progs_dump

A Quake mapping devkit by dumptruck_ds
lango.lan.party@gmail.com
version 1.1.1
2019-26-8

This is a Quake mod compilation designed for mappers. Some of this code is from 2019 other bits date back to 1997! Please read the credits section for more detailed info, credits and links.

NOTE: **If you use this DevKit, your project should be released as a stand-alone mod and installed into its own folder in the Quake directory.**

To get started, unzip *mod_template.zip* into your Quake directory and rename the *my_mod* folder to the name of your mod. Use progs_dump itself for a learning tool and the *my_mod* folder for your project. You should not include the progs_dump sample maps with your mod. However, there are a few models, sprites and sounds you are free to use included in the template folder.

progs_dump is intended to give Quake mappers more creative options than “vanilla” Quake while retaining the look and feel of the original. Mappers can add the following features to their projects:

- **trigger spawned monsters**
- **custom sounds** (ambient and triggered)
- **custom models and sprites with animation playback**
- **multiple targets, targetnames and killtargets**
- **visual effects like explosions, lightning and particles**
- **enhanced trains** (spawnflag to start and stop via trigger)
- **enhanced triggers** (is_waiting flag to delay triggering)
- **enhanced plats** (elevators, toggle, start at bottom, etc.)
- **trigger_setgravity** (on player and / or monsters)
- **trigger_usekey** (trigger volume that requires a key to fire)
- **two varieties of breakables**
- and **many** more

This progs also fixes the Rotfish monster kill count bug in addition to making Rotfish non-solid as soon as they are killed. Other minor bugs have been addressed as well. See the changelog for info.

There are sample maps included with this package that are meant to show what is possible with this project. See the sample maps section below for more info.

Please read *progs_dump-1.1.1-README.txt* for important info and last minute changes.

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, Johnny Law, iJed, ionous, McLogenog, Danz and many others 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 who assisted.

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

You can inquire about progs_dump on the Quake Mapping Discord:

https://discordapp.com/invite/j5xh8QT

dumptruck’s Quake videos:

https://www.youtube.com/c/dumptruckds

What’s New in 1.1.1

* Added func_elvtr_button and ELEVATOR functionality to func_new_plat (iw)
* Fix Fiends never going into their pain animation (iw)
* Fix a critical bug involving func_elvtr_button (iw)
* Address various FTEQCC warnings (iw)
* Fix kill count bug with trigger-spawned Rotfish (iw)
* Fixed qc syntax errors in monsters.qc (iw)
* Fix various issues involving gravity and ladders (iw)
* Make trigger_ladder ignore bad (up/down) angles (iw)
* Fix items not respawning in deathmatch mode (iw)
* Fix a "friendly monster" bug in trigger spawned monsters (iw)
* Fix pain_target never firing in some cases (iw)
* Fix item_health collision with BSP model entities (iw)
* Fix the weapon_shotgun spawnflag test (iw)
* Allow "wait" to be set for elevators (iw)
* other tweaks to DOE elevator code in doeplats.qc (iw)
* pd_elevator demo level (iw)

Features

Trigger Spawned Monsters

Setting spawnflags to 8 allows monsters to be hidden in place until triggered. If you use the included .fgd file, the spawnflag *Trigger Spawn* is selectable on all monsters. This makes spawning monsters much easier than in vanilla Quake.

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. Setting the *wait* key to 1 will disable the teleport visual effects and sounds. In the included .fgd can use the drop down menu to select *Spawn Silently*.

Set *spawn_angry* to 1 to have monsters immediately attack the player. 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 (see below.)

Monsters can have a custom *health* levels. Use this with care! You should warn the player with a message or some other communication if you change any of these dramatically. e.g. “These grunts seems pretty tough!” Finally, you can stop Ogres, Grunts and Enforcers from dropping backpack ammo by setting the *keep_ammo* key to 1.

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.

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.

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*

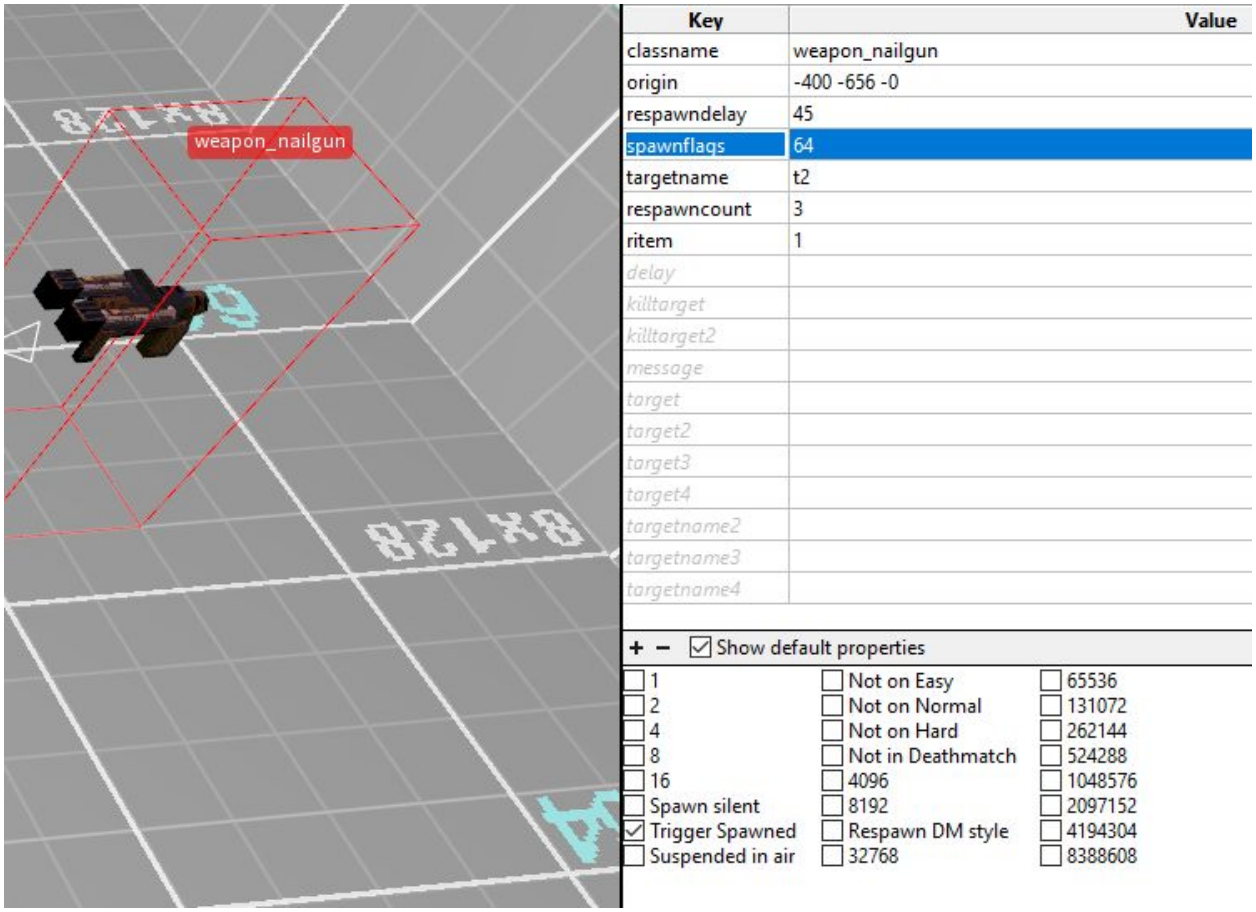
original targetname field to give entities unique identifiers, and use the remaining fields for group triggers.

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.

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 settings for a deathmatch game. You can set a custom respawn time using the *respawndelay* key and control how many times an items respawns with the *respawncount* key | value. By default items will display the “tfog” effects when respawning. You can mimic deathmatch respawns with the *Respawn DM Style* spawnflag. This skips the “tfog” effect and plays a more subtle sound effect.

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



weapon_shotgun

This is a new entity that should be used when you want a player to spawn with only an axe. (trigger_take_weapon) There are two models to choose from. Spawnflag 2 (the default) selects an unused model from Rubicon 2 by metlslime and spawnflag 4 is from Slapmap and has been used in a few mods.

Custom Sounds

play_sound_triggered
play_sound
ambient_general

IMPORTANT - New 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.

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.

- toggle (spawnflags): 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.

- 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

Plays sound of thunder sound at a random interval.

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.

Custom Models and Sprites

misc_model

A point entity for displaying models and sprites. A frame range can be given to animate the model.

- 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.
- first_frame: The starting frame of the animation.
- last_frame: The last frame of the animation.
- speed: The frames per second of animation. Divide 1 by the fps for this value.
- angles: pitch roll yaw (up/down, angle, tilt left/right)

IMPORTANT: Set the angle value to 0 if using angles key to rotate mdls (see gib_ section below)

Enhanced Triggers

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

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_push_custom

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

If *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 *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!

trigger_setgravity

If *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 equal lower gravity. Setting 100 is also normal gravity. Numbers above 100 will make the player "heavier", i.e. harder to jump.

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, gibs, ammo, etc... also kills player. Spawnflags can be used to protect players or monsters.

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* instead of *targetname*.

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 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.

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. 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.

NOTE: **Currently entities targeted by func_counter and/or func_oncount cannot display *messages* to the player.** This is due to the way messages are coded in the game. This might be fixed in a future version, but for now you'll need to use other triggers to combine messages with func_counter setups.

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.

FireAmbient

This is a simple looping sound from the torches. Use this if you are using custom fire sprites or models.

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."

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_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 wall is turned off. *noise1* is the sound to play when 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

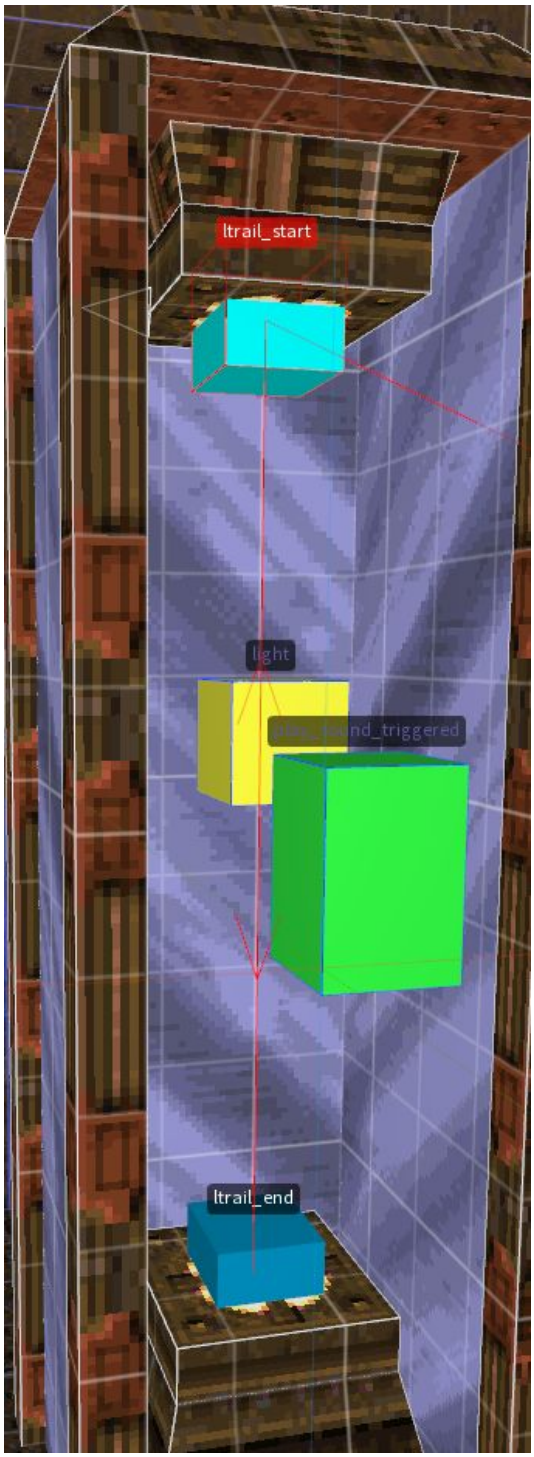
Just like the standard Quake train entity but with the *RETRIGGER* spawnflag set the train will stop at each path corner and wait to be retriggered before moving again. This will be great for more complicated lifts, doors and of course… trains. Set the *sounds* key to 3 to use custom sounds, then set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

func_laser

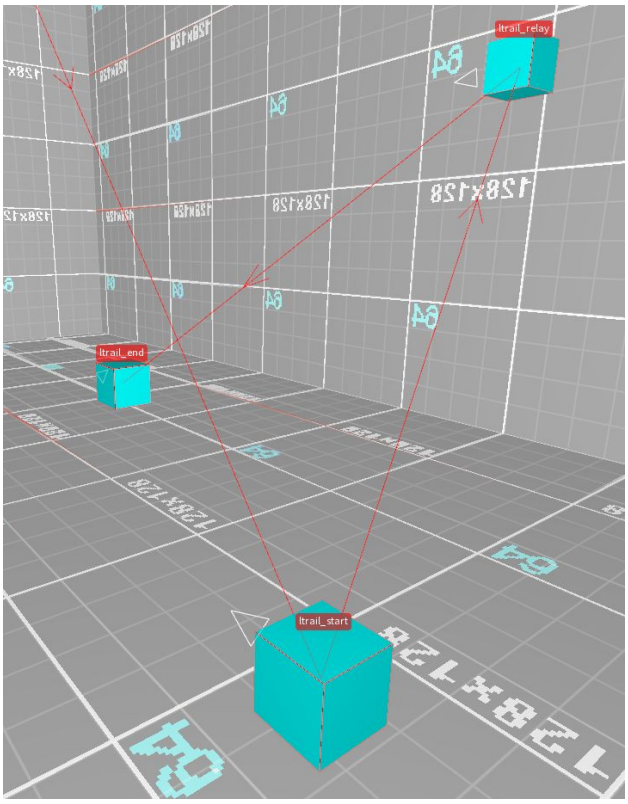
A togglable laser, hurts to touch, can be used to block players. START_OFF: Laser starts off. LASER_SOLID: Laser blocks movement while turned on. Keys: *dmg* damage on touch. default 1 *alpha* approximate alpha you want the laser drawn at. default 0.5. alpha will vary by 20% of this value. *message*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

ltrail_start
ltrail_relay
ltrai_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 and limit the amount of fields used by “recycling” unused fields to save memory.

ltrail_start Starting point of a lightning trail. Set *currentammo* to amount of damage you want the lightning to do. Default is 25. Set *frags* to amount of time before the next item is triggered. Default is 0.3 seconds. Set *weapon* to 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 amount of damage you want the lightning to do. Default is 25. Set *frags* to amount of time before the next item is triggered. Default is 0.3 seconds. Set *weapon* to 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 amount of time before the next item is triggered. Default is 0.3 seconds.

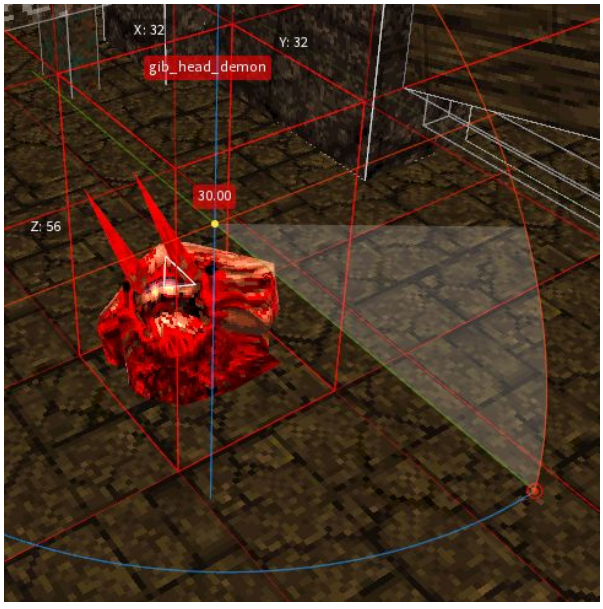
NOTE: To have a continuously firing bolt between two points, have a ltrail_start and ltrail_end targeting each other in a loop and set frags to -1. The sound this makes is not ideal, so consider making these silent and use a play_sound_triggered with a custom looping sound. This is shown in the pd_lasers 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 make 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 make 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.

tele_fog

When triggered, tele_fog plays the teleport particle effects and sound. Use this when killtargeting an entity if the player can see it happen. You can see an example of this near the trigger_use key entity in *The Gallery* map.

Worldspawn

Added a *reset_items* key. Set to 1 to make the player start with default shotgun and axe.

light_candle

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

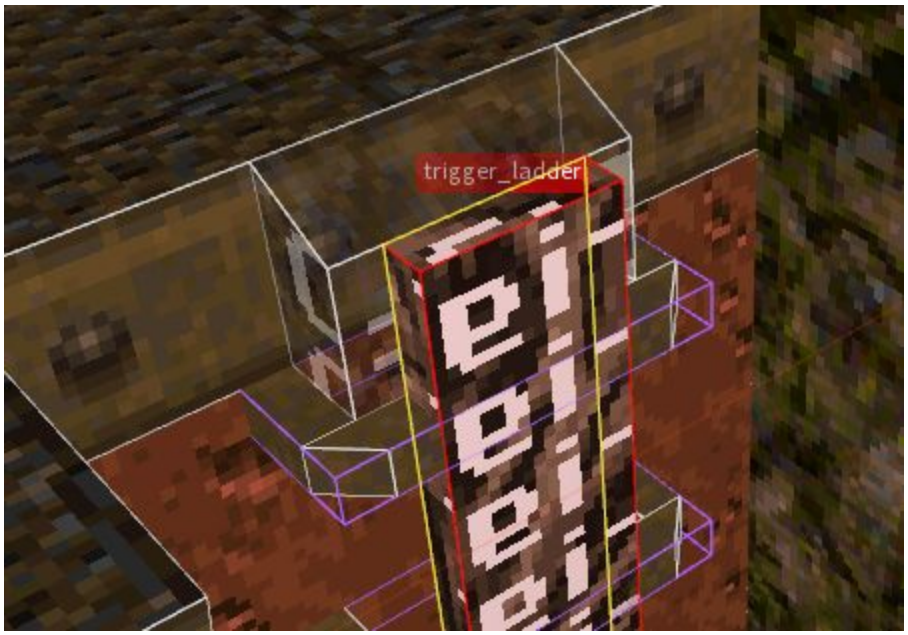
meat_shower

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

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 pd_ladders.map for examples.



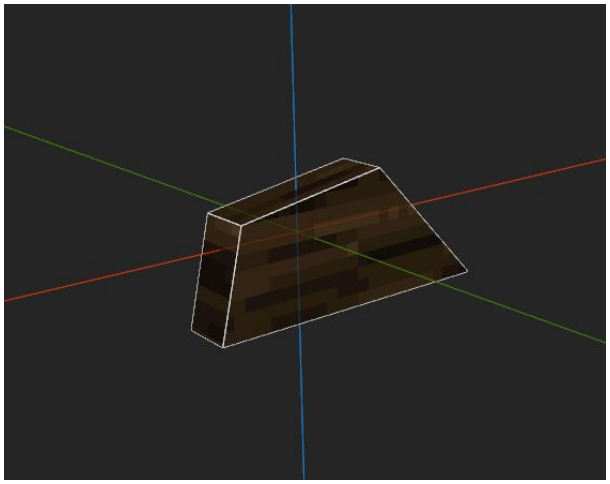
Breakables

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 three options: 0 = Green Metal, 1 = Red Metal and 2 = Concrete. By default, the breakable with 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.

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. The 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.

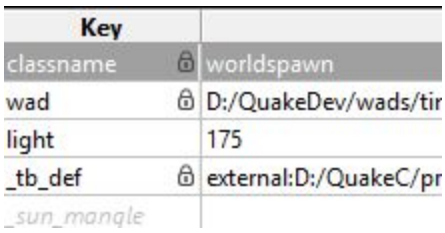
Key	Value
classname	func_breakable
break_template1	maps/debris/wood1.bsp
brk_obj_count1	5
spawnflags	4
break_template2	
break_template3	
break_template4	
break_template5	
brk_obj_count2	
brk_obj_count3	
brk_obj_count4	
brk_obj_count5	
cnt	*5*
dmg	*20*
health	*20*

+ − ☑ Show default properties

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



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.

Enhanced Zombies

Zombies have more options in progs_dump 1.1.0. First off, there are motionless and 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:

Effect Entities

In addition to the **tele_fog** and **meat_shower** effects you can now trigger the following visual effects.

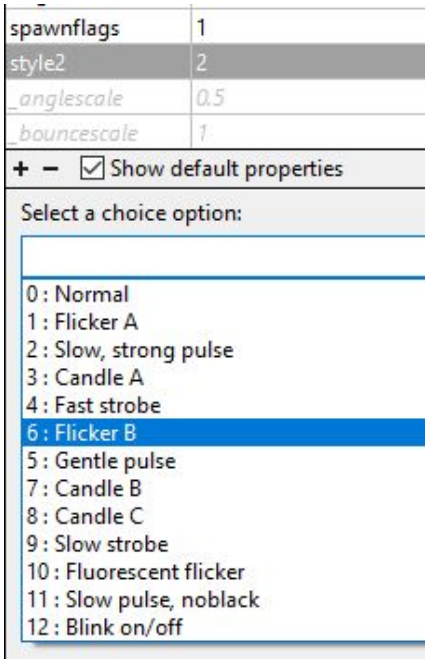
Effect	**Details**
play_explosion	grenade explosion, causes damage
play_tbabyexplode	Spawn death explosion, causes damage
play_lavalsplash	large particle effect, can have custom sound

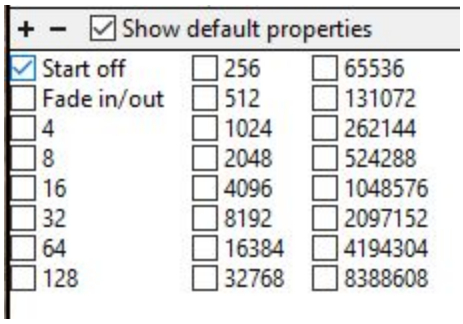
Switchable Animated Light Styles

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.





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.

Particle Effects

misc_sparks

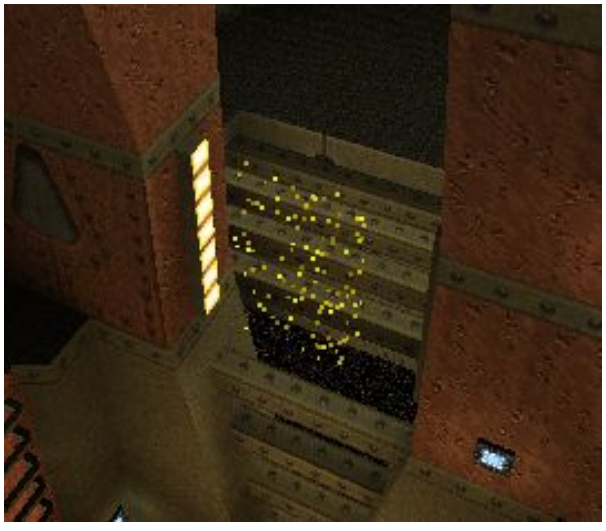
Produces a burst of yellow sparks at random intervals. If targeted, it will toggle between on or off. If it targets a light, that light will flash along with each burst of sparks. NOTE: targeted lights should be set to START_OFF. Spawnflags = SPARKS_BLUE: sparks are blue in color SPARKS_PALE sparks are 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

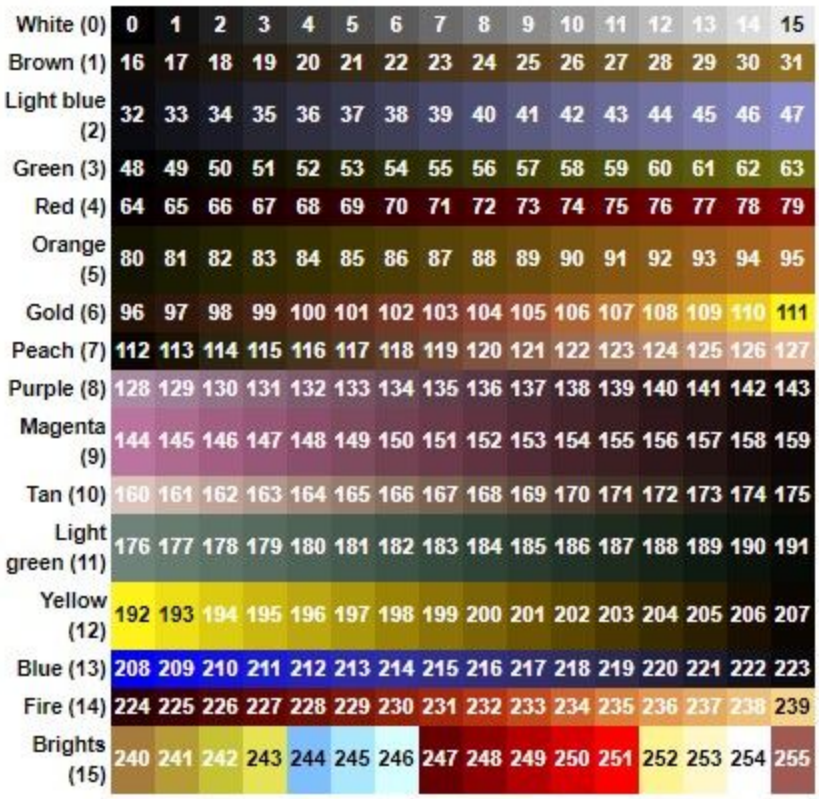
A particle stream! It appears when triggered. This entity is one end of the stream, target another entity as the other end-point. Usually an *info_notnull*, but you should be able to target anything (like monsters). *target* = This entities origin is the end-point of the stream *dmg* = 1st Color, use 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:



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.

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! 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, 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 train to rotate at rate specified by "rotate". Use '0 0 0' to stop rotation. ANGLES tells 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 to 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 train stops. "noise1" contains the name of the sound to play when the train moves. Both "noise" and "noise1" defaults depend upon "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 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_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
start	n/a	

Credits

QuakeC

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/

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 Johnny Law
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 code by Simon O'Callaghan
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

Rubicon 2 code 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/

Slipgate by Michael Coburn
https://github.com/c0burn/Slipgate

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

Map Credits

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_meat
pd_void
pd_zombies