JAILBREAK SOURCE

MAPPING GUIDE

Contents

Jailbreak: Source Mapping Guide

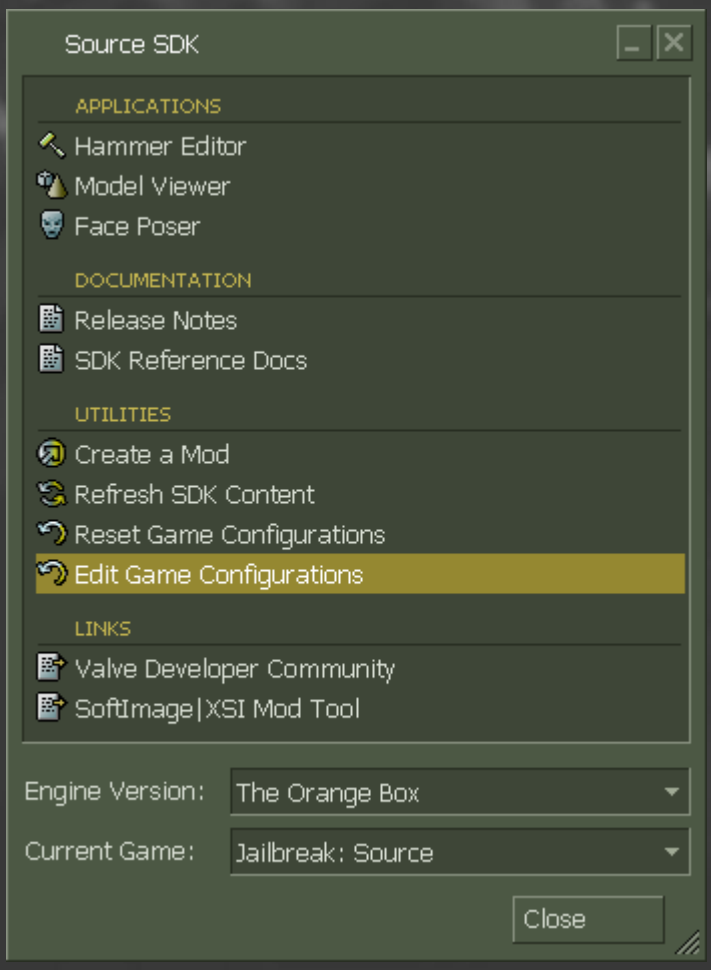
This guide will take you through the process of creating maps for each of the three game modes included in Jailbreak: Source 0.6. It will take you through all the new entities, how they are used, the new input/output system for the Jails, alongside explaining how to create devious executions and escape routes.

We have included three example VMFs in the Mapping Pack, so you can see exactly how some of our maps have been put together, as well as seeing practical examples of the I/O systems and entities at work.

Setting up Source SDK:

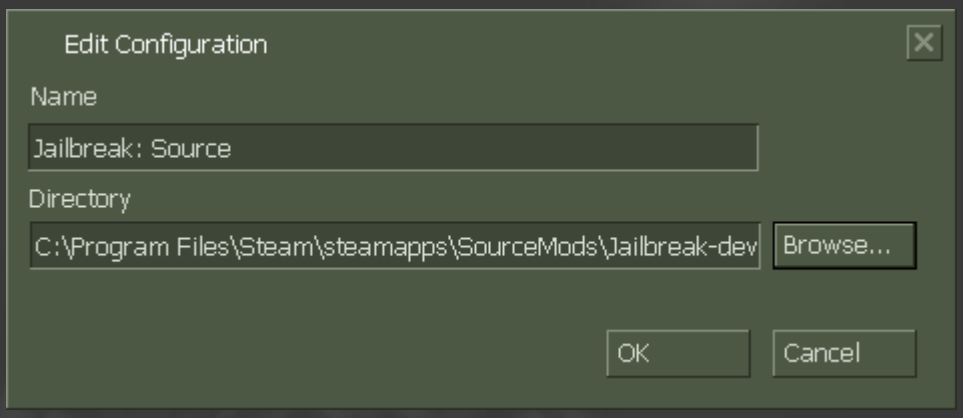
To create maps for Jailbreak, you need to have the Source SDK Tools installed, you can download these from the Tools tab in Steam. Once installed, launch the SDK Tools, and wait for it to copy files across.

Before launching Hammer, double click Reset Game Configurations. Now click Edit Game Configurations.



On the following screen, click Add and then type in *Jailbreak: Source*. Then Browse to the Jailbreak-dev Folder, which you can find inside:

C:\Program Files\Steam\SteamApps\SourceMods\Jailbreak-dev



Specifying the Jailbreak-dev folder allows you to use all of our models and textures. With this new game configuration created, select *Orange Box* as the Engine Version, and *Jailbreak: Source* as the Current Game.

Now Launch Hammer!

Configuring Hammer for Jailbreak: Source

Next you need to install the Jailbreak.fgd file, included with this pack and add it to the Hammer Configurations.

Either download the latest FGD file from here:

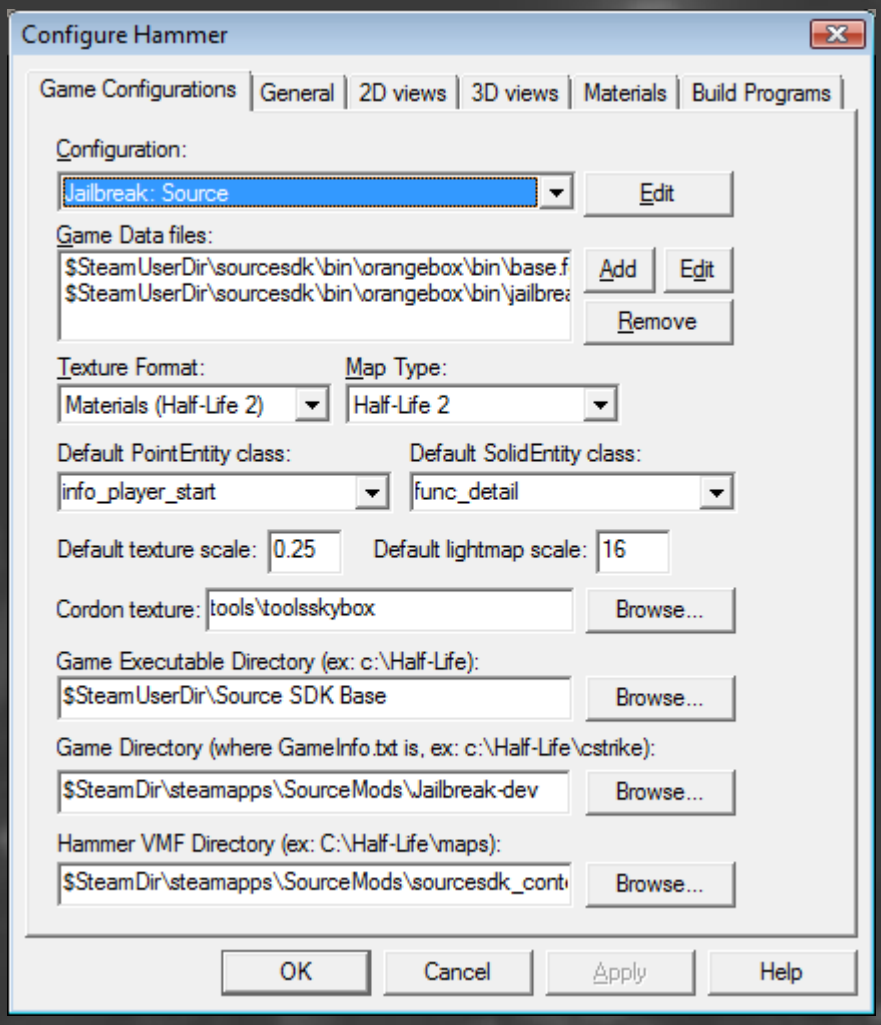
http://www.jailbreaksource.com/betafiles/jailbreak.fgd

Or install it using the version included in the Mapping Pack, copy and paste the FGD file itself into the following directory:

C:\Program Files\Steam\steamapps\username\sourcesdk\bin\orangebox\bin

So click Tools; Options from the menu bar in Hammer, then click Add in the Game Data Files box out, and choose the Jailbreak.fgd file.

Use the image below as a guide to setting up the rest of the Game Configuration window.



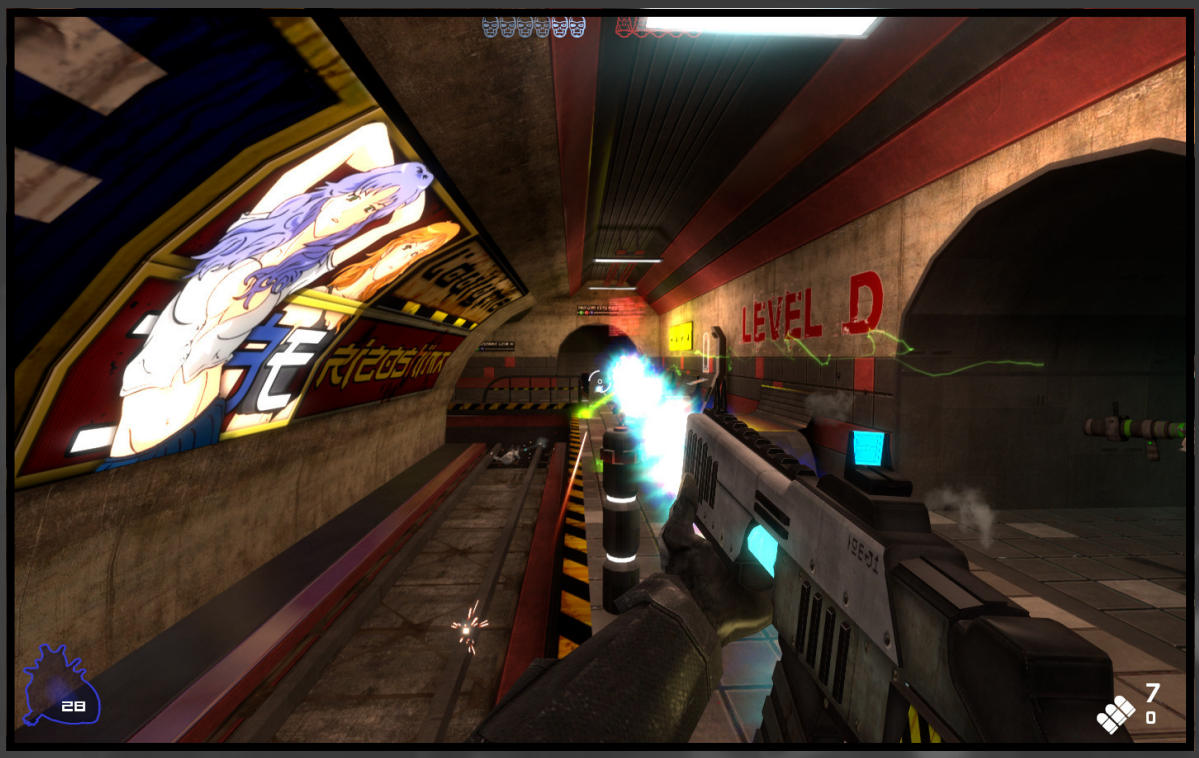
Click Apply/OK to confirm your choices, and then click File; New from the menu bar to start making your new map!

First a little Theory:

Jailbreak maps consist of several key features which make them differ from the usual team deathmatch maps, but in basic principle, the maps should be designed with team deathmatch in mind.

Both teams should have a clearly defined base, and to aid player guidance it's preferable to texture the maps with Red for Dinosaurs and Blue for Robots. In the official maps, we have used a combination of coloured lighting *and* textures to signify which side of the map a player is currently in.

Each base should contain a Jail area, this jail should be set back from the base entrances to allow defenders to prevent attackers from getting to the release button. The jail itself should be large enough to contain at least eight players comfortably. Spawn points can be placed fairly close together however, so don't be too concerned, likewise tele-fragging is prevented by a spawn waiting system.



Each Jail should have an escape route, although this is optional, players tend to get *angry* if there's no way of escaping… **You have been warned!**

Escape routes should be balanced, and if your map is not mirrored, this can be difficult to achieve, so bare this in mind when designing your layout.

Finally, each Jail should have an execution. There are two ways of creating an execution, either using the in-Jail system or by creating the new *British Bulldog Execution*, which we'll touch on later in this guide.

Finally, it's worth setting up spawn points so that players spawn either as a whole team, or in squads. Jailbreak is *way less fun* if everyone spawns and runs off in random directions. Teamwork is the key, and your map should promote this in every possible way.

Game Mode: Standard Jailbreak

There are currently three different game modes in Jailbreak: Source 0.6. The primary mode is standard Jailbreak. This game mode revolves around killing members of the opposite team, thereby respawning them inside your jail. You must defend this jail and keep the enemy inside, whilst breaking out your own team mates from the enemy jail.

The round goes back and forth until one team succeeds in jailing the entire enemy team. The losing team is then executed and a new round begins!

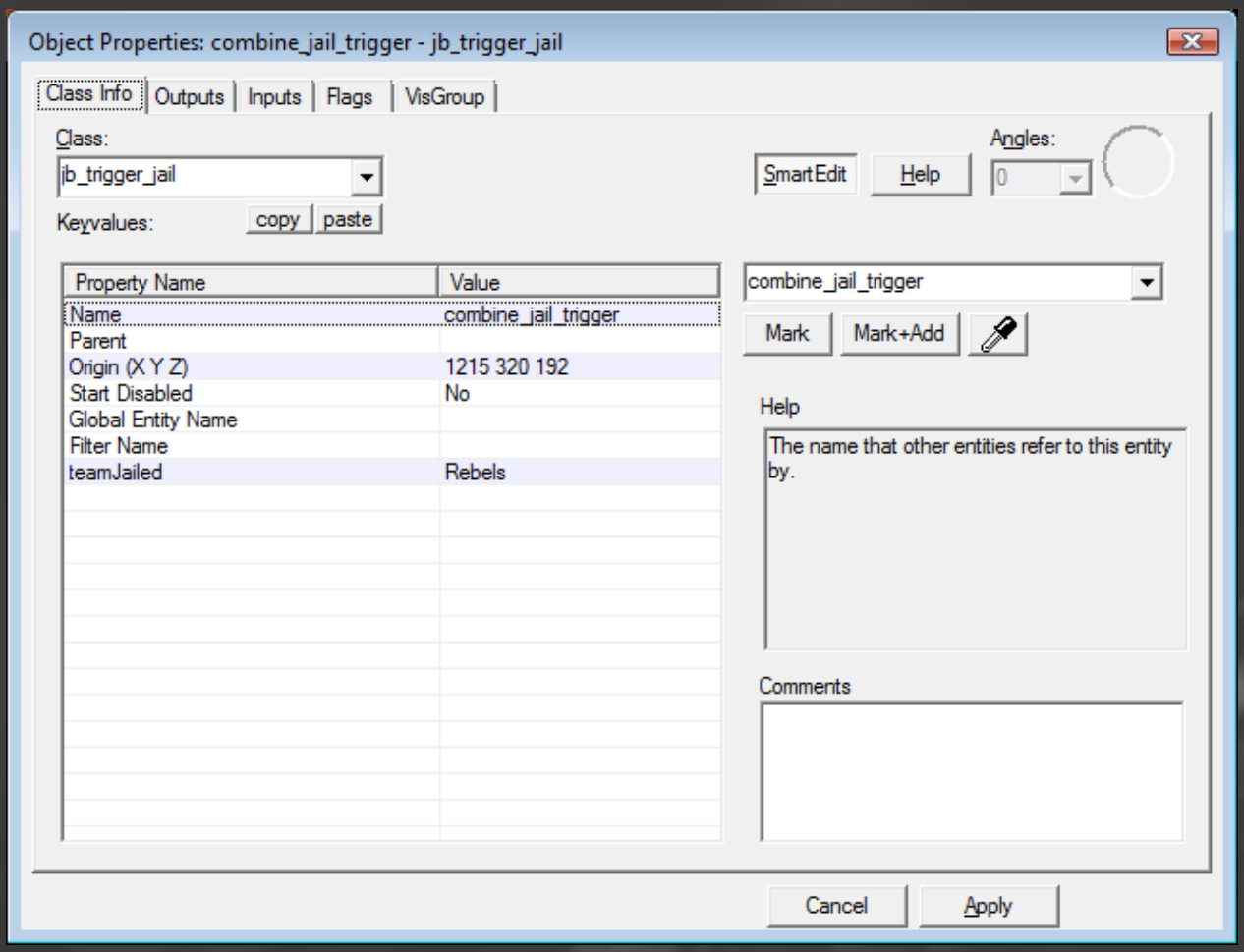
Jail Area:

The jail area is created using a trigger volume entity. Start by using the trigger texture found in the material browser, and using the Brush Tool, fill your Jail *volume* with several of these trigger blocks. You also need to fill the escape routes with these trigger volumes, to ensure that players aren't classified as "out of jail" as soon as they enter the escape route.

Once you've created all the trigger volumes and it covers every square inch of your jail, you need to select them all one by one (using CTRL to select multiple objects), then press CTRL-T together to create a brush entity.

Scroll through the list and find **jb_trigger_jail** then click apply to tie all the trigger volumes together as a single **jb_trigger_jail** entity.

Now you need to set the Key Values for your jail. Firstly, name your Jail, so name it something appropriate, as this part often gets confusing. Secondly, you need to specify which team spawns inside this Jail, in this case the Dinosaurs (Rebels).



That's all you need to do to create the jail area. Now, once a Dinosaur player spawns or steps inside this trigger volume, they will be classified as "In Jail" by the game.

You'll also need to make doors for jail players to get out through when they're released, so create a brush based door, and tie it (CTRL-T) to a func_door entity. Make sure that *Delay before Reset* is set to –1 to ensure that the doors only close when they're *explicitly* told to.

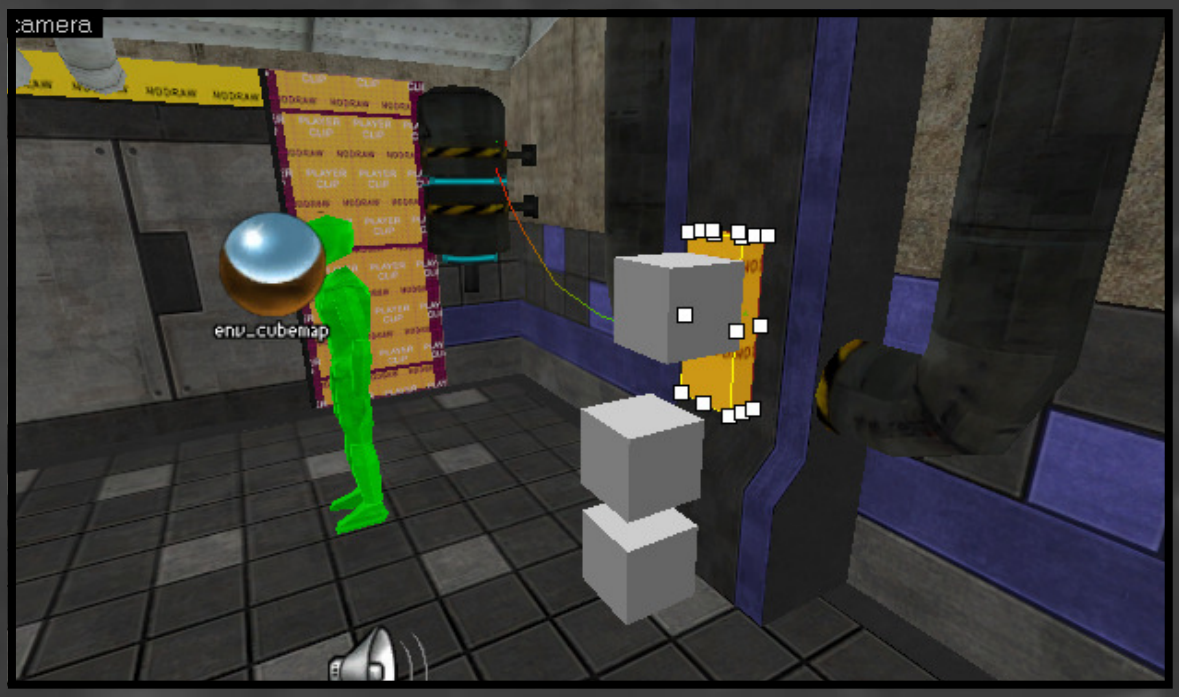
The other keyvalue settings are up to you, but I advise setting Force Closed and Ignore Debris to On, and setting Blocking Damage to a high value, to ensure that players can't block the doors open. Also check the Flags tab to ensure Touch Opens is turned off.

Jail Control Entities — The Button:

This is where things get a little more complex. Your jail has doors, which must only open and close dependent on the actions of your team members. So you need a button which opens the doors, but this button needs to be filtered by the **jb_jailcontrol_rebels** / **jb_jailcontrol_combine** entity.

So, assuming we're creating the Robot Base / Area where the Dinosaurs are Jailed, then start by placing the Jailbreak Button model *(models/izzy/izzycontrolpadcombine.mdl)* on a wall or plinth near to the Jail or the doors.

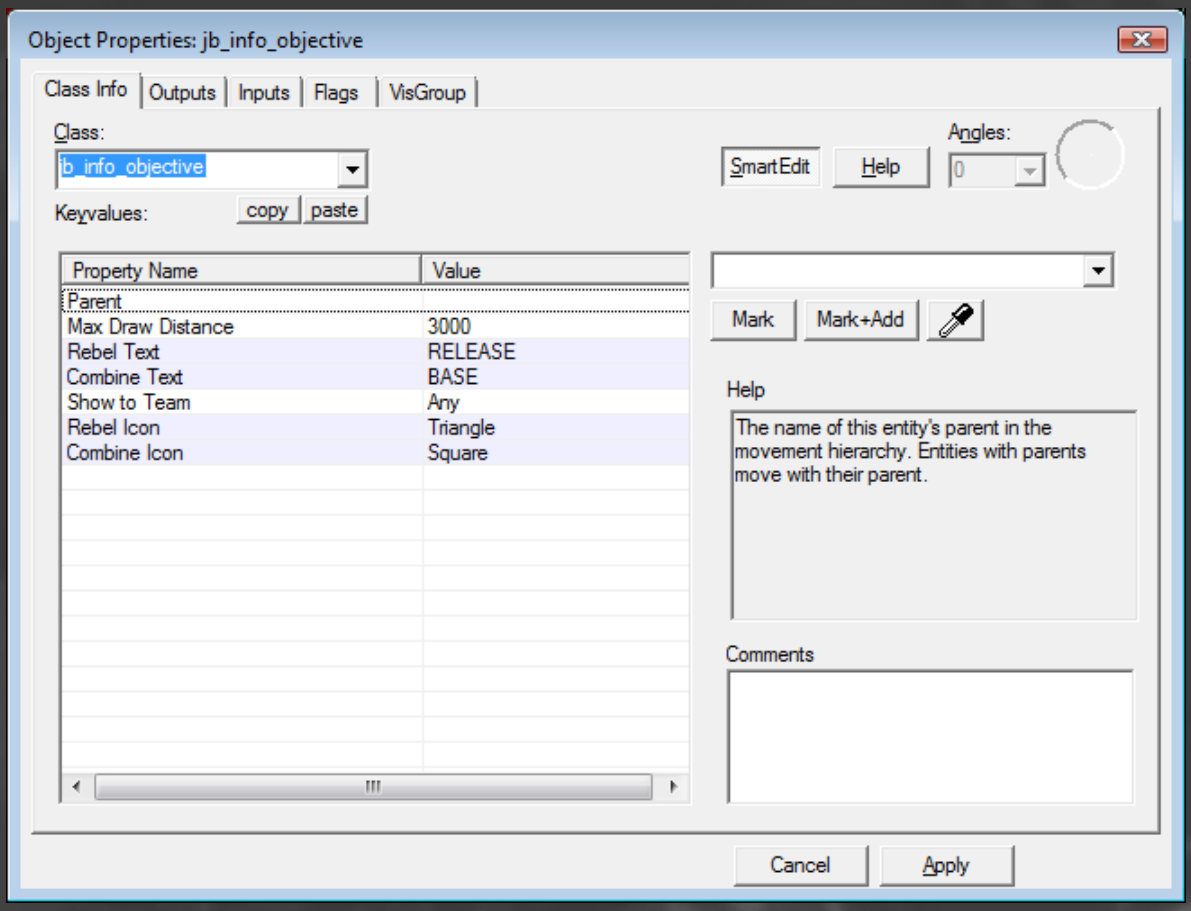
Surround this button with a brush textured with NODRAW from the mate-



rial browser, as shown in the following image:
Tie this brush to a func_button entity using CTRL-T. Then check the Key-Value for *Delay before reset* and set this to 1. Then change to the flags tab and set *Don't Move* and *Use Activates* to On with the checkbox.

Jail Control Entities — Markers:

In order for players to find the release buttons easily, we've implemented a marker system. Simply place a jb_info_objective entity using the Entity Tool, over the release button itself, then set the KeyValues appropriately, or



use the image on the next page for guidance.

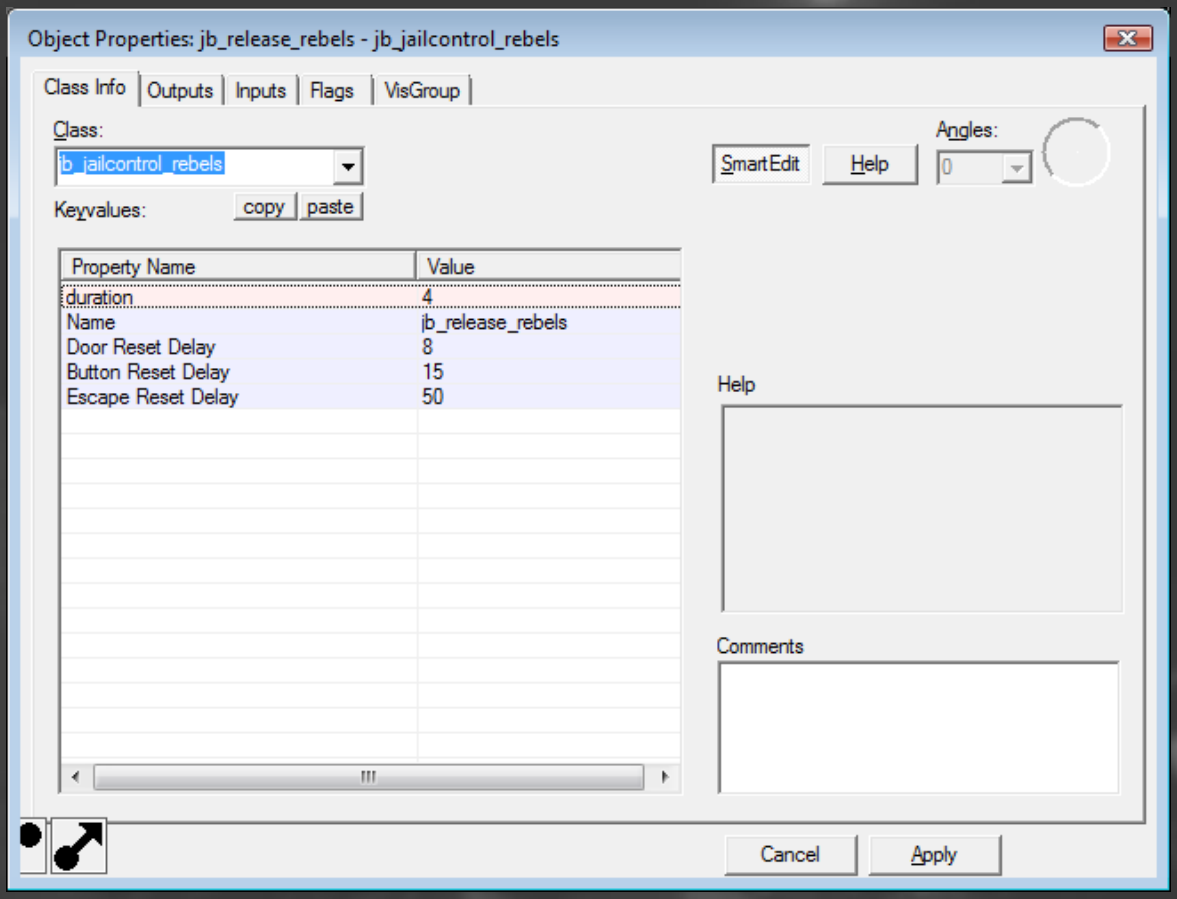
The **Rebel Text** value, is the text that the Dinosaur team see, and the **Combine Text** Keyvalue is the text that the Robot team see. You can also specify the type of marker each team sees (**Icon**), and whether one, the other or both teams can see the marker (**Show to Team**). This allows flexibility in the use of the markers, allowing you to use them for all manner of purposes, such as objectives.

Jail Control Entities — Jail Control:

The jb_jailcontrol_combine and jb_jailcontrol_rebels entities are the filters that control how the escape doors and escape routes function. These entities also control Lockdown and filter which team has pressed the button (preventing players releasing the enemy team by accident).

Place a jb_jailcontrol_rebels entity near to your release button (so it's easy to find), and open up the KeyValues by pressing ALT-ENTER or double clicking it in the 3D View.

The following Object Properties will appear, allowing you to define the Key-Values. You need to set several values here, but some are already defined



by default.

- **Door Reset Delay:** This tells the doors how long to remain open.
- **Button Reset Delay:** This tells the button how long to remain locked before it can be used again.
- **Escape Reset Delay:** This tells the escape doors how long to remain closed before opening again after a release.

Now, these KeyValues on their own are useless, unless we specify targets for the Outputs that this entity fires when the time values are met. So change to the Output Tab.

Here you will find a blank screen of entries, as no Outputs have been set yet. Outputs define how and what occurs when a condition is met. Click **Add** to add your first Output.
To start with we need to Lock the button the moment it's pressed, to pre-

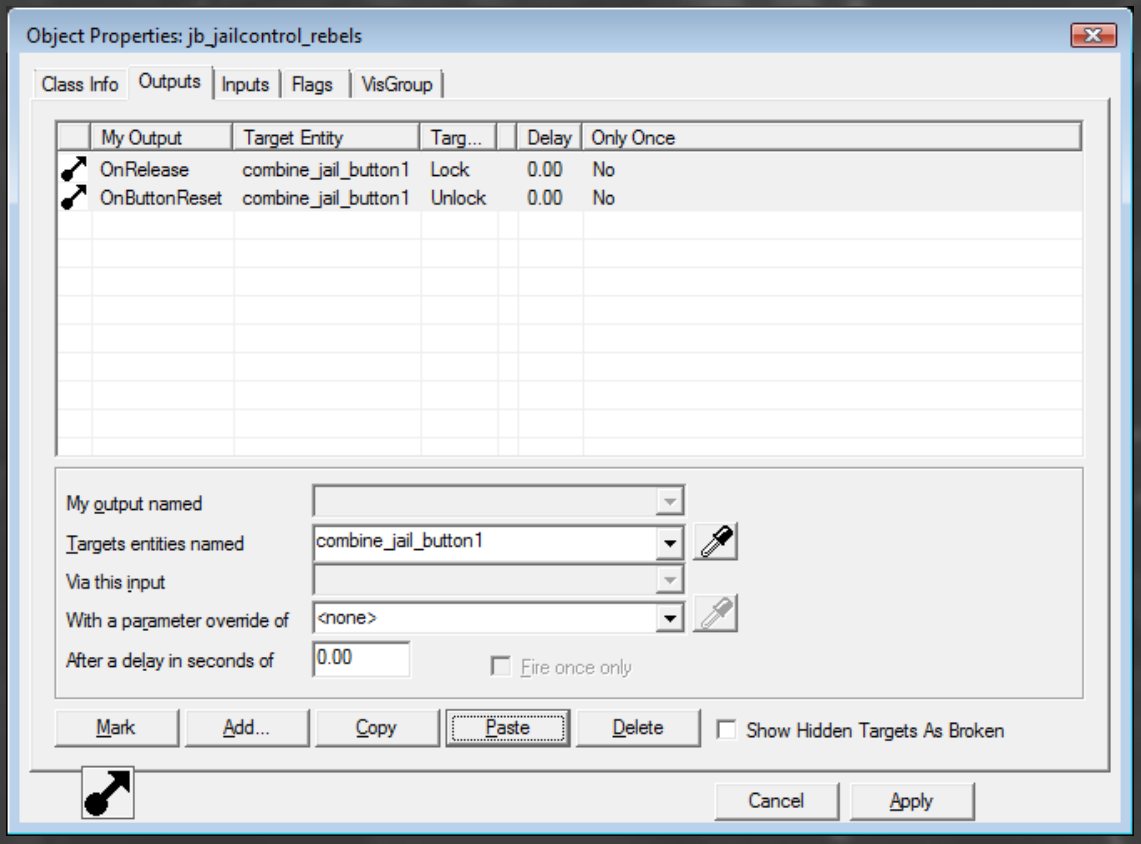
vent players spamming the release button. So using the options input the following (replace combine_jail_button1 with the name of your own button).

My output named: OnRelease
Target entities named: combine_jail_button1
Via this Input: Lock
After a delay in seconds of: 0

Then create a second Output with the following values:

My output named: OnButtonReset
Target entities named: combine_jail_button1
Via this Input: Unlock
After a delay in seconds of: 0

This ensures the button locks and unlocks, only when the outputs are fired from the jail control entity. The length of time in between these two outputs firing is governed by the Button Reset Delay keyvalue that you already set.



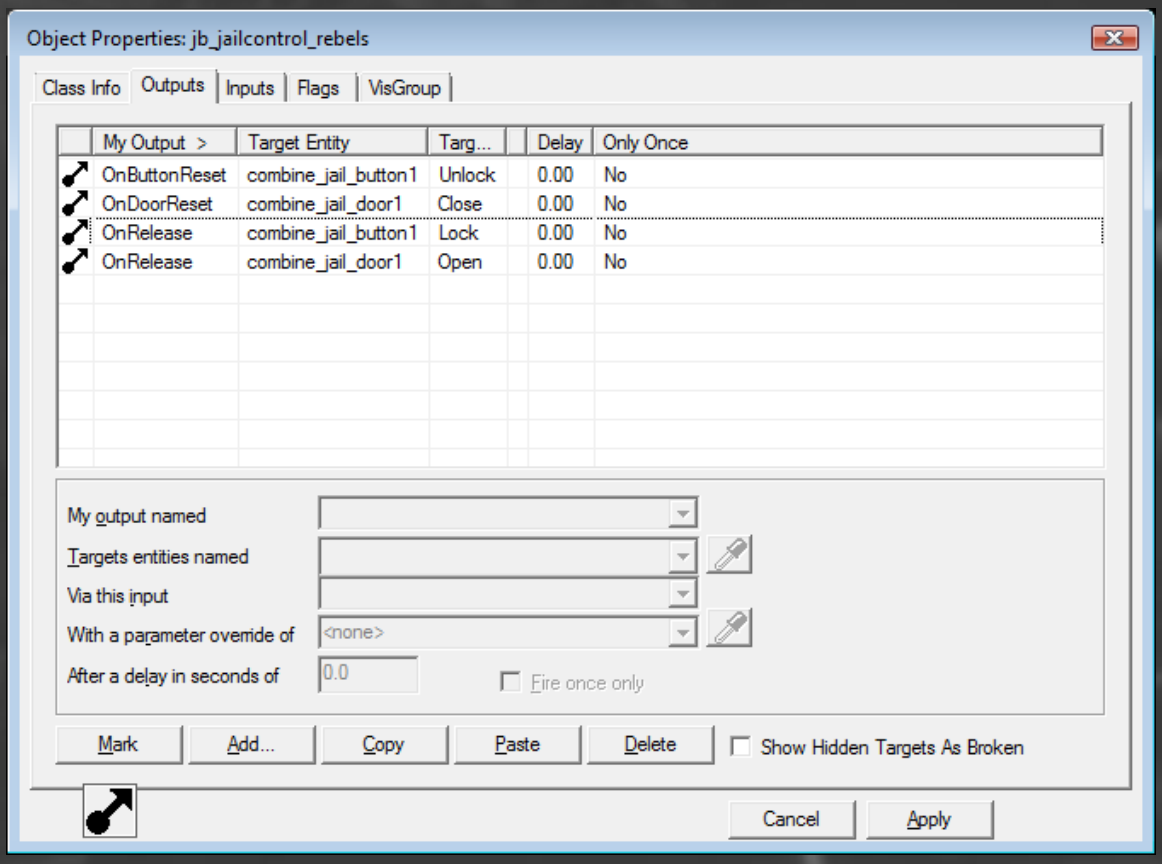
Next we need to trigger the doors to open when the release button is

pressed, and to close when the release duration is over. So add another Output and set the following options:

My output named: OnRelease
Target entities named: combine_jail_door1
Via this Input: Open
After a delay in seconds of: 0

Then add a second Output to close the doors:

My output named: OnDoorReset
Target entities named: combine_jail_door1
Via this Input: Close
After a delay in seconds of: 0



With these outputs in place, you have the basics sorted. You can add much more to the Jail Control Output system though, so take a look at the included VMF of Abalos to see what else we've done.

Escape Routes:

Jailed players can escape via inbuilt escape routes in all of the official maps. We strongly suggest you build escape routes in your own maps, mainly because they're fun, but also because players sitting around doing absolutely nothing is a situation we'd like to avoid!

So, what makes a good escape route? An achievable challenge. You need to create something for players to navigate or puzzle their way through, that is challenging, either in a physical or mental way, but is achievable using either skill or time.

Take for example, the escape route in Arroyo. Players must jump across three platforms, on either side of a large ventilation system. As soon as a player touches a platform, it begins to retract, forcing them to time their jumps and momentum to precisely land on the final platform and make their escape. Once through this, the players must quickly get to the end of a long tunnel, in order to get out of jail.

This escape route has three elements to it:

- Challenges the players movement skills
- Slows down the amount of players escaping as players behind the successful escapee must wait for the platforms to reset
- Time restraint, as the players must travel a long corridor to get out, giving defenders a chance to defend or win the round.

Escape routes should never be too difficult, there should always be an element of hope! Otherwise you risk frustrating new players. To implement an escape route, you need to create the following:

- An escape route entrance (with a door)
- An escape route area (with the jb_trigger_jail volume)
- An escape route exit (blocked so enemies cannot get inside)

Escape Cool Down System:

For 0.6 we have implemented an Escape Cool Down system, this means that when a player presses the release button, the escape entrance door closes for set amount of time (specified by the mapper), and then reopens

once that time has passed.

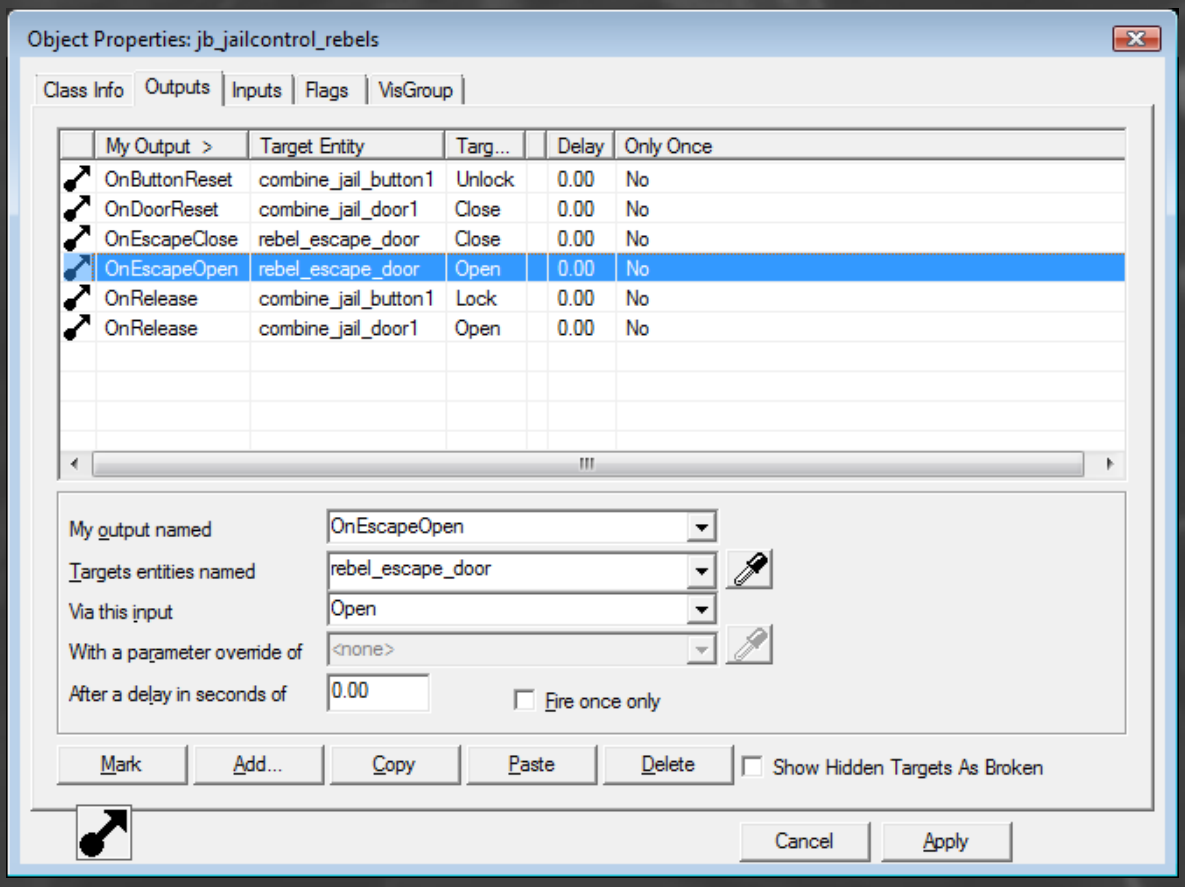
There are several reasons for this:

- Players were often inside the escape route when “released”.
- Only *some* players would get “released” as a result, dividing the team and making for uncoordinated games.
- Base swapping would occur too often as a result of escapees overwhelming defending players in ones and two’s.
- The benefit is that players either *escape* as a team or get *released* as a team, potentially turning the tide of the round.

To implement the escape cool down, we use the **jb_jailcontrol** entities, using the Output system explained above.

Having built your escape, create another func_door, with the same properties as the Jail Doors, and name it something like *rebel_escape_door*.

Return to your jb_jailcontrol_rebels entity, and go to the Outputs tab, then



add two new outputs for OnEscapeOpen and OnEscapeClose, based on the image below.
These are the outputs used in the image above:

My output named: OnEscapeOpen
Target entities named: rebel_escape_door
Via this Input: Open
After a delay in seconds of: 0

Then create a second Output with the following:

My output named: OnEscapeClose
Target entities named: rebel_escape_door
Via this Input: Close
After a delay in seconds of: 0

Now go back to the KeyValues (Class Info tab), and specify the amount of time you want the escapes to close for using the Escape Reset Delay key-value. To create a countdown timer, place a jb_info_objective entity over the escape door, and simply parent it to your jail_control entity. Easy as that!

Executions:

By now you should have got a good grasp of how to use the Input/Output system in Hammer, so it's time to start creating the executions.

Place a single jb_execute_rebels entity near to your release button entities (to make them easy to find). Then open up the KeyValues (ALT-ENTER). You will see two KeyValues: Name and Duration.

- Name allows this entity to fire Outputs.
- Duration specifies the length of the execution in seconds.

The entities outputs are fired by setting OnExecute in the Outputs tab. For example, to execute the Dinosaurs in your map by starting a fire place several env_fire entities in your jail, all named fire1, then one Output as follows:

OnExecute; fire1; StartFire; 0.00 Seconds.

You can set as many Outputs as you want using this system.
Don't stop there with the executions though, be as creative as you can be, as the Input/Output system allows for all sorts of insane, complicated and devious execution. In Metropolis, the execution entity triggers a Helicopter to take off, follow a set route, fire upon enemies, and then drop a series of bombs into the jail. This explosions then destroys the outer walls of the jail and kills all the players inside!

British Bulldog Executions:

For the latest release, we have implemented a second way of executing prisoners. This method allows you to specify a new location for either or both teams to spawn in, allowing to get as creative as you want with the executions. You can specify whether just the losing team spawns, or if both teams spawn, and if they get to keep their weapons when they respawn.

This method allows you to let the winning team be an interactive part of the execution, letting them trigger events themselves.

Examples of BBD Executions:

For example, in **Woodneck**, players respawn inside a Weather Testing Facility. Losing players respawn inside a giant funnel in the ground, whilst winning players respawn in front of a rack of control panels. The winners then get to select which method of execution the losers go through, or a combination! Pressing one button creates a giant tornado, which sucks the players up and out of the tube, impacting upon the ceiling and killing them instantly.

Abalos on the other hand respawns the losers in the path of an oncoming train, whilst the winners watch from cameras mounted along the tracks and attached to the train itself!

Creating a BBD Execution is fairly straightforward. Just open the Flags tab of your execution entity, and check the flags for "Respawn Winners", "Respawn Losers" and choose whether to Strip Weapons or not.

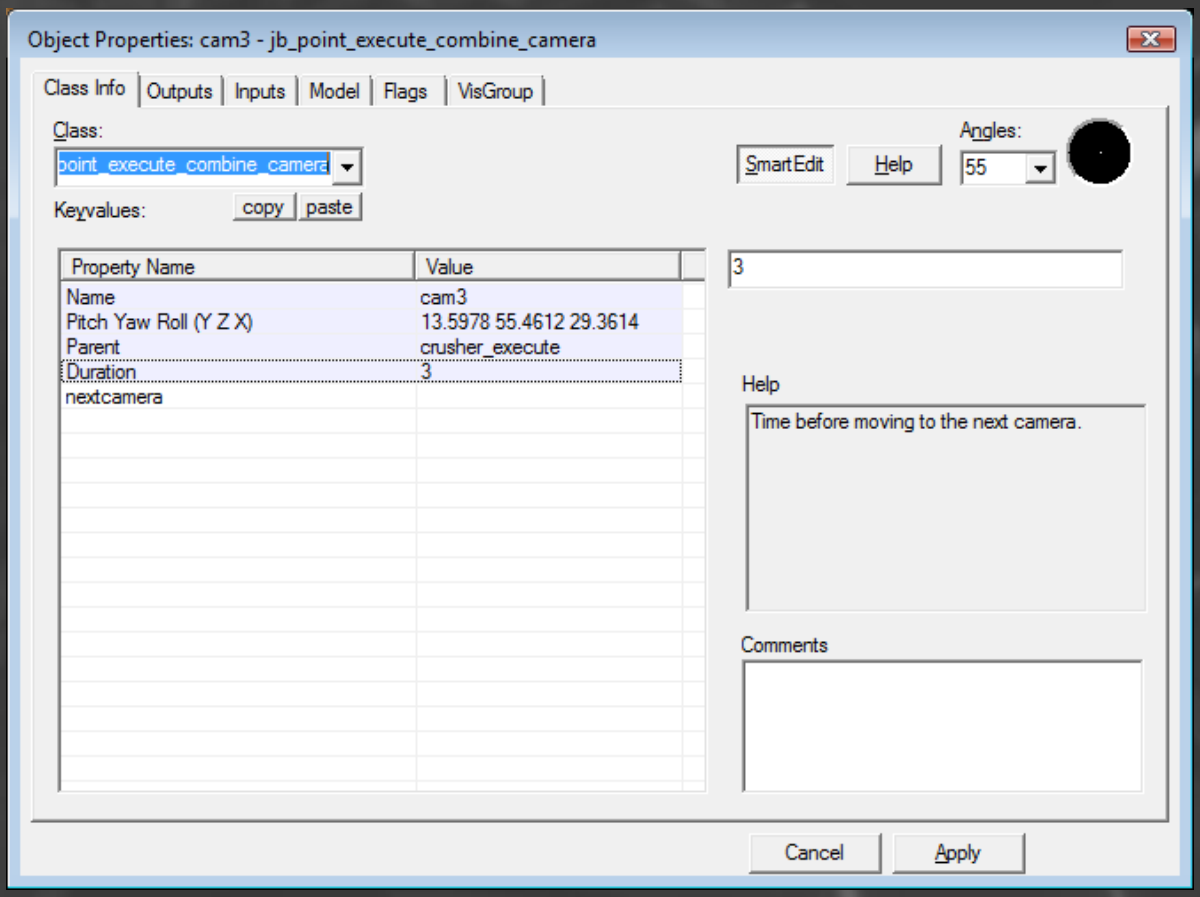
Then just place info_loser_spawn and jb_info_winner_spawn points inside the new area created for the execution, or within your map somewhere.

Execution Cameras:

In both normal and BBD Executions, cameras can be used for the winning team to observe the losing team being executed. To place cameras, use the Entity Tool and place a jb_point_execute_rebels_camera (for the Robot team to watch the Dinosaurs being executed).

The entity is in the form of a small movie cameras, you can angle the camera to any orientation, and you can parent it to moving objects to create moving cameras.

The entity KeyValues contain a Duration entry and a NextCamera entry, allowing you to chain cameras together into a movie sequence. To create a camera chain, you must specify which camera is the “First Camera” in the Flags tab. Then name each camera in turn, and specify how long the camera is active for, and which camera the players should view once the duration (in seconds) has been reached.



Game Mode: Deathball

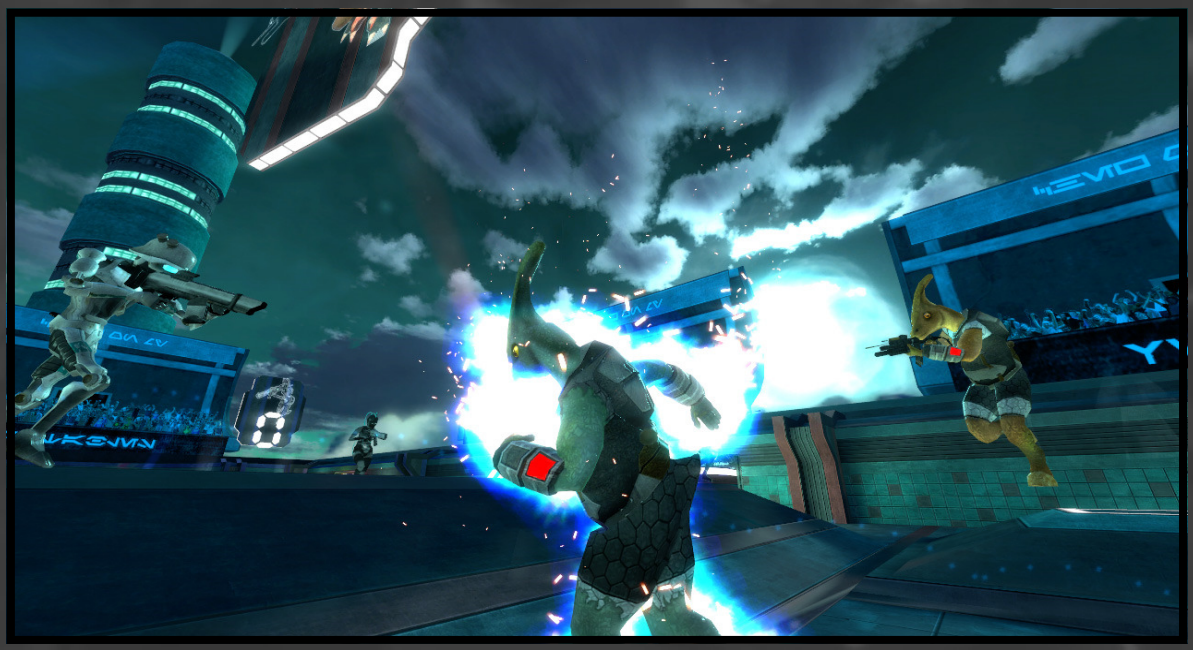
Deathball is a new game mode for Jailbreak: Source 0.6, introducing a new method of playing Jailbreak. It is designed as a sort of futuristic Blood Sport. The arena has two Goals, and two Jails, and in the middle is a giant Deathball. The ball can only be pushed by either shooting or hitting it, it cannot be picked up and held.

If you are killed in Deathball, you respawn in Jail… but you can *let yourself out*, if the ball *isn't* in the goal. Your team must push the Deathball into the opposite teams goal. Once the ball is inside the goal, the enemy jail becomes locked, and they can no longer get out. The remaining enemy players must push the ball out of their goal to release their team mates, and your team must kill them all before they do!

Think of the ball as a Key, once it's inside the Lock, the Jail is closed. Remove the Key, and the Jail opens!

Deathball Layout:

As Deathball is a competitive sport, all Deathball arenas must be symmetrical, ensuring a fair playing field. There are also other elements to Deathball, such as Jump Pads and Sink Holes, allowing you to rapidly move yourself and the ball around the level at high speed.



Creating a Deathball Arena:

Creating a Deathball Arena is not an easy process. The original system was built over several weeks by a rather clever chap called Nyght. The system uses a combination of logic_auto entities alongside side team filters and trigger volumes to detect where and what the ball is doing.

The easiest and fastest way to create a Deathball map is to copy and paste the entities from the example VMF included in this pack into your own map, then adjust their locations and settings to suit your needs.

Chained I/O System:

In principle, the system consists of a trigger_volume inside the jail area, which outputs to a filter system which allows only entities by the name of “Deathball” to trigger the outputs from the filter. The outputs then lock the jail escape button whilst the Deathball is inside the volume, and once it leaves (onEndTouch) the volume, the outputs unlock the doors.

This output also controls the onscreen HUD messages, telling players who has scored, and what’s happening. Finally, in the case of db_bleach, the output also adds a point to the giant scoreboards.

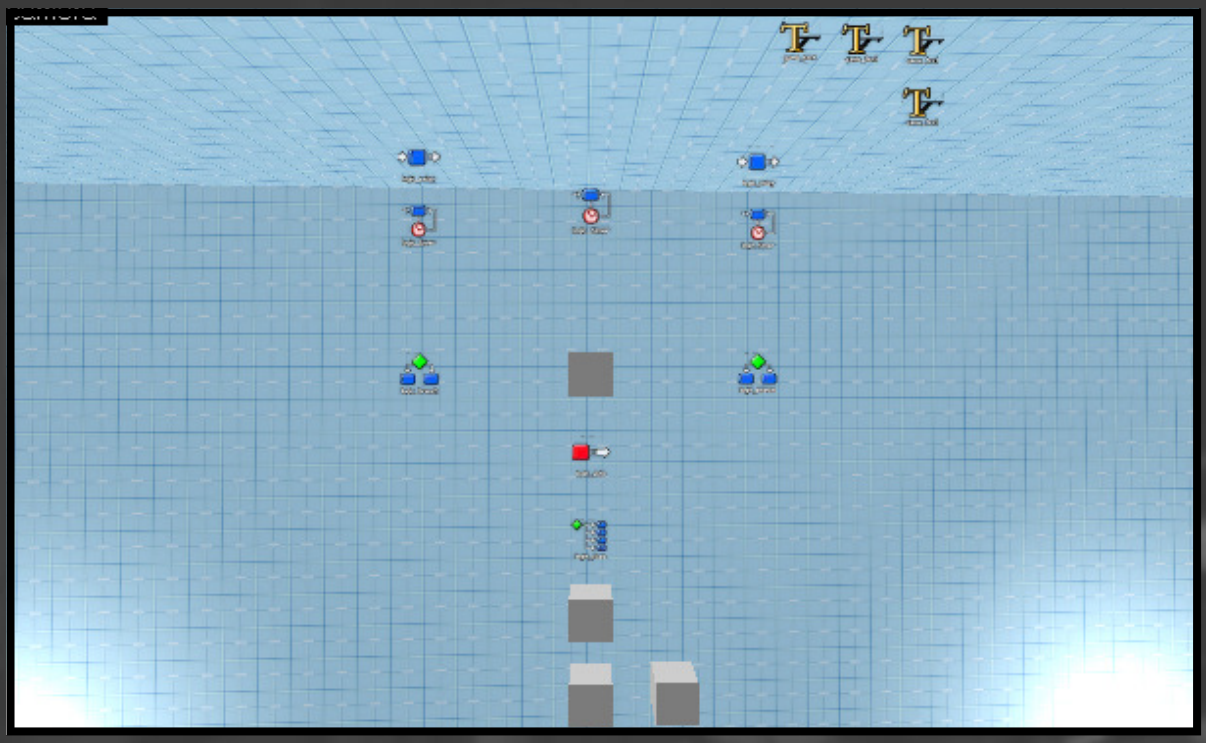
No Escapes:

The jails themselves do not have an escape route as the button is located inside the jail. This button is not linked to the jailcontrol entity as in a Standard Jailbreak map, but a jailcontrol entity is still present in order to control Lockdown, if a round goes on too long. The Lockdown overrides the filter system, and shuts the jails permanently, creating a last team standing sudden death mode.

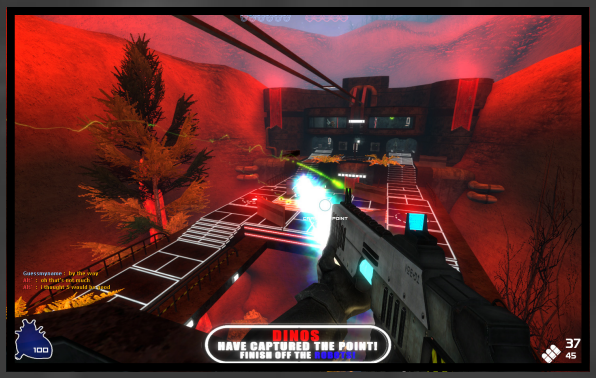
As you can see, it’s quite a complex and lengthy chained system, so it’s easiest and safest to simply import the entities into your map from the example VMF and adjust the trigger volume shape, size and location to suit the size of your goals.

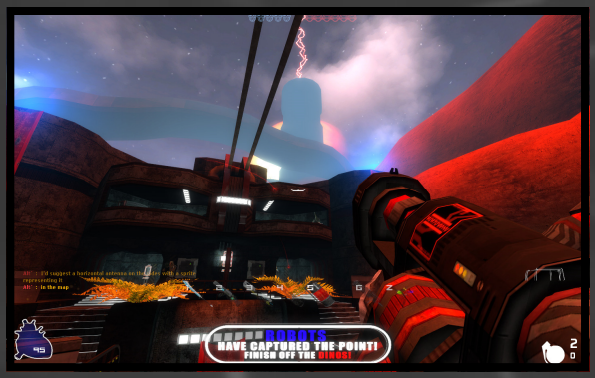
Game Mode: King of the Hill

King of the Hill Jailbreak is the second new game mode implemented into Jailbreak: Source 0.6, this time courtesy of Guessmyname. The concept is similar to Team Fortress 2's capture point game mode. Your team must capture and hold The Hill in the centre of the map. Once the Hill is captured, the enemy jail closes, and yours opens. Your team must hold the hill for the duration of the game in order to win.



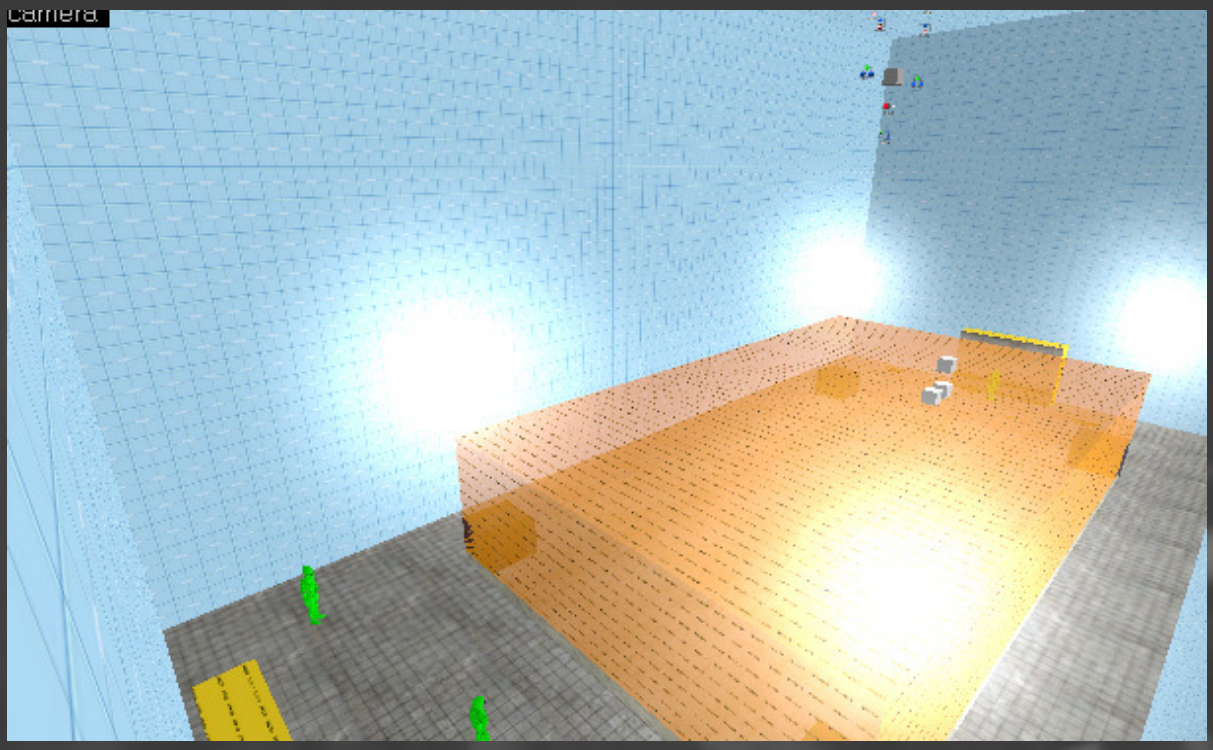
If the hill is not held, or becomes contested by an enemy player, both jails close, and the escape routes open. If the enemy team has the hill, your teams escape routes open up, allowing them to run for freedom!





Creating a King of the Hill Map:

This game mode is again, a very complex series of logic_auto's and team filter entities, which I'll do my best to explain below. However, as with Deathball, it's easier if you copy and paste the entities from the example VMF into your own map, as the entity system shown below is rather complex. The only thing you will need to do is adjust is the location and size of the four overlaid trigger volumes that define the Hill's location.



Both bases should have escape routes, which are opened up when the enemy team takes control of the hill. Escape routes are only open when the jail doors are shut. These escape routes close the moment your team takes the hill, and the jail doors open up.

The Hill itself gives every player standing on it bonus points for every five seconds they are in possession of the hill, so it's important that these players are able to defend, but are in a vulnerable position to make it possible for attacking players to overcome them and retake the hill.

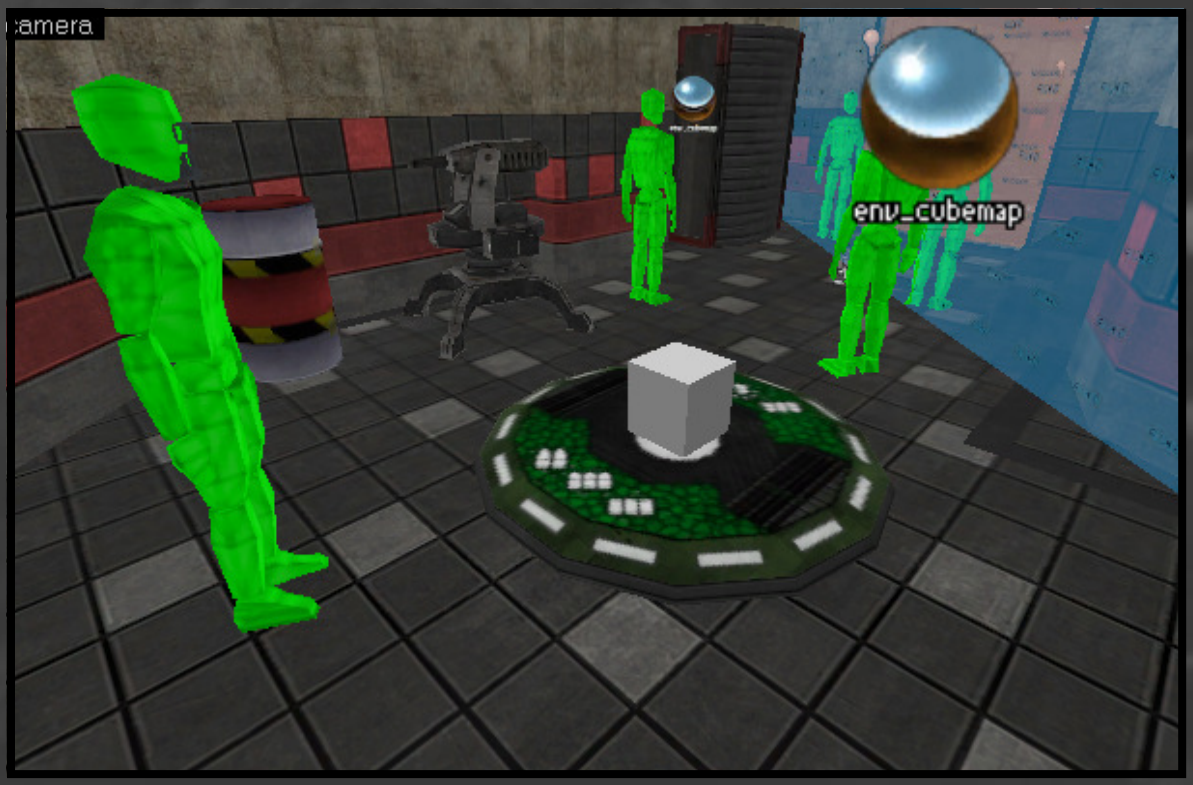
When the hill is contested, both teams jail doors close, and the escape routes open up, giving both teams a chance to get out if they're quick!

Weapons & Ammo

Ammo Dispensers:

One of the biggest changes to Jailbreak: Source 0.6 is the inclusion of Ammo Dispensers. These round ammo pads can be found in the bases of each team, and occasionally in the centre of a map. Players stand over the ammo pad and are dispensed ammo at varying rates depending on the power of the weapons they are currently holding.

This does away with the need to remember which types of ammo work for the different weapons in the game, as all ammo types are dispensed at the pads. It also forces players back to their own bases to rearm, as they cannot rearm from enemy dispensers. Ammo Dispensers can be set to be either team dependent or for any team using the KeyValues (ALT-ENTER).



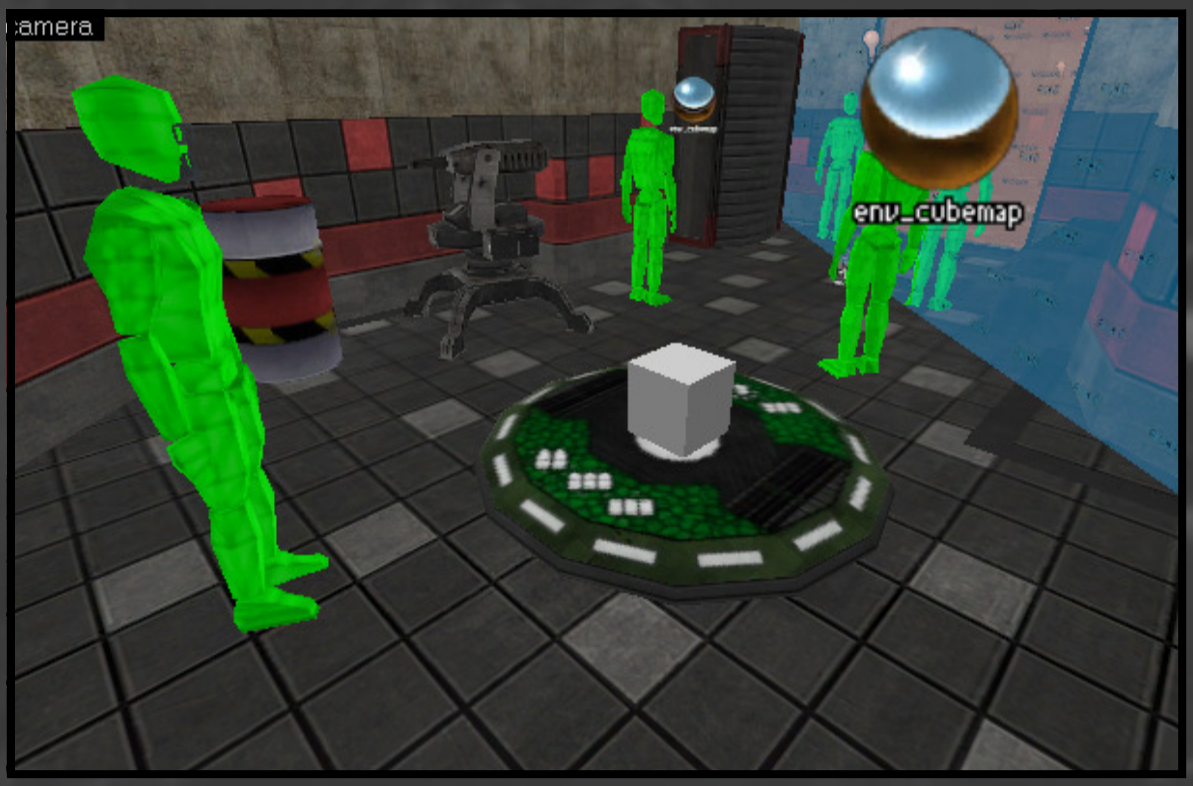
We have also placed **jb_info_objective** markers on top of the Ammo Pads in our maps, with a low (512) distance setting, so that players can see where the nearest Ammo Pad is located through walls. These are also team specific.

Weapon Dispensers:

Another similar change is the way Weapons are picked up. Instead of weapons lying randomly on the floor throughout the levels, you can pick up or swap weapons from weapon dispensers.

The KeyValues of a weapon dispenser allow you set, using a drop down box, which weapon will be dispensed. You can also set the dispenser to dispense random weapons each time it respawns, alongside the respawn duration/rate.

The weapon pads automatically emit the correct particle effect depending on their setting in the KeyValue's so again you needn't worry about that!



Random weapons can be useful to ensure that new players get a fair chance when they first start playing the game.

Note: We haven't removed the standard weapon entities from 0.6, so you can still place weapons individually if you wish.

Entity List:

Below is the list of entities used in Jailbreak: Source 0.4

info_player_rebel:
Spawn points for the Rebels; these should be placed where you want the Rebel side to spawn. It is best to place sixteen.

jb_info_player_rebel_jail:
Jail spawn points for the Rebels; these should be placed where you want the Rebel side to spawn in jail. It is best to place sixteen.

info_player_combine:
Spawn points for the Combine; these should be placed where you want the Combine side to spawn. It is best to place sixteen.

jb_info_player_combine_jail:
Jail spawn points for the Combine; these should be placed where you want the Combine side to spawn in jail. It is best to place sixteen.

jb_execute_rebels:
This is an Input/Output entity used to control the execution of the Rebel side. The entities outputs are fired by setting the following: OnExecute. For example, to execute the Rebels in your map by starting a fire (providing you have placed env_fire entities in the Jail and named them "fire1"), you would set one Output as follows:

OnExecute; fire1; StartFire; 0.00 Seconds. You can set as many Outputs as you want using this system.

jb_execute_combine:
This is an Input/Output entity used to control the execution of the Combine side. The entities outputs are fired by setting the following: OnExecute. For example, to execute the Combine in your map by gassing them (providing you have placed env_steam entities in the Jail and named them "gas1"), you would set one Output as follows:

OnExecute; gas1; TurnOn; 0.00 Seconds. You can set as many Outputs as you want using this system.

jb_jailcontrol_rebels:
This is an Input/Output entity used to control the release of the Rebel side. You should create a button which has one Output of: OnPressed; jb_release_rebels; FireRelease to trigger this entity. The entities outputs are fired by setting the following: OnRelease and OnDoorReset. For example, to open a Jail Door (named rebel_jail_door1) in your map, you would set two Outputs as follows:

OnRelease; rebel_jail_door1; Unlock; 0.00 Seconds
OnRelease; rebel_jail_door1; Open; 0.50 Seconds.

To close the doors use OnDoorReset. You can set as many Outputs as you want using this system. You can also set the Duration of the release in the KeyValues fields.

You should also set the following Outputs to disable your button for the desired amount of time specified in Button Reset Delay KeyValue:

OnRelease; rebel_release_button; Lock; 0.00 Seconds
OnButtonReset; rebel_release_button; Unlock; 0.50 Seconds.

jb_jailcontrol_combine:
This is an Input/Output entity used to control the release of the Combine side. You should create a button which has a single Output of: OnPressed; jb_release_combine; FireRelease to trigger this entity. The entities outputs are fired by setting the following: OnRelease. For example, to open a Jail Door (named combine_jail_door1) in your map, you would set two Outputs as follows:

OnRelease; combine_jail_door1; Unlock; 0.00 Seconds
OnRelease; combine_jail_door1; Open; 0.50 Seconds.

To close the doors use OnDoorReset. You can set as many Outputs as you want using this system. You can also set the Duration of the release in the KeyValues fields.

You should also set the following Outputs to disable your button for the desired amount of time specified in Button Reset Delay KeyValue:

OnRelease; combine_release_button; Lock; 0.00 Seconds

OnButtonReset; combine_release_button; Unlock; 0.50 Seconds.

jb_info_objective
This entity should be placed over the release button at the Combine Jail (where the Rebels are imprisoned) as it is used to govern the position of the release triangle on the players HUD and on the minimap overview.

jb_trigger_jail:
This is a trigger entity applied by creating a brush (or multiple linked brushes) with the "trigger" texture applied. Once you have covered the entire area of your Jail in this brush, select each of the brushes using CTRL-SELECT then press CTRL-T or press "To Entity" on the right hand side of the Hammer screen.

You can then select and apply jb_trigger_jail from the drop down menu in the Object Properties dialogue box. Once applied, you can name the entity (e.g. combine_jail). You must also specify the teamJailed in the KeyValues (e.g. Combine).

jb_point_execute_combine:
This entity is used to define the position of the execution camera for the Combine execution. It should be placed in the jail where the Combine are executed.

jb_point_execute_rebels:
This entity is used to define the position of the execution camera for the Rebel execution. It should be placed in the jail where the Rebels are executed.

jb_turret_floor:
This is the turret entity, place this where you like. You can set it to start as active for the Combine or Rebels using the Start Team Number KeyValue. It can also be set to None, this option will make the turret active for the first team to pick it up.

jb_trigger_player_location:
This entity is used to name and define the area a player is currently in, this area is shown when a Radio or Team message is sent during play. This is a trigger entity applied by creating a brush with the "trigger" texture applied.

You should create two jb_trigger_player_location entities next to each other covering the playable area that a player might walk through when entering your new location from either side.

You should name the entity (e.g. Dive Pool) in the Name KeyValue, and specify the name to be displayed in the locationname KeyValue (e.g. Dive Pool). The last trigger a player touches defines the location they are currently in, so these entities should be placed in doorways or streets which enter or exit the area. Look at the example map for an idea of how this is best achieved.

Weapon, Item and Ammo Entities:
jb_weapon_spawner
jb_weapon_rebel_smg
jb_weapon_combine_smg
jb_weapon_combine_rifle
jb_weapon_chaingun
jb_weapon_medigun
jb_weapon_silenced_pistol
jb_weapon_railgun
jb_weapon_autoshotgun
jb_weapon_grenadelauncher
jb_weapon_stickylauncher
weapon_rpg

jb_ammo_dispenser
jb_ammo_slug
jb_ammo_chaingun
jb_ammo_pistol
jb_ammo_medigun
jb_ammo_shotgun
jb_ammo_rifle
jb_ammo_smg
item_rpg_round

CORE TEAM:
Richard "Dogmeat" Slaughter:
Coding and General Development

Andrew "Wills" Wills:
Mapping, Textures, Graphics, Skins, Modelling, Sounds, Voices

Rob "IZaNaGI" Molde:
Mapping, Textures, Graphics, Skins, Weapon & Prop Modelling, Sounds

Axel "Azelito" Dahlstedt:
HUD Graphics & Icons, Weapon Modelling (Rebel SMG)

CONTRIBUTING TEAM MEMBERS:
Jahhur - Some sexy particle effects
Laura - Voices, Additional Textures, Graphics, Ideas, Problem Solving
Colonel Sanders - Music and Soundtrack for all maps
Mike Gomersall - Main Menu and Deathball Music
Nyght - For creating the Deathball concept and I/O system
Guessmyname - For creating the King of the Hill concept and I/O system
Sherbie - For the Icons, TTF Files and Crosshairs

SPECIAL THANKS TO THE JAILBREAK WIVES & GIRLFRIENDS:
Nikki & Sanna

SPECIAL THANKS TO:
Blink - For the hosting assistance over at Interlopers beta testing with us!
Stenchy, INtense and Henley at ModDB - For everything they've done to help promote Jailbreak!
UberMensch and Jay (Gmodhost) - For hosting the Official 0.4 Servers!
Jahhur - For suggesting solutions to a few of the FPS sapping bugs in 0.5
Steve Foster - For the music for the main Jailbreak Trailer.
Mike Gomersall - For the music for the Deathball Trailer.
Tony Sergi - For once again helping us out in the final stages of 0.5 / 0.6!

BETA TESTERS:
Altaire, Asskicker, Blink, Colonel Sanders, damatman, DistortedGoat, Draken, Freshy Fresh, Guessmyname, Jahhur, Jayjack, Mastix, nipzor^, nyght, Oceed, op89x, Panther, pretty_little_geek_girl, qpacman, Qu1f, Sherbie, Tomroc and winter.