EntityPlus Manual

based on version 1.1.10

Manual by Eraser
eraesr@gmail.com
https://github.com/TheEnginesOfCreation/EntityPlus

Introduction....3
1 – Changes to the single player game....4
1.1 – Player spawning and dying....4
1.2 – Bots as enemies....4
1.3 – Some other changes....5
1.4 – Difficulty levels....5
1.5 – Scoring....6
2 – New entities....8
2.1 – func_breakable....8
2.2 – func_door_rotating....9
2.3 – holdable_key_*....9
2.4 – info_backpack....10
2.5 – info_camera....10
2.6 – info_waypoint....11
2.7 – item_armor_vest....11
2.8 – shooter_bfg....12
2.9 – target_botremove....12
2.10 – target_botspawn....13
2.11 - target_cutscene....14
2.12 – target_debrisemitter....15
2.13 – target_earthquake....15
2.14 – target_effect....16
2.15 – target_finish....17
2.16 – target_gravity....17
2.17 – target_laser....17
2.18 – target_logic....18
2.19 – target_mapchange....18
2.20 – target_modify....18
2.21 – target_music....19
2.22 – target_objective....19
2.23 – target_playerspeed....19
2.24 – target_playerstats....19
2.25 – target_secret....20
2.26 – target_skill....20
2.27 – target_unlink....20
2.28 – target_variable....20
2.29 – trigger_death....21
2.30 – trigger_frag....21
2.31 – trigger_lock....22
3 – Modified entities....23
3.1 – worldspawn....23
3.2 – func_bobbing....23
3.3 – func_button....23
3.4 – func_door....24
3.5 – func_rotating....24
3.6 – func_static....24
3.7 – func_timer....24

3.8 – func_train....25
3.9 – holdable_teleporter....25
3.10 – path_corner....25
*3.11 – shooter_**....25
3.12 – target_delay....25
3.13 – target_give....26
3.14 – target_print....26
3.15 – target_push....26
3.1*6* – target_relay....26
3.1*7* – target_remove_powerups....26
3.18 – target_speaker....27
3.1*9* – trigger_hurt....27
3.*20* – trigger_push....27
3.*21* – trigger_teleport....27
3.*22* – Item pickup entities....27
3.*23* – Spawnpoints....27
3.*24* – targeting entities....28
3.*25* – notep key....28
3.*26* – Constant light on func_* entities....28
4 – Miscellaneous info for level designers....30
4.1 – Configuring GtkRadiant....30
4.2 – Creating ladders....31
4.3 – Fade-in....31
4.4 – The effect of sv_maxclients....31
4.5 – Creating an .arena file for your map....31
4.6 – surfaceparm flesh....32
5 – New cvars and console commands....33

Introduction

This document describes the various additions and changes that the EntityPlus mod makes to the standard Quake III Arena game. The aim of the mod is to provide an expanded set of tools which level designers can use to create single player experiences in Quake III Arena. While Quake III Arena offers a single player game, it is little more than a multi-player deathmatch game against AI controlled opponents. EntityPlus' single player game will be more traditional in the sense that the player is tasked with reaching the end of the level, solving puzzles and completing objectives along the way. Opposition will be there in the form of AI controlled enemies. The final experience will be more akin to Quake or Quake 2's single player campaign than that of Quake III Arena. The goal of the mod is not to provide such levels but to provide the means necessary to build them.

This document can be used as a guide to building such levels using the tools that EntityPlus offers. It describes the changes EntityPlus makes to the game to achieve it's goal and the tools a level designer can use to create a compelling single player experience.

This document assumes you have complete knowledge of level design in Quake III Arena using a map editor like GtkRadiant. It assumes you have knowledge of what entities are and how they work.

Important bits of information that are easily missed within all the text are highlighted through attention boxes. These are separate text boxes that contain important facts in a bigger font.

1 – Changes to the single player game

EntityPlus is a single player Quake 3 mod pur sang. This means that Free For All deathmatch, Team Deathmatch, Capture the Flag or any other multiplayer gametypes are not part of EntityPlus.

Because multiplayer is no part of EntityPlus, multiplayer related options have been removed from all of the in-game menus. Level designers will also notice that the entity definitions no longer have gametype related keys associated to them.

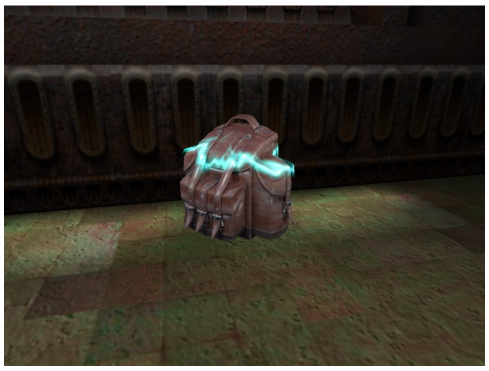
1.1 – Player spawning and dying

When a map is loaded, the player is spawned into the map at any info_player_start entity. The player is spawned with a Gauntlet and a Machinegun with 100 ammo and full health as usual.

When the player dies, he will be respawned at the *nearest* spawnpoint (info_player_start entity). This is a subtle but important difference from vanilla Quake 3, where the player would respawn at the farthest spawnpoint. This allows the level designer to have more control over where a player respawns after he died. More on controlling respawn behavior is discussed later in this document.

“Players respawn at the *nearest* spawnpoint”

There is another difference when players die. In vanilla Quake 3, a dead player would drop the weapon he or she held at the time of dying. In EntityPlus the player will drop a backpack instead of a weapon. This backpack holds the complete inventory of the player at the moment of dying. This includes the player's weapons, the ammo for those weapons and any holdable items (such as keys, the medkit or the personal teleporter powerup). It does not include temporary powerups such as a Quad Damage or Battle Suit. The backpack will remain at the spot where the player died until it is picked up again by the player. Upon picking up the backpack, all items in the backpack are given to the player again. Any ammo in the backpack is added on top of the ammo the player currently holds. Do note that the backpack contains 100 bullets (for the MG) less than what the player died with, with a minimum of 0 of course. This is done because the player respawns with 100 “free” bullets.



The goal of this mechanic is to have a consistent world experience that is uninterrupted by loading a savegame or something similar. The player will re-enter the world as it is when the player died, but in a relatively weak state. The priority will shift to retrieving the backpack before the player is willing to progress.

Because players sometimes die in unreachable areas (such as the void or lava) it can occur that the backpack sits in a location where it cannot be retrieved anymore. Read the description of the info_backpack entity for a solution to this problem.

1.2 – Bots as enemies

A single player FPS game is no fun without any bad guys to shoot. The current implementation of EntityPlus uses the Quake III Arena bots as enemies for the game. How this exactly works with spawning bots into the level is covered in the *target_botspawn* section. Here, some basics on how the bots are handled by the game are covered.

The AI of the bots in EntityPlus is currently not very different from the AI of the bots in vanilla Quake 3. In single player mode, a few changed have been made though:

- Bots don't attack each other. They don't see each other as a target that should be attacked.
- Bots can not accidentally hurt each other. They are invincible to each others weapon fire.
- A bot that dies is removed from the game. This means that a fight with a bot is a one-off thing. When the bot is dead, he stays dead.
- Bots don't drop any items when they die, unless the “loottarget” key is set on the relevant target_botspawn entity (see the chapter on target_botspawn for more info).
- Bots don't chat to keep up the illusion of them being single player style “monsters”.
- Bots cannot pick up any items in the level.

All these changes make the bots more believable as single player enemies. The idea is that through the use of target_botspawn entities, bots can be used as more traditional single player enemies.

1.3 – Some other changes

Various other changes have been applied to the mod to create a more believable single player experience. It is easiest to simply list the changes as a few bullet points:

- Non-essential textual feedback is removed. This includes on-screen messages such as countdowns for match starting, the dual score scoreboard on bottom right of the HUD, the names of enemies over the crosshair, console messages about players (bots) joining or leaving the game, console messages about who fragged who (also known as “obituaries”).
- Removed other visual feedback such as the score plums when a frag is made and awards (such as humiliation, impressive and excellent)
- Removed non-essential aural feedback. This includes the announcer voice announcing bot names that enter the game, the announcer voice warning about frag- or timelimits, the beep sound when a fired shot hits an enemy.
- Items do not automatically respawn anymore. If you want an item to respawn, you should specify a wait key.
- When pressing the TAB key, the scoreboard is no longer shown. Instead, the player is presented an overview of his current objectives. There is a primary and a secondary objective. See the topic on target_objective for more info on setting objectives.
- The main and single player menus are completely redesigned.
- A new (minimalistic) loading screen is introduced, which no longer shows which items are being loaded so it doesn't spoil the contents of the level.
- Ammo limits are no longer set at 200 which means the player can carry a maximum of 999 ammo for all weapons.

“Items do not automatically respawn”

1.4 – Difficulty levels

Difficulty levels are hardcoded into the game itself. The difficulty can be set using the `g_spskill` cvar, where 1 is the easiest (“Beginner”) and 5 is the hardest (“Nightmare”). Bots become tougher as the difficulty increase, similar to standard Quake 3 games.

In EntityPlus, the bots will deal less damage to prevent the game from becoming too hard. The amount of damage bots do depends on the difficulty level. Currently damage is distributed among the difficulty levels as follows:

1 (Beginner): bots deal 5% of the original damage.

2 (Easy): bots deal 15% of the original damage.

3 (Medium): bots deal 25% of the original damage.

4 (Hard): bots deal 35% of the original damage.

5 (Nightmare): bots deal 45% of the original damage.

For any skill level, the percentage of damage a bot does is calculated like this:

(skill * 0.1) – 0.05

Simply said, for any 1 point of skill increase, the bot does an additional 10 percentage points of damage.

A similar system is used for the bot's health. A bot will spawn with 100 health by default, but it is up to the level designer to tweak the health a bot has to make them into well balanced enemies, as 100 health is often way too much. The skill level of the bot, however, influences the amount of health the bot has. The health is distributed as follows:

1 (Beginner): bots have 60% of their original health.

2 (Easy): bots have 80% of their original health.

3 (Medium): bots have 100% of their original health.

4 (Hard): bots have 120% of their original health.

5 (Nightmare): bots have 140% of their original health.

For any skill level, the percentage of health a bot has is calculated like this:

1 + ((skill – 3) * 0.2)

Simply said, for any 1 point the bot's skill level is below 3, the bot loses 20% of health. For any 1 point the bot's skill level is above 3, the bot gains 20% of health.

While a bots skill level is initially based on the value assigned to g_spskill, the target_botspawn entity does have a “skill” key which can be used to determine an individual bot's relative skill level. A bot's damage output and health can rise above the values for skill level 5 when the skill key is used.

1.5 – Scoring

EntityPlus introduces a scoring system that will allow players to score points through actions in the game. The highest score will be saved to a file on disk and this high score will be displayed in the single player level select menu. See the chapter on the target_finish entity for more information on how to use the scoring system in your map.

The way a player's score is calculated works as follows. A player can win or lose points based on the following areas: damage dealt (referred to as “carnage” in-game), number of deaths, accuracy, secrets found, skill or in-level entities that add score.

Carnage score:

Every time the player kills an enemy, the amount of health that enemy spawned with is added to the carnage score of the player. Also, when the player hits a target_score entity, the score of this entity is added to the carnage score.

Accuracy bonus:

The accuracy bonus is a bonus score that is based on the player's carnage score and the aim accuracy of the player in that level. The accuracy itself is expressed as a percentage from 0% (none of the shots were hits) to 100% (every shot hit an enemy or breakable object).

To calculate the accuracy bonus, the game takes 50% of the carnage score and takes a percentage from that equal to the accuracy percentage.

This basically means that if a player has a carnage score of 1500 points and he has an accuracy of 60%, then the accuracy bonus will be worth 450 points (1500 / 2 * 60%).

Skill bonus:

The current skill level the player is playing at results in a bonus as well. This bonus is a percentage of the carnage score of the player. Playing at a higher skill level increases the percentage of the bonus:

Beginner (easiest) : 0%

Easy : 5%

Medium: 10%

Hard: 15%

Nightmare (hardest): 20%

Secrets bonus:

A level designer can hide secret areas in his map. Such an area can be marked with a trigger and a target_secret entity. When the target_secret entity is activated, the secret is found and the player receives a secrets bonus. This secrets bonus equals 5% of the player's carnage score for each secret found. So if a player has a carnage score of 1200 and three secrets are found, the secrets bonus will be equal to 180 (15% of 1200).

Deaths penalty:

Dying will cause the player to incur a “death penalty” at the end of the map. This death penalty will be 5% of the player's carnage score for each time that the player died. So if the player has a carnage score of 1000 and he died twice, his death penalty will be 100 points.

Total score:

The total score of a player is equal to the carnage score + accuracy bonus + skill bonus + secrets bonus + deaths penalty.

2 – New entities

This section of the document describes the new entities that are part of EntityPlus.

2.1 – func_breakable

The func_breakable entity is an entity that is applied to a brush or set of brushes, similar to most other func_ entities. A func_breakable will look like an ordinary brush and acts similar as a func_static. The difference between func_static and func_breakable is that a func_breakable can be shot to “destroy” the entity. When a set amount of damage has been inflicted to the entity, it will disappear, optionally leaving behind debris. It can also target another entity which will be triggered when the func_breakable is destroyed. If none of the debris type spawnflags is selected, the func_breakable will revert to DEBRIS_LIGHT.

func_breakable entities can damage other func_breakable entities when the dmg key is set. However, when a func_breakable receives enough damage from another func_breakable so that it'll break, the breaking happens with a slight delay. The radius in which a func_breakable deals damage can be set with the radius key. The default radius of 120 is equal to the radius of an exploding rocket.

Keys:

- health : The entire entity is removed once this amount of damages has been inflicted to it.
- count : The entity will spew out this number of chunks of debris when it's broken (default = 0).
- target : Targetted entity will be triggered once the func_breakable is destroyed.
- dmg : Amount of splashdamage the func_breakable will deal when it is removed (default = 0).
- radius : The radius in which the splashdamage is dealt (default = 120).
- breaksound : Sound to play when it's broken (by default, no sound will be played)
- target : Targetted entity will be triggered once the func_breakable is destroyed.
- target2 : Targetted entity will be triggered once the func_breakable is destroyed.
- targetname : activating entity points to this.
- targetname2 : activating entity points to this.

Spawnflags:

- DEBRIS_LIGHT : Emit light colored pieces of concrete debris.
- DEBRIS_DARK : Emit dark colored pieces of concrete debris.
- DEBRIS_LIGHT_LARGE : Emit large light colored pieces of concrete debris.
- DEBRIS_DARK_LARGE: Emit large dark colored pieces of concrete debris.
- DEBRIS_WOOD : Emit pieces of wooden debris.
- DEBRIS_FLESH : Emit gibs.
- DEBRIS_GLASS : Emit shards of glass.
- DEBRIS_STONE : Emit chunks of stone.
- NO_PLAYER : The player cannot damage the func_breakable.
- NO_BOTS : Bots cannot damage the func_breakable.
- NO_SHOOTER : Shooter entities cannot damage the func_breakable.
- EXPLOSION : When the func_breakable is destroyed, an explosion effect is shown.

2.2 – func_door_rotating

This is a door that rotates instead of slides when it is activated.

Keys:

- distance : determines how many degrees the door will rotate (default 90).
- speed : determines how fast the door moves (degrees/second).
- wait : number of seconds before door returns (default 2, -1 = return immediately, -2 return only when triggered again)
- targetname : if set, a func_button or trigger is required to activate the door.
- targetname2 : if set, a func_button or trigger is required to activate the door.
- dmg : damage to inflict on player when he blocks operation of door (default 2). Door will reverse direction when blocked unless CRUSHER spawnflag is set.
- health : (default 0) if set to any non-zero value, the button must take damage (any amount) to activate.
- team : assign the same team name to multiple doors that should operate together (see Notes).
- startsound : if set, overrides the sound to play when the door starts moving.
- endsound : if set, overrides the sound to play when the door has stopped moving.
- nobots : bots cannot activate this door.
- nohumans : human player cannot activate this door.
- light : constantLight radius of .md3 model included with entity. Has no effect on the entity's brushes (default 0).
- color : constantLight color of .md3 model included with entity. Has no effect on the entity's brushes (default 1 1 1).
- model2 : path/name of model to include (eg: models/mapobjects/pipe/pipe02.md3).
- origin : alternate method of setting XYZ origin of .md3 model included with entity (See Notes).

Spawnflags:

- START_OPEN : the door will spawn in the open state and operate in reverse.
- CRUSHER : door will not reverse direction when blocked and will keep damaging player until he dies or gets out of the way.
- REVERSE : opens the door in the other direction.
- X_AXIS : open on the x axis instead of the x axis
- Y_AXIS : open on the y axis instead of the x axis

2.3 – holdable_key_*

There are a number of new entities whose names begin with "holdable_key_". These entities act as keys that the player can pick up. There are eight different keys in total. These are:

- holdable_key_blue
- holdable_key_green
- holdable_key_red
- holdable_key_yellow

- holdable_key_iron
- holdable_key_silver
- holdable_key_gold
- holdable_key_master

The first four keys are keycards with a high tech look that best fit levels with a tech look. The last four keys are keys that really look like keys that have a somewhat magical touch to them. These are best used in gothic style levels.

Unlike the holdable_teleporter and holdable_medkit, the player can carry more than one key at the same time, but only one key of each type. So a player could carry the iron key and the gold key simultaneously, but not carry two gold keys. Having a key in your inventory also does not prohibit the player from picking up a holdable_medkit or a holdable_teleporter. However, the player can still only carry only one of these two powerups.

Keys work in conjunction with the trigger_lock entity. See the topic on trigger_lock for more information.

Keys:

- wait : time in seconds before item respawns after being picked up (default 60, -1 = never respawn).
- random : random time variance in seconds added or subtracted from "wait" delay (default 0).
- team : set this to team items. Teamed items will respawn randomly after team master is picked up (see Notes).
- target : picking up the item will trigger the entity this points to.
- targetname : a target_give entity can point to this for respawn freebies.

Spawnflags:

- SUSPENDED : item will spawn where it was placed in map and won't drop to the floor.

2.4 – info_backpack

In EntityPlus, the player will drop a backpack if he dies. This backpack can be retrieved to restore the inventory (weapons/ammo/keys/powerups) the player had when he died. Because it is possible for players to die in unreachable areas (such as the void, lava or slime), the backpack can essentially become irretrievable. Because this is unwanted in many cases, such areas can be filled with a brush that have the “common/nodrop” shader. As soon as the backpack hits a nodrop brush, the game will find the nearest info_backpack entity and teleport the backpack straight to that location. The info_backpack entity could, for instance, be placed at the edge of the pit into which the player can fall so that the backpack itself will sit there when the player returns.

2.5 – info_camera

The info_camera entity functions as a camera view for in-game cutscenes. Such cutscenes can be started by using the target_cutscene entity. The info_camera entity determines the position of the camera and the direction in which the camera is aimed. A camera can either cut directly to the next camera or pan linearly from one to the next. The "wait" key of the camera determines how long this camera position will be held or how long the panning movement will last. If a camera does not target another info_camera the cutscene ends after the wait period and the player regains control. Multiple cameras can be linked together by making the first camera target the second camera. Note that the last camera will never pan which means that if the cutscene should end with a panning movement, the last camera's wait key should explicitly be set to 0.

While "aiming" a camera can be done using the "angles" key, it is much easier to use an info_notnull or target_position entity which is targeted by the info_camera. The camera will be aimed at the info_notnull or target_position entity. This works identical to how info_player_intermission works in that regard. One detail to keep in mind is that if you use the "target" key to target another info_camera, the target2 key must be used for aiming the camera (or vice versa).

It is possible to target multiple info_notnull or target_positions (by giving them all the same targetname) but the camera will pick one randomly selected target.

It is possible to write camera debug info to the console when a map is loaded. See the description for the g_debugCameras cvar in the cvars section of this document.

Keys:

- targetname : activating entity points to this.
- targetname2 : activating entity points to this.
- target : this points to the next camera to jump to in the cutscene.
- target2 : this points to an info_notnull or target_position entity to aim the view of the camera.
- fov : zoom fov for this camera.
- wait : number of seconds this camera will hold the view (default 1).
- angles: alternate "pitch, yaw, roll" angles method of aiming cutscene camera (default 0 0 0).

Spawnflags:

- PAN : Instead of jumping to the next camera position, the camera view pans linearly to the next camera position.

2.6 – info_waypoint

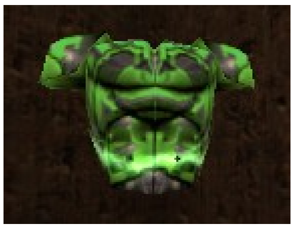
The info_waypoint entity can be used to plot patrol routes for bots. A target_botspawn can refer to an info_waypoint entity through its target key. A bot spawned at this botspawn will then move towards the info_waypoint entity. An info_waypoint entity itself can point towards another info_waypoint entity. The bot will move to the second info_waypoint entity after reaching the first. info_waypoint entities can be used to create a patrol route with a clear beginning and end, in which case the bot will move back and forth along this path, or it can be a circular path, in which case the bot will continue moving in circles.

As soon as the bot that follows the path sees the player, he will break free of the patrolling path and attack the player.

Keys:

- target : point at another info_waypoint to chain waypoints together
- targetname : target another info_waypoint or a target_botspawn entity at this to make a bot move to this waypoint after reaching the previous waypoint or spawning.
- wait : number of seconds bot will wait at waypoint before moving on. -1 will make the bot wait indefinitely. When attacked, the bot will still break free and return fire (default 0).

2.7 – item_armor_vest



This is an item similar to the red and yellow armors already found in Quake III Arena. The only difference is that this one is green and gives you 25 points of armor.

Keys:

- wait : time in seconds before item respawns after being picked up (default 25, -1 = never respawn).
- random : random time variance in seconds added or subtracted from "wait" delay (default 0 - see Notes).
- team : set this to team items. Teamed items will respawn randomly after team master is picked up (see Notes).
- target : picking up the item will trigger the entity this points to.
- targetname : a target_give entity can point to this for respawn freebies.

Spawnflags:

- SUSPENDED : item will spawn where it was placed in map and won't drop to the floor.

2.8 – shooter_bfg

This entity was put in for completeness sake. It functions in the same way as shooter_rocket, shooter_grenade and shooter_plasma but fires a BFG bolt instead. Note that a BFG bolt does equal damage to a rocket, so this is really just a visual variant of shooter_rocket.

Keys:

- angles: this sets the pitch and yaw aiming angles of shooter (default 0 0). The roll angle does not apply.
- targetname : activating trigger points to this.
- target : this points to a target_position entity for aiming the bfg shots.
- random : random aiming variance in degrees from the straight line to the targeted entity (default 0 - see Notes).
- dmg: amount of damage a bfg projectile will deal (default 100).

Spawnflags:

- TARGET_PLAYER : The shooter will always aim directly at the player.
- NO_BOTS : The shooter will not harm bots
- NO_HUMANS : The shooter will not harm the player.

2.9 – target_botremove

This entity is used to remove bots that were spawned by target_botspawn entities from the game. It can target a target_botspawn and when activated, it will remove all bots from the game that were spawned by the targeted target_botspawn entity. By default, removed bots will simply disappear from the game instantly. When using the LETHAL_INJECTION and SPONTANEOUS_COMBUSTION spawnflags, the bots will not immediately disappear but either die on the spot or gib.

Keys:

- target : this points to a target_botspawn entity from which all bots should be removed from the game.
- target2 : this points to a target_botspawn entity from which all bots should be removed from the game.
- targetname : activating entity points to this.
- targetname2 : activating entity points to this.

Spawnflags

- LETHAL_INJECTION : Instead of being removed from the game immediately, the bot will die.
- SPONTANEOUS_COMBUSTION : Instead of being removed from the game immediately, the bot will explode in a shower of gibs.

2.10 – target_botspawn

The target_botspawn entity is one of the most important entities for creating single player levels. It allows the level designer to spawn bots into the game which act as bad guys that need to be taken care of before the player can progress.

The target_botspawn should be positioned as a spawnpoint in the position where the bot should spawn. The angle key can be used to determine which way the bot will face when it spawns. Whenever the target_botspawn is triggered by the entity that is targetting it (this could be a trigger_multiple, a func_button, etc) the specified bot will be spawned into the game at that position with the specified amount of health (through the health key of the entity) and the defined weapons loadout (this can be controlled through the various WP_* spawnflags).

When using bots in your level, it is important to remind yourself of the fact that it is still the Quake III Arena deathmatch AI powering these bots. Each bot has it's own definition files in which it's behavior and weapon preferences are determined. This means that if you choose to spawn a specific bot with a rocket launcher and a shotgun, it might opt to use the shotgun at all times if its weapon preference for the SG is a lot higher than that of the RL.

Bots are removed from the game after they are fragged. This means that bots are more usable as single player enemies (they don't keep returning after they're dead). It is, of course, possible to trigger a target_botspawn multiple times.

“Bots are removed from the game after they are fragged”

The use of info_waypoint entities in combination with target_botspawn is a powerful and important way of making bots patrol certain areas. This gives them a far more realistic behavior for single player games.

The skill level of the bots is determined by the g_spskill cvar, which ranges from 1 (easiest) to 5 (hardest). target_botspawn allows the level designer to apply a relative skill level for the bot though. This can be done through the “skill” key, which adds the value to the skill level of the bot. Negative values make bots easier, positive values make bots harder. Decimal values (such as 1.5 or -2.7) are allowed. A bot's skill level will never drop below 1 but may rise above 5. For instance, when playing at skill level 4 (Hardcore) a target_botspawn with a relative skill level of 2 will spawn bots that have a skill of 6. Note that a bot's skill level influences two things:

1 – Its AI level, which is just the general skill level of the bot like it is in regular Quake 3 as well.

2 – The amount of damage a bot can do.

Because bot AI skill levels do not go above 5 or below 1, the bot's AI skill level is bound between these values. This means that when playing at skill level 4, a bot with relative skill 2 will still have an AI skill level of 5.

The amount of damage a bot can do has a lower skill level limit of 1 but it has no upper limit. This means our bot with skill level 2 will deal as much damage as a level 6 bot would. See chapter 1.4 (Difficulty levels) about damage dealing.

Bots can drop items (weapons, ammo, health, armor or powerups) or holdables (medkit, teleporter or keys) by making the target_botspawn’s *loottarget* key target the desired item or holdable. The targeted item or holdable won’t spawn into the game, but will appear over the bot’s corpse when it dies.

Keys:

- angle : direction in which bot will look when spawning in the game.
- clientname : the name of the bot to spawn. When omitted it defaults to Sarge.
- health : the amount of health the bot will have when it spawns into the game.
- skill : sets the relative skill level of the bot (default = 0).
- target : this points to entities to activate when this entity is triggered.
- targetname : activating trigger points to this.
- deathtarget: this points to entities to activate when a bot spawned by the target_botspawn dies.
- loottarget: this points to any item or holdable entity. The item will be dropped by the bot when it dies.

Spawnflags:

- WP_GAUNTLET: The bot spawns into the game with a Gauntlet.
- WP_MACHINEGUN : The bot spawns into the game with a Machinegun.
- WP_SHOTGUN : The bot spawns into the game with a Shotgun.
- WP_GRENADE_LAUNCHER : The bot spawns into the game with a Grenade Launcher.
- WP_ROCKET_LAUNCHER : The bot spawns into the game with a Rocket Launcher.
- WP_LIGHTNING : The bot spawns into the game with a Lightninggun.
- WP_RAILGUN : The bot spawns into the game with a Railgun.
- WP_PLASMAGUN : The bot spawns into the game with a Plasmagun.
- WP_BFG : The bot spawns into the game with a BFG.
- IGNORE_PLAYER : The bot ignores the player, even when attacked.
- PATROL_WALK : The bot will walk (instead of run) while patrolling along info_waypoint entities.
- ALWAYS_WALK : The bot will always walk (instead of run).
- SPAWN_EFFECT: Show a (teleport) effect when the bot spawns.

2.11 - target_cutscene

target_cutscene entities can be used to show cutscenes. Cutscenes in EntityPlus are basically a sequence of camera positions to which the player's view will jump after set amounts of time. Camera positions are defined by info_camera entities.

A target_cutscene entity should always target an info_camera entity using its target or target2 keys. Without at least one info_camera entity, target_cutscene is useless.

The info_camera entity that the target_cutscene points to is the first camera position for the cutscene. Note that if the target_cutscene targets multiple info_camera entities, the first info_camera entity that is found will be used as the first camera position. All other camera entities the target_cutscene points to are ignored.

The game will continue running while the cutscene is playing, so any enemies will remain active. By setting the HALT_AI spawnflag, all enemies can be frozen in place while the cutscene is running.

Because of a serious problem with running cutscenes while g_synchronousClients is enabled (the game will crash), the game will automatically disable g_synchronousClients while an in-game cutscene is played. This means that playback of the cutscene in a recorded demo is slightly less smooth than normal gameplay. The g_allowSyncCutscene cvar may be used to override this behavior, but it is not recommended.

Keys:

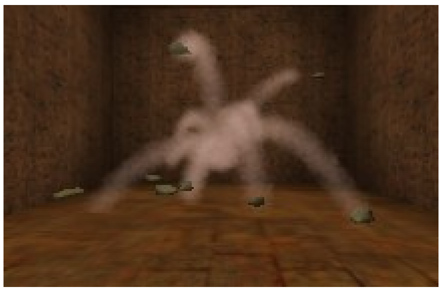
- target : this points to an info_camera entity to which the camera view should go.
- target2 : this points to an info_camera entity to which the camera view should go.
- targetname : activating entity points to this.
- targetname2 : activating entity points to this.

Spawnflags

- HALT_AI : Freezes all enemies in place while the cutscene is running.
- START_AT_PLAYER : The first info_camera's position and viewing angles are ignored and replaced with the player's position and viewing angles at the time of activating the cutscene.

2.12 – target_debrisemitter

The target_debrisemitter is an entity that's used purely for visual effects. It will, when triggered, spew out a bunch of debris pieces which fly out from the entity's position with some smoke, dust or blood trailing behind them. There are a few variations on the effect, ranging from pieces of concrete to bits of wood, to gibs.



Keys:

- targetname : activating trigger points to this.
- count : The number of debris pieces to emit (default = 10).

Spawnflags:

- DEBRIS_LIGHT : Emit light colored pieces of concrete debris.
- DEBRIS_DARK : Emit dark colored pieces of concrete debris.
- DEBRIS_LIGHT_LARGE : Emit large light colored pieces of concrete debris.
- DEBRIS_DARK_LARGE: Emit large dark colored pieces of concrete debris.
- DEBRIS_WOOD : Emit pieces of wooden debris.
- DEBRIS_FLESH : Emit gibs.
- DEBRIS_GLASS : Emit shards of glass.
- DEBRIS_STONE : Emit chunks of stone.

2.13 – target_earthquake

When triggered, the player will experience an earthquake, The screen will be shaken while the earthquake occurs.

Keys:

- targetname : activating trigger points to this.
- length : duration of the earthquake in seconds (2 - 32)
- intensity : intensity of the earthquake (1 - 16)

2.14 – target_effect

This entity is used to create various in-game effects. A number of effect types are available: an explosion effect, four particle effects and an overlay effect. The explosion effect simply shows an explosion at the position of the entity when it is triggered. The particle effect shows a burst of particles being emitted from the entity. The color of the particles and the number of particles can be configured.

The OVERLAY effect can be used to draw an image over the entire screen. It is advised to use a TGA image with smart use of the alpha channel for this, as any non-transparent pixels will completely block the player's view. Note that due to the way graphics cards work, images should preferably have dimensions that are in powers of two, such as 128, 256, or 512 pixels. It is also important to know that your overlay image will be drawn on a reference canvas of 640x480 pixels, meaning that any images higher than 480 pixels or wider than 640 pixels simply do not fit the canvas. After this canvas is drawn, it is stretched to the resolution the game is running at. Therefore it is advised not to attempt to put too many intricate details into your image, as they will most likely not scale well to higher resolutions.

The image shows an example of target_effect being used to add a vignetting effect to the screen.

The FADE effect is used to create a screen fade effect. A start color and end color (including transparency) and a duration can be defined. The fade effect will be drawn layered on top of any overlays but under the HUD layer.

Keys:

- targetname : activating trigger points to this.
- color : color of the particles or smokepuff (default 1.0 1.0 1.0).
- count : number of particles for particle effect (0 - 255, default 100).
- speed : speed of the particles (default 100) or smokepuff (default 16).
- overlay : specifies an image to overlay over the screen. Leave empty to remove existing overlay.
- startcolor : specifies the rgba value of the starting color of a fade (default 0.0 0.0 0.0 0.0).
- endcolor : specifies the rgba value of the ending color of a fade (default 0.0 0.0 0.0 1.0).
- wait : When using FADE, this specifies the amount of time it takes to fade from the start color to the end color. When using SMOKEPUFF, this specifies the duration the smokepuff stays visible. (default 2)
- scale : specifies the size scale factor of smokepuffs (default 1).

Spawnflags

- EXPLOSION : Shows an explosion when activated.
- PARTICLES_GRAVITY : Generates particles that are affected by gravity.
- PARTICLES_LINEAR : Generates particles that are not affected by gravity.
- PARTICLES_LINEAR_UP : Generates particles that are not affected by gravity that only move upwards.
- PARTICLES_LINEAR_DOWN : Generates particles that are not affected by gravity that only move downwards.
- OVERLAY : Applies the specified overlay to the screen.
- FADE : The screen will fade from the specified start color to the specified end color.
- SMOKEPUFF : Emit a single puff of smoke.

2.15 – target_finish

This entity will, when triggered, completely ends the game. It will move the player to the intermission screen, which will display a number of variables that add up to the player's final score for that level. When the player clicks a mouse button to continue, the game will return to the main menu. The entity will also compare the player's current score to the highscore for that level. If the player's score is higher, then this score will be saved as the new highscore. When the intermission screen is displayed, the camera is moved to the intermission camera position that can be defined using an info_player_intermission entity as usual.

An important aspect of this entity is how it works in combination with target_mapchange. Because it should be possible to create a single player campaign that stretches across multiple maps, any score accumulated in consecutive maps after the first one are attributed to the first map. In other words, if we start map1 through the single player menu, hit a target_mapchange there which loads up map2 and map2 contains a target_finish, the total highscore (which is built up across both maps) is stored for map1. This means that such a multi-level campaign is considered as a single level when it comes to scoring. Note that this entity can only be triggered by human players (not bots).

Keys:

- targetname : activating trigger points to this.

2.16 – target_gravity

The target_gravity entity can be used to set the gravity for one or all players (including bots) in the game. The default gravity value for Quake 3 Arena is 800.

Keys:

- count : amount of gravity to apply (defaults to the value set to g_gravity).
- targetname : activating trigger points to this.

Spawnflags:

- GLOBAL : all players are affected instead of only the activator.

2.17 – target_laser

The target laser can be used to add laser beams to a map. To do this, add a target_laser to the map and have it target a target_position entity. The laser will aim itself at the target_position and the beam will reach as far as it can, until it is blocked by a wall, player or bot.

Keys:

- targetname : activating entity points to this.
- dmg : the amount of damage to deal whenever a player, bot or func_breakable is blocking the laser.
- color : color of the laser beam (default 1.0 0.0 0.0).
- noise : path/name of .wav file to play while laser is turned on (eg. sound/weapons/lightning/lg_hum.wav).

Spawnflags:

- START_ON : the laser will be turned on as it is spawned.

Note that a target_laser can target entities like a func_door or func_train to create moving lasers. These entities must have an origin brush.

2.18 – target_logic

This entity can be targeted by multiple triggers. Only when each of the triggers that are targeting the entity are triggered, target_logic will fire it's own target entity. This can be used to, for example, create doors that are opened after three buttons have been pushed. Note that target_logic can handle no more than 10 triggers. If a trigger that's targetting a target_logic is triggered twice, it is considered to be not triggered anymore, unless the STAY_ON spawnflag is set. In other words, without this spawnflag triggers toggle their own state.

Keys

- targetname : activating trigger points to this.
- target : this points to entities to activate when this entity is triggered.

Spawnflags:

- RANDOM : only one of the targeted entities will be triggered at random.
- STAY_ON : a trigger will remain triggered until the target_logic is reset.

2.19 – target_mapchange

Use this entity to progress the player to a new map. When activated, the entity will force the server to load a new map. It allows level designers to create multi-level single player campaigns.
Only the player is moved over to the new map. Any bots currently in the map are removed from the game. Also, the player keeps his inventory as he moves to the next map. This means that health, armor, weapons, ammo and holdable items, as well as your current score, are retained in the next map.

Keys:

- mapname : The map to load. When omitted, the current map will be restarted.
- targetname : activating trigger points to this.

2.20 – target_modify

This entity can be used to modify the value for a specific key on a map entity. It can, for example, change the target of a trigger or the message of a target_print. The following keys can be changed by the target_modify:

angle, clientname, color, count, deathtarget, health, key, light, mapname, message, overlay, target, target2, targetname, targetname2, targetshadername, teleportertarget, spawnflags, speed, skill, value, wait, armor, damage, dmg, wait.

Do note that the *color* and *light* keys are only applicable to func_ entities, not to light entities. Light entities can never be modified in-game.

Keys:

- target : this points to entities to modify when this entity is triggered.
- target2 : this points to entities to modify when this entity is triggered.
- targetname : activating trigger points to this.
- key : the key to modify on the target entity.
- value : the value to set for the key on the target entity.

2.21 – target_music

The target_music entity can be used to start or stop playing a music track.

Keys:

- targetname : activating entity points to this.
- targetname2 : activating entity points to this.
- music : path/name of looping .wav file used for level's music (eg. music/sonic5.wav).

2.22 – target_objective

Use this entity to set primary or secondary objectives for the player. These objectives will be displayed on the objectives overlay when the player holds the TAB key. Objectives offer no technical limitations for the player, they are purely there for aesthetics so to speak. It is a tool for the level designer to guide the player. When an objective (primary or secondary) is changed by a target_objective entity, the player is notified about this through an on-screen message.

Keys:

- targetname : activating trigger points to this.
- message : the objective text.

Spawnflags:

- SECONDARY : The secondary objective is set instead of the primary objective.
- SILENT : When set, the player will not be notified that the objectives are updated.

2.23 – target_playerspeed

This entity can be used to modify the movement speed of the player that activated the entity. Use with caution, as at the time of this writing, the entity isn't properly tested in situations where player movement speed is altered, such as the use of jumppads or when swimming.

Keys:

- speed : the movement speed to apply to the player(s).
- targetname : the activating trigger points to this.

2.24 – target_playerstats

The target_playerstats can be used to set the player's health/armor to a specific value. It cannot be triggered by bots.

Keys:

- health : the amount of health to give the player
- armor : the amount of armor to give the player
- targetname : the activating trigger points to this.
- targetname2 : the activating trigger points to this.

Spawnflags:

- ONLY_WHEN_LOWER : Only sets the player's health/armor if it is lower than the specified amount of health/armor.
- NO_HEALTH : The player's health is never set by this entity.
- NO_ARMOR : The player's armor is never set by this entity.

2.25 – target_secret

Allows the level designer to include secrets in the map. When activated, a target_secret informs the game that the player has found a secret. Optionally a message can be displayed when the secret is found. Each secret that is found will add 5% of the player's carnage score to his total score. Note that a map may not contain more than 64 secrets. If a map is spread across multiple map files (BSP files), using target_mapchange to link them together, then the total of all secrets across all maps may not be higher than 64.

Keys:

- targetname : activating entity points to this.
- targetname2 : activating entity points to this.
- message : text string to print on screen when secret is found. When omitted it defaults to "You found a secret!"

Spawnflags:

- SILENT : No message is printed to the screen when the secret is found.

2.26 – target_skill

This sets the skill level for the next map that will be loaded. The changed skill level is *not* applied immediately, so usually this will be used in a level that also implements a target_mapchange entity.

Keys:

- targetname : activating entity points to this.
- skill : the skill level to apply. 1 is easiest, 5 is hardest (default = 2).

2.27 – target_unlink

The target_unlink entity links and unlinks entities from the world. In other words, it removes entities from the game world or puts them back in again. This does not work on light entities. Looping target_speaker entities will also not stop playing their sound (see the chapter on target_speaker for more information about starting and stopping target_speaker entities). Some other func_ entities are also unaffected by target_unlink.

Keys:

- targetname : activating trigger points to this.
- target : this points to entities to link/unlink when this entity is triggered.

Spawnflags:

- ALWAYS_UNLINK : The targeted entity will always be unlinked from the world.
- ALWAYS_LINK : The targeted entity will always be linked to the world.
- IMMEDIATELY : The target_unlink will link/unlink its targeted entities when it is spawned.

2.28 – target_variable

The target_variable entity is an entity with two separate functions. When neither its COMPARE_EQUALS and COMPARE_NOT_EQUALS spawnflags have been set, it will (when triggered) write a variable with the specified name (key) and value to memory. The level designer is free to choose a name for the key as long as they don't include the backslash (`\`), semicolon (`;`) and double quote (`"`) characters.

When either of the aforementioned spawnflags is set, the target_variable entity will read the specified variable instead of writing to it. It will then compare the value of the variable to the specified value and activate its targets if the two values match or don't.

Variables that are written to memory are persisted across map changes if the map change was initialized by a target_mapchange entity. This means that it is possible to activate entities in map B based on actions performed in map A.

Keys:

- targetname : activating entity points to this.
- targetname2 : activating entity points to this.
- target : this points to entities to activate when this entity is triggered.
- target2 : this points to entities to activate when this entity is triggered.
- key : The name of the variable to read or write.
- value : The value that will be written or compared to.

Spawnflags:

- COMPARE_EQUALS : The entity reads the specified variable and triggers its targets if the specified value matches the value stored in the variable.
- COMPARE_NOT_EQUALS : The entity reads the specified variable and triggers its targets if the specified value does not match the value stored in the variable.
- IMMEDIATELY : The entity will be activated when it is spawned.

2.29 – trigger_death

The trigger_death entity can be used to trigger other entities when a number of deaths have occurred. These deaths can be the player or bots dying. The entity is configurable with a number of deaths that have to be recorded before the entity fires its target.

The entity is very similar to the trigger_frag entity, but differs in the fact that trigger_death responds to *any* type of death (including falling in the void, lava or slime and drowning or cratering) and sees the player or bot whose death resulted in the trigger being activated as the activating player.

Keys:

- target : this points to the entity to activate.
- nobots : dying bots do not trigger this entity.
- nohumans : the player's death does not trigger this entity.
- count : the number of deaths that must have occurred before the entity activates its targets

Spawnflags:

- TRIGGER_ONCE : The trigger will only fire once.

2.30 – trigger_frag

The trigger_frag entity can be used to trigger other entities when a number of frags have been made. These frags can be made by the player or bots. The entity is configurable with a number of frags that have to be recorded before the entity fires its target. Note that the number of frags is cumulative

The entity is very similar to the trigger_death entity, but differs in the fact that trigger_frag responds only to frags (so not to the player or bots falling into the void, lava or slime and drowning or cratering) and sees the player or bot whose frag resulted in the trigger being activated as the activating player.

Keys:

- target : this points to the entity to activate.
- nobots : bots scoring a frag do not trigger this entity.
- nohumans : the player scoring a frag does not trigger this entity.
- count : the number of frags that must be scored before this entity is triggered

Spawnflags:

- TRIGGER_ONCE : The trigger will only fire once.
- NO_SUICIDE : The trigger does not respond to the player or bots blowing themselves up.

2.31 – trigger_lock

trigger_lock works the same way as a trigger_multiple entity. The only difference is that the level designer can specify which keys (holdable_key_* entities) are required to activate the lock. All required keys (if the player has them all) are removed from the player's inventory as the lock is activated unless the KEEP_KEYS spawnflag is set.

Keys:

- target : this points to the entity to activate.
- message : message displayed when user triggers lock without the required keys.
- lockedsound : sound to play when player triggers lock without the required keys (eg. sound/world/klaxon1.wav).
- unlockedsound : sound to play when player triggers lock while having the required keys (eg. sound/world/klaxon2.wav).
- wait : time in seconds until trigger becomes re-triggerable after it's been touched (default 0.2, -1 = trigger once).

Spawnflags:

- KEY_RED : only triggered when player has holdable_key_red item.
- KEY_GREEN : only triggered when player has holdable_key_green item.
- KEY_BLUE : only triggered when player has holdable_key_blue item.
- KEY_YELLOW : only triggered when player has holdable_key_yellow item.
- KEY_MASTER : only triggered when player has holdable_key_master item.
- KEY_GOLD : only triggered when player has holdable_key_gold item.
- KEY_SILVER : only triggered when player has holdable_key_silver item.
- KEY_IRON : only triggered when player has holdable_key_iron item.
- KEEP_KEYS : the player keeps the keys used to activate this trigger. Otherwise, used keys are removed from inventory.

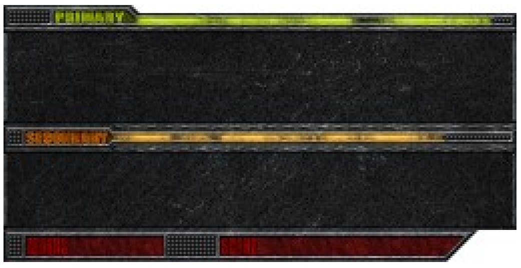
3 – Modified entities

This chapter describes the changes made to existing entities (those already available in vanilla Quake 3 Arena).

3.1 – worldspawn

The worldspawn entity has received a few new keys. Two of them are music related: “scoreboardmusic” and “deathmusic”. These keys can be used in the same way as the existing “music” key. The scoreboardmusic key defines the music file that will be played when the end-game scoreboard is displayed. The deathmusic defines the music file that will be played when the player is dead and waiting to respawn.

There are also two new model releated keys. These are “playermodel” and “playerheadmodel”. These keys can be used to force a certain playermodel onto the player. For example, when specifying “visor/gorre” as playermodel, the player will always use the visor model with the gorre skin regardless of what the player has configured in the player setup menu himself. The playerheadmodel key is used to specify the model and skin used for the head. Note that if a playermodel is specified without specifying a headmodel, the headmodel will revert to the model specified with playermodel. If neither playermodel or playerheadmodel are specified, the game will use the model the player has picked in the player setup menu himself.



Another new feature is the “objectivesoverlay” key. This can be used to specify a custom image to use as objectives overlay. The objectives overlay is the screen that displays the current objectives. The background on which the objectives and deaths/score stats are superimposed is what can be specified through the objectivesoverlay key. When this key is omitted, the game will revert to the default overlay. A valid value for the objectivesoverlay key would be *menu/objectives/overlay.tga* which would display the default objectives overlay.

The worldspawn entity also received the “enabledust” and “enablebreath” keys. When the enablebreath key has a value of 1, the player and bots will show breath puffs. This can be used to create the illusion of a very cold environment. The enabledust key enables dust puffs when the player or bots walk over surfaces which have a shader with the “dust” surfaceparm. These keys were implemented from the Team Arena code. See the Team Arena “Terrain Manual” for more information on these two keys. Note that these effects can be disabled by the player through the cg_enableDust and cg_enableBreath cvars.

3.2 – func_bobbing

This entity has been largely unchanged, except its spawnflags. The new func_bobbing has three spawnflags instead of two: X_AXIS, Y_AXIS and Z_AXIS. The func_bobbing will oscillate along the Z-axis by default (if no spawnflags are set). As soon as any of the three spawnflags is set, the default is ignored and the entity will oscillate along the selected axes.

When multiple axes have been selected, the func_bobbing will move equal amounts of space along those axes, resulting in a diagonally moving func_bobbing.

3.3 – func_button

Two things are changed for func_button entities. First, a “sound” key can be set that will refer to a wav file. This sound, if set, will override the default sound that's played when the button is pressed.

The second change is that the wait key can be set to -2, which means that the button will never return and can only be pressed once.

3.4 – func_door

Doors have been changed in ways similar to func_button. With doors, it is also possible to set the wait key to -2, which means the doors will never close once open, unless the door is triggered by a triggering entity, in which case the door will open or close only when it's triggered.

func_door received two new keys: “startsound” and “endsound”. These can be used to override the default sounds that are played when the door starts moving or when the door stops moving (it has fully opened).

Additionally, a func_door’s health can be set to a value lower than 0 to prevent it from being shot open.

3.5 – func_rotating

The func_rotating entity will rotate along the Z-axis if none of its spawnflags (X_AXIS, Y_AXIS and Z_AXIS) are set. When one or more spawnflags are set, the default rotation axis is ignored and the func_rotating will rotate along the selected axes. When multiple axes have been selected, the func_rotating will show some erratic rotation behaviour.
Additionally, the TOGGLEABLE spawnflag allows the func_rotating to be targeted by a trigger which will stop or resume the entity's rotational movements. Note that enabling this spawnflag will cause collision detection between the func_rotating and a player or bot to not function correctly anymore. It is therefore advisable to not use this spawnflag for func_rotating entities that can be touched by the player or a bot. Note that setting the START_OFF spawnflag (which will spawn the func_rotating in a non-moving state) implicitly enables the TOGGLEABLE spawnflag. It is not strictly necessary to enable both spawnflags.

3.6 – func_static

The func_static entity received a new START_UNLINKED spawnflag. This spawnflag can be used to remove the brush entity from the world when the map is first loaded. The entity can be put back into the world with the target_unlink entity. It is also possible for another triggering entity to target a func_static directly. When activated, the func_static will be removed or put back into the world. This way, a target_unlink entity is not necessary.

3.7 – func_timer

func_timer has been changed in two ways. First, it received a START_DELAYED spawnflag. When this spawnflag is set, the func_timer will start its timer clock when it is turned on and activate its targets for the first time when the timer reaches the specified amount of time. When the spawnflag is not set, the func_timer operates as it normally would and it will immediately activate its targets for the first time when it is turned on. In other words, there is no delay between turning the timer on and the timer activating its targets when the spawnflag is not set while setting the spawnflag will make sure there is a delay between the two events.

The other change is the count key. This key limits the number of times the func_timer will activate its targets. When this limit is reached, the func_timer will turn itself off and reset its limit counter. This means that if the count key is set to 5 and after 4 times the func_timer is turned off (for example, by being triggered by another entity), the func_timer will activate its targets 5 times if its turned on again. Even if the timer is turned on after hitting the specified limit, it can be turned on again for another 5 activations.

3.8 – func_train

The func_train entity has new “startsound” and “endsound” keys. These work the same as the “startsound” and “endsound” keys as on func_door and func_door_rotating. The endsound will be played when the train stops moving at a path_corner entity which has a wait key set. The startsound will be played when the train begins moving again.

func_trains can be targeted by a trigger to make them start moving at a path_corner again. See path_corner for more information.

func_train also accepts a *health* key. When set to any non-zero value, any amount of damage dealt to the func_train will make the train activate the targets it has targeted by *target* or *target2*. Since the *target* key is usually used to target a path_corner entity (which cannot be activated), any targeted entities need to be targeted using the *target2* key.

3.9 – holdable_teleporter

This entity received a new key: “teleportertarget”. Enter the targetname of a spawnpoint (or misc_teleporter_dest) entity here and the personal teleporter will always teleport the player that uses it to that location.

3.10 – path_corner

The path_corner entity has been modified so that it accepts a *wait* key of -1. When this key is set to -1, the func_train entity targeting this path_corner entity will stop moving at the path_corner until the func_train is triggered again. Only then will the func_train continue moving.

*3.11 – shooter_**

All of the shooter entities (shooter_grenade, shooter_plasma, shooter_rocket and the new shooter_bfg) received a new “dmg” key which can be used to specify a custom amount of damage the shooter will do. When this key is omitted, the shooter will do the default amount of damage applicable for the projectile it fires. Do note that the damage shooter entities deal is scale similar to the damage bots deal.

The speed of the respective projectiles (grenades, plasma bolts, rockets and BFG bolts) fired by the shooter_ entities can be controlled by setting the “speed” key. Different projectiles have different default speeds.

All shooter entities have also received a TARGET_PLAYER spawnflag. When set, the shooter will always aim directly at the player. It is not necessary to use a target_position or info_notnull entity to target the shooter then.

Shooter entities have also received the new NO_BOTS and NO_HUMANS spawnflags. These spawnflags can be used to determine who can receive damage from the shooter. If the NO_BOTS spawnflag is set, bots will not be hurt by the shooter. If the NO_HUMANS spawnflag is set, the player will not be hurt by the spawnflag.

3.12 – target_delay

The new TOGGLE spawnflag on target_delay does pretty much what it says. Normally, when a target_delay is triggered a second time after it has been triggered, the timer will be reset. When the TOGGLE spawnflag is set, the target_delay will stop counting if it has been activated for a second time before it actually activated its own targets. This allows you to basically stop a target_delay from activating its target if it has already been activated itself.

3.13 – target_give

Target_give received a new GIVE_PLAYER spawnflag. When this spawnflag is not set, the player or bot that triggered the target_give will receive the item(s) targeted by the target_give entity. When the spawnflag is set, the player will receive the item(s), regardless of who activated the entity.

3.14 – target_print

The target_print entity received a new SUBTITLE spawnflag. When this spawnflag is set, the entity's message will not be printed to the center of the screen but to the bottom and in a smaller font. This can be used for subtitles and less obtrusive messages. The level designer can use the *wait* key to specify how long the subtitle should stay on-screen (note that the wait key only works for subtitles, not center-printed messages). If no wait key has been specified, the game will calculate the value of the wait key itself by dividing the number of characters in the message by 15. Any message with less than 15 characters will remain on-screen for 1 second.
Note that the player can completely disable subtitles by setting cg_drawsubtitles to 0. This means level designers can not depend on players actually seeing subtitle messages.
To run long subtitles along multiple lines, use the '\n' character to insert a line break, like this:
"this will end up on line 1.**\n**This will end up on line 2."

3.15 – target_push

In vanilla Quake 3, the target_push entity plays a "beep" sound when the BOUNCEPAD spawnflag isn't set. In EntityPlus, the target_push remains silent when no spawnflags are set. It has also received an additional spawnflag: JUMP. Setting this spawnflag will make the character that activates this entity play its jump sound. The BOUNCEPAD and JUMP spawnflags cannot be used simultaneously. JUMP will be ignored in this case.

3.16 – target_relay

Target relay has received a single new key: "count". This key can be used to determine the number of times the target_relay must be triggered before it activates its target. Multiple entities may target the target_relay. Each time an entity targeting the target_relay is triggered it will be counted as one trigger. When the key is omitted or has a value of 0 or 1, the trigger_relay functions as it normally would. Note that the counter on target_relay can be manipulated using the target_modify entity. The counter for target_relay is stored in the "damage" key of the entity. This means that a target_modify entity that modifies the target_relay's *damage* key to a value of 0 basically resets the target_relay.

The entity has also received a new "ONCE" spawnflag. When this spawnflag is set, the target_relay will only activate its targets once and becomes inactive after doing so.

3.17 – target_remove_powerups

This entity has a number of added spawnflags: POWERUPS, WEAPONS and HOLDABLES. When none of these spawnflags have been selected, the entity reverts to its original behavior of removing all powerups. By selecting any of the spawnflags, the level designer can control what the entity will remove from the activating player. Here's a short overview of what enabling the spawnflags will remove.

POWERUPS: Remove all item_* powerups, except for health and armor. This does include items like the Quad Damage, Battlesuit, Haste and Flight powerups.

WEAPONS: Removes *all* weapons and ammo the player is carrying.

HOLDABLES: Removes the medkit, personal teleporter and all keys.

3.18 – target_speaker

The target_speaker entity didn't actually change, but it has been fixed. In standard Quake 3, it was impossible to toggle target_speaker entities which have the LOOPED_ON or LOOPED_OFF spawnflags off (target_speakers with LOOPED_OFF could be toggled on). In EntityPlus it is possible to toggle target_speakers with either of these spawnflags on or off by targeting them with another trigger.

3.19 – trigger_hurt

trigger_hurt is pretty much unchanged, except that the START_OFF spawnflag was removed. This spawnflag didn't have any effect on the entity due to a bug in the existing Quake III Arena code.

3.20 – trigger_push

When the new SILENT spawnflag is set, the trigger_push will no longer emit a sound when it is touched by the player (or a bot).

3.21 – trigger_teleport

The trigger_teleport entity has received a new PORTAL spawnflag. When this is set, the trigger_teleport will function like a portal (similar to the portal's in Valve Software's game "Portal"). When the player hits a trigger_teleport with this spawnflag, he/she is moved to the targeted misc_teleporter_dest. The new position is not exactly at the destination's origin but relative to the trigger's origin at the moment the trigger was hit. This means that if you hit the trigger to the left of its origin, you will be placed an equal amount of units left of the destination. You will also maintain momentum and viewing angles. Note that this feature is very basic and not fully complete. Use at your own risk.

3.22 – Item pickup entities

Item pickup entities are all item_*, weapon_*, ammo_* and holdable_* entities. In other words: all entities that represent items that can be picked up by the player. All of these entities received the new SILENT spawnflag. When this spawnflag is set the item will not play its default "pickup sound" when the player picks up the item.

3.23 – Spawnpoints

In EntityPlus, there is only one type of spawnpoint left: *info_player_start*. All other spawnpoint entities (including info_player_deathmatch) are gone. info_player_start can be used like any regular old spawnpoint entity, but it has a few added features.

info_player_start has a new "count" key, which can be used to limit the number of times a player can spawn at that location. When that number of spawns have been executed, the spawnpoint deactivates itself. Level designers should be aware that if a map has no active spawnpoints, the player will be booted from the game with an error message.

The info_player_start can also be manually disabled. This can be done by having a triggering entity target the spawnpoint entity. When the trigger is activated, it will disable or enable the spawnpoint entity. Additionally, info_player_start has the DISABLED spawnflag so that the spawnpoint is disabled from the start.

Note that a spawnpoint that is disabled because it has reached its maximum number of allowed spawns can not be enabled anymore.
This enabling and disabling of spawnpoints can be used to implement a sort of checkpointing system in your map. The level designer could enable or disable spawnpoints depending on where the player is in the map by the use of, for instance, trigger_multiple entities. Spawnpoints can be initially disabled and enabled one by one as the player progresses through the map. Because

EntityPlus seeks the nearest (enabled) spawnpoint to respawn the player at, the level designer can minimalize the need for the player to run through already cleared sections of the map after he has died.

3.24 – targeting entities

All entities that use the target key to point to another entity which it will activate can now target a second entity with a different targetname through the target2 key. For instance, a trigger_multiple could have a target key of “t1” and a target2 key of “t2”, triggering both entities with either of these targetnames. Note that the game will first find and activate all entities whose targetname match the target key and then find and activate all entities whose targetname match the target2 key. If the order of activating entities is important, it is important to remember that target always preceeds target2.

On top of that, a new targetname2 key has been introduced as well, which allows one entity to have two targetnames. There is no relation between target/targetname and target2/targetname2, so one entity with a target key of “t1” will activate an entity with a targetname2 key of “t1”.

Examples of these entities with a target2 key are all trigger_* entities (except trigger_push and trigger_teleport), func_button, target_relay, target_delay, target_logic, target_disable, func_breakable, func_timer, ammo_*, item_*, weapon_*, holdable_*, path_corner (but not to link them together) and all spawnpoints.

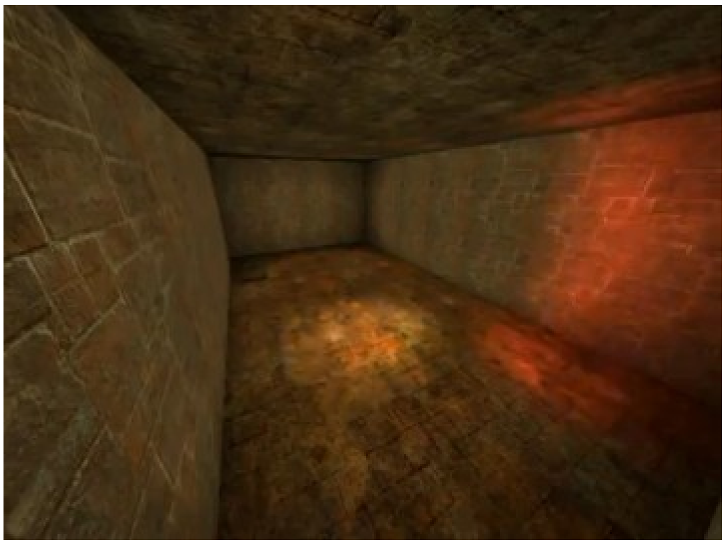
Some entities that do not allow the use of the target2 key are: target_give, target_push, target_teleporter, func_door, func_train, info_player_intermission, info_waypoint, light entities, misc_model, misc_portal_camera, misc_portal_surface, path_corner, shooter_*, target_botspawn, target_modify, trigger_push, trigger_teleport.

Note that the GtkRadiant editor will not draw target lines between entities that use the target2 or targetname2 keys to target each other.

3.25 – notep key

All entities can use the “notep” key. This key works similar to the *notq3a* and *notta* keys (note: these two keys were deprecated from EntityPlus). When the value of the *notep* key is set to 1, the entity will not spawn in EntityPlus. This allows level designers to create maps that are compatible both with vanilla Quake 3 and EntityPlus.

3.26 – Constant light on func_* entities



All of the func_ entities accept both a “color” and a “light” key. Using these keys allows level designers to have these entities emit (dynamic) light. Unfortunately, due to a bug in Quake III Arena, this functionality is broken. EntityPlus fixes this bug and thus it is possible for level designers to add very simple dynamic lighting effects to their maps. The “light” and “color” keys accept values similar to the _light and _color keys on light entities. This means that the light key accepts integer values that determine the intensity of the light. The color key accepts three decimal values ranging from 0 to 1 which specify the red, green and blue values of the color of the light.

The screenshot to the right shows a very simple example of a red light and a white light added to the map. When seen in-game, the lights move along the walls or floor.

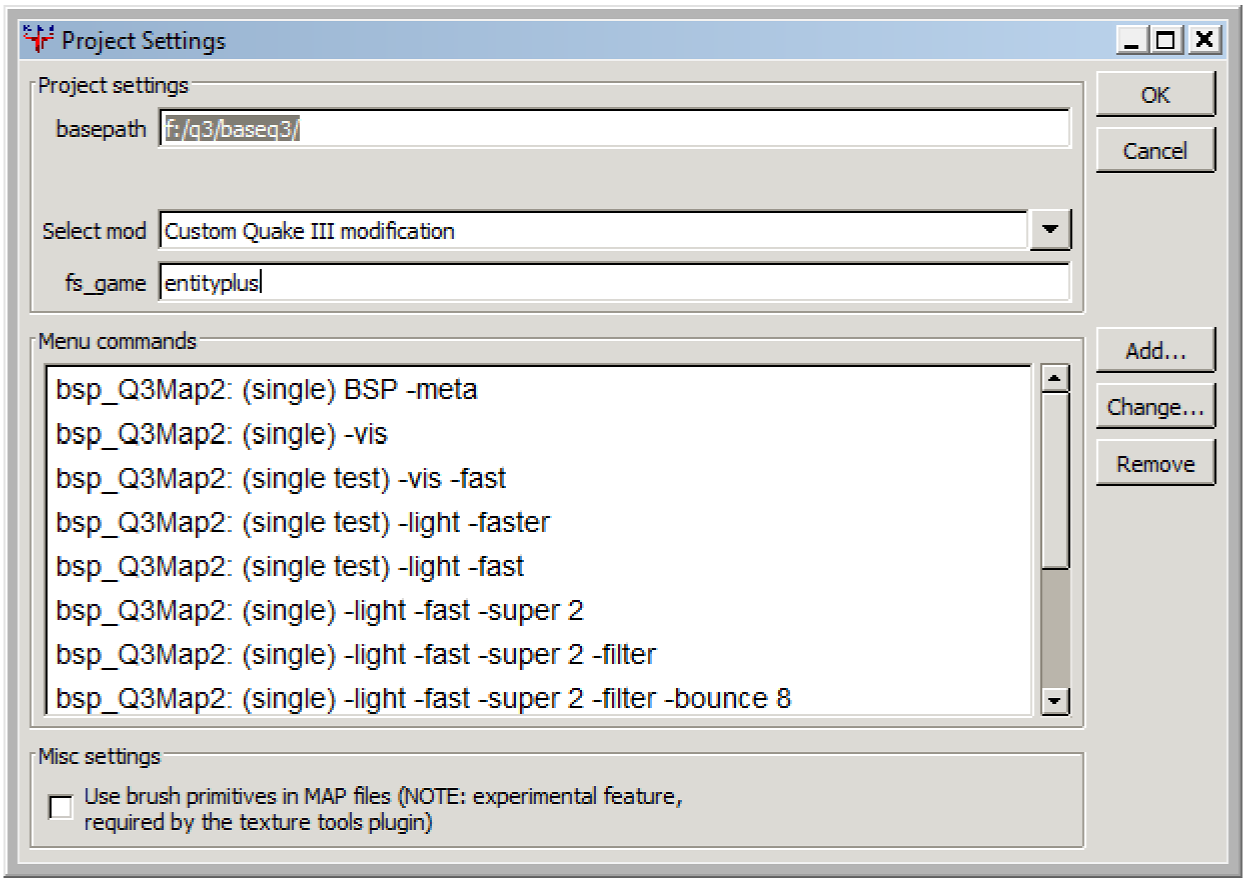
For a video of the shown example, go to http://www.youtube.com/watch?v=q7IGSeWesTE

4 – Miscellaneous info for level designers

This chapter lists a few bits of information that are relevant to creating levels and packaging them for distribution.

4.1 – Configuring GtkRadiant

Because EntityPlus features a lot of new entities, the entities.def file that GtkRadiant reads to find all the definitions of the entities has been customized as well. This file can be found in the "baseq3\scripts" folder in your Quake III Arena installation folder. It is best not to overwrite this file though, but to configure GtkRadiant to use a different mod folder instead. To do this you should, after installing EntityPlus, open GtkRadiant and in the main menu click on "File" and then "Project settings". In the dialog that opens, pick "Custom Quake III modification" from the dropdown list next to "Select mod". Underneath that dropdown menu, there is an input field labelled "fs_game". Enter "entityplus" here. If those two settings look as they are in the screenshot shown below, then click ok.



Project settings

basepath: f:/q3/baseq3/

Select mod: Custom Quake III modification

fs_game: entityplus

Menu commands

```
bsp_Q3Map2: (single) BSP -meta
bsp_Q3Map2: (single) -vis
bsp_Q3Map2: (single test) -vis -fast
bsp_Q3Map2: (single test) -light -faster
bsp_Q3Map2: (single test) -light -fast
bsp_Q3Map2: (single) -light -fast -super 2
bsp_Q3Map2: (single) -light -fast -super 2 -filter
bsp_Q3Map2: (single) -light -fast -super 2 -filter -bounce 8
```

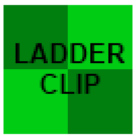
Misc settings

☐ Use brush primitives in MAP files (NOTE: experimental feature, required by the texture tools plugin)

The EntityPlus mod includes a special shaderlist.txt that lists all shader files that should be available in GtkRadiant. The shaderlist.txt that comes with EntityPlus includes all the default Quake III Arena shaders and the entityplus shader file, which means that the "ladderclip" shader becomes available under the "common" shaders. If you've configured GtkRadiant to use the EntityPlus folder as your working folder, then add new shader files to the shaderlist.txt in the entityplus\scripts folder instead of the baseq3\scripts folder.

You can always set the "Select mod" option in the "Project settings" dialog back to "Quake III Arena" to get back to the default GtkRadiant configuration.

4.2 – Creating ladders



EntityPlus has built-in ladder functionality. The mod adds a “ladderclip” shader under the “common” shaders. This shader can be applied to a brush that represents a ladder volume. When the player is inside such a brush, he can freely move up and down along the Z-axis. Because the ladder property is applied through a shader, ladderclips can be part of entities such as a func_door, func_breakable or func_plat.

4.3 – Fade-in

In EntityPlus, a map will start with a black screen, which fades to a normal view after a few seconds. This initial black screen will also display the level's name, which can be defined through the “message” key on the worldspawn entity. If no message key is defined, no text will be displayed.

Because the black screen is simply a black overlay over the normal view and the game is actually already running underneath it, it is important to keep this in mind when designing your level. This means that, for instance, spawning a bot right in front of the player at the moment the level is loaded is a bad idea because the player cannot see what is happening while the bot will already attack. Make sure the player initially spawns into a safe area or delay any triggered actions that the player should see with at least 4 seconds.

If no “message” key has been defined in the worldspawn entity, the screen will stay black for a shorter period of time than when no message key was defined.

4.4 – The effect of sv_maxclients

Because the enemies in EntityPlus levels are bots which take up a client slot, it is possible that the sv_maxclients setting is hit when an attempt is made to spawn a bot into the game. Starting a level through the in-game single player menu will automatically set the sv_maxclients cvar to 64 so that at any time a maximum of 63 enemies can be present in the map (the 64th client being the player himself). If sv_maxclients is altered manually before manually loading up a map with the map or devmap commands, it is still possible to hit the sv_maxclients limit with the result that bots do not spawn into the map, potentially breaking level design logic that depends on these bots.

4.5 – Creating an .arena file for your map

EntityPlus has a different 'single player' menu than standard Quake III Arena. It is more akin to the menu found in the “create” submenu of the multiplayer menu. The big difference is that no gametype can be selected and that this menu only lists maps that are compatible with EntityPlus' single player mode. The player can pick a map, continue to select a difficulty level and then start the game.

To make your own map show up in this menu, you'll need to include an .arena file with your map. The creation of such a file is done in the same way as you would for ordinary Quake III Arena maps and explaining the basics for such a file is not within the scope of this document.

There are a few differences between creating an arena file for Q3A and EntityPlus though. Here is the arena file for the ep_example map:

```
{
map "ep_example"
longname "Temple Escape"
description "This is the description for this map."
author "Eraser"
type "entityplus"
minversion "1.0"
}
```

It is not necessary to specify frag- time- or capturelimits because these are not required for EntityPlus. However, do note that *if* you do specify any of these attributes, you will have to set them to 0. Setting a timelimit or fraglimit here will cause the game to respond to those values. It is also unnecessary to specify bots as bots specified here will not be spawned into the game.

As with standard Quake 3 maps, the **map** key defines the filename of the map. If you use multiple maps (bsp files) in your single player campaign, simply specify the name of the first map the player should be spawned into.

Another standard key that can still be used is the **longname** key. This key can be used to specify the full name of the map. The full name of the map will be displayed in the map selection screen. If no longname is specified, the map selection screen will display the value specified in the **map** key instead.

A new key is the **description** key. This can be used to write a short description or introduction for your map. This description will show up when the map is selected in the map selection screen. If no description was specified, the map selection screen will display the text “no description available...”

The **author** key is also new. This can be used to specify the name of the author of this map.

The **type** field should contain the string “entityplus”. Adding “entityplus” to the list of accepted gametypes for this map will ensure it shows up in the “new game” menu.

Another new key that can be set in an arena file is the **minversion** key. With this key, the level designer can specify the minimum version of EntityPlus that is required to run this map. Because features and new functionality are continually added to the mod, it could be that a player downloads a map that uses features that are not yet available in the version he/she is running. If the level designer correctly sets the minversion key, the player will be warned about requiring a more recent version of the mod in the map selection screen. The text “Requires version x.x” will show up in big yellow letters to tell the player which version of the mod is required. If no *minversion* is specified in the arena file, EntityPlus will assume that *any* version is ok to run this map so such a message would never appear for a player. It is advised to simply enter the version number you, as level designer, are running when compiling the final build of your map. You can find the version number in the readme.txt file or by entering \epversion in the console while in a game. Alternatively, the “minversiontool” can be found on the EntityPlus website. This is a commandline tool that analyzes a .map file and will report the minversion for that map file.

If you create a single player adventure that spans across a sequence of multiple maps, it is best to only include an arena file for the first map of the sequence.

4.6 – surfaceparm flesh

In Quake 3, the *surfaceparm flesh* shader keyword won't have any influence on the footstep sounds a player makes. This appears to be a bug or at least a shortcoming of the Quake 3 sourcecode. This issue is fixed in EntityPlus. This means that different footstep sounds will be played for any surfaces that have a shader applied to them that have the *surfaceparm flesh* shader keyword in them. Currently EntityPlus uses the barefoot walking sound (such as the sound of Xaero or Uriel) for fleshy surfaces, so there is no audible difference for players using such characters. The sound effects that are used may be changed in the future though, to have flesh-specific sounds.

5 – New cvars and console commands

EntityPlus adds a few new cvars which can be interesting to level designers or players. This chapter lists the new cvars and how to use them.

cg_bigheadMode (default = 0)
Setting this cvar to 1 will cause all enemies to have gigantic heads. Big Head mode can also be enabled through the “Game options” menu.

cg_disableLevelStartFade (default = 0, cheat protected)
When set to 1, the fade-in at level start will be disabled, which means you can see the map right after loading has finished. This cvar was added to save level designers from having to wait 4 seconds before being able to see something every time the map is loaded.

cg_drawHoldableItems (default = 1)
When set to 0, icons for holdable items (such as the personal teleporter, medkit or keys) aren't drawn on the HUD.

cg_drawsubtitles (default = 1)
The new and improved target_print entity allows level designers to show subtitles in a level. These subtitles can be disabled by the player by setting cg_drawsubtitles to 0.

cg_drawSyncMessage (default = 1)
When demos are recorded with g_synchronousclients enabled, playing back the demo will cause a “snc” message to be displayed on the screen. By setting cg_drawSyncMessage to 0, this snc message will no longer be displayed.

cg_enableBreath (default = 1)
Enables breath puffs effect for maps that have this enabled. For more info, see the chapter on the worldspawn entity.

cg_enableDust (default = 1)
Enables dust puffs effect for maps that have this enabled. For more info, see the chapter on the worldspawn entity.

cg_gibs (default = 1)
Sets the amount of in-game gore. 0 is no gibs, 1 is normal gibs. When set to 2, gibbed enemies will produce more gibs and more blood. It will also cause each enemy to gib upon death, no matter how much health was lost. This setting can be configured through the game options menu.

cg_letterBoxSize (default = 80)
When target_cutscene starts playing a cutscene, the screen will be displayed in a letterboxed aspect ratio, which means there are black bars at the top and bottom of the screen. The cvar sets the number of pixels these black bars should be high. Note that this is in relation to the virtual 2D space Quake 3 uses to draw 2D graphics to, which is a 640x480 canvas. This means that setting cg_letterBoxSize to 240 will make the screen completely black. Setting it to 120 will mean the top and bottom quarter of the screen are black. If you want to completely disable the letterbox effect, set cg_letterBoxSize to 0.

cg_lodScale (default = 5)
This cvar is a wrapper around the existing r_lodScale cvar. The problem with r_lodScale is that it is cheat protected and thus cannot be changed without running in cheats enabled mode. It was implemented to allow level designers to use things like foliage in a more useful manner. The r_lodScale is by default set to a value which makes such things not very pretty looking, so cg_lodScale allows you to override r_lodScale's setting.

cg_marks (default = 1)
This is not a new cvar but it has been enhanced. cg_marks now determines how long marks stay on the wall. While a value of 0 still means that wallmarks aren't drawn and a value of 1 still corresponds to the default of 10 seconds, increasing the value of cg_marks adds 10 seconds. In the game options menu, the values ranging from 0 to 3 are supported (none, normal, long, extreme which are 0s, 10s, 20s and 30s respectively).

cg_paintballMode (default = 0)
When set to 1, all weapons shoot paintballs that leave brightly coloured splashes of paint all over the wall. This doens't mean they are any less lethal though. Paintball mode can also be enabled through the “Game options” menu.

g_allowSyncCutscene (default = 0)
Due to a problem with running in-game cutscenes with g_synchronousClients set to 1 (the game would crash), the game will automatically turn off g_synchronousClients when a cutscene is started. When the cutscene ends, the cvar is restored to whatever value it had before the cutscene was started. This is hopefully just a temporary solution until the real problem is solved.
The g_allowSyncCutscene cvar is disabled by default but it can be set to 1. When it is set to 1, the system that automatically disables synchronousclients is turned off. This means that the game will most likely crash when starting a cutscene with synchronousclients enabled but it allows the player to manually control g_synchronousClients which might allow for improved demo recording.

g_debugBotspawns (default = 0, cheat protected)
Normally all console messages are suppressed while a bot is being spawned by a target_botspawn. The Quake 3 engine logs some messages to the console when adding a bot to the game and these are suppressed. By setting g_debugBotspawns to 1 (this can only be done when cheats are enabled) these messages are no longer suppressed, allowing a level designer to see exactly when each bot is loaded and added to the game.

g_debugCameras (default = 0, cheat protected)
To use this cvar, cheats must be enabled. When this cvar is set to 1, the game will dump some info about info_camera entities to the console when a map is loaded. The best way to review this data is by writing the console contents to a textfile by using the *condump* command:

```
\condump console.txt
```

The file will contain a few lines with information for each info_camera present in the map:

```
----g_debugCameras----
 cutscene: 74
 origin (-192 16 192)
 angles (-352 9 0)
 wait: 4.000000
 pan: 1
 target: cpos1
 target2: c2
 targetname: c1
 targetname2: (null)
```

It will tell you the values of the wait, target, target2, targetname and targetname2 keys for that camera. It will also tell you the origin (in an X Y Z format), the view angles (yaw pitch roll) and whether or not the camera is set to pan to the next camera. The cutscene value is the entity number for the target_cutscene that is related to this camera. This allows you to find cameras that belong to a single cutscene together.
This functionality can be most useful to determine the angles value after targeting a camera with a target_position entity. The target_position entity can be removed after getting the debug info from

the camera and the angles value can be copied to the angles key of the info_camera entity (without the parenthesis). Additionally this would allow the level designer to modify the roll value as well, which is impossible with a target_position.

g_debugScore (default = 0, cheat protected)
When this debug cvar is enabled, the console will output debugging information about your score. Whenever the player scores points (carnage score), the console will log the raw number of points that have been scored. This means that any skill modifier or accuracy bonusses aren't included. Also, whenever the player displays the objectives overlay, the console will display a breakdown of the score of the player. Note that the carnage score displayed here differs from what the end-game scoreboard will show because the end-game scoreboard shows a cumulative total of the carnage score and accuracy bonusses. Also note that the console continues spewing out these console messages for as long as the objectives overlay is displayed.

g_debugVariables (default = 0, cheat protected)
This cvar can only be used when cheats are enabled. When set to 1, any target_variable entity will, when triggered, log its actions to the console. The entities will inform you about the variable they read or write, the value they write or compare and if they activate any targets based on variable comparison. The raw variable info string will also be logged to the console.

g_disableCutscenes (default = 0, cheat protected)
When this cvar is set to 1, cutscenes will not be started. The player will remain free to move around the map. This can be used during development of the map to easily skip past lengthy cutscenes. The cvar has been cheat protected because it is really intended as a development tool. Cutscenes can not be disabled or skipped by regular players because events that should take place during the cutscene and should not be disturbed by the player will continue to happen as normal.

g_listEntity (default = 0)
The g_listEntity cvar is an existing cvar that has been slightly altered. The first change is that the cvar is now cheat protected, so it can no longer be used to get preliminary insight in the contents of the level. The second change is that it accepts both “1” and “2” as values now. When the cvar is set to 1, it works as it previously did and it lists the classnames associated with all 1024 available entity slots. When set to 2, it will list only the entity slots that actually have an in-use entity inside them. In both cases though, it will print out the number of in-use entities.

g_listEntity is a good tool to determine if the player of your map is likely to hit an entity limit. There's a global limit of 1024 entities that can be present in the game at once. This is a hard limit that Quake 3 imposes.

Notarget (cheat protected)
When this command is entered in the console, bots will ignore the player. This can be useful to observe bot behavior without interference by the player.

position
This is a console command that's only available when cheats are enabled. When \position is entered into the console, it echoes back the current position of the player (origin) and the pitch/yaw/roll of the current view of the player (viewangles).

setviewpos (cheat protected)
Only available when cheats are enabled. This console command allows the player to instantly teleport to another location in the map with pinpoint accuracy. While the setviewpos command already exists in the unmodified Quake 3 game, it has been updated so no teleporter kick is applied when your use the command (a little push forward when teleported to the new location) and you can set the pitch/yaw/roll for the viewangles as well.

It allows level designers to instantly jump to the part of the map they're working on or, when used in combination with “noclip”, to create screenshots with interesting angles (most notably the camera *roll* value) or to create comparison shots from the exact same location between different builds of the map. Use the “position” command to find your current location.

trigger_target (cheat protected)
The trigger_target command can be used to trigger entities with certain targetname or targetname2 values. A targetname/targetname2 value can be specified with the trigger_target command. All entities with matching targetname or targetname2 values will be activated. This can be used to debug complex entity constructions.