Last revised March 10, 2013

What Is XashXT?

XashXT is an extension mod that brings lots of new functionality for mod authors working with Xash3D. Xash3D is an independent game engine that is fully compatible with GoldSource (Half-Life) and has a number of advantages over it. In particular, these are extended interfaces that make it possible to implement realistic physics and graphics. XashXT, also known as the *Xash mod*, is an add-on that showcases the engine's capabilities in action. Despite its primary purpose (engine tech demo), the Xash mod is a stable, thoroughly debugged product with tons of exciting possibilities, which tear down the constraints that mappers routinely face.

The Xash mod gets updates and new features on a regular basis. Bugs are fixed shortly after they are found. This means that no long-standing bugs will be carried over from one version to another while the developer is busy adding new functionality—which incidentally many modern products are notorious for. Also worth mentioning is that the development kit is fully compatible with the original Half-Life (the same map, model and sprite editors are supported, the same entities are provided) and partly compatible with the Spirit of Half-Life mod (some global entity options and entities are the same).

Who This Help Is For

This documentation assumes an audience who know the terms related to working with the GoldSource engine, are familiar with the Xash3D engine and have some experience in making levels for these engines. If you have not yet reached that stage, then before consulting this help, please familiarise yourself with any of the Half-Life mapping how-tos available on the appropriate forums. Otherwise, reading this documentation will be pointless.

- Full compatibility with the original Half-Life entity set
- Support for forming complex hierarchies of entities (Parent System)
- Mirrors that can skip reflection of any specified object or, conversely, show ordinarily invisible objects
- Surveillance monitors (colour and monochrome)
- Portals that can take you from one point to another (as in Portal, but more basic)
- Weather effects (rain and snow)
- A powerful particle system with a standalone visual editor
- A system of prefixes for making complex scripted scenes
- An extended decal projection system that works on both level geometry and characters (players and NPCs)
- Realistic lasers that can be reflected off mirror surfaces (used in gameplay)
- A large set of logical entities for implementing any conditions in scripted scenes or complex arrangements
- A camera with flexible settings for cutscenes, letting you switch to any character's point of view (including inanimate ones)
- Configuration options for setting the attitude of any NPC towards others and towards the player
- Dynamic coloured lighting that can project and play an animated texture as a light source
- Playback of AVI files on level polygons and brush objects
- A 3D skybox for simulating expansive areas
- Bloom effects
- Supports for HD textures (up to 4096x4096)
- Background maps in the game menu (as in Source)
- Support for large game levels (32768x32768x32768 units)
- Shadow maps for dynamic shadows
- More believable lighting
- Realistic rigid body physics powered by a third-party engine (PhysX)
- Realistic water
- Dynamic day and night cycles

Before setting up the mod, make sure you have Xash3D build 1905 or higher installed, and the **valve** game folder exists. Technically, Xash3D has no dependency on the **valve** folder (you can copy the necessary resources to the **xash** folder instead), but such decisions are up to mod authors. You will also need the Valve Hammer Editor or QuArK to build game levels. Editor setup is covered by the documentation for particular editors, so it will not be described here. An up-to-date entity definition file for Valve Hammer Editor is located in the **xash\devkit** folder.

The following are mandatory resources required for correct operation of the toolkit:

- sprites\640_train.spr
- sprites\decimal.spr
- sprites\null.spr
- sprites\raindrop.spr
- sprites\snowflake.spr
- sprites\waterring.spr
- cl dlls\client.dll
- cl_dlls\server.dll
- delta.lst

All other resources are required only for demo maps to work correctly and do not affect the program code in any way. This is an important point for users who want to make their own game based on the Xash mod. In addition, make sure the Nvidia PhysX driver is installed. Without this driver, all physics objects will just hang there.

The **devkit** folder contains, in addition to the entity definition files for map editors:

- An improved and extended studio model compiler: bugs of the original Valve version have been fixed, and new features have been added
- A custom build of the VHLT map compiler by vluzacn (differences from the original are minimal: QuArK's console output has been corrected and

support for the XashXT's **func_light** entity has been added)

- A visual editor for the Aurora particle system
- The complete source code for **server.dll**, **client.dll** and **xash.exe**

Running the Mod

You can run the mod using the following methods:

- From the command line, for example: hl.exe -game xash
- Using the standalone **xash.exe** launcher

Global systems are entity parameters that can be applied either to all objects without exception or to the majority of them, except the ones where applying them is pointless (for example, drawing options for entities without a visible model).

Parent System Reflection Types
Prefixes and Postfixes for Switching Target Object States
States and Masters
Mirrors
Water
Environment Variables
Custom Sounds
Monster Classes and Behaviour

The parent system in XashXT, similarly to its counterpart in Half-Life 2, lets you attach an arbitrary number of objects to any object. The focus is on kinematic objects such as doors, buttons, trains, lifts and so on. Unique to XashXT is the ability of a hierarchy to get from one game level to another without losing functionality or coming apart (which happens in Half-Life 2). You can also force object attachment on the client side (disregarding physics). The latter feature lets mappers use NPCs' attachments to directly link point-based objects: sprites, lights, laser beams and in certain cases models.

The system also supports detaching a group of objects (while preserving the hierarchy of course) and reattaching it to another arbitrary object. The parenting is done by using the special **parent** field where the level designer specifies the targetname of the parent. If the name is followed by a dot followed by a number, this switches to the client parent system, and the number is treated as the attachment number on the parent model.

The client parent system works only for studio models. This means that only models of this type can act as parents, but brush and sprite models cannot. To make porting of Spirit of Half-Life maps easier, the **parent** field has the alias **movewith**, which works identically.

- Not all objects can be linked correctly using this system. For example, linking NPCs with it is discouraged, because this may cause interpolation issues and will eventually fail to work as expected by the level designer.
- Attaching the player is also disallowed, but you can attach objects to the player through the **trigger_changeparent** entity by specifying one of the following in the **m_iszNewParent** field:
 - The *locus keyword, provided that trigger_changeparent was
 activated using a trigger_once, trigger_multiple or trigger_inout
 touched by the player (who thereby activated it)
 - The ***player** keyword
- The *player keyword should not be entered directly in the parent field, because the player joins the server last, after all the other objects have been interlinked.
- Static lights (precalculated at compile time) has no effect when attached somewhere. However, you can use dynamic lights, which will move along with your objects.
- Some objects that have an effect only on the client will not work correctly either. Such are **env_bubbles**, **env_rain**, **env_static**, **env_funnel** and possibly **func_mortar_field**.

For any visible entity, you can configure the way it is reflected in mirrors. Use the **reflection** parameter, which is found in all objects. Three values are possible:

- 0 default; the entity is always visible
- 1 the entity is not reflected in mirrors; if there are no mirrors in the level, the effect of this setting is wasted
- 2 the entity is visible only in mirrors but, unless it is made intangible through its options, it remains solid

Note that this parameter has no effect on monitors and portals, just mirrors.

- Obviously, the parameter does not affect entities that have no visible model, such as logic objects.
- The parameter does not affect the player and objects that act only on the client, such as **env_bubbles**, **env_rain**, **env_static**, **env_funnel**, **env_particle**, **env_projector** and **env_dynlight**.
- In addition, the parameter may be ignored by entity emitters, such as **gibshooter** and **env_warpball**.

An early implementation of this system was available in the Spirit of Half-Life mod. The purpose is to introduce a way to explicitly define the behaviour of a controllable target. This means that objects with two or more possible states can be explicitly forced to switch to any of these states. The classic example involves a door: **func_door**. If a door is in "toggle" mode, then subsequent activations open and close it intermittently, but the mapper has no way to activate the door so that it opens every time and, by extension, to ignore the open request if the door is already open.

The prefix system addresses this precise issue and gives the mapper ample possibilities. In addition, it makes scripts more stable and less dependent on the actions of the player, who might block objects in such a way as to break the predefined script behaviour. The object activation field (usually named **target**) can contain the following prefixes:

(what

+	Activate target, turn on
-	Deactivate target, turn off
<	Set a numeric parameter for the object further on in the postfix parameter is affected depends on the object)

- Reset the object parameter that was set using the previous method or perform a custom action (depends on the object)
- ! Remove the object from the map

The + and - prefixes work identically for all entities except those that are not supposed to ever be activated, such as **worldspawn** or **player**. The ! prefix works likewise, and it removes all objects except those that are protected by the engine from inadvertent removal. Such are players and **worldspawn**.

The < and > prefixes, however, are tailored for each entity, because the current implementation does not support configuring arbitrary parameters of arbitrary

objects. Therefore, only the highest-priority parameter (if any) is configured for each entity. For example, in a **func_rotating** entity these prefixes let you change the rotation speed and motion direction; in a **func_screenmovie** entity they make playback start from a specific time in the video file, and so on.

For details about the use of prefixes, refer to the descriptions of particular entities. A postfixed numeric value usually complements a prefix, but some logic entities can pass this numeric value on intact, without using or changing it. Such specifics are also detailed in the descriptions of particular objects.

The primary limitation is that the effect of some prefixes is unpredictable. This detracts from the usefulness of the technique as a global system.

This system has been borrowed virtually unchanged from the Spirit of Half-Life mod. In a nutshell, it enables any entity to be the master of another entity. Being a master means blocking activation or deactivation. Entity states take into account each entity's specifics and purpose. The following states are available:

State	The object is
STATE_ON	Turned on and working
STATE_TURN_ON	Turning on (for example, func_rotating spinning up)
STATE_TURN_OFF	Turning off (for example, func_rotating spinning down)
STATE_OFF	Turned off
STATE_DEAD	Broken (applicable to inanimate objects), doesn't respond to activation attempts any more
STATE_IN_USE	Being used by the player or an NPC

All of these states can be monitored using the specialised **multi_watcher** entity. Alternatively, you can use the **multisource** entity with its simplistic treatment of states: STATE_ON means the object is on, and any other state means it is off. XashXT lets you block the same object with multiple masters just by giving these would-be masters the same name and specifying the name in the **master** field of the entity you want to block. For more details about this technique, see the **description of the multisource** entity.

For this system, the **master** field has been added to almost all objects that support activation. Importantly, all objects that have the **item**_ prefix can be made masters to disallow picking them up.

Some objects cannot be made masters of other objects or of themselves, because that would break compatibility with Half-Life. Such are all **light** entities and the **multi_manager** entity. However, you can still monitor their states using the **multi_watcher** entity and use it as a master.

In the Xash mod, mirrors are placed on a map by applying a texture named **REFLECT** to brush entities or world polygons. On brush entities, you can also change the drawing type, set the transparency or make the mirror breakable—that is, you can use the texture as a regular (non-reflecting) texture.

Mirrors cannot reflect one another.

Water with Fresnel reflection and refraction is rendered on world brushes or on a **func_water** entity if you use a texture named **!REFLECT**.

Note that proper waves are possible only if the normal of the water surface points straight up. You can also change the intensity of ripples using the **WaveHeight** parameter for **func_water** or for **worldspawn**. A **func_water** entity can be correctly attached to any other brush entity and keep its physical properties.

- Water works correctly only with flat surfaces whose normals point straight up.
- Water cannot reflect other water or mirrors. However, it does reflect 3D sky, portals and monitors.

The following parameters affect the configuration of sky and water throughout the game for all players (in multiplayer) and stored in savegames if they are changed (in singleplayer). They are console variables that can be modified in game—for example, using the **trigger_command** entity or a custom-written entity.

Variable	Details
sv_wateramp	Controls wave height for world water.
sv_skyname	Skybox name without a path or extension. Can be modified dynamically in game.
sv_skydir_x, sv_skydir_y, sv_skydir_z	Vector that sets the sky rotation axis. Should be normalised and positive.
sv_skyangle	Current skybox rotation angle about the rotation axis from the previous parameter.
sv_skyspeed	Skybox rotation speed.
sv_wateralpha	World water transparency (requires a special type of compiler with support for translucent water).

There are also the following new options for the **worldspawn** entity:

Variable	Range	Default	Details
gravity	0-800	800	Global gravity factor.
fog			Global, always-enabled, exponential fog throughout the map. Has four integer parameters for the fog field: colour in RGB format followed by fog density (10 is thin, 255 is super dense). The global fog can change its settings temporarily while the player is under water.

Nearly all brush objects that emit any kind of sound have settings for replacement of those sounds. In classic Half-Life, the level designer could only choose a preset that was the closest thing to the required sound, and the path to the sound resource was hardcoded in the engine. XashXT lets you specify the path to the sound in the same field that you normally use for specifying preset numbers. For that, turn off Smart Edit in your Valve Hammer Editor and enter the new sound in the field (usually the **movesnd** and **stopsnd** fields for most brush entities).

For talking monsters, you can specify the common sentence prefix in the special **SpeakAs** field, which you may remember from Spirit of Half-Life. The common prefix is the prefix that all sentences of a particular monster start with. For example, for Barney the prefix is **BA**, for a scientist it is **SC**.

- You cannot redefine sentences for doors and buttons in the **unlocked_sentence** and **locked_sentence** fields. However, you can enter sentence names directly in the **unlocked_sound** and **locked_sound** fields.
- You cannot redefine the error sound for **func_trackchange** and **func_trackautochange**.

This system originated in the Spirit of Half-Life mod. The purpose was to flexibly configure the behaviour of any monster regardless of its class. This lets you have a good and an evil Barney at once, a friendly soldier and a hostile one, and so on. The behaviour of a monster is determined by the **m_iClass** parameter, which is available in all entities with the **monster_** prefix.

Class Details

0	The monster's regular (default) behaviour
1	Machine (for example, a turret)
2	Player
3	Scientist
4	Soldier
5	Alien soldier (for example, monster_alien_grunt)
6	Xen fauna (oblivious of the environment, and vice versa)
7	Regular alien (for example, monster_islave)
8	Headcrab
9	Bullsquid
10	Cockroach, rat
11	Barney
12	Hornet gun projectiles fired by the player
13	Hornet gun projectiles fired by a monster_alien_grunt
14	Class A (hostile to everyone except its own class)
15	Class B (hostile to everyone except its own class)
16	Class C (hostile to everyone except its own class)

ambient_music env_counter env_customize

env_dynlight

env local

env model

env_particle

env_projector

env rain

env_rainmodify

env_sky

env_spritetrain

env static

env_warpball

func clock

func light

func_monitor

func_physbox

func_platform

func_portal

func screenmovie

func_traindoor

generator

hud_sprite

info compile parameters

info portal destination

info_texlights

<u>item_generic</u>

momentary_rot_door

monster_target

multi_switcher

multi_watcher

physboxmaker

player_keycatcher

pushablemaker

scripted_trainsequence
train_setspeed
trigger_bounce
trigger_changeparent
trigger_command
trigger_gravity_field
trigger_impulse
trigger_inout
trigger_lightstyle
trigger_onsight
trigger_playerfreeze
trigger_sound

trigger_startpatrol

Turns the music for the map on and off. The following format for music tracks are supported: mp3 , wma , ogg , xm , it and s3m .

Spawnflags

SF_START_ON	1 (initially turned on)
SF_LOOP	2 (the track is looped)

Option

message Path to the sound file.	message Path to the sound file.
---------------------------------	---------------------------------

Usage Details

- This entity uses the third-party **fmod.dll** library for file playback. Therefore, savegames are not aware of the files it plays, and playback does not continue after a game load.
- The Xash mod is designed to work whether or not this library is available, so using the library is not recommended. Instead, use **target_cdaudio** or **trigger_cdaudio**.

A range-aware counter that can assume special "key" states, carry a value to another counter in a chain and so on.	

Options

maxframe	Sets the upper bound of the counter's range. For example, in a clock that counts up to 12 hours, 59 minutes and so on.
keyframe	Number of the frame where the entity switches to the ON state.
type	 Condition for switching to the ON state: 0—equals 1—is greater than or equal to 2—is less than or equal to
model	Path to a custom model. By default, sprites\decimal.spr is used.

Usage Details

- Activation with the + **prefix** increments the counter value by 1; activation with the prefix decrements it by 1.
- Activation with the < prefix makes the counter's displayed value appear and disappear intermittently, which can be used for indicating the current numeric input. This does not affect the internal state of the counter in any way—the effect is purely visual.
- Activation with a number sets the counter to that number. If the counter is part of a chain of counters, the more significant counters will also get the set command. In addition, this causes the number to be displayed if the previous activation hid it (good for a flashing effect).
- You can reset the counter to zero using the < prefix and -1 as a parameter.
- Activation with the > prefix resets the counter to zero. The reset affects the other counters in the chain.
- This entity can be used for creating floor indicators in lifts, digital clocks, keypads, puzzles, arcade cabinets and so on.

Faithful recreation of the identically named entity from Spirit of Half-Life. Lets you change a variety of parameters for NPCs.

Spawnflags

ISE CISTUM AFFECTION	1 (allows applying settings to dead monsters)
SF_CUSTOM_ONCE	2 (delete the object after use)
SF_CUSTOM_DEBUG	4 (output debugging messages about the object in the console)

m_iVisible	Show or hide the visible model. Works for all objects that have a model, not just for NPCs.	
m_iSolid	Make the model solid or intangible. Works for most objects. Note that if a model is originally intangible, you cannot make it solid. You can only switch between solid and non-solid in a model that is originally solid.	
m_iszModel	The parameter is the path to a new model. Lets you dynamically change the model for any object. Mind that the object with a new model can get stuck in the level architecture. Besides, the physics hull will not be recalculated for it. Use with caution.	
m_voicePitch	Range: 75–120. The pitch of a talking monster's voice. Has no effect on other types of object.	
m_iPrisoner	Makes the monster stop attacking its adversaries. Has the same meaning as the PRISONER spawnflag in monster properties.	
m_iMonsterClip	Makes the monster unaware of invisible barriers called func_monsterclip or aware of them again.	
m_iClass	New monster class. Determines the attitude towards all other monsters and the player. For details, see <u>Monster Classes and Behaviour</u> .	
m_iPlayerReact	 The monster's reaction to the player: 0—class-specific 1—totally oblivious of the player 2—friendly until hurt 3—friendly if hurt inadvertently, meaning that the player hit it with a stray bullet rather than placed an aimed shot; whether or not the shot was aimed is determined by the hit spot 4—unconditionally hostile 	

m_iProvoked	Makes the NPC remember or forget who it was that hurt them. In other words, makes a friendly monster hostile without changing its other parameters. Shooting a monster with the player's weapon does the same.
m_iBloodColor	Colour of the NPC's blood. The colour is a number in the range from 0 to 255 that corresponds to an index in the palette. Red is 247. Yellow is 195.
m_fFramerate	Lets you change the framerate for an arbitrary entity that uses the pev->framerate variable. Whether or not a particular entity will behave as expected is strictly a matter of trial and error, because this has not been carefully tested.
m_fController0	Changes the position of bone controller 0 for an NPC or env_model .
m_fController1	Changes the position of bone controller 1 for an NPC or env_model .
m_fController2	Changes the position of bone controller 2 for an NPC or env_model .
m_fController3	Changes the position of bone controller 3 for an NPC or env_model .
skin	Sets the skin number for a model or the type of contents for a brush. For example, water or lava.
body	Changes the model body. Works for all entities that have models.
m_iReflection	 Lets you change mirror reflection settings: 0—default 1—not reflected in mirrors 2—visible only in mirror reflections

• For parameters where you can perform an explicit switch (turn on or off), response to activation types is highly configurable.

This parameter is specified next to the corresponding option:

CUSTOM_FLAG_NOCHANGE	0 (do not change this parameter)
CUSTOM_FLAG_ON	1 (apply this parameter regardless of the activation type)
CUSTOM_FLAG_OFF	2 (disable this parameter regardless of the activation type)
CUSTOM_FLAG_TOGGLE	3 (invert the state of the parameter)
CUSTOM_FLAG_USETYPE	4 (take into account the activation type—for example, USE_ON will result in turning on, USE_OFF in turning off and USE_TOGGLE in toggling)
CUSTOM_FLAG_INVUSETYPE	5 (the same as CUSTOM_FLAG_USETYPE, but the effects of USE_ON and USE_OFF are inverted)

For objects with multiple possible values, such as **body**, **skin** or **m_iClass**, use the reserved value of **-1** to ignore parameter application.

• The entity also supports the *locus keyword in the target field for applying settings to the activator of the entity.

A dynamic omnidirectional light source without support for shadows.	

Spawnflags

SF_DYNLIGHT_START_ON	1 (initially turned on)
SF_PROJECTOR_NO_ATTEN	2 (turns off the falloff texture; the light is distributed without attenuation as far as you can see)
	4 (the light is turned off when inside a solid object)

radius	Radius within which the light intensity falls off to zero.
lightstyle	Applies a standard light style effect (1–12) to the dynamic light.

- This light source is not designed to be customisable, unlike **env_projector**. Its main purpose is to provide a similar light source to the one in Spirit of Half-Life.
- You can set the light colour using **rendercolor**.

One of the objects of the logic system. Can be turned on and off after a configurable delay. The entity state can be read by other entities. In addition, it can activate and deactivate objects after a delay.

Spawnflag

SF_LOCAL_START_ON 1 (initially turned on and in the ON state)

delay	Delay (in seconds) before turning on.
wait	Delay (in seconds) before turning on.
	Preset value that can be passed on to another objects during activation or deactivation.

- The object activates or deactivates its target after a delay that you set.
- You can use it as a master for other objects.
- It can be a state source for **multi watcher**.
- The object is fully controllable in any of its states and can be turned on and off regardless of the state it is in. To force the ON state, use the < prefix and the value 1. To force the OFF state, use the < prefix and the value 0. In other situations, use the + or prefixes or none at all. This will make the entity reverse its state on every activation.

Similar to env_sprite , but for models. A faithful recreation of the entity from Spirit of Half-Life.

Spawnflags

SF_ENVMODEL_OFF	1 (switches between the two animations specified in m_iszSequence_On and m_iszSequence_Off)
SF_ENVMODEL_DROPTOFLOOR	2 (put the model down on the ground at start)
SF_ENVMODEL_SOLID	4 (the model is solid)

m_iszSequence_On	Animation that is played in an active state: name or number.
m_iszSequence_Off	Animation that is played in an inactive (idle) state: name or number.
m_iAction_On	What happens after the animation has been played in an active state: • 0—stop • 1—replay • 2—switch to the OFF state
m_iAction_Off	What happens after the animation has been played in an inactive state: • 0—stop • 1—replay • 2—switch to the ON state

• Remember that although activating and deactivating the model is essentially switching between the ON and OFF states, effectively this is switching between two animations with flexible post-playback configuration.

A particle emitter whose behaviour is defined in an external script file with the recommended extension .aur.

Spawnflag

SF_PARTICLE_START_ON 1 (initially turned on)

aurora	Path to the particle system configuration file.
attachment	Number of the attachment (overrides the number specified in the script file).

- All particle system settings are located in an external script file. A visual editor is available for editing these files. The editor facilitates particle system creation considerably. For details about particle system configuration, see <u>Aurora Particle System Settings</u>.
- It is safe to use the parent system to attach the emitter to objects, and the emitter's current orientation (and consequently, the particle motion direction) is preserved.

A dynamic projectional light source with flexible settings and support for video texture projection.	

Spawnflags

SF_PROJECTOR_START_ON	1 (initially turned on)
SF_PROJECTOR_NO_ATTEN	2 (turns off the falloff texture; the light is distributed without attenuation as far as you can see)
SF_PROJECTOR_ASPECT4X3	4 (use a rectangular rather than a square beam)
SF_PROJECTOR_ASPECT3X4	8 (use an "upright" rectangular rather than a square beam)
SF_PROJECTOR_NOWORLDLIGHT	32 (ignore world polygons and shine only on brush and studio model)
SF_PROJECTOR_NOSHADOWS	64 (disable shadows for this light source)
SF_PROJECTOR_FLIPTEXTURE	128 (flip the projected texture horizontally; this can provide some interesting effects)

fov	The beam's "field of view", the cone angle.
radius	Radius within which the light intensity falls off to zero.
texture	Name of the texture: a TGA file, a texture from a WAD file (with the .mip extension), an AVI file or a sprite.
falloff	Light falloff pattern within the radius: • 0—automatic, based on the radius • 1—fast (abrupt) • 2—regular • 3—slow (gradual)
lightstyle	Applies a standard light style effect (1–12) to the dynamic light.
framerate	Only for sprites: determines the framerate.

- Playback of animated textures and AVI files is looped. The entity has no options to control this. When the object is turned off and on again, playback resumes from the frame where it stopped, for both sprites and AVI files.
- This parameter is written to savegames.
- You can specify the path to a TGA or .mip texture that is not a single image but a cubemap. For that, omit the view direction suffixes but include the file extension. Although the resulting file name does not exist, it will be treated correctly by the engine as long as the textures that it stands for really exist. Note however that specifying a cubemap will make the light source omnidirectional, and the **fov** option will become meaningless.
- The light source can be attached using the parent system both on the server side and on the client side.
- You can set the light colour using **rendercolor**.

Limitations

- The number of lights in a map is not explicitly limited. However, the player's PVS can contain no more than 64 light sources at a time.
- The current implementation being relatively simple, it may shite through walls. However, this problem is also found in the original Half-Life, where light sources are non-projective.

Creates a rain or snow effect in the map. Used only in conjunction with env_rainmodify.

m_flDistance	Radius within which it rains or snows. The recommended value is 1000 .
m_iMode	Precipitation type: • 0—rain • 1—snow

• The altitude of the entity determines the altitude from which the precipitation falls.

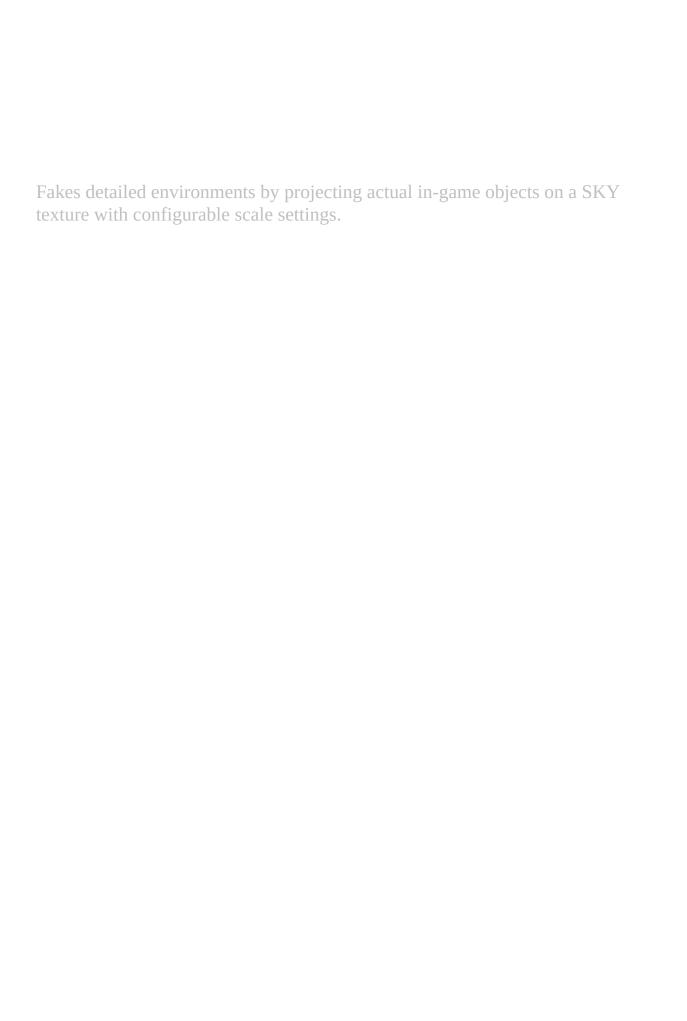
Sets the initial parameters for rain. By placing multiple env_rainmodify enitites in a map, you can start and stop rain, and change its settings.

Spawnflag

SF_RAIN_CONSTANT	1 (it rains constantly and always the same way; other env_rainmodify entities cannot affect this rain)
------------------	---

m_iDripsPerSecond	Number of drops per second. The recommended values are 4000–8000 for rain and 500–800 for snow.
m_flWindX	Wind strength on the X axis.
m_flWindY	Wind strength on the Y axis.
m_flRandX	Randomisation on the X axis.
m_flRandY	Randomisation on the Y axis.
m_flTime	How many seconds it takes to apply the settings (0 means immediately).

- To make rain that can be started and stopped, you need two **env_rainmodify** entities: one with the "it's raining" parameters and the other with the number of drops set to **0**. In both of them, you should apply parameters over time for smooth transitions.
- Similar entities can be added for wind strength control, rain intensity and randomisation.
- Activation occurs through the **trigger_auto** entity or by other means; this is up to the mapper.



Spawnflag

SF_ENVSKY_START_OFF 1 (skybox initially turned off)

fov	Sky field-of-view angle. Changing the value distorts the projection.
scale	Scale relative to the game level. If left blank, causes the sky to act like a painted skybox.

- To create a 3D skybox, use a separate enclosed area. The contents of this area will be projected onto the map's main boundaries covered with the "sky" texture. Using the "sky" texture in the skybox is discouraged, even though it should not cause any problems. If you encounter skybox entities at their original scale "peeking" through the sky texture, try moving the skybox elsewhere so it is not in the line of sight from anywhere in the map. For example, place it below the bottom of the level or use the trick from the demo_1 map.
- The 3D skybox is projected onto a regular SKY texture, unlike Spirit of Half-Life, where a NULL texture or, in later versions, a 3DSKY texture was required.
- In theory, you can attach the entity using the parent system to create interesting effects. However, in practice its motion should take into account the scale of the skybox relative to the level—this means the motions should be slight.
- The default scale (**scale 1**) is taken to be 1/100. That is, the sky is 100 times bigger than the level; at a **scale** of **1.5** it is 150 bigger, at a **scale** of **0.8** 80 times bigger and so on. Note that when scaling is used, the level's total size takes into account the size and position of the skybox (which can affect the level size). The scale can change as a result, and this can throw off the alignment of the world borders with the skybox borders. Therefore, make sure you set up the skybox no sooner than the map is completed. Otherwise, you will need to repeatedly adjust the scale as you go along.
- Since version 0.62, you can turn skyboxes off. This allows you to have multiple skyboxes and substitute them; for example, you can simulate a day and night cycle this way.

Note: If multiple skyboxes are turned on at once, then only the one listed first in the entity list will be drawn.

Limitations

• Using fog, mirrors and portals inside a skybox is not recommended. Skybox behaviour has never been tested in such situations, and the result is undefined. However, the skybox itself is reflected in mirrors and visible in portals and monitors.

A sprite or model that travels on a path whose corner points are set using **path_corner**. This is a counterpart of the entity from Opposing Force.

Spawnflag

SF_SPRITE_STARTON 1 (initially turned on and in motion)

TAPOPI	Name of the first path_corner that the sprite snaps to when activated.
framerate	Sprite animation framerate.
CDAAL	Speed of movement until the next path_corner (path_corner can irreversibly change the sprite's speed).

- The entity respects all **path_corner** settings, as a real **func_train** does. When the sprite reaches a **path_corner** that has a non-zero **delay** value or has the SF_WAIT_FOR_RETRIGGER spawnflag set, the entity stops moving accordingly but keeps animating the sprite.
- Deactivation stops the motion and hides the model.

A static intangible object that does not interact with anything and exists only on the client without wasting a server entity.

Spawnflags

SF_STATIC_SOLID	1 (makes the object intangible and creates a precise physics hull for it based on the visible hull)
SF_STATIC_DROPTOFLOOR	2 (put the model down on the ground at start)

sequence	Number of the animation.
frame	Number of the frame.
COLOCHIAN	Colours of the model's custom textures. For details about the format, see the description of this option in <u>Global Parameters</u> .
skin	Skin.

- Models can play looped animations.
- The entity is written to client-side savegames but is not transferred between levels.
- It is not affected by the parent system.
- Entities with the SF_STATIC_SOLID flag set are not transferred to the client and cannot play back animations.

A prefab object that consists of a sprite and twenty beams, and simulates the effect of a teleporter that releases hostile NPCs.

The settings are identical to those in the Blue Shift counterpart.

Spawnflags

SF_REMOVE_ON_FIRE	1 (delete after activation—for one-off uses)
SF_KILL_CENTER	2 (destroy the teleported object)

radius	Beam reach radius.
warn taropt	Name of the object where the effect appears; by default, it is the entity itself.
damage_delay	Delay before damage is dealt if spawnflag 2 is set.

This object lets you implement the hour, minute or sweep hand in a clock. By using three such objects at once you can make a fully functional clock face.

Spawnflags

SF_DOOR_ROTATE_Z	64 (rotation about the Z axis, similarly to func_door_rotating)
SF_DOOR_ROTATE_X	64 (rotation about the X axis, similarly to func_door_rotating)

	Hand type:
type	0—sweep1—minute2—hour
curtime	Applies only to the hour hand. Format: XX XX XX (vector). Hours (0–12), minutes (0–59), seconds (0–59). The hour hand sets the time for the minute and sweep hands automatically after it finds them within the radius that equals its length.
event	Target for activation. Applies only the hour hand. When the hour changes, two activations are done: first the number of hours is communicated through USE_SET, and then the target is activated through USE_ON. This behaviour is handled well by the generator entity, and you can use it to strike the correct number every hour. If you specify an ambient_generic as a target, then there will be one strike (sound) for every hour.

- A clock cannot be activated or deactivated, its state is always ON.
- The hands are totally independent from one another, and are synchronised only once at map start if the current time is set for the hour hand.
- For correct operation all hands should be oriented to point at 12.00.00.
- Do not try to set the hour using angles. This will be ignored.

A light source.

Spawnflag

SF_LIGHT_START_ON	1 (initially turned on)
-------------------	-------------------------

health	Controls the lamp toughness and whether it is breakable. 0 means unbreakable.
delay	How many seconds it takes the lamp to fully turn on. During this time it will flicker, simulating a fluorescent lamp.

- This light source is static and precompiled. It will not follow your lamp about the level. Make sure the light-emitting texture is included in the global **lights.rad** file or in the **info_texlights** entity.
- A lamp will not sustain any damage unless the name of your texture starts with the ~ character or the +0~ sequence.
- The < prefix lets you break a lamp with a script even if its health option is set to zero.
- A broken lamp cannot be turned on again. When broken, a lamp can flicker randomly for some time.
- Note that for a light-emitting texture it is best to use a combination of +0~ and +A~ textures, where the +A~ is a copy of the texture in a turned-off state. The reverse combination will not work, because the compiler gets information about a texture's emission from the one that is specified explicitly.
- A lamp takes the delay and activation type into account when activating what is in the **target** field. This means that the + prefix is passed for a turn-on activation, and the prefix for a turn-off activation. If a lamp gets broken, the prefix is also passed.

Limitations

- Do not place more than 3 lamps close together, because this would exceed the light style limit in the map and some light sources would look wrong in certain combinations. The compiler shows a message like the following in this situation: "Too many light styles on a face".
- This entity works correctly only with those compilers that are shipped with the Xash mod and located in the **devkit** folder.

An implementation of a real-time surveillance monitor that displays video from a camera. The monitor can be black-and-white.

Spawnflags

SF_MONITOR_START_ON	1 (initially turned on)
SF_MONITOR_PASSABLE	2 (intangible)
SF_MONITOR_USEABLE	4 (pressing +use toggles fullscreen mode, as in Duke Nukem 3D)
SF_MONITOR_NOASPECT	8 (use the horizontal FOV value for both sides; produces a square camera feed)
SF_MONITOR_MONOCHROME	16 (black-and-white screen)

	Name of the camera. The camera can be an info_target entity with spawnflag 1 set.
fov	Field of view.

- Monitors can show themselves through the camera, but you should avoid abusing this feature, because it affects performance.
- Note that a black-and-white monitor will display colour video in fullscreen mode.
- The surveillance camera can be switched in-game using **trigger changetarget**. A monitor shines in the dark when it is on and does not shine when it is off.
- Monitors should be assigned a special kind of texture, whose name starts with **monitor**.
- If video from the monitor is upside down, simply turn the texture around in the map editor.

An object that simulates rigid body physics. Can interact with other objects with the player.	and

Spawnflags

SF_PHYS_BREAKABLE	128 (make the object breakable)
	256 (the object breaks from one crowbar hit regardless of the specified toughness)
I SH PHVS HIII IIABI H	512 (the player can pick the object up, carry it and drop it with the Use key)

material	Material, as in func_breakable and func_pushable . Determines the damage sounds, debris type, sparks and so on.
gibmodel	Path to the custom debris model in case the object is destroyed.
model	Path to the primary physics object model. This can be an mdl or brush model.
health	Object toughness.

- Brush entities cannot be transferred from one level to another, but entities with a regular model are transferable by default.
- The physics hull of an object is built automatically at game start. However, because the hull is convex and PhysX imposes a limit of 256 independent vertices, the hull can be calculated incorrectly for some objects. XashXT produces a corresponding message in the console if it fails to build a hull. For complex concave objects, an approximated best-fitting convex shape will be generated. To view the physics hull, use the following console command: phys_debug 1.
- Stable operation of movers (such as lifts, trains and doors) is not guaranteed due to the way collisions are calculated for the HLBSP format, where 4 approximate hulls are used in collision testing. Therefore, you will get the best results from using cube-shaped objects whose sizes match the sizes of the hulls. Hull sizes are described in **gameinfo.txt**.

A special mover that facilitates construction of multi-storey lifts.	

netname	Target for env_counter to indicate the current floor.
height	Height of a single storey (they must all be the same height).
target	Target that is activated on arrival at the specified floor (even if the current floor is specified).
dmg	If blocked, deal this much damage to the blocking entity.
volume	Volume of motion and stop sounds (0–1).
movesnd	Motion sound (0–13).
stopsnd	Stop sound (0–8).

- The algorithm is based on the notion that given the height of a single storey (this is either the **height** value or the height of the lift itself), the current position of the lift and the location of the first storey (this is always the lift spawn point), you can easily calculate where the lift should go by simply specifying the floor number using the < **prefix** and the floor number as the value.
- Note that this is the only way you can use the object. It does not respond to other types of activation.
- The object moves only vertically, but this limitation can be worked around by rotating its parent as necessary. Generally, you can think of this lift as a discrete mover—an object that can move the requested number of steps, where the step size is known.

An implementation of a portal, similar to those in Portal but less realistic.

Spawnflag

SF_PORTAL_START_OFF 1 (initially turned on and invisible)

target	Name for info_portal_destination.
firetarget	Target that is activated when the portal is passed through.

- All portals must have origin brushes. Portals without one will be deleted from the map.
- Portals use a special texture whose name starts with **portal**. The texture must be precisely aligned to the brush edges, otherwise rendering artifacts are possible.
- A portal must always face the player's zeroed-out YAW angle. In other words, the portal texture must be applied in such a way as to face the gaze of the **info_player_start** whose orientation is 0 0 0. Failing to meet this requirement will cause the image in the portal to distort in uncontrollable ways. To orient the portal in-game as necessary, change its **angles**.
- Portal width also affects image distortion. The distortion is minimal at a width of 1 or 2 units.
- When passing through a portal, the player or any other object keeps the motion direction, acceleration and view angle, but the latter is affected by the angles of the **info portal destination**.
- You can safely attach a portal using the parent system. However, be careful with ROLL rotations.
- If anything is obstructing passage through the portal at the spawn point (for example, the exit is blocked by crates), then the passage will not happen until the blocking object is gone. This ensures that teleportation never results in things getting stuck.
- For double-sided portals (as in the **qstyle** map, for example), player motion tracking and portal reorientation is implemented so that the portal cannot be entered from behind.
- Passage through a portal looks fairly seamless if you disregard the sides.
- A portal can be turned on and off. A turned-off portal is invisible and intangible.

A brush model that plays back a video (AVI) file without sound on its surface.	

Spawnflags

SF_SCREENMOVIE_START_ON	1 (initially turned on)
SF_SCREENMOVIE_PASSABLE	2 (intangible)
SF_SCREENMOVIE_LOOPED	4 (the video is looped)
SF_SCREENMOVIE_MONOCRHOME	8 (black-and-white movie)

movie Path to the AVI file in the **media** folder.

- The brush side where the "movie" is played back must be assigned a texture whose name starts with **movie**.
- The AVI file can have an arbitrary duration, but make sure it uses no rare codecs that are available only on your system.
- Activation with the < prefix (without a parameter) pauses playback. A subsequent activation resumes it. Activation with the < prefix and a parameter instantly skips to the specified frame. The parameter stands for the number of frames from the start of the video file.
- Activation with the > prefix rewinds to the beginning.
- Regular activation with the + and prefixes either disables the screen and stops playback or enables them again.

A door that works in conjunction with **func_tracktrain**. It originated in Spirit of Half-Life 1.9, where it was introduced as an attempt to make up for the limitations of the Spirit parent system. In the Xash mod, it is far less significant but has been kept for compatibility.

Spawnflags

SF_TRAINDOOR_INVERSE	1 (change which way the door slides in a left- or right-door train car, as in Half-Life)
SF_TRAINDOOR_OPEN_IN_MOVING	2 (lets you open the door while the train, func_trackautochange is in motion)
SF_TRAINDOOR_ONOFF_MODE	3 (enables the correct response to the + μ - prefixes; if this is omitted, then the regular func_door behaviour is reproduced: switching)

movesnd	Sound that the door makes when it moves (as in func_door).
stopsnd	Sound that the door makes when it stops (as in func_door).
speed	Movement speed.
train	Name of the train that the door is linked to.

- A door cannot slide up or down, but only left and then backwards or right and then backwards.
- You can adjust the direction using **angles**, but that is pretty much a trick.
- A door that is open will automatically close when its parent train is activated. If the player or anything else is blocking the door at that moment, the train will not start. However, this does not happen if the SF_TRAINDOOR_OPEN_IN_MOVING spawnflag is set—it makes the train stop minding the door state and lose control over it.

Generator of a specified number of target activation impulses. The number of activations can be changed dynamically in-game.

Spawnflag

SF_GENERATOR_START_ON 1 (initially turned on)

maxcount	Maximum number of impulses (with randomisation accounted for) that the generator can produce before it turns off (0 means indefinite).
mode	 Operation mode: 0—regular 1—random 2—random with generation slowdown, as in a "one-armed bandit"
delay	Delay between activations.

- For random operation modes, **maxcount** is mandatory. If it is omitted, the entity will set it automatically in the range from 100 to 200.
- For operation mode 2, the **delay** option is ignored.
- Activation with the < **prefix** and a value sets a new **maxcount**.
- Activation with the > prefix resets the current impulse counter to zero.
- Activation with both the < and > prefixes does not change the generator state.
- Activation with the < prefix and a value is used by the **func_clock** entity to convert the number of hours into the number of target activations.

Faithful recreation of the entity from Spirit of Half-Life. Shows and hides ico	ons

Spawnflag

|--|

model	Name of the icon to display, according to hud.txt . For example, dmg_bio , number_0 , d_skull or anything else. Of course, you can add your own icons by editing hud.txt .
rendercolor	Icon colour.

- Icons are shown whether or not the player is wearing the suit.
- Potentially, icons are capable of following the player from level to level, but consider attaching them using the parent system to be on the safe side.
- The entity can be used for indicating possession of custom items.

HLT compiler options that override those specified at the command line. Options change from version to version, so using this entity is discouraged.	

An alias for **info_target** with spawnflag **1** set. Can be used by **func_monitor** as a camera.

The local version of light-emitting texture definitions in addition to lights.rad .	

A keyvalue pair

The key is the name of the light-emitting texture. The value is the RGB colour and brightness specified as a series of four integers in the range from 0 to 255.

A universal item. Plays back a custom sound and activates a target when picked up. Can be used for creating items in a map editor.

Spawnflag

SF_ONLY_IF_IN_SUIT 1 (allow pickup only if the player is wearing the suit)

model	Model of the item.
noise	Sound or sentence that is played back on pickup.
netname	Name of the icon that shows up on pickup, as for a regular item (see the description in hud.txt).

- The item itself does not modify the player's inventory in any way. You can use it to activate a target in the map so that the pickup is associated with a specific event. This lets you update the player with keys, access cards and so on.
- The player can also carry the item, as they can other items.

An alias for the **momentary rot button** entit with the **Door Hack** flag set. Provided merely for mappers' convenience. The other settings are also identical.

A faithful recreation of the entity from Spirit of Half-Life. Creates an invisible point-based monster that the AIs of regular monsters are aware of.

Spawnflag

SF_MONSTERTARGET_OFF 1 (initially turned off, dead)

frags class of the monster (see Monster Classes and Behaviour).

- This monster cannot be destroyed in the usual way, because it is intangible and invisible. However, you can turn it on and off, which to AIs looks like it dies. You can also attach this monster using the parent system.
- The main purpose is to provide a target (which can be inanimate) for NPCs without interfering with the AI operation and interrupting it with a **scripted_sequence**. For example, you can make friendly NPCs destroy some inanimate strategic barriers set up by the enemy (such as **func_breakable**). For that, simply place the entity in front of the barrier. The NPCs will try to hit the **monster_target**, but since it is intangible, they will hit the object behind it instead.
- Likewise, you can set these monsters up in front of targets in a shooting gallery and turn them off when the targets are hit.

A target switcher that can shuffle targets randomly.

Spawnflag

SF_SWITCHER_START_ON

1 (the switcher is turned on and changing targets as specified by the **mode** option)

mode	 How targets are changed: 0—next 1—previous 2—shuffle
delay	Interval between target changes in automatic mode.

- Regular activations must pass unchanged to the target through the switcher. The switcher itself responds only to the < and > <u>prefixes</u>, which can be used for controlling its behaviour in-game.
- The < prefix without a value turns on an inactive switcher that has the SF_SWITCHER_START_ON spawnflag set. If the SF_SWITCHER_START_ON spawnflag is not set, then each subsequent activation with the < prefix but without a value will initiate the target change in the order specified in the **mode** field. That is, in mode **0** a single activation with the < prefix causes a switch to the next target (without activating it); in mode **1** it causes a switch to the previous target (without activating it); in mode **2** the target will be chosen randomly (without being activated). Activation with the < prefix and a value causes a switch to the target whose number the value indicates.
- Activation with the > prefix resets the current target to the first one and disables automatic target change mode for a switcher with the SF_SWITCHER_START_ON spawnflag set. To enable automatic switching again, activate the switcher with the < prefix without a value, as mentioned above.
- All other kinds of activation are passed on unchanged directly to the target.

An entity that implements the "if-else" logical condition. Turns on if specified entities are in specified states.

logic	Operation logic (AND, OR, NAND, NOR, XOR, XNOR). The words in parentheses should be entered directly in the logic field.
state	Global state that the multi_watcher watches for in all specified entities. If you want to monitor individual states for specific entities, leave this field empty. The following values are accepted: ON, OFF, TURN ON, TURN OFF, IN USE, DEAD. These words should be entered either directly in the state field or next to the individual target names that you specify.
target	Target that is activated when all conditions are met.
offtarget	Target that is activated when the multi_watcher goes from the ON (valid) state to the OFF (invalid) state.

The other fields are filled in as in multi_manager, where the key is the target name to watch and the value is either the state to watch for or, if a global **state** is specified, **0**.

- This is a modified version of the object from Spirit of Half-Life with extended functionality. This object can implement virtually any logical condition that is based on objects' internal states. The states of objects are closely related to their purposes. In other words, for each object a change of state causes object-specific transitions. Some objects are always turned off —for example, logical objects such as trigger_relay or trigger_auto.
 Other objects may be in the IN USE state while the player is using them. Lifts are in the ON state while in motion and so on.
- Overview of operation logic for those unfamiliar with Boolean algebra:
 - AND—The condition is met if all monitored objects assume the state specified in the **multi_watcher** configuration.
 - OR—The condition is met if one or more of the monitored objects assume the state specified in the **multi_watcher** configuration.
 - NAND—The AND condition, inverted. Effectively swaps **target** and **offtarget**.
 - NOR—The OR condition, inverted. Effectively swaps target and offtarget.
 - XOR—Exclusive "or". The condition is met if only one of the monitored objects assumes the specified state, whereas all the others have non-matching states.
 - XNOR—The XOR condition, inverted.
- This entity is multifunctional and can be used anywhere, from keypads to complex interactive scripted scenes that require player participation.
- The object works well as a <u>master</u>.

A fusion of **func_physbox** and **monstermaker**. Inherits options from both entities.

Spawnflags

SF_PHYSBOXMAKER_START_ON	1 (initially turned on)
SF_PHYSBOXMAKER_CYCLIC	4 (each activation creates a new physics object; otherwise, each activation enables and disables their generation)

boxcount	Maximum number of physics objects that the object can generate before it is deleted from the map (0 means indefinite).
m_imaxliveboxes	How many boxes the entity can generate at a time (0 means indefinite).

The other options are the same as for **func_physbox**.

This entity is the logical extension of the functionality of **trigger_impulse**. Captures player keypresses for activating events in the map.

m_iszKeyToCatch	Name of the key to listen for. A single player_keycatcher can listen for only a single (arbitrary) key.
m_iszKeyPressed	Target to activate once when the player presses the specified key.
m_iszKeyHoldDown	Target to activate at every frame as long as the player holds the specified key.
m_iszKeyReleased	Target to activate once when the player releases the specified key.
master	Master that blocks the entity. A blocked player_keycatcher does not activate its targets.

- A map can have no more than 64 instances of the entity.
- You can have multiple **player_keycatcher** entities listening for the same key. Their state can be tracked (STATE_ON means the key is pressed, STATE_OFF means it is released).
- The following table lists the virtual keys:

attack	Primary attack
jump	Jump
duck	Duck/Crouch
forward	Move forward
back	Move backwards
use	Use/Activate
left	Turn left
right	Turn right
moveleft	Move left
moveright	Move right
attack2	Secondary attack
run	Run
reload	Reload
alt1	Unassigned key; to bind it, use the following command: bind < <i>key to listen for</i> > +alt1
score	Unassigned key (Xash mod-specific); to bind it, use the following command: bind < key to listen for > + score

- It doesn't matter where in the map a **player_keycatcher** is located.
- In multiplayer, use the entity with caution, because it watches keypresses by all players on the server.
- The entity does not work if the player is frozen by a **trigger_playerfreeze** or by the **Freeze Player** flag in a **trigger_camera**, because keypresses are bypassed by the engine in these cases, and there is no way to capture them.

A fusion of **func_pushable** and **monstermaker**. Inherits options from both entities.

Spawnflags

SF_PUSHABLEMAKER_START_ON	1 (initially turned on)
SF_PUSHABLEMAKER_CYCLIC	4 (each activation creates a new box; otherwise, each activation enables and disables their generation)

Maximum number of boxes that the object can generate before it is deleted from the map (0 means indefinite).	
How many boxes the entity can generate at a time (0 means indefinite).	

The other options are the same as for **func_pushable**.

• Since the generated objects use the same model, which is built into the map, they all have the same lighting and decals. In other words, a decal applied to one of the boxes will be reproduced on all boxes made by the same **pushablemaker**. This limitation is by design, and it cannot be lifted.

A faithful recreation of the entity from Spirit of Half-Life. Flexibly controls the movement of a func_train .

Spawnflags

SF_TRAINSEQ_REMOVE	1 (remove when the sequence ends)
I SH IRAINSHII IIIRHI I	2 (move towards the object specified by the m_iszDestination option)
SF_TRAINSEQ_DEBUG	4 (debug mode with console messages)

m_iDirection	 Train movement direction: 0—no movement 1—forward 2—backward 3—stop the train 4—as specified by m_iszDestination 	
m_iszEntity	Name of the train to control.	
m_iszDestination	Name of the entity that the train will move towards when the scripted_trainsequence is activated.	
m_iszTerminate	Target activated when the entity has completed its operation.	

Changes the speed and direction of a **func tracktrain** dynamically.

Spawnflags

	1 (activate the func_tracktrain on start)
SF_TRAINSPEED_DEBUG	2 (output debugging information about train starts and stops, and its current speed)

time	How many seconds it takes the train to reach the set speed.
	How the speed setting is applied:
mode	 0—instantly 1—over the specified time interval
train	Name of the train that the new speed is applied to.
speed	New speed.

• The primary purpose of this entity is to slow down a train that approaches a station and speed up a train that departs; using specially placed **path_track** entities with fine-tuned speed settings is clunky. However, you can use the object to better control trains in other ways, because it lets you activate a train, stop it and change its direction.

A faithful recreation of the entity from Spirit of Half-Life. Lets you set bounce options for standard objects that use Half-Life physics (except physics objects).

Spawnflags

SF_TRIGGER_ALLOWMONSTERS	1 (makes the trigger respond to NPCs)
SF_TRIGGER_NOCLIENTS	2 (prevents the trigger from responding to the player)
SF_TRIGGER_PUSHABLES	4 (makes the trigger respond to func_pushable)
SF_TRIGGER_CHECKANGLES	8 (borrowed from Quake; the trigger works only if its angles and the angles of the object are pointing in the same general direction on YAW)
SF_BOUNCE_CUTOFF	16 (decrease the bounce speed to the minimum specified in the armorvalue field)

netname	Filter for activating only objects of a specific class (classname) or with a specific name (targetname).	
armorvalue	Minimum bounce speed.	
frags	 How hard the bounce is: 0—halt 1—100% bounce 	
angles	Bounce direction.	

• The object ignores physics objects and affects only the player, **func_pushable** and various projectiles such as rockets, grenades and snarks.

An entity for dynamically changing an object's parent or unparenting an object.	

target	Entity for which parenting should be changed or broken.
m_iszNewParent	Name of the new parent (leave empty to unparent).

- You can specify the *locus keyword in the m_iszNewParent field. This causes the entity in the target field to become a child of the entity that activated the trigger_changeparent.
- Because the pointer to the activator is usually passed on unchanged by most entities, an arrangement like the following is possible: an entity passes a trigger_multiple or trigger_once, which activates a trigger_changeparent, which makes the object in its target field a child of the entity that passed the trigger. This makes it possible to implement a magnetic crane, for example. However, this is by far not the only use for the technique.

Runs a console command on activation.

netname Name and parameters of the console command.

• Be careful, because not all commands that you use for debugging will work correctly in user mode (think cheats) or some local commands will not work in muliplayer. Generally, use of this entity is strongly discouraged.

Creates an area with modified gravity in the map. Unlike **trigger_gravity**, returns gravity to normal when the area is exited.

The object works correctly only with the player.

gravity

Gravity factor—a factor of the global **sv_gravity** option in the range from 0 to 1.

Example: If **sv_gravity** is 800, a factor of **0.5** produces gravity 400, a factor of **0.25** produces gravity 200 and so on.

• In theory, this entity has the complete set of options found in **trigger_inout**. In practice, however, you may run into problems while changing gravity for NPCs, because most of them rarely leave the ground. What's more, the parameter will have absolutely no effect on physics objects, because PhysX does not support local gravity for arbitrary objects, but only for the entire scene.

This entity implements a feature that was sorely missed by many mappers: an out-of-the-box way to activate an object on a map when a key is pressed.

targetname	Name of the object (used during activation and can be left empty by default).
master	Name of the blocking entity.
target	Target on the map.
impulse	Filter for impulses.

- The steps to set up activation are as follows. The user binds the desired key to generation of an impulse in the range from 1 to 50, for example: bind b "impulse 1"
 Subsequently, when the B key is pressed, this impulse is sent to the server from the player code and activates the universal target named game_firetarget. The activation contains a value that specifies the number of the impulse. The trigger_impulse entity takes over in this chain of operations and is set up in advance to use game_firetarget.
- Specifying an impulse number lets you make sure that only this impulse is caught. Omitting the number makes the trigger react to any impulse it receives.
- The **master** option lets you explicitly block the effects of user keypresses.

A recreation of the entity from Spirit of Half-Life. Triggers twice: activates what is in the **target** field when the trigger area is entered and activates the **m_iszAltTarget** field when the area is exited.

Spawnflags

SF_TRIGGER_ALLOWMONSTERS	1 (makes the trigger respond to NPCs)
SF_TRIGGER_NOCLIENTS	2 (prevents the trigger from responding to the player)
SF_TRIGGER_PUSHABLES	4 (makes the trigger respond to func_pushable)
SF_TRIGGER_CHECKANGLES	8 (borrowed from Quake; the trigger works only if its angles and the angles of the object are pointing in the same general direction on YAW)
SF_TRIGGER_ALLOWPHYSICS	16 (makes the trigger respond to any physics objects, not necessarily func_physbox)

nama	Filter for activating only objects of a specific class (classname) or with a specific name (targetname).
target	Target that is activated when the trigger area is entered.
m_iszAltTarget	Target that is activated when the trigger area is exited.
m_iszBothTarget	Target that is activated when the trigger area is entered and exited.

• If you delete a **trigger_inout** from the map, it activates the targets in **m_iszAltTarget** and **m_iszBothTarget** before it disappears. This happens only in XashXT, but not in Spirit of Half-Life.

A trigger for controlling light styles for a static light. A faithful recreation of the entity from Spirit of Half-Life.

namarn	Custom flickering style that is applied to the target object (and consequently, to all light sources that use the same light style).	
m_iFade	Makes the light fade smoothly.	
m_iWait	How many seconds pass before the light style in the style field is applied.	
style	Light style that is set after the timeout in the m_iWait field (if specified) or immediately (if not specified).	

• Remember that control of static lighting has a number of limitations. In particular, lighting is applied to a specific light style rather than a specific lamp, and the light style can be shared by multiple light sources. For example, changing light style **0** is strongly discouraged, because it will modify lighting for the entire map. However, you can assign a special style to **light_environment** and thereby create a rudimentary day and night cycle.

A faithful recreation of the entity from Spirit of Half-Life. It is triggered when an NPC or scientist is "looking" at it. This is an evolution of Quake's SF_TRIGGER_CHECKANGLES spawnflag.

Spawnflags

>	1 (skip checking for objects in the way of the looker's gaze at the trigger)
SF_ONSIGHT_NOGLASS	2 (ignore glass when checking for objects in the way of the looker's gaze at the trigger)

max_health	Field-of-view angle within which the trigger is considered visible. Use -1 for monsters.
frags	Minimal distance between the objects at which visibility testing occurs. 0 turns off distance-based testing.
message	Classname or targetname of the object that must be looked at instead of the trigger_onsight (this has little significance in XashXT, because a trigger_onsight can be easily parented).
netname	Targetname of the object that looks at the trigger_onsight . To specify the player, leave blank or enter the *player keyword.
target	Target that is activated when the looker's gaze meets the trigger_onsight .
noise	Target that is activated when the looker's gaze leaves the trigger_onsight after meeting it.
noise1	Target that is activated to go into USE_ON when the gaze meets the trigger and then to go into USE_OFF when it leaves the trigger.

• In the classic Spirit of Half-Life demo this object was used for creating a realistic retinal scanner, but more interesting uses can be invented for it. For example, you can unfreeze still monsters as soon as the player turns away from them only to see them in a mirror. Tricks of this sort are standard horror fare.

Classic trigger from Blue Shift and Opposing Force that freezes the player until the next activation. The frozen player can do nothing but look around.

• If the object was activated by something other than a player, then the first player will be frozen. Mind this if you want to use the entity in multiplayer.

A substitute for **env_sound** that lets you define a DSP effect change area precisely rather than approximately.

roomtype DSP preset (0–29).	roomtype	DSP preset (0–29).
-----------------------------	----------	--------------------

- A single **trigger_sound**, can change settings only for one effect, similarly to an **env_sound**.
- To reset a DSP effect to zero, you will need a second object.
- The trigger cannot be activated by targetname, but only by the player touching it.

A faithful recreation of the entity from Spirit of Half-Life. Makes monsters start patrolling interconnected path_corner entities.

_	Targetname of the first path_corner where patrolling starts. Make sure it is within the monster's line of sight. Also note that the monster will stop patrolling when it sees an enemy.
target	Targetname of the monster that patrols the area.

• The technique whereby monsters follow **path_corner** entities originated in Quake and has not really changed in the meantime. Therefore, if a monster refuses to start patrolling, this means that it is in combat or that it cannot find the path to the **path_corner** where patrolling should start.

Global Options New in Existing Half-Life Entities

Global parameters apply to most entities or all entities. Exceptions are cases where applying the parameter makes no sense (for example, rendering options for an object without a model) or there is a limitation introduced out of necessity. Such limitations are documented in the usage notes for individual entities. Note that this section mentions not only new but also customary Half-Life global parameters. The new ones are marked "(new)".

targetname	Found in most entities that have multiple states (normally they are two states: "on" and "off") and for objects that cannot be controlled directly but can have their internal state read, for example through the use of multi_watcher.
target	Contains the name of the target that the object can control. Controlling means not only state change, but also smooth analogue adjustment for some entities (such as momentary_door).
classname	Class of the object. The class cannot be changed by the mapper and is used as a key name with some code attached to it for controlling this class of object. For example, the code attached to the name func_door simulates door behaviour, the code attached to the name monster_barney acts like a regular Black Mesa security guard, and so on.
globalname	The object's global name used for unconditional transfer of the object to a new level if object transfer is not implied by default. For example, there is no point in setting this name for NPCs, because they are transferred to new levels anyway. The main purpose of globalname is to carry over logical objects and objects with brush models that are part of the current map. For successful transfer, set the same name for the necessary objects on all the maps you need. Note: The decals on the object will be transferred along

	with it.
rendermode	 Defines the object's drawing mode. Not all modes are applicable to all types of entities and all models. This is what GoldSource does, and this behaviour was kept in XashXT. • 0 — normal. Opaque objects. Applies to brushes, models and sprites. • 1 — color. Translucent objects without textures. The rendercolor parameter controls object colour. Applies to brushes. • 2 — texture. Translucent objects with textures. The renderant parameter controls transparency. Applies to models, brushes and sprites. Ignores rendercolor. • 3 — glow. Glow sprites with depth testing disabled. The sprite size increases in proportion with distance. The sprites observe the rendercolor and renderant parameters. Applies only to sprites. • 4 — solid. Textures with transparent areas in them, as in grates and bars. The rendercolor μ renderant are ignored. Applies only to brushes. Note: Models and sprites have an internal flag that enables this rendermode, overriding the user setting. • 5 — additive. Contrary to its name, doesn't support applying an arbitrary colour to entities (except sprites). Models in this mode observe lighting. • 6 — world glow. Additional mode from Half-Life 2. Used only for sprites. The difference from glow is that the size of the sprite in world space is fixed, rather than in screen space (which really means scaling the sprite to compensate for the distance from the observer).
rendercolor	Colour in RGB format in the range 0–255. The default value is 255 255 255 (set in the game engine). Works together with rendermode color (for models, sprites and brushes) and rendermode glow , worldglow (only for

	sprites).
renderamt	Object transparency in the range 0–255. Works in the following rendermodes: color , texture , glow , additive , world glow .
renderfx	A set of predefined effects, mainly for models. Normally used for changing model transparency by a preset function. Includes flickering, smooth fade, holographic effect,
	glow shell and a few special modes for internal use.
	Specified as run-together literals of two numbers in the range from 0 to 255: topcolor and bottomcolor (from the palette). For example, for colours 48 and 132 specify 48132 .
	When the value is ambiguous (for example, colours 11 and 12), consider using a hexadecimal value instead of a decimal one—this ensures that each colour gets exactly two places in a four-digit number (for example, 11 and 12 will be represented by 0x0B0C).
	In practice, only the topcolor is often specified, and bottomcolor omitted.
colormap	Works only for models that contain so called remap textures, where part of the palette can be replaced by custom values.
	This feature is normally used for colouring the players on different teams, but it can be used in singleplayer as long as the model contains remap textures. The textures have special naming conventions and distribute the palette in a particular way; for details, see the appropriate documentation.
	Remember also that this parameter increases map loading times, because the engine takes some time to create texture copies.
spawnflags	A set of option flags individual to entities. This is where you should put the values documented in the Spawnflags sections of entity description pages. Most editors represents these flags as a group of check boxes for the convenience of mappers.

m_iClass (new)	Found in all objects whose names begin with monster _ and for func_tank . Lets you set the attitude towards the player and other monsters. For a detailed description of classes, see Monster Classes and Behaviour.
m_iPlayerReact (new)	Found in all objects whose names begin with monster Lets you set the monster's attitude towards the player. For a detailed description, see the description of the env_customize entity.
parent (new)	Name of the parent entity to attach the object to. The objects will be attaches exactly as arranged by the level designer on the map.
movewith (new)	Alias for the parent field. Introduced for compatibility with the Spirit of Half-Life mod.
reflection (new)	 How the entity is reflected in mirrors: 0 — default 1 — do not draw in mirrors 2 — draw only in mirrors
master	Name of the blocking entity. Applies to nearly all objects that can be enabled and disabled.

To keep it brief, only the *new* options are documented; the existing options are omitted.

Naturally, the purposes of the entities will not be mentioned, because the entities are familiar to all Half-Life and Counter-Strike 1.6 mappers.

```
env beam env fade
env funnel
env laser
env render
env shooter
func button
func conveyor
func door (func door rotating, func water)
func_plat (func_platrot)
func_pushable
func rotating
func tank (func tankrocket, func tankmortar, func tanklaser)
func tankcontrols
func trackautochange
func trackchange (func trackautochange)
func tracktrain
func train
gibshooter
info player start
info target
light (light_spot, light_environment)
momentary_door
momentary rot button (momentary rot door)
monster alien grunt
monster barney
monster generic
monster human grunt
multisource
```

multi_manager
path_track
player_weaponstrip
scripted_sequence
trigger_auto
trigger_camera
trigger_changetarget
trigger_multiple
trigger_once
trigger_push
trigger_relay
trigger_teleport
weapon_cycler
worldspawn

Spawnflag

SF_BEAM_SOLID	512 (sets rendermode to normal)

target	Target object that is activated when the beam is crossed by the player or a monster, or an object specified in the netname field.
netname	Classname or targetname of the object that activates the object in the target field by crossing the beam.

• A negative value in the **dmg** field heals the object that crosses the beam.

Spawnflag

SF_FADE_PERMANENT 8 (leaves the screen permanently faded after it reaches the specified dimness)

netname Path to the custom model used by the effect.

Spawnflag

SF_BEAM_SOLID	512 (sets rendermode to normal)

m_iProjection	 Laser projection type: 0—no projection 1—depends on the angle relationship 2—like 1, but mirror reflections are taken into account
m_iStoppedBy	 Filter for objects that can obstruct the laser: 0—glass and monsters 1—only monsters (traced by their hitboxes) 2—glass and monster hulls 3—only monster hulls 4—only glass 5—ignore all except world geometry

- The laser implementation has a limitation whereby crossing the beam and subsequent activation of the object in the **target** field do not work for projection type **2**.
- The **EndSprite** option accepts either a path to the sprite or the name of an existing **env_sprite** in the scene for more flexible sprite configuration.

m_fScale	New scale for the target object.	
netname	Activates what is in the target field after the transformation is complete (the object that activated env_render is treated as the activator).	
frags	How long (in seconds) it takes to apply the effect (relevant to rendercolor , renderamt and scale).	
armorvalue	In how many steps to apply fading. The default value of 0 is recommended.	

:T+NI-	Name that is not for each with		
m_iszTargetName	Name that is set for each gib.		
m_iszSpawnTarget	Target that is activated on each gib spawn; the gib itself acts as the activator.		
m_iBloodColor	Gib blood colour. Red is number 247. Yellow is 195.		
m_iszTouch	Target that is activated when a gib collides with an object or the world (the gib itself acts as the activator).		
m_iszTouchOther	Target that is activated when a gib collides with any object but not the world (the gib itself acts as the activator).		
m_iPhysics	 Type of physics to simulate: 0 — regular gib physics 1 — stickygibs; sticks to the first thing it touches, including moving objects 2 — noclip; no collisions at all 3 — FLYMISSILE; ignores gravity and uses a special hull for collision testing 4 — BOUNCEMISSILE; bounces off walls, ignores gravity and uses a special hull for collision testing 5 — TOSS; does not use extra bouncing, but simply slides on the floor, respecting gravity 6 — PHYSIC; uses the physics engine for rigid body simulation 		
skin	Particle skin. -1 is random skin.		
body	Particle body1 is random body.		
framerate	Gib animation speed (for sprite models).		

scale	Scale (for models and sprites).	
frame	Frame number (for models and sprites).	
m_fFriction	Friction magnitude. Be careful when setting this for physiscs types that don't respect gravity, because they are friction-aware, and your particle will decelerate very rapidly.	
m_vecSize	Particle size (deprecated).	

• Currently, the particles that are simulated by the physics engine perform testing for collision with the player. Because they are small, this can cause them to be pushed outside the level. Be careful.

Spawnflags

	16 (the button cannot be used through thin walls, which was a known issue in the original Half-Life)
SF_BUTTON_DAMAGED_AT_LASER	128 (the button will be activated when hit by an env laser beam; see the demo maps)

• There is an undocumented feature that makes a button with the SF_BUTTON_DAMAGED_AT_LASER spawnflag set respond to a **weapon_egon** beam in addition to an **env_laser** beam. Consider using this quirk to arrange alternative quest routes.

SF_CONVEYOR_STARTOFF 4 (the conveyor is initially deactivated)

• In XashXT, a conveyor can be flexibly controlled using **prefixes**. The < prefix changes the conveyor's current speed, multiplying it by a number in the range from -1 to 1. For example a conveyor travelling at 500 with the speed change factor set to -1 will be given a negative speed of -500; a factor of **0.5** will result in a speed of 250, and so on. Speed change configuration example:

"target" "<conveyorname.0.15"

• Note that this kind of activation will only queue the new speed for the conveyor but will not change it immediately if the conveyor is in an active state. You should either change the speed for an inactive conveyor (the speed will be applied automatically when it starts) or confirm the speed change with a subsequent activation that uses the > prefix. Speed change configuration and confirmation example:

"target" "<conveyorname.0.15"

"target" ">conveyorname"

- To perform double activation, you will need two <u>trigger_relay</u> entities or a <u>multi_manager</u>. As an added bonus, you can achieve smooth speed change using a <u>momentary_rot_button</u> entity (an example is available in the demo maps).
- Activation with the < prefix but without the current value will cause the conveyor to stop.

Note: As illogical as this control scheme is, it is necessary for compatibility with the original Half-Life and its maps.

4 (formerly in Quake, "Door don't link"; SF_DOOR_ONOFF_MODE makes the door respond to being activated with the + and - prefixes)

chaintarget	Name of the next door in a chain of activations (deprecated).	
message	Activates the specified target as soon as the door is completely open.	

	Makes the platform slide not only up and down but also sideways. Used together with height , it also makes it slide diagonally.	
angles	Used together with the width option to set the motion direction for the func_plat .	

	512 (the player can pick up and carry the object, as in Half-Life 2)
SE DIISH DSDCOLLISION	1024 (use the object's precise shape instead of a bounding box for collision testing; a good choice for making a boat, for example)

• The SF_PUSH_BSPCOLLISION spawnflag not only used the object's visible physical shape, but also lets a **func_pushable** move other objects placed on top of it. Examples can be found in demo maps.

SF_ROTATING_STOP_AT_START_POS

1024 (the brush will stop at the exact position it was launched from; good for rotating lifts)

target to activate when the object is blocked. Activation is repeated every half-second until the blocking object is removed.

- The < prefix, followed by a dot and a number, lets you flexibly control the object speed, multiplying it by a number in the range from -1 to 1, as in **func conveyor**.
- If the **Acc/Dec** flag is turned on for the entity, then the speed will be applied over time rather than immediately.
- Activating a stopped object with the < prefix will cause it to start. Activating an object with the < prefix but without a numeric value will cause it to stop.
- Whether an object stops smoothly also depends on the state of the **Acc/Dec** flag (this is a standard flag in a **func_rotating**).
- Activation with the > prefix and a value of **0** lets you reverse the rotation of a ventilator and give it maximum speed.
- Activation with the > prefix and any non-zero value gives the object an
 angular impulse whose magnitude is determined by the value you specify.
 The object must be in the STATE_OFF state when this happens. The
 impulse itself does not change the state of the ventilator, which remains
 turned off. This effect can be used, for example, to fake equipment
 malfunctions.

SF_TANK_LINEOFSIGHT	16 (this flag is present in the original SDK but not documented anywhere; the tank shoots only if the enemy is directly visible)
SF_TANK_CANCONTROL	32 (unlike the original implementation, here this flag prevents the machine gun from working in automatic mode)
SF_TANK_MATCHTARGET	128 (in manual control mode, the tank's main gun aims where the player is looking; good for controlling multiple guns)

firemaster	Master that blocks shooting but allows control and aiming. Can be used for simulating depleted ammunition.
firetarget	Target that the tank fires at in automatic mode. Accepts a classname or a targetname.
m_iClass	Class of the tank—really an NPC class. For more details about classes, see <u>Monster Classes and Behaviour</u> .

- If **firetarget** and **m_iClass** are not used, the **func_tank** fires only at the player.
- Note that the IN USE state is achieved whenever the tank attempts to fire, whether or not shooting is blocked by a master. This state can be set both manually and automatically.

- The object supports up to 48 controllable tanks at once, with identical or different names.
- To specify tanks with identical names, use the **target** field. For tanks with different names, use the **multi_manager** technique, where the tank name is a key, and the value is **0**.
- Technically, the number of controllable tanks can be higher, but in practice you run out of fields in the Valve Hammer Editor.

message	Activates the target when the object has reached its low point.	
netname	Activates the target when the object has reached its high point.	
width	Makes the platform slide not only up and down but also sideways. Used together with height , it also makes it slide diagonally.	
anoiec	Used together with the width option to set the motion direction for the entity.	

• Leave the **trainname** field empty if you want the entity to work only with the train that activated it. In this case, activation by objects that are not trains will have no effect, because the train will not be found.

SF_TRACK_ONOFF_MODE enables explicit activation using

32 (similarly to the flag in **func_door**, the - and + <u>prefixes</u>)

radius	Custom radius for detecting the train that needs to have its track changed.
message	Activates the target when the object has reached its low point.
netname	Activates the target when the object has reached its high point.
width	>Makes the platform slide not only up and down but also sideways. Used together with height , it also makes it slide diagonally.
angles	Used together with the width option to set the motion direction for the entity.

• Leave the **trainname** field empty if you want the entity to look for the eligible train within the radius of the largest dimension of its physical size, or within the custom radius if its own size is too large (for example, in a long **func_trackchange**).

SF_TRACKTRAIN_NO_FIRE_ON_PASS

16 (ignore target activation for **path_track** entities specified in the **message** field)

soundstart	Path to the custom train start sound.
soundstop	Path to the custom train stop sound.
master	Blocks train control.

- Do not attach **func_traincontrols** using the parent system, because this is not needed.
- Note that **train_setspeed** supports the ***locus** keyword for changing the speed of the train that crosses a specific **path_track**.

SF_TRAIN_SETORIGIN

2 (align the **func_train** over a **path_corner** using the origin brush instead of the model's actual centre

netname Target to activate when the train crosses a **path_corner**.

m_iszTargetName	Name that is set for each gib.	
	Target that is activated on each gib spawn; the gib itself acts as the activator.	
m_iBloodColor	Gib blood colour. Red is number 247. Yellow is 195.	

SF_START_WITH_SUIT 1 (start the game with the suit on; applies only to singleplayer)

SF_HACK_VISIBLE 1 (forces the transfer of the object to the client while keeping it invisible)

m_iOnStyle	Light style number used when the light is on.
m_iOffStyle	Light style number used when the light is off.
m_iTurnOnStyle	Light style number used while the light is being turned on over time.
m_iTurnOffStyle	Light style number used while the light is being turned off over time.
m_iTurnOnTime	How many seconds (an integer, not a fraction) it takes to turn on the light with the light style in m_iTurnOnStyle applied.
How many seconds (an integer, not a fraction) it to turn on the light with the light style in m_iTurnOffStyle applied.	
firetarget	Target that will be activated when the light switches to the "On" or "Off" state.

• As before, you can use **pattern** only for lights that are on, and light styles for lights that are being turned on or off are limited to the standard set numbered from 1 to 19. This technique can be used for faking luminescent lamps, which flicker while turning on.

2 (accepts an explicitly set speed rather than the delta which depends on the distance to travel)
4 (formerly in Quake, "Door don't link"; makes the door respond to being activated with the + and - prefixes)

movesnd	Name of the door motion sound (the same field as in func_door).
speed	Used in conjunction with the SF_DOOR_CONSTANT_SPEED flag.
distance	The same field as in momentary_rot_button , but here map units are specified instead of degrees (by default, this is the door size minus the "lip").

m_flStartPos Sets the initial state of the button. Ranges from 0 (default) to 1.

- Unlike the original Half-Life, **momentary_rot_button** lets you change the output value smoothly within the range from 0 to 1 not only for **momentary_** objects, but also for any others that may benefit from such a change—for example, **func_rotating** or **func_conveyor**.
- The object responds to the > <u>prefix</u> accompanied by a dot-preceded value in the range from 0 to 1. This lets you instantly set the rotation angle you need when a specific event occurs.
- When the < prefix is accompanied by a dot-preceded value in the range from 0 to 1, the entity sets the requested angle at the speed that is specified in the button settings, rather than instantly.

SF_MONSTER_WPN_DROP 1024 (on death, drop **weapon_hgun**)

SF_MONSTER_NO_WPN_DROP 1024 (on death, hold on to the weapon)

frags	Set to 1 to equip Barney with a Magnum.
א א במחל	Initial prefix for all sentences that the monster uses (for example, BA for Barney, SC for a scientist).
	Sentence spoken by the NPC when it refuses to follow the player.

Spawnflag

SF_HEAD_CONTROLLER 8 (makes the monster turn its head in search of someone to talk to, usually the player)

m_bloodColor Blood colour (also determines the gib type).

Spawnflag

SF_MONSTER_NO_WPN_DROP 1024 (on death, hold on to the weapon)

offtarget Activated during reverse state changes—that is, when the master blocks an object again.

Usage Details

• The **multisource** implementation has been corrected for proper handling of NPCs' **TriggerTarget** field. In the original the names of entities specified in these fields were plainly ignored. An example of a script running into this kind of problem can be found in map c2a4a in Half-Life. Two **monster_headcrab** entities have the name **multisource** specified in their **TriggerTarget** fields, but the **multisource** cannot process them correctly. In the end, the script *does* work, but only thanks to monster AIs and a radius configured for the **scripted_sequence**.

Spawnflags

SF_MULTIMAN_LOOP	4 (target activations occur in cycles)
SF_MULTIMAN_ONLYONCE	8 (remove the object from the map after all targets have been activated)
SF_MULTIMAN_START_ON	16 (the entity is initially activated at map start)

delay Global delay before all target activations.

Usage Details

- While the global delay is in effect, the entity is in the STATE_TURN_ON state. During target activations, it is in the STATE_ON state. You can enable and disable target activation using the and + prefixes. A new activation will start the cycle over.
- The > prefix causes the object to reset the cycle without turning off. However, this same prefix causes the object to turn on if it is off.
- The < prefix activates all targets at once.

Spawnflag

SF_PATH_TELEPORT

16 (like **path_corner**, allows teleporting of a **func_tracktrain** and all objects attached to it)

	Target that is activated only when the train passes the path_track in the regular direction.
m_iszFireRev	Target that is activated only when the train passes the path_track in the reverse direction.

Usage Details

• If no alternative path is specified, then activating a **path_track** causes it to turn off, simulating a broken track. If an alternative path is specified, activation causes switching to it or switching back. That said, this is regular behaviour for the object, and it does the same in Half-Life.

Spawnflags

SF_REMOVE_SUIT	1 (takes the suit away from the player)
SF_REMOVE_CYCLER	2 (takes the weapon_cycler away from the player)

Usage Details

• All other weapons are taken away independently of the cycler and the suit.

m_iszFireOnBegin Activates the target as soon as the monster starts running the sequence.

Spawnflag

SF_FIRE_ON_XASH 2 (activate the target only if XashXT is running on the Xash3D engine, not GoldSource)

m_iszViewEntity

Name of the entity whose point of view the camera assumes.

Usage Details

- The entity accepts the *locus keyword, which switches the target to the activator of this entity.
- It can help change the target for area-patrolling monsters using **path_corner** and **func_monitor** entities (or **func_portal** entities) to switch the camera (or the portal's destination).

Spawnflags

SF_TRIGGER_CHECKANGLES	8 (borrowed from Quake; the trigger works only if its angles and the angles of the object are pointing in the same general direction on YAW)
SF_TRIGGER_ALLOWPHYSICS	16 (tells the trigger to respond to all physical objects, not necessarily func_physbox)

netname Filter for activating only objects of a specific class (classname) or with a specific name (targetname).

Spawnflags

SF_TRIGGER_CHECKANGLES	8 (borrowed from Quake; the trigger works only if its angles and the angles of the object are pointing in the same general direction on YAW)
SF_TRIGGER_ALLOWPHYSICS	16 (tells the trigger to respond to all physical objects, not necessarily func_physbox)

netname Filter for activating only objects of a specific class (classname) or with a specific name (targetname).

target

Target (for example, an **info_target**) that will switch the entity to Quake 3 Arena-like mode on activation. This means the location of the target will mark the place towards which the trigger will launch the player or an object.

m_iszAltTarget	Target that is activated if the object is blocked by a master.
Wallie	Lets you override the dot-preceded value in the object activation field (deprecated).

Spawnflags

SF_TRIGGER_CHECKANGLES	8 (borrowed from Quake; the trigger works only if its angles and the angles of the object are pointing in the same general direction on YAW)
SF_TRIGGER_ALLOWPHYSICS	16 (tells the trigger to respond to all physical objects, not necessarily func_physbox)

netname Filter for activating only objects of a specific class (classname) or with a specific name (targetname).

aeniov	Ordinal number of the animation played when the weapon is equipped.
HOISIEL	Ordinal number of the animation played when the weapon is holstered.
primary	Ordinal number of the animation played during primary fire.
secondary	Ordinal number of the animation played during secondary fire.

Usage Details

- Unlike the original, this entity now occupies a slot, shows up in the HUD as weapon_question and can play animations when the weapon is fired, equipped or holstered.
- In addition, **player_weaponstrip** has a special flag to make it possible to take this entity away from the player.

See Environment Variables.

XashXT provides a number of debugging facilities intended for making a level designer's job easier. Most of these features are enabled through console variables or commands, but sometimes debugging is turned on for specific entities using special spawn flags (see the descriptions of <u>new entities</u> or <u>revisions of existing entities</u>) for details.

Listed below are console variables and commands, with usage details.

Note: Do not try to use these commands and variables as part of gameplay. Most of them will not work in user mode.

Console Variables

phys_debug	Enables collision rendering for the PhysX engine.
phys_qdebug	The current implementation only draws centres on func_door_rotating entities.
p_speeds	Outputs information about the current state of the PhysX engine and objects in the scene.
r_speeds	Takes values in the range from 1 to 9. Outputs miscellaneous information about the renderer state.
r_showtextures	Takes values in the range from 1 to 14. Displays loaded textures grouped by specified criteria.
sv_novis	Disables PVS culling of entities on the server so that you can distinguish it from client-side culling.
r_novis	Disables PVS culling of entities on the client. This normally affects only world polygons, because entities are culled on the server.
r_lockpvs	Prevents PVS refreshing so that you can see the drawing distance.
r_lightmap	Draws lightmaps without diffuse textures.
gl_wireframe	Enables edge drawing, showing how faces are split.
r_drawentities	The default value is 1 . Modes 2 to 6 draw bones, hitboxes etc.
gl_renderer	The default value is 1 . Setting this to 0 switches to the pure engine renderer and lets you compare the graphics.
showtriggers	Shows triggers on the map. Works only after a map restart.

Console Commands

edicts_info	Outputs brief information about edicts.
entity_info	Outputs detailed information about entities.
entpatch	Dumps a script file with entity definitions. The file can be edited manually. Subsequently, the engine loads this file instead of built-in entities.
fire	Enables activation of entities that the player is looking at or activation of the entity that is specified as an argument of the command.
impulse 103	Outputs information about the state of a monster AI.
impulse 104	Outputs information about the state of global entities (their state can be modified using env_global).
impulse 106	Outputs information about the state of regular entities (name, global name, model, parent).
impulse 107	Outputs the name of the texture that the player is looking at.
impulse 203	Removes an object from the map (except world and players, of course).
showtriggers_toggle	An alternative to the showtriggers console variable. The difference is that it shows triggers without requiring a map restart. However, most contemporary compilers (ZHLT, VHLT) perform an optimisation that deletes all triggers with the AAATRIGGER assigned and leave only the physics hull without a visible hull. To disable this behaviour, compile your map with the -nonullifytrigger command-line parameter using hlcsg . Alternatively, use a texture with a different name for triggers.

The Aurora particle system first appeared in Spirit of Half-Life 1.2, but was severely limited. In particular, the way it was attached to objects was fairly buggy. Because of that, the system could not be used to its full potential.

In XashXT, most of the system's bugs have been fixed, which has made possible the creation of new interesting effects. A visual editor (xash\devkit\tools\AuroraEditor.exe) has also been added, making it a lot easier to create particle systems.

You can test the effects of changes to most parameters directly in the particle editor, so I will not go into further detail. Only the mandatory essentials will be described.

Adding a Particle System to a Map

To put a particle system in a map, use the **env_particle** entity. Remember that linking to the attachments on models cannot be done using mapping techniques, because a particle system is aware of the attachment number only on the model of the entity that it is linked to. Since **env_particle** contains no model, this linking method makes no sense. However, it can be useful for developing a new mod based on XashXT, where you can make, for example, new rocket jets, smoke trails etc.

Note that it takes the object some time to restore particles on savegame load. As a rule, such behaviour is common to most particle system implementations in mods and game engines, so this is nothing out of the ordinary.

The path to the script file must be full and include the extension. This lets mod authors store particle system definitions in folders of their own choice. The extension does not matter—instead of **aur**, it can be **txt** or anything else.

For examples of script files, see the **xash\aurora** folder.

Main Section Options Auxiliary (Emitter) Section Options

This section is not enclosed in braces, and its parameters can occur only once per definition.

particles	The total number of particles that the system can handle. The number of particles consumed does not depend on this parameter—it is determined by several other factors: mean particle lifetime, particle deletion method and so on. If the number of particles is set too high, some portion of them may never come to be used. Conversely, if the number is too low, the effect will be "diluted".
maintype	Name of the primary particle emitter; its definition follows the main parameters in the first section.
attachment	Attachment number on the model of the entity that the system is attached to. As mentioned elsewhere, this parameter has no effect in the current implementation.
lightmodel	 Particle lighting type: 0 — regular 1 — sample the environment lights 2 — sample the environment lights and blend in the env projector projection
killcondition	Type of medium that disables the particle emitter on contact. There are three types: • empty — doesn't work in the open air • water — extinuished by water • solid — extinguished by the level architecture

These options are enclosed in braces and define particle emitters, including drawing settings and physics.

name	Name specified in the maintype field in the main section, in the spraytype field in the emitter section or in the overlaytype field in the overlay definition section.
gravity	Rate of gravity influence on particles. Note that the rate should be negative if you want your particles to travel towards the ground. The parameter can take a random value from a range specified using a double dot. Example: -300280.
windyaw	Direction of the virtual wind (which affects only this particle system)—the YAW angle in the range from 0 to 360 degrees. The parameter can take a random value from a range specified using a double dot.
windstrength	Strength of the virtual wind. The parameter can take a random value from a range specified using a double dot.
sprite	Path to the sprite to use as the particle texture for this emitter.
startalpha	Transparency value that a particle is assigned at birth. The value can range from zero to one. The parameter can take a random value from a range specified using a double dot.
endalpha	Transparency value that a particle is supposed to have when it dies. The value can range from zero to one. The parameter can take a random value from a range specified using a double dot.
startred	Red colour value that a particle is assigned at birth. The value can range from zero to one. The parameter can take a random value from a range specified using a double dot.
	Red colour value that a particle is supposed to have when it dies. The value can range from zero to one. The

endred	parameter can take a random value from a range specified using a double dot.
startgreen	Green colour value that a particle is assigned at birth. The value can range from zero to one. The parameter can take a random value from a range specified using a double dot.
endgreen	Green colour value that a particle is supposed to have when it dies. The value can range from zero to one. The parameter can take a random value from a range specified using a double dot.
startblue	Blue colour value that a particle is assigned at birth. The value can range from zero to one. The parameter can take a random value from a range specified using a double dot.
endblue	Blue colour value that a particle is supposed to have when it dies. The value can range from zero to one. The parameter can take a random value from a range specified using a double dot.
startsize	Scale of a particle at birth. The value must be greater than zero; otherwise, the particle will not be drawn. The parameter can take a random value from a range specified using a double dot.
endsize	Scale of the particle when it dies. The value must be greater than zero; otherwise, the particle will not be drawn. The parameter can take a random value from a range specified using a double dot.
sizedelta	Mean random scale of a particle (for situations where smooth size changes throughout a particle's lifetime are not needed). The parameter can take a random value from a range specified using a double dot.
startangle	Tilt angle of a particle at birth. The angle can be in the range 0–360. The parameter can take a random value from a range specified using a double dot.
angledelta	Change to the tilt angle of a particle per frame. The angle can be in the range 0–360. The less the value, the slower the particle sprite rotates. The parameter can take a

	random value from a range specified using a double dot.
startframe	Start frame set for a particle at birth. The value cannot be greater than the total number of frames in the sprite used. The parameter can take a random value from a range specified using a double dot.
endframe	Used as an alternative to framerate . End frame set for when a particle dies. The value cannot be greater than the total number of frames in the sprite used. The parameter can take a random value from a range specified using a double dot.
framerate	Used as an alternative to endframe . Sets how many frames the particle sprite plays back per second. The parameter can take a random value from a range specified using a double dot.
lifetime	Lifetime of a particle in seconds. The parameter can take a random value from a range specified using a double dot.
spraytype	Name of a new particle emitter that is a child of this particle emitter.
overlaytype	Name of a new overlay particle emitter that is a child of this particle emitter.
sprayrate	Spray intensity. Directly affects how the number of particles allocated to the system is consumed. The parameter can take a random value from a range specified using a double dot.
sprayforce	Particle acceleration rate for this emitter (added to that of the parent emitter). The parameter can take a random value from a range specified using a double dot.
spraypitch	Local PITCH angle component of the particle spray orientation. A local angle is used because the global angle is assumed to come from the orientation of the env_particle entity—that angle may be local if env_particle is attached to some entity on the server. The angle can be in the range from 0 to 360. The parameter can take a random value from a range specified using a double dot.

sprayyaw	Local YAW angle component of the particle spray orientation. A local angle is used because the global angle is assumed to come from the orientation of the env_particle entity—that angle may be local if env_particle is attached to some entity on the server. The angle can be in the range from 0 to 360. The parameter can take a random value from a range specified using a double dot.
drag	Drag coefficient (generalised, not skin friction) that decreases the speed of a particle throughout its lifefime. The parameter can take a random value from a range specified using a double dot.
bounce	Coefficient for particles bouncing off solid surfaces. At 0 , the particle sticks; greater values make it bounce off harder. Ranges from zero to two. The parameter can take a random value from a range specified using a double dot.
bouncefriction	Friction coefficient for slowing down a particle on bounce. The parameter can take a random value from a range specified using a double dot.
rendermode	Drawing mode for all particles of the emitter. Can have the following values: additive , solid , texture , color .
drawcondition	 Type of medium where a particle can be drawn. When it hits a different medium, it is not drawn any more. By default, it is drawn everywhere. The parameter can have the following values: empty — drawn in the air solid — drawn only inside architecture special1, special2, special3 — custom medium types that you can set for func_illusionary to prevent particle rendering outside that medium spotlight — virtual env_projector frustum inside which a particle is drawn (only drawn; the lightmodel parameter is not affected in any way)— this is a good way to fake dust specks in a cone of light

Note: Emitter definitions must be enclosed in curly braces and given names. The main parameters must not be enclosed in braces.

- To what extent is XashXT compatible with standard Half-Life entities?
- To what extent is XashXT compatible with standard Spirit of Half-Life entities?
- What's the current released XashXT version?
- What version of the Xash3D engine is required for XashXT 0.6 to work correctly?
- Will it work on the original GoldSource?
- Does XashXT work in multiplayer?
- This is a great project; how can I help?
- I'd like to continue the development of the toolkit on my own. How should I go about it?
- I've downloaded it, and now it won't install/crashes/hangs up; what do I do?
- Bloody hell! Can you believe this shit?!

To what extent is XashXT compatible with standard Half-Life entities?

Compatibility approaches 100% with regard to entity settings and rendering results. Many graphics enhancement mods for Half-Life are known to change the look of the game dramatically, even if none of the enhancements are enabled. During XashXT development, extra care was taken to maintain the compatibility. You can compare the visuals rendered by XashXT's custom renderer with the engine renderer by toggling the **gl_renderer** console variable. As regards entity compatibility, I should note that Half-Life can be completed under XashXT without issues. In theory there may be discrepancies affecting user maps, but examples of this have not yet cropped up.

To what extent is XashXT compatible with standard Spirit of Half-Life entities?

In this area, compatibility is much lower, because the mod never aimed at full compatibility with Spirit. Wherever entities or options are borrowed from Spirit,

this is mentioned in the documentation. Still, some entities should work well, and this has confirmed by some of the Spirit demo maps running successfully.

What's the current released XashXT version?

The current final version of XashXT is 0.61. This is a stable and thoroughly debugged version with detailed documentation. You can use it for your mods as it is or with your custom modifications, which is possible due to its sources being open. If I the author of XashXT find that the mod is popular and used for games or interesting demo maps that give the project good publicity, then I may resume development and release version 0.7 that would use bump and parallax mapping, and a lot of other exciting stuff widely used in modern games. Sadly, at this point I simply don't have the time. Generally speaking, if some kind of community emerges, this will increase the chances of the project's continued development.

What version of the Xash3D engine is required for XashXT 0.6 to work correctly?

The recommended version is Xash3D 0.95 build 2009 or higher. However, the mod also works on older builds, all the way down to and including build 1905, although some features become disabled. It is not guaranteed to work on builds prior to 1905.

Will it work on the original GoldSource?

Unfortunately not. The main goal of XashXT has been to advertise the Xash3D engine and its new interfaces, which make it possible to add lots of exciting features that are either unavailable for GoldSource by definition or available through hacks and prone to cause instability and resource hogging (most custom renderers for GoldSource keep multiple texture copies in video RAM, because this is the only way to handle them). Thus, XashXT showcases the new features that are there precisely because of the extended interfaces, so naturally it will not work on GoldSource. You can actually run XashXT on GoldSource, but most

objects will be dead and will not budge. And the custom renderer will be disabled altogether.

Does XashXT work in multiplayer?

Yes it does, but its main purpose is singleplayer. Besides, the player prediction system is incomplete in Xash3D, so you will experience noticeable lagging while playing over the Internet. Some problems with physics objects may also occur for remote players, but this has not yet been carefully tested.

This is a great project; how can I help?

You can make a donation—the wallet number is 410011011597633 in the Yandex.Money system.
There is also the WebMoney wallet R921161786039

I'd like to continue the development of the toolkit on my own. How should I go about it?

If you are serious about implementing my ideas independently, I recommend naming the project XashXT: Custom Build so that there is no confusion. Personally, I don't mind a project like that at all as long as the quality stays high.

I've downloaded it, and now it won't install/crashes/hangs up; what do I do?

Technical support in case of problems is available at our forum in <u>this section</u> (link opens in a new window).

Register an account at the forum and ask your question in English or Russian. It will be answered for sure.

Bloody hell! Can you believe this shit?!

And how!

Main developer

Unkle Mike

Additional coding

n00b

Art

Chorus Small Link

Demo maps

Unkle Mike thambs Scrama

Project hosting and technical support

XaeroX

Beta testers

Qwertyus

FiEctro thambs Raid Flash_AD

Documentation and translation

nemyax