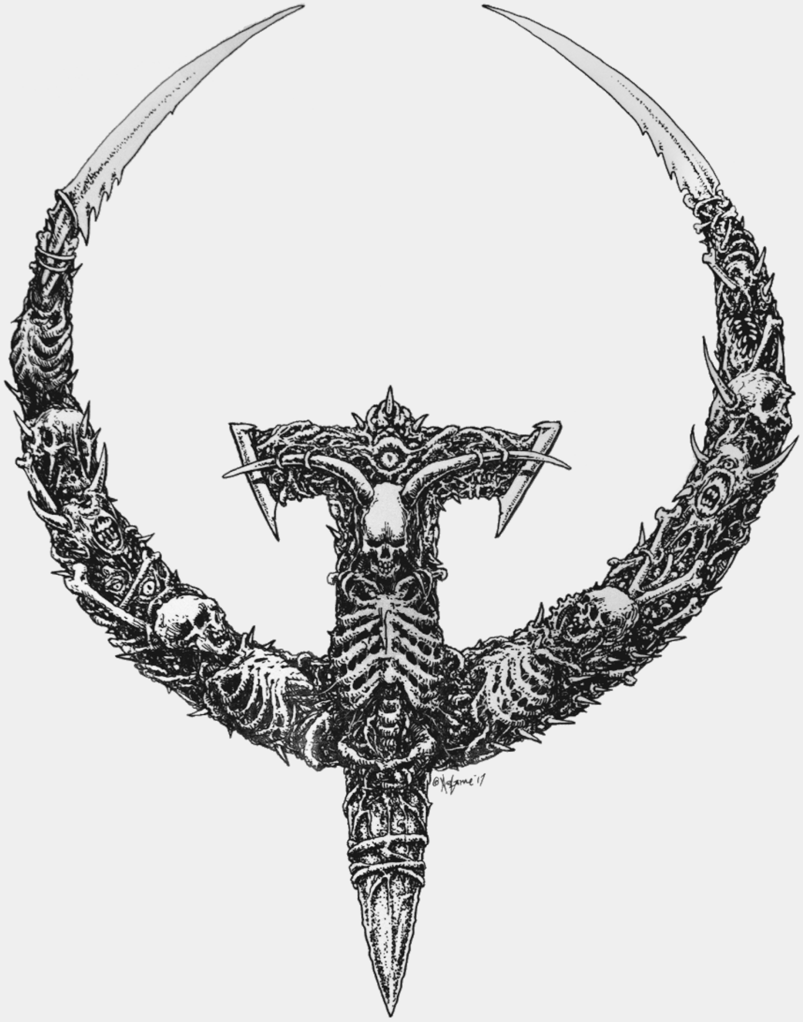


progs_dump

a devkit for QUAKE

by dumptruck_ds & iw
lango.lan.party@gmail.com

version 1.2.0 Release Candidate 1
2020-13-11

This is a release candidate intended for testing and does not represent the final version. It is not fully tested and is missing a few elements. Use at your own risk. The manual is missing some content as well. Please read the readme for guidance.

Contents

What is progs_dump?

progs_dump is a development kit for id software's Quake. Its purpose is to give mappers more creative options and "quality-of-life" improvements over the original "vanilla" version of the game. At the same time, progs_dump tries to retain the look and feel of the original as much as possible. The progs_dump dev kit has dozens of unique and powerful features that are explained in this manual and in the included sample maps.

The devkit consists of two elements:

First, there's the progs_dump "mod" folder. This holds all the sample maps, mapping assets, source code and the documentation you are reading now.

The second element is the *my_mod* folder inside the *mod_template.zip* file. This is a streamlined version of progs_dump with just the assets you need to make your mod.

The workflow is simple:

Use the progs_dump mod as a learning tool, then create your own Quake mod with the *my_mod* folder as a base.

Your project should be released as a stand-alone mod and installed into its own folder in the Quake directory <u>NOT</u> in progs_dump.

What can you do with progs_dump?

The devkit started as a simple project to add custom sounds and models to the game but has grown into a powerful toolkit aimed at beginner and intermediate mappers. Most features are from existing mods both old and recent, but there is a lot of new code in progs_dump as well.

Features include:

Monster Customization:

Add custom sounds, skins, models, health, damage, names, obituaries and much more without any coding required. This includes customizing monsters' heads, gibs and projectiles. Grunts, Enforcers and Ogres have multiple new attack options and we've added killable, gibbable versions of the original Quake bosses as well. Even Rotfish will gib now. No other Quake mod allows this amount of customization in such an easy way.

Quality-of-Life Features:

Trigger spawned monsters, continuous monster spawning and random monster spawning. Respawn items and suspend them in the air. Add custom backpack pickups, drag and drop gore decorations and create visual effects like explosions and lightning effects. Custom models, sprites and sound effects. Multiple targets and targetnames, dormant triggers, enhanced platforms and more.

Unique new features like trigger_look, sight_trigger, pain_target, Doom style door behaviors and item_key_custom.

Mission pack additions like custom gravity triggers, rotating entities, candles and elevators.

Enhanced teleporters with random destinations, monster only options, changeable destinations and more.

Popular requests like ladders, cutscenes and breakables are included. In fact, there are two styles of breakable. An "easy" method and a completely "custom" method.

Collisions for most objects are diabled in noclip.

Bug fixes:

Traditional fixes to the Shambler's collision during combat, the Rotfish "kill count" bug, door unlock sounds and many more "under-the-hood" code fixes.

Acknowledgements

Thanks to the following people for their assistance and generosity. I could not have compiled this mod without their guidance either directly, through tutorials, mapping, code comments or forum posts:

Qmaster, RennyC, c0burn, ydrol, Preach, Joshua Skelton, Spike, Khreathor, Shamblernaut, ericw, metlslime, necros, negke, Baker, sock, G1ftmacher, NewHouse, Joel B, iJed, ionous, McLogenog, Danz, whirledtsar, therektafire, thoth, vbs, Lunaran, Voidforce, NullPointPaladin, ZungryWare, Twitchy, Paril, fairweather, shinola, SunkPer, KONair, xaGe, hemebond and many others on the Quake Mapping Discord and on func_msgboard.

I also want to thank Pinchy, Mugwump, Len and PalmliX for their help with bug hunting. I apologize if I am forgetting anyone else who assisted.

A special thank you to Ian “iw” Walshaw for his excellent coding, detailed comments and for fixing a massive list of bugs starting with version 1.1.1

Simply put, progs_dump would not have been as stable and ambitious without him.

You can inquire about progs_dump on the Quake Mapping Discord.

Check out dumptruck’s Quake videos including the progs_dump how-to playlist, on YouTube.

A note about key | value pairs. You may notice some strange naming of key fields in progs_dump. For example, to select spawn silent for a monster you set the *wait* key. Another example is the *currentammo* in a lightning trail relay representing damage. This is because some of the code in progs_dump is from a time when the memory requirements for Quake were high. Programmers would reuse certain keys to save memory, as each key field took up precious RAM. 20 years later we have plenty of processing and RAM to dispose of. As a result, some of the newer keys are a bit more intuitive.

Installation

<u>Do not copy new versions of progs_dump over existing installs</u>. It's always best to make a new folder and move any work-in-progress maps and assets there.

1. Unzip the progs_dump archive into your Quake folder. This will create a *pd_120* folder inside your Quake directory. This directory will be a learning tool and reference for the features of the dev kit. Play it like any other Quake mod using the start map to explore a hub with sample maps.

   The development folder contains the FGD and DEF files that allow JACK, TrenchBroom and other editors to use the features of the devkit. Please refer to your map editor documentation for information on how to load mods and FGD files. In addition, there is a wad file that you can use to load the textures used in the sample maps. The QuakeC source code is included as well.

   Please read *progs_dump-1.2.0-README.txt* for important info and last minute changes.

2. When you are ready to create your own mod, unzip the *mod_template.zip* into your Quake directory and rename the *my_mod* folder to the name of your mod (with no spaces). This folder is a stripped down version of progs_dump without the sample maps and other files. However, the new models, sounds, sprites, progs.dat and QuakeC source code are included.

3. When you are ready to release your mod, zip up your mod directory and please include the progs_dump-devkit-readme.txt file and the QuakeC source folders in your release. If you modify the QuakeC code, make sure and include that in your zip instead of the original QuakeC files. **Remove any cfg files, screenshots or save game files before zipping up your mod folder.**

4. Please **do not** include the original progs_dump sample maps in your mod.

5. Make sure and share your work on the Quake Mapping Discord in the #progs_dump channel! https://discordapp.com/invite/j5xh8QT

6. Good luck and happy modding!

Monsters

Some of the new features in progs_dump can drastically change the way the game plays. Always use proper game design principles and communicate to the player that this is something different than they might expect.

Version 1.2.0 of progs_dump heavily focuses on monster customization. There are a lot of new key | values that can seem overwhelming at first glance. To make things easier to digest, the new features can be broken down into three categories: behavior mods, models and sounds.

But first, the most requested feature of any general purpose Quake mod is trigger spawned monsters. This makes spawning monsters much easier. All you need to do is select the spawnflag and target the monster with a trigger when you want them to appear. See more spawnflag details below.

Behavior Modifiers

Key	**Details**
berserk	Skips certain pain animations similar to skill 3 Makes a semi-nightmare monster! • 0 Off (Default) • 1 Berserk (skip pain animations) Excludes Bosses, Zombies and Spawn.
damage_mod	Multiply all damage from this monster by this number (e.g. 4 = Quad damage)
delay	The *delay* key allows you to add a custom delay to each trigger spawn. Normally, multiple targets will spawn simultaneously. If you want to stagger the time each monster enters the map, add a delay. Use the drop down menu to select some predefined values or enter a custom value in seconds if you need a specific time set.
effects	Add a visual effect to an entity • 0 None (Default) • 1 Brightfield (yellow particles) • 4 Bright light • 8 Dim light
health	Monsters can have custom *health* levels.

keep_ammo	Stop Ogres, Grunts and Enforcers from dropping backpack ammo by setting to 1.
obit_name	Custom description of WHO killed the player. When used with obit_method, this will set part of the text for a custom obituary. e.g. a Super Soldier!
obit_method	Custom description of HOW this monster killed the player. When used with obit_name, will set part of the text for a custom obituary. e.g. eviscerated - If empty, defaults to killed. Using the examples above, the obituary would read: Player was eviscerated by a Super Soldier!
pain_target	see description below
pain_threshold	see description below
sight_trigger	Set *sight_trigger* to 1 to have monsters trigger targets when they see the player. This means they can not trigger an event upon their death. You can still use *pain_targets*.
spawn_angry	Only when trigger spawned: • 0 default behavior - not angry • 1 set to 1 to spawn angry at player
wait	Play an effect when trigger spawned? • 0 Teleport Effects (Default) • 1 Spawn Silently

Custom Monster Models

In Quake it's easy to replace a monster model. Simply place a different model in the correct directory with the same name as the monster you want to replace. The drawback is that every iteration of that monster will have that model in the game.

In progs_dump you can use the key | value pairs below to load any compatible model for a given monster. This allows you to mix and match monster types in the same project. Using this with custom health, damage, sounds and other behaviors allows you to have a variety of monsters in your map without any coding required.

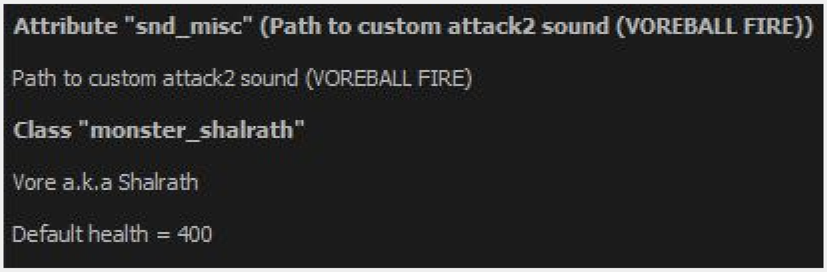
Not all of these key | values appear on each monster. For example, the Spawn has no head and therefore, no head model!

Key	**Details**
mdl_body	Path to custom body model e.g. dev/super_soldier.mdl
mdl_head	Path to custom head model
mdl_proj	Path to custom projectile model
skin	Skin index number of the body model if multiple built-in skins are included. Defaults to 0
skin_head	Skin index number for head model if multiple built-in skins are included. Defaults to 0
skin_proj	Skin index number for projectile model if multiple built-in skins are included. Defaults to 0
mdl_gib1	Path to custom 1st gib model
mdl_gib2	Path to custom 2nd gib model
mdl_gib3	Path to custom 3rd gib model

You can use Quake compatible sprites and BSPs in addition to models!

Custom Monster Sounds

As with models, progs_dump allows you to replace the sounds a monster makes with custom audio. Most, but not all sounds can be replaced. Each monster entry in the FGD and DEF details any sounds that are not obvious with a hint in ALL CAPS. e.g. snd_misc2 will replace the Enforcer's “HALT!” sight sound. You can see tips on how to “audition” existing sounds and models in the Custom Monster Example section below.



Attribute "snd_misc" (Path to custom attack2 sound (VOREBALL FIRE))

Path to custom attack2 sound (VOREBALL FIRE)

Class "monster_shalrath"

Vore a.k.a Shalrath

Default health = 400

Custom sound files used with these entities must be in the SOUND folder of your mod (or a sub folder under that SOUND folder.) There is no need to add “sound” in the path. (e.g. *boss2/sight.wav)* Most Quake source ports require a mono sound file for custom sounds. Do not use stereo files in your mod except for music.

Key	**Details**
snd_attack	Path to custom attack sound.
snd_death	Path to custom death sound.
snd_hit	Path to custom hit sound. e.g. laser hits wall
snd_idle	Path to custom idle sound.
snd_misc	Path to custom sound. Context will be different for various monsters. e.g. Enforcer’s “FREEZE” sight sound.
snd_misc1	same as above
snd_misc2	same as above
snd_misc3	same as above
snd_move	Path to custom sound for Chthon rising from lava.
snd_pain	Path to custom pain sound.
snd_sight	Path to custom sight sound.

pain_threshold
pain_target

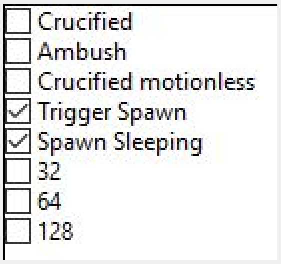
When a monster's health drops below it's *pain_threshold*, it's *pain_targets* are triggered. You can use this to call in reinforcements mid-battle or spawn items or fire other triggers when a monster reaches a certain level of health. You can also target things upon a monster's death, as always. Default values for monster health have been added to the FGD for reference.

monster_boss2
monster_oldone2

These are killable variants of the original boss monsters. On Skill 1 (Easy) these both have 1000 HP. On Normal, Hard and Nightmare the HP is set to 3000. You can also set a custom *heath* value, as with other monsters. Upon death, Shub will always gib but Chthon will only gib if his HP drops below -50 with one hit (Quad Damage, etc.)

Enhanced Zombies

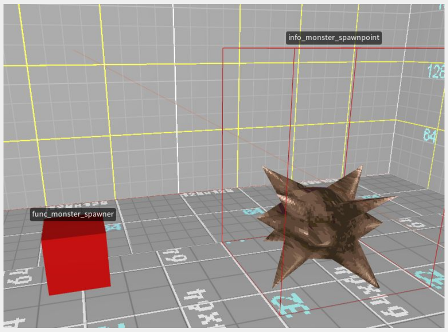
Zombies have more options in progs_dump. First off, there are motionless, silent versions of the crucified "decorative" zombie. You can also spawn a *sleeping* zombie that will not awaken until triggered. You must target these zombies if the *Spawn Sleeping* spawnflag is set. If you trigger spawn a sleeping zombie into a map, you will have to target them a second time to "wake" them up. You can see examples of the new features in the pd_zombies sample map. Spawnflag examples:



☐ Crucified
☐ Ambush
☐ Crucified motionless
☑ Trigger Spawn
☑ Spawn Sleeping
☐ 32
☐ 64
☐ 128

func_monster_spawner

Spawns standard id1 monsters to targeted *info_monster_spawnpoint*. The monster total is updated upon a spawn. Choose *Style* via dropdown, *Style2* set to a value of 1 overrides *Style* and chooses a random monster (excluding Zombies and Vores). *Count* is how many monsters to spawn in (default is 5). *Wait* is the default time between spawns (default is 5 seconds; 12 for Zombies and Vores to reduce telefrags). Can set *Berserk* to 1 to skip most pain animations. Can only use default health, models and sounds.



info_monster_spawnpoint

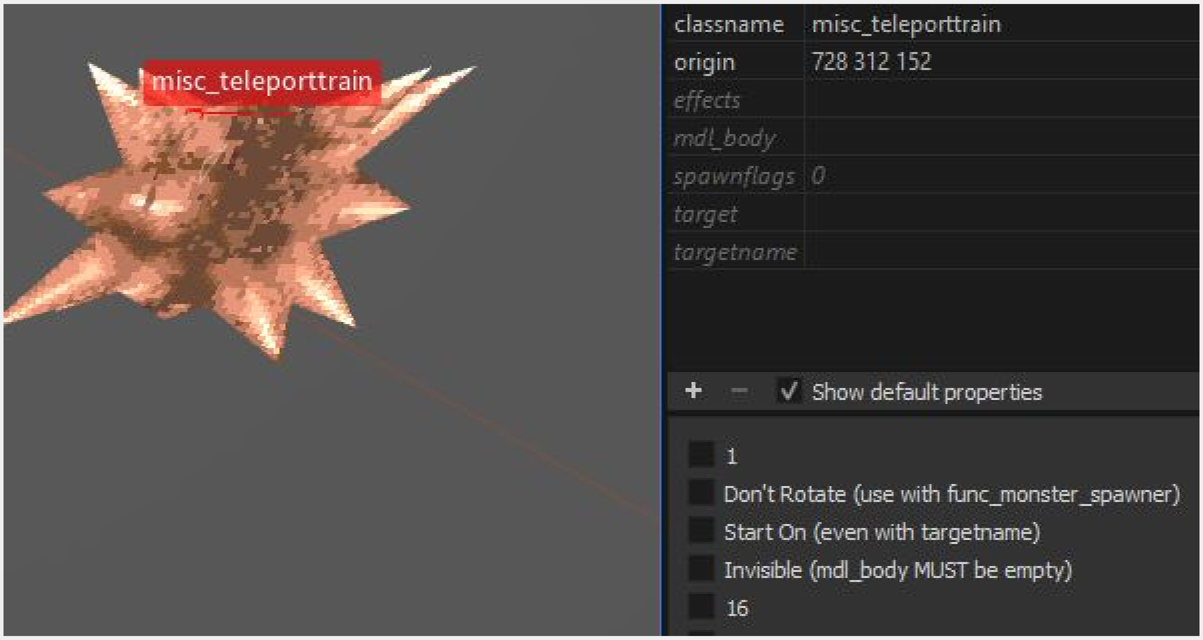
Destination for func_monster_spawner. You can also use a misc_teleporttrain for a moving spawn point. See below.

misc_teleporttrain

This was used for the final boss level in the original game. In progs_dump you can use this as a moving decoration with a custom model or even target it as a spawn point for a *func_monster_spawner*. Make sure and select the *Don't Rotate* spawnflag in this case, or you'll experience a pretty hilarious game breaking effect!

You set this up like a *func_train* using *path_corners*. By default, it will move automatically between path corners upon map load. However, you can have it wait to move by giving it a *targetname*. If you need to target it as a spawner and want it to move on map load, use the *Start On* spawnflag. Here's a video tutorial on how to use path_corners in Quake.

You can add *effects* using the dropdown, use a custom model using the *mdl_body* keyvalue and even make it *Invisible* with spawnflag 8.



classname	misc_teleporttrain
origin	728 312 152
effects	
mdl_body	
spawnflags	*0*
target	
targetname	

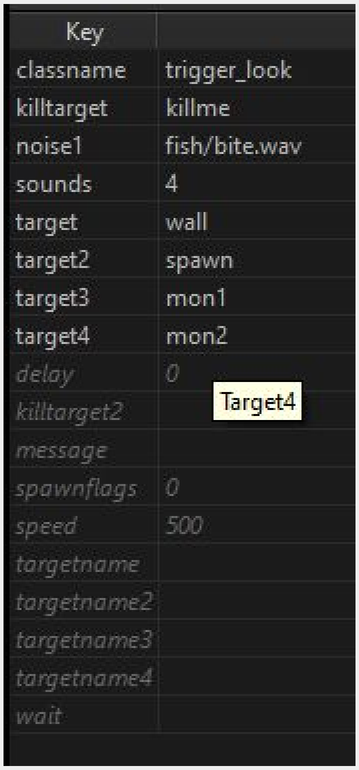
☑ Show default properties

☐ 1
☐ Don't Rotate (use with func_monster_spawner)
☐ Start On (even with targetname)
☐ Invisible (mdl_body MUST be empty)
☐ 16

Multiple targets, targetnames and killtargets

Most entities can now trigger up to four separate targets at once (target, target2, target3 and target4). They can also have multiple targetnames (targetname, targetname2, targetname3 and targetname4). Mappers can also create setups with killtarget and killtarget2. In addition, mappers can use target and killtarget in the same entity. This is not possible in vanilla Quake.

Multiple triggers can be used in nearly any combination or order. For example: target3 can trigger targetname2 in a different entity.



Key	
classname	trigger_look
killtarget	killme
noise1	fish/bite.wav
sounds	4
target	wall
target2	spawn
target3	mon1
target4	mon2
delay	0
killtarget2	
message	
spawnflags	0
speed	500
targetname	
targetname2	
targetname3	
targetname4	
wait	

IMPORTANT: When using path corners or other similar entities, use the primary target and targetname fields for navigation only. The additional numbered fields may not function as expected in these cases. The Quoth mod has the same feature and the rule of thumb there applies here. As Preach states on the Quoth tutorial site: *A recommended structure is to use the original targetname field to give entities unique identifiers, and use the remaining fields for group triggers.*

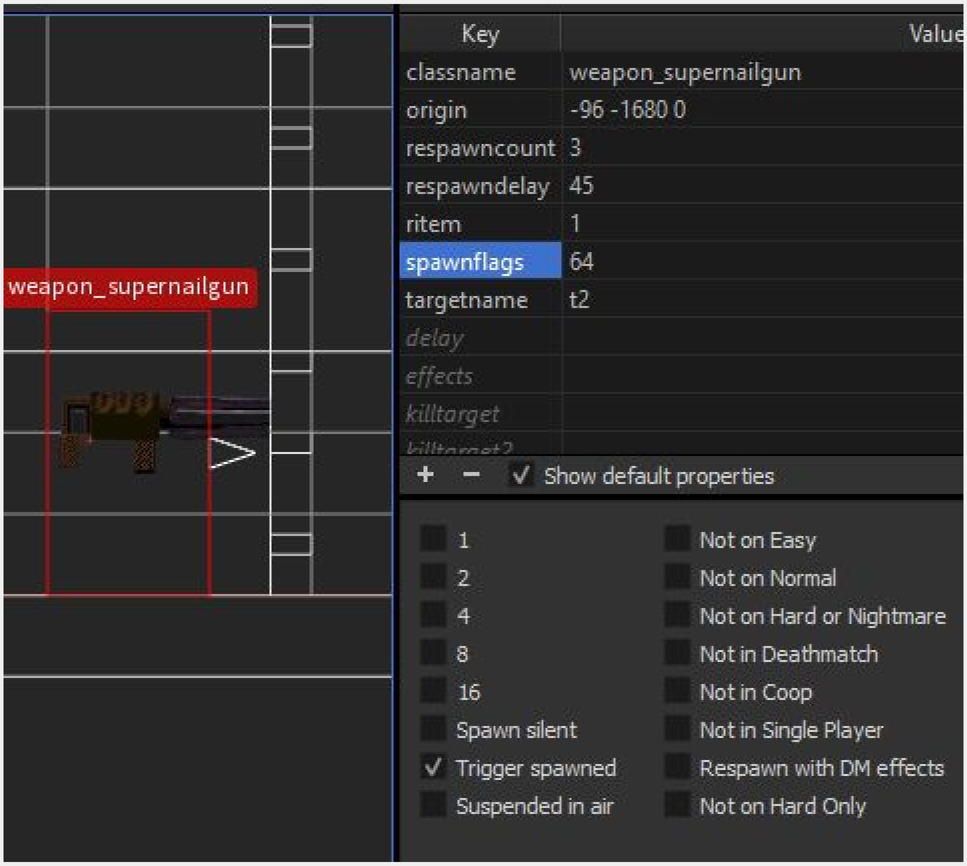
Items

Most items have enhanced capabilities in progs_dump. This includes ammo, weapons, keys and power-ups. Items can be suspended in mid-air via a spawnflag or trigger spawned just like monsters. Set a targetname for the item and select the *Trigger Spawned* spawnflag. Like monsters, they can spawn silently or with the “tfog” teleport visual and sound effects.

The *effects* key allows you to add some built-in visual effects to items. A drop down is available with brightfield (yellow particles), bright light and dim light.

Items can also be set to respawn. Setting the *ritem* key value to 1 will cause the items to respawn. Items will respawn based on the default time settings for a deathmatch game. You can set a custom respawn time using the *respawndelay* key and control how many times an item respawns with the *respawncount* key. By default items will display the “tfog” effects when respawning. You can mimic deathmatch respawns with the *Respawn with DM Effect* spawnflag. This skips the “tfog” effect and plays a more subtle sound effect.

In the example below, the super nailgun is trigger spawned when the player presses a button targeting “t2”. After the player picks up the weapon it will respawn in 45 seconds but only 3 times.



Key	Value
classname	weapon_supernailgun
origin	-96 -1680 0
respawncount	3
respawndelay	45
ritem	1
spawnflags	64
targetname	t2
delay	
effects	
killtarget	

☑ Show default properties

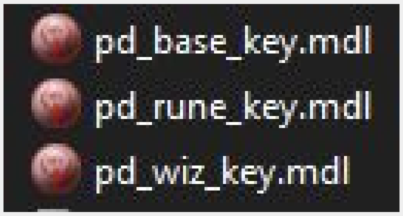
☐ 1
☐ 2
☐ 4
☐ 8
☐ 16
☐ Spawn silent
☑ Trigger spawned
☐ Suspended in air
☐ Not on Easy
☐ Not on Normal
☐ Not on Hard or Nightmare
☐ Not in Deathmatch
☐ Not in Coop
☐ Not in Single Player
☐ Respawn with DM effects
☐ Not on Hard Only

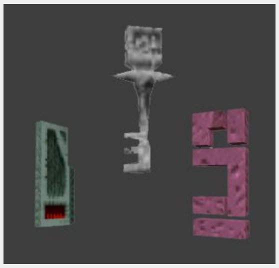
item_key_custom

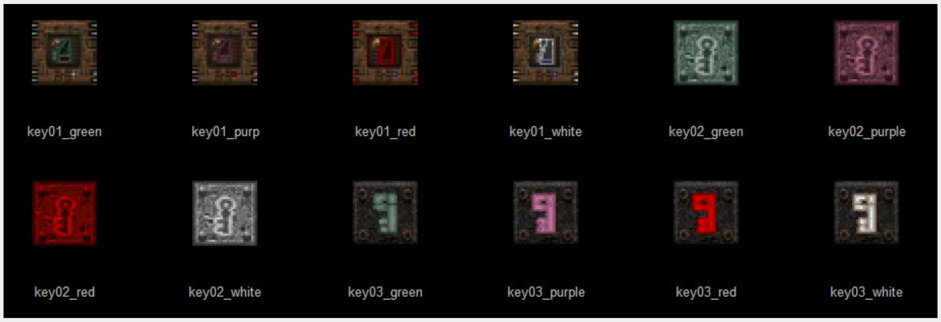
This allows mappers to use any Quake compatible model, sprite or BSP as a key. We’ve also included a new key model with different variations outlined below.

keyname: name of the key, e.g. 'bronze key' (required), *mdl*: model file path (required) *noise*: sound file for the pickup sound (default is per worldtype), *skin*: skin index (default 0), *frame* (default 0): display this single frame of the model, if animated. NOTE: The key will not display any animation.

Three new models based on id’s original keys are included. One for each worldtype: base, runic and wizard. Each of these has four color variations, also referred to as their skin index: jade (green, skin 0), runic (magenta, skin 1), blood (red, skin 2) and alabaster (gray, skin 3). The development/wads folder has a small wa with textures for use with the different styles seen below.


pd_base_key.mdl
pd_rune_key.mdl
pd_wiz_key.mdl





The *keyname* value is used both for the pickup message and to associate the key with the entity that it unlocks. To make a *func_door* or *trigger_usekey* require this key, set the *keyname* value of that entity so that it matches the *keyname* value of the *item_key_custom* entity.

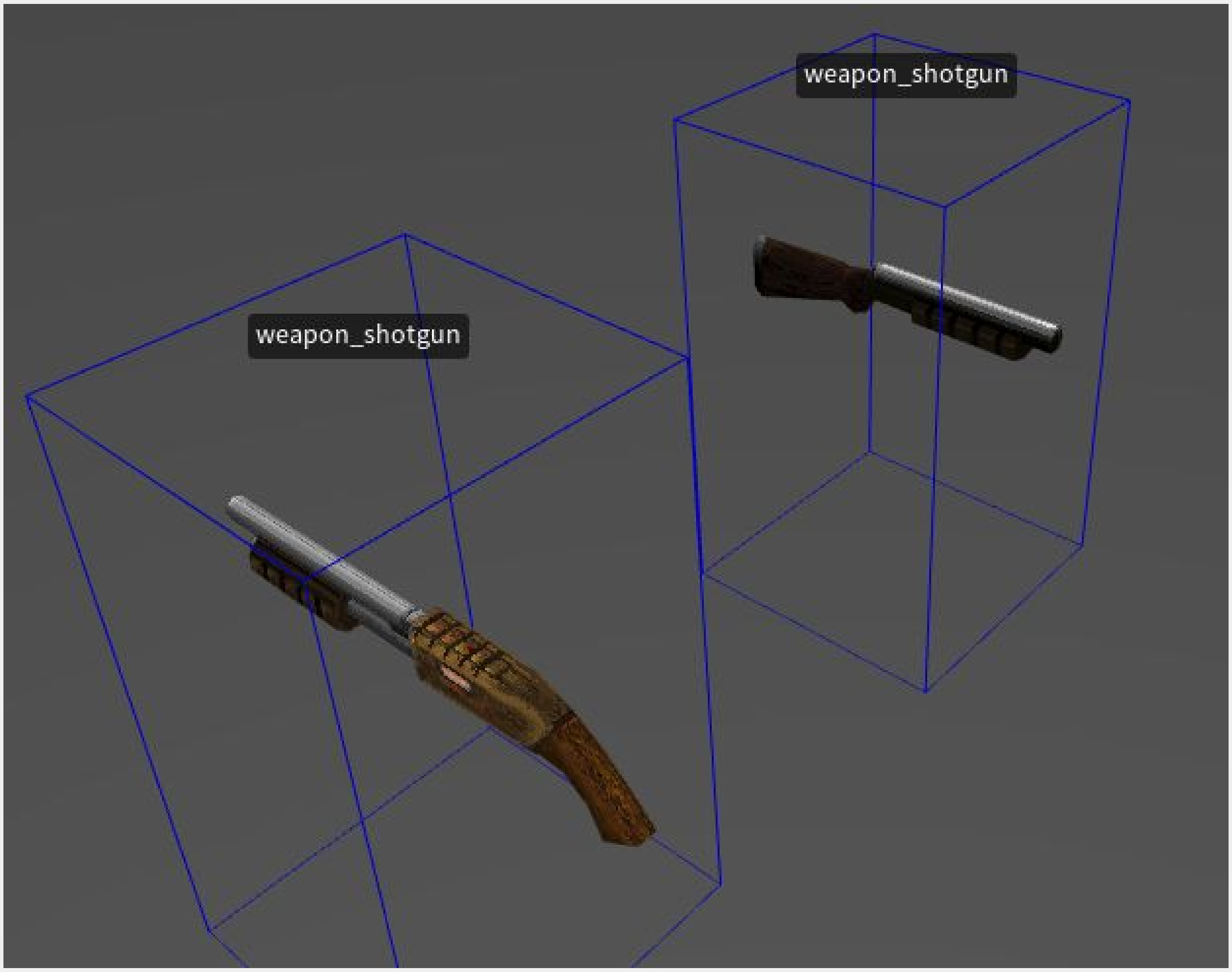
If different *item_key_custom* entities have the same *keyname* value, they will be treated as different copies of the same key and may be used interchangeably.

A map may have a maximum of 23 unique keyname values across all entities.

The behavior of an *item_key_custom* should be as the player expects (based on the behavior of the silver and gold keys), except for the fact that it will not appear as an icon in the player's status bar when picked up. This is a limitation of the engine. Finally, there is a sample map called *pd_keys* you can review.

weapon_shotgun

This is a pickup model that should be used when you want a player to spawn with only an axe and then later get the shotgun: (*trigger_take_weapon* or *reset_items 2* in worldspawn). There are two models to choose from. *Spawnflag* 2 (the default) selects an unused “classic look” model from Rubicon 2 by metlslime. *Spawnflag* 4 is an alternative from Slapmap and has been used in a few mods.



Custom Sounds

Custom sound files used with these entities must be in the SOUND folder of your mod (or a sub folder under that SOUND folder.) There is no need to add “sound” in the “noise” path. (e.g. *boss2/sight.wav)* Most Quake source ports require a mono sound file for custom sounds. Do not use stereo files in your mod except for music.

Attenuation

A note on the “speed” key (a.k.a attenuation factor) in sound entities. Attenuation in Quake means the reduction of a sound over a distance. Here’s a table of what the different speed keys mean in progs_dump.

Speed	QuakeC name	Attenuation effect
-1	ATTN_NONE	heard everywhere
1	ATTN_NORM	fades to zero at 1000 units
2	ATTN_IDLE	fades to zero at 512 units
3	ATTN_STATIC	fades to zero at 341 units

play_sound_tiggered

Play a sound when triggered. Most of these key / value pairs can be left to their defaults. Can be looping or a “one off” sound. NOTE: Looping sounds that are triggered ON will NOT play after the player loads a saved game. They will have to be triggered OFF then ON again. Also, you may encounter problems triggering sounds that are far away from the player. If you do, move the trigger closer.

Key	**Details**
toggle (spawnflag)	sound can be stopped and started when triggered
volume	how loud (1 default full volume)
noise	path of the sound to play (e.g. *blob/sight1.wav*)
impulse	sound channel 0-7 (0 automatic is default)
speed	attenuation factor (default recommended)

play_sound

Plays a “one off,” non-looped sound at a random interval. Like thunder or a monster sound. IMPORTANT: Do NOT use looped sounds with this entity. For looped sounds see *ambient_general* below. Check out this video tutorial on creating looping sounds for Quake.

Key	Details
volume	how loud (1 is default full volume)
noise	path of the sound to play (e.g. *boss2/sight.wav*)
wait	random time between sounds (default 20)
delay	minimum delay between sounds (default 2)
impulse	sound channel 0-7 (0 automatic is default)
speed	attenuation factor

ambient_general

Plays a custom looped sound. Cannot be toggled off or triggered. *noise* = path of the sound to play (e.g. *ambience/suck1.wav*)

ambient_thunder

Originally unused in the game. Plays the sound of thunder at a random interval. You only need one of these in your map. It will play everywhere. If you want it to play locally instead, use a play_sound with a different speed setting. The path for the sound is: ambience/thunder1.wav

ambient_water1

Swirling water sound effect. Usually this is added automatically to maps with water when you run VIS. If you want to place these in your map by hand, you can run VIS with the -noambient command line switch.

ambient_wind2

Howling wind sound effect. Usually this is added automatically to outdoor sections of maps with sky textures. If you want to place these in your map by hand, you can run VIS with the -noambient command line switch.

ambient_fire

This is a simple looping sound from the torches. Use this if you are using custom fire sprites or models. This is the same effect as FireAmbient in earlier versions of progs_dump.

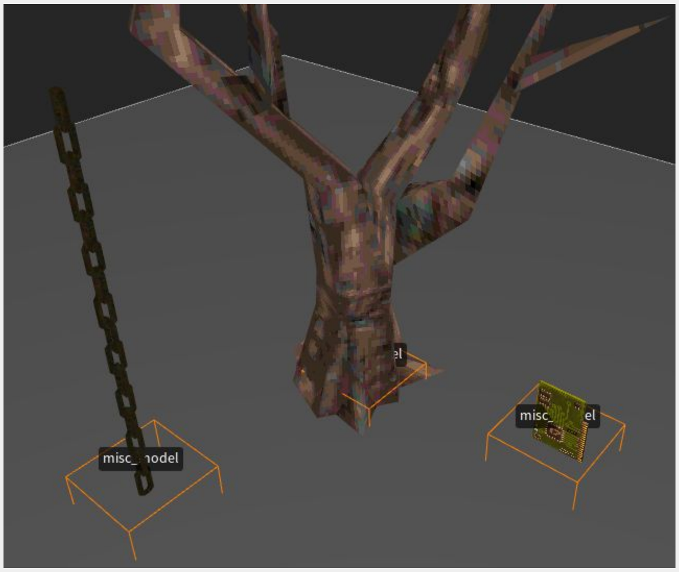
Custom Models and Sprites

misc_model

A point entity for displaying models and sprites. A frame range can be given to animate the model. NOTE: Sprites will not display in mapping programs but models should in TrenchBroom if the path is set correctly. Here’s a link to an older progs_dump video that has some more info and tips for using *misc_models* in your mod.

Key	**Details**
mdl	The model to display. Can be of type mdl, bsp, or spr.
frame	Single frame to display. Can also be used to offset the animation starting frame for variations (e.g. when using fire sprites)
first_frame	The starting frame of the animation.
last_frame	The last frame of the animation.
speed	The frames per second of the model’s animation. Divide 1 by the fps for this value. (Default 10)
angles	pitch roll yaw (up/down, angle, tilt left/right)

IMPORTANT: Set the *angle* (no S) value to 0 if using *angles* (with S) key to rotate mdls (see gib_ section for more info or the video linked above.)



Enhanced Triggers

This mod has some enhancements to triggers that allow some to start off or even toggled off and on. See the table below for more information.

is_waiting

If this value is set to 1, certain triggers will do nothing until another trigger activates it. The FGD provides a dropdown selection or you can enter the value by hand. The following table shows which triggers use *is_waiting 1*:

trigger	**is_waiting (start off)**
trigger_once	yes
trigger_multiple	yes
trigger_teleport*	yes (use targetname2)
trigger_changelevel	yes

NOTE: *In order to use *is_waiting* on a trigger_teleport, make sure and use *targetname2* to “wake it up” instead of *targetname*.

trigger_changelevel

On triggers that point to a hub or start map, the *Use info_player2_start* spawnflag will spawn the player on the info_player_start2 entity when the map changes. NOTE: **You’ll need an info_player_start2 on the map you are changing to!** Use this to skip skill selection when completing an episode as in the original game. Or you can return the player to a different part of a hub map.

trigger_heal

When a player enters this trigger they are healed at a rate of 5 HP per second (by default.)

Key	**Details**
heal_amount	Healing per second (default is 5 HP)
health_max	The upper limit for healing (default 100, max 250)
spawnflags	**Start on -** Start on if using targetname. Only needed if triggered by something other than touching. **Player only, Monsters only**
noise	path to custom healing sound

trigger_look

John Romero wanted this in the original game and thanks to NullPointPaladin, it's finally here! This will trigger when a player is within the brush trigger and looks *directly* at a target entity. Use the first *target* key for the "looked at entity" and use the subsequent targets (2-4) to trigger other events. See sample map pd_cutscenes for one setup using a skip textured *func_wall* (more on this below.).

The entity targeted by *trigger_look* needs to be an entity with a bounding box (bbox), collision (solid), be targetable and visible (not moved out of bounds). A *func_wall* is a great example although entities like gibs, lasers, dead monsters, or others can be targeted as long as they are "solid" by using the *solid* spawnflag where applicable.

A *func_illusionary* doesn't work because it doesn't have a bbox that interacts with the player. A *func_detail* is ignored, as it cannot be targeted. NOTE: It is not recommended that you use items or monsters as the trigger as they can be removed and /or lose their bbox upon death.

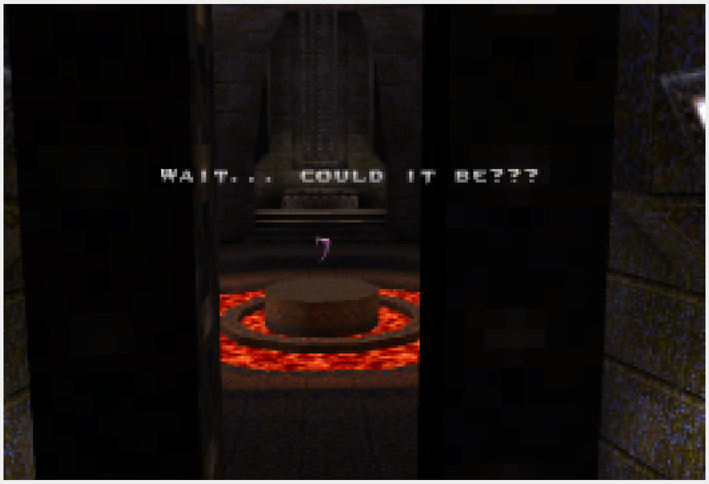
If you need an invisible brush, use a skip textured func_wall. If you need to avoid triggering a second time use a *trigger_relay* to killtarget the trigger_look's target or the *trigger_look* itself. Also, the default distance from the player to the look target is 500 Quake units. If the target needs to be farther away or closer, set the *speed* key to the proper distance. Use a brush as a "ruler" to measure the distance in TrenchBroom if needed. You can set a custom sound by setting sounds to 4 and adding a path to the sound in the *noise1* value.

This will **not** work or is not recommended with the following brushes: *func_illusionary, any func detail variant, func_group, func_particle_field, non-solid func_laser, func_bossgate* or *func_togglewall*.

Brushes textured in "clip" will not work.

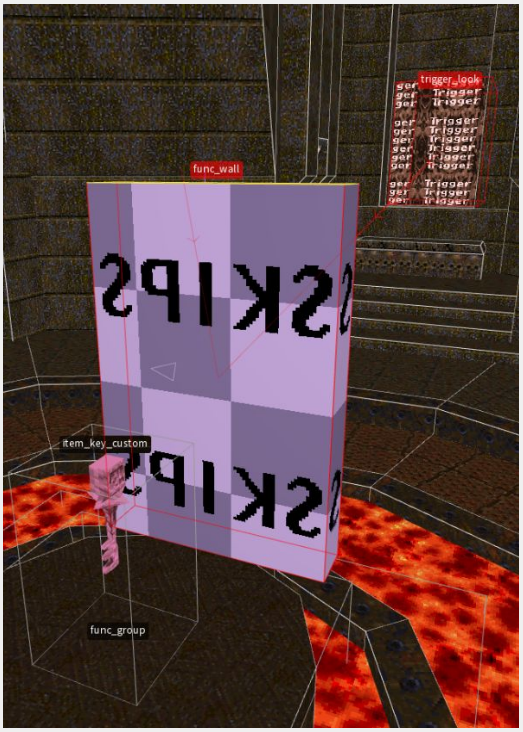
Key	Details
speed	Distance from player to search for trigger, adjust if the target is too far from the trigger (default 500 units)
wait	Time between re-triggering (default 0)
sounds	0-3 are standard Quake trigger choices, 4 allows a custom sound, requires a path set in noise1 key
noise1	Path to custom sound. Use with sounds key set to 4 (e.g. fish/bite.wav)

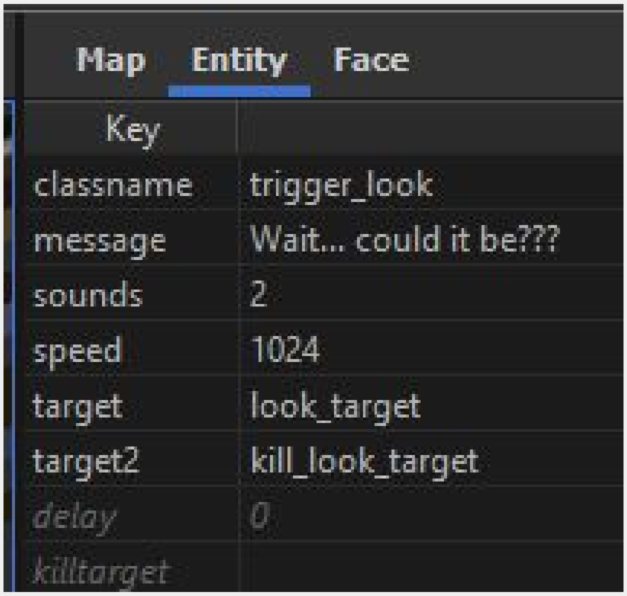
You can see *trigger_look* in action near the beginning of the sample map *pd_cutscenes*. When the player looks at the rune a message plays. Yes, you could do this with a *trigger_once* using an *angle* key, but using this method it will *only* trigger if the player looks directly at the platform.



WAIT... COULD IT BE???

In this example, we used a *func_wall* textured in skip. The reason this is used instead of the rune is that the rune's bounding box is smaller *and* farther away, making it harder to trigger correctly during testing.





Key	
classname	trigger_look
message	Wait... could it be???
sounds	2
speed	1024
target	look_target
target2	kill_look_target
delay	0
killtarget	

We used a larger brush in this case and killtargeted it before the player could collide with it. If the setup is in a smaller area you will likely be able to use the entity itself. Test early! Test often! One tip is to have a backup method of targeting entities in case the player doesn't look at what you need them to trigger! In this map, we used the button nearby to killtarget the func_wall just in case the player missed this moment in the "story."

trigger_push_custom

This can be used to create traps, jump pads, water currents and more.

If the *Start Off* spawnflag is set the entity will not trigger until targeted. This can be targeted and toggled off and on. If the *Silent* spawnflag is set it won't make the standard “windfly” sound. Use *Custom Noise* spawnflag and the noise key/value together to use a custom push sound. Custom sounds should be "one off” sounds NOT looping sounds. A good way to simulate a water current is to have the trigger_push_custom under the surface of your water brush by about 32 units. You can see an example in the pd_gallery.map

trigger_monster_jump

If the *Start Off* spawnflag is set the entity will not trigger until targeted. This can be targeted and toggled off and on. So monsters can be attacking from a distance and then be triggered to jump. NOTE: The way *trigger_monsterjump* works requires a monster to be “awake” and “angry” at the player before the jump is activated. Keep this in mind when using this new functionality.

trigger_take_weapon

This will remove the shotgun from the player’s inventory and all shells. Place this over an info_player_start to have the player start with only the axe… or use this trigger to surprise the player in some devious way. Make sure and place a weapon_shotgun in your map for the player to get eventually!

An alternative to this would be setting *reset_items* to 2 in your worldspawn if you want an “axe only” start.

trigger_setgravity

If the *Start Off* spawnflag is set the entity will not trigger until targeted. This trigger changes the gravity on a player or monster that touches it. The trigger itself can be toggled on and off.

NOTE: the amount of gravity can only be changed by touching *another* trigger_setgravity with a different setting. The *gravity* key defaults to 0 which is normal gravity. Lower numbers (e.g. 25) equal lower gravity. Setting 100 is normal gravity. Numbers above 100 will make the player “heavier”, i.e. harder to jump. If you want multiple *trigger_setgravity* triggers in one room or area, make sure the brushes are not touching each other. This can cause the triggers not to work properly.

trigger_shake

Earthquake trigger - shakes players in it's radius when active. Strength of tremor is greatest at the centre.

dmg is strength at center (default is 120.) *wait* duration of shake (default is 1.) *count* effect radius (default is 200.) *noise* path of sound to play when starting to shake. *noise1* path of sound to play when stopping. *targetname* must be triggered. The *VIEWONLY* spawnflag shakes the view, but player movement is not affected.

trigger_usekey

Variable sized single use trigger that requires a key to trigger targets. Must be targeted at one or more entities. Use the *message* key to create a custom message for this. e.g. “Bring the Gold Key here mortal!” This trigger cannot start off or be toggled. Setting *cnt* to 1 will not remove the key from the player’s inventory, which mimic’s the key behavior of Doom. Make sure and add this key | value to all doors and / or let the player know the default key behavior has changed. e.g. Perhaps a pickup message on the keys that reads: “This key works on many doors.”

trigger_void

Use this for a 'void' area. Removes monsters, ammo, etc... also kills players. Spawnflags can be used to protect players or monsters.

trigger_cdtrack

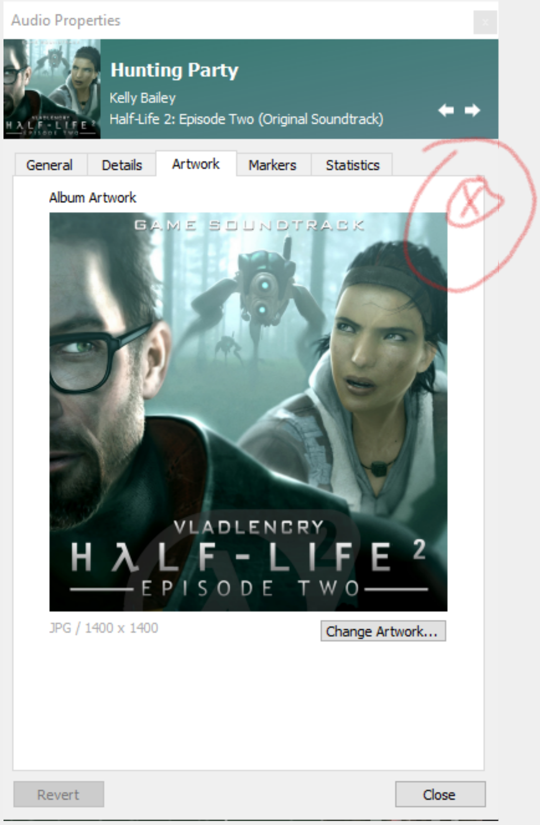
A point entity that changes the currently playing music track when triggered. The number of the track to play goes in the count key. e.g. 32 for track32.ogg See trigger_changemusic below for more information on formats and more. NOTE: the track number uses the count key here but trigger_changemusic uses the sound key for the same info.

trigger_changemusic

A trigger brush that changes the currently playing music track. The number of the track to play goes in the *sounds* key (just like worldspawn). Most Quake engines require the trackXX.xxx format. For example: track02.ogg or track98.mp3 Different Quake engines play different formats. This is why Quakespasm, Quakespasm-Spike and vkQuake are recommended, as they can play mp3, ogg, wav and FLAC formats. You can read more about formats here and much more detailed information about how to format music for Quake here.

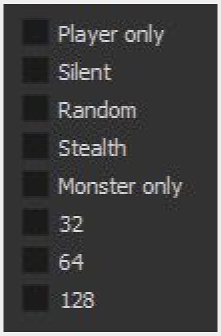
If you are adding custom music to your mod, we recommend you avoid using track01-11. track01 does not exist in Quake and track02-track11 are Quake's original tracks. Also note that FTEQW does not play mp3 and Mark V doesn't play ogg formats! Additionally, Mark V cannot play mp3s that contain embedded images. You can use a program like Ocenaudio to open the file and remove the image easily. In this case, you go to Audio Properties and click the small "x" in the upper right (not captured below) to remove the image then save the file.

NOTE: **We recommend including both mp3 and ogg formatted music tracks to ensure compatibility with various Quake engines.**



trigger_teleport
info_destination_random

progs_dump adds a number of new features to *trigger_teleport*. These are selected with new spawnflags. In the case of the *random* spawnflag, there is a new entity *info_teleport_random* which is a random destination marker for the trigger. *info_teleport_changedest* in another entity that can be used to tell a *trigger_teleport* to change its target value to a different destination. More info below.



☐ Player only
☐ Silent
☐ Random
☐ Stealth
☐ Monster only
☐ 32
☐ 64
☐ 128

Spawnflag	**Details**
Player Only	as in original Quake
Silent	no ambient sounds as in original Quake, use for monster closets and secrets etc.
Random	can teleport to random destination entities
Stealth	No particles or sound effects
Monster Only	can be used with other spawnflags: stealth, silent and random in any combination

Using the *random* spawnflag on *trigger_teleport* requires use of the *count* key and targeting a *info_teleport_random* (instead of the regular *info_teleport_destination*). This causes the teleporter to send the player to a random destination among the *info_teleport_random* markers in the level. In the *count* key, add a number equal to the number of *info_teleport_random* entities you placed.

info_teleport_changedest

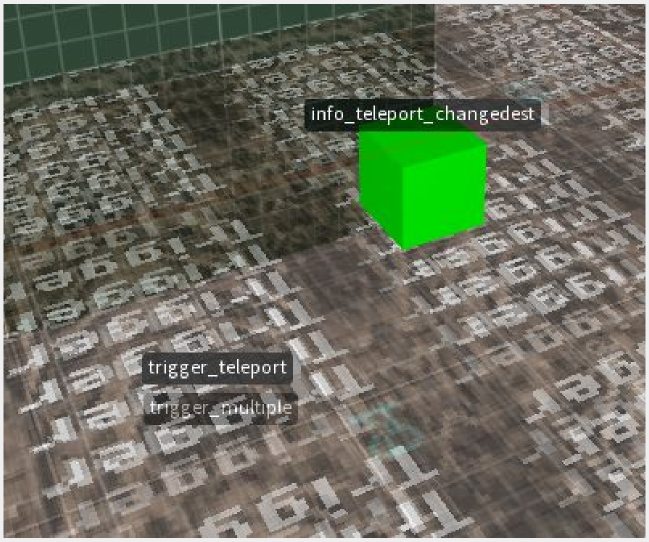
Allows a mapper to change the target of a teleport_trigger. Useful in maps where the player may fall into a void and the mapper wants to update where they "respawn" as they progress through the level. Could also be used for teleport puzzles and more. This requires the addition of a trigger_multiple in some setups.

Key	**Details**
target	the targetname of the trigger_teleport to change
message	new info_teleport_destination's targetname to switch to
targetname	name of this entity so we can trigger it with a trigger_mulitple or other

NOTE: The best way to understand how this works is to look at the sample map pd_teleport.

The basic workflow is to set up a *trigger_teleport* with a target as usual, then add a targetname. However, instead of using an *info_teleport destination*, the targetname will be referenced by an *info_teleport_changedest* point entity. The *info_teleport_changedest's target* key should match the targetname of the *trigger_teleport*. The *message* key is the “replacement” targetname for the trigger_teleport. Finally, the *targetname* is used to trigger this entity. This pattern can be duplicated multiple times to change the destination of a single *trigger_teleport*.

NOTE: when a *trigger_teleport* has a targetname it must be triggered to operate, so adding an overlapping *trigger_multiple* targeting the *trigger_teleport* will be necessary. Place the *trigger_multiple* 8 units inside the *trigger_teleport* and this will re-trigger it. If you need to “kill” the *trigger_teleport*, killtarget the *trigger_mutiple* and the teleporter will no longer work.



Enhanced Platforms

func_new_plat

This entity adds new capabilities to plats. It uses spawnflags to dramatically change the behavior of the entity. As with the standard plat, build your plat in the raised position so the entity will be lit correctly when you compile your map.

NOTE: You must use one of the following spawnflags with *func_new_plat*. Even though they use the same entity name, each spawnflag creates a very different plat.

Spawnflag 1: Setting the *Plat Start at Top* spawnflag creates a plat that starts at the top and when triggered, goes down, waits, then comes back up. *health* = number of seconds to wait (default 5)

Spawnflag 2: Setting *Toggle Plat* creates a plat that will change between the top and bottom each time it is triggered.

NOTE: You must use the *height* key when *Toggle Plat* is used. Use a negative height number to start the plat off in a lower position.

Spawnflag 16: *Plat2* creates a plat in the bottom position, just like the standard plat. If a plat2 is the target of a trigger, it will be disabled in the lowered position until it has been triggered. *Delay* is the time before the plat returns to its original position.

IMPORTANT: *Plat2* can be finicky so it's advised to create your plat the exact height you need it to travel (as opposed to having parts sticking into the ground or in hollow pockets below the plat for cosmetic reasons.) You can set the *height* to tweak the amount of lip needed. See *The Gallery* map for an example.

Elevators

func_elvtr_button

This entity turns a *func_new_plat* into a multi-floor elevator. Here are the steps to follow to create one. You can see this setup in the *pd_elevator* demo map:

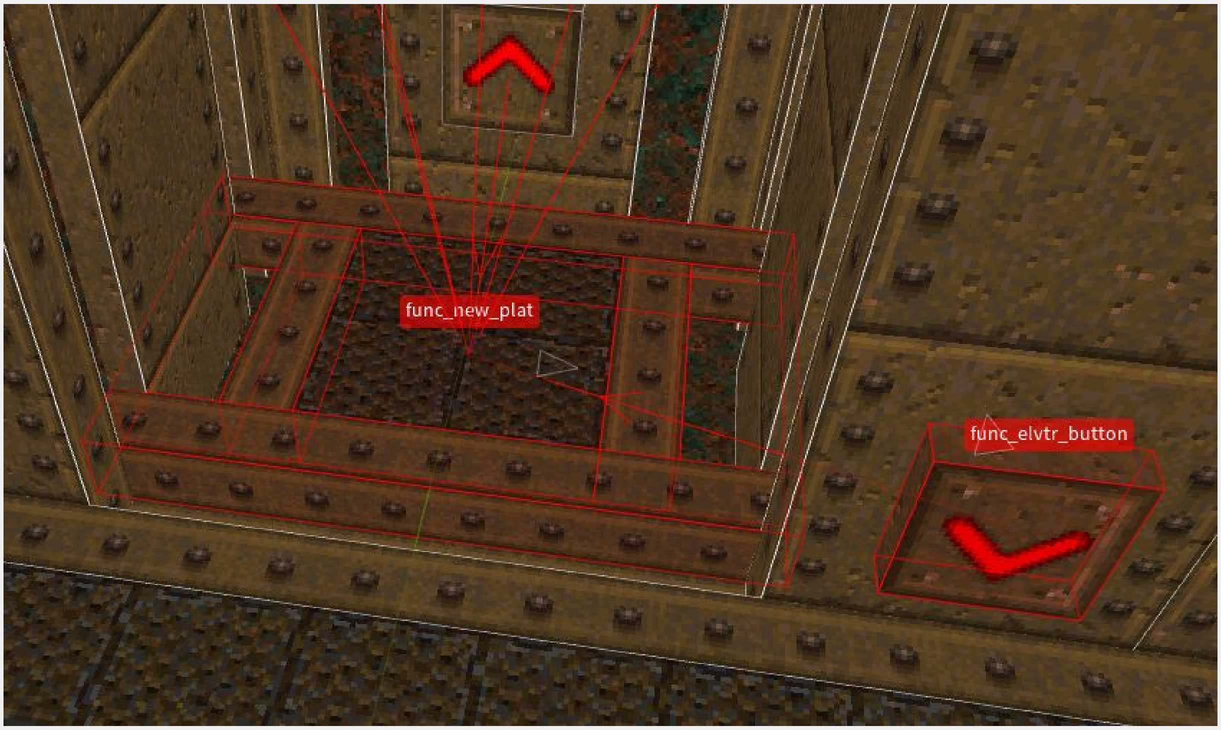
First, create a *func_new_plat*. Select the *Elevator* spawnflag (4). Set the *cnt* key to the number of floors (3 in the demo map). Next, set the *height* key to the vertical distance between floors (256 in the demo map). Then, give the func_new_plat a targetname.

By default, the elevator starts at the bottom floor, so that's where the func_new_plat needs to be positioned in the editor. Alternatively, if the mapper wants it to start at the top floor, they can manually position the bmodel at the top floor and set spawnflag (8) Elevator Start at Top.

With the func_new_plat done, create any number of func_elvtr_button entities. Make each func_elvtr_button target the func_new_plat. A func_elvtr_button is an "up" button by default. To make it a "down" button, use the spawnflag Down Button.

When the spawnflags are set to elevator the wait key on a func_new_plat is defaulted to zero. This means the player will be able to hit another button right away between floors as seen in the demo map. The wait key on a func_elvtr_button behaves just as a regular func_button would, controlling how long before you can hit a button each subsequent time.

NOTE: any func_elvtr_button will act as a "call" button if the elevator isn't already at that floor.



Misc Entities

trap_spikeshooter, trap_shooter, trap_shooter_switched

The original trap_spikeshooter shot only nails and lasers. All three of these entities can now shoot lavaballs, rockets, Voreballs, grenades or gibs. Set the spawnflag accordingly. Use the silent spawnflag if needed. Use the key *state* 0 for initially off, 1 initially on. (0 default) Refer to the table below for specifics on how to trigger these.

Entity	**Details**
trap_spikeshooter	use a trigger_mulitple to fire
trap_shooter	fires continuously (use killtarget to stop)
trap_shooter_switched	toggle on and off with triggers, buttons

func_counter

This is used to trigger things in a series. You can do some amazing new game play setups with these. Make sure and take some time to play with this one and take a look at the pd_counter sample map.

TOGGLE causes the counter to switch between an on and off state each time the counter is triggered. *LOOP* causes the counter to repeat infinitely. The count resets to zero after reaching the value in *count*. *STEP* causes the counter to only increment when triggered. Effectively, this turns the counter into a relay with counting abilities. *RESET* causes the counter to reset to 0 when restarted. *RANDOM* causes the counter to generate random values in the range 1 to *count* at the specified interval. *FINISHCOUNT* causes the counter to continue counting until it reaches *count* before shutting down even after being set to an off state. *START_ON* causes the counter to be on when the level starts. *count* specifies how many times to repeat the event. If *LOOP* is set, it specifies how high to count before resetting to zero. Default is 10. *wait* the length of time between each trigger event. Default is 1 second. *delay* how much time to wait before firing after being switched on. You can see *func_counter* in action when the sarcophagi burst open in pd_zombies.map and when used to animate the particle fields in pd_ladders.map.

func_oncount

For use as the target of a *func_counter*. When the counter reaches the value set by *count*, *func_oncount* triggers its targets. *count* specifies the value to trigger on. Default is 1. *delay* how much time to wait before firing after being triggered. You can see *func_oncount* in action when the sarcophagi burst open in pd_zombies.map and when used to animate the particle fields in pd_ladders.map.

func_door

Setting *cnt* to 1 will not remove keys from the player’s inventory, which mimic’s the key behavior of Doom. Make sure and add this key / value to all doors and let the player know the default key behavior has changed. e.g. Perhaps a pickup message on the key that reads: “This key works on many doors.” Normally, key doors remain opened once unlocked (wait -1) but a new spawnflag in 1.2.0 allows you to set *cnt* 1 key doors to reclose as they would in Doom.

func_explobox

An explosive brush entity. Works just like *misc_explobox* but is made from a brush you create as opposed to the default model.

func_fall

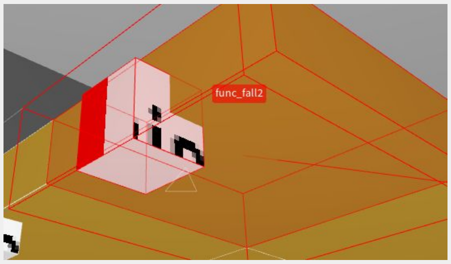
A brush that drops and fades away when touched and/or triggered. Add some spice to your jumping puzzles or other scripted sequences! Monsters will not trigger *func_fall* but will be gibbed if one falls on them. NOTE: When a *func_fall* brush touches another brush or entity it will stop, which can look odd in certain situations. *noise* = sound to play when triggered, the default is a switch sound. *wait* = wait this long before falling. Use the *DONT_FADE* spawnflag if desired.

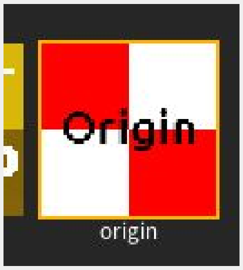
func_fall2

This is an enhanced version of *func_fall* that has different properties than the original and a lot more overall functionality. For example, *func_fall2* will not gib monsters but you can set them to trigger it unlike the original. These take a bit of set up, so refer to *pd_prefab_func_fall2.map* for more information and examples you can modify for your maps.

Key	**Details**
wait	how long until the brush begins falling
noise	the sound to make when touched / activated
noise2	the sound to make before it's removed, pain_finished of -1 disables noise2 as the object stays forever
cnt	0 is default behavior, 1 means collisions are disabled while falling, 2 turns the brush into a bouncing entit
pain_finished	default of 0.01, higher value has the object/brush fade out faster thus in turn affecting how long it stays. -1 stays forever
speed	speed as to how fast something falls per game frame, default is 10, higher values mean faster falling. Only for cnt of 1. Recommended to use lip for max fall speed with cnt 0 and 2 as they follow Quake's default gravity
lip	maximum fall speed that can be achieved, caps 'speed' variable. Default is -800
avelocity	brush spins when activated using X, Y, Z vector coordinates. *cnt* of 2 ignores avelocity. Use an origin brush at the center of your brush(es) for proper spin!
spawnflags	Player or Monster only flags

When using the avelocity key add an origin textured brush at the point you want the brush to rotate around. This brush must be part of the *func_fall2* brush entity. In TrenchBroom you would control click both brushes and then right click to make them a *func_fall2*.





func_togglewall

Creates an invisible wall that can be toggled on and off. *START_OFF* spawnflag means the wall doesn't block until triggered. *noise* is the sound to play when the wall is turned off. *noise1* is the sound to play when the wall is blocking. *dmg* is the amount of damage to cause when touched. You can see an example of this in the pd_ladders example map above the barred teleport area.

func_train

Just like the standard Quake train entity but with the *RETRIGGER* spawnflag set the train will stop at each path corner and wait to be retriggered before moving again. This will be great for more complicated lifts, doors and of course… trains. Set the *sounds* key to 3 to use custom sounds, then set noise3 as the start/stop sound and noise4 for the “in motion” sound.

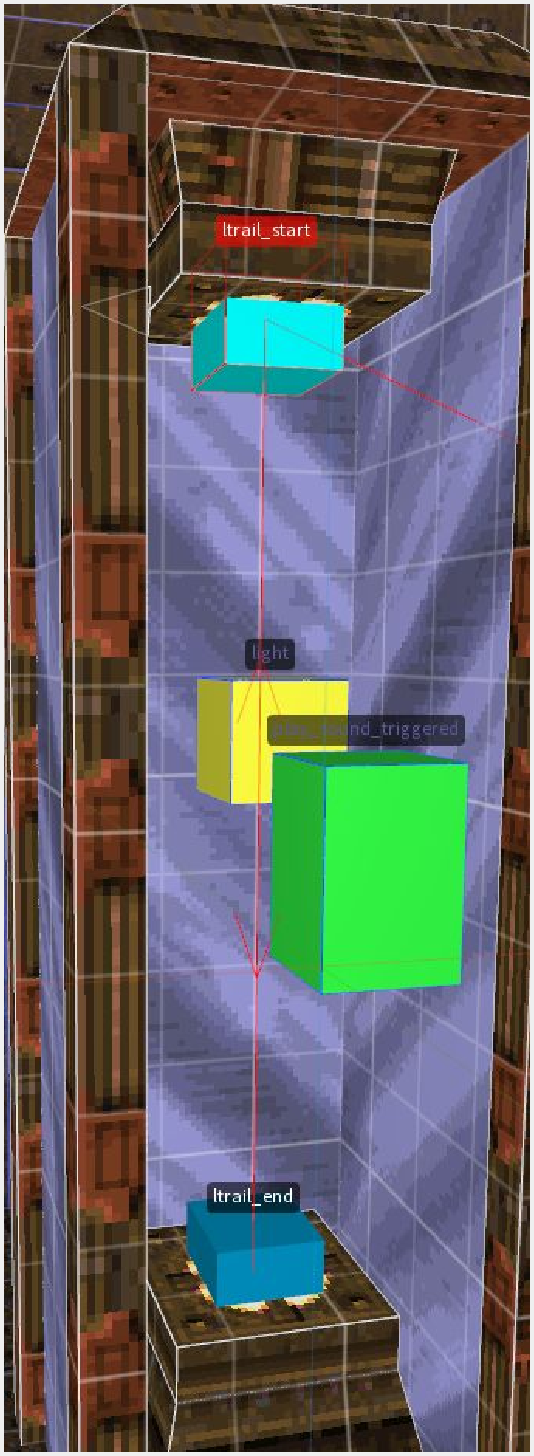
func_laser

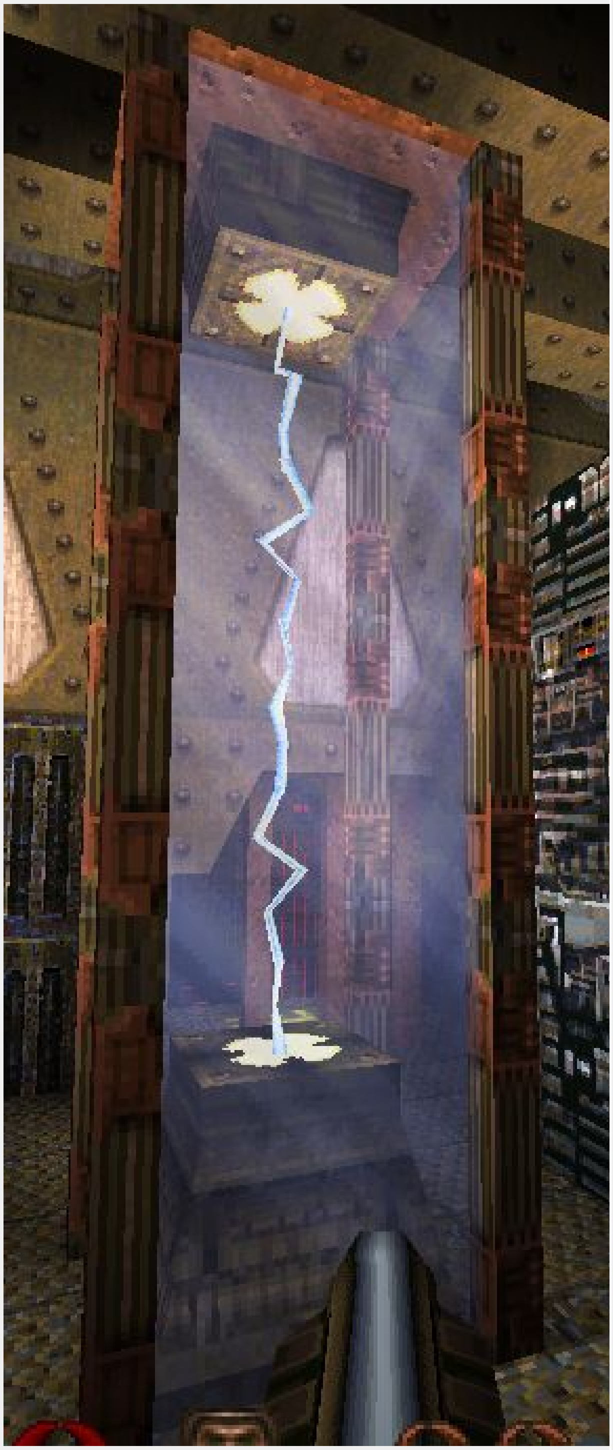
A togglable laser, hurts to touch, can be used to block players. START_OFF: Laser starts off. LASER_SOLID: Laser blocks movement while turned on. Keys: *dmg* damage on touch. default 1 *alpha* approximate alpha you want the laser drawn at. default 0.5. alpha will vary by 20% of this value. *message* message to display when activated *message2* message to display when deactivated.

Lightning

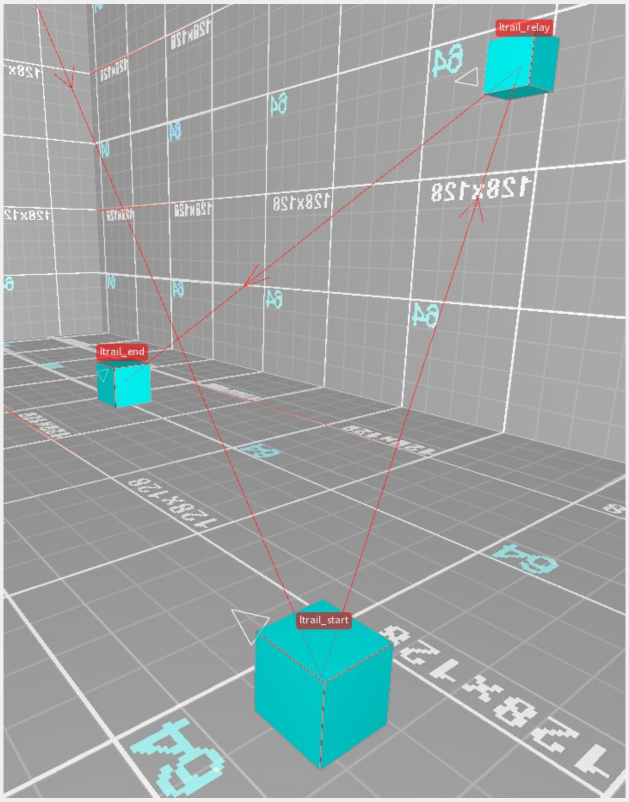
ltrail_start
ltrail_relay
ltrail_end

These lightning trail entities can be used for traps, decoration or for other scripted events. For the example below there are two entities. *ltrail_start* and *ltrail_end*, they are targeting each other.





If you want a chain of lightning events you would use a number of ltrail_relays between the start and end targeting one to the other, much like you would a path_corner with a func_train.



NOTE: The key / values are weirdly named in these entities. This is a quirk of QuakeC, where coders try to limit the amount of fields used by “recycling” unused fields to save memory.

ltrail_start Starting point of a lightning trail. Set *currentammo* to the amount of damage you want the lightning to do. Default is 25. Set *frags* to the amount of time before the next item is triggered. Default is 0.3 seconds. Set *weapon* to the amount of time to be firing the lightning. Default is 0.3 seconds. Set *sounds* to 1 for no sound. (Yes, it is weird.) Set the *TOGGLE* spawnflag if you want the lightning shooter to continuously fire until triggered again. Set the *START ON* spawnflag to have the lightning shooter start on. Do NOT use both these spawnflags at once.

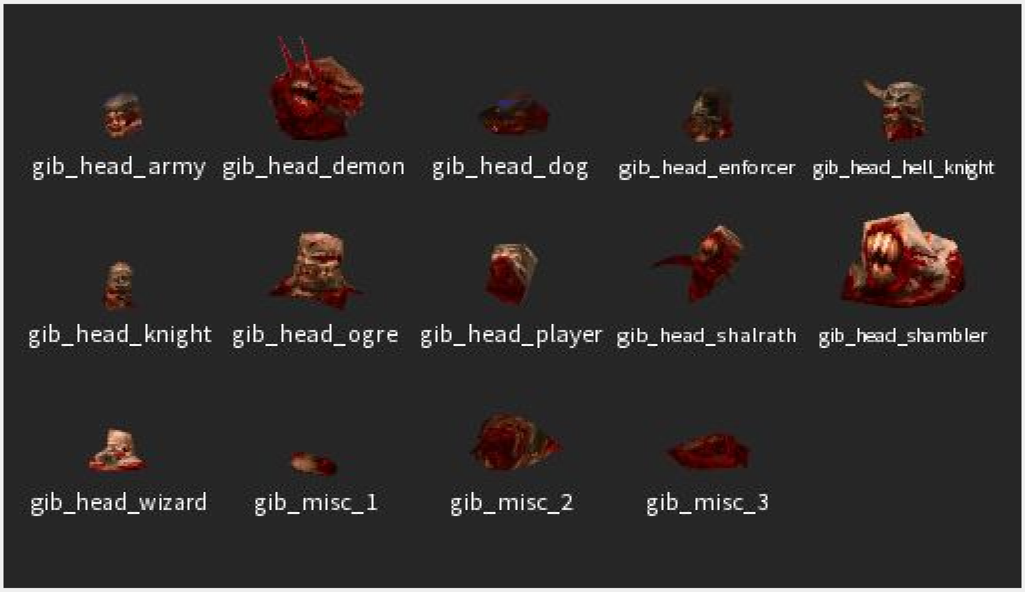
ltrail_relay Relay point of a lightning trail. Set *currentammo* to the amount of damage you want the lightning to do. Default is 25. Set *frags* to the amount of time before the next item is triggered. Default is 0.3 seconds. Set *weapon* to the amount of time to be firing the lightning. Default is 0.3 seconds Unfortunately, ltrail_relay entities cannot be set to silent.

ltrail_end Ending point of a lightning trail. Does not fire any lightning. Set *frags* to the amount of time before the next item is triggered. Default is 0.3 seconds.

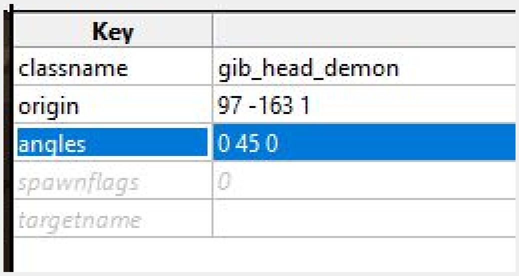
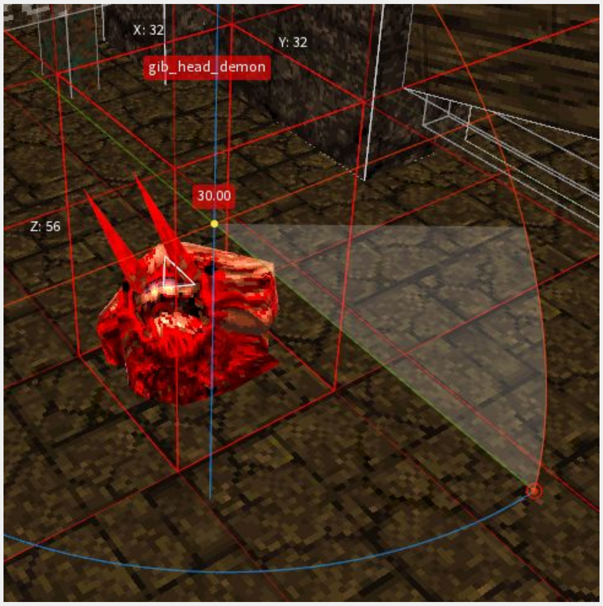
NOTE: To have a continuously firing bolt between two points, have a ltrail_start and ltrail_end targeting each other in a loop and set frags to -1. The sound this makes is not ideal, so consider making these silent and use a play_sound_triggered with a custom looping sound. This is shown in the pd_lasers sample map. In the devkit, sounds/dump/elec22k.wav is included for this very reason.

gib_(classname)

Easily add these bloody decorations to your map. (Also see *monster_dead_(classname)* below. You can use the SOLID spawnflag to enable collision on the model but clip brushes will work even better.



If you are using TrenchBroom take extra care when rotating these entities. The way TrenchBroom handles rotations for custom models requires a work around in some cases. If you want to simply rotate the gib model around the z axis there is no problem. However, if you wish to rotate the model in the X and Y or any combination, you will need to manually type in X Y and Z values *before* using the rotate tool. To do this, use the *angles* key (with an s) and type in something like 0 45 0 as the values. Then you can select the rotate tool and adjust the other values using the widget. Keep in mind the values 0 0 0 will not work. Also the *angle* key (no s) should be blank or set to 0 when using the *angles* key.

Key	
classname	gib_head_demon
origin	97 -163 1
angles	0 45 0
spawnflags	*0*
targetname	

monster_dead_(classname)

e.g. *monster_dead_ogre* More decorations for your maps. You can use the SOLID spawnflag to enable collision on the model but clip brushes will work even better. Keep in mind the same issue with rotation mentioned above applies to these models as well.

monster_dead_army
monster_dead_demon
monster_dead_dog
monster_dead_enforcer
monster_dead_hell_knight
monster_dead_knight
monster_dead_ogre
monster_dead_shalrath
monster_dead_shambler
monster_dead_wizard

Worldspawn

Features a *reset_items* key (default 0). Set to 1 to make the player start with default shotgun and axe. Set to 2 for an axe only start.

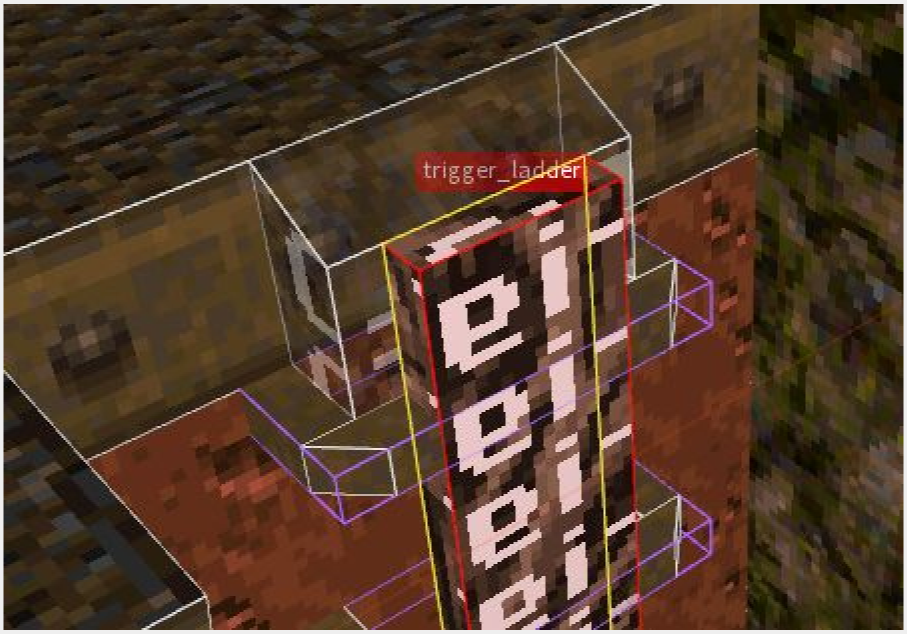
light_candle

A simple light emitting candle from Mission Pack 2. You can place them into the ground for shorter varieties.

Ladders

trigger_ladder

Create a small *trigger_ladder* brush covered with the trigger texture. Make sure the outside edge of the brush is flush with your ladder geometry. Set the *angle* key to the direction the player is facing when approaching the ladder. You can use a wedge shaped clip brush to smooth out any “sticky” movements at the top of the ladder as seen below. Please refer to pd_ladders.map for examples.

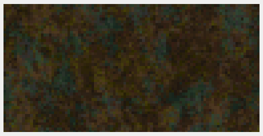
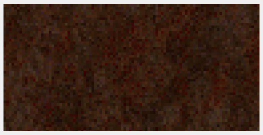
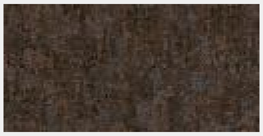
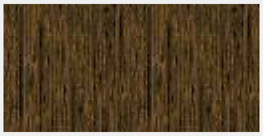
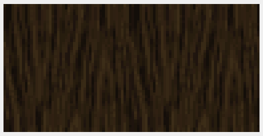
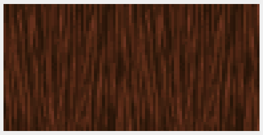
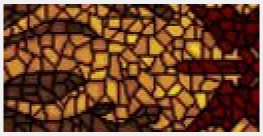
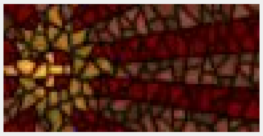
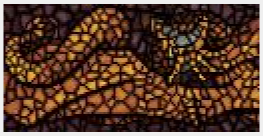
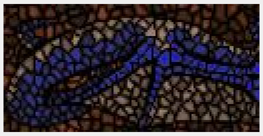
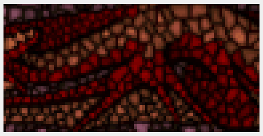


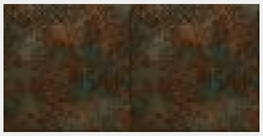
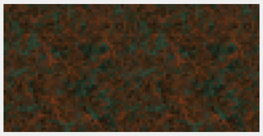
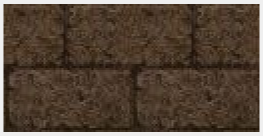
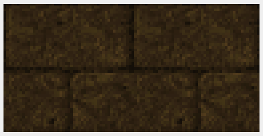
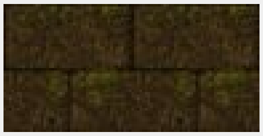
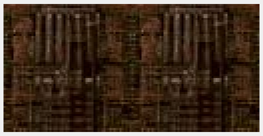
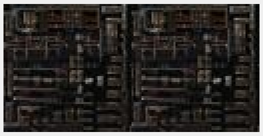
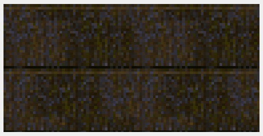
Breakables

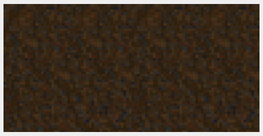
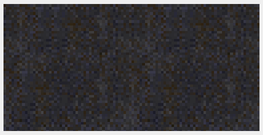
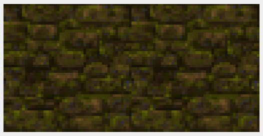
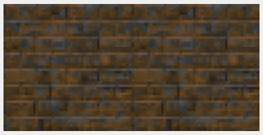
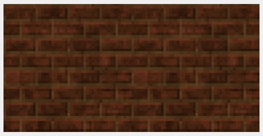
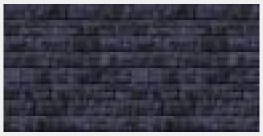
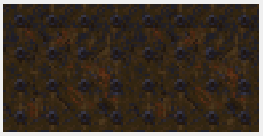
func_breakable

Breakables may seem overwhelming to new mappers, however it’s not as complicated as it looks. Also, there are two methods to choose from. One is the *Built-in* (easy) method and the other is the *Custom* method (more flexible.)

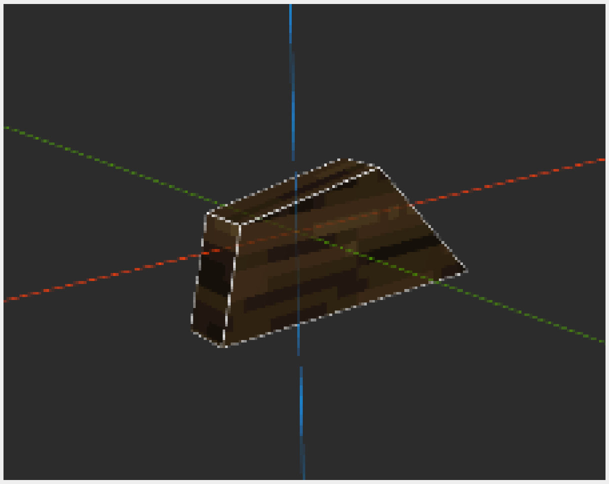
The Built-in method: Create your brush and make it a *func_breakable*. You can ignore any keys that begin with *brk* or *breakable*. Those are used with the custom method. With the built-in method you will set the *style* to one of thirty-two options listed below. By default, the breakable spawn 5 pieces of debris. You can change this amount with the *cnt* key/value. The default *health* of the brush is 20.There is a placeholder sound but you can use the *noise1* key to set a custom sound path. If you give the breakable a *targetname* it will only break when triggered. Use the *Explosion* spawnflag for an explosive brush. Use the *dmg* key to set a custom damage value. You can also use the *No Monster Damage* spawnflag to keep monsters from breaking the brush.

Style	Texture basis	Image	Description
0	custom		Green Metal (default)
1	custom		Red Metal
2	custom		Concrete
3	wood1_1		Pine wood
4	wizwood1_3		Brown wood
5	dung01_2		Red wood
6	window02_1		Stained Glass Yellow Flames
7	window01_4		Stained Glass Red Rays
8	window01_3		Stained Glass Yellow Dragon
9	window01_2		Stained Glass Blue Dragon
10	window01_1		Stained Glass Red Dragon

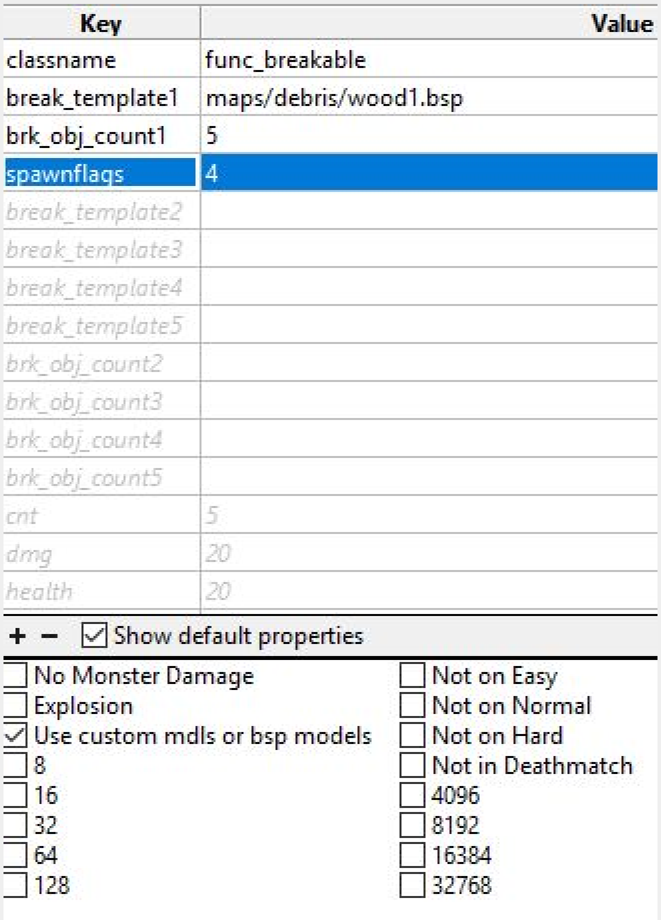
11	cop2_3		Light Copper
12	cop1_1		Dark Copper
13	wiz1_4		Tan Bricks Large
14	wbrick1_5		Brown Bricks Large
15	wswamp2_1		Green Bricks Large
16	tlight08		Generic Light Brown
17	comp1_5		Red Brown Computer
18	comp1_1		Grey Black Computer
19	metal4_5		Blue Green Metal
20	metal4_4		Blue Green Runic Wall
21	metal2_2		Brown Metal

22	metal1_3		Dark Brown Metal
23	metal1_2		Medium Brown Metal
24	m5_8		Blue Metal
25	city8_2		Green Stonework
26	city6_7		Blue Stonework
27	city2_8		Brown Bricks
28	city2_7		Tan Blue Bricks
29	city2_1		Red Bricks
30	city2_5		Blue Bricks
31	wizmet1_2		Metal Rivets

The Custom Method: This method uses external, custom models (.mdl format) or brush models (.bsp format) instead of the built-in system. You can make small pieces of debris by shaping them in a level editor and compiling them into a .bsp (See *Creating Debris* below.)



You can also use .bsps from other mods (check if you have permission to do so.) In the example below, we are only using one piece and duplicating it when the brush is “broken.” Set the Use custom mdls or bsp models spawnflag to enable this mode. Then set the path to the .bsp or model in *break_template1*. The *brk_obj_count1* determines how many instances of that bsp will be used. You can have 5 different pieces of debris total (break_template1-5) and control how many instances each of those templates spawns with *brk_obj_count1-5. noise1* is the path to the sound when breaking. *Style* and *cnt* are not used in this method but *health* and *dmg* are.



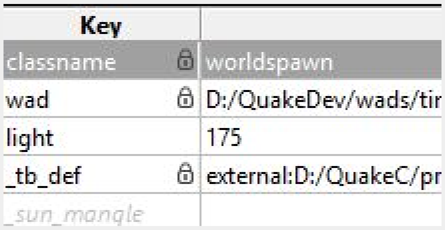
Key	Value
classname	func_breakable
break_template1	maps/debris/wood1.bsp
brk_obj_count1	5
spawnflags	4
break_template2	
break_template3	
break_template4	
break_template5	
brk_obj_count2	
brk_obj_count3	
brk_obj_count4	
brk_obj_count5	
cnt	*5*
dmg	*20*
health	*20*

+ − ☑ Show default properties

☐ No Monster Damage
☐ Explosion
☑ Use custom mdls or bsp models
☐ 8
☐ 16
☐ 32
☐ 64
☐ 128
☐ Not on Easy
☐ Not on Normal
☐ Not on Hard
☐ Not in Deathmatch
☐ 4096
☐ 8192
☐ 16384
☐ 32768

Creating debris

You can create *break_templates* as tiny maps and compile them into bsps. Create one piece at a time as their own map file. Create the debris at the center of the map (origin 0, 0, 0) Compile with qbsp.exe and light. No need to run vis.exe on these. You can add a *light* key/value to the Worldspawn to uniformly light the piece of debris.



Key	
classname 🔒	worldspawn
wad 🔒	D:/QuakeDev/wads/tir
light	175
_tb_def 🔒	external:D:/QuakeC/pr
_sun_mangle	

Place these pieces in your maps folder or a subfolder under maps called debris or breakables and remember to include these when you distribute your map.

NOTE: If you want debris to fall during an earthquake or similar event, use a skip texture to create an invisible breakable. Make sure the player cannot touch the brush, as skip textured brushes have collision. Clip textures won’t work for this.

Effect Entities

You can trigger the following visual effects.

Effect	**Details**
play_explosion	grenade explosion, causes damage
play_tbabyexplode	Spawn death explosion, causes damage
play_lavalsplash	large particle effect, can have custom sound
play_brlight	Toggles a bright lighting effect on or off.
play_dimlight	Toggles a lighting effect on or off.
play_mflash	When triggered, it plays a brief muzzle flash effect.
play_brfield	When triggered, toggles a spherical yellow particle effect.
play_gibs	When triggered, it plays gib effects and sound. Same as *meat_shower* from earlier versions. See an example of *meat_shower* used with a *func_counter* in pd_meat.map When triggered this entity will spawn a shower of gibs. *style* = 0 is regular gib effect, 1 is more violent *fly_sound* = 0 is silent, 1 plays randomized gib sounds *targetname* = Must be triggered
play_tele	When triggered, shows the teleport particle effects and sound. Same as *tele_fog* from earlier versions. Use this when killtargeting an entity if the player can see it happen. You can see an example on the Shambler near the *trigger_use* key entity in *The Gallery* map.

NOTE: *tele_fog* and *meat_shower* have been deprecated and renamed. The QuakeC code is still present for backward compatibility but the entities have been removed from the FGD.

func_bob

This will create a brush that gently moves back and forth or up and down depending on the angle. Use *targetname* to trigger it on, *angle* direction movement, use "360" for angle 0 *height* direction intensity (def=8) *count* = direction cycle timer (def=2s, minimum=1s) *waitmin* = Speed up scale (def=1) 1+=non linear, *waitmin2* = Slow down scale (def=0.75) *delay* = Starting time delay (def=0, -1=random) *style* If set to 1, starts off and waits for trigger _*dirt* -1 = will be excluded from dirtmapping, _*minlight* = Minimum light level for any surface of the brush model, _*mincolor* = Minimum light color for any surface (def='1 1 1' RGB) _*shadow* = Will cast shadows on other models and itself, _*shadowself* = Will cast shadows on itself. Use the BOB_COLLISION spawnflag for solid and conversely, BOB_NONSOLID.

misc_bob

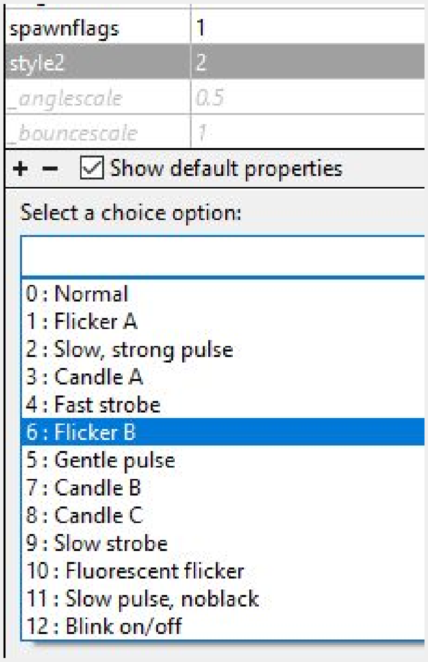
Same as above but uses a custom model instead of a brush. Use the *mdl* key to set the path of the model.

Switchable Animated Light Styles

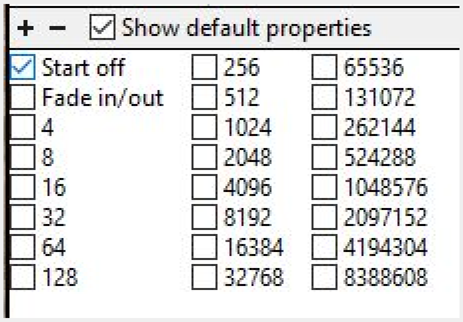
Normally, if you apply a style to a light (e.g. candle flicker, strobe) those cannot be triggered on and off. Progs_dump now has this feature, borrowed from c0burn's in-progress *Slipgate* mod. Just choose a *style2* selection from the dropdown and target the light as normal. Use the *START OFF* spawnflag if needed.

Select the *FADE IN / OUT* spawnflag for a beautiful fade in / out effect on normal lights.

NOTE: Fades will not work on animated lights (e.g. style or style2). Also, the FDG now includes all keys for ericw's lighting tools.



spawnflags	1
style2	2
_anglescale	0.5
_bouncescale	1



☑ Start off	☐ 256	☐ 65536
☐ Fade in/out	☐ 512	☐ 131072
☐ 4	☐ 1024	☐ 262144
☐ 8	☐ 2048	☐ 524288
☐ 16	☐ 4096	☐ 1048576
☐ 32	☐ 8192	☐ 2097152
☐ 64	☐ 16384	☐ 4194304
☐ 128	☐ 32768	☐ 8388608

Particle Effects

misc_sparks

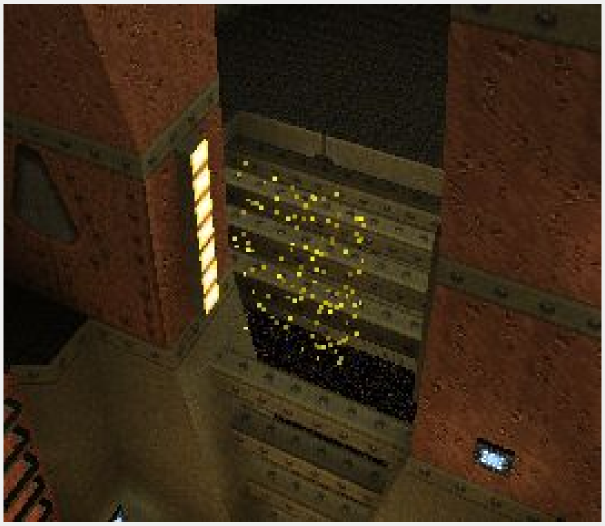
Produces a burst of yellow sparks at random intervals. If targeted, it will toggle between on or off. If it targets a light, that light will flash along with each burst of sparks. NOTE: targeted lights should be set to START_OFF. Spawnflags = SPARKS_BLUE: sparks are blue in color SPARKS_PALE sparks are pale yellow in color. *wait* is the average delay between bursts (variance is 1/2 wait). Default is 2. *cnt* is the average number of sparks in a burst (variance is 1/4 cnt). Default is 15. sounds 0 = no sound, 1 = sparks TIP: target a play_sound_triggered for a custom sound

misc_particle_stream

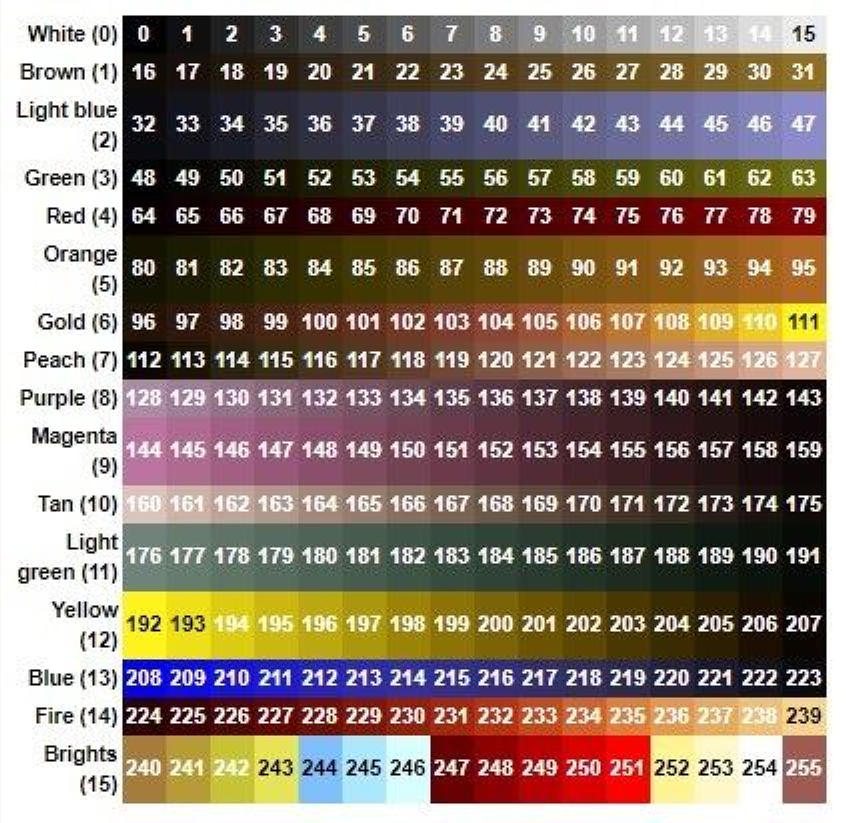
A particle stream! It appears when triggered. This entity is one end of the stream, target another entity as the other end-point. Usually an *info_notnull*, but you should be able to target anything (like monsters). *target* = This entity's origin is the end-point of the stream *dmg* = 1st Color, use this by itself if you want a single color stream *cnt* = 2nd Color, mixes particles of both colors. noise = Sound to play when triggered. See color palette reference below. NOTE: You can see this in action in the *pd_counter* sample map and at the end of the *pd_lasers* map.

func_particlefield

Creates a brief particle flash roughly the size of the defining brush each time it is triggered. You can see an example of this in the *pd_ladders* example map. In this case, the particle fields are animated in sequence to create a force field effect. *USE_COUNT* when the activator is a func_counter, the field will only activate when count is equal to cnt. Same as using a func_oncount to trigger. *cnt* is the count to activate on when *USE_COUNT* is set. *color* is the color of the particles. Default is 192 (yellow). *count* is the density of the particles. Default is 2. *noise* is the sound to play when triggered. Do not use a looping sound here. *dmg* is the amount of damage to cause when touched.



If you want to use another color for the particle field, refer to the Quake color palette below NOTE: not all colors will work:



White (0)	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Brown (1)	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Light blue (2)	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
Green (3)	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
Red (4)	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
Orange (5)	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
Gold (6)	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111
Peach (7)	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127
Purple (8)	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143
Magenta (9)	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159
Tan (10)	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175
Light green (11)	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191
Yellow (12)	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207
Blue (13)	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223
Fire (14)	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239
Brights (15)	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255

misc_particles

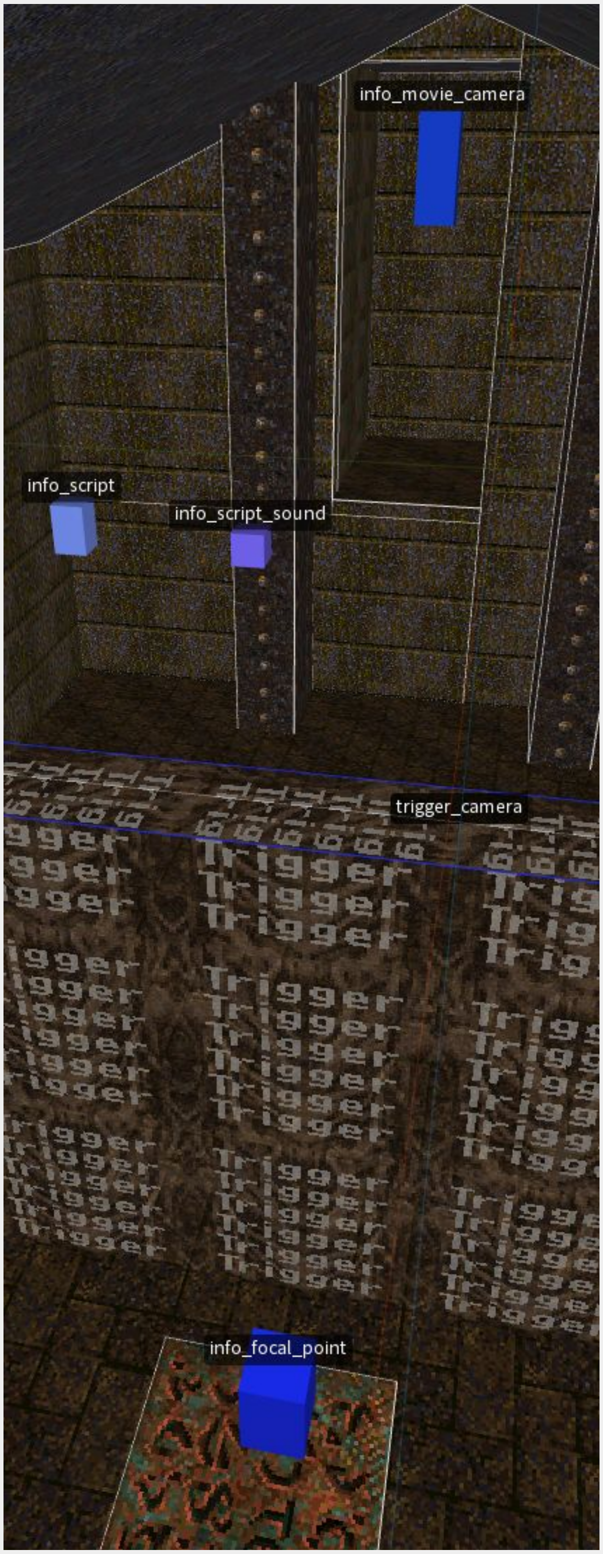
Produces a continuous particle splash for waterfalls and other effects. Can be triggered and toggled. Spawnflags = START_OFF The default behavior has the particles shimmering in an upward motion. *color* = color of particles. 0 through 15, corresponds to a row of the quake palette (see above for palette numbers). (default 0) *movedir* = average movement vector of particles (default 0 0 4) NOTE: Play with negative numbers to change the movement direction. *wait* = time between particle generation cycles. (default 0.1) volume = density of particles. (default 10)

misc_particlespray

Shoots particles either when triggered, or continuously when not triggered by anything. *color* is the palette color of the particles. (default 47) *movedir* is the vector distance that the particles will travel before disappearing. (in x y z) NOTE: Play with negative numbers to change the movement direction. *delay* is the delay between each triggering (default 0.1) *duration* is the amount of time that it will continue to release particles so that it can release a long stream of particles with only one triggering, *count* is the number of particles to make each time (default 15) *noise* is the name of the .wav file to play when triggered.

The two particle effects above are similar but there are some key differences. Take a look at the pd_counter map for some different examples.

Cutscenes



The cutscene system is taken from the Drake mod beta devkit. Scenes take a bit of testing and tweaking to set up, so please read this section *carefully* if you want to include them in your projects. One missing key | value or typo will blow up the whole operation! **It's best to start with a small test level and learn how they work before moving forward.** Also play very close attention to the "best practices" section below! Finally, take a look at my video tutorial (FIXME)

NOTE: unlike many Quake entities, there are key | value pairs that are required to be set even though they may not seem to do anything.

Your first step should be to play the sample map *pd_cutscenes*, then open the map in your map editor and take a look at the different setups. There's a secret area of the map that shows the most simple setup, with one message and one camera. There are also three other, more complex setups in the map. Cutscenes can be skipped by pressing a weapon key or any impulse command.

There are a minimum of four entities required to make a cutscene work.

First, a *trigger_camera* that the player will enter to begin the scene. You can also use a *trigger_camera_point* if you need to trigger your scene without the player touching the trigger. You will see both methods in the sample map.

The second required entity is an *info_movie_camera*. This will "aim" the camera at the third required entity: an *info_focal_point*.

The fourth required entity is an *info_script*. This controls how long the player is in the scene, triggers events and holds any text messages that will play during the cutscene.

trigger_camera

This will begin a cutscene when touched. Some of these keys need to match the corresponding fields in the targeted *info_movie_camera* and *info_script*. IMPORTANT: most of the following keys are required unless noted.

Key	**Details**
focal_point	Point the targeted camera at this point.
script	Match script_num field of the info_script
script_delay	The amount of time to stay on the first script page. NOTE: You can usually set this to 1 because the *script_delay* key of the matching *info_script* will override this value.
target	Targetname of the first camera in the cutscene.
targetname (optional)	If the *trigger_camera* has a *targetname*, it will be dormant until triggered.

info_movie_camera

This is the target of the *trigger_camera* and controls the viewport of the cutscene. When using multiple cameras in a sequence, you need at least three cameras (see “complex cutscenes” below).

Key	**Details**
focal_point	Point the camera at this point.
targetname	The name of this camera.
delay (optional)	When the camera moves, don't track the focal_point’s position, keep the initial view angle.
speed (optional)	This controls the rate of travel to this camera in (Quake units per second) from another camera.
wait (optional)	Wait here in seconds, before moving to next camera if part of a sequence
target (optional)	*targetname* of the next *info_movie_camera* in a sequence.

info_focal_point

This is the point that the camera will face. It should have a *targetname* value matching the *camera_trigger* and *info_movie_camera's focal_point* fields. When using multiple cameras the focal points can change (see "complex cutscenes" below).

info_script

This controls the on-screen timing and the optional message text fields.

Key	Details
script_num	This should match the *script* field of the *trigger_camera* or *trigger_camera_point*. IMPORTANT: **Every *script_num* in a map needs to be unique!**
next_script	This is the *script_num* field of the next *info_script*, if part of a sequence. Set to zero if this is the last script in a series. IMPORTANT: **This value must be set by hand even if the number is zero!** This is unlike almost every other entity field in Quake mapping! Your cutscene will fail without this set.
script_delay	How many seconds to stay on this script. This overrides the same key on *trigger_camera*.
message (optional)	Optional text that will stay on screen for the amount of time set in script_delay. It's safest to limit this to a max of 64 characters.
target1-4 (optional)	Use these fields to trigger other events in time with the current script. Use *trigger_relays* if you need to *killtarget* something.

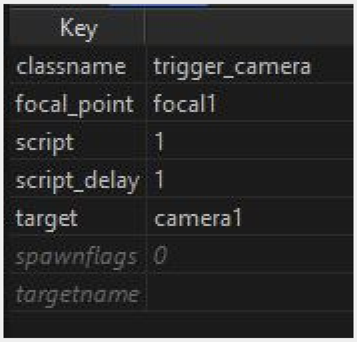
info_script_sound

You can use this optional entity to add a sound when text is displayed or you can even trigger custom sounds and add dialogue to your scenes!

Key	Details
sounds	Default Quake sounds for messages. Select 4 if you want to use a custom sound file.
noise1	Path to custom sound file. Requires sounds key set to 4.
targetname	Name of entity. You can use this multiple times in the same level but note the sound is directional.

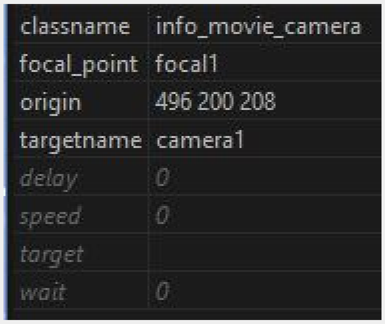
Creating a Simple Cutscene

Create a *trigger_camera* brush and give it these key | values:



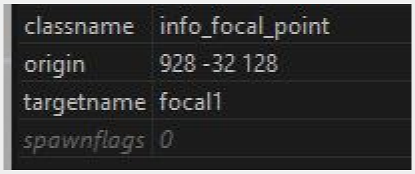
Key	
classname	trigger_camera
focal_point	focal1
script	1
script_delay	1
target	camera1
spawnflags	*0*
targetname	

Next add an *info_movie_camera* and give it these key | values:



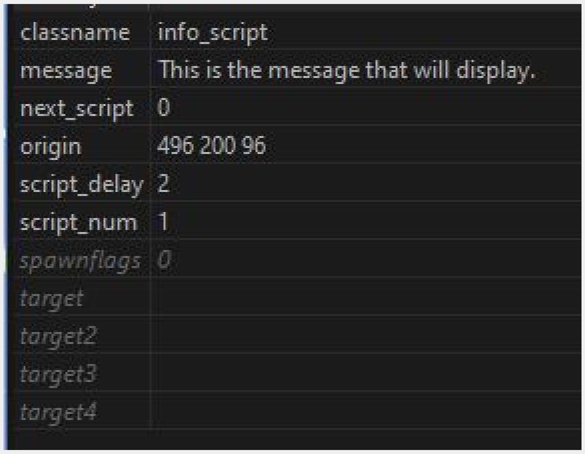
classname	info_movie_camera
focal_point	focal1
origin	496 200 208
targetname	camera1
delay	*0*
speed	*0*
target	
wait	*0*

Notice they both have the same *focal_point* key. Now create that *info_focal_point* give it these key | values:



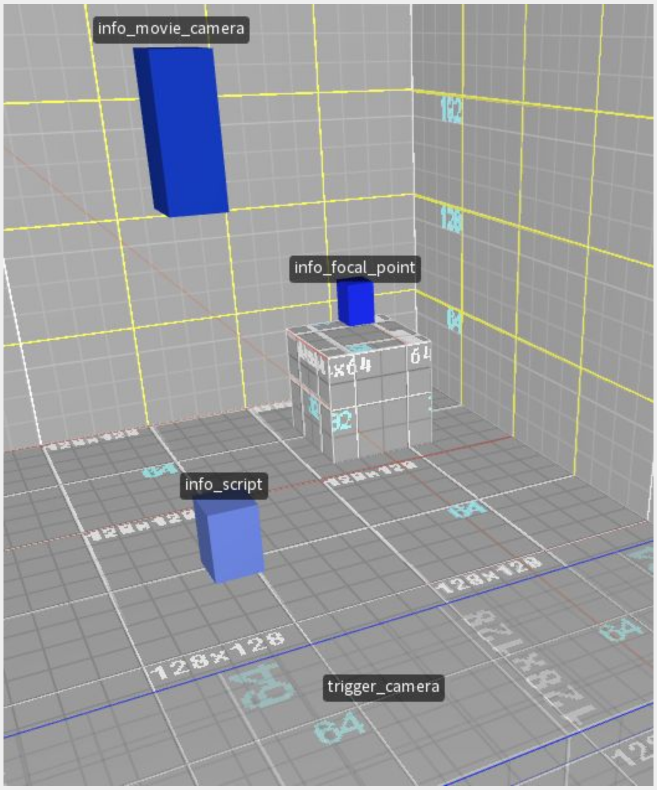
classname	info_focal_point
origin	928 -32 128
targetname	focal1
spawnflags	*0*

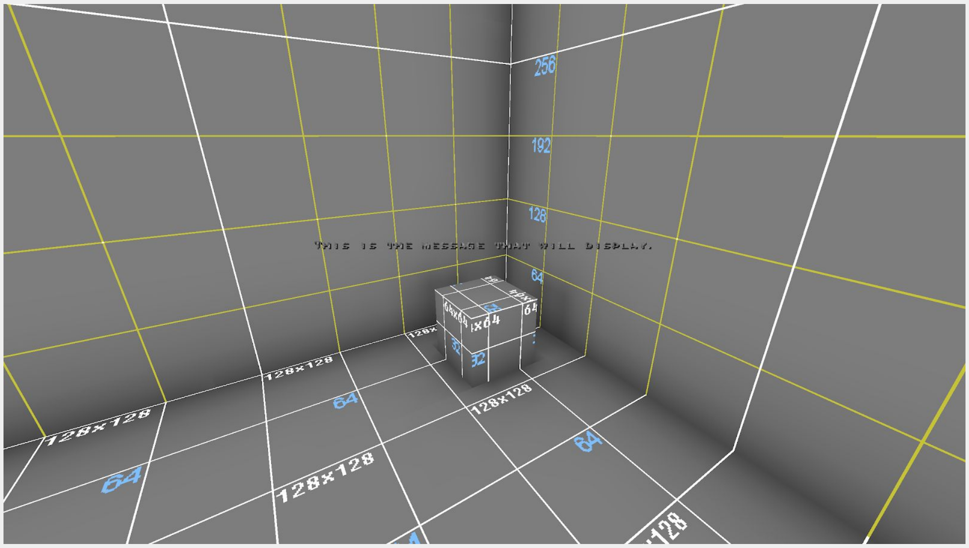
Now for the *info_script*. Note the *script_num* matches the *script* key from *trigger_camera*. The *next_script* value is <u>set by hand to zero</u>, this is **really** important to add or your cutscene will break! It's set to zero, because it's the last script of the scene.



classname	info_script
message	This is the message that will display.
next_script	0
origin	496 200 96
script_delay	2
script_num	1
spawnflags	*0*
target	
target2	
target3	
target4	

The length of the scene is controlled by the *script_delay* key in the *info_script*. Leave the message key blank if you just want a shot without text. Now you can move the focal point and camera around for your desired “angle”. The following screenshots show in-editor and then the in-game vantage points.




This is the message that will display.

Complex Cutscenes

You can add more cameras, scripts and focal points for a more complex scene. You can also animate the camera from one point to another but getting good looking “shots” takes a bit of time and tinkering.

I’ve created smaller demo levels for easy reference in addition to the *pd_cutcenes* map. These maps are not accessible from the progs_dump start map but you can load them via the console.

Here’s what each map demonstrates:

Map Name	**Details**
pd_cutscn_simple	A static camera, one message scene.
pd_cutscn_tracking	Animated camera between two points with two messages and a blank script between them for timing. Notice how the *delay* key is set on all the cameras and only one focal point is needed. This makes each camera focus in the same direction for each move.
pd_cutscn_cuts	Scene that “cuts” between three vantage points, with three separate focal points. Notice how the speed key is set to 999999 to make the move nearly instantaneous. You may still see a “flash frame” between the edits here. There’s no real way around this.

When moving the camera, even between just two points, you will need three info_movie_cameras. (This is due to some quirks in the original QuakeC and took me a long time to figure out!) If you only want two vantage points in your scene, simply use the *wait* key on the second to last camera. Set this to a longer amount of time than is controlled by the *script_delay* key in your *info_script* for that section of the cutscene. You can see this clearly in *pd_cutscn_tracking* and in the other more complex demos.

Cutscene Best Practices

- Make small test maps to set up your cutscenes. Scenes require a lot of testing and tweaking. When they are working, paste them into your map and adjust as needed.

- Do not quit the game while in a cutscene, this will reset your mouse sensitivity and console viewsize. Add this info in the readme for your mod so players know not to quit!

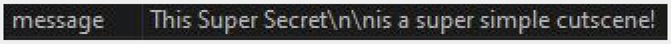
- Keep your cutscenes as simple as possible. Things can break very quickly as you ramp up the complexity.

- Timing is controlled in two places when using multiple cameras. (*script_delay* in *info_script* and *wait* in *info_movie_camera*) That makes it harder to make small changes. Break up longer sequences up into smaller parts.

- If you move the camera, keep the move on the same axis as the focal point. Any panning or tilting of the viewport will cause the screen to judder. It looks terrible and should be avoided.

- Camera moves in X and Y will display a bit of “player bob”. Play with the speed key to make the move faster and it won’t be as apparent.

- As in other Quake entities, you can add a line break in a message by adding \n with no space before the text of the second line. Here I’ve added two to make a blank line between sentences.



```
message    This Super Secret\n\nis a super simple cutscene!
```

- Do not reuse *info_scripts* for different cutscenes. Things will break. You *can* reuse *info_movie_cameras* as long as they are triggered by different *trigger_camera* entities.

- As mentioned above, *info_script_sound* entities can be reused. Be aware that because the way Quake handles audio, the sound direction can change depending on where the entity is in relation to the camera. If you have sound coming from only one direction center the *info_script_sound* on the focal point of the camera.

- Don’t over do it. Quake is a fast paced game. Few players want to watch a three hour Quake movie with terrible voice acting!

Rotation Entities

By request, the Hipnotic (Quake Mission Pack 1) rotation entities have been added but are **unsupported.** They *should* work, but are untested, so use at your own risk! (Other mappers have used these without issue). The text below is taken from the Quake C code for the rotation system. Refer to the included sample map *pd_rotate* for examples. Enjoy!

func_rotate_entity

Creates an entity that continually rotates. Can be toggled on and off if targeted. TOGGLE = allows the rotation to be toggled on/off START_ON = whether the entity is spinning when spawned. If TOGGLE is 0, the entity can be turned on, but not off.

If "deathtype" is set with a string, this is the message that will appear when a player is killed by the train. "rotate" is the rate to rotate. "target" is the center of rotation. "speed" is how long the entity takes to go from standing still to full speed and vice-versa.

path_rotate

(Train with rotation functionality) Path for rotate_train. ROTATION tells the train to rotate at a rate specified by "rotate". Use '0 0 0' to stop rotation. ANGLES tells the train to rotate to the angles specified by "angles" while traveling to this path_rotate. Use values < 0 or > 360 to guarantee that it turns in a certain direction. Having this flag set automatically clears any rotation. STOP tells the train to stop and wait to be retriggered. NO_ROTATE tells the train to stop rotating when waiting to be triggered. DAMAGE tells the train to cause damage based on "dmg". MOVETIME tells the train to interpret "speed" as the length of time to take moving from one corner to another. SET_DAMAGE tells the train to set all targets damage to "dmg" "noise" contains the name of the sound to play when the train stops. "noise1" contains the name of the sound to play when the train moves. "event" is a target to trigger when the train arrives at path_rotate.

func_rotate_train

In path_rotate, set speed to be the new speed of the train after it reaches the path change. If speed is -1, the train will warp directly to the next path change after the specified wait time. If MOVETIME is set on the path_rotate, the train interprets "speed" as the length of time to take moving from one corner to another. "noise" contains the name of the sound to play when the train stops. "noise1" contains the name of the sound to play when the train moves. Both "noise" and "noise1" defaults depend upon the "sounds" variable and can be overridden by the "noise" and "noise1" variable in path_rotate.

Also in path_rotate, if STOP is set, the train will wait until it is retriggered before moving on to the next goal.

Trains are moving platforms that players can ride. "path" specifies the first path_rotate and is the starting position. If the train is the target of a button or trigger, it will not begin moving until activated. The func_rotate_train entity is the center of rotation of all objects targeted by it.

If "deathtype" is set with a string, this is the message that will appear when a player is killed by the train. *speed* (default 100) *dmg* (default 0) *sounds* 1 = ratchet metal

func_movewall

Used to emulate collision on rotating objects. VISIBLE causes brush to be displayed. TOUCH specifies whether to cause damage when touched by a player. NONBLOCKING makes the brush non-solid. This is useless if VISIBLE is set. "dmg" specifies the damage to cause when touched or blocked.

rotate_object

This defines an object to be rotated. Used as the target of func_rotate_door.

func_rotate_door

Creates a door that rotates between two positions around a point of rotation each time it's triggered. STAYOPEN tells the door to reopen after closing. This prevents a trigger-once door from closing again when it's blocked. "dmg" specifies the damage to cause when blocked. Defaults to 2. Negative numbers indicate no damage. "speed" specifies how the time it takes to rotate "sounds" 1 = medieval (default), 2 = metal, 3 = base 4 = silent

Sample Maps

You can find these in the source folder along with a wad file containing all the textures used. You are welcome to copy and paste the entity setups and adjust as needed to use them in your maps. Please do not copy brushes or geometry from these maps. The following chart shows what examples exist in each map. NOTE: not all entities are demonstrated in the sample maps.

Map	Entity Setup Examples	Notes
pd_breakables	func_breakable, misc_candle	
pd_counter	func_counter, func_oncount, misc_particle, misc_particle_stream, misc_particlespray	
pd_elevator	func_new_plat, func_elvtr_button	
pd_ladders	trigger_ladder, func_particlefield, misc_sparks, func_breaklable, func_togglewall	
pd_lasers	func_laser, ltrail_start,ltrail_relay, ltrail_end, switchable light styles, misc_particle_stream, trigger spawned monsters	Can you find the YA secret?
pd_lightning	func_counter, ltrail_start,ltrail_relay, ltrail_end, trap_switched_shooter	
pd_lava	play_lavasplash, func_fall, trigger_shake, misc_particle, func_train (triggered)	
pd_meat	meat_shower, func_counter, gib_*, monster_dead_*	
pd_void	func_bob, suspended items, trigger spawn items, func_breakables	
pd_zombies	func_counter, func_oncount, enhanced zombies, trigger spawned monsters	
pd_rotate	func_rotate_entity, path_rotate, func_rotate_train, func_movewall, rotate_object, func_rotate_door	This is Hipnotic's original sample map. Slight changes to lighting and renamed.
pd_ionous	is_waiting, trigger spawned monsters	from 1.0.0
pd_yoder	trigger_push_custom, multiple trigger names, trigger_setgravity	from 1.0.0
pd_gallery	all entities from version 1.0.0	NOTE: If you are new to the mod review this one!
pd_gravity	trigger_setgravity, trigger spawned monsters	from 1.0.0
pd_keys	item_key_custom	currently no entrance for this in the start map

Credits

QuakeC Sources

misc_model.qc, math.qc by Joshua Skelton
https://gist.github.com/joshuaskelly/15fe10fbaaa1bf87b341cba6e3ad2ebc

Trigger Spawned Monsters added via Preach's excellent tutorial:
https://tomeofpreach.wordpress.com/2017/10/08/teleporting-monsters-flag/

various .qc from custents by Carl Glave
http://www.quaketastic.com/files/tools/windows/quakec/custents.zip

various .qc from Hipnotic's Quake Mission Pack Scourge of Armagon
Original Code written by Jim Dose and Mark Dochtermann
http://www.quaketastic.com/files/tools/windows/quakec/soa_all.zip

various .qc from Rogue's Quake Mission Pack Dissolution of Eternity
Original Code written by Peter Mack et al.
http://www.quaketastic.com/files/tools/windows/quakec/doe_qc.zip

Preach's clean Quake 1.06 source courtesy of Joel B
https://github.com/neogeographica/quakec/tree/1.06_Preach

various .qc from Rubicon Rumble Pack Devkit by ijed / Louis
http://www.quaketastic.com/files/single_player/mods/RRP_DEVKIT.zip

Arcane Dimensions breakable and music code by Simon O'Callaghan et al.
http://www.simonoc.com/pages/design/sp/ad.htm

Honey source by czg
https://www.quaddicted.com/reviews/honey.html

Zerstörer QuakeC Development Kit - Dave 'Ace_Dave' Weiden and Darin McNeil
https://www.quaddicted.com/reviews/zer.html

various .qc code from Rubicon 2 copyright 2011 John Fitzgibbons.
https://www.quaddicted.com/reviews/rubicon2.html

deadstuff version 1.0 - Tony Collen
ftp://archives.gamers.org/pub/idgames2/quakec/level_enhancements/deadstuf.zip

Remake Quake code by Supa, ijed and (?)
https://icculus.org/projects/remakequake/

switchable lightstyles and DEF entries from Slipgate by Michael Coburn
https://github.com/c0burn/Slipgate

cutscenes and various .qc code from Drakebeta by Patrick Martin
http://www.quaketastic.com/files/single_player/mods/drakebeta.zip

trigger_look code by NullPointPalidin
https://nullpointpaladin.wordpress.com/

Copper style noclip code from Copper by Lunaran
http://lunaran.com/copper/modding/

Additional code assistance and examples from iw, NullPointPaladin Qmaster, RennyC, Khreathor, Spike and c0burn.

Map / Model Credits

progs_dump logo by D.E.F.A.M.E.
https://defameart.com/

Celeritate satus
start by Danz

Ineffable Crown of Darkness
pd_ionous by Ionous

Eigenstate
pd_gravity by Ionous
voice.of.the.nephilim@gmail.com
@voiceovnephilim

Magic River
pd_yoder by Yoder
AndrewYoder@live.com
@Mclogenog

maps by dumptruck_ds:

pd_breakables
pd_counter
pd_elevator (based on an original map by iw)
pd_gallery
pd_ladders
pd_lasers
pd_lava
pd_lightning
pd_meat
pd_void
pd_zombies
pd_keys (by iw)

candle.mdl, mervup.mdl and lspike.mdl - from the Rogue mission pack.
debris.mdl - from Rubicon2 edited by dumptruck_ds
s_flame.spr - from Duke Nukem 3D (?)
g_shotgn.mdl - from Rubicon2 by metlslime
g_shotty.mdl - created by Slapmap
spark.mdl - from Rubicon2 by metlslime
s_null.spr - from Quoth (by Kell?)
h_boss.mdl by c0burn with edits by Khreathor