BLACK MESA

GUIDE TO MAKING BLACK MESA MODS AND MAPS

- **Setting up the Black Mesa Tools**
  - *Setting up Hammer*
  - *VProject*
- **Publishing Mods on the Black Mesa Workshop**
  - *Setting up your folder structure*
  - *Generating a VPK*
  - *Publishing to the workshop*
- **Black Mesa Specific Mapping Entities**
  - *Loading Screens*
  - *Info_Observer_Menu*
  - *Generate Images*
  - *Hard Respawns*

Setting up the Black Mesa Tools

The Black Mesa tools should require no special setup, and work out of the box. You can find them located in *Steamapps/common/Black Mesa/bin.* Here you will find all of the usual Source tools like Hammer, Modelviewer (hlmv.exe), and everything else you would expect from the usual Source pipeline.

SETTING UP HAMMER

Everything within Hammer should work out of the box, all you will need to do to get creating maps for Black Mesa is to point Hammer to use the correct FGDs (entity definition files).

In Hammer, go to tools > options. Under the section which reads “Game Data files”, click the “add” button to add our FGD files. The 3 FGD files you should tell Hammer to use are:

```
Black Mesa/bin/base.fgd
Black Mesa/bin/bms.fgd
Black Mesa/bin/halflife2.fgd
```

Once you have set up the FGDs, Hammer is set up. Everything should work perfectly!

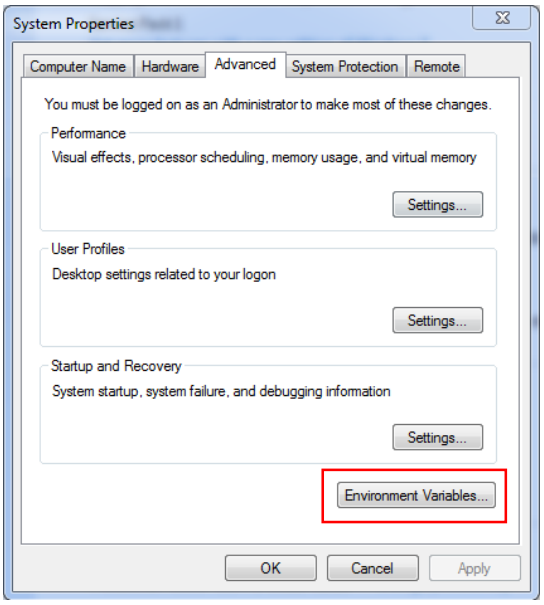
VPROJECT

This will not happen to most users. However, if you have worked on other Source games, you may get an error related to VPROJECT in Hammer. If this happens, you

need to set the environment variable to point to Black Mesa. Hammer may not work properly.

Right click on *"My Computer,"* then go to *"Properties".* Then *"Advanced system settings".*

On the dialog which appears, go to the *"advanced"* tab, and then click the *"Environment Variables"* button.



Under "user variables for *your name*", find VProject in the list. Click it, then press "Edit," and point it to your Black Mesa "bms" folder, which for most installs, will be "*C:\Program Files (x86)\Steam\SteamApps\common\Black Mesa\bms"*.

If VProject is not in the list, click "New" and add it. Give it the "Variable name" VProject, and a "Variable value" of the "bms" folder.

That's it, done!

Publishing Mods on the Black Mesa Workshop

This guide will explain how to publish a mod/map on the Black Mesa workshop.

To begin with, you need to have created a mod/map and have it in a state where you are happy to publish it. Fully featured mods and maps both follow the same conventions for publishing. You can check out this guide to find out how to use our SDK tools to make your own mod.

SETTING UP YOUR FOLDER STRUCTURE

The first step is to create a folder you will store your mod(s) in: *bms-mods*, or something to that effect (don't use spaces in path names, Source hates that). This folder can be anywhere. This will serve as the master folder which you store your

various mods/maps in, to make life easier.

Let's say the mod you are making is called *gaben_world*. In your *bms-mods* folder, make a folder called *gaben_world*. Imagine your mod contains some materials, some models, and a few maps/nodegraphs. The inside of the folder *gaben_world* should be made to mirror the structure of the BMS folder. So in this instance, you would create a materials, models, and maps folder inside the *gaben_world* folder, and then nest your appropriate content in those folders, how they would work if you wanted to install them.

Example file structure for mod *gaben_world:*

```
c:/bms-mods
>>
c:/bms-mods/gaben_world
>>
c:/bms-mods/gaben_world/materials/gaben_wall.vtf
c:/bms-mods/gaben_world/materials/gaben_wall.vmt
c:/bms-mods/gaben_world/models/gaben.mdl
c:/bms-mods/gaben_world/maps/dm_gaben.bsp
c:/bms-mods/gaben_world/maps/nodegraphs/dm_gaben.ain
```

GENERATING A VPK

Ensure all of your mod content is appropriately stored here, as it would be for the BMS folder. Now your mod folder is finished (in this example gaben_world), you need to turn the folder into a VPK. What is a VPK? You can think of a VPK as a zip file, or a form of compressed archive. All of your mod content will be contained within a single VPK, which mimics the folder structure of the BMS folder. Black Mesa reads these VPK files, and loads them as addons when the game boots up. This not only compresses the mod to reduce file size, but makes it quicker to load and install/uninstall too, as well as allowing you to mix and match mods!

Making a VPK is extremely simple. Simply take your *gaben_world* folder, select it, and drag and drop it onto VPK.exe. VPK.exe is found in the Black Mesa/bin folder, alongside all of Black Mesa's other modding tools. If you want to make things simpler for yourself, you can copy/paste VPK.exe into your bms-mods folder, and then use it from there. You can also write a batch file to run vpk for a specific folder. The batch file to make a VPK for *gaben_world* would look like this, in our example:

vpk "C:\bms-mods\gaben_world"

You need to run the batch file or drag/drop the folder onto VPK.exe every time you make a change/update to the map.

Assuming everything goes properly, you should now have a VPK in the same folder as VPK.exe, with the same name as the original folder. There are some caveats. The workshop only allows VPK of up to 100mb to be uploaded, and a mod upload can only contain a single VPK. This is just an unfortunate limitation of the workshop system. If your VPK is bigger than this file size limit, you will need to split it up into multiple parts, and upload them separately, as different mods. Make sure when you

upload mods which require multiple parts, that it is made clear that there are multiple compulsory parts!

If your mod is big enough to need to be split into multiple parts, you need to run a batch file to generate a multi-part VPK. Here is an example batch file for *gaben_world*:

vpk -M -c 75 "C:\bms-mods\gaben_world"

The -M switch tells VPK.exe to generate a multi-part VPK. The -c switch and the 75 tells it to try and keep the parts in a size smaller than 75MB.

PUBLISHING TO THE WORKSHOP

Now you have your VPK(s), you are ready to publish it/them to the workshop! This is done using the P2MapPublish.exe tool, also found in the bin folder, where you found VPK.exe. Again, for ease of use/simplicity, you can copy/paste this executable to the bms-mods folder if you find that nicer to work with.

The P2Publish tool is fairly straightforward. Run the tool, to access its main interface.

On the main interface, click "Add" to create a new mod. When using the P2MapPublish tool, make sure your Steam program is logged on and signed into the correct account you want, as this is the account it will publish the content under. Once you've told the P2Publish tool to create a new mod, enter in an appropriate title for the mod - if it is a map, Black Mesa typically uses the naming convention *xx_mapname*, with xx being the gametype prefix - bm for singleplayer, dm for multiplayer. But you can name it what you like, if so you desire. If it is a multi-part VPK, the mod name should specifiy this (example: *Gaben World [Part 1/3]*) You must add a preview image, this is the image people will see on the workshop that represents your mod. Choose it wisely! You must then fill in a small description, and accept the terms of use.

Once you are ready, click publish, and your VPK will be uploaded onto the Black

Mesa workshop. Congratulations, you have just published your first Black Mesa mod! You can then visit the workshop item on your profile to add more preview images, edit links, add/remove contributors, and plenty of other good stuff. That's not quite the end, though. Always test that your mod works by subscribing to it and making sure it functions in game, or asking a friend to do so for you!

Black Mesa Specific Mapping Entities

LOADING SCREENS

By default, Black Mesa has a nice little loading screen system, where during loading, the game UI displays a randomly selected screenshot of the map, and once loading is complete, the screen will seamlessly fade into the actual map, at the same location, creating a very nice transitional effect. The loading screen also displays some set information, like number of recommended players, a description of the map, etc. This system is extensible and flexible, allowing mappers and modders to generate their own loading screens and bundle them with their maps.

The system for producing these loading screens is relatively simple and elegant.

INFO_OBSERVER_MENU

The first step happens while mapping in Hammer. You must place an *info_observer_menu* in the map, and assign it an ID. The *info_observer_menu* is where the screenshot for the loading screen will be taken from, and where the UI will fade out to, leaving that as the starting camera position. The Black Mesa standard per map is to have 3 *info_observer_menus*, though you can have more or less if you desire. Assign them a unique ID in their properties. If you're using 3 cameras, the IDs should be 0, 1 and 2. These cameras should be places somewhere nice that has a good view of important parts of the map - stuff you'd want to show off in the loading screen and once the map loads up. This may require tweaking and playing around with.

That's the early prep finished! Pretty simple! Compile the map with these entities present.

GENERATE IMAGES

The next step is to generate the images from the cameras, which is a simple process. Load up the map at the desired resolution (make sure you've built cubemaps first). Open the console, and use the command *bm_generate_menu_images*. Your perspective will then jump between the cameras, and generate the 3 menu images. These are generated into the folder *materials/vgui/multiplayer/map_backgrounds/xx_mapname.vtf/vmt*. You will need to do this process 3 times, for Black Mesa's different supported aspect ratios - 4:3, 16:9, and 16:10. You will have to change resolution to do it. The recommended resolutions for the screens are - 1280 x 1024 (4:3), 1920 x 1080 (16:9) and 1650 x 1050 (16:10 - or 1920x1200 if your monitor supports it).

The end result is in *materials/vgui/multiplayer/xx_mapname.vtf/vmt,* you should have

9 vtfs. 3 for each aspect ratio. 3 will be standard, 3 will have the suffix _widescreen, and another 3 with the *suffix _widescreen_16_10*. Check the vtfs to make sure the screens generated properly, sometimes they will get messed up because of an error generating and the process may need to be repeated. Now you have your images, you just have to hook them up to be used and understood by the engine. This process is simple - it's done via a text file located in the maps folder. Here is how the file looks for the map *dm_gasworks*:

Let's break this file down a bit.

```
"dm_gasworks"
{
// The 'non mapper' name of the level.
"name" "GASWORKS"

// The number of recommended players.
"players" "6 - 16 PLAYERS"

// Image materials to draw. Randomly chosen.
// Path relative to materials. Must point to a
// VMT.
"Images"
{
"0" "multiplayer/map_backgrounds/dm_gasworks_0"
"1" "multiplayer/map_backgrounds/dm_gasworks_1"
"2" "multiplayer/map_backgrounds/dm_gasworks_2"
}

// Map specific descriptions. Also randomly chosen.
"Descriptions"
{
"0" "This Gasworks complex was built to power early teleportation tests.
Some clues about its experimental history lurk beneath the surface..."
"1" "This Gasworks complex was built to power early teleportation tests.
Some clues about its experimental history lurk beneath the surface..."
"2" "This Gasworks complex was built to power early teleportation tests.
Some clues about its experimental history lurk beneath the surface..."
}
}
```

The top line is the map's filename. This is needed so the engine knows what map to associate the information with. The text file itself should also have the same name as the map's filename. The "name" field is the "proper" name of the map, which will be displayed on the loading screen. The "players" field displays underneath the map's name, and is the recommended playercount for the map, for a good experience - not the min/max number of supported players. This is just an estimate from the mapper. For example, Gasworks can theoretically support up to 32 players (it has 24 spawns), but would not play well above 16 players, in the mapper's opinion.

The next 2 fields, for *"images"* and *"descriptions"* are randomly selected by the game. You can have as many as you want and the game will randomly pick one. So let's say the game picks image "1", it will also pick description "1" to go with it. This functionality is not used by default in Black Mesa, but modders/mappers could take advantage of this to tailor descriptions for each unique image. The Black Mesa standard is to use 3 images for each map (and 1 description), but the game can

theoretically support as many as you want. We recommend 3, to stop filesize bloat from having too many images, while keeping variety by having a few images. Notice how each image/description has an ID number before it. This is how the game knows which *info_observer_menu* ID to fade the image to. The *"0" "1"* and *"2"* fields correspond to the number of the *info_observer_menu* entity you placed in the map earlier. So in this example, when the menu decides randomly to display the image *"dm_gasworks_2",* it knows to fade to the camera ID *"2"* when it has finished loading.

Voila! You have created a nice flashy loading screen effect.

HARD RESPAWNS

All items on the map can be set to have a specific respawn timer, which by default is 15. Using the default behaviour, this means the item will respawn 15 seconds after it is picked up.

Simple.

But competitive players/level designers may want to cater to a different style of play. So we created the "hard respawn" setting, which can be set on any item in the editor. Let's say an item is set to hard respawn, with a respawn time of 60 seconds. That means that the item will respawn every 60 seconds, regardless of when it is picked up. This allows you to create maps where the weapons spawn at predictable times, which may be more desirable for the competitive types. For instance, you could have a Gluon in the center of the map with a hard respawn time of 2 minutes. In a 10 minute match, this means the Gluon will spawn on the 10:00, 8:00, 6:00, 4:00, and 2:00 minute marks, allowing the level designer to have better control of the game flow, and allowing competitive players to run different setups and plan accordingly.