.

Return values

void No return.

---

void AEVec2Neg (AEVec2 * pResult,
AEVec2 * pVec0
)

Set pResult to the negative of pVec0.

Parameters

`[out]` **pResult** Pointer to **AEVec2** to be set.
`[in]` **pVec0** Pointer to **AEVec2** for input.

Return values

void No return.

---

void AEVec2Normalize (AEVec2 * pResult,
AEVec2 * pVec0
)

Set pResult to the normalized of pVec0.

Parameters

`[out]` **pResult** Pointer to **AEVec2** to be set.
`[in]` **pVec0** Pointer to **AEVec2** for input.

Return values

void No return.

---

void AEVec2Project (AEVec2 * pResult,
AEVec2 * pVec0,
AEVec2 * pVec1
)

Set pResult to the projection of pVec1 onto pVec0.

Parameters

[out]	**pResult**	Pointer to **AEVec2** to be set.
[in]	**pVec0**	Pointer to **AEVec2** for input.
[in]	**pVec1**	Pointer to **AEVec2** for input.

Return values

void No return.

```
void AEVec2ProjectPerp ( AEVec2 *  pResult,
                         AEVec2 *  pVec0,
                         AEVec2 *  pVec1
                       )
```

Set pResult to the perpendicular projection (rejection) of pVec0 onto pVec1.

Parameters

[out]	**pResult**	Pointer to **AEVec2** to be set.
[in]	**pVec0**	Pointer to **AEVec2** for input.
[in]	**pVec1**	Pointer to **AEVec2** for input.

Return values

void No return.

```
void AEVec2Scale ( AEVec2 *  pResult,
                   AEVec2 *  pVec0,
                   f32       s
                 )
```

Set pResult to (pVec0 * s).

Parameters

[out]	**pResult**	Pointer to **AEVec2** to be set.

[in]	**pVec0**	Pointer to **AEVec2** for input.
[in]	**s**	Value to scale with.

Return values

void No return.

---

void AEVec2ScaleAdd (**AEVec2 *** **pResult,**
AEVec2 * **pVec0,**
AEVec2 * **pVec1,**
f32 **s**
)

Set pResult to ((pVec0 + pVec1) * s).

Warning

Purpose unclear or error in implementation.

Parameters

[out]	**pResult**	Pointer to **AEVec2** to be set.
[in]	**pVec0**	Pointer to **AEVec2** for input.
[in]	**pVec1**	Pointer to **AEVec2** for input.
[in]	**s**	Value to scale with.

Return values

void No return.

---

void AEVec2ScaleSub (**AEVec2 *** **pResult,**
AEVec2 * **pVec0,**
AEVec2 * **pVec1,**
f32 **s**
)

Set pResult to ((pVec0 - pVec1) * s).

Warning

Purpose unclear or error in implementation.

Parameters

[out]	**pResult**	Pointer to **AEVec2** to be set.
[in]	**pVec0**	Pointer to **AEVec2** for input.
[in]	**pVec1**	Pointer to **AEVec2** for input.
[in]	**s**	Value to scale with.

Return values

void No return.

```
void AEVec2Set ( AEVec2 *  pResult,
                 f32       x,
                 f32       y
               )
```

Set pResult to (x, y).

Parameters

[out]	**pResult**	Pointer to **AEVec2** to be set.
[in]	**x**	X value to set with.
[in]	**y**	Y value to set with.

Return values

void No return.

```
f32 AEVec2SquareDistance ( AEVec2 *  pVec0,
                           AEVec2 *  pVec1
                         )
```

Calculate the squared distance between 2 points.

Parameters

[in]	**pVec0**	Pointer to **AEVec2** of first point.
[in]	**pVec1**	Pointer to **AEVec2** of second point.

Return values

f32 Returns the squared distance between pVec0 and pVec1.

---

f32 AEVec2SquareLength (AEVec2 * pVec0)

Calculate the squared length of pVec0.

Parameters

`[in]` **pVec0** Pointer to **AEVec2** for input

Return values

f32 Returns the squared length of pVec0.

```
void AEVec2Sub ( AEVec2 *  pResult,
                 AEVec2 *  pVec0,
                 AEVec2 *  pVec1
               )
```

Set pResult to (pVec0 - pVec1).

Parameters

`[out]` **pResult** Pointer to **AEVec2** to be set.
`[in]` **pVec0** Pointer to **AEVec2** for input.
`[in]` **pVec1** Pointer to **AEVec2** for input.

Return values

void No return.

```
s32 AEVec2Test ( s32  i1,
                 s32  i2
               )
```

---

void AEVec2Zero (AEVec2 * pResult)

Set pResult to zero.

Parameters

[out] **pResult** Pointer to **AEVec2** to be set.

Return values

void No return.

Alpha Engine

Here is a list of all file members with links to the files they belong to:

- a -

- AE_API : **AEExport.h**
- AE_ASSERT : **AEEngine.h**
- AE_ASSERT_ALLOC : **AEEngine.h**
- AE_ASSERT_MESG : **AEEngine.h**
- AE_ASSERT_PARM : **AEEngine.h**
- AE_FATAL_ERROR : **AEEngine.h**
- AE_GFX_BM_ADD : **AEGraphics.h**
- AE_GFX_BM_BLEND : **AEGraphics.h**
- AE_GFX_BM_NONE : **AEGraphics.h**
- AE_GFX_BM_NUM : **AEGraphics.h**
- AE_GFX_MDM_LINES : **AEGraphics.h**
- AE_GFX_MDM_LINES_STRIP : **AEGraphics.h**
- AE_GFX_MDM_NUM : **AEGraphics.h**
- AE_GFX_MDM_POINTS : **AEGraphics.h**
- AE_GFX_MDM_TRIANGLES : **AEGraphics.h**
- AE_GFX_RM_COLOR : **AEGraphics.h**
- AE_GFX_RM_NONE : **AEGraphics.h**
- AE_GFX_RM_NUM : **AEGraphics.h**
- AE_GFX_RM_TEXTURE : **AEGraphics.h**
- AE_GFX_TM_AVERAGE : **AEGraphics.h**
- AE_GFX_TM_PRECISE : **AEGraphics.h**
- AE_GS_QUIT : **AEGameStateMgr.h**
- AE_GS_RESTART : **AEGameStateMgr.h**
- AE_WARNING : **AEEngine.h**
- AE_WARNING_MESG : **AEEngine.h**

- AE_WARNING_PARM : **AEEngine.h**
- AEACos() : **AEMath.h**
- AEACosDeg : **AEMath.h**
- AEAnimatedCircleToStaticCircle() : **AEMath.h**
- AEAnimatedCircleToStaticLineSegment() : **AEMath.h**
- AEAnimatedPointToStaticCircle() : **AEMath.h**
- AEAnimatedPointToStaticLineSegment() : **AEMath.h**
- AEASin() : **AEMath.h**
- AEASinDeg : **AEMath.h**
- AEATan() : **AEMath.h**
- AEATanDeg : **AEMath.h**
- AEBuildLineSegment2() : **AELineSegment2.h**
- AECalcDistCircleToCircle() : **AEMath.h**
- AECalcDistCircleToRect() : **AEMath.h**
- AECalcDistPointToCircle() : **AEMath.h**
- AECalcDistPointToConvexPoly() : **AEMath.h**
- AECalcDistPointToLineSeg() : **AEMath.h**
- AECalcDistPointToRect() : **AEMath.h**
- AECalcDistRectToRect() : **AEMath.h**
- AEClamp() : **AEMath.h**
- AECos() : **AEMath.h**
- AECosDeg : **AEMath.h**
- AEDegToRad() : **AEMath.h**
- AEFrameRateControllerEnd() : **AEFrameRateController.h**
- AEFrameRateControllerGetFrameCount() : **AEFrameRateController.h**
- AEFrameRateControllerGetFrameTime() : **AEFrameRateController.h**
- AEFrameRateControllerInit() : **AEFrameRateController.h**
- AEFrameRateControllerReset() : **AEFrameRateController.h**
- AEFrameRateControllerStart() : **AEFrameRateController.h**
- AEGameStateDraw : **AEGameStateMgr.h**
- AEGameStateFree : **AEGameStateMgr.h**
- AEGameStateInit : **AEGameStateMgr.h**
- AEGameStateLoad : **AEGameStateMgr.h**
- AEGameStateMgrAdd() : **AEGameStateMgr.h**
- AEGameStateMgrInit() : **AEGameStateMgr.h**
- AEGameStateMgrUpdate() : **AEGameStateMgr.h**
- AEGameStateUnload : **AEGameStateMgr.h**
- AEGameStateUpdate : **AEGameStateMgr.h**

- AEGetTime() : **AEUtil.h**
- AEGfxAxis() : **AEGraphics.h**
- AEGfxBlendMode : **AEGraphics.h**
- AEGfxBox() : **AEGraphics.h**
- AEGfxColInterp() : **AEGraphics.h**
- AEGfxCone() : **AEGraphics.h**
- AEGfxEnd() : **AEGraphics.h**
- AEGfxExit() : **AEGraphics.h**
- AEGfxGetCamPosition() : **AEGraphics.h**
- AEGfxGetWinMaxX() : **AEGraphics.h**
- AEGfxGetWinMaxY() : **AEGraphics.h**
- AEGfxGetWinMinX() : **AEGraphics.h**
- AEGfxGetWinMinY() : **AEGraphics.h**
- AEGfxInit() : **AEGraphics.h**
- AEGfxLine() : **AEGraphics.h**
- AEGfxMeshDraw() : **AEGraphics.h**
- AEGfxMeshDrawMode : **AEGraphics.h**
- AEGfxMeshEnd() : **AEGraphics.h**
- AEGfxMeshFree() : **AEGraphics.h**
- AEGfxMeshStart() : **AEGraphics.h**
- AEGfxPoint() : **AEGraphics.h**
- AEGfxPrint() : **AEGraphics.h**
- AEGfxQuad() : **AEGraphics.h**
- AEGfxRenderMode : **AEGraphics.h**
- AEGfxReset() : **AEGraphics.h**
- AEGfxSaveTextureToFile() : **AEGraphics.h**
- AEGfxSetBackgroundColor() : **AEGraphics.h**
- AEGfxSetBlendColor() : **AEGraphics.h**
- AEGfxSetBlendMode() : **AEGraphics.h**
- AEGfxSetCamPosition() : **AEGraphics.h**
- AEGfxSetPosition() : **AEGraphics.h**
- AEGfxSetRenderMode() : **AEGraphics.h**
- AEGfxSetTextureMode() : **AEGraphics.h**
- AEGfxSetTintColor() : **AEGraphics.h**
- AEGfxSetTransform() : **AEGraphics.h**
- AEGfxSetTransform3D() : **AEGraphics.h**
- AEGfxSetTransparency() : **AEGraphics.h**
- AEGfxSphere() : **AEGraphics.h**
- AEGfxStart() : **AEGraphics.h**
- AEGfxSurface : **AEGraphics.h**

- AEGfxTexture : **AEGraphics.h**
- AEGfxTextureLoad() : **AEGraphics.h**
- AEGfxTextureLoadFromMemory() : **AEGraphics.h**
- AEGfxTextureMode : **AEGraphics.h**
- AEGfxTextureSet() : **AEGraphics.h**
- AEGfxTextureUnload() : **AEGraphics.h**
- AEGfxTri() : **AEGraphics.h**
- AEGfxTriAdd() : **AEGraphics.h**
- AEGfxVertexAdd() : **AEGraphics.h**
- AEGfxVertexBuffer : **AEGraphics.h**
- AEGfxVertexList : **AEGraphics.h**
- AEInputCheckCurr() : **AEInput.h**
- AEInputCheckPrev() : **AEInput.h**
- AEInputCheckReleased() : **AEInput.h**
- AEInputCheckTriggered() : **AEInput.h**
- AEInputExit() : **AEInput.h**
- AEInputGetCursorPosition() : **AEInput.h**
- AEInputGetCursorPositionDelta() : **AEInput.h**
- AEInputInit() : **AEInput.h**
- AEInputReset() : **AEInput.h**
- AEInputShowCursor() : **AEInput.h**
- AEInputUpdate() : **AEInput.h**
- AEInRange() : **AEMath.h**
- AEIsPowOf2() : **AEMath.h**
- AELineSegment2 : **AELineSegment2.h**
- AELogBase2() : **AEMath.h**
- AEMax() : **AEMath.h**
- AEMin() : **AEMath.h**
- AEMtx33 : **AEMtx33.h**
- AEMtx33Concat() : **AEMtx33.h**
- AEMtx33Determinant() : **AEMtx33.h**
- AEMtx33GetCol() : **AEMtx33.h**
- AEMtx33GetRow() : **AEMtx33.h**
- AEMtx33Identity() : **AEMtx33.h**
- AEMtx33Inverse() : **AEMtx33.h**
- AEMtx33InvTranspose() : **AEMtx33.h**
- AEMtx33MultVec() : **AEMtx33.h**
- AEMtx33MultVecArray() : **AEMtx33.h**
- AEMtx33MultVecArraySR() : **AEMtx33.h**
- AEMtx33MultVecSR() : **AEMtx33.h**

- AEMtx33Orthogonalize() : **AEMtx33.h**
- AEMtx33Rot() : **AEMtx33.h**
- AEMtx33RotDeg() : **AEMtx33.h**
- AEMtx33RowCol : **AEMtx33.h**
- AEMtx33Scale() : **AEMtx33.h**
- AEMtx33ScaleApply() : **AEMtx33.h**
- AEMtx33SetCol() : **AEMtx33.h**
- AEMtx33SetRow() : **AEMtx33.h**
- AEMtx33Trans() : **AEMtx33.h**
- AEMtx33TransApply() : **AEMtx33.h**
- AEMtx33Transpose() : **AEMtx33.h**
- AENextPowOf2() : **AEMath.h**
- AERadToDeg() : **AEMath.h**
- AERandFloat() : **AEUtil.h**
- AEReflectAnimatedCircleOnStaticCircle() : **AEMath.h**
- AEReflectAnimatedCircleOnStaticLineSegment() : **AEMath.h**
- AEReflectAnimatedPointOnStaticCircle() : **AEMath.h**
- AEReflectAnimatedPointOnStaticLineSegment() : **AEMath.h**
- AESin() : **AEMath.h**
- AESinDeg : **AEMath.h**
- AEStaticPointToStaticLineSegment() : **AEMath.h**
- AESysDoesWindowExist() : **AESystem.h**
- AESysExit() : **AESystem.h**
- AESysFrameEnd() : **AESystem.h**
- AESysFrameStart() : **AESystem.h**
- AESysGetWindowHandle() : **AESystem.h**
- AESysInit() : **AESystem.h**
- AESysReset() : **AESystem.h**
- AESysSetWindowTitle() : **AESystem.h**
- AETan() : **AEMath.h**
- AETanDeg : **AEMath.h**
- AETestCircleToCircle() : **AEMath.h**
- AETestCircleToRect() : **AEMath.h**
- AETestPointToCircle() : **AEMath.h**
- AETestPointToRect() : **AEMath.h**
- AETestRectToRect() : **AEMath.h**
- AEVec2 : **AEVec2.h**
- AEVec2Add() : **AEVec2.h**
- AEVec2CrossProductMag() : **AEVec2.h**
- AEVec2Distance() : **AEVec2.h**

- AEVec2DotProduct() : **AEVec2.h**
- AEVec2FromAngle() : **AEVec2.h**
- AEVec2Length() : **AEVec2.h**
- AEVec2Lerp() : **AEVec2.h**
- AEVec2Neg() : **AEVec2.h**
- AEVec2Normalize() : **AEVec2.h**
- AEVec2Project() : **AEVec2.h**
- AEVec2ProjectPerp() : **AEVec2.h**
- AEVec2Scale() : **AEVec2.h**
- AEVec2ScaleAdd() : **AEVec2.h**
- AEVec2ScaleSub() : **AEVec2.h**
- AEVec2Set() : **AEVec2.h**
- AEVec2SquareDistance() : **AEVec2.h**
- AEVec2SquareLength() : **AEVec2.h**
- AEVec2Sub() : **AEVec2.h**
- AEVec2Test() : **AEVec2.h**
- AEVec2Zero() : **AEVec2.h**
- AEVK_0 : **AEInput.h**
- AEVK_1 : **AEInput.h**
- AEVK_2 : **AEInput.h**
- AEVK_3 : **AEInput.h**
- AEVK_4 : **AEInput.h**
- AEVK_5 : **AEInput.h**
- AEVK_6 : **AEInput.h**
- AEVK_7 : **AEInput.h**
- AEVK_8 : **AEInput.h**
- AEVK_9 : **AEInput.h**
- AEVK_A : **AEInput.h**
- AEVK_B : **AEInput.h**
- AEVK_BACK : **AEInput.h**
- AEVK_BACKQUOTE : **AEInput.h**
- AEVK_BACKSLASH : **AEInput.h**
- AEVK_C : **AEInput.h**
- AEVK_CAPSLOCK : **AEInput.h**
- AEVK_COMMA : **AEInput.h**
- AEVK_D : **AEInput.h**
- AEVK_DELETE : **AEInput.h**
- AEVK_DOWN : **AEInput.h**
- AEVK_E : **AEInput.h**
- AEVK_END : **AEInput.h**

- AEVK_EQUAL : **AEInput.h**
- AEVK_ESCAPE : **AEInput.h**
- AEVK_F : **AEInput.h**
- AEVK_F1 : **AEInput.h**
- AEVK_F10 : **AEInput.h**
- AEVK_F11 : **AEInput.h**
- AEVK_F12 : **AEInput.h**
- AEVK_F2 : **AEInput.h**
- AEVK_F3 : **AEInput.h**
- AEVK_F4 : **AEInput.h**
- AEVK_F5 : **AEInput.h**
- AEVK_F6 : **AEInput.h**
- AEVK_F7 : **AEInput.h**
- AEVK_F8 : **AEInput.h**
- AEVK_F9 : **AEInput.h**
- AEVK_G : **AEInput.h**
- AEVK_H : **AEInput.h**
- AEVK_HOME : **AEInput.h**
- AEVK_I : **AEInput.h**
- AEVK_INSERT : **AEInput.h**
- AEVK_J : **AEInput.h**
- AEVK_K : **AEInput.h**
- AEVK_L : **AEInput.h**
- AEVK_LALT : **AEInput.h**
- AEVK_LBRACKET : **AEInput.h**
- AEVK_LBUTTON : **AEInput.h**
- AEVK_LCTRL : **AEInput.h**
- AEVK_LEFT : **AEInput.h**
- AEVK_LSHIFT : **AEInput.h**
- AEVK_M : **AEInput.h**
- AEVK_MBUTTON : **AEInput.h**
- AEVK_MINUS : **AEInput.h**
- AEVK_N : **AEInput.h**
- AEVK_NUM_DIVIDE : **AEInput.h**
- AEVK_NUM_MINUS : **AEInput.h**
- AEVK_NUM_MULTIPLY : **AEInput.h**
- AEVK_NUM_PERIOD : **AEInput.h**
- AEVK_NUM_PLUS : **AEInput.h**
- AEVK_NUMLOCK : **AEInput.h**
- AEVK_NUMPAD0 : **AEInput.h**

- AEVK_NUMPAD1 : **AEInput.h**
- AEVK_NUMPAD2 : **AEInput.h**
- AEVK_NUMPAD3 : **AEInput.h**
- AEVK_NUMPAD4 : **AEInput.h**
- AEVK_NUMPAD5 : **AEInput.h**
- AEVK_NUMPAD6 : **AEInput.h**
- AEVK_NUMPAD7 : **AEInput.h**
- AEVK_NUMPAD8 : **AEInput.h**
- AEVK_NUMPAD9 : **AEInput.h**
- AEVK_O : **AEInput.h**
- AEVK_P : **AEInput.h**
- AEVK_PAGEDOWN : **AEInput.h**
- AEVK_PAGEUP : **AEInput.h**
- AEVK_PAUSE : **AEInput.h**
- AEVK_PERIOD : **AEInput.h**
- AEVK_PRINTSCREEN : **AEInput.h**
- AEVK_Q : **AEInput.h**
- AEVK_QUOTE : **AEInput.h**
- AEVK_R : **AEInput.h**
- AEVK_RALT : **AEInput.h**
- AEVK_RBRACKET : **AEInput.h**
- AEVK_RBUTTON : **AEInput.h**
- AEVK_RCTRL : **AEInput.h**
- AEVK_RETURN : **AEInput.h**
- AEVK_RIGHT : **AEInput.h**
- AEVK_RSHIFT : **AEInput.h**
- AEVK_S : **AEInput.h**
- AEVK_SCROLLLOCK : **AEInput.h**
- AEVK_SEMICOLON : **AEInput.h**
- AEVK_SLASH : **AEInput.h**
- AEVK_SPACE : **AEInput.h**
- AEVK_T : **AEInput.h**
- AEVK_TAB : **AEInput.h**
- AEVK_U : **AEInput.h**
- AEVK_UP : **AEInput.h**
- AEVK_V : **AEInput.h**
- AEVK_W : **AEInput.h**
- AEVK_X : **AEInput.h**
- AEVK_Y : **AEInput.h**
- AEVK_Z : **AEInput.h**

- AEWrap() : **AEMath.h**

Alpha Engine

Here is a list of all file members with links to the files they belong to:

- d -

- DECLARE_FUNCTION_FOR_ANDROID : **AEExport.h** , **AEGraphics.h**
- DECLARE_FUNCTION_FOR_ANDROID_2_INT : **AEExport.h** , **AEGraphics.h**

Alpha Engine

Here is a list of all file members with links to the files they belong to:

- e -

- EPSILON : **AEExport.h**
- EXPORT_ANDROID : **AEExport.h**
- EXPORT_WINDOWS : **AEExport.h**

Alpha Engine

Here is a list of all file members with links to the files they belong to:

- f -

- f32 : **AETypes.h**
- f64 : **AETypes.h**

Alpha Engine

Here is a list of all file members with links to the files they belong to:

- g -

- gAEGameStateCurr : **AEGameStateMgr.h**
- gAEGameStateInit : **AEGameStateMgr.h**
- gAEGameStateNext : **AEGameStateMgr.h**
- gAEGameStatePrev : **AEGameStateMgr.h**
- gAESysAppActive : **AESystem.h**
- gAESysWindowHandle : **AESystem.h**
- ghAESysAppInstance : **AESystem.h**
- gpAESysWinClassName : **AESystem.h**
- gpAESysWinTitle : **AESystem.h**

Alpha Engine

Here is a list of all file members with links to the files they belong to:

- h -

- HALF_PI : **AEExport.h**

Alpha Engine

Here is a list of all file members with links to the files they belong to:

- i -

- IMPLEMENT_FUNCTION_FOR_ANDROID : **AEExport.h**
- IMPLEMENT_FUNCTION_FOR_ANDROID_2_INT : **AEExport.h**
- isEqual : **AEUtil.h**
- isZero : **AEUtil.h**

Alpha Engine

Here is a list of all file members with links to the files they belong to:

- p -

- PI : **AEExport.h**
- PRINT : **AEExport.h**
- PRINT_INFO : **AEExport.h**

Alpha Engine

Here is a list of all file members with links to the files they belong to:

- r -

- ReadFromFile() : **AEUtil.h**

Alpha Engine

Here is a list of all file members with links to the files they belong to:

- s -

- s16 : **AETypes.h**
- s32 : **AETypes.h**
- s64 : **AETypes.h**
- s8 : **AETypes.h**

Alpha Engine

Here is a list of all file members with links to the files they belong to:

- t -

- TWO_PI : **AEExport.h**

Alpha Engine

Here is a list of all file members with links to the files they belong to:

- u -

- u16 : **AETypes.h**
- u32 : **AETypes.h**
- u64 : **AETypes.h**
- u8 : **AETypes.h**

Alpha Engine

Here is a list of all file members with links to the files they belong to:

- w -

- winClass : **AESystem.h**

Alpha Engine

- a -

- AEACos() : **AEMath.h**
- AEAnimatedCircleToStaticCircle() : **AEMath.h**
- AEAnimatedCircleToStaticLineSegment() : **AEMath.h**
- AEAnimatedPointToStaticCircle() : **AEMath.h**
- AEAnimatedPointToStaticLineSegment() : **AEMath.h**
- AEASin() : **AEMath.h**
- AEATan() : **AEMath.h**
- AEBuildLineSegment2() : **AELineSegment2.h**
- AECalcDistCircleToCircle() : **AEMath.h**
- AECalcDistCircleToRect() : **AEMath.h**
- AECalcDistPointToCircle() : **AEMath.h**
- AECalcDistPointToConvexPoly() : **AEMath.h**
- AECalcDistPointToLineSeg() : **AEMath.h**
- AECalcDistPointToRect() : **AEMath.h**
- AECalcDistRectToRect() : **AEMath.h**
- AEClamp() : **AEMath.h**
- AECos() : **AEMath.h**
- AEDegToRad() : **AEMath.h**
- AEFrameRateControllerEnd() : **AEFrameRateController.h**
- AEFrameRateControllerGetFrameCount() : **AEFrameRateController.h**
- AEFrameRateControllerGetFrameTime() : **AEFrameRateController.h**
- AEFrameRateControllerInit() : **AEFrameRateController.h**
- AEFrameRateControllerReset() : **AEFrameRateController.h**

- AEFrameRateControllerStart() : **AEFrameRateController.h**
- AEGameStateMgrAdd() : **AEGameStateMgr.h**
- AEGameStateMgrInit() : **AEGameStateMgr.h**
- AEGameStateMgrUpdate() : **AEGameStateMgr.h**
- AEGetTime() : **AEUtil.h**
- AEGfxAxis() : **AEGraphics.h**
- AEGfxBox() : **AEGraphics.h**
- AEGfxColInterp() : **AEGraphics.h**
- AEGfxCone() : **AEGraphics.h**
- AEGfxEnd() : **AEGraphics.h**
- AEGfxExit() : **AEGraphics.h**
- AEGfxGetCamPosition() : **AEGraphics.h**
- AEGfxGetWinMaxX() : **AEGraphics.h**
- AEGfxGetWinMaxY() : **AEGraphics.h**
- AEGfxGetWinMinX() : **AEGraphics.h**
- AEGfxGetWinMinY() : **AEGraphics.h**
- AEGfxInit() : **AEGraphics.h**
- AEGfxLine() : **AEGraphics.h**
- AEGfxMeshDraw() : **AEGraphics.h**
- AEGfxMeshEnd() : **AEGraphics.h**
- AEGfxMeshFree() : **AEGraphics.h**
- AEGfxMeshStart() : **AEGraphics.h**
- AEGfxPoint() : **AEGraphics.h**
- AEGfxPrint() : **AEGraphics.h**
- AEGfxQuad() : **AEGraphics.h**
- AEGfxReset() : **AEGraphics.h**
- AEGfxSaveTextureToFile() : **AEGraphics.h**
- AEGfxSetBackgroundColor() : **AEGraphics.h**
- AEGfxSetBlendColor() : **AEGraphics.h**
- AEGfxSetBlendMode() : **AEGraphics.h**
- AEGfxSetCamPosition() : **AEGraphics.h**
- AEGfxSetPosition() : **AEGraphics.h**
- AEGfxSetRenderMode() : **AEGraphics.h**
- AEGfxSetTextureMode() : **AEGraphics.h**
- AEGfxSetTintColor() : **AEGraphics.h**
- AEGfxSetTransform() : **AEGraphics.h**
- AEGfxSetTransform3D() : **AEGraphics.h**
- AEGfxSetTransparency() : **AEGraphics.h**
- AEGfxSphere() : **AEGraphics.h**
- AEGfxStart() : **AEGraphics.h**

- AEGfxTextureLoad() : **AEGraphics.h**
- AEGfxTextureLoadFromMemory() : **AEGraphics.h**
- AEGfxTextureSet() : **AEGraphics.h**
- AEGfxTextureUnload() : **AEGraphics.h**
- AEGfxTri() : **AEGraphics.h**
- AEGfxTriAdd() : **AEGraphics.h**
- AEGfxVertexAdd() : **AEGraphics.h**
- AEInputCheckCurr() : **AEInput.h**
- AEInputCheckPrev() : **AEInput.h**
- AEInputCheckReleased() : **AEInput.h**
- AEInputCheckTriggered() : **AEInput.h**
- AEInputExit() : **AEInput.h**
- AEInputGetCursorPosition() : **AEInput.h**
- AEInputGetCursorPositionDelta() : **AEInput.h**
- AEInputInit() : **AEInput.h**
- AEInputReset() : **AEInput.h**
- AEInputShowCursor() : **AEInput.h**
- AEInputUpdate() : **AEInput.h**
- AEInRange() : **AEMath.h**
- AEIsPowOf2() : **AEMath.h**
- AELogBase2() : **AEMath.h**
- AEMax() : **AEMath.h**
- AEMin() : **AEMath.h**
- AEMtx33Concat() : **AEMtx33.h**
- AEMtx33Determinant() : **AEMtx33.h**
- AEMtx33GetCol() : **AEMtx33.h**
- AEMtx33GetRow() : **AEMtx33.h**
- AEMtx33Identity() : **AEMtx33.h**
- AEMtx33Inverse() : **AEMtx33.h**
- AEMtx33InvTranspose() : **AEMtx33.h**
- AEMtx33MultVec() : **AEMtx33.h**
- AEMtx33MultVecArray() : **AEMtx33.h**
- AEMtx33MultVecArraySR() : **AEMtx33.h**
- AEMtx33MultVecSR() : **AEMtx33.h**
- AEMtx33Orthogonalize() : **AEMtx33.h**
- AEMtx33Rot() : **AEMtx33.h**
- AEMtx33RotDeg() : **AEMtx33.h**
- AEMtx33Scale() : **AEMtx33.h**
- AEMtx33ScaleApply() : **AEMtx33.h**
- AEMtx33SetCol() : **AEMtx33.h**

- AEMtx33SetRow() : **AEMtx33.h**
- AEMtx33Trans() : **AEMtx33.h**
- AEMtx33TransApply() : **AEMtx33.h**
- AEMtx33Transpose() : **AEMtx33.h**
- AENextPowOf2() : **AEMath.h**
- AERadToDeg() : **AEMath.h**
- AERandFloat() : **AEUtil.h**
- AEReflectAnimatedCircleOnStaticCircle() : **AEMath.h**
- AEReflectAnimatedCircleOnStaticLineSegment() : **AEMath.h**
- AEReflectAnimatedPointOnStaticCircle() : **AEMath.h**
- AEReflectAnimatedPointOnStaticLineSegment() : **AEMath.h**
- AESin() : **AEMath.h**
- AEStaticPointToStaticLineSegment() : **AEMath.h**
- AESysDoesWindowExist() : **AESystem.h**
- AESysExit() : **AESystem.h**
- AESysFrameEnd() : **AESystem.h**
- AESysFrameStart() : **AESystem.h**
- AESysGetWindowHandle() : **AESystem.h**
- AESysInit() : **AESystem.h**
- AESysReset() : **AESystem.h**
- AESysSetWindowTitle() : **AESystem.h**
- AETan() : **AEMath.h**
- AETestCircleToCircle() : **AEMath.h**
- AETestCircleToRect() : **AEMath.h**
- AETestPointToCircle() : **AEMath.h**
- AETestPointToRect() : **AEMath.h**
- AETestRectToRect() : **AEMath.h**
- AEVec2Add() : **AEVec2.h**
- AEVec2CrossProductMag() : **AEVec2.h**
- AEVec2Distance() : **AEVec2.h**
- AEVec2DotProduct() : **AEVec2.h**
- AEVec2FromAngle() : **AEVec2.h**
- AEVec2Length() : **AEVec2.h**
- AEVec2Lerp() : **AEVec2.h**
- AEVec2Neg() : **AEVec2.h**
- AEVec2Normalize() : **AEVec2.h**
- AEVec2Project() : **AEVec2.h**
- AEVec2ProjectPerp() : **AEVec2.h**
- AEVec2Scale() : **AEVec2.h**
- AEVec2ScaleAdd() : **AEVec2.h**

- AEVec2ScaleSub() : **AEVec2.h**
- AEVec2Set() : **AEVec2.h**
- AEVec2SquareDistance() : **AEVec2.h**
- AEVec2SquareLength() : **AEVec2.h**
- AEVec2Sub() : **AEVec2.h**
- AEVec2Test() : **AEVec2.h**
- AEVec2Zero() : **AEVec2.h**
- AEWrap() : **AEMath.h**

- d -

- DECLARE_FUNCTION_FOR_ANDROID() : **AEGraphics.h**
- DECLARE_FUNCTION_FOR_ANDROID_2_INT() : **AEGraphics.h**

- r -

- ReadFromFile() : **AEUtil.h**

Alpha Engine

- AEGameStateDraw : **AEGameStateMgr.h**
- AEGameStateFree : **AEGameStateMgr.h**
- AEGameStateInit : **AEGameStateMgr.h**
- AEGameStateLoad : **AEGameStateMgr.h**
- AEGameStateUnload : **AEGameStateMgr.h**
- AEGameStateUpdate : **AEGameStateMgr.h**
- gAEGameStateCurr : **AEGameStateMgr.h**
- gAEGameStateInit : **AEGameStateMgr.h**
- gAEGameStateNext : **AEGameStateMgr.h**
- gAEGameStatePrev : **AEGameStateMgr.h**
- gAESysAppActive : **AESystem.h**
- gAESysWindowHandle : **AESystem.h**
- ghAESysAppInstance : **AESystem.h**
- gpAESysWinClassName : **AESystem.h**
- gpAESysWinTitle : **AESystem.h**
- winClass : **AESystem.h**

Alpha Engine

- AEGfxSurface : **AEGraphics.h**
- AEGfxTexture : **AEGraphics.h**
- AEGfxVertexBuffer : **AEGraphics.h**
- AEGfxVertexList : **AEGraphics.h**
- AELineSegment2 : **AELineSegment2.h**
- AEMtx33 : **AEMtx33.h**
- AEVec2 : **AEVec2.h**
- f32 : **AETypes.h**
- f64 : **AETypes.h**
- s16 : **AETypes.h**
- s32 : **AETypes.h**
- s64 : **AETypes.h**
- s8 : **AETypes.h**
- u16 : **AETypes.h**
- u32 : **AETypes.h**
- u64 : **AETypes.h**
- u8 : **AETypes.h**

Alpha Engine

- AEGfxBlendMode : **AEGraphics.h**
- AEGfxMeshDrawMode : **AEGraphics.h**
- AEGfxRenderMode : **AEGraphics.h**
- AEGfxTextureMode : **AEGraphics.h**

Alpha Engine

- AE_GFX_BM_ADD : **AEGraphics.h**
- AE_GFX_BM_BLEND : **AEGraphics.h**
- AE_GFX_BM_NONE : **AEGraphics.h**
- AE_GFX_BM_NUM : **AEGraphics.h**
- AE_GFX_MDM_LINES : **AEGraphics.h**
- AE_GFX_MDM_LINES_STRIP : **AEGraphics.h**
- AE_GFX_MDM_NUM : **AEGraphics.h**
- AE_GFX_MDM_POINTS : **AEGraphics.h**
- AE_GFX_MDM_TRIANGLES : **AEGraphics.h**
- AE_GFX_RM_COLOR : **AEGraphics.h**
- AE_GFX_RM_NONE : **AEGraphics.h**
- AE_GFX_RM_NUM : **AEGraphics.h**
- AE_GFX_RM_TEXTURE : **AEGraphics.h**
- AE_GFX_TM_AVERAGE : **AEGraphics.h**
- AE_GFX_TM_PRECISE : **AEGraphics.h**

Alpha Engine

- a -

- AE_API : **AEExport.h**
- AE_ASSERT : **AEEngine.h**
- AE_ASSERT_ALLOC : **AEEngine.h**
- AE_ASSERT_MESG : **AEEngine.h**
- AE_ASSERT_PARM : **AEEngine.h**
- AE_FATAL_ERROR : **AEEngine.h**
- AE_GS_QUIT : **AEGameStateMgr.h**
- AE_GS_RESTART : **AEGameStateMgr.h**
- AE_WARNING : **AEEngine.h**
- AE_WARNING_MESG : **AEEngine.h**
- AE_WARNING_PARM : **AEEngine.h**
- AEACosDeg : **AEMath.h**
- AEASinDeg : **AEMath.h**
- AEATanDeg : **AEMath.h**
- AECosDeg : **AEMath.h**
- AEMtx33RowCol : **AEMtx33.h**
- AESinDeg : **AEMath.h**
- AETanDeg : **AEMath.h**
- AEVK_0 : **AEInput.h**
- AEVK_1 : **AEInput.h**
- AEVK_2 : **AEInput.h**
- AEVK_3 : **AEInput.h**
- AEVK_4 : **AEInput.h**
- AEVK_5 : **AEInput.h**
- AEVK_6 : **AEInput.h**

- AEVK_7 : **AEInput.h**
- AEVK_8 : **AEInput.h**
- AEVK_9 : **AEInput.h**
- AEVK_A : **AEInput.h**
- AEVK_B : **AEInput.h**
- AEVK_BACK : **AEInput.h**
- AEVK_BACKQUOTE : **AEInput.h**
- AEVK_BACKSLASH : **AEInput.h**
- AEVK_C : **AEInput.h**
- AEVK_CAPSLOCK : **AEInput.h**
- AEVK_COMMA : **AEInput.h**
- AEVK_D : **AEInput.h**
- AEVK_DELETE : **AEInput.h**
- AEVK_DOWN : **AEInput.h**
- AEVK_E : **AEInput.h**
- AEVK_END : **AEInput.h**
- AEVK_EQUAL : **AEInput.h**
- AEVK_ESCAPE : **AEInput.h**
- AEVK_F : **AEInput.h**
- AEVK_F1 : **AEInput.h**
- AEVK_F10 : **AEInput.h**
- AEVK_F11 : **AEInput.h**
- AEVK_F12 : **AEInput.h**
- AEVK_F2 : **AEInput.h**
- AEVK_F3 : **AEInput.h**
- AEVK_F4 : **AEInput.h**
- AEVK_F5 : **AEInput.h**
- AEVK_F6 : **AEInput.h**
- AEVK_F7 : **AEInput.h**
- AEVK_F8 : **AEInput.h**
- AEVK_F9 : **AEInput.h**
- AEVK_G : **AEInput.h**
- AEVK_H : **AEInput.h**
- AEVK_HOME : **AEInput.h**
- AEVK_I : **AEInput.h**
- AEVK_INSERT : **AEInput.h**
- AEVK_J : **AEInput.h**
- AEVK_K : **AEInput.h**
- AEVK_L : **AEInput.h**
- AEVK_LALT : **AEInput.h**

- AEVK_LBRACKET : **AEInput.h**
- AEVK_LBUTTON : **AEInput.h**
- AEVK_LCTRL : **AEInput.h**
- AEVK_LEFT : **AEInput.h**
- AEVK_LSHIFT : **AEInput.h**
- AEVK_M : **AEInput.h**
- AEVK_MBUTTON : **AEInput.h**
- AEVK_MINUS : **AEInput.h**
- AEVK_N : **AEInput.h**
- AEVK_NUM_DIVIDE : **AEInput.h**
- AEVK_NUM_MINUS : **AEInput.h**
- AEVK_NUM_MULTIPLY : **AEInput.h**
- AEVK_NUM_PERIOD : **AEInput.h**
- AEVK_NUM_PLUS : **AEInput.h**
- AEVK_NUMLOCK : **AEInput.h**
- AEVK_NUMPAD0 : **AEInput.h**
- AEVK_NUMPAD1 : **AEInput.h**
- AEVK_NUMPAD2 : **AEInput.h**
- AEVK_NUMPAD3 : **AEInput.h**
- AEVK_NUMPAD4 : **AEInput.h**
- AEVK_NUMPAD5 : **AEInput.h**
- AEVK_NUMPAD6 : **AEInput.h**
- AEVK_NUMPAD7 : **AEInput.h**
- AEVK_NUMPAD8 : **AEInput.h**
- AEVK_NUMPAD9 : **AEInput.h**
- AEVK_O : **AEInput.h**
- AEVK_P : **AEInput.h**
- AEVK_PAGEDOWN : **AEInput.h**
- AEVK_PAGEUP : **AEInput.h**
- AEVK_PAUSE : **AEInput.h**
- AEVK_PERIOD : **AEInput.h**
- AEVK_PRINTSCREEN : **AEInput.h**
- AEVK_Q : **AEInput.h**
- AEVK_QUOTE : **AEInput.h**
- AEVK_R : **AEInput.h**
- AEVK_RALT : **AEInput.h**
- AEVK_RBRACKET : **AEInput.h**
- AEVK_RBUTTON : **AEInput.h**
- AEVK_RCTRL : **AEInput.h**
- AEVK_RETURN : **AEInput.h**

- AEVK_RIGHT : **AEInput.h**
- AEVK_RSHIFT : **AEInput.h**
- AEVK_S : **AEInput.h**
- AEVK_SCROLLLOCK : **AEInput.h**
- AEVK_SEMICOLON : **AEInput.h**
- AEVK_SLASH : **AEInput.h**
- AEVK_SPACE : **AEInput.h**
- AEVK_T : **AEInput.h**
- AEVK_TAB : **AEInput.h**
- AEVK_U : **AEInput.h**
- AEVK_UP : **AEInput.h**
- AEVK_V : **AEInput.h**
- AEVK_W : **AEInput.h**
- AEVK_X : **AEInput.h**
- AEVK_Y : **AEInput.h**
- AEVK_Z : **AEInput.h**

- d -

- DECLARE_FUNCTION_FOR_ANDROID : **AEExport.h**
- DECLARE_FUNCTION_FOR_ANDROID_2_INT : **AEExport.h**

- e -

- EPSILON : **AEExport.h**
- EXPORT_ANDROID : **AEExport.h**
- EXPORT_WINDOWS : **AEExport.h**

- h -

- HALF_PI : **AEExport.h**

- i -

- IMPLEMENT_FUNCTION_FOR_ANDROID : **AEExport.h**
- IMPLEMENT_FUNCTION_FOR_ANDROID_2_INT : **AEExport.h**
- isEqual : **AEUtil.h**
- isZero : **AEUtil.h**

- p -

- PI : **AEExport.h**
- PRINT : **AEExport.h**
- PRINT_INFO : **AEExport.h**

- t -

- TWO_PI : **AEExport.h**

Alpha Engine

AEGfxTexture Member List

This is the complete list of members for **AEGfxTexture**, including all inherited members.

mpName	**AEGfxTexture**
mpSurface	**AEGfxTexture**

Alpha Engine

AEGraphics.h

Go to the documentation of this file.

```
  1 /*****************************************************************************/
 15 /*****************************************************************************/
 16
 17 #ifndef AE_GRAPHICS_H
 18 #define AE_GRAPHICS_H
 19
 20 // ---------------------------------------------------------------------------
 21 // Defines
 22
 24 typedef enum
 25 {
 26     AE_GFX_RM_NONE,
 27     AE_GFX_RM_COLOR,
 28     AE_GFX_RM_TEXTURE,
 29
 30     AE_GFX_RM_NUM
 31 }AEGfxRenderMode;
 32
 33
 34 // ---------------------------------------------------------------------------
 35
 37 typedef enum
```

```
38  {
39      AE_GFX_BM_NONE = 0,
40      AE_GFX_BM_BLEND,
41      AE_GFX_BM_ADD,
42
43      AE_GFX_BM_NUM
44  }AEGfxBlendMode;
45
46  // ----------------------------------------
    ---------------------------------
47
48  typedef enum
49  {
50      AE_GFX_TM_PRECISE = 0,
51      AE_GFX_TM_AVERAGE
52  }AEGfxTextureMode;
53
54  // ----------------------------------------
    ---------------------------------
55
56  typedef enum
57  {
58      AE_GFX_MDM_POINTS = 0,
59      AE_GFX_MDM_LINES,
60      AE_GFX_MDM_LINES_STRIP,
61      AE_GFX_MDM_TRIANGLES,
62
63      // Keep this one last
64      AE_GFX_MDM_NUM
65  }AEGfxMeshDrawMode;
66
67  // ----------------------------------------
    ---------------------------------
68  // Struct/Class definitions
69
70  typedef struct AEGfxVertexBuffer
    AEGfxVertexBuffer;
```

```
 71
 72 typedef struct AEGfxVertexList
 73 {
 74     AEGfxVertexBuffer       *mpVtxBuffer;
 75     u32 vtxNum;
 76 }AEGfxVertexList;
 77
 78 // ---------------------------------------
    ---------------------------------
 79
 80 typedef struct AEGfxSurface AEGfxSurface;
 81
 82 typedef struct AEGfxTexture
 83 {
 84     AEGfxSurface *mpSurface;
 85     s8 mpName[256];
 86 }AEGfxTexture;
 87
 88 // ---------------------------------------
    ---------------------------------
 89 // Extern variables
 90
 91 // ---------------------------------------
    ---------------------------------
 92 // Function prototypes
 93 // ---------------------------------------
    ---------------------------------
 94
 95 #ifdef __cplusplus
 96
 97 extern "C"
 98 {
 99 #endif
100
101 // ---------------------------------------
    ---------------------------------
102
```

```
103 /*************************************************
    *********************************/
122 /*************************************************
    *********************************/
123 AE_API s32 AEGfxInit(s32 Width, s32 Height);
124 DECLARE_FUNCTION_FOR_ANDROID_2_INT(AEGfxInit,
    s32 Width, s32 Height);
125
126 /*************************************************
    *********************************/
137 /*************************************************
    *********************************/
138 AE_API void AEGfxReset();
139
140 /*************************************************
    *********************************/
151 /*************************************************
    *********************************/
152 AE_API void AEGfxExit();
153
154 /*************************************************
    *********************************/
165 /*************************************************
    *********************************/
166 AE_API void AEGfxStart();
167 DECLARE_FUNCTION_FOR_ANDROID(AEGfxStart);
168
169 /*************************************************
    *********************************/
180 /*************************************************
    *********************************/
181 AE_API void AEGfxEnd();
182 DECLARE_FUNCTION_FOR_ANDROID(AEGfxEnd);
183
184 /*************************************************
    *********************************/
204 /*************************************************
```

```
    ************************************/
205 AE_API void AEGfxSetBackgroundColor(f32 Red,
    f32 Green, f32 Blue);
206
207 /*************************************************
    ************************************/
223 /*************************************************
    ************************************/
224 AE_API void
    AEGfxSetRenderMode(AEGfxRenderMode RenderMode);
225
226 /*************************************************
    ************************************/
238 /*************************************************
    ************************************/
239 AE_API void AEGfxSetBlendMode(AEGfxBlendMode
    BlendMode);
240
241 /*************************************************
    ************************************/
250 /*************************************************
    ************************************/
251 AE_API f32 AEGfxGetWinMinX(void);
252
253 /*************************************************
    ************************************/
262 /*************************************************
    ************************************/
263 AE_API f32 AEGfxGetWinMaxX(void);
264
265 /*************************************************
    ************************************/
274 /*************************************************
    ************************************/
275 AE_API f32 AEGfxGetWinMinY(void);
276
277 /*************************************************
```

```
    ************************************/
286 /*****************************************
    ************************************/
287 AE_API f32 AEGfxGetWinMaxY(void);
288
289 /*****************************************
    ************************************/
304 /*****************************************
    ************************************/
305 AE_API void AEGfxSetCamPosition(f32 X, f32
    Y);
306
307 /*****************************************
    ************************************/
322 /*****************************************
    ************************************/
323 AE_API void AEGfxGetCamPosition(f32 *pX, f32
    *pY);
324
325 // Sets/Gets the camera distance, used to
    zoom in/out
326 //AE_API void AEGfxSetCamZoom(f32 Distance);
327 //AE_API f32 AEGfxGetCamZoom();
328
329 /*****************************************
    ************************************/
344 /*****************************************
    ************************************/
345 AE_API void AEGfxSetPosition(f32 X, f32 Y);
346
347 /*****************************************
    ************************************/
363 /*****************************************
    ************************************/
364 AE_API void AEGfxSetTransform(f32
    pTransform[3][3]);
365
```

```
366 /*****************************************************
    ***********************************/
381 /*****************************************************
    ***********************************/
382 AE_API void AEGfxSetTransform3D(f32
    pTransform[4][4]);
383
384 /*****************************************************
    ***********************************/
400 /*****************************************************
    ***********************************/
401 AE_API void AEGfxSetTransparency(f32 Alpha);
402
403 /*****************************************************
    ***********************************/
429 /*****************************************************
    ***********************************/
430 AE_API void AEGfxSetBlendColor(f32 Red, f32
    Green, f32 Blue, f32 Alpha);
431
432 /*****************************************************
    ***********************************/
458 /*****************************************************
    ***********************************/
459 AE_API void AEGfxSetTintColor(float Red,
    float Green, float Blue, float Alpha);
460
461 /*****************************************************
    ***********************************/
473 /*****************************************************
    ***********************************/
474 AE_API void AEGfxMeshStart  ();
475
476 /*****************************************************
    ***********************************/
536 /*****************************************************
    ***********************************/
```

```
537| AE_API void AEGfxTriAdd     (f32 x0, f32 y0,
   u32 c0, f32 tu0, f32 tv0,
538|                                                f32
   x1, f32 y1, u32 c1, f32 tu1, f32 tv1,
539|                                                f32
   x2, f32 y2, u32 c2, f32 tu2, f32 tv2);
540|
541| /*************************************************
   *********************************/
571| /*************************************************
   *********************************/
572| AE_API void AEGfxVertexAdd  (f32 x0, f32 y0,
   u32 c0, f32 tu0, f32 tv0);
573|
574| /*************************************************
   *********************************/
586| /*************************************************
   *********************************/
587| AE_API AEGfxVertexList* AEGfxMeshEnd    ();
588|
589| /*************************************************
   *********************************/
608| /*************************************************
   *********************************/
609| AE_API void AEGfxMeshDraw   (AEGfxVertexList*
   pVertexList, AEGfxMeshDrawMode MeshDrawMode);
610|
611| /*************************************************
   *********************************/
626| /*************************************************
   *********************************/
627| AE_API void AEGfxMeshFree   (AEGfxVertexList*
   pVertexList);
628|
629| /*************************************************
   *********************************/
642| /*************************************************
```

```
    ***********************************/
643 AE_API AEGfxTexture*
    AEGfxTextureLoad(const s8 *pFileName);
644
645 /*****************************************
    ***********************************/
666 /*****************************************
    ***********************************/
667 AE_API void AEGfxTextureSet(AEGfxTexture
    *pTexture, f32 offset_x, f32 offset_y);
668
669 /*****************************************
    ***********************************/
681 /*****************************************
    ***********************************/
682 AE_API void AEGfxTextureUnload(AEGfxTexture
    *pTexture);
683
684 /*****************************************
    ***********************************/
704 /*****************************************
    ***********************************/
705 AE_API AEGfxTexture*
    AEGfxTextureLoadFromMemory(u8 *pColors, u32
    Width, u32 Height);
706
707 /*****************************************
    ***********************************/
724 /*****************************************
    ***********************************/
725 AE_API void
    AEGfxSaveTextureToFile(AEGfxTexture* pTexture,
    s8 *pFileName);
726
727 /*****************************************
    ***********************************/
739 /*****************************************
```

```
    ***********************************/
740 AE_API void
    AEGfxSetTextureMode(AEGfxTextureMode
    TextureMode);
741
742 /*************************************************
    ***********************************/
769 /*************************************************
    ***********************************/
770 AE_API void AEGfxPoint      (f32 x0, f32 y0,
    f32 z0, u32 c0);
771
772 /*************************************************
    ***********************************/
832 /*************************************************
    ***********************************/
833 AE_API void AEGfxLine       (f32 x0, f32 y0,
    f32 z0, f32 r0, f32 g0, f32 b0, f32 a0,
834                                              f32
    x1, f32 y1, f32 z1, f32 r1, f32 g1, f32 b1, f32
    a1);
835
836 /*************************************************
    ***********************************/
890 /*************************************************
    ***********************************/
891 AE_API void AEGfxTri        (f32 x0, f32 y0,
    f32 z0, u32 c0,
892                                              f32
    x1, f32 y1, f32 z1, u32 c1,
893                                              f32
    x2, f32 y2, f32 z2, u32 c2);
894
895 /*************************************************
    ***********************************/
962 /*************************************************
    ***********************************/
```

```
963| AE_API void AEGfxQuad       (f32 x0, f32 y0,
   f32 z0, u32 c0,
964|                                                f32
   x1, f32 y1, f32 z1, u32 c1,
965|                                                f32
   x2, f32 y2, f32 z2, u32 c2,
966|                                                f32
   x3, f32 y3, f32 z3, u32 c3);
967|
968| /*************************************************
   *********************************/
1012| /*************************************************
   *********************************/
1013| AE_API void AEGfxBox        (f32 x0, f32 y0,
   f32 z0, f32 sizeX, f32 sizeY, f32 sizeZ, u32
   c0, u32 c1);
1014|
1015| /*************************************************
   *********************************/
1059| /*************************************************
   *********************************/
1060| AE_API void AEGfxSphere     (f32 x0, f32 y0,
   f32 z0,
1061|                                                f32
   radius, u32 c0, u32 c1,
1062|                                                u32
   division);
1063|
1064| /*************************************************
   *********************************/
1121| /*************************************************
   *********************************/
1122| AE_API void AEGfxCone       (f32 x0, f32 y0,
   f32 z0,
1123|                                                f32
   x1, f32 y1, f32 z1,
1124|                                                f32
```

```
    radius, u32 c0, u32 c1, u32 division);
1125
1126 /*****************************************
    ***********************************/
1145 /*****************************************
    ***********************************/
1146 AE_API void AEGfxAxis       (f32 scale);
1147
1148 /*****************************************
    ***********************************/
1171 /*****************************************
    ***********************************/
1172 AE_API u32 AEGfxColInterp  (u32 c0, u32 c1,
    f32 t);
1173
1174 /*****************************************
    ***********************************/
1201 /*****************************************
    ***********************************/
1202 AE_API void AEGfxPrint      (s32 x, s32 y,
    u32 color, s8* pStr);
1203
1204 // ---------------------------------------
    --------------------------------
1205
1206 #ifdef __cplusplus
1207 }
1208 #endif
1209
1210 // ---------------------------------------
    --------------------------------
1211
1212 #endif // AE_GRAPHICS_H
1213
```

Alpha Engine

AEGfxVertexList Member List

This is the complete list of members for **AEGfxVertexList**, including all inherited members.

mpVtxBuffer	**AEGfxVertexList**
vtxNum	**AEGfxVertexList**

Alpha Engine

AELineSegment2 Member List

This is the complete list of members for **AELineSegment2**, including all inherited members.

mN	**AELineSegment2**
mNdotP0	**AELineSegment2**
mP0	**AELineSegment2**
mP1	**AELineSegment2**

Alpha Engine

AELineSegment2.h

Go to the documentation of this file.

```
 1 /*****************************************************
   ***********************************/
15 /*****************************************************
   ***********************************/
16
17 #ifndef AE_LS2_H
18 #define AE_LS2_H
19
20 // ----------------------------------------------
   ---------------------------------
21
22 typedef struct AELineSegment2
23 {
24     AEVec2 mP0;
25     AEVec2 mP1;
26     AEVec2 mN;
27     float mNdotP0;
28 }AELineSegment2;
29
30
31 // ----------------------------------------------
   ---------------------------------
32 #ifdef __cplusplus
33 extern "C"
34 {
35 #endif
```

```
36
37 /*****************************************************
   ********************************/
63 /*****************************************************
   ********************************/
64 AE_API s32 AEBuildLineSegment2(AELineSegment2
   *pLS, AEVec2 *pPt0, AEVec2 *pPt1);
65
66 #ifdef __cplusplus
67 }
68 #endif
69
70 // -----------------------------------------
   ----------------------------------
71
72 #endif // VEC2_H
```

Alpha Engine

AEMtx33 Member List

This is the complete list of members for **AEMtx33**, including all inherited members.

m	**AEMtx33**

Alpha Engine

AEMtx33.h

Go to the documentation of this file.

```
  1 /*****************************************************
    *********************************/
 15 /*****************************************************
    *********************************/
 16
 17 #ifndef AE_MTX33_H
 18 #define AE_MTX33_H
 19
 20 // ---------------------------------------------------
    ---------------------------------
 21 // Defines
 22
 23 #define AEMtx33RowCol(pMtx, row, col) (pMtx)-
    >m[(row)][(col)]
 24
 25 // ---------------------------------------------------
    ---------------------------------
 26 // Struct/Class definitions
 27
 36 typedef struct AEMtx33
 37 {
 38     f32 m[3][3];
 39 }AEMtx33;
 40
 41 // ---------------------------------------------------
    ---------------------------------
```

```
 42|
 43| #ifdef __cplusplus
 44| extern "C"
 45| {
 46| #endif
 47|
 48| // ----------------------------------------
    ----------------------------------
 49|
 50| /*************************************************
    *********************************/
 62| /*************************************************
    *********************************/
 63| AE_API void AEMtx33Identity          (AEMtx33*
    pResult);
 64|
 65| /*************************************************
    *********************************/
 81| /*************************************************
    *********************************/
 82| AE_API void AEMtx33Transpose         (AEMtx33*
    pResult, AEMtx33* pMtx);
 83|
 84| /*************************************************
    *********************************/
100| /*************************************************
    *********************************/
101| AE_API f32 AEMtx33Inverse           (AEMtx33*
    pResult, AEMtx33* pMtx);
102|
103| /*************************************************
    *********************************/
119| /*************************************************
    *********************************/
120| AE_API void AEMtx33InvTranspose      (AEMtx33*
    pResult, AEMtx33* pMtx);
121|
```

```
122| /*****************************************************
   ***********************************/
142| /*****************************************************
   ***********************************/
143| AE_API void AEMtx33Concat           (AEMtx33*
   pResult, AEMtx33* pMtx0, AEMtx33* pMtx1);
144|
145| /*****************************************************
   ***********************************/
163| /*****************************************************
   ***********************************/
164| AE_API void AEMtx33Orthogonalize    (AEMtx33*
   pResult, AEMtx33* pMtx);
165|
166| /*****************************************************
   ***********************************/
178| /*****************************************************
   ***********************************/
179| AE_API f32 AEMtx33Determinant      (AEMtx33*
   pMtx);
180|
181| // ---------------------------------------------
   ---------------------------------
182|
183| /*****************************************************
   ***********************************/
205| /*****************************************************
   ***********************************/
206| AE_API void AEMtx33SetCol           (AEMtx33*
   pResult, u32 col,  AEVec2* pVec);
207|
208| /*****************************************************
   ***********************************/
230| /*****************************************************
   ***********************************/
231| AE_API void AEMtx33SetRow           (AEMtx33*
   pResult, u32 row,  AEVec2* pVec);
```

```
232|
233| /*****************************************
   ********************************/
254| /*****************************************
   ********************************/
255| AE_API void AEMtx33GetCol           (AEVec2*
   pResult, AEMtx33* pMtx, u32 col);
256|
257| /*****************************************
   ********************************/
278| /*****************************************
   ********************************/
279| AE_API void AEMtx33GetRow           (AEVec2*
   pResult, AEMtx33* pMtx, u32 row);
280|
281| // -------------------------------------
   --------------------------------
282|
283| /*****************************************
   ********************************/
303| /*****************************************
   ********************************/
304| AE_API void AEMtx33Trans            (AEMtx33*
   pResult, f32 x, f32 y);
305|
306| /*****************************************
   ********************************/
331| /*****************************************
   ********************************/
332| AE_API void AEMtx33TransApply       (AEMtx33*
   pResult, AEMtx33* pMtx, f32 x, f32 y);
333|
334| /*****************************************
   ********************************/
354| /*****************************************
   ********************************/
355| AE_API void AEMtx33Scale            (AEMtx33*
```

```
    pResult, f32 x, f32 y);
356|
357| /*************************************************
   ***********************************/
382| /*************************************************
   ***********************************/
383| AE_API void AEMtx33ScaleApply       (AEMtx33*
   pResult, AEMtx33* pMtx, f32 x, f32 y);
384|
385| /*************************************************
   ***********************************/
402| /*************************************************
   ***********************************/
403| AE_API void AEMtx33Rot              (AEMtx33*
   pResult, f32 angle);
404|
405| /*************************************************
   ***********************************/
422| /*************************************************
   ***********************************/
423| AE_API void AEMtx33RotDeg           (AEMtx33*
   pResult, f32 angle);
424|
425| // -----------------------------------------
   ---------------------------------
426|
427| /*************************************************
   ***********************************/
447| /*************************************************
   ***********************************/
448| AE_API void AEMtx33MultVec          (AEVec2*
   pResult, AEMtx33* pMtx, AEVec2* pVec);
449|
450| /*************************************************
   ***********************************/
477| /*************************************************
   ***********************************/
```

```
478 AE_API void AEMtx33MultVecArray     (AEVec2*
    pResult, AEMtx33* pMtx, AEVec2* pVec, u32
    count);
479
480
481 AE_API void AEMtx33MultVecSR        (AEVec2*
    pResult, AEMtx33* pMtx, AEVec2* pVec);
482 AE_API void AEMtx33MultVecArraySR   (AEVec2*
    pResult, AEMtx33* pMtx, AEVec2* pVec, u32
    count);
483
484 // ----------------------------------------
    ---------------------------------
485
486
487
488
489 #ifdef __cplusplus
490 }
491 #endif
492
493 #endif // MTX_H
```

Alpha Engine

AEVec2 Member List

This is the complete list of members for **AEVec2**, including all inherited members.

x AEVec2
y AEVec2

Alpha Engine

AEVec2.h

Go to the documentation of this file.

```
  1 /*****************************************************
    ***********************************/
 15 /*****************************************************
    ***********************************/
 16
 17 #ifndef AE_VEC2_H
 18 #define AE_VEC2_H
 19
 20 #include "AEExport.h"
 21 #include "AETypes.h"
 22
 23 // ---------------------------------------------
    ---------------------------------
 24
 25 //AE_API typedef struct AEVec2
 26 typedef struct AEVec2
 27 {
 28     f32 x, y;
 29 }AEVec2;
 30
 31 // ---------------------------------------------
    ---------------------------------
 32 #ifdef __cplusplus
 33 extern "C"
 34 {
 35 #endif
```

```
 36|
 37| //For testing? Function purpose unclear
 38| AE_API s32 AEVec2Test              (s32 i1,
     s32 i2);
 39|
 40| /**************************************************
     *********************************/
 52| /**************************************************
     *********************************/
 53| AE_API void AEVec2Zero             (AEVec2*
     pResult);
 54|
 55| /**************************************************
     *********************************/
 75| /**************************************************
     *********************************/
 76| AE_API void AEVec2Set              (AEVec2*
     pResult, f32 x, f32 y);
 77|
 78| /**************************************************
     *********************************/
 94| /**************************************************
     *********************************/
 95| AE_API void AEVec2Neg              (AEVec2*
     pResult, AEVec2* pVec0);
 96|
 97| /**************************************************
     *********************************/
117| /**************************************************
     *********************************/
118| AE_API void AEVec2Add              (AEVec2*
     pResult, AEVec2* pVec0, AEVec2* pVec1);
119|
120| /**************************************************
     *********************************/
140| /**************************************************
     *********************************/
```

```
141| AE_API void AEVec2Sub                (AEVec2*
   pResult, AEVec2* pVec0, AEVec2* pVec1);
142|
143| /*****************************************
   ********************************/
159| /*****************************************
   ********************************/
160| AE_API void AEVec2Normalize          (AEVec2*
   pResult, AEVec2* pVec0);
161|
162| /*****************************************
   ********************************/
182| /*****************************************
   ********************************/
183| AE_API void AEVec2Scale              (AEVec2*
   pResult, AEVec2* pVec0, f32 s);
184|
185| /*****************************************
   ********************************/
211| /*****************************************
   ********************************/
212| AE_API void AEVec2ScaleAdd           (AEVec2*
   pResult, AEVec2* pVec0, AEVec2* pVec1, f32 s);
213|
214| /*****************************************
   ********************************/
240| /*****************************************
   ********************************/
241| AE_API void AEVec2ScaleSub           (AEVec2*
   pResult, AEVec2* pVec0, AEVec2* pVec1, f32 s);
242|
243| /*****************************************
   ********************************/
263| /*****************************************
   ********************************/
264| AE_API void AEVec2Project            (AEVec2*
   pResult, AEVec2* pVec0, AEVec2* pVec1);
```

```
265|
266| /*************************************************
   *********************************/
287| /*************************************************
   *********************************/
288| AE_API void AEVec2ProjectPerp       (AEVec2*
   pResult, AEVec2* pVec0, AEVec2* pVec1);
289|
290| /*************************************************
   *********************************/
319| /*************************************************
   *********************************/
320| AE_API void AEVec2Lerp              (AEVec2*
   pResult, AEVec2* pVec0, AEVec2* pVec1, f32 t);
321|
322| /*************************************************
   *********************************/
334| /*************************************************
   *********************************/
335| AE_API f32 AEVec2Length            (AEVec2*
   pVec0);
336|
337| /*************************************************
   *********************************/
349| /*************************************************
   *********************************/
350| AE_API f32 AEVec2SquareLength      (AEVec2*
   pVec0);
351|
352| /*************************************************
   *********************************/
368| /*************************************************
   *********************************/
369| AE_API f32 AEVec2Distance          (AEVec2*
   pVec0, AEVec2* pVec1);
370|
371| /*************************************************
```

```
    ***********************************/
387 /*******************************************
    ***********************************/
388 AE_API f32 AEVec2SquareDistance    (AEVec2*
    pVec0, AEVec2* pVec1);
389
390 /*******************************************
    ***********************************/
406 /*******************************************
    ***********************************/
407 AE_API f32 AEVec2DotProduct        (AEVec2*
    pVec0, AEVec2* pVec1);
408
409 /*******************************************
    ***********************************/
426 /*******************************************
    ***********************************/
427 AE_API f32 AEVec2CrossProductMag   (AEVec2*
    pVec0, AEVec2* pVec1);
428
429 /*******************************************
    ***********************************/
445 /*******************************************
    ***********************************/
446 AE_API void AEVec2FromAngle         (AEVec2*
    pResult, f32 angle);
447
448 #ifdef __cplusplus
449 }
450 #endif
451
452 // ----------------------------------------
    ---------------------------------
453
454 #endif // VEC2_H
```

Alpha Engine

AEEngine.h

Go to the documentation of this file.

```
  1 /*****************************************************************************/
 29 /*****************************************************************************/
 30
 31 #ifndef AE_ENGINE_H
 32 #define AE_ENGINE_H
 33
 34 // ---------------------------------------------------------------------------
 35 // Includes
 36 #include <stdlib.h>
 37 #include <stdio.h>
 38 #include <string.h>
 39 #include <assert.h>
 40
 41 #include <windows.h>
 42
 43 #include "AEExport.h"
 44
 45 // ---------------------------------------------------------------------------
 46 // assert defines
 47
 48 #ifndef AE_FINAL    //User should define AE_FINAL to remove asserts
```

```
49
50 // ----------------------------------------
   ---------------------------------
51
52 #define AE_ASSERT(x)
    \
53 {
    \
54     if((x) == 0)
    \
55     {
    \
56         PRINT("AE_ASSERT: %s\nLine: %d\nFunc:
   %s\nFile: %s\n",      \
57             #x, __LINE__, __FUNCTION__,
   __FILE__);                  \
58         exit(1);
    \
59     }
    \
60 }
61
62 // ----------------------------------------
   ---------------------------------
63
64 #define AE_ASSERT_MESG(x, ...)
    \
65 {
    \
66     if((x) == 0)
    \
67     {
    \
68         PRINT("AE_ASSERT_MESG: %s\nLine:
   %d\nFunc: %s\nFile: %s\n", \
69             #x, __LINE__, __FUNCTION__,
   __FILE__);                  \
```

```
70          PRINT("Mesg: "__VA_ARGS__);
 \
71          PRINT("\n");
 \
72          exit(1);
 \
73      }
 \
74  }
75
76  // ----------------------------------------
 ---------------------------------
77
78  #define AE_ASSERT_PARM(x)
 \
79  {
 \
80      if((x) == 0)
 \
81      {
 \
82          PRINT("AE_ASSERT_PARM: %s\nLine:
 %d\nFunc: %s\nFile: %s\n", \
83              #x, __LINE__, __FUNCTION__,
 __FILE__);                  \
84          exit(1);
 \
85      }
 \
86  }
87
88  // ----------------------------------------
 ---------------------------------
89
90  #define AE_ASSERT_ALLOC(x)
 \
91  {
```

```
    \
92      if((x) == 0)
    \
93      {
    \
94          PRINT("AE_ASSERT_ALLOC: %s\nLine:
    %d\nFunc: %s\nFile: %s\n",    \
95              #x, __LINE__, __FUNCTION__,
    __FILE__);                        \
96          exit(1);
    \
97      }
    \
98  }
99
100
101 // ----------------------------------------
    --------------------------------
102
103 #define AE_WARNING(x)
    \
104 {
    \
105     if((x) == 0)
    \
106     {
    \
107         PRINT("AE_WARNING: %s\nLine:
    %d\nFunc: %s\nFile: %s\n", \
108             #x, __LINE__, __FUNCTION__,
    __FILE__);              \
109     }
    \
110 }
111
112 // ----------------------------------------
    --------------------------------
```

```
113
114 #define AE_WARNING_MESG(x, ...)
    \
115 {
    \
116     if((x) == 0)
    \
117     {
    \
118         PRINT("AE_WARNING_MESG: %s\nLine:
    %d\nFunc: %s\nFile: %s\n",    \
119             #x, __LINE__, __FUNCTION__,
    __FILE__);                          \
120         PRINT("Mesg: "__VA_ARGS__);
    \
121         PRINT("\n");
    \
122     }
    \
123 }
124
125 // ----------------------------------------
    --------------------------------
126
127 #define AE_WARNING_PARM(x)
    \
128 {
    \
129     if((x) == 0)
    \
130     {
    \
131         PRINT("AE_WARNING_PARM: %s\nLine:
    %d\nFunc: %s\nFile: %s\n",    \
132             #x, __LINE__, __FUNCTION__,
    __FILE__);                          \
133     }
```

```
    \
134 }
135
136 // ----------------------------------------
    ----------------------------------
137
138 #else // AE_FINAL
139
140 // ----------------------------------------
    ----------------------------------
141
142 #define AE_ASSERT(x)
143 #define AE_ASSERT_MESG(x, ...)
144 #define AE_ASSERT_PARM(x)
145 #define AE_ASSERT_ALLOC(x)
146
147 #define AE_WARNING(x)
148 #define AE_WARNING_MESG(x, ...)
149 #define AE_WARNING_PARM(x)
150 #define AE_WARNING_ALLOC(x)
151
152 // ----------------------------------------
    ----------------------------------
153
154 #endif // AE_FINAL
155
156 // ----------------------------------------
    ----------------------------------
157
158 #define AE_FATAL_ERROR(...)                \
159 {                                          \
160     PRINT("AE_FATAL_ERROR: "__VA_ARGS__);  \
161     exit(1);                               \
162 }
163
164 // ----------------------------------------
    ----------------------------------
```

```
165 // Alpha engine includes
166
167 #include "AETypes.h" //Typedefs
168 #include "AEMath.h" //Maths libraray
169 #include "AEUtil.h" //Utility library
170 #include "AEFrameRateController.h" //Frame
    controller
171 #include "AESystem.h" //System library
172 #include "AEGraphics.h" //Graphics library
173 #include "AEInput.h" //Input library
174 #include "AEGameStateMgr.h" //Gamestates
    manager
175
176 // ----------------------------------------
    ---------------------------------
177
178 #endif // AE_ENGINE_H
179
```

Alpha Engine

AEExport.h

Go to the documentation of this file.

```
 1 /*****************************************************
    ***********************************/
15 /*****************************************************
    ***********************************/
16
17 #ifndef AE_EXPORT_H
18 #define AE_EXPORT_H
19
20 #ifndef EPSILON
21     #define EPSILON 0.00001f
22 #endif
23
24 #ifndef PI
25     #define PI      3.1415926f
26 #endif
27
28 #define HALF_PI (PI * 0.5f)
29 #define TWO_PI  (PI * 2.0f)
30
31
32 #define EXPORT_WINDOWS                  1
33 #define EXPORT_ANDROID                  0
34
35 #if(1 == EXPORT_WINDOWS)
36
37     #ifdef AE_FINAL
```

```
38          #define PRINT(...)
39      #else
40          #define PRINT(...)
    printf(__VA_ARGS__)
41      #endif
42
43      #ifdef AE_FINAL
44          #define PRINT_INFO(...)
45      #else
46          #define PRINT_INFO(...) PRINT("File:
    %s\nLine: %d\nFunc: %s\n\n",                \
47
    __FILE__,__LINE__, __FUNCTION__);
48      #endif
49
50
51
52  //  #ifdef AE_EXPORTS
53          #define AE_API __declspec(dllexport)
54  //  #else
55  //      #define AE_API __declspec(dllimport)
56          //#define AE_API
57  //  #endif
58
59      #define
    DECLARE_FUNCTION_FOR_ANDROID(functionName)
60      #define
    IMPLEMENT_FUNCTION_FOR_ANDROID(functionName)
61
62      #define
    DECLARE_FUNCTION_FOR_ANDROID_2_INT(functionName
    , paramType1, paramType2)
63      #define
    IMPLEMENT_FUNCTION_FOR_ANDROID_2_INT(functionNa
    me, paramType1, paramType2)
64
65  #elif(1 == EXPORT_ANDROID)
```

```
66
67      #include <jni.h>
68      #include <android/log.h>
69      #define AE_API
70      #define  LOG_TAG    "AlphaEngine-Lib"
71      #define  LOGI(...)
    __android_log_print(ANDROID_LOG_INFO,LOG_TAG,__
    VA_ARGS__)
72      #define  LOGE(...)
    __android_log_print(ANDROID_LOG_ERROR,LOG_TAG,_
    _VA_ARGS__)
73
74      #ifdef AE_FINAL
75          #define PRINT(...)
76      #else
77          #define PRINT(...)  LOGI(__VA_ARGS__)
78      #endif
79
80
81
82      #define
    DECLARE_FUNCTION_FOR_ANDROID(functionName) \
83          JNIEXPORT void
    Java_com_example_alphaengine_AEGLRenderer_##fun
    ctionName(JNIEnv * env, jobject obj);
84
85      #define
    IMPLEMENT_FUNCTION_FOR_ANDROID(functionName) \
86              JNIEXPORT void
    Java_com_example_alphaengine_AEGLRenderer_##fun
    ctionName(JNIEnv * env, jobject obj)      \
87              {       \
88                  functionName();     \
89              }
90
91
92      #define
```

```
    DECLARE_FUNCTION_FOR_ANDROID_2_INT(functionName
    , paramType1, paramType2) \
 93             JNIEXPORT s32
    Java_com_example_alphaengine_AEGLRenderer_##fun
    ctionName(JNIEnv * env, jobject obj,
    paramType1, paramType2);
 94
 95     #define
    IMPLEMENT_FUNCTION_FOR_ANDROID_2_INT(functionNa
    me, paramType1, paramType2) \
 96             JNIEXPORT s32
    Java_com_example_alphaengine_AEGLRenderer_##fun
    ctionName(JNIEnv * env, jobject obj, paramType1
    param1, paramType2 param2)     \
 97             {       \
 98                 return functionName(param1,
    param2);        \
 99             }
100
101 #endif
102
103
104
105 #endif      // File
```

Alpha Engine

AEFrameRateController.h

Go to the documentation of this file.

```
/******************************************************************************/
/******************************************************************************/

#ifndef AE_FRAME_RATE_CONTROLLER_H
#define AE_FRAME_RATE_CONTROLLER_H

// ---------------------------------------------------------------------------
// Externs

// ---------------------------------------------------------------------------
// Function prototypes
// ---------------------------------------------------------------------------

#ifdef __cplusplus

extern "C"
{
#endif

// ---------------------------------------------------------------------------
```

```
34
35 // Frame management
36
37 /*****************************************
   ********************************/
51 /*****************************************
   ********************************/
52 AE_API void AEFrameRateControllerInit   (u32
   FrameRateMax);
53
54 /*****************************************
   ********************************/
65 /*****************************************
   ********************************/
66 AE_API void AEFrameRateControllerReset  ();
67
68 /*****************************************
   ********************************/
80 /*****************************************
   ********************************/
81 AE_API void AEFrameRateControllerStart  ();
82
83 /*****************************************
   ********************************/
95 /*****************************************
   ********************************/
96 AE_API void AEFrameRateControllerEnd    ();
97
98 /*****************************************
   ********************************/
107 /*****************************************
   ********************************/
108 AE_API f64
   AEFrameRateControllerGetFrameTime();
109
110 /*****************************************
   ********************************/
```

```
119 /*****************************************************
    ***********************************/
120 AE_API u32
    AEFrameRateControllerGetFrameCount();
121
122 // ------------------------------------------
    ---------------------------------
123
124 #ifdef __cplusplus
125 }
126 #endif
127
128 // ------------------------------------------
    ---------------------------------
129
130 #endif // AE_FRAME_RATE_CONTROLLER_H
131
```

Alpha Engine

AEGameStateMgr.h

Go to the documentation of this file.

```
/*****************************************************************/
/*****************************************************************/

#ifndef AE_GAME_STATE_MGR_H
#define AE_GAME_STATE_MGR_H

// ---------------------------------------------------------------
// defines and enums

#define AE_GS_RESTART   0xFFFFFFFE
#define AE_GS_QUIT      0xFFFFFFFF

// ---------------------------------------------------------------
// externs

extern AE_API u32 gAEGameStateInit;
extern AE_API u32 gAEGameStateCurr;
extern AE_API u32 gAEGameStatePrev;
extern AE_API u32 gAEGameStateNext;

// ---------------------------------------------------------------
```

```
35
36 extern AE_API void (*AEGameStateLoad)(void);
37 extern AE_API void (*AEGameStateInit)(void);
38 extern AE_API void (*AEGameStateUpdate)
   (void);
39 extern AE_API void (*AEGameStateDraw)(void);
40 extern AE_API void (*AEGameStateFree)(void);
41 extern AE_API void (*AEGameStateUnload)
   (void);
42
43 // ------------------------------------------
   ----------------------------------
44 // Function prototypes
45 // ------------------------------------------
   ----------------------------------
46
47 #ifdef __cplusplus
48
49 extern "C"
50 {
51 #endif
52 // ------------------------------------------
   ----------------------------------
53
54 /*****************************************
   *********************************/
93 /*****************************************
   *********************************/
94 AE_API void AEGameStateMgrAdd(u32
   gameStateIdx,
95                               void (*pLoad)
   (),
96                               void (*pInit)
   (),
97                               void (*pUpdate)
   (),
98                               void (*pDraw)
```

```
    (),
 99                                  void (*pFree)
    (),
100                                  void (*pUnload)
    ());
101
102 /*************************************************
    *********************************/
120 /*************************************************
    *********************************/
121 AE_API void AEGameStateMgrInit(u32
    gameStateInit);
122
123 /*************************************************
    *********************************/
135 /*************************************************
    *********************************/
136 AE_API void AEGameStateMgrUpdate();
137
138 // ---------------------------------------------
    ---------------------------------
139
140 #ifdef __cplusplus
141 }
142 #endif
143
144 // ---------------------------------------------
    ---------------------------------
145
146 #endif // AE_GAME_STATE_MGR_H
```

Alpha Engine

AEInput.h

Go to the documentation of this file.

```
 1 /*****************************************************************************/
15 /*****************************************************************************/
16
17 #ifndef AE_INPUT_H
18 #define AE_INPUT_H
19
20 // ---------------------------------------------------------------------------
21 // Defines/Enums
22
23 // These defines don't match the direct input defines and you should be using these
24 // defines instead of the ones in "dinput.h"
25 // For more key codes go to http://msdn.microsoft.com/en-us/library/windows/desktop/dd375731(v=vs.85).aspx
26
27 #define AEVK_LBUTTON          VK_LBUTTON
28 #define AEVK_RBUTTON          VK_RBUTTON
29 #define AEVK_MBUTTON          VK_MBUTTON
30
31 #define AEVK_BACK            VK_BACK
32 #define AEVK_TAB             VK_TAB
```

```
33 #define AEVK_RETURN          VK_RETURN
34 #define AEVK_LSHIFT          VK_LSHIFT
35 #define AEVK_RSHIFT          VK_RSHIFT
36 #define AEVK_LCTRL           VK_LCONTROL
37 #define AEVK_RCTRL           VK_RCONTROL
38 #define AEVK_LALT            VK_LMENU
39 #define AEVK_RALT            VK_RMENU
40 #define AEVK_PRINTSCREEN     VK_SNAPSHOT
41 #define AEVK_SCROLLLOCK      VK_SCROLL
42 #define AEVK_PAUSE           VK_PAUSE
43 #define AEVK_CAPSLOCK        VK_CAPITAL
44
45 #define AEVK_ESCAPE          VK_ESCAPE
46
47 #define AEVK_SPACE           VK_SPACE
48 #define AEVK_PAGEUP          VK_PRIOR
49 #define AEVK_PAGEDOWN        VK_NEXT
50 #define AEVK_END             VK_END
51 #define AEVK_HOME            VK_HOME
52 #define AEVK_LEFT            VK_LEFT
53 #define AEVK_UP              VK_UP
54 #define AEVK_RIGHT           VK_RIGHT
55 #define AEVK_DOWN            VK_DOWN
56 #define AEVK_INSERT          VK_INSERT
57 #define AEVK_DELETE          VK_DELETE
58
59 #define AEVK_0               0x30
60 #define AEVK_1               0x31
61 #define AEVK_2               0x32
62 #define AEVK_3               0x33
63 #define AEVK_4               0x34
64 #define AEVK_5               0x35
65 #define AEVK_6               0x36
66 #define AEVK_7               0x37
67 #define AEVK_8               0x38
68 #define AEVK_9               0x39
69
```

```
#define AEVK_A                  0x41
#define AEVK_B                  0x42
#define AEVK_C                  0x43
#define AEVK_D                  0x44
#define AEVK_E                  0x45
#define AEVK_F                  0x46
#define AEVK_G                  0x47
#define AEVK_H                  0x48
#define AEVK_I                  0x49
#define AEVK_J                  0x4A
#define AEVK_K                  0x4B
#define AEVK_L                  0x4C
#define AEVK_M                  0x4D
#define AEVK_N                  0x4E
#define AEVK_O                  0x4F
#define AEVK_P                  0x50
#define AEVK_Q                  0x51
#define AEVK_R                  0x52
#define AEVK_S                  0x53
#define AEVK_T                  0x54
#define AEVK_U                  0x55
#define AEVK_V                  0x56
#define AEVK_W                  0x57
#define AEVK_X                  0x58
#define AEVK_Y                  0x59
#define AEVK_Z                  0x5A

#define AEVK_NUMPAD0            VK_NUMPAD0
#define AEVK_NUMPAD1            VK_NUMPAD1
#define AEVK_NUMPAD2            VK_NUMPAD2
#define AEVK_NUMPAD3            VK_NUMPAD3
#define AEVK_NUMPAD4            VK_NUMPAD4
#define AEVK_NUMPAD5            VK_NUMPAD5
#define AEVK_NUMPAD6            VK_NUMPAD6
#define AEVK_NUMPAD7            VK_NUMPAD7
#define AEVK_NUMPAD8            VK_NUMPAD8
#define AEVK_NUMPAD9            VK_NUMPAD9
```

```
107
108 #define AEVK_NUM_MULTIPLY   VK_MULTIPLY
109 #define AEVK_NUM_PLUS       VK_ADD
110 #define AEVK_NUM_MINUS      VK_SUBTRACT
111 #define AEVK_NUM_PERIOD     VK_DECIMAL
112 #define AEVK_NUM_DIVIDE     VK_DIVIDE
113 #define AEVK_NUMLOCK        VK_NUMLOCK
114
115 #define AEVK_F1             VK_F1
116 #define AEVK_F2             VK_F2
117 #define AEVK_F3             VK_F3
118 #define AEVK_F4             VK_F4
119 #define AEVK_F5             VK_F5
120 #define AEVK_F6             VK_F6
121 #define AEVK_F7             VK_F7
122 #define AEVK_F8             VK_F8
123 #define AEVK_F9             VK_F9
124 #define AEVK_F10            VK_F10
125 #define AEVK_F11            VK_F11
126 #define AEVK_F12            VK_F12
127
128 #define AEVK_SEMICOLON      VK_OEM_1
129 #define AEVK_SLASH          VK_OEM_2
130 #define AEVK_BACKQUOTE      VK_OEM_3
131 #define AEVK_LBRACKET       VK_OEM_4
132 #define AEVK_BACKSLASH      VK_OEM_5
133 #define AEVK_RBRACKET       VK_OEM_6
134 #define AEVK_QUOTE          VK_OEM_7
135
136 #define AEVK_EQUAL          VK_OEM_PLUS
137 #define AEVK_MINUS          VK_OEM_MINUS
138 #define AEVK_PERIOD         VK_OEM_PERIOD
139 #define AEVK_COMMA          VK_OEM_COMMA
140
141 // ----------------------------------------
    ---------------------------------
142 // Function prototypes
```

```
143
144 #ifdef __cplusplus
145
146 extern "C"
147 {
148 #endif
149
150 // ----------------------------------------
    ---------------------------------
151
152 /*****************************************
    ***********************************/
164 /*****************************************
    ***********************************/
165 AE_API s32 AEInputInit();
166
167 /*****************************************
    ***********************************/
178 /*****************************************
    ***********************************/
179 AE_API void AEInputReset();
180
181 /*****************************************
    ***********************************/
192 /*****************************************
    ***********************************/
193 AE_API void AEInputUpdate();
194
195 /*****************************************
    ***********************************/
206 /*****************************************
    ***********************************/
207 AE_API void AEInputExit();
208
209 /*****************************************
    ***********************************/
226 /*****************************************
```

```
    ***********************************/
227 AE_API u8 AEInputCheckCurr      (u8 key);
228
229 /*******************************************
    ***********************************/
246 /*******************************************
    ***********************************/
247 AE_API u8 AEInputCheckPrev      (u8 key);
248
249 /*******************************************
    ***********************************/
266 /*******************************************
    ***********************************/
267 AE_API u8 AEInputCheckTriggered (u8 key);
268
269 /*******************************************
    ***********************************/
286 /*******************************************
    ***********************************/
287 AE_API u8 AEInputCheckReleased  (u8 key);
288
289 /*******************************************
    ***********************************/
306 /*******************************************
    ***********************************/
307 AE_API void AEInputGetCursorPosition(s32 *pX,
    s32 *pY);
308
309 /*******************************************
    ***********************************/
326 /*******************************************
    ***********************************/
327 AE_API void AEInputGetCursorPositionDelta(s32
    *pDeltaX, s32 *pDeltaY);
328
329 /*******************************************
    ***********************************/
```

```
344 /*****************************************************
    *********************************/
345 AE_API void AEInputShowCursor(s32 Show);
346
347 // ------------------------------------------
    ----------------------------------
348
349 #ifdef __cplusplus
350 }
351 #endif
352
353 // ------------------------------------------
    ----------------------------------
354
355 #endif // AE_INPUT_H
356
```

Alpha Engine

AEMath.h

Go to the documentation of this file.

```
  1 /*****************************************************
    ************************************/
 15 /*****************************************************
    ************************************/
 16
 17 #ifndef AE_MATH_H
 18 #define AE_MATH_H
 19
 20 // ---------------------------------------------
    ---------------------------------
 21
 22 #include "AEVec2.h" //Vector 2D
 23 #include "AEMtx33.h" //Matrix 3x3
 24 #include "AELineSegment2.h" //LineSegment 2D
 25
 26 #include <float.h>
 27
 28 #include "math.h"
 29
 30 // ---------------------------------------------
    ---------------------------------
 31 #ifdef __cplusplus
 32
 33 extern "C"
 34 {
 35 #endif
```

```
 36
 37 //AE_API void MathInit();
 38
 39 // ----------------------------------------
    ----------------------------------
 40
 41 /*****************************************
    ********************************/
 53 /*****************************************
    ********************************/
 54 AE_API f32 AEDegToRad(f32 x);
 55
 56 /*****************************************
    ********************************/
 68 /*****************************************
    ********************************/
 69 AE_API f32 AERadToDeg(f32 x);
 70
 71 /*****************************************
    ********************************/
 83 /*****************************************
    ********************************/
 84 AE_API f32 AESin    (f32 x);
 85
 86 /*****************************************
    ********************************/
 98 /*****************************************
    ********************************/
 99 AE_API f32 AECos    (f32 x);
100
101 /*****************************************
    ********************************/
113 /*****************************************
    ********************************/
114 AE_API f32 AETan    (f32 x);
115
116 /*****************************************
```

```
    ***********************************/
128 /*******************************************
    ***********************************/
129 AE_API f32 AEASin   (f32 x);
130
131 /*******************************************
    ***********************************/
143 /*******************************************
    ***********************************/
144 AE_API f32 AEACos   (f32 x);
145
146 /*******************************************
    ***********************************/
158 /*******************************************
    ***********************************/
159 AE_API f32 AEATan   (f32 x);
160
161 // ---------------------------------------
    ---------------------------------
162
164     #define AESinDeg(x)
    AESin(AEDegToRad(x))
166     #define AECosDeg(x)
    AECos(AEDegToRad(x))
167     #define AETanDeg(x)
    AETan(AEDegToRad(x))
168     #define AEASinDeg(x)
    AERadToDeg(AEASin(x))
169     #define AEACosDeg(x)
    AERadToDeg(AEACos(x))
170     #define AEATanDeg(x)
    AERadToDeg(AEATan(x))
171
172
173 // ---------------------------------------
    ---------------------------------
174
```

```
175 /*****************************************
    ********************************/
197 /*****************************************
    ********************************/
198 AE_API u32 AEIsPowOf2  (u32 x);
199
200 /*****************************************
    ********************************/
221 /*****************************************
    ********************************/
222 AE_API u32 AENextPowOf2(u32 x);
223
224 /*****************************************
    ********************************/
238 /*****************************************
    ********************************/
239 AE_API u32 AELogBase2  (u32 x);
240
241 /*****************************************
    ********************************/
263 /*****************************************
    ********************************/
264 AE_API f32 AEClamp     (f32 X, f32 Min, f32
    Max);
265
266 /*****************************************
    ********************************/
290 /*****************************************
    ********************************/
291 AE_API f32 AEWrap      (f32 x, f32 x0, f32
    x1);
292
293 /*****************************************
    ********************************/
311 /*****************************************
    ********************************/
312 AE_API f32 AEMin       (f32 x, f32 y);
```

```
313
314
315 /*****************************************************
    ********************************/
333 /*****************************************************
    ********************************/
334 AE_API f32 AEMax       (f32 x, f32 y);
335
336 /*****************************************************
    ********************************/
359 /*****************************************************
    ********************************/
360 AE_API s32 AEInRange   (f32 x, f32 x0, f32
    x1);
361
362 // ---------------------------------------------
    --------------------------------
363
364 /*****************************************************
    ********************************/
394 /*****************************************************
    ********************************/
395 AE_API f32 AECalcDistPointToCircle
    (AEVec2* pPos, AEVec2* pCtr, f32 radius);
396
397 /*****************************************************
    ********************************/
438 /*****************************************************
    ********************************/
439 AE_API f32 AECalcDistPointToRect
    (AEVec2* pPos, AEVec2* pRect, f32 sizeX, f32
    sizeY);
440
441 /*****************************************************
    ********************************/
469 /*****************************************************
    ********************************/
```

```
470| AE_API f32 AECalcDistPointToLineSeg
   (AEVec2* pPos, AEVec2* pLine0, AEVec2* pLine1);
471|
472| /*************************************************
   *********************************/
504| /*************************************************
   *********************************/
505| AE_API f32 AECalcDistPointToConvexPoly
   (AEVec2* pPos, AEVec2* pVtx, u32 vtxNum);
506|
507| /*************************************************
   *********************************/
540| /*************************************************
   *********************************/
541| AE_API f32 AECalcDistCircleToCircle
   (AEVec2* pCtr0, f32 radius0, AEVec2* pCtr1, f32
   radius1);
542|
543| /*************************************************
   *********************************/
587| /*************************************************
   *********************************/
588| AE_API f32 AECalcDistCircleToRect
   (AEVec2* pCtr, f32 radius, AEVec2* pRect, f32
   sizeX, f32 sizeY);
589|
590| /*************************************************
   *********************************/
641| /*************************************************
   *********************************/
642| AE_API f32 AECalcDistRectToRect
   (AEVec2* pRect0, f32 sizeX0, f32 sizeY0,
   AEVec2* pRect1, f32 sizeX1, f32 sizeY1, AEVec2*
   pNormal);
643|
644| /* Functions not defined
645| AE_API f32 AECalcDistCircleToLineSeg
```

```
   (AEVec2* pPos, AEVec2* pLine0, AEVec2* pLine1);
646| AE_API f32 AECalcDistCircleToConvexPoly
   (AEVec2* pPos, AEVec2* pVtx, u32 vtxNum);
647| */
648|
649| // ----------------------------------------
   --------------------------------
650|
651| /**************************************************
   ********************************/
678| /**************************************************
   ********************************/
679| AE_API s32 AETestPointToCircle
   (AEVec2* pPos, AEVec2* pCtr, f32 radius);
680|
681| /**************************************************
   ********************************/
716| /**************************************************
   ********************************/
717| AE_API s32 AETestPointToRect
   (AEVec2* pPos, AEVec2* pRect, f32 sizeX, f32
   sizeY);
718|
719| /**************************************************
   ********************************/
750| /**************************************************
   ********************************/
751| AE_API s32 AETestCircleToCircle
   (AEVec2* pCtr0, f32 radius0, AEVec2* pCtr1, f32
   radius1);
752|
753| /**************************************************
   ********************************/
794| /**************************************************
   ********************************/
795| AE_API s32 AETestCircleToRect
   (AEVec2* pCtr, f32 radius, AEVec2* pRect, f32
```

```
    sizeX, f32 sizeY);
796|
797| /*****************************************
   ********************************/
841| /*****************************************
   ********************************/
842| AE_API s32 AETestRectToRect
   (AEVec2* pRect0, f32 sizeX0, f32 sizeY0,
   AEVec2* pRect1, f32 sizeX1, f32 sizeY1);
843|
844| // -------------------------------------
   ---------------------------------
845|
846| /*****************************************
   ********************************/
870| /*****************************************
   ********************************/
871| AE_API f32
   AEStaticPointToStaticLineSegment(AEVec2 *pPos,
   AELineSegment2 *pLine);
872|
873| /*****************************************
   ********************************/
906| /*****************************************
   ********************************/
907| AE_API f32
   AEAnimatedPointToStaticLineSegment(AEVec2
   *pStart, AEVec2 *pEnd, AELineSegment2 *pLine,
   AEVec2 *pInter);
908|
909| /*****************************************
   ********************************/
948| /*****************************************
   ********************************/
949| AE_API f32
   AEAnimatedCircleToStaticLineSegment(AEVec2
   *pStart, AEVec2 *pEnd, f32 radius,
```

```
    AELineSegment2 *pLine, AEVec2 *pInter);
 950|
 951| /*****************************************
    ********************************/
 989| /*****************************************
    ********************************/
 990| AE_API f32
    AEReflectAnimatedPointOnStaticLineSegment(AEVec
    2 *pStart, AEVec2 *pEnd, AELineSegment2 *pLine,
    AEVec2 *pInter, AEVec2 *pReflect);
 991|
 992| /*****************************************
    ********************************/
1037| /*****************************************
    ********************************/
1038| AE_API f32
    AEReflectAnimatedCircleOnStaticLineSegment(AEVe
    c2 *pStart, AEVec2 *pEnd, f32 radius,
    AELineSegment2 *pLine, AEVec2 *pInter, AEVec2
    *pReflect);
1039|
1040| /*****************************************
    ********************************/
1078| /*****************************************
    ********************************/
1079| AE_API f32
    AEAnimatedPointToStaticCircle(AEVec2 *pStart,
    AEVec2 *pEnd, AEVec2 *pCtr, f32 radius, AEVec2
    *pInter);
1080|
1081| /*****************************************
    ********************************/
1124| /*****************************************
    ********************************/
1125| AE_API f32
    AEReflectAnimatedPointOnStaticCircle(AEVec2
    *pStart, AEVec2 *pEnd, AEVec2 *pCtr, f32
```

```
    radius, AEVec2 *pInter, AEVec2 *pReflect);
1126|
1127| /*****************************************
    **********************************/
1169| /*****************************************
    **********************************/
1170| AE_API f32
    AEAnimatedCircleToStaticCircle(AEVec2 *pCtr0s,
    AEVec2 *pCtr0e, f32 radius0, AEVec2 *pCtr1, f32
    radius1, AEVec2 *pInter);
1171|
1172| /*****************************************
    **********************************/
1220| /*****************************************
    **********************************/
1221| AE_API f32
    AEReflectAnimatedCircleOnStaticCircle(AEVec2
    *pCtr0s, AEVec2 *pCtr0e, f32 radius0, AEVec2
    *pCtr1, f32 radius1, AEVec2 *pInter, AEVec2
    *pReflect);
1222|
1223|
1224|
1225| /*
1226| // sweep a circle with radius 'radius' from
    ctr0 -> ctr1 againts a point
1227| // * return -ve if circle does not touch the
    point at any time
1228| AE_API f32 AESweepCircleToPoint (AEVec2*
    pCtr0, AEVec2* pCtr1, f32 radius, AEVec2* pP);
1229|
1230| // sweep a circle with radius 'radius' from
    ctr0 -> ctr1 againts a line segment
1231| // * return -ve if circle does not intersect
    the line segment at any time
1232| AE_API f32 AESweepCircleToLineSeg   (AEVec2*
    pCtr0, AEVec2* pCtr1, f32 radius, AEVec2* pP0,
```

```
    AEVec2* pP1, AEVec2* pN);
1233|
1234|
1235| //TO TEST
1236| //Sweeps a moving point against a static line
1237| AE_API f32 AESweepPointToLine       (AEVec2
    *pPos, AEVec2 *pVel, AEVec2 *pPnt, AEVec2
    *pDirection);
1238|
1239| //TO TEST
1240| //Sweeps a moving circle against a static
    line
1241| AE_API f32 AESweepCircleToLine      (AEVec2
    *pCtr, f32 radius, AEVec2 *pVel, AEVec2 *pPnt,
    AEVec2 *pDirection);
1242|
1243| //TO TEST
1244| //Reflects a moving point on a static line.
    Returns 0 if there is no
1245| //collision between the point and the line.
1246| AE_API s32 AEReflectPointOnLine (AEVec2
    *pPos, AEVec2 *pVel, AEVec2 *pPnt, AEVec2
    *pDirection, AEVec2 *pNewPosition, AEVec2
    *pNewVelocity);
1247|
1248|
1249| //TO TEST
1250| //Reflects a moving circle on a static line.
    Returns 0 if there is no
1251| //collision between the circle and the line.
1252| AE_API s32 AEReflectCircleOnLine    (AEVec2
    *pCtr, f32 radius, AEVec2 *pVel, AEVec2 *pPnt,
    AEVec2 *pDirection, AEVec2 *pNewPosition,
    AEVec2 *newVelocity);
1253| */
1254| // ----------------------------------------
    -------------------------------
```

```
1255
1256 #ifdef __cplusplus
1257 }
1258 #endif
1259
1260 // ----------------------------------------
    ---------------------------------
1261
1262 #endif // AE_MATH_H
```

Alpha Engine

AESystem.h

Go to the documentation of this file.

```
  1 /*****************************************************
    ************************************/
 15 /*****************************************************
    ************************************/
 16
 17 #ifndef AE_SYSTEM_H
 18 #define AE_SYSTEM_H
 19
 20 // -----------------------------------------
    ---------------------------------
 21
 22 // window related variables
 23 extern HINSTANCE    ghAESysAppInstance;
 24 extern HWND         gAESysWindowHandle;
 25 extern WNDCLASS     winClass;
 26
 27 extern const s8*    gpAESysWinTitle;
 28 extern const s8*    gpAESysWinClassName;
 29
 30 extern AE_API s32 gAESysAppActive;
 31
 32 // -----------------------------------------
    ---------------------------------
 33 #ifdef __cplusplus
 34 extern "C"
 35 {
```

```
36 #endif
37
38 /*****************************************
   ********************************/
80 /*****************************************
   ********************************/
81 AE_API s32 AESysInit                (HINSTANCE
   hAppInstance, s32 show, s32 WinWidth, s32
   WinHeight, s32 CreateConsole, u32 FrameRateMax,
   LRESULT (CALLBACK *pWinCallBack)(HWND hWnd,
   UINT msg, WPARAM wp, LPARAM lp));
82
83 /*****************************************
   ********************************/
95 /*****************************************
   ********************************/
96 AE_API void AESysReset               ();
97
98 // Already called in AESysFrameEnd
99 //AE_API void AESysUpdate                ();
100
101 /*****************************************
   ********************************/
113 /*****************************************
   ********************************/
114 AE_API void AESysExit                ();
115
116 /*****************************************
   ********************************/
128 /*****************************************
   ********************************/
129 AE_API void AESysFrameStart();
130
131 /*****************************************
   ********************************/
143 /*****************************************
   ********************************/
```

```
144 AE_API void AESysFrameEnd();
145
146 /*****************************************
    ********************************/
155 /*****************************************
    ********************************/
156 AE_API HWND AESysGetWindowHandle    ();
157
158 //AE_API s32* AESysGetAppActive     ();
159
160 /*****************************************
    ********************************/
173 /*****************************************
    ********************************/
174 AE_API void AESysSetWindowTitle     (const s8
    *pTitle);
175
176 /*****************************************
    ********************************/
186 /*****************************************
    ********************************/
187 AE_API s32 AESysDoesWindowExist     ();
188
189 // ---------------------------------------
    ----------------------------------
190
191
192
193 // ---------------------------------------
    ----------------------------------
194
195
196 #ifdef __cplusplus
197 }
198 #endif
199
200 #endif // AE_SYSTEM_H
```

```
201
```

Alpha Engine

AETypes.h

Go to the documentation of this file.

```
  1 /*****************************************************
    ***********************************/
 15 /*****************************************************
    ***********************************/
 16
 17 #ifndef AE_TYPES_H
 18 #define AE_TYPES_H
 19
 20 // -----------------------------------------
    ---------------------------------
 21 // typdef types
 22
 23 typedef char s8;
 24 typedef unsigned char u8;
 25 typedef signed short s16;
 26 typedef unsigned short u16;
 27 typedef signed int s32;
 28 typedef unsigned int u32;
 29 typedef signed long long s64;
 30 typedef unsigned long long u64;
 31 typedef float f32;
 32 typedef double f64;
 33
 34 // -----------------------------------------
    ---------------------------------
 35
```

```
36 #endif //AE_TYPES_H
```

Alpha Engine

AEUtil.h

Go to the documentation of this file.

```
  1 /************************************************
    ********************************/
 15 /************************************************
    ********************************/
 16 #ifndef AE_UTIL_H
 17 #define AE_UTIL_H
 18
 19 // ---------------------------------------------
    ---------------------------------
 20 #ifdef __cplusplus
 21 extern "C"
 22 {
 23 #endif
 24
 25 /************************************************
    ********************************/
 38 /************************************************
    ********************************/
 39 AE_API f64 AEGetTime(f64* pTime);
 40
 41 /************************************************
    ********************************/
 50 /************************************************
    ********************************/
 51 AE_API f32 AERandFloat();
 52
```

```
53
54 #define isZero(x) ((x < EPSILON) && (x > -
   EPSILON))
55
56 #define isEqual(x,y) (((x >= y) ? (x-y) : (y-
   x)) < EPSILON)
57
58
59
60
61 #ifdef __cplusplus
62 }
63 #endif
64
65 /*****************************************
   ********************************/
82 /*****************************************
   ********************************/
83 s8* ReadFromFile(const s8 *fileName);
84
85 // -------------------------------------
   ---------------------------------
86
87 #endif // AE_UTIL_H
88
89
```

Alpha Engine

include Directory Reference

Files

file **AEEngine.h** [code]
Header file for the Alpha Engine.

file **AEExport.h** [code]
Header file for the library settings.

file **AEFrameRateController.h** [code]
Header file for the frame rate controller.

file **AEGameStateMgr.h** [code]
Header file for the game state manager.

file **AEGraphics.h** [code]
Header file for the graphics library.

file **AEInput.h** [code]
Header file for the input library.

file **AELineSegment2.h** [code]
Header file for the 2D line segment library.

file **AEMath.h** [code]
Header file for the math library.

file **AEMtx33.h** [code]
Header file for the 3x3 matrix library.

file **AESystem.h** [code]
Header file for the system library.

file **AETypes.h** [code]
Header file for the typedefs.

file **AEUtil.h** [code]
Header file for the utility library.

file **AEVec2.h** [code]
Header file for the 2D vector library.