Category:Mapping

This page contains links to various articles on mapping for Dystopia, designed for third party mappers.

If you can't find answers here, then take a look at the Valve Developer Network, especially their level design section (although the others can be helpful too).

If you have never made a Dystopia map, use this article as a guide.

Subcategories

This category has only the following subcategory.

D

- Dystopia Entity Guide

Pages in category "Mapping"

The following 17 pages are in this category, out of 17 total.

A

- Adding Jack-In Points

B

- Before You Start

C

- CTF Mapping Guide
- Climbable Surfaces
- Creating Objectives

M

- Map paths
- Map testing errors
- Mapping Measurements

O

- Overviews

P

- Phistball Mapping

R

- RES Structure Guide

S

- Screen Setup
- Setting Hammer Up for Dystopia
- Spawn points

T

- Toggle Screens
- Turret setup

Y

- Your First Dystopia Map

Retrieved from "http://www.dystopia-game.com/wiki/index.php?title=Category:Mapping"

Categories: Support Pages | Updated Game Info

navigation

- Main Page
- Community portal
- Current events
- Recent changes
- Random page
- Help

search

Go Search

toolbox

- What links here
- Related changes
- Special pages
- Printable version
- Permanent link

Category:Dystopia Entity Guide

Cant Find the entity you are looking for ? it probably hasnt been imported over to the wiki yet, but can be found here : http://dystopia-game.com/forum/viewtopic.php?t=2356 . That thread contains all the new entities that Dystopia introduces into mapping that havent been imported into here yet. Very desciptive and provides lots of examples on how create certain groups of entities.

Subcategories

This category has only the following subcategory.

C

- Cyberspace Entities

Pages in category "Dystopia Entity Guide"

The following 28 pages are in this category, out of 28 total.

C

- Cyber drain
- Cyber floor
- Cyber gravity
- Cyber ice
- Cyber jumppad
- Cyber speedpad

D

- Dys ammodisp
- Dys cyberscreen
- Dys emp

D cont.

- Dys forcefield
- Dys forceloadout
- Dys helper
- Dys item
- Dys jackpoint
- Dys objective
- Dys onscreeninfo
- Dys screen
- Dys spawn
- Dys spawn point

D cont.

- Dys spidershooter

F

- Func monitor

I

- Info camera corpswin
- Info camera punkswin

N

- Npc turret ceiling

P

- Point camera

S

- Sky camera cyber

T

- Trigger item

W

- Weapon lightning

Retrieved from "http://www.dystopia-game.com/wiki/index.php?title=Category:Dystopia_Entity_Guide"

Categories: Support Pages | Mapping

navigation

- Main Page
- Community portal
- Current events
- Recent changes
- Random page
- Help

search

Go Search

toolbox

- What links here
- Related changes
- Special pages
- Printable version
- Permanent link

Category:Cyberspace Entities

This is a category of the entities used in the creation of cyberspace.

Pages in category "Cyberspace Entities"

The following 9 pages are in this category, out of 9 total.

C

- Cyber drain
- Cyber floor
- Cyber gravity

C cont.

- Cyber ice
- Cyber jumppad
- Cyber speedpad

D

- Dys cyberscreen

D cont.

- Dys forcefield

S

- Sky camera cyber

Retrieved from "http://www.dystopia-game.com/wiki/index.php?title=Category:Cyberspace_Entities"

Categories: Support Pages | Dystopia Entity Guide

navigation

- Main Page
- Community portal
- Current events
- Recent changes
- Random page
- Help

search

Go Search

toolbox

- What links here
- Related changes
- Special pages
- Printable version
- Permanent link

Adding Jack-In Points

The Jack-In-Point

Jack-in points are the link in Dystopia maps between the real world (meatspace) and cyberspace (the network). Players must use either the cyberdeck or enhanced cyberdeck implant near a JIP to enter cyberspace.

Components of a Jack-In-Point

In order to add a functioning jack-in point (JIP) to a map, *two* entities are *required*:

1. A `dys_jackpoint`, a brush entity placed in meatspace
2. A `point_camera`, a point entity in cyberspace

Many jack-in points also come with a screen, allowing other players to watch the decker fight in cyberspace, which is aligned inside a metal panel. This is *not* required, but makes the jack-in point easily recognizable. Don't feel restricted to this method, however; creative JIPs can spice up a map and make it more enjoyable.

Steps to Create a Standard JIP

1. Create a `prop_static`, and set its world model in the properties menu to `prop_jackin_*.mdl` (for the purpose of this tutorial, * must be replaced with either `public`, `dtrust`, or `punks`). Place this panel in meatspace at the desired location for the JIP.
   - Note: `prop_jackin_public.mdl` is known to have a bad collision model, and should not be used.
2. Select the `tools/nodraw` texture, and create a block brush using the selected `tools/nodraw` texture with dimensions *30x30x2* (Height x Width x Thickness). Line the brush up inside the model as shown in the picture, so that the brush rests inside the depression of the jack-in panel.
3. Create a `point_camera` in cyberspace, and name the camera in the **name** field so that it can be used in *Step 6*. It is a good idea to prefix the name with `jipcam` so that it can be easily identified. In **The camera render texture target.** field, enter `camera1`, `camera2`, or `camera3`. This texture is used in *Step 7* to create the screen.

   Note: `camera4` is used by the rocket launcher's screen, so it should not be used (sometimes it is okay in cyberspace).
4. A player will "jack-in" facing the direction that the camera points, with gravity perpendicular to the bottom face of the camera. Rotate the `point_camera` and move it so that the player will jack-in at the desired location, with correct gravity.
5. Select the brush created in *Step 2* and convert it into the `dys_jackpoint` brush entity; the default shortcut for tying a brush to an entity is Ctrl+T.
6. Leave all the fields in the object properties menu default, except for **Camera Name**. Enter the name of the `point_camera` created in *Step 3* in the **Camera Name** field.
7. Using the Face Edit Dialog, select the front face of the `[dys_jackpoint]`, which the player views. Then select the texture `dys_monitor1a`, `dys_monitor2a`, or `dys_monitor3a`; in respect to the camera number you gave the `point_camera` in *Step 3*.

The brush aligned inside the panel

Retrieved from "http://www.dystopia-game.com/wiki/index.php?title=Adding_Jack-In_Points"

Categories: Support Pages | Updated Game Info | Mapping

navigation
- Main Page
- Community portal
- Current events
- Recent changes
- Random page
- Help

search
Go Search

toolbox
- What links here
- Related changes
- Special pages
- Printable version
- Permanent link

Before You Start

This is a slightly updated version of the post that used to adorn the top of the internal mapping forum, as part of the welcome post to new mappers on the team. I hope it will now help all the new third party mappers too. -Tim 'Termi' Grant

Self Preparation

Things to ask yourself before starting to map for Dystopia... -IcePick66

Making a Dystopia map is very different from other games, like Counter Strike: Source. It requires extreme planning and thinking outside the box. If you can't be bothered to think and draw up ideas for your map, you might as well not bother to even start.

Make sure to get a good idea to hold on to, and try to imagine the finished map in your head. *"Mapper's block"* is the single biggest killer of Dystopia maps.

I recommend getting an I-Google widget, like Stickynote; something that you can use to record your ideas as soon as you think of them. Google's one is particularly good, because you don't have to save or open individual files; it's just there every time you visit I-Google. Alternatively, just put a notepad file called `ideas.txt` or something similar on your desktop, and modify it at your leisure.

Also, discuss ideas with other people! Mappers are always on IRC, so be sure to get their opinions on things. Peer review and constructive criticism can be life-saving as far as level development goes.

Make time; get yourself committed to finish at least a beta, that's the first hurdle. Once you have done that, you can start a new topic for your map in the mapping forum, and even a wiki page for it. (Be sure to follow the style set by other posts).

Keep updated; spend at least a little time one day each week working on the core features of the map. This way you won't get bored of the same map and give up! However, try not to map out all the "fun stuff" first. Get the basic shell down in hammer, then work on details and fun stuff. Don't leave *all* the detailing until the later stages, otherwise you will get very bored! (Try to keep early detail modular, so you don't have to scrap it entirely when making an unplanned layout change).

Be realistic; even thinking about starting a map is a lot of hard work and effort. Make sure you have some decent knowledge of mapping before you complete the planning stage. Otherwise, you will end up with your hands tied down the road, trying to figure out how to create a major feature.

If you *are* a beginner, it's probably best to start off mapping for games that require less extensive knowlege of Hammer, like Counter Strike:Source or Half Life:Deathmatch. Even for good mappers, the step up into Dystopia is a hard one!

Map Preparation

Things to think about before diving in and making a map... -Termi

1. Think of your objectives in detail.
   - *What events* do you want to orchestrate in your map? *What kind of battles* do you want around them? *Where* should they take place? *How long* should they last? *How biased* should they be to either side to maintain overall balance?
2. Think of gameflow and layout.
   - *What theme* do you want the map to be based upon? Is it a fight between two sky scrapers? A maintenance area rumble? A hot pursuit up flights of stairs?
3. Think of reference areas, each with a distinguishing feature, that can be summed up in a few words.
   - You may think of the old "flag room" or "ramp room" from other multiplayer games. In Dystopia, there are obviously more descriptive ones; like "pipe room", "maintenance room", "the wall", "the gap", and "that room with the glowy thing"; because of the relatively large size of the maps. These areas help players identify where they are and make communication easier. Because they cannot be explicitly named, giving each one a prominent feature will help players make up their own terminology.
4. Always make *one definite path* to the next objective first, and then possibly one or two *alternate* paths.
   - As we heard at Valve: "Don't give the players too many choices, they will get lost and bored. Don't waste time on rooms that serve no purpose whatsoever. Give the illusion of choice rather than true choice."

As always, if you have any troubles with *anything*, from entities to "Why the fuck isn't my map compiling/loading in hl2/opening in hammer?!?", ask in the third party mapping forum, IRC, or just contact one of the mappers on the team.

Mapping Efficiently

1. Keep your brushwork clean.
   - Keep your grid to a decent size, the higher the better (generally speaking). Keep in mind that brushes bigger than 1024x1024x1024 will be broken up into 1024x1024x1024 chunks by the compiler. Usually, this will badly add to your poly count. The general rule of thumb is that, if you make the cuts to the brushes yourself, they will be more effective than whichever arbitrary way the compiler decides to throw them.
2. Use dev textures until the block-out and game-flow stuff is done.
   - Yes it's ugly. Yes, it feels like they are raping your eyes. Yes your testers will bitch about it. But they will help you keep your proportions right, and let you get your block-out done faster without worrying about the details. They will speed up the creation of prototypes and revisions, too.
3. Use prefabs.
   - There are a bunch of handy dandy prefabs (pre-built brush layouts) in the prefabs section of Hammer. There is simple stuff, like steps and ladders in there, which make it a little easier for newbies to pick stuff up. Also, for cyberspace stuff, keep in mind the "tubekit" in the `/mapsrc` directory of the root Dystopia folder.
4. **BACK UP YOUR MAP.**
   - That means not just locally, but on the net via FTP. If they are getting big, zip them up. Just be sure to keep a copy somewhere that isn't near your main computer, or on any other computers you may have. You can even email yourself a copy to a G-Mail account or something; just make sure you have a copy of the VMF somewhere safe for hard drive failures or other catastrophes.
5. Make sure hammer is set up correctly.
   - Follow the article here on the Wiki to be sure. All your VMFs should be in the `%steamuserdir%/dystopia/Dystopia/mapsrc` directory, BSPs get copied to the `/maps` directory after compile. **Only compiled BSPs should be in** `steamuserdir%/dystopia/Dystopia/maps` (and their related screen setup file). Neat directories are easier to manage!

Further Reading

Valve has provided a huge amount of documentation with the SDK, more than most developers ever provide with dev kits to the public. Here's a rundown of things to look for:

1. **SDK REFERENCE DOCS**
   - There is a link to them in the Source SDK menu. These documents are a godsend, and the base for the rest of these resources.
2. Hammer Documentation
   - You can find this documentation on the VDC. It's almost all you should need to get started, read it well and carefully.
3. Entity I/O Overview
   - You can also find input/output documentation on the VDC. They are a complete description of how the Source engine entity input/output system works, including debugging commands for in-game use.
4. Performance and VIS Info
   - The VDC has an article on performance and visibility for keeping FPS high on your map. Also, consider reading this third party guide. If you build your level from the start with these pages in mind, you shouldn't have any problems down the line.
5. Three Dimensional Skyboxes
   - Don't forget about 3D skyboxes. They can add so much to the immersion of the map, and if your level is a location in the middle of nowhere, it will only look average without them when it could look hot with a good one.
6. Prop Type Overview
   - There are a number of different prop types in Hammer. You should read up to have a solid idea of what `prop_` entity to use where, and the limitations of each type.
7. Material Creation Walkthrough
   - Many mappers eventually want to make their own textures. All mappers should be self-reliant, at least for placeholder textures in their maps.
8. Source Entity Guide
   - Combined with the entity guide for Dystopia, the full VDC entity guide will tell you everything you need to know about entity specifics.
9. Displacement Documentation
   - Displacement in Hammer are a very useful but largely unused tool. Their primary use is for organic looking terrain.

Retrieved from "http://www.dystopia-game.com/wiki/index.php?title=Before_You_Start"

Categories: Support Pages | Updated Game Info | Mapping

navigation
- Main Page
- Community portal
- Current events
- Recent changes
- Random page
- Help

search
Go Search

toolbox
- What links here
- Related changes
- Special pages
- Printable version
- Permanent link

Climbable Surfaces

SUPPORT 1.1 VERSION

This article has been marked as needing scrutiny. Please see the talk page for details.

Why are they Necessary?

Ladders and other climbable surfaces are used by players in Dystopia to rapidly ascend or descend vertical spaces without the need of implants (Leg Boosters) or fear of fall damage. They are useful when creating chokepoints, sniper positions, or adding new routes and space in a cramped map.

Components of a Climbable Surface

A climbable surface is only required to have *one* thing to function:

1. A brush completely covered with the `tools/toolsinvisibleladder` texture.

Most climbable surfaces, such as a ladder, are also accompanied by a ladder model or detailed brushwork. These objects are useful both as a visual identifier and an aesthetic detail, so players don't appear to "climb air".

Making a Standard Ladder

1. Using the `dev/dev_measureladder01` texture, construct the base of your ladder. This will allow you to get the dimensions right.
   - It's always a good idea to texture unseen faces with `tools/toolsnodraw`, so do this to the faces of the ladder base that are touching the floor, ceiling, and wall.
2. Copy and paste the base brush, but extrude it slightly so that the old base brush sits inside of the new brush.
3. To finish, texture the entirety of the newly placed brush with `tools/toolsinvisibleladder`.

Making a Prop Ladder

When your map's game-play is balanced, you may want to change placeholders to a more detailed ladder.

1. Create a prop_static, and change its "World Model" field to "props_c17/metalladder001.mdl". Then change its "Collisions" field to "Not solid".

The ladder model against the wall

2. Align the ladder against the desired wall or incline, as shown in the picture.

3. Select the "tools/nodraw" texture and create a 24x128x6 (Width by Height by Depth) using the "tools/nodraw" texture. Align the brush against the same wall so that the ladder is inside of it.

The ladder brush with ladder texture

4. Using the Face Edit Dialog select the front face of the ladder brush, which the player climbs on. Then select the texture "tools/toolsinvisibleladder", and apply the texture, as shown in the picture.

For further or clarifying instructions, see the valve developer community here. http://developer.valvesoftware.com/wiki/Working_Ladders

Retrieved from "http://www.dystopia-game.com/wiki/index.php?title=Climbable_Surfaces"

Categories: Support Pages | Outdated Game Info | Articles Needing Scrutiny | Mapping

navigation
- Main Page
- Community portal
- Current events
- Recent changes
- Random page
- Help

search
Go Search

toolbox
- What links here
- Related changes
- Special pages
- Printable version
- Permanent link

Creating Objectives

SUPPORT

It has been proposed that this article be deleted. Please discuss any opinions about deletion in the talk page.

This article has been marked as needing scrutiny. Please see the talk page for details.

Dystopia is driven by an objective system, and so the methodology of creating these objectives approaches an art form. It takes careful and deliberate skill to meld together the perfect blend of cyberspace combat, cyberspace puzzles, meatspace combat, traps, and a triggerable object to create a great objective.

The first thing you should ask in creating your objective is probably "What is the trigger?". Essentially, the "trigger" consists of some entity or group of entities that a player must reach and interact with to complete the objective. The most basic ones are simple meatspace breakables and panels. More complicated versions involve cyberspace assault; either from a JIP that the defending team must hold or from spawns. Even more elaborate objectives can involve meatspace or cyberspace puzzles, where a series of buttons must be pressed to accomplish a goal. Physics entities can even be used, like a dys_objective or a phistball. Some involve a king-of-the-hill mentality, where a team must hold a location for a certain amount of time.

-More to come-

Categories: Support Pages | Articles to be Deleted | Articles Needing Scrutiny | Mapping

navigation
- Main Page
- Community portal
- Current events
- Recent changes
- Random page
- Help

search
Go Search

toolbox
- What links here
- Related changes
- Special pages
- Printable version
- Permanent link

CTF Mapping Guide

This is a guide to making Capture the Flag maps for Dystopia, working under version 1.2. See Gameplay Styles#CTF for more information on Capture the Flag.

Capture The Flag: In The Beginning

In the beginning, there was only one: Ctf_skyforts_b2_d. It was found deep within the hidden halls of Zeroflame's . The template for CTF maps outlined here comes from a decompile of the map, and if anyone knows who made this map, please enlighten us and give due credit to the only person who actually knew how to use dys_item. Hopefully soon, there will be a whole host of CTF maps comparable in number to that of the Gameplay Styles#Phistball maps.

How-To: What You Need

There are a few basic things which must be had for a CTF map:

- Flag
- Capture point

Easy, right? Well, unless you're new to Dystopia, you should know that NOTHING is that easy. Nothing. But I digress. There are actually only a few things which you need to make a CTF map, outside of designing the level itself, which I assume you already know how to do at least to the point which you can make a room and some spawns. What you ACTUALLY need is:

1. dys_item. This acts as the flag. This IS capture the flag, after all. Two of them, specifically, so that each team has a flag.
2. trigger_item. trigger_item is a special trigger put into Dystopia for catching players carrying dys_item, and will take the place of the capture point. You'll need two of these, one for each team.
3. filter_activator_team. This does exactly what it's name implies; it filters our inputs based on which team activates the trigger. You'll need two of these as well.
4. env_entity_maker. This spawns entities, as best as I can tell. It is coupled with the next item on the list to produce a trigger-action set.
5. point_template. point_template is exactly that, a point-based entity which takes a template from a selected item (or one of 16 possible items, to be precise)
6. logic_auto. Good old logic_auto, incalculably valuable for an equally incalculable number of things.

So, you may be saying to yourself, "he's gone mad, that's not simple at all!" Well, you're half right, being that I'm half mad, and it's half not simple. However, I aim to make it as simple as possible for you, so just keep reading, you can do it!

How-To: Get To It, Already!

First, build yourself a nice little room, or take a map you already built, or something. Now place a couple of block which will act as triggers. If you don't know how to turn blocks into entities, you should probably start with a simpler map. Set those to trigger_item. We'll configure them later.

Next, in the middle of each of those, add a dys_item. Give one of them the name 'punkflag' and the other of them the name 'corpflag', or whatever names you would use. Go down to the world model, and pick a model you think would work. There are no proper flags, so I used stop signs. Yep, stop signs. Why? Because they look kind of flag-like, in that they're a sign/banner/flag on a long pole. We'll leave the rest of the setup of the flags to later, for when we actually have everything placed. Lastly, set respawn delay to 60 seconds.

Then place next to each trigger a filter_activator_team. Name the one next to the dys_item 'punkflag', or whatever you named it, 'filterthrupunk', because it allows punks to pass the filter, and set the filter team number to punks. Likewise, set the one next to the corp's flag to 'filterthrucorps', and the team number to corps.

Next to the filters, add a point_template. Give the punk's one the name 'punkflagtemplate', and set template 1 to the punk's flag; give the corp's one the name 'corpflagtemplate', and set template 1 to the corp's flag. And next to those, add an env_entity_maker. Set the name of the punk's one to 'punkflagmaker', and the point_template to spawn to 'punkflagtemplate'; set the name of the corp's one to 'corpflagmaker', and the point_template to spawn to 'corpflagtemplate'. Aaannnddd next to those, add a logic_auto, with an output of OnMapSpawn to the respective env_entity_maker, with via the input ForceSpawn, and a delay of 5 seconds.

Now go back to the flags themselves, and set the punk's flag filter to 'filterthrucorps', and the corp's flag to 'filterthrupunks'. That way you can pick up the enemy's flag, but not your own.

After that, all you have left to do are the trigger_item outputs. Copy the one from the logic_auto to the trigger_item, except instead of OnMapSpawn, set it to OnItem.

That's it, you're done! Sort of. You still have to add a way of keeping track of the score (if you're stuck, decompile one of the phistball maps (HINT: they start with pb_ instead of dys_) and check out how they do it), and while easy, that's part of the fun. And if you REALLY get stuck, decompile either ctf_figure8 or ctf_face2face. As the designer of the CTF system on both of those maps, I give you permission.

Map paths

Console Commands

- **mpath_show** -- shows existing map paths for your team in developing mode
- **mpath_showall** -- shows all map paths, not just your team
- **mpath_hide** -- The opposite of mpath_show; make the debug view go away
- **mpath_auto <0/1>** -- enables auto mode, while moving you'll place a pathnode every 200 units
- **mpath_create** -- creates a path node at your current position
- **mpath_jumpoff_node** <nodeid> -- will set the parent node for to be used in mpath_auto mode
- **mpath_calculate_active_nodes** -- recalculate node network
- **mpath_move <nodeid>** -- moves a node to your current position
- **mpath_report** -- spams your console with a textual summary of all the nodes in the map
- **mpath_set_obj <node> <obj>** -- set an objective for the given node
- **mpath_save <mapname>** -- save path network to a file
- **mpath_load <mapname>** -- load path network from the given file
- **mpath_set_subobj <id>** -- If you set a node to be controlled by a subobjective, it won't allow the flow of "active juice" unless its team controls that subobjective. Setting a subobjective like this does NOT cause the node to be a "source" of active juice, it just makes it block when the subobjective is not held

HowTo

Map paths are done by using console commands while ingame; no entities required! You don't need hammer at all, or any other tool besides a running copy of the game.

To get an overall idea how those work you could load up any official map and type **mpath_show** in console to see what's behind the arrows you normally see in game.

Map paths are described by a network of nodes. Each node has a position in space, up to two children, and a few other variables which I will cover later. You can create a node at your current position with the mpath_create command. A node's children are the nodes that it connects to. It's like a daisychain of nodes, where 1 connects to 2, which connects to 3, and so on. You can set a node's child with mpath_set_child <parent> <child>. For example, to make node 45 be the child of node 102, you would type "mpath_set_child 102 45". A node can also have a second child, so you can make branches in the path. To do this, use mpath_set_child2. A node can only have two children, so if you want to have a three-way-branch you have to use multiple nodes (like 1 -> [2,3] 2 -> [4,5]) Merging branches together is easy; just make one node be the child of multiple other nodes (and there's no limit on how many can merge together in one place)

An easier way to place a bunch of nodes than using mpath_create over and over is to use mpath_auto 1. It makes you plop down a node every 200 units, just by walking around, and it strings them together for you. You have to use the variable mpath_jumpoff_node to control where your starting point is, though. The first node that gets created will be the child of the jumpoff node. You often have to manually create the very first node in a particular path, set it as the jumpoff node, and then turn on mpath_auto. This is kind of annoying; sorry.

So, there are a lot of different paths that can be defined in a map: there are different paths for each team on each objective. We have to be able to define when they are seen. Team is easy: whichever team you are when you're creating a path, that's what team that path is for. Objectives are more complicated. What you do is set an objective for a particular node with the mpath_set_obj <node> <obj> command. This has two purposes: the first is that it makes a node "active" during that objective. The "active juice" flows to all its children, and to its children's children, and so on forever. So they'll all be active during that objective. The second purpose is that an objective node can block "active juice" from flowing, if it doesn't match the current objective.

The nodes with green on top of them are currently-active nodes. It's currently objective 1. The nodes at the bottom of the screen are active, but the obj2 and obj3 nodes are NOT active. They are blocking the active juice because it's obj 1 right now. That obj1 node in the upper right is active though. In fact, even if the bottom nodes WEREN'T active, that obj1 node would be active anyway.

This complexity with blocking wouldn't be needed if we just made a *completely new* set of paths for every objective, but that would be a pain. We want to be able to reuse certain sections of path on multiple objectives, but not other sections, and this system is flexible enough to let us do that. Check out the paths on vaccine and you should be able to see how it works.

After you change around nodes, it won't update which ones are active until you tell it to, with the mpath_calculate_active_nodes command.

You can't delete nodes! If you find yourself wanting to, just reuse that node somewhere else. Or make it sit there alone with nothing pointing to it. (if you go into the txt file and try to delete a node that way, you will probably break it)

You can save and load paths with mpath_save <mapname> and mpath_load <mapname>. For example, "mpath_load dys_vaccine". Don't put .txt at the end. They're text files in resource/mappaths/ When you're working on these, be sure to save a lot because there's no autosave and it sucks when you get a crash! There's also no undo so be careful.

Sometimes you need to have tightly-packed nodes in order to get the logic right for what is active during what objective. This can make the particles turn into an ugly blob at that spot, so you can set a node as "hidden" with mpath_set_hidden <id>. A hidden node will still propagate active juice, but it won't draw any particles itself.

You will probably want to bind "mpath_show" to a key; I have it on mouse4. You have to press it a lot to see the names of the nodes.

Now hopefully you know HOW to make paths. Here are some higher-level issues about what paths to make.

Paths for the attacking team are easy: send them to the objective. But remember, when it switches to objective 2, there are people in the objective 1 area. You can't just have obj2 paths consist of spawn -> obj2. Those people hanging around in the obj1 area need a path to obj2. Same for 3, 4, however high it goes. Anyone in a "previous" part of the map needs a path to the current objective. (if you were wondering why there were obj1, obj2, and obj3 nodes right at the beginning of vaccine, this is why)

Paths for the defending team are tricky. Where should you send them? To the objective? To the enemy spawn? Don't give them any paths at all? I think there are different objectives where all three of these are the right answer. It depends on the specifics of the objective. We should probably send them to a "good" place. Think to yourself, "where is the most helpful place for a newbie to mill around" and send them there. Also, if there are any side routes that go to unhelpful places, it's probably a good idea to have a path going AWAY from that route towards a helpful area. For example: on silo 1 defense, you can go to the power jip. If a newbie wanders down there, he is in the Wrong Place and he is being Useless. There should be a path sending him back up the ramp to a useful area. There's a turret near there, and it is often hostile! We probably don't want to send newbies into turrets unless we absolutely have to (like on vacc 2 attack). So that path for the defenders should probably end on that little bridge.

I think it's a bad idea to send defenders towards an objective jip, because they will deck in uselessly. In fact, we probably shouldn't send defenders towards any jip, ever, since newbies decking is so disruptive to teams. Once a player is comfortable enough that they don't rely on mappath, then they can get into decking.

Alternate routes are a little tricky, too. If an alternate route is complex or dangerous, it might be best to not even give it a path. The important thing about paths isnt that they show all the options, but that they give players saying "where the hell do I go" a good answer. Go THAT way.

Map testing errors

SUPPORT

This article has been marked as needing scrutiny. Please see the talk page for details.

Recently Dystopia mappers have had difficulties in testing their maps. Most of the time, it hasn't been a problem with the map, but a problem with the Source Engine. Here's a multi-tiered fix to ensure that you will be able to test your map.

Contents [hide]

Verify integrity of tool cache

1. Open Steam
2. Click "Tools"
3. Right-click on "Source SDK"
4. Click "Properties"
5. Click the "Local Files" tab
6. Click "Verify integrity of tool cache"
7. Wait patiently for it to finish

Defragment cache files

1. Open Steam
2. Click "Tools"
3. Right-click on "Source SDK"
4. Click "Properties"
5. Click the "Local Files" tab
6. Click "Defragment cache files"
7. Wait patiently for it to finish

Copy Steam Client DLL

1. Open Steam
2. Click "Tools"
3. Make sure you have "Source Dedicated Server" downloaded and installed
4. If you don't, double-click on the grayed-out "Source Dedicated Server" and wait patiently for it to download before continuing
5. Browse your computer until you find your local Steam folder (usually C:\Program Files\Steam\Steam)
6. Open "SteamApps"
7. Open your Steam username's folder
8. Open "Steam Dedicated Server"
9. Open "bin"
10. Find "steamclient.dll" and press CTRL+C
11. Navigate back to your username's folder
12. Open "Source SDK Base"
13. Open "bin"
14. Look to see if there's already a "steamclient.dll" here.
    1. If there is, rename it to steamclient.dll.bak"
15. Press CTRL+V

Make desktop shortcut

1. Go to your desktop
2. Right-click on the desktop
3. Click New -> Shortcut
4. Input the following as the location of your file:

```
"C:\Program Files\Steam\Steam.exe" -applaunch 215 -game "c:\program files\steam\SteamApps\SourceMods\dystopia_v1"
```

   with quotes, substituting the paths for your local Steam paths)
5. Click "Next"
6. Type in "dystopia" or something
7. Click "Finish"
    1. If you're like Adam and you like to WindowsKey+R to do everything, you can copy this shortcut and paste it in your %SystemRoot% directory (usually C:\WINDOWS). If you name it something like "dystopia," you can press WindowsKey+R, type in "dystopia," and test your maps instantaneously

Run your maps in LAN mode

1. Use the shortcut we made
2. Press ~ to open the console
3. Type "sv_lan 1" (without the quotes)
4. Close the console
5. Click "Create Server"
6. Choose your map
7. Click "Start"

Categories: Support Pages | Articles Needing Scrutiny | Mapping

navigation
- Main Page
- Community portal
- Current events
- Recent changes
- Random page
- Help

search
Go Search

toolbox
- What links here
- Related changes
- Special pages
- Printable version
- Permanent link

Mapping Measurements

SUPPORT

Vents

- Light-only, crouch- 48 units square, clipped down to 44 units tall
- Light and medium, crouch- 56 units square, clipped down to 47 units tall
- Light, medium, heavy, crouch- 56 units square
- Light can stand, medium and heavy crouch- 64 units square, clipped down to 61 units tall
- Light and medium can stand, heavy crouch- 72 units square, clipped down to 64 units tall
- Light, medium, heavy stand- 80 units square

Mapping related measurements

Height

Room Heights

A good height to make rooms is 160 or 192 units. It's high enough for bunny hopping and doors but keeps a respectable profile.

Elevation

Tallest walk-on-to-able object is 18 inches/units.

Minimum height to grab without augmentation is 112 inches/units (heavy's can't grab).

Jumping height

Maximum height for a jump-crouch light/medium is 65 inches/units.
Maximum height for a jump-crouch heavy is 68 inches/units.

Maximum height for a normal jump is 48 inches/units.
Maximum height for a couch jump is 48 inches/units.

Maximum height for an augmented normal jump heavy is 76 inches/units.
Maximum height for an augmented normal jump medium is 130 inches/units.
Maximum height for an augmented normal jump light is 184 inches/units.

Maximum height for an augmented jump-crouch heavy is 96 inches/units.
Maximum height for an augmented jump-crouch medium is 147 inches/units.
Maximum height for an augmented jump-crouch light is 202 inches/units.

Maximum height for an augmented jump-grab medium is 194 inches/units.
Maximum height for an augmented jump-grab light is 249 inches/units.

Jumping Distance

Maximum length for an unaugmented jump of even height for a light is ~200 inches/units.
Maximum length for an augmented jump of even height for a light is ~400 inches/units.
Maximum length for an augmented running jump of even height for a light is ~500 inches/units.

Maximum length for an unaugmented jump of even height for a medium is just short of 200 inches/units.
Maximum length for an augmented jump of even height for a medium is just short of 300 inches/units.
Maximum length for an augmented running jump of even height for a medium is just short of 350 inches/units.

Maximum length for an unaugmented jump of even height for a heavy is just short of 150 inches/units.
Maximum length for an augmented jump of even height for a heavy is just short of 200 inches/units.
Maximum length for an augmented running jump of even height for a heavy is just over 200 inches/units.

Maximum length for an unaugmented jump of height of 96 to 0 for a heavy is just short of 200 inches/units.
Maximum length for an augmented jump of height of 96 to 0 for a heavy is just over 200 inches/units.
Maximum length for an augmented running jump of height of 96 to 0 for a heavy is just short of 300 inches/units.

Maximum length for an unaugmented jump of height 194 to 0 for a medium is short of 250 (but over 200) inches/units.
Maximum length for an augmented jump of height 194 to 0 for a medium is ~300 inches/units. Maximum length for an augmented running jump of height 194 to 0 for a medium is ~400 inches/units.

Maximum length for an unaugmented jump of height 249 to 0 for a light is ~300 inches/units (taking damage).
Maximum length for an augmented jump of height 249 to 0 for a light is ~500 inches/units.
Maximum length for an augmented running jump of height 249 to 0 for a light is ~600 inches/units.

Player

Crouching

Minimum clearance for light while crouching is 45 inches/units.
Minimum clearance for medium while crouching is 48 inches/units.
Minimum clearance for heavy while crouching is 53 inches/units.

Standing

Minimum clearance for light while standing is 62 inches/units.
Minimum clearance for medium while standing is 65 inches/units.
Minimum clearance for heavy while standing is 73 inches/units.

- the mininum is 1 unit more then the size itself, eg a standing light is 61inches/units

Extra info

Minimum width to walk through is 33 inches/units.

Player size in cyberspace = 64x64x64

TAC Scan range = 2048 3d units

Move Speed

Light

Run speed = 250

Sprint speed = 312.5

Crouched speed = 125

Medium

Run speed = 200

Sprint speed = 250

Crouched speed = 100

Heavy

Run speed = 160

Sprint speed = 200

Crouched speed = 80

Run speed with minigun spinning = 96

Crouched speed with minigun spinning = 7.27

Overviews

SUPPORT

View Valve's Original Article

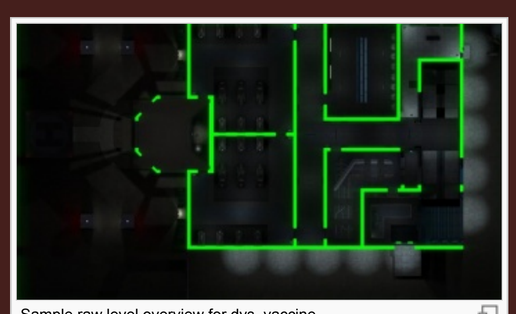
This will describe to you how to make the new V1 Level OverView (the radar, mini-map) The process is broken down to three main steps, capturing, editing, implementing.

Make the raw overview image

1. Start the game. Click **Options** and switch to the **Video** tab. Change **Resolution** to "1280x1024" pixels and the **Aspect Ratio** to "Normal".
2. Load the new level with the map `<mapname>` console command and enable `"sv_cheats 1"`.
3. Bind the screenshot command to a key, etc. `"bind p screenshot"` using the console. This allows you to get an uncompressed TGA screenshot, rather than a JPG with compression marks.
4. Remove any HUD elements with console commands `"cl_drawhud 0"`.
5. Remove disturbing world effects `"r_skybox 0; fog_override 1; fog_enable 0; r_drawstaticprops 0"`
6. Turn on Full Brightness by using `mat_fullbright 1`
7. Move in spectator mode to a position where you can see the whole level from above. For multi level maps, move to the middle area of your "floor"
8. Switch to orthographic mode with the console command "`cl_leveloverview x`", where *x* is the scale factor.
9. Set `"cl_leveloverviewmarker 1024"` to show a red helper line on the right. You will need to briefly toggle `cl_drawhud 1`, line it up, and then toggle the hud off again.

Several people have commented that the red line does not appear when doing this at 1280x1024 resolution. If you don't see the red line, read the No red line section below.

10. You can move around while *cl_leveloverview* mode is on. You are still in spectator mode and moving, so move the arrow keys to change the position. Now change scale so the map fits into the square defined by the red line. The right edge of your map should be on the left of the red line, anything to the right of the red line will be chopped off in a later step. Here is a screenshot how it would look for the map `dys_vaccine`



Sample raw level overview for dys_vaccine

11. The current projection data (position and scale) will be updated in the console, eg "Overview: `scale 6.00, pos_x -2651, pos_y 4027`"
12. Once everything is perfect, take a screenshot (using the key you bound to "screenshot") and **write down** the projection data values. You will need them in a later step.
13. Switch back to normal view `cl_leveloverview 0"` and move to the next "floor" in your map. (If your map is only a single floor, then ignore this, if your map has multiple floors (areas ontop of areas, such as dys_vaccine, or dys_silo, then take a screenshot so that your upper areas are visible)

No red line?

If you don't see the red line appear at step 8, make sure you have set `cl_drawhud 1` (turn this off again before taking the screenshot though). After this, if you still can't see the line, you may want to try starting over but use 1024x768 resolution instead of 1280x1024. Set `"cl_leveloverviewmarker 768"`.

Edit the overview image

1. Close Dystopia and open the screenshot(s) in an image editor. Screenshots are found in the `"(Steam directory)\(username)\dystopia\Dystopia\screenshots"` directory. Using the image editor, remove the unused border right of the red line so the image size is reduced to 1024x1024 pixels. Look for any option to reduce the "canvas size", you do not want to resize the image, but rather crop off the right side so you're left with a perfect square 1024x1024 or 768x768. It MUST be square and its dimensions must be a power of two, or Vtex will give an error.
2. If you had an issue viewing the red line and used the "1024x768" mode, make sure the image size is a power of two. Resize your image to 512x512 or 1024x1024 or Vtex will fail.
3. Download Overview Template This is a template used to re-create the v1 overviews used in the official maps. Take out all of the background material

--Add steps about border and alpha transparency. --

4. Save it as a TGA image with the same name as your .bsp file, in your material source folder (eg. `"SourceMods\Dystopia_v1\materialsrc\overviews\mapname.tga"`).
5. If you are using Photoshop, you can instead download the "VTF-Plugin for Photoshop". You then select "Save as" from the menu, change the file type to *.vtf* and give it the exact same name as your map. Click *OK* and a box will pop-up. Select "Clamp S", "Clamp T", "No Compression" and "Eight Bit Alpha", and then select *OK*. Be sure to select where available No-MipMaps, this will reduce the size of your overview greatly.

Create the material files

1. Convert your image into a vtf using Vtex. (See Vtex for step-by-step instructions on how.) The texture file should end up as `SourceMods\Dystopia_v1\materials\overviews\mapname.vtf`.
2. In the same folder as this texture file, using a text editor, add a new VMT file called *mapname.vmt*, which should look like this:

```
"UnlitGeneric"
{
        "$translucent" "1"
        "$basetexture" "overviews/mapname"
        "$vertexalpha" "1"
        "$no_fullbright" "1"
        "$ignorez" "1"
}
```

3. Finally, in a text editor, create a text file called *mapname.txt* in `\resource\overviews\` (eg. `SourceMods\Dystopia_v1\resource\overviews\dys_vaccine.txt`) that contains the necessary projection data:

```
"mapname"
{
        "material"      "overviews/mapname"     // texture file
        "pos_x"         "-2651" // X coordinate,
        "pos_y"         "4027"  // Y coordinate,
        "scale"         "6.0"   // and used scale used when taking the screenshot
        "rotate"        "0"     // map was rotated by 90 degress in image editor
        "zoom"          "1.3"   // optimal zoom factor if map is shown in full size
}
```

Multi-Level OverViews

Now that you have your overviews you need to "stack" them. The order is very important as it starts with the TOP FLOOR FIRST

```
"mapname"
{
        "floorname" //This can be anything, but should be there
        {
                "material"      "overviews/dys_map_topfloor"
                "pos_x"         "-1337"
                "pos_y"         "31337"
                "pos_z"         "100"   // Minimum Altitude to be in area
                "scale"         "6.0"
                "rotate"        "0"
                "zoom"          "1.3"
        }
        "floorname2"
        {
                "material"      "overviews/dys_map_basement"
                "pos_x"         "-1337"
                "pos_y"         "31337"
                "pos_z"         "-200"  // Minimum Altitude to be in area
                "scale"         "6.0"
                "rotate"        "0"
                "zoom"          "1.3"
        }
}
```

See also

Retrieved from "http://www.dystopia-game.com/wiki/index.php?title=Overviews"

Categories: Support Pages | Mapping

navigation
- Main Page
- Community portal
- Current events
- Recent changes
- Random page
- Help

search
Go Search

toolbox
- What links here
- Related changes
- Special pages
- Printable version
- Permanent link

Phistball Mapping

SUPPORT

This Section will tell you what entities you need to make the Phistball game type.

Contents [hide]

What you need

Before you begin make out of brushwork the following:

1) Arena - Simple brushed based area, where the ball is not going to be stuck.

2) 2 Goals - Make them as pretty as you like, does not really matter what shape or size

3) Spawn points - Make 2 separate spawn rooms for each team and somehow make it link to the arena.

Phistball Construction

Making The Ball

Using a phist_ball entity (same as prop_physics), you can use the default phistball ball located in: *models/items/phistball_01.mdl* Or you can customise and provide your own ball. Don't forget to name the ball something rememberable as we will need this later. You may also want to adjust the mass scale parameter of the physics ball, its decimal, so 0.5 will be 50% of the normal mass of the ball, and thus will move a lot easier.

Making The Goals

Edit V1

Making The Score System

Edit V1

Making Arena Phistball friendly & fun

Gameplay is paramount in a phistball map, you need you map to be fun, exciting and interesting to attract players, how you do this is up to you, but here are some useful tips:

We also do not want the ball to get stuck, or fall off the map, here are a few neat tricks to help you. Make sure there are no 90 Degree corners as the ball is almost certain to get stuck. If you do do a 90 Degree corner use meathods in the next section to prevent it from getting stuck

You want the spawn points away from the actual arena of play, use teleporters and other clever ways to get players into the arena, but not them escape out.

Don't have anything to distract players from the game, if your map is good enough players wont need toys to keep them entertained. This also includes secret areas etc.

Please don't make your map full bright, or I will hate you forever, you spent enough time creating the map, don't let that go to waste and spend some time adding a lighting, you can even do it with 1 entity (light_env), its not hard.

This comes as obvious, but test your map thoroughly, think outside the box and try and exploit your map in every way possible, as bored players will!

Usefull brush entitys

func_vclip_physics - Apply this entity to a trigger brush and this will stop all physics objects from passing through it.

trigger_multiple - This is what will make your phistball arena powerful, you can setup systems related to the balls position, when a ball enters this brush it will fire as many outputs as you want, so you could reset the ball, fire the ball out of a pinball hole etc.

trigger_push - you can use this to push balls in any direction you want

trigger_teleport - you can use this to teleport players or balls to any given destination,

For information on how to use these brush entitys please vist the valve Developer comunity wiki [1]

Melee Only & Instant Respawn

Melee Only, can be done using a logic auto and a point_servercommand. Use the logic auto outputs to fire the command "dys_meleeonly 1" to the point_servercommand.

Instant Respawn, This can be done using a logic_timer, and make the timer start enabled with a delay of 1 second between the triggers, then use the logic_timer outputs to forcespawn the dys_spawn entity's (1 for the punks and 1 for the corps spawn)

(information has not been verified and the content may be invalid)

Written by Icepick66

Categories: Support Pages | Mapping

navigation
- Main Page
- Community portal
- Current events
- Recent changes
- Random page
- Help

search
Go Search

toolbox
- What links here
- Related changes
- Special pages
- Printable version
- Permanent link

RES Structure Guide

A Sample RES File

```
"radioshack_serverDefence.res"
{
   "Background"
   {
      "ControlName"     "MaterialImage"
      "fieldName"               "Background"
      "xpos"                    "0"
      "ypos"                    "0"
      "zpos"                    "-2"
      "wide"                    "256"
      "tall"                    "256"

      "material"                "dys_monitor1a"
    }

    "TitleLabel"
    {
       "ControlName"    "Label"
       "fieldName"              "TitleLabel"
       "xpos"                   "32"
       "ypos"                   "16"
       "wide"                   "192"
       "tall"                   "16"
       "autoResize"     "0"
       "pinCorner"              "0"
       "visible"                "0"
       "enabled"                "1"
       "tabPosition"    "0"
       "labelText"              "Server Defences Control"
       "textAlignment"  "center"
       "dulltext"               "0"
       "brighttext"     "0"
       "wrap"                   "0"
    }

    "DisableDefence"
    {
       "ControlName"   "MaterialButton"
       "fieldName"             "DisableTurrets"
       "xpos"                  "32"
       "ypos"                  "96"
       "wide"                  "192"
       "tall"                  "64"
       "autoResize"    "0"
       "pinCorner"             "0"
       "visible"               "1"
       "enabled"               "1"
       "tabPosition"   "2"
       "labelText"             "Toggle Defences"
       "textAlignment" "center"
       "dulltext"              "0"
       "brighttext"    "0"
       "Default"               "0"
       "command"               "button1"
       "paintborder"   "0"
       "font"                  "SpaceOneSmall"

       "enabledImage"
       {
          "material"   "vgui/screens/vgui_button_enabled"
          "color"      "255 255 255 255"
       }

       "mouseOverImage"
       {
          "material"   "vgui/screens/vgui_button_hover"
          "color"      "255 255 255 255"
       }

       "pressedImage"
       {
          "material"   "vgui/screens/vgui_button_pushed"
          "color"      "255 255 255 255"
       }

       "disabledImage"
       {
          "material"   "vgui/screens/vgui_button_disabled"
          "color"      "255 255 255 255"
       }
    }
 }
```

Overall Structure Explained

1	"radioshack_serverDefence.res"	This is the name of the RES file.
2	{	
3	"Background"	This defines what the background of the screen will be (the camera image)
4	{	
5-13	...	
14	}	
15	"Title Label "	The title of the screen.
16	{	
17-32	...	
33	}	
34	"DisableDefences"	A button on the screen (you can have more than one by just adding more blocks of code like this one)
35	{	
36-54	...	
55	"enabledImage"	Ignore this, you dont need to change it.
56	{	
57-58	...	
59	}	
60	"mouseOverImage"	Ignore this, you dont need to change it.
61	{	
62-63	...	
64	}	
65	"pressedImage"	Ignore this, you dont need to change it.
66	{	
67-68	...	
69	}	
70	"disabledImage"	Ignore this, you dont need to change it.
71	{	
72-73	...	
74	}	
75	}	
76	}	

Screen Setup

SUPPORT

This article has been marked as new, and may contain grammatical errors.

Contents [hide]

Introduction

The cool looking interactive screens we have in dystopia consist of 3 parts:

1. The entity in the map (dys_screen or dys_cyberscreen)
2. The _screens.txt file (located in your /maps directory)
3. The .res files (located in your /scripts/screens directory)

Screens come in two flavours, dys_screen and dys_cyberscreen. The difference being that dys_screens are just basic screens in the real world while cyberscreens are used in cyberspace (ie. can be password protected and have ice linked to them).

Components needed for working screens

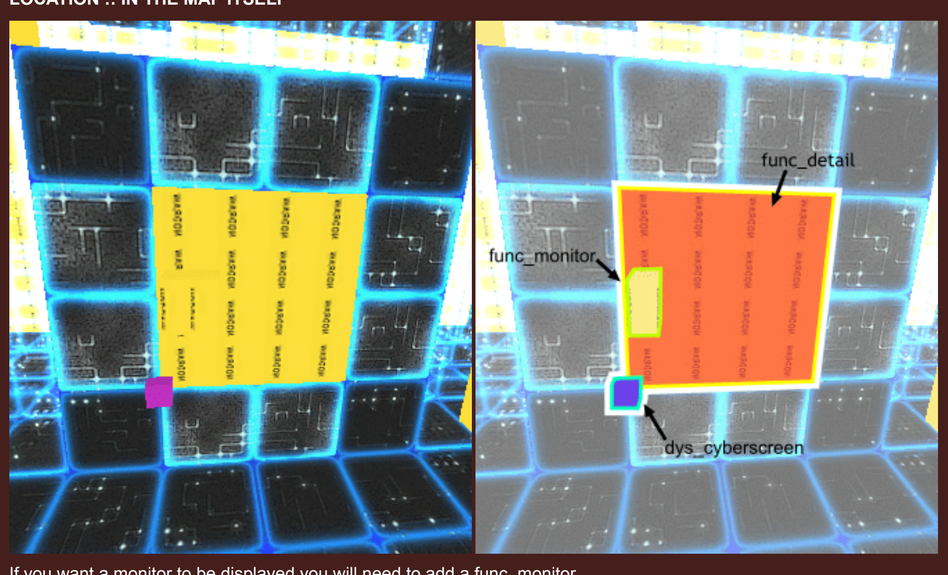
NOTE : IF YOU FAIL TO SET UP ANY OF THESE COMPONENTS CORRECTLY, YOUR SCREENS WILL NOT WORK/SHOW UP IN GAME.

The Entity in the map

When adding a screen you will need to determine what kind it will be. If it is in meatspace, you want a dys_screen. If it is in cyberspace, and it can be passworded/encrypted/ICEd, you want a dys_cyberscreen. You can also put dys_screens in cyberspace, if you don't want it to be passworded/encrypted/ICEd.

1. Add a point_camera where you want your screen; it is much easier to rotate a point_camera than a screen entity. The screen will render in game using the center of the entity as the bottom left corner of the screen. So you will need to place the screen in the bottom left part of where you want it to display. Then rotate the point_camera so that the x-axis (green line in 3D view) and y-axis (blue line in 3D view) both lead to the right and upwards of the screen, respectively.
2. Change the point_camera to the desired screen entity (dys_screen or dys_cyberscreen); this preserved the rotation data from the camera, which ensures the screen will be oriented correctly in-game. (If the rotation was not preserved, just set the Angles property in the Object Properties and set it to rotate to face how ever you want. This renders the whole point_camera method redundant.)
3. Give the screen a name under *Panel Name*. This name will be used for the .res file and the .txt file.
4. Adjust the Panel width and Panel height in the entity to fit your screen. This may take some trial and error to get right. (It is helpful to create a brush and fit it to the desired area as a visual measure, as you can simply copy over the dimensions to Panel width and Panel height and delete the brush when done).

LOCATION :: IN THE MAP ITSELF



If you want a monitor to be displayed you will need to add a func_monitor.

The _screens.txt

Known as the 'screen declaration file' its a basic .txt file which declares all the screens you use in your map, and must follow the naming convention <mapname>_screens.txt (so if your map was dys_vaccine.bsp the file would be dys_vaccine_screens.txt)

LOCATION :: Add the .txt file to your /dystopia/maps/ directory, along with your .bsp

Tip: Basically, If the screen appears in the real world (meatspace), it is a dys_screen and must be declared as such, if its a screen that appears in cyberspace, its a dys_cyberscreen. screens cannot be declared as both screen and cyberscreen. The easiest way to work out how the declaration files work is check out the dys_vaccine_screens.txt in the /maps/ directory.

You will need to create a .txt file using the previously stated naming convention and add the following.

```
"VGUI_Screens"
{

}
```

In between the brackets you will need to add the name of all of your screens using the following syntax. For dys_screen's you will to add the Panel Name of the dys_screen and declare it as a dys_screen. If you have a dys_screen and the Panel Name was **MyMap_Turrets**. Then you would need to add

```
"MyMap_Turrets"
{
        "type" "dys_screen"
}
```

To add a dys_cyberscreen you simply need to change the "type" from "dys_screen" to "dys_cyberscreen." Add a dys_cyberscreen with a Panel Name of **MyMap_CyberTurrets** and the full .txt file would look like this.

```
"VGUI_Screens"
{
    "MyMap_Turrets"
        {
        "type" "dys_screen"
        }
    "MyMap_CyberTurrets"
        {
        "type" "dys_cyberscreen"
        }
}
```

The .res files

Each screen is drawn from a .res file which uses basic vgui scripting to compose what is rendered on the screen in-game. We recommend for ease of use that you prefix your screens with your map name, and dont re-use the .res files from other maps in your own, as if we change the screen, or another mapper changes their screen, it may break its functionality on your map.

The .res file is where you choose how many buttons you want, the size of each button, the text displayed on the button(s), and the material for the monitor(if you choose to use one).

LOCATION :: /dystopia/scripts/screens/

Tip: check out the other .res files in there, they are named fairly obviously, and are quite simple, if you can understand html files, you should get the gist of these. Also, check out the RES Structure Guide for a more detailed explanation.

Packaging Your Map

Your maps must be packaged up with **both** the _screens file and the res files. otherwise the screens wont show up.

If you are still confused, check out the template map

Camera Usage In relation to screens

When making your screen you may want the background or a part of your screen to contain a camera,

Simply put, the 'render target' of the point_camera acting as the source of the image should have 1 of 4 settings

- camera1
- camera2
- camera3
- camera4

note that camera4 should only be used in cyberspace as it is also the screen used by heavies on the rocket launcher

Now depending on which target you use, you need an applicable texture on your screen in the .res file

ie.

- camera1 will use
  - dys_monitor1a
  - dys_bwmonitor1
- camera2 will use
  - dys_monitor2a
  - dys_bwmonitor2
- camera3 will use
  - dys_monitor3a
  - dys_bwmonitor3

and so on....

The func_monitor should be textured with toolsnodraw, and placed close to the screen entity. The screen will take the camera name from the func_monitor when deciding which point_camera to render the view from. Note that this goes against the normal use of func_monitor, and is a special case for dealing with screens.

Restrictions

1. cameras are only updated when within line of sight and 512 units away (1024 units in cyberspace)
2. cameras can be re-used as long as they cant be seen from each other, so you can have up to 4 different screens in 1 room, then another 4 (using the same textures and camera targets on a different set of cameras and screens) in another room etc. which allows for alot of flexibility in your maps.
3. camera4 should only be used in cyberspace as it is used in meatspace by heavies for their rocket launcher cam; it can be used in meatspace if it is in a place heavies can't reach/see, like small vents, but try to use textures 1-3 first.

Setting Hammer Up for Dystopia

(The bottom of this article has instructions for directly embedding an entry into **GameConfig.txt**).

> **Note:** The Hammer configuration window accepts `$SteamUserDir` as a substitute for the file path of your Steam user name's directory, and it is used throughout this article for the same purpose.

Before You Start

If you've never set hammer up manually before, you should probably read the VDC's article on settting Hammer up manually first. If you get confused at any point in this article, refer to that article for additional help.

The Source SDK

Hammer is accessed through a tool-set called the Source SDK. Anyone who owns a Source game has free access to it. To start it, open Steam, and click the **Tools** tab near the top. In the resulting list is the SourceSDK; launch it (if it is grayed out, you will have to wait for it to download first).

A nifty little routing program opens up. All the important tools of the SDK can be accessed here, but before you can use any of them correctly with Dystopia, you need to set up the Game Configurations through Hammer.

Since the Orange Box was released, there are actually two variations of most programs in the SDK; one set for the Episode One version of Source, and another set for the new Orange Box Source. Dystopia currently runs on the Orange Box Source engine, so in the **Engine Version** drop-down menu, select "Source Engine 2007".

> **Note:** If you do not own a Valve game that uses the Orange Box Source version of Source, you will not be able to select "Source Engine 2007" from the drop-down menu. Follow the instructions at the bottom of this article to configure Hammer a different way.

Dystopia should not be a **Current Game** yet, so just launch Hammer by double clicking "Hammer Editor" (see the reference image at right).

Steam and the SourceSDK.

Game Configuration

The SourceSDK can be set up to edit any Source-based game or mod, but all the configuring for this is done through the Hammer World Editor. Once the editor loads, open the "Tools" menu in the top left and then click on "Options". A window titled **Configure Hammer** boldly pops up to announce success.

This window is the business end of SourceSDK configuration (don't let its out-dated title fool you). It should display the **Game Configurations** tab by default; if not, open it. Then, start to create a new configuration for Dystopia by clicking on the "Edit" button to the right of the **Configuration** field. Another window should pop up, titled **Edit Game Configurations**. Click on the "Add" button and type in `Dystopia` in the text-box. Hit "OK", and then close the **Game Configurations** window.

Now, from the **Configuration** drop-down menu, select "Dystopia". And voila, a mostly-blank configuration should be shown.

To the right of the **Game Data files** field is the "Add" button; click it. Enter in `$SteamUserDir\dystopia\dystopia\dystopia.fgd` and click "OK". The new file path should show up in the **Game Data files** field. Then, put these required values into their specified fields:

Texture Format
"Materials (Half-Life 2)"

Map Type
"Half-Life 2"

Game Executable Directory:
`$SteamUserDir\dystopia`

Game Directory:
`$SteamUserDir\dystopia\Dystopia`

Hammer VMF Directory:
`$SteamUserDir\dystopia\Dystopia\mapsrc`

The other fields are optional, and you can use the default settings. But here is what Dystopia mappers reccomend:

Default PointEntity class
"light"

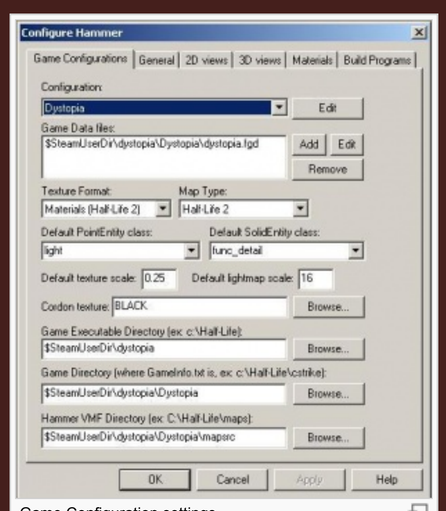
Default SolidEntity class
"func_detail"

Default texture scale
`0.25`

Default lightmap scale
`16`

Cordon texture
`tools/toolsblack`

See the reference image at the right for what the finished values should be.



Game Configuration settings.

Build Programs Tab

Next, click on the **Build Programs** tab, all the way to the right at the top of the **Configure Hammer** window. This tells Hammer what programs to use when compiling your map, which needs to be done before the map can be played. Select "Dystopia" from the **Configuration** drop-down menu and enter the following required paths in their respective fields:

Game executable
`$SteamUserDir\dystopia\hl2.exe`

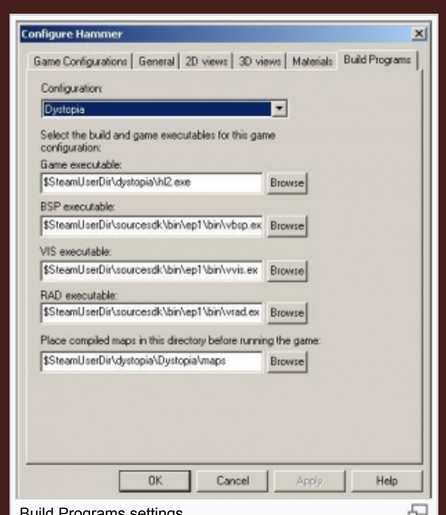
BSP executable
`$SteamUserDir\sourcesdk\bin\source2007\bin\vbsp.exe`

VIS executable
`$SteamUserDir\sourcesdk\bin\source2007\bin\vvis.exe`

RAD executable
`$SteamUserDir\sourcesdk\bin\source2007\bin\vrad.exe`

Place compiled maps in this directory before running the game
`$SteamUserDir\dystopia\Dystopia\maps`

Again, look at the reference image at right to compare the final result. There are some third party compile tools on the internet that you can download and use in place of Valve's standard ones, and if you have any, change the above fields to make Hammer use them. Third party tools are *not* required, some people just prefer them over the default ones.



Build Programs settings.

> **Note:** The reference image at the right is outdated and should be updated to match the instructions above. Simply replace 'ep1' with 'source2007'.

Directly Embedding the Configuration

As an alternative to the above process, you can also paste the text below into a file named **GameConfig.txt**, with a file path of `$SteamUserDir\sourcesdk\bin\source2007\bin\GameConfig.txt` (if it does not exist, create it first).

> **Note:** If you put a trailing slash (`/`) in the `"GameDir"` field, *THE WORLD WILL IMPLODE AND HAMMER WILL FAIL AT ALMOST EVERYTHING*, so don't put a trailing slash in the `"GameDir"` field.

```
"Configs"
{
	"Games"
	{
		"Dystopia"
		{
			"GameDir"	"<your_steam_user_directory>\dystopia\dystopia"
			"hammer"
			{
				"GameData0"	"<your_steam_user_directory>\dystopia\dystopia\dystopia.fgd"
				"TextureFormat"	"5"
				"MapFormat"	"4"
				"DefaultTextureScale"	"0.250000"
				"DefaultLightmapScale"	"16"
				"GameExe"	"<your_steam_user_directory>\dystopia\hl2.exe"
				"DefaultSolidEntity"	"func_detail"
				"DefaultPointEntity"	"light"
				"BSP"	"<your_steam_user_directory>\sourcesdk\bin\source2007\bin\vbsp.exe"
				"Vis"	"<your_steam_user_directory>\sourcesdk\bin\source2007\bin\vvis.exe"
				"Light"	"<your_steam_user_directory>\sourcesdk\bin\source2007\bin\vrad.exe"
				"GameExeDir"	"<your_steam_user_directory>\dystopia"
				"MapDir"	"<your_steam_user_directory>\dystopia\dystopia\mapsrc"
				"BSPDir"	"<your_steam_user_directory>\dystopia\dystopia\maps"
				"CordonTexture"	"tools\toolsblack"
				"MaterialExcludeCount"	"0"
			}
		}
	}
	"SDKVersion"	"3"
}
```

You may already have several games declared under the `"Games"` field. If this is the case, simply copy the block under the `"Dystopia"` field. You must manually replace `<your_steam_user_directory>` with the file path of your Steam user name's directory. Once `<your_steam_user_directory>` has been replaced, save the file, and launch Hammer with "Dystopia" as the **Current Game**.

Categories: Support Pages | Updated Game Info | Mapping

navigation
- Main Page
- Community portal
- Current events
- Recent changes
- Random page
- Help

search
Go Search

toolbox
- What links here
- Related changes
- Special pages
- Printable version
- Permanent link

Spawn points

This page gives mappers information on spawn points and the spawn system in Dystopia.

Spawns

Dystopia involves the use of spawn points to spawn players. These points can change control during objectives, changing the flow of the map during play, and allow players to respawn after death. Team spawn based on where they can legally spawn, defined as an Enabled spawn set to their team. When two or more legal spawns are available, Punks will spawn at the highest available SpawnID and Corps will spawn at the lowest available SpawnID. Although you can rely on this when mapping, it is considered good practice to disable all spawns except the ones in current use.

Components of Spawn Points

Functional spawn points in a map consist of two required entities:

1. A dys_spawn, which controls spawning and can receive inputs.

2. One or more dys_spawn_points, which spawns individual players and contains a built-in model.

Spawn points are usually placed in a room that the opposing team can not enter or shoot into and often may contain any mixture of Jack-In-Points, turrets, forcefields, ammo dispensers, and other useful objects.

Using dys_spawn

A dys_spawn entity controls all the spawn points under its id. This makes for an easy spawn management system, since you only have to disable one entity instead of every individual spawn point.

Key Values

Name: This is used to control the input/output of the entity.

Team: Set the initial team to Punks, Corps, or none.

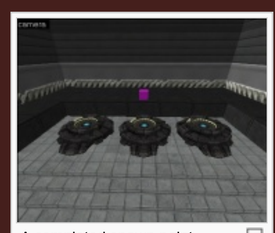
SpawnID: This is what you use to select which spawn points are in its control. Numbered through PunksHQ(1) to 6, allowing you to have up to 6 spawn groups.

Flags

Enabled: Tells the game if players can spawn at that point. It will enable all the spawns in its control. Although this may be overridden to prevent crashing—See below for details.

Using the dys_spawn_points

A dys_spawn_point entity is the actual place where a player may spawn. It contains its own model that can be disabled, if desired.


A completed spawn point

Key Values

Pitch Yaw Roll: Sets the orientation of the spawn pad. This is important since it also effects the orientation of the spawned player.

SpawnID: This will match the spawn point to the spawn controller (a dys_spawn entity).

Flags

Disable Model: This will disable the spawn pad model.

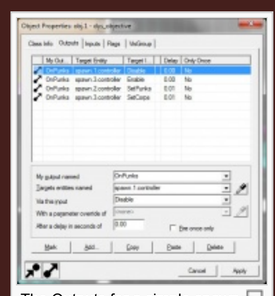
Controlling Spawns

To properly change the spawns for both teams on an event, such as an objective gets captured, the first step is to send an output to the dys_spawn you want to start using and enable it, and send another output to the dys_spawn you no longer want any team to spawn at and disable it. Then send another output to the new dys_spawn with a delay of 0.01 seconds, setting its team value to the team you want to spawn there. And finally send another output to the spawn you want to give to the other team settings its team value to the team taking over. Sound complicated? That's okay, I included a picture for you.

Example

We have three dys_spawn entities: spawn.1.controller, spawn.2.controller, and spawn.3.controller. spawn.1.controller is enabled and set to Punks, spawn.2.controller is enabled and set to Corps, and Spawn.3.controller is Disabled and set to None. On an objective capture, Corps will move to Spawn 3, Punks will move to Spawn 2, and Spawn 1 will become inactive. To accomplish this, first we disable spawn 1 and enable spawn 3 (Since we have to enable it to use it). Then after a brief delay of 0.01 seconds, we actually send the commands to change spawns by telling Spawn 3 it now belongs to Corps and telling spawn 2 it now belongs to Punks. Now you may be asking yourself, "Wait, where do punks spawn for that 0.01 second that their only spawn is disabled?" That's a good question. Punks do not have a legal spawn at this split second, and therefore will spawn at SpawnID 1 (PunkHQ), explained in slightly more detail further down.

Setting the old dys_spawn to Disable is part of good mapping practice, as insurance that the spawn won't be used. Notice how first the new spawn is enabled, then 0.01 seconds later its team value is set. If there is no slight delay then the spawn will not properly change to the spawn area for that team.


The Outputs for a simple spawn change

Additional Information

Before v1.2 was released there was the common occurrence of maps crashing when a player spawns due to bugs in the spawn system. Most of these appear eliminated in 1.2 and thus you do not have to worry about them.

A side effect of bug fixing is that if a team can not find a legal spawn—that is, a spawn that is enabled and set to that team—then that team will spawn at SpawnID 1 (PunkHQ), no matter what its settings are, even if that means two teams spawning in the same room.

Categories: Support Pages | Updated Game Info | Mapping

navigation
- Main Page
- Community portal
- Current events
- Recent changes
- Random page
- Help

search
Go Search

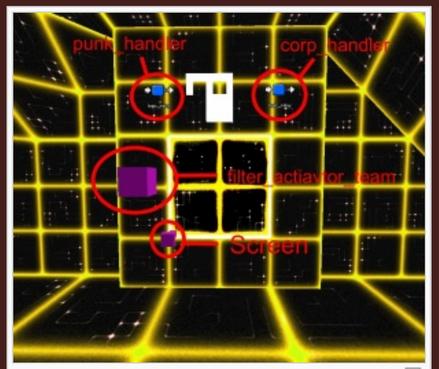
toolbox
- What links here
- Related changes
- Special pages
- Printable version
- Permanent link

Toggle Screens

SUPPORT

This is an article on how to implement "Toggle Screens". These screens are the single button screens that both sides can push; and when that team pushes it, they immediately get its benefit. This makes locating and pushing the button much easier for both teams, preventing the "I was at the screen but pressed the wrong button!", it's the most newb friendly, and it's easy to set up once you know how. So, here are the steps for implementing one:

1. Set up your screen entity (be it a dys_screen or dys_cyberscreen). Refer to the Screen Setup article for more information on this.
2. Make a new point-entity called "filter_activator_team" and give it a name. Put this somewhere near the screen so you can find it when you need it; its actual location has no effect in-game.
3. Clear all the outputs from the screen entity and add a single output, on "Button1", to the filter_activator_team you just created, with the result "TestActivator".
4. Go into the key values for the filter_activator_team, and make sure "Allow only passing criteria" is set to "Yes", and "Team" is set to "Punks".
5. Add two logic_relay entities, somewhere near the screen and filter_activator_team. Affix one's name with "corp_handler" and the other with "punk_handler" (so, for instance, if this is toggling a steam pipe, name one of them "obj1_steam_corp_handler").
6. Go to the filter_activator_team, and add two new outputs; one to the corp handler for "OnFail", and one to the punk handler for "OnPass".
7. Go to both logic_relay handlers, and set up their outputs to any external objects accordingly; when corps press the screen, the outputs from the "corp_handler" will be fired, when punks press it, the "punk_handler".
8. Go to the punk_handler, and add a new output "OnTrigger". Have it go to itself ("*_punk_handler"), and the action "Disable" after a delay of .01 seconds (this ensures all the other actions get performed). Then, add another output "OnTrigger" to the corp handler, doing "Enable" after the same .01 seconds.
9. Select the corp_handler and do the same thing, so that "OnTrigger" the corp_hanlder gets "Disable" after .01 seconds and "OnTrigger" the punk_handler gets "Enable" after .01 seconds.
10. Finally, go to the logic_relay handler whose respective team first controls the node, and set the "Start Disabled" key to "true".



What your toggle screen should look like once complete.

Now, when the offending team hits the screen, it will perform that team's action once and only once; then the defending team can perform their action, and it can be toggled back and forth. You may even wish to add logic_timers and disable the screen for a small amount of time, to prevent one player on each team sitting at the screen and taking turns pressing it. But this is the basic set-up and shouldn't ever fail you.

Categories: Support Pages | Mapping

navigation
- Main Page
- Community portal
- Current events
- Recent changes
- Random page
- Help

search
Go Search

toolbox
- What links here
- Related changes
- Special pages
- Printable version
- Permanent link

Turret setup

If you just add a turret (npc_turret_ceiling) into your map, you might notice it going spaz. Turrets are very volatile and should be approached with extreme caution! Here's how to tame the savage beasts...

1.
The circle base of a turret needs to be sitting slighting into a func_brush (and only a func_brush) with Solidity = Never Solid and Solid BSP = No.

2.
Then, you need to place a logic_auto entity with an output of:

OnMapSpawn
TurretName
Enable

Categories: Support Pages | Mapping

navigation
- Main Page
- Community portal
- Current events
- Recent changes
- Random page
- Help

search
Go Search

toolbox
- What links here
- Related changes
- Special pages
- Printable version
- Permanent link

Your First Dystopia Map

It is strongly recommended that you learn mapping on Half-Life 2 or Counter-Strike:Source, as mapping for Dystopia is harder than these.

That said, the author of this guide made a functional, if very simple, Dystopia within 24 hours of loading Hammer for the first time. So if you too are foolhardy and enthusiastic, this guide should show you how to do the same.

Contents [hide]

Step 1: Set Up Hammer

Follow the instructions in Setting Hammer Up for Dystopia. The easiest way to do this is to the GameConfig.txt file and place it in the appropriate directory.

Step 2: Make Your First HL2 Map

Follow the guide to create your first map at the Valve developer wiki. This wiki is also a useful resource for anything else you might want to know with Hammer.

Note that that tutorial is for Half-Life 2 mapping. The easiest way to launch Hammer in the right configuration is to open the SDK launcher, and at the bottom make sure that Engine Version is Half-Life 2: Episode 1, and the Current Game is Half-Life 2.

By the end of that tutorial, you should be able to make a simple "box" room containing the player, and a few entities like crates, fences and oildrums.

Step 3: Play With Hammer

Play around with Hammer for a bit. Add some more things like the crate. The Block tool has some interesting features for making primitives - there's a good guide at this webpage, as well as a lot of other helpful ideas.

The reason I say to do this is because basic familiarity with making an enclosed room and adding stuff to it makes the next step easy.

Step 4: Start Again, For Dystopia

Close Hammer. Set the Current Game (at the very bottom of the SDK launcher) to Dystopia. Launch Hammer.

If you open the map you just created, you're likely to get a few weird problems because the entities won't be recognised properly - Hammer was configured to use file from HL2, not Dystopia. So it's a good idea not to reopen that map. Instead, create an empty room again. If you can't do it from memory, you should have spent more time on Step 3.

Add a crate to the centre of the room - exactly like in the tutorial, as a func_breakable, etc. Again, if you can't do it, go back to step 3.

Step 5: Plan The Map

This is where things get tricky. Dystopia has different requirements to Half-Life 2 - it's a fairly thorough modification. A map needs a few things to not crash or do weird, unexpected stuff. At the very least, you map needs a spawn point and a spawn. We'll add a few more things so that it actually runs as a real map.

The Bare Minimum

Your map needs:

- one spawn and one spawn point

Extra Stuff

We'll also add

- a second spawn and spawn point (for the other team)
- an objective
- an entity to tie the objective to (we'll use the crate primitive you just added)
- a damage filter for the objective (to stop it being destroyed by its own team)
- a filter activator, to apply the damage filter to the objective
- a start position camera (or the camera will be drawn at the origin by default, which can result in some very weird rendering if it's on a face)
- a camera for when punks win
- a camera for when corps win

Then you need to link appropriate things together to make everything function as it should. Fret not! It isn't that hard.

The Entity Guide is your friend for a lot of these. If you want to know what something does, most things are explained in there. The previously-linked Steam wiki is helpful as well.

Step 6: Spawns

Read the pages on spawns and spawn points. One dys_spawn can control multiple dys_spawn_point's. Punks spawn at the lowest enabled spawn. Corps spawn at the highest enabled spawn. Spawns are added like any other entity - select the entity tool, click where you want it on the 3D map view, select the selector tool, and double click on it in the 3D map view to set its type and properties. You can't see spawns in-game - they're an invisible controller thingummy.

Place a dys_spawn on one side of the room.

Double click it to open its Object Properties.

Name it Obj1PunkSpawn.

Set its team to Punks, and its SpawnID to PunksHQ.

Place a dys_spawn on the other side of the room.

Name it Obj1CorpSpawn.

Set its team to Corps and its SpawnID to 2.

Now we add spawn points to go with them. Spawn points are added in the same way - with the Entity tool - but they have a model.

Place a dys_spawn_point beside the PunksHQ dys_spawn. Set its SpawnID to PunksHQ.

Place a dys_spawn_point beside dys_spawn #2. Set its SpawnID to 2.

Congratulations, spawning is set up.

Step 7: The Objective

You should have made a destroyable box-shaped primitive and placed it in the middle of the room. Set its name to Obj1Crate. We want to make an objective that requires punks to shoot this crate, and corps to defend it.

dys_objective

To do this, we place a dys_objective entity. It's another invisible controller thingummies. Place it beside the crate for convenience. Double click on it to bring up its Object Properties.

Set is Name to Obj1 (or whatever you want).

Set its Team to Corps. This is a Corp-held objective (i.e. it hasn't been blown up yet).

Set its Number to 1.

Leave it as Primary Objective, and in Meatspace.

Set its Objective Text for Punks to something descriptive. "Smash Crate" will do. Likewise set its Objective Text for Corps to "Defend Crate".

Set its Objective Entity to Obj1Crate in the pull-down menu.

Select the Flags tab at the top of the screen.

Check the "Final Objective" box, since this is the only objective we have.

Now we've set up the objective. A spinning blue O will draw on the HUD at the crate's position. But we still need to connect it to the crate so that the objective is captured by punks when it's destroyed.

Objective Outputs

Double click the crate to open its Object Properties.

Select the Outputs tab at the top of the screen.

Select the Add button at the bottom.

Set the "My output named:" OnBreak. (We want it to trigger an output when the crate breaks).

Set "Targets entities named:" Obj1. (We want the output to connect to the objective)

Set "Via this input:" SetPunks. (The punks capture the objective).

Set "After a delay of:" 0.00 seconds (insta-trigger).

Check the Fire Once Only box.

The crate is now connected to the objective. When the crate is destroyed, it sets the objective to Punks.

But we haven't set a damage filter for it, so both Corps and Punks can damage the precious crate. Corps don't want to destroy their own crate, so we'll assume that they are prevented from harming it by a new technology involving nanobots and a friendly wizard. To implement this:

filter_activator_team

Create a filter_activator_team entity and place it beside the crate (and the dys_objective).

Double click it to open its Object Properties.

Name it CorpsNotDamage.

Set its Filter mode to "Disallow entities that match criteria".

Set its Filter Team Number to Corps.

So the filter is created! Now we need to tie it to the objective crate.

Double click the Obj1Crate to open its Object Properties.

Set its Damage Filter to CorpsNotDamage. Congratulations, it is now impervious to harm from Corps.

Step 8: Cameras

We need to set up a start position camera, which is what camera we use when waiting for the round to begin. If we don't set one, the default used is the origin, and this will either look weird and unplanned, or even break things if it's on a surface. So we'll set a camera to prevent this.

info_camera_start

Place an info_camera_start on the roof above the crate (or anywhere else you want, really).

Double click it to open its Object Properties.

Select its Pitch Yaw Roll, and select Point At.

Move the Object Properties window, and click on the Obj1Crate. This points the camera at the crate.

That's the start set up, but we want something to look at when the round is over and people are gloating over getting three awards.

info_camera_punkswin

This is what you'll see behind the scoreboard after a Punk victory. It's exactly like the info_camera_start for now, but it's got a different name. Place it wherever you want.

info_camera_corpswin

This is what you'll see behind the scoreboard after a Corp victory. Again, like the other cameras, place it wherever you want.

Step 9: Compiling and Running

Press F9.

Check the box "Don't run the game after compiling" - we'll start the game separately, it may crash if you run it immediately after compiling. Feel free to try with this box unchecked, though.

Press OK. The map should compile. If you have leaks, you should have spent more time on Step 3.

Launch Dystopia. Select "Create Server" and the name of the map you made. It should launch quickly - it's not a complicated map. You should see the view from your info_camera_start camera.

Join a team. The round doesn't start until you do. You'll see it's a 90 second wait.

Useful Console Commands

mp_startdelay X (where X is the delay time. Default is 90 seconds, which gets boring) map_restart (mp_startdelay doesn't activate until the next restart - use this to activate it) map dys_whatever (changes to map dys_whatever. Useful for changing maps quickly.)

When you spawn as Corps

Your objective menu should say "Guard Crate" or whatever you wrote. The crate should be invulnerable.

When you spawn as Punks

Your objective menu should say "Smash Crate" or whatever you wrote. The crate has 1 life if you made it properly, so it'll die to anything. This should trigger the objective capture and round end - triggering the view from your info_camera_punkswin.

Further Reading

Now that you've got that working, play around with Hammer. Add some more objectives, spawns, whatever. Mapping is one of those things best learnt by doing.

If you seek further guidance, perhaps its time to make Your Second Dystopia Map.

Retrieved from "http://www.dystopia-game.com/wiki/index.php?title=Your_First_Dystopia_Map"

Category: Mapping

navigation
- Main Page
- Community portal
- Current events
- Recent changes
- Random page
- Help

search
Go Search

toolbox
- What links here
- Related changes
- Special pages
- Printable version
- Permanent link

Cyber drain

SUPPORT 1.1 VERSION

The correct title of this article is "cyber_drain". The initial letter is capitalised due to technical restrictions.

Entity Description

A trigger volume that periodically drains the energy of any player who touches it.

Keyvalues

- **Name**
  Defines the name that other entities refer to this entity by.
- **Parent**
  Specifies the targetname of this entity's Parent in a movement hierarchy. *Child* entities move with their Parent.
  If a trigger does not start out with a parent, but rather gets one during run-time, this value should be set to a static entity. This is so the entity's solidity is set to VPhysics rather than BSP.
- **Origin (X Y Z)**
  <origin> The position of this entity's center in the world. Rotating entities rotate around their origin.
- **Start Disabled**
  <boolean>
- **Global Entity Name**
  <string> Name by which this entity is linked to another entity in a different map. When the player transitions to a new map, entities in the new map with globalnames matching entities in the previous map will have the previous map's state copied over their state.
- **Filter Name**
  <filterclass> Filter to use to see if activator triggers me. See the filter_activator_name entity for further explanation.
- **DrainVel**
  The amount of energy drained per second

Flags

- 1 : Applies to Clients
- 2 : Applies to NPCs
- 4 : Applies to Pushables
- 8 : Applies to Physics Objects
- 16 : Applies to Player Ally NPCs
- 32 : Applies to Clients in Vehicles
- 64 : Applies to Everything
- 512 : Applies to Clients not in Vehicles
- 1024 : Applies to Physics debris
- 2048 : Applies only to Player Ally NPC's in vechiles

Inputs

- **Kill**
  Removes this entity from the world.
- **KillHierarchy**
  Removes this entity and all its children from the world.
- **AddOutput <string>**
  Evaluates a keyvalue/output on this entity. It can be potentially very dangerous, use with care.
  Format: `<key> <value>`
  Format: `<output name> <targetname>:<inputname>:<parameter>:<delay>:<max times to fire (-1 == infinite)>`
- **FireUser*1-4***
  Causes this entity's OnUser*1-4* outputs to be fired. See User Inputs and Outputs.
- **Enable**
  Enable this entity.
- **Disable**
  Disable this entity.
- **SetParent <target_destination>**
  Changes the entity's parent in the movement hierarchy.
- **SetParentAttachment <string>**
  Change this entity to attach to a specific attachment point on its parent. Entities must be parented before being sent this input. The parameter passed in should be the name of the attachment.
- **SetParentAttachmentMaintainOffset <string>**
  Change this entity to attach to a specific attachment point on it's parent. Entities must be parented before being sent this input. The parameter passed in should be the name of the attachment. The entity will maintain it's position relative to the Parent's Origin and Angles at the time it is attached.
- **ClearParent**
  Removes this entity from its current movement hierarchy.
- **Toggle**
  (usage unknown)

Outputs

- **OnUser*1-4***
  Fired in response to FireUser*1-4* *inputs. See User Inputs and Outputs.*
  !activator = activator
- **OnStartTouch**
  Fired when an entity starts touching this trigger. The touching entity must pass this trigger's filters to cause this output to fire. !activator = toucher
- **OnEndTouch**
  Fired when an entity stops touching this trigger. Only entities that passed this trigger's filters will cause this output to fire. !activator = exiting entity
- **OnEndTouchAll**
  Fires when the last entity in the entity's area exits this trigger or when this entity is disabled. Only entities that passed this trigger's filters are considered. !activator = last exiting entity

Categories: Support Pages | Outdated Game Info | Dystopia Entity Guide | Cyberspace Entities

navigation
- Main Page
- Community portal
- Current events
- Recent changes
- Random page
- Help

search
Go Search

toolbox
- What links here
- Related changes
- Special pages
- Printable version
- Permanent link

Cyber floor

SUPPORT 1.1 VERSION

The correct title of this article is "cyber_floor". The initial letter is capitalised due to technical restrictions.

Entity Description

A brush entity that will affect a player's orientation/gravity so that any face on the brush will become the floor when the player touches it. These are typically indicated with distinctive paths in cyberspace.

Keyvalues

- **Name**

  Defines the name that other entities refer to this entity by.

Flags

None

Inputs

- **Kill**

  Removes this entity from the world.

- **KillHierarchy**

  Removes this entity and all its children from the world.

- **AddOutput <string>**

  Evaluates a keyvalue/output on this entity. It can be potentially very dangerous, use with care.

  Format: `<key> <value>`

  Format: `<output name> <targetname>:<inputname>:<parameter>:<delay>:<max times to fire (-1 == infinite)>`

- **FireUser*1-4***

  Causes this entity's OnUser*1-4* outputs to be fired. See User Inputs and Outputs.

Outputs

OnUser*1-4*

Fired in response to FireUser*1-4* *inputs. See User Inputs and Outputs*.

!activator = activator

Retrieved from "http://www.dystopia-game.com/wiki/index.php?title=Cyber_floor"

Categories: Support Pages | Outdated Game Info | Dystopia Entity Guide | Cyberspace Entities

navigation

- Main Page
- Community portal
- Current events
- Recent changes
- Random page
- Help

search

Go Search

toolbox

- What links here
- Related changes
- Special pages
- Printable version
- Permanent link

Cyber gravity

SUPPORT 1.1 VERSION

The correct title of this article is "cyber_gravity". The initial letter is capitalised due to technical restrictions.

Entity Description

A point entity that, when activated by a **trigger_multiple**, will change a decker's physics from link (no gravity) to node (with gravity). Set the pitch-yaw-roll of this entity so that it points into the room with gravity, or it will not work properly.

Keyvalues

- **name**
  Defines the name that other entities refer to this entity by.
- **Pitch Yaw Roll (Y Z X)**
  This entity's angular orientation in the world (also used for angular effect entities).

Flags

None

Inputs

- **AddOutput <string>**
  Adds an entity I/O connection to this entity. Format: <output name> <targetname>:<inputname>:<parameter>:<delay>:<max times to fire (-1 == infinite)>. Very dangerous, use with care.
- **FireUser*1-4***
  Causes this entity's OnUser*1-4* outputs to be fired. See User Inputs and Outputs.
- **GravityMe**
  Changes the cyber gravity characteristics of the activator
- **Kill**
  Removes this entity from the world.
- **KillHierarchy**
  Removes this entity and all its children from the world.

Outputs

- **OnUser1-4**
  Fired in response to FireUser1-4 inputs. See User User Inputs and Outputs.
  !activator = activator

Retrieved from "http://www.dystopia-game.com/wiki/index.php?title=Cyber_gravity"

Categories: Support Pages | Outdated Game Info | Dystopia Entity Guide | Cyberspace Entities

navigation
- Main Page
- Community portal
- Current events
- Recent changes
- Random page
- Help

search
Go Search

toolbox
- What links here
- Related changes
- Special pages
- Printable version
- Permanent link

Cyber ice

SUPPORT 1.1 VERSION

The correct title of this article is "cyber_ice". The initial letter is capitalised due to technical restrictions.

Entity Description

A solid, brush-based entity, used to protect cyber nodes that house screens. When active, it will only allow friendly deckers through, and enemy deckers must hack it (*Note: You can set ICE so it cannot be hacked, see **Flags***) or find another way around.

Orientation

The cyber ice MUST be linked to a dys_cyberscreen if you want the hack programs to show up on it. It gets the Orientation from the cyber screen. meaning the ice must be parellel to the surface of the cyberscreen. If you set your ice to non hackable then you can ignore this.

Keyvalues

- **Name**
  Defines the name that other entities refer to this entity by.
- **Team**
  Sets the owner of the ICE when it spawns.
  <choices>

Literal Value	Description
0	Blocks Both Teams
2	Punks own/Can Pass
3	Corps Own/Can Pass

- **Wedge Delay**
  Sets how long it takes a wedge program to disable the ICE rather than the default? (Does not apear to be implemented)

Flags

- **Enabled**
  If checked, the ICE wall is on
  Not checked, ICE can be created at this location from the associated dys_cyberscreen or by other game/map logic
- **Hackable**
  If checked, this ICE can be affected by cyberspace programs such as 'Wedge' and 'Break'
  Not check, the ICE exists as is and can only be affected by game/map logic

Inputs

- **Kill**
  Removes this entity from the world.
- **KillHierarchy**
  Removes this entity and all its children from the world.
- **AddOutput <string>**
  Evaluates a keyvalue/output on this entity. It can be potentially very dangerous, use with care.
  Format: `<key> <value>`
  Format: `<output name> <targetname>:<inputname>:<parameter>:<delay>:<max times to fire (-1 == infinite)>`
- **FireUser*1-4***
  Causes this entity's OnUser*1-4* outputs to be fired. See User Inputs and Outputs.
- **SetCorps**
  Sets ownership of this entity to the Corp team
- **SetPunks**
  Sets ownership of this entity to the Punk team
- **CrackIce**
  Completely disables the ICE entity, allowing either team to pass
- **Reactivate**
  Restores the ICE entity, using the previously set team filter
- **WedgeIce**
  Temporarily removes the ICE entity allowing either team to pass

Outputs

- **OnUser*1-4***
  Fired in response to FireUser*1-4* *inputs. See User Inputs and Outputs.*
  !activator = activator

Retrieved from "http://www.dystopia-game.com/wiki/index.php?title=Cyber_ice"

Categories: Support Pages | Outdated Game Info | Dystopia Entity Guide | Cyberspace Entities

navigation
- Main Page
- Community portal
- Current events
- Recent changes
- Random page
- Help

search
Go Search

toolbox
- What links here
- Related changes
- Special pages
- Printable version
- Permanent link

Cyber jumppad

SUPPORT 1.1 VERSION

The correct title of this article is "cyber_jumppad". The initial letter is capitalised due to technical restrictions.

Entity Description

A brush-based entity that, when collided with, will remove the player's **on ground** flag and give them a boost in the direction of the angle that is entered. It is useful in nodes with lots of vertical movement, or to boost an avatar across a gap. This entity works like a trigger volume, rather than a cyber_floor, and as such will not show any textures placed on it. (Note: This entity does ***NOT*** need to be triggered.)

Keyvalues

- **Name**
  Defines the name that other entities refer to this entity by.
- **Parent**
  Specifies the targetname of this entity's Parent in a movement hierarchy. *Child* entities move with their Parent.
  If a trigger does not start out with a parent, but rather gets one during run-time, this value should be set to a static entity. This is so the entity's solidity is set to VPhysics rather than BSP.
- **Origin (X Y Z)**
  <origin> The position of this entity's center in the world. Rotating entities rotate around their origin.
- **Start Disabled**
  <boolean>
- **Global Entity Name**
  <string> Name by which this entity is linked to another entity in a different map. When the player transitions to a new map, entities in the new map with globalnames matching entities in the previous map will have the previous map's state copied over their state.
- **Filter Name**
  <filterclass> Filter to use to see if activator triggers me. See the filter_activator_name entity for further explanation.
- **Pitch Yaw Roll (Y Z X)**
  This entity's angular orientation in the world (also used for angular effect entities).
- **JumpVel**
  The amount of energy used to push the activator in the specified direction

Flags

- 1 : Applies to Clients
- 2 : Applies to NPCs
- 4 : Applies to Pushables
- 8 : Applies to Physics Objects
- 16 : Applies to Player Ally NPCs
- 32 : Applies to Clients in Vehicles
- 64 : Applies to Everything
- 512 : Applies to Clients not in Vehicles
- 1024 : Applies to Physics debris
- 2048 : Applies only to Player Ally NPC's in vechiles

Inputs

- **Kill**
  Removes this entity from the world.
- **KillHierarchy**
  Removes this entity and all its children from the world.
- **AddOutput <string>**
  Evaluates a keyvalue/output on this entity. It can be potentially very dangerous, use with care.
  Format: `<key> <value>`
  Format: `<output name> <targetname>:<inputname>:<parameter>:<delay>:<max times to fire (-1 == infinite)>`
- **FireUser*1-4***
  Causes this entity's OnUser*1-4* outputs to be fired. See User Inputs and Outputs.
- **Enable**
  Enable this entity.
- **Disable**
  Disable this entity.
- **SetParent <target_destination>**
  Changes the entity's parent in the movement hierarchy.
- **SetParentAttachment <string>**
  Change this entity to attach to a specific attachment point on its parent. Entities must be parented before being sent this input. The parameter passed in should be the name of the attachment.
- **SetParentAttachmentMaintainOffset <string>**
  Change this entity to attach to a specific attachment point on it's parent. Entities must be parented before being sent this input. The parameter passed in should be the name of the attachment. The entity will maintain it's position relative to the Parent's Origin and Angles at the time it is attached.
- **ClearParent**
  Removes this entity from its current movement hierarchy.
- **Toggle**
  (usage unknown)

Outputs

- **OnUser*1-4***
  Fired in response to FireUser*1-4* *inputs. See User Inputs and Outputs.*
  !activator = activator
- **OnStartTouch**
  Fired when an entity starts touching this trigger. The touching entity must pass this trigger's filters to cause this output to fire. !activator = toucher
- **OnEndTouch**
  Fired when an entity stops touching this trigger. Only entities that passed this trigger's filters will cause this output to fire. !activator = exiting entity
- **OnEndTouchAll**
  Fires when the last entity in the entity's area exits this trigger or when this entity is disabled. Only entities that passed this trigger's filters are considered. !activator = last exiting entity

Cyber speedpad

SUPPORT 1.1 VERSION

The correct title of this article is "cyber_speedpad". The initial letter is capitalised due to technical restrictions.

Entity Description

A brush-based entity that, when passed through, will increase the respective decker's speed (respecting their current direction) to a given value. Useful inside links (no gravity areas).

Keyvalues

- **Name**
  Defines the name that other entities refer to this entity by.
- **Parent**
  Specifies the targetname of this entity's Parent in a movement hierarchy. *Child* entities move with their Parent.
  If a trigger does not start out with a parent, but rather gets one during run-time, this value should be set to a static entity. This is so the entity's solidity is set to VPhysics rather than BSP.
- **Origin (X Y Z)**
  <origin> The position of this entity's center in the world. Rotating entities rotate around their origin.
- **Start Disabled**
  <boolean>
- **Global Entity Name**
  <string> Name by which this entity is linked to another entity in a different map. When the player transitions to a new map, entities in the new map with globalnames matching entities in the previous map will have the previous map's state copied over their state.
- **Filter Name**
  <filterclass> Filter to use to see if activator triggers me. See the filter_activator_name entity for further explanation.
- **NewSpeed**
  Speed to increase decker's movement to

Flags

- 1 : Applies to Clients
- 2 : Applies to NPCs
- 4 : Applies to Pushables
- 8 : Applies to Physics Objects
- 16 : Applies to Player Ally NPCs
- 32 : Applies to Clients in Vehicles
- 64 : Applies to Everything
- 512 : Applies to Clients not in Vehicles
- 1024 : Applies to Physics debris
- 2048 : Applies only to Player Ally NPC's in vechiles

Inputs

- **Kill**
  Removes this entity from the world.
- **KillHierarchy**
  Removes this entity and all its children from the world.
- **AddOutput <string>**
  Evaluates a keyvalue/output on this entity. It can be potentially very dangerous, use with care.
  Format: `<key> <value>`
  Format: `<output name> <targetname>:<inputname>:<parameter>:<delay>:<max times to fire (-1 == infinite)>`
- **FireUser*1-4***
  Causes this entity's OnUser*1-4* outputs to be fired. See User Inputs and Outputs.
- **Enable**
  Enable this entity.
- **Disable**
  Disable this entity.
- **SetParent <target_destination>**
  Changes the entity's parent in the movement hierarchy.
- **SetParentAttachment <string>**
  Change this entity to attach to a specific attachment point on its parent. Entities must be parented before being sent this input. The parameter passed in should be the name of the attachment.
- **SetParentAttachmentMaintainOffset <string>**
  Change this entity to attach to a specific attachment point on it's parent. Entities must be parented before being sent this input. The parameter passed in should be the name of the attachment. The entity will maintain it's position relative to the Parent's Origin and Angles at the time it is attached.
- **ClearParent**
  Removes this entity from its current movement hierarchy.
- **Toggle**
  (usage unknown)

Outputs

- **OnUser*1-4***
  Fired in response to FireUser*1-4* *inputs. See User Inputs and Outputs.*
  !activator = activator
- **OnStartTouch**
  Fired when an entity starts touching this trigger. The touching entity must pass this trigger's filters to cause this output to fire. !activator = toucher
- **OnEndTouch**
  Fired when an entity stops touching this trigger. Only entities that passed this trigger's filters will cause this output to fire. !activator = exiting entity
- **OnEndTouchAll**
  Fires when the last entity in the entity's area exits this trigger or when this entity is disabled. Only entities that passed this trigger's filters are considered. !activator = last exiting entity

Retrieved from "http://www.dystopia-game.com/wiki/index.php?title=Cyber_speedpad"

Categories: Support Pages | Outdated Game Info | Dystopia Entity Guide | Cyberspace Entities

navigation
- Main Page
- Community portal
- Current events
- Recent changes
- Random page
- Help

search
Go Search

toolbox
- What links here
- Related changes
- Special pages
- Printable version
- Permanent link

Dys ammodisp

SUPPORT 1.1 VERSION

The correct title of this article is "dys_ammodisp". The initial letter is capitalised due to technical restrictions.

Entity Description

Places an ammo dispenser in the world. It will give ammo to all nearby players and show up on the radar automatically.

Keyvalues

- **name**
  Defines the name that other entities refer to this entity by.

Flags

- **Disable Model**
  Hides the model and makes the dispenser not solid in game.

Inputs

- **AddOutput <string>**
  Adds an entity I/O connection to this entity. Format: <output name> <targetname>:<inputname>:<parameter>:<delay>:<max times to fire (-1 == infinite)>. Very dangerous, use with care.
- **FireUser*1-4***
  Causes this entity's OnUser*1-4* outputs to be fired. See User Inputs and Outputs.
- **Kill**
  Removes this entity from the world.
- **KillHierarchy**
  Removes this entity and all its children from the world.

Outputs

- **OnUser1-4**
  Fired in response to FireUser1-4 inputs. See User User Inputs and Outputs.
  !activator = activator

Retrieved from "http://www.dystopia-game.com/wiki/index.php?title=Dys_ammodisp"

Categories: Support Pages | Outdated Game Info | Dystopia Entity Guide

navigation
- Main Page
- Community portal
- Current events
- Recent changes
- Random page
- Help

search
Go Search

toolbox
- What links here
- Related changes
- Special pages
- Printable version
- Permanent link

Dys cyberscreen

SUPPORT 1.1 VERSION

The correct title of this article is "dys_cyberscreen". The initial letter is capitalised due to technical restrictions.

Entity Description

Cyber VGUI Screen, used for hackable, map-logic controlling, screens. Can have protections applied to it, and an ICE entity associated with it (if named ICE entity is specified), and therefor required to be hacked.

Keyvalues

- **Name**
  Defines the name that other entities refer to this entity by.
- **Parent**
  Specifies the targetname of this entity's Parent in a movement hierarchy. *Child* entities move with their Parent.
  If a trigger does not start out with a parent, but rather gets one during run-time, this value should be set to a static entity. This is so the entity's solidity is set to VPhysics rather than BSP.
- **Pitch Yaw Roll (Y Z X)**
  This entity's angular orientation in the world (also used for angular effect entities).
- **Panel Name** <string>
  Name of the panel to use. The name must also match the .res filename used for the screen's layout (found in /scripts/screens/) and referenced in the *map_name*_screens.txt file in the dystopia /maps/ directory.
- **Overlay Material** <string>
  Name of a material to overlay over the top of the VGUI screen.
  Note: This material must write Z for the VGUI screen to work.
- **Panel Width in World** <integer>
  Width of the panel in units.
- **Panel Height in World** <integer>
  Height of the panel in units.
- **Team**
  Sets the owner of the screen when it spawns with protection specified (see below).
  <choices>

Literal Value	Description
2	Punks own/Can Pass
3	Corps Own/Can Pass

- *Start w/Protection*
  Specifies type of protection level the screen should spawn with, if any.
  <choices>

Literal Value	Description
0	None
1	Password Protected (weakest)
2	Encryption Protected (strongest)

- *ICE entity name*
  If there is an ICE entity blocking access to this screen, enter the entity name here

Flags

None

Inputs

- **Kill**
  Removes this entity from the world.
- **KillHierarchy**
  Removes this entity and all its children from the world.
- **AddOutput <string>**
  Evaluates a keyvalue/output on this entity. It can be potentially very dangerous, use with care.
  Format: `<key> <value>`
  Format: `<output name> <targetname>:<inputname>:<parameter>:<delay>:<max times to fire (-1 == infinite)>`
- **FireUser*1-4***
  Causes this entity's OnUser*1-4* outputs to be fired. See User Inputs and Outputs.
- **SetInactive**
  Disables screen
- **SetActive**
  Enables screen
- **SetParent** **<target_destination>**
  Changes the entity's parent in the movement hierarchy.
- **SetParentAttachment** **<string>**
  Change this entity to attach to a specific attachment point on its parent. Entities must be parented before being sent this input. The parameter passed in should be the name of the attachment.
- **SetParentAttachmentMaintainOffset** **<string>**
  Change this entity to attach to a specific attachment point on it's parent. Entities must be parented before being sent this input. The parameter passed in should be the name of the attachment. The entity will maintain it's position relative to the Parent's Origin and Angles at the time it is attached.
- **ClearParent**
  Removes this entity from its current movement hierarchy.

Outputs

- **Button*1-6***
  Fired in response to player actions (pressing or selecting parts of the VGUI screen) which correspond to screen areas defined in its associated *panel_name*.RES file.
- **OnUser*1-4***
  Fired in response to FireUser*1-4* *inputs. See User Inputs and Outputs*.
  !activator = activator

Retrieved from "http://www.dystopia-game.com/wiki/index.php?title=Dys_cyberscreen"

Categories: Support Pages | Outdated Game Info | Dystopia Entity Guide | Cyberspace Entities

navigation
- Main Page
- Community portal
- Current events
- Recent changes
- Random page
- Help

search
Go Search

toolbox
- What links here
- Related changes
- Special pages
- Printable version
- Permanent link

Dys emp

The correct title of this article is "dys_emp". The initial letter is capitalised due to technical restrictions.

Entity Description

A static point entity which causes an EMP (*Electro-Magnetic Pulse*) effect to surrounding players, temporarily disrupting/disabling their implants.

Keyvalues

- **Name**
  Defines the name that other entities refer to this entity by.
- **Explosion radius**
  Defines how large an area is affected by the entity. Players within this radius will have their implants disrupted.
  (For reference, the EMP grenade's radius is 256 units).
- **Explosion damage**
  Defines how much associated physical damage is also meeded out with the EMP's disruption.
  (For reference, the EMP grenade's damage is 25 units).

Flags

none

Inputs

- **Kill**
  Removes this entity from the world.
- **KillHierarchy**
  Removes this entity and all its children from the world.
- **AddOutput <string>**
  Evaluates a keyvalue/output on this entity. It can be potentially very dangerous, use with care.
  Format: `<key> <value>`
  Format: `<output name> <targetname>:<inputname>:<parameter>:<delay>:<max times to fire (-1 == infinite)>`
- **FireUser*1-4***
  Causes this entity's OnUser*1-4* outputs to be fired. See User Inputs and Outputs.
- **Explode**
  Trigger the EMP effect on surrounding players.
- **SetDamge**
  Used to change the amount of damage the EMP entity will meed out.
- **SetRadius**
  Used to change the area of effect of the EMP entity.

Outputs

- **OnUser*1-4***
  Fired in response to FireUser*1-4*' *inputs. See User Inputs and Outputs*.
  !activator = activator

Retrieved from "http://www.dystopia-game.com/wiki/index.php?title=Dys_emp"

Categories: Support Pages | Outdated Game Info | Dystopia Entity Guide

navigation
- Main Page
- Community portal
- Current events
- Recent changes
- Random page
- Help

search
Go Search

toolbox
- What links here
- Related changes
- Special pages
- Printable version
- Permanent link

Dys forcefield

SUPPORT 1.1 VERSION

The correct title of this article is "dys_forcefield". The initial letter is capitalised due to technical restrictions.

Entity Description

A solid, brush-based entity, used to protect areas from one team or the other. When active, it will only allow friendly team members through.

Keyvalues

- **Name**
  Defines the name that other entities refer to this entity by.
- **Team**
  Sets the owner of the ICE when it spawns.
  <choices>

Literal Value	Description
2	Punks own/Can Pass
3	Corps Own/Can Pass

Flags

None

Inputs

- **Kill**
  Removes this entity from the world.
- **KillHierarchy**
  Removes this entity and all its children from the world.
- **AddOutput <string>**
  Evaluates a keyvalue/output on this entity. It can be potentially very dangerous, use with care.
  Format: `<key> <value>`
  Format: `<output name> <targetname>:<inputname>:<parameter>:<delay>:<max times to fire (-1 == infinite)>`
- **FireUser*1-4***
  Causes this entity's OnUser*1-4* outputs to be fired. See User Inputs and Outputs.
- **Enable**
  Enable this entity.
- **Disable**
  Disable this entity.
- **SetCorps**
  Sets ownership of this entity to the Corp team
- **SetPunks**
  Sets ownership of this entity to the Punk team

Outputs

- **OnUser*1-4***
  Fired in response to FireUser*1-4* *inputs. See User Inputs and Outputs*.
  !activator = activator

Retrieved from "http://www.dystopia-game.com/wiki/index.php?title=Dys_forcefield"

Categories: Support Pages | Outdated Game Info | Dystopia Entity Guide | Cyberspace Entities

navigation
- Main Page
- Community portal
- Current events
- Recent changes
- Random page
- Help

search
Go Search

toolbox
- What links here
- Related changes
- Special pages
- Printable version
- Permanent link

Dys forceloadout

SUPPORT 1.2 VERSION

The correct title of this article is "dys_forceloadout". The initial letter is capitalised due to technical restrictions.

Entity Description

An entity which, when triggered, forces the passed player entity into the defined class, with the selected weapon and implants. This does not kill players. Currently used for training/tutorial maps, not intended for game maps.

Keyvalues

- **Name**
  Defines the name that other entities refer to this entity by.
- **Class**
  The class the players are to become.
  <choices>

Literal Value	*Player Class*
1	Light
2	Medium
3	Heavy

- **Weapon**
  Defines the weapon the player should now have.
  <choices>

Literal Value	*Weapon* (by Player Class -- Light, *Medium*, Heavy)
1	Shotgun, *Assault Rifle*, or Rocket Launcher
2	Laser Rifle, *Grenade Launcher*, or Minigun
3	Boltgun, *MK808 Rifle*, or Ion Cannon
4	Dual Smartlock Pistols, *Tesla Rifle*, or Basilisk

Flags

- **Enhanced cyberdeck**
  If checked, the player will have Enhanced cyberdecking capabilities.
- **Cyberdeck**
  If checked, the player will have some Basic cyberdecking capabilities.
- **TAC scanner**
  If checked, the player will have Tactical Scanning capability.
- **Thermal**
  If checked, the player will have Thermal vision capability.
- **SWT**
  If checked, the player will have Sound Wave Triangulation capabilities.
- **IFFInfo**
  If checked, the player will have enhanced *Identification Friend or Foe* information tracking capabilities.
- **Cortex**
  If checked, the player will have a Cortex bomb implanted in their head.
- **Stealth**
  If checked, the player will have a Stealth suit on with its optional protection.
- **Mediplant**
  If checked, the player will have Self & Team member Healing capabilities.
- **Cold Suit**
  If checked, the player will have a Cold suit on with its persistent protection.
- **Leg Boosters**
  If checked, the player will have Leg boosting capabilities.
- **Sound Suppressor**
  If checked, the player will have Sound suppression capability.
- **Reflexes**
  If checked, the player will have enhanced (Wired) Reflexes.
- **SCS**
  If checked, the player will have enhanced energy (Superconductor Capacitor Storage) capability.
- **No melee weapon**
  If checked, the player will spawn without a Katana (or Fatman Fist if the player is a Heavy).
- **No secondary weapon**
  If checked, the player will spawn without a Machine Pistol.
- **No primary weapon**
  If checked, the player will spawn without any Primary Weapons.
- **No grenades**
  If checked, the player will spawn without Grenades.

Inputs

- **Kill**
  Removes this entity from the world.
- **KillHierarchy**
  Removes this entity and all its children from the world.
- **AddOutput <string>**
  Evaluates a keyvalue/output on this entity. It can be potentially very dangerous, use with care.
  Format: `<key> <value>`
  Format: `<output name> <targetname>:<inputname>:<parameter>:<delay>:<max times to fire (-1 == infinite)>`
- **FireUser*1-4***
  Causes this entity's OnUser*1-4* outputs to be fired. See User Inputs and Outputs.
- **ForceLoadout**
  Trigger the forced loadout onto all players.

Outputs

- **OnUser*1-4***
  Fired in response to FireUser*1-4*' *inputs. See User Inputs and Outputs*.
  !activator = activator

Retrieved from "http://www.dystopia-game.com/wiki/index.php?title=Dys_forceloadout"

Categories: Support Pages | Updated Game Info | Dystopia Entity Guide

navigation
- Main Page
- Community portal
- Current events
- Recent changes
- Random page
- Help

search
Go Search

toolbox
- What links here
- Related changes
- Special pages
- Printable version
- Permanent link

Dys helper

SUPPORT 1.1 VERSION

The correct title of this article is "dys_helper". The initial letter is capitalised due to technical restrictions.

Entity Description

This entity displays a "popup" message on the bottom of the player's screen, as seen in various dystopia v1.0 compatible maps such as Broadcast

Keyvalues

- **Name**
  Defines the name that other entities refer to this entity by.
- **Parent**
  Specifies the targetname of this entity's Parent in a movement hierarchy. *Child* entities move with their Parent.
  If a trigger does not start out with a parent, but rather gets one during run-time, this value should be set to a static entity. This is so the entity's solidity is set to VPhysics rather than BSP.
- **.res file**
  Name of the resource (.res) file that is used to define the helper box that is to be displayed
  (files stored in /resources/helper/*.res) See RES_Structure_Guide
- **Display Time**
  The amount of time in seconds that this information will be displayed on screen. *10 Seconds is default*
- **Sound Name**
  An *additional* sound that will be played along with the default chime. (Can be used to play a .WAV containing additional verbal information for the player, or a sound effect relevent to the timing of the helper's display)

Flags

None

Inputs

- **Kill**
  Removes this entity from the world.
- **KillHierarchy**
  Removes this entity and all its children from the world.
- **AddOutput <string>**
  Evaluates a keyvalue/output on this entity. It can be potentially very dangerous, use with care.
  Format: `<key> <value>`
  Format: `<output name> <targetname>:<inputname>:<parameter>:<delay>:<max times to fire (-1 == infinite)>`
- **FireUser*1-4***
  Causes this entity's OnUser*1-4* outputs to be fired. See User Inputs and Outputs.
- **SetParent <target_destination>**
  Changes the entity's parent in the movement hierarchy.
- **SetParentAttachment <string>**
  Change this entity to attach to a specific attachment point on its parent. Entities must be parented before being sent this input. The parameter passed in should be the name of the attachment.
- **SetParentAttachmentMaintainOffset <string>**
  Change this entity to attach to a specific attachment point on it's parent. Entities must be parented before being sent this input. The parameter passed in should be the name of the attachment. The entity will maintain it's position relative to the Parent's Origin and Angles at the time it is attached.
- **ClearParent**
  Removes this entity from its current movement hierarchy.
- **ShowToTeam**
  Specifies exact team to show the helper to when activated. (Most common useage and most reliable)
  <choices>

Literal Value	Description
2	Display to Punks
3	Display to Corps

- **ShowToPlayer**
  Shows helper to player who triggered it
- **ShowToAll**
  Show to all players when triggered
- **ShowToPlayersTeam**
  Show to team of player who triggered it, team info may not pass through some entities. (Use ShowToTeam instead)

Outputs

- **OnUser*1-4***
  Fired in response to FireUser*1-4* *inputs. See User Inputs and Outputs.*
  !activator = activator

Retrieved from "http://www.dystopia-game.com/wiki/index.php?title=Dys_helper"

Categories: Support Pages | Outdated Game Info | Dystopia Entity Guide

navigation
- Main Page
- Community portal
- Current events
- Recent changes
- Random page
- Help

search
Go Search

toolbox
- What links here
- Related changes
- Special pages
- Printable version
- Permanent link

Dys item

SUPPORT 1.2 VERSION

The correct title of this article is "dys_item". The initial letter is capitalised due to technical restrictions.

Entity Description

An entity that can be picked up and moved. Extremely usefull in "steal data" and CTF style maps. As of version 1.2, dys_item requires a trigger_item to detect whether someone is holding the item or not, as well as a host of other entities governing respawning, all of which is covered in the CTF Mapping Guide.

Keyvalues

- **Name**
  Defines the name that other entities refer to this entity by.
- **world model**
  Name of the *model* (.mdl) file that is to be displayed and reacted with for this entity
  (files stored in /models/*.mdl) (Maximum string length: 128)
- **skin**
  <integer> Some models have multiple versions of their textures called skins. Set this to a number other than 0 to use that skin instead of the default.
- **disableshadows**
  <boolean> Used to disable dynamic shadow casting from this entity.
- **Filter**
  <filterclass> Filter to use to see if activator triggers me. See the filter_activator_name entity for further explanation.
  The dys_item can only be picked up by people that pass this filter. If this is left blank, then anyone can pick up this item.
- **Respawn Delay**
  Sets the time it will sit on the ground before it respawns.

Flags

- **Spawn only once**
  I'm assuming that if this is checked the item wont respawn after the respawn time runs out..?

Inputs

- **AlternativeSorting**
  ??????????? [under construction]
- **Kill**
  Removes this entity from the world.
- **KillHierarchy**
  Removes this entity and all its children from the world.
- **AddOutput <string>**
  Evaluates a keyvalue/output on this entity. It can be potentially very dangerous, use with care.
  Format: `<key> <value>`
  Format: `<output name> <targetname>:<inputname>:<parameter>:<delay>:<max times to fire (-1 == infinite)>`
- **FireUser*1-4***
  Causes this entity's OnUser*1-4* outputs to be fired. See User Inputs and Outputs.
- **DisableShadow**
  Used to disable dynamic shadow casting from this entity.
- **EnableShadow**
  Used to enable dynamic shadow casting from this entity.
- **Respawn**
  Causes the dys_item to return to it's original location.
- **Skin**
  <integer> Sets the skin of the entity's model. (Ignored if multiple skins are not defined for the model)

Outputs

- **OnUser*1-4***
  Fired in response to FireUser*1-4* *inputs. See User Inputs and Outputs.*
  !activator = activator
- **OnIgnite**
  Fired when this entity is ignited (somehow)
- **OnPickup**
  Fired when this entity is picked up by a player in the game
- **OnRespawn**
  Fired when this entity respawns after being dropped

Retrieved from "http://www.dystopia-game.com/wiki/index.php?title=Dys_item"

Categories: Support Pages | Updated Game Info | Worldmodel Entities | Dystopia Entity Guide

navigation
- Main Page
- Community portal
- Current events
- Recent changes
- Random page
- Help

search
Go Search

toolbox
- What links here
- Related changes
- Special pages
- Printable version
- Permanent link

Dys jackpoint

SUPPORT 1.1 VERSION

The correct title of this article is "dys_jackpoint". The initial letter is capitalised due to technical restrictions.

Entity desciption

A brush-based monitor that renders the view from a given point_camera entity in CyberSpace, and allowing a decker to enter cyberspace from this brush's location.

When a dys_jackpoint is created, the surface on which the view from the point_camera will be displayed must be assigned to a material that uses the special value "_rt_Camera" in its "$basetexture" property (such as the dys_bwmonitor1a-4a, dys_monitor1a-4a, dys_transmonitor1a-4a textures, etc.). It is also a good idea to 'Fit' the texture in the Face Edit dialog so the camera view isn't cut off.

Keyvalues

- **Name**
  Defines the name that other entities refer to this entity by.
- **Parent**
  Specifies the targetname of this entity's Parent in a movement hierarchy. *Child* entities move with their Parent.
  If a trigger does not start out with a parent, but rather gets one during run-time, this value should be set to a static entity. This is so the entity's solidity is set to VPhysics rather than BSP.
- **Origin (X Y Z)**
  <origin> The position of this entity's center in the world. Rotating entities rotate around their origin.
- **renderfx**
  <choices>

Literal Value	Description
0	Normal
1	Slow Pulse
2	Fast Pulse
3	Slow Wide Pulse
4	Fast Wide Pulse
5	Slow Fade Away
6	Fast Fade Away
7	Slow Become Solid
8	Fast Become Solid
9	Slow Strobe
10	Fast Strobe
11	Faster Strobe
12	Slow Flicker
13	Fast Flicker
14	Constant Glow
15	Distort
16	Hologram (Distort + fade)
17	Scale up
22	Spotlight FX
23	Cull By Distance (TEST)
24	Fade Wider Pulse

- **rendermode**
  <choices> Used to set a non-standard rendering mode on this entity. See also 'FX Amount' and 'FX Color'.

Literal Value	Render Mode	Description (Click for more info.)
0	Normal	Default rendering.
1	Color	Obsolete.
2	Texture	Plain opacity.
3	Glow	No Z buffer checks. Fixed size *on screen*.
4	Solid / Alphatest	Obsolete. (Use alphatest capabilities in materials instead.)
5	Additive	Obsolete. Add this entity's color to what's behind it.
7	Additive Fractional Frame	Obsolete? Blend between animation frames.
8	Alpha Add (Not in FGD.)	Add alpha.
9	World Space Glow	No Z buffer checks. Fixed size *in world* (as opposed to on screen).
10	Dont Render / None	Don't render.

- **FX Amount (0 - 255)**
  <integer> *renderamt* - How opaque the entity will be rendered. (**0** is fully transparent, while **255** is fully opaque). Will not work if the *rendermode* keyvalue is set to *Normal* or *Dont Render*.
- **FX Color (R G B)**
  <color255> - *rendercolor* - What color the entity will be rendered in (to the degree of the opacity set with *renderamt*).
- **Global Entity Name**
  <string> Name by which this entity is linked to another entity in a different map. When the player transitions to a new map, entities in the new map with globalnames matching entities in the previous map will have the previous map's state copied over their state.
- **InputFilter**
  Used to specify which inputs this entity will accept.

Literal Value	Description
0	Allow Everything
22	Touch/Untouch: Players
20	Touch/Untouch: Players & NPCs
16	Touch/Untouch: Players, NPCs, Pushables
18	Touch/Untouch: Players & Pushables
21	Touch/Untouch: NPCs
17	Touch/Untouch: NPCs & Pushables
19	Touch/Untouch: Pushables
6	Touch/Untouch & Use: Players
2	Touch/Untouch & Use: Players & Pushables
4	Touch/Untouch & Use: Players & NPCs
5	Touch/Untouch & Use: NPCs
1	Touch/Untouch & Use: NPCs & Pushables
3	Touch/Untouch & Use: Pushables
14	Use: Players
12	Use: Players & NPCs
8	Use: Players, NPCs, & Pushables
10	Use: Players & Pushables
13	Use: NPCs
9	Use: NPCs & Pushables
11	Use: Pushables

- **Start Disabled**
  <boolean> >
- **disableshadows**
  <boolean> > Used to disable dynamic shadow casting from this entity.
- **Minimum Light Level**
  <integer> *_minlight* -- The minimum level of ambient light that hits this brush.
- **Solidity**
  <choices> Used to control the solidity/collision of these brushes.

Literal Value	Description
0	Toggle
1	Never Solid
2	Always Solid

- **NPC class excluded from collisions**
  <target_destination> *excludednpc* -- If an NPC classname (i.e. npc_zombie) is specified here, NPCs of that type won't collide with these brushes.
- **Invert NPC class exclusion**
  <boolean> > *invert_exclusion* -- If set, then the excluded NPC class will consider this brush solid, and all other NPC classes will consider it non-solid.
- **Solid BSP**
  <boolean> > Set this if this brush is in hierarchy with a moving object of some kind, and the player can stand on this brush.
- **Shadows**
  <boolean> > Set this if this brush casts lightmap shadows.
- **Camera Name**
  <target_destination> The point_camera in which the viewpoint is rendered from.

Flags

1 : Ignore Player +USE

If checked, deckers will not be able to jack into cyberspace with their 'USE' key, forcing them to use the 'Fx' key or other configured key
If not checked, deckers can enter cyberspace with both their configured keys as well as the "USE" key

Inputs

- **Kill**
  Removes this entity from the world.
- **KillHierarchy**
  Removes this entity and all its children from the world.
- **AddOutput <string>**
  Evaluates a keyvalue/output on this entity. It can be potentially very dangerous, use with care.
  Format: `<key> <value>`
  Format: `<output name> <targetname>:<inputname>:<parameter>:<delay>:<max times to fire (-1 == infinite)>`
- **FireUser1-4**
  Causes this entity's OnUser1-4 outputs to be fired. See User Inputs and Outputs.
- **Color <color>**
  Set the rendercolor.
- **Alpha <byte>**
  Set the renderamt.
- **SetParent <target_destination>**
  Changes the entity's parent in the movement hierarchy.
- **SetParentAttachment <string>**
  Change this entity to attach to a specific attachment point on its parent. Entities must be parented before being sent this input. The parameter passed in should be the name of the attachment.
- **SetParentAttachmentMaintainOffset <string>**
  Change this entity to attach to a specific attachment point on it's parent. Entities must be parented before being sent this input. The parameter passed in should be the name of the attachment. The entity will maintain it's position relative to the Parent's Origin and Angles at the time it is attached.
- **ClearParent**
  Removes this entity from its current movement hierarchy.
- **Enable**
  Enable this entity.
- **Disable**
  Disable this entity.
- **DisableShadow**
  Turn shadow off.
- **EnableShadow**
  Turn shadow on.
- **SetExcluded**
  <target_destination> Set excludednpc
- **SetInvert**
  <integer> Set invert_exclusion

Outputs

- **OnJackIn**
  Fires when player jacks into cyberspace
- **OnJackOut**
  Fires when player exits cyberspace
- **OnUser1-4**
  Fired in response to FireUser1-4 *inputs*. *See User Inputs and Outputs*.
  !activator = activator

Retrieved from "http://www.dystopia-game.com/wiki/index.php?title=Dys_jackpoint"

Categories: Support Pages | Outdated Game Info | Dystopia Entity Guide

navigation
- Main Page
- Community portal
- Current events
- Recent changes
- Random page
- Help

search
Go Search

toolbox
- What links here
- Related changes
- Special pages
- Printable version
- Permanent link

Dys objective

SUPPORT

The correct title of this article is "dys_objective". The initial letter is capitalised due to technical restrictions.

Contents [hide]

Entity Description

This represents an objective in the map (such as "disable security", "destroy the core", etc). This entity controls who the objective belongs to, what the text for it is, and so forth. The position of this entity will be the position of the blue objective marker on the players' HUDs.

Keyvalues

- *Team*
  Which team owns this objective when the map starts. Can be Punks, Corps or None.
- *Objective Number*
  Objective number, from 1 to 6. If Punks are attacking, number in ascending order. If Corps are attacking, use descending order. For example, in Vaccine, the dock objective is #1 and the core objective is #3. If Punks were defending, the dock objective would be #3 and the core would be #1. Primary objectives are numbered separately from secondary objectives. *Confirmation needed as of 1.3*
- *Objective in cyberspace?*
  Used to display the hud icon only when jacked into cyberspace.
- *Primary Objective*
  Should, in theory, allow you to add this objective as either primary or secondary. This doesn't work, yet, though.
- *Objective text for Punks*
  Objective text for Punks, limited to 32 characters.
- *Objective text for Corps*
  Objective text for Corps, limited to 32 characters.

Flags

- *Final Objective*
  Ends the game when completed.

Inputs

- *Set Corps*
  Set the owner of this objective as the Corps.
- *Set Punks*
  Set the owner of this objective as the Punks.
- *Set Touched*
  Set the owner of this objective as the team of the player who trigger this input.
  Warning: Avoid using SetTouched at all costs, funky things happen when players get autobalanced, or the obj won't trigger at all.
- *SetHealth*
  Use with the OnHealthChanged output of a func_breakable to give it a health bar.

Outputs

- *On Corps*
  Fire when the team is set to Corps.
- *On Punks*
  Fire when the team is set to Punks.

Retrieved from "http://www.dystopia-game.com/wiki/index.php?title=Dys_objective"

Categories: Support Pages | Dystopia Entity Guide

navigation
- Main Page
- Community portal
- Current events
- Recent changes
- Random page
- Help

search
Go Search

toolbox
- What links here
- Related changes
- Special pages
- Printable version
- Permanent link

Dys onscreeninfo

SUPPORT 1.2 VERSION

This article has been marked as new, and may contain grammatical errors.

The correct title of this article is "dys_onscreeninfo". The initial letter is capitalised due to technical restrictions.

Entity Description

A static point entity which causes an Displays a icon simular to objective icons in the world, allowing it to display text on a crosshair over.

Keyvalues

- **Name**
  Defines the name that other entities refer to this entity by.
- **Parent**
  Children move in hirarcy with their parent entitys.
- **Info Text**
  Defines the text to be shown to players.
- **Max Draw Distance**
  Defines the max distance the entity can be seen fully before fading out.
  (For reference, the fade in distance is about 250units, setting the max distance lower than this will not make the entity visable).

Flags

- *Start Enabled for corps*
  Allow corps to see the entity
- *Start Enabled for punks*
  Allow punks to see the entity
- *Draw offscreen*
  Cause the icon to 'stick' to the edge of the screen simular to objective icons, instead of disapearing entirely.
- *Cyberspace*
  Only make the entity visable in cyberspace
- *CyberDeck*
  Only make the entity visable to thoes with the cyberdeck implant

Inputs

- **Kill**
  Removes this entity from the world.
- **KillHierarchy**
  Removes this entity and all its children from the world.
- **AddOutput <string>**
  Evaluates a keyvalue/output on this entity. It can be potentially very dangerous, use with care.
  Format: `<key> <value>`
  Format: `<output name> <targetname>:<inputname>:<parameter>:<delay>:<max times to fire (-1 == infinite)>`
- **FireUser*1-4***
  Causes this entity's OnUser*1-4* outputs to be fired. See User Inputs and Outputs.
- **SetText**
  Change the displayed text on the fly.
- **Disable**
  Stop showing the entity to everyone.
- **Enable**
  Show the entity to everyone.
- **EnablePunks**
  Show the entity to Punks.
- **EnableCorps**
  Show the entity to Corps.
- **EnableActivator**
  Show the entity to a specific player after firing a trigger_multiple/func_button etc.

Outputs

- **OnUser*1-4***
  Fired in response to FireUser*1-4* *inputs. See User Inputs and Outputs*.
  !activator = activator

Retrieved from "http://www.dystopia-game.com/wiki/index.php?title=Dys_onscreeninfo"

Categories: Support Pages | Updated Game Info | New Pages | Dystopia Entity Guide

navigation
- Main Page
- Community portal
- Current events
- Recent changes
- Random page
- Help

search
Go Search

toolbox
- What links here
- Related changes
- Special pages
- Printable version
- Permanent link

Dys screen

SUPPORT

The correct title of this article is "dys_screen". The initial letter is capitalised due to technical restrictions.

Entity Description

Meatspace VGUI Screen, for interactive panels in the meatworld

Keyvalues

- *Name*
  The name that other entities refer to this entity by.
- *Parent*
  The name of this entity's parent in the movement hierarchy. Entities with parents move with their parents.
- *Pitch Yaw Roll (Y Z X)*
  This entity's orientation in the world. Pitch is rotation around the Y axis, yaw is the rotation around the Z asix, and roll is the rotation around the X axis.
- *Panel Name*
  The panel name (as defined in /maps/mapname_screens.txt) to display on this panel. The name must also match the .res filename used for the screens layout (found in /scripts/screens/).
- *Overlay Material*
  Name of the material to overlay over the top of the VGUI screen. NOTE: This material must write Z for the VGUI screen to work.
- *Panel Width in World*
  Width of the panel in units.
- *Panel Height in World*
  Height of the panel in units.
- *Resolution Width*
  The resolution of the panel. NOTE: It helps to think of screens like computer monitors. Your screen has a width and a height, and a resolution. Usually the dimensions are expressed as a single value representing the diagonal length, and the resolution is a number such as 1024x768.
- *Resolution Height*
  The resolution of the panel.

Flags

None

Inputs

- *AddOutput* Adds an output to the screen
- *ClearParent* Clears the parent
- *Kill* Removes the entity from the world
- *KillHierarchy* Removes the entity and all entities which are parented to it (children) from the world
- *SetInactive* Disables the screen
- *SetActive* Enables the screen
- *SetParent* Sets the parent
- *SetText* Sets the text (Syntax unknown)

Outputs

- *Button1-6* fires in response to buttons in the .res file

Notes

To get the pannel to start as disabled, use a logic_auto to fire a SetInactive

Retrieved from "http://www.dystopia-game.com/wiki/index.php?title=Dys_screen"

Categories: Support Pages | Dystopia Entity Guide

navigation
- Main Page
- Community portal
- Current events
- Recent changes
- Random page
- Help

search
Go Search

toolbox
- What links here
- Related changes
- Special pages
- Printable version
- Permanent link

Dys spawn

SUPPORT

The correct title of this article is "dys_spawn". The initial letter is capitalised due to technical restrictions.

Contents [hide]

Entity Description

This entity dictates the location of a spawn, and also has all the inputs/outputs to control spawning. The position of this entity also effects the spawn's radar positioning. It is effectively a spawn controller for a group of dys_spawn_points

Keyvalues

- *Team*

  Which team the spawn starts as at the start of the map. Can be either neutral, punks or corps.

- *SpawnID*

  This is important. This is used to set the order of which spawns should be in the map. The punk's first spawn should be set to 1 (1 = Punks HQ), and each spawn should be set in order after this (eg. corps first spawn 2, next spawn 3, etc) and then the corps last spawn should be set to the last number (in this case 4). If you get these numbers wrong, it will most likely make your map crash when you load it.

Note: Be sure to have all spawnids used up to the highest one, if you leave any gaps like having 1, 2, 4, 5 as spawnids your map will crash.

Flags

None

Inputs

- *Enable*

  enables the spawn point

- *Disable*

  disables the spawn point

- *Toggle*

  changes the state of the spawn from disabled to enabled and vice versa

- *SetNeutral*

  keeps the spawn enabled, but sets it to neutral so neither team spawns there but it still shows on the radar

- *SetPunks*

  sets the spawn ownership to the punks team

- *SetCorps*

  sets the spawn ownership to the corps team

- *SetTouched*

  sets objective to the team of the person who activates the output. DO NOT USE THIS - it will cause things to break if players change teams.

- *ForceSpawn*

  immediately spawns any players waiting in the spawn queue

Outputs

- *OnCorpsCap*

  fires this output when the spawn ownership is changed to corps

- *OnPunksCap*

  fires this output when the spawn ownership is changed to punks

Categories: Support Pages | Dystopia Entity Guide

navigation
- Main Page
- Community portal
- Current events
- Recent changes
- Random page
- Help

search
Go Search

toolbox
- What links here
- Related changes
- Special pages
- Printable version
- Permanent link

Dys spawn point

SUPPORT

The correct title of this article is "dys_spawn_point". The initial letter is capitalised due to technical restrictions.

Contents [hide]

Entity Description

These are used to set the physical locations that players will spawn at. You should have at least 6 of these in each spawn location, they are controlled by dys_spawn entities

Keyvalues

- *SpawnID*

  Set this to match the dys_spawn's spawnid, so it knows where to spawn the players it has in its queue

Flags

- *Disable Model*

  Disables the model: Note that the player will fall a few units once spawned, thus they will not be able to change loadout

Inputs

None

Outputs

None

Retrieved from "http://www.dystopia-game.com/wiki/index.php?title=Dys_spawn_point"

Categories: Support Pages | Dystopia Entity Guide

navigation

- Main Page
- Community portal
- Current events
- Recent changes
- Random page
- Help

search

Go Search

toolbox

- What links here
- Related changes
- Special pages
- Printable version
- Permanent link

Dys spidershooter

SUPPORT

The correct title of this article is "dys_spidershooter". The initial letter is capitalised due to technical restrictions.

Contents [hide]

Entity Description

This entity spawns Spider Grenades. Those Spiders are teamless and attacking all Players in sight.

Keyvalues

- *Delay*

  Delay between spider spawns.
- *Spider count*

  Spawns the given amount of spiders when activated.

Flags

- *Repeatable*

  Can be triggered again and again.

Inputs

- *Shoot*

  Spawn spiders.

Outputs

None

Retrieved from "http://www.dystopia-game.com/wiki/index.php?title=Dys_spidershooter"

Categories: Support Pages | Dystopia Entity Guide

navigation
- Main Page
- Community portal
- Current events
- Recent changes
- Random page
- Help

search
Go Search

toolbox
- What links here
- Related changes
- Special pages
- Printable version
- Permanent link

Func monitor

SUPPORT

The correct title of this article is "func_monitor". The initial letter is capitalised due to technical restrictions.

Contents [hide]

Entity Description

This brush entity is used to render a camera view of another part of the map. All you have to do is apply the approprite monitor texture to this brush entity (dev_monitor* for _rt_Camera camera's, dys_monitor1* for camera1 cams, dys_monitor2* for camera2 cams etc). The engine checks if the monitor is in line of sight to the player to decide which ones will be rendered, For interactive monitors, see dys_screen / dys_cyberscreen (for cyberspace screens)

For extra Non-Dystopia Specific information: see Valve's SDK Article

Keyvalues

- *Camera name*

  The name for the camera it will render on the face of the monitor.

Flags

- *Enable Microphone*

  Allows audio to be passed through

Inputs

- *Enable*
- *Disable*
- *SetCamera*

  Sets the camera to use for this monitor. Takes the name of a point_camera entity in the map.

Outputs

None

Retrieved from "http://www.dystopia-game.com/wiki/index.php?title=Func_monitor"

Categories: Support Pages | Dystopia Entity Guide

navigation

- Main Page
- Community portal
- Current events
- Recent changes
- Random page
- Help

search

Go Search

toolbox

- What links here
- Related changes
- Special pages
- Printable version
- Permanent link

Info camera corpswin

SUPPORT

The correct title of this article is "info_camera_corpswin". The initial letter is capitalised due to technical restrictions.

Entity Description

This point entity is used for the "end of round" view on corp offensive maps

Keyvalues

- *Parent*

Flags

None

Inputs

None

Outputs

None

Categories: Support Pages | Dystopia Entity Guide

navigation

- Main Page
- Community portal
- Current events
- Recent changes
- Random page
- Help

search

Go Search

toolbox

- What links here
- Related changes
- Special pages
- Printable version
- Permanent link

Info camera punkswin

SUPPORT

The correct title of this article is "info_camera_punkswin". The initial letter is capitalised due to technical restrictions.

Entity Description

This point entity is used for the "end of round" view on punk offensive maps

Keyvalues

- *Parent*

Flags

None

Inputs

None

Outputs

None

Retrieved from "http://www.dystopia-game.com/wiki/index.php?title=Info_camera_punkswin"

Categories: Support Pages | Dystopia Entity Guide

navigation

- Main Page
- Community portal
- Current events
- Recent changes
- Random page
- Help

search

Go Search

toolbox

- What links here
- Related changes
- Special pages
- Printable version
- Permanent link

Npc turret ceiling

SUPPORT

The correct title of this article is "npc_turret_ceiling". The initial letter is capitalised due to technical restrictions.

Contents [hide]

Entity Description

A defensive turret that drops from the roof and rains the pain on enemies! Can't see people who're stealthed. Will only attack people who aren't on their team. Has heaps of inputs/outputs, so you can hack it from cyberspace & stuff.

Keyvalues

- *Team*

  Set to Punks (2) or Corps (3). This is the team the turret will start on, and how it knows who to attack and who not to.

Flags

- *Thermal*

  Allows the turret to track stealthed players as well as non-stealthed players

- *StartInactive*

  Always use this, to activate automatically, use a logic_auto, or the turret will not work right

Inputs

- *Disable*
- *Enable*
- *Toggle*
- *SetTeamTouched*

Outputs

- *OnDeploy*

  Probably triggers upon redeployment of turret after it's been shot down or disabled.

- *OnRetire*

  Probably triggers upon 'death' of turret after it's been shot down or disabled.

Categories: Support Pages | Dystopia Entity Guide

navigation

- Main Page
- Community portal
- Current events
- Recent changes
- Random page
- Help

search

Go Search

toolbox

- What links here
- Related changes
- Special pages
- Printable version
- Permanent link

Point camera

SUPPORT

The correct title of this article is "point_camera". The initial letter is capitalised due to technical restrictions.

Contents [hide]

Entity Description

Same as the hl2 camera, this is used to setup a camera that can render to a monitor, used extensively for cyberspace (it's also the entity used for a dys_jackpoint spawn destination in cyberspace)

Keyvalues

- *Render Target*

  This sets which camera texture it will use. This is used when setting up multiple monitors to be shown at once. The default texutre is _rt_Camera, for any other screens you want rendered at once, use camera1, camera2, camera3 & camera4.

Flags

None

Inputs

None

Outputs

None

Categories: Support Pages | Dystopia Entity Guide

navigation

- Main Page
- Community portal
- Current events
- Recent changes
- Random page
- Help

search

Go Search

toolbox

- What links here
- Related changes
- Special pages
- Printable version
- Permanent link

Sky camera cyber

SUPPORT

The correct title of this article is "sky_camera_cyber". The initial letter is capitalised due to technical restrictions.

Contents [hide]

Entity Description

Works the same as a normal sky_camera, only this one is used while the player is in cyberspace. When hammer compiles the map, it won't render any light_enviroment's in this skybox, thankfully.

Keyvalues

None

Flags

None

Inputs

None

Outputs

None

Categories: Support Pages | Dystopia Entity Guide | Cyberspace Entities

navigation
- Main Page
- Community portal
- Current events
- Recent changes
- Random page
- Help

search
Go Search

toolbox
- What links here
- Related changes
- Special pages
- Printable version
- Permanent link

Trigger item

SUPPORT 1.2 VERSION

The correct title of this article is "trigger_item". The initial letter is capitalised due to technical restrictions.

Entity Description

The trigger corresponding to dys_item, used to detect whether or not the client is holding an item.

Keyvalues

Name <String>
The name that other entities refer to this entity by.

Parent <String>
The name of this entity's parent in the movement hierarchy. Entities with parents move with their parents.
Note: If a trigger does not start out with a parent, but rather gets one during run-time, this value should be set to a static entity. This is so the entity's solidity is set to VPhysics rather than BSP.

Origin (X Y Z) <Origin>
The position of this entity's center in the world. Rotating entities typically rotate around their origin.

Start Disabled <Boolean>
Whether or not the entity starts disabled.

Global Entity Name
Name by which this entity is linked to another entity in a different map. When the player transitions to a new map, entities in the new map with globalnames matching entities in the previous map will have the previous map's state copied over their state.

Filter Name <Targetname>
Filter to use to see if activator triggers me. See filter_activator_name for details.

- **ItemsRequired**
  How many items are required to fire the trigger.
- **Item Name**
  If a special type of item is required, enter its name here. Assumed but not confirmed that if left blank, would fire on any item.

Flags

- **Trigger only once**
  Triggers only once

Inputs

- **Clear Parent**
  Removes this entity from its current movement hierarchy.
- **Enable**
  Enables this entity to perform its task. It might also disappear from view.
- **Disable**
  Disables this entity from performing its task. It might also disappear from view.
- **Reset**
  ??? Assumed that it caused the entity to reset in some form?
- **SetParent <Targetname>**
  Move with this entity.
- **SetParentAttachment <String>**
  Attach to a named attachment on the current parent. The entity will teleport so that the position of its root bone matches that of the attachment.
- **SetParentAttachmentMaintainOffset <String>**
  As above, but without teleporting. The entity retains its position relative to the attachment at the time of the input being received.
- **Toggle**
  Toggles this trigger between enabled and disabled states.
- **Kill**
  Removes this entity from the world.
- **KillHierarchy**
  Removes this entity and all its children from the world.
- **AddOutput <string>**
  Evaluates a keyvalue/output on this entity. It can be potentially very dangerous, use with care.
  Format: `<key> <value>`
  Format: `<output name> <targetname>:<inputname>:<parameter>:<delay>:<max times to fire (-1 == infinite)>`
- **FireUser*1-4***
  Causes this entity's OnUser*1-4* outputs to be fired. See User Inputs and Outputs.

Outputs

- **OnEndTouch**
  Fires the trigger when a client stops touching the trigger.
- **OnEndTouchAll**
  Fires the trigger when all clients stop touching the trigger.
- **OnItem**
  Fires the trigger when a client carrying an item touches the trigger.
- **OnStartTouch**
  Fires the trigger when a client starts touching the trigger.
- **OnStartTouchAll**
  Fires the trigger when all clients start touching the trigger.
- **OnUser*1-4***
  Fired in response to FireUser*1-4*' *inputs. See User Inputs and Outputs*.
  !activator = activator

Retrieved from "http://www.dystopia-game.com/wiki/index.php?title=Trigger_item"

Categories: Support Pages | Updated Game Info | Dystopia Entity Guide

navigation
- Main Page
- Community portal
- Current events
- Recent changes
- Random page
- Help

search
Go Search

toolbox
- What links here
- Related changes
- Special pages
- Printable version
- Permanent link

Weapon lightning

SUPPORT

The correct title of this article is "weapon_lightning". The initial letter is capitalised due to technical restrictions.

Entity Description

A special weapon, it's Teddy's physics gun (aka the Lightning Rod). You can place these in your map, but only heavies can pick them up and use them. The weapon will drop when the player dies, fun for some!

Keyvalues

None

Flags

- **Start Constrained**

  Item spawns where placed on the map rather than falling to the floor.

Inputs

None

Outputs

None

Categories: Support Pages | Dystopia Entity Guide

navigation

- Main Page
- Community portal
- Current events
- Recent changes
- Random page
- Help

search

Go Search

toolbox

- What links here
- Related changes
- Special pages
- Printable version
- Permanent link

Your Second Dystopia Map

Assumed reading: Your First Dystopia Map.

Let's add some more features to the map. We aren't ready to make the big jump to adding cyberspace just yet - we'll leave that for Your Third Dystopia Map.

In this map, we'll add turrets, buttons, doors and forcefields and who knows what else?

Contents [hide]

Step 1: Make a Copy

After making the first map you should have an enclosed box with two spawns and an objective. File -> Save As -> dys_whateverNameYouWant2

In Hammer, you want to make backups. Lots and lots of backups. If you screw something up, there's no undo key after you compiled, closed Hammer and loaded it up in Dystopia. Any damage is irreversible, so make backups.

We'll work on the new copy.

Step 2: Add A npc_turret_ceiling

Turrets are just another kind of entity.

Add a npc_turret_ceiling by clicking somewhere on the roof.

Double click on it to open its Object Properties.

Name it PunkTurret1.

Set its team to Punks.

If you want, set its health to whatever, or make it invulnerable.

For various reasons, we need a logic_auto invisible-magic-controller-thing to set the turret up properly. Create a logic_auto entity and place it near the turret (for convenience).

Double click on it to open its Object Properties.

Select the Outputs tab.

Click Add.

In "My output named:" type "OnNewGame"

Set "Target entities named:" PunkTurret1

Set "Via this input:" Enable.

This way, if in the first round, the turret is shot down or disabled just before map end, it'll re-enable for the next round. Additionally, I found the turret was invulnerable without one of those.

Press F9 to compile the map, close Hammer, and load up the map in Dystopia. Join as a Punk, make sure the turret is on, and destructible.

Step 3: Add A func_button

We'll add a button to toggle the turret on and off. Buttons are brush entities, not node entities.

Make a very small block - say 4x4x4 units. Put it on wall.

Apply a texture to it.

Double click on it to open its Object Properties.

Change its class to func_button in the pull-down menu.

Set the Name to ToggleTurretButton.

Set its Lip to 1 (or another number less than 4. we want it to move into the wall leaving only 1 inch sticking out).

Set its Speed to 10 (or another high number, we want the button to move fast.

Set the Delay Before Reset to 1 (or a similar low number).

Set the Sounds to something so you have audible confirmation it's working.

Select the Outputs tab.

Click Add.

Set "Targets Entities Named": PunkTurret1.

Set "Via This Input": Toggle.

Set "My output named": OnPressed.

Compile and run the map. Pressing the button as punk should disable or enable the turret.

Move Direction

You may notice that the button moves in a strange direction instead of receding into the wall. By default, all buttons (and doors) move in the positive-X direction (to the right, from default).

You can make it move in any direction - up, down, into or out of a wall, left or right - by setting the Move Direction property. Let's play with it now.

Double click on it to open its Object Properties.

Select the Move Direction field. There will be three numbers - by deault, 0 0 0. These are the pitch, yaw and roll. This is easiest to explain in terms of aeroplanes; pitch is when the nose of the plane goes up and down, yaw is when the nose of the aeroplane turns left or right, and roll is when the body of the aeroplane rotates around its longest axis (the wings spin around).

For a normal button or door, we can just set the Yaw value and leave the others zero. Assuming your room is square, set the Yaw value (the second of the three) to a multiple of 90; 0, 90, 180, 270.

Compile and run the map. Pressing the button as punk should disable or enable the turret. Watch which way the button moves. You should be able to see it move in a different direction.

If you want a button (or door) to move up or down, select Up or Down from the pull-down menu.

Set the button to move whatever direction you want; I personally like them to retract into the wall.

Step 4: Add a func_door

A door is a brush entity that moves a fixed distance in a straight line, in some direction.

Obviously, this has applications as a sliding door, but also has many other uses that we will get to.

There is also a func_door_rotating which has a hinged door that rotates some angle about some axis. This too has uses beyond just a door.

Draw a rectangular block the size and shape of your door. Draw two other blocks about the same size either side of it - these will be doorposts or walls (they aren't necessary but they make it easier to visualise).

Select the door block and click ToEntity.

Select func_door from the pull-down menu.

Name it door1.

Give it a few sounds for when it starts and stops moving, or locks, or unlocks.

Click Apply.

Go to the Flags tab, and check Use Opens. This will allow the door to be opened with the E key as well as just by touching it.

Compile and run the map. The door should move in some direction when touched. We want it to slide aside and let us through the doorway.

Re-launch Hammer and set the Move Direction so that the door opens how you want it to.

Compile and run the map to make sure the door works properly now.

The door will move along its entier length in the direction that it is set to move in.

Door Properties

You can set a few other interesting properties of doors.

- block damage, which damages anything caught in the door as it starts to close. The airlock doors on Vaccine have this set to a number approaching infinity.
- reset time. If you set it to -1, the door stays open. Set it to a number >= 0, the door will wait that long and then start to shut itself. Default is 4 seconds.
- lock or unlock. In the Flags tab, a Start Locked flag can be set - the door can be unlocked by a button in meatspace or cyberspace, or when an objective changes status.
- start position. Default is closed, but it can be set to open. If you want, you can set start position to open and tick Starts Locked, then have a button that will unlock, close and relock the door, to seal off an old objective or an alternate route.
- Touch Opens, in the Flags tab. Uncheck this and leave Use Opens unchecked, and you can make a door only controllable by cyberspace or a button.

Fun With Doors

Doors don't need to be attached to walls, floor or ceiling, or can be attached to any combination of the three. Think of them as floating blocks of any shape that move a certain distance in any direction, and then back again.

Sliding doors are just the beginning. Hatches, trapdoors, moving walls, retractable stairways and ladders, sliding platforms and moving pistons are all possible.

navigation
- Main Page
- Community portal
- Current events
- Recent changes
- Random page
- Help

search
Go Search

toolbox
- What links here
- Related changes
- Special pages
- Printable version
- Permanent link