Source Engine Blender Collection

Click to watch the youtube video!

Welcome to the **Source Engine Blender Collection!** A Blender archive of **45,000+** models, **45,000+** materials, and **550+** maps ported from eight of Valve's Source games! These ports were made with optimization and efficiency in mind, while catering to ease of usability. Below, you will find the installation instructions, download links for the ported archives, and general tips you may find useful. I've jam-packed this with features and nifty tricks, so this is definitely worth a read! If you're interested in the process, that is at the bottom of this document. A lot of love went into the project. I promise you it wasn't as simple as importing an asset!

Unfortunately, I do not have the luxuries of preparing a professional website for this, so hopefully Github's markdown is satisfactory!

Minimum Blender ver.: 3.6

- Credits

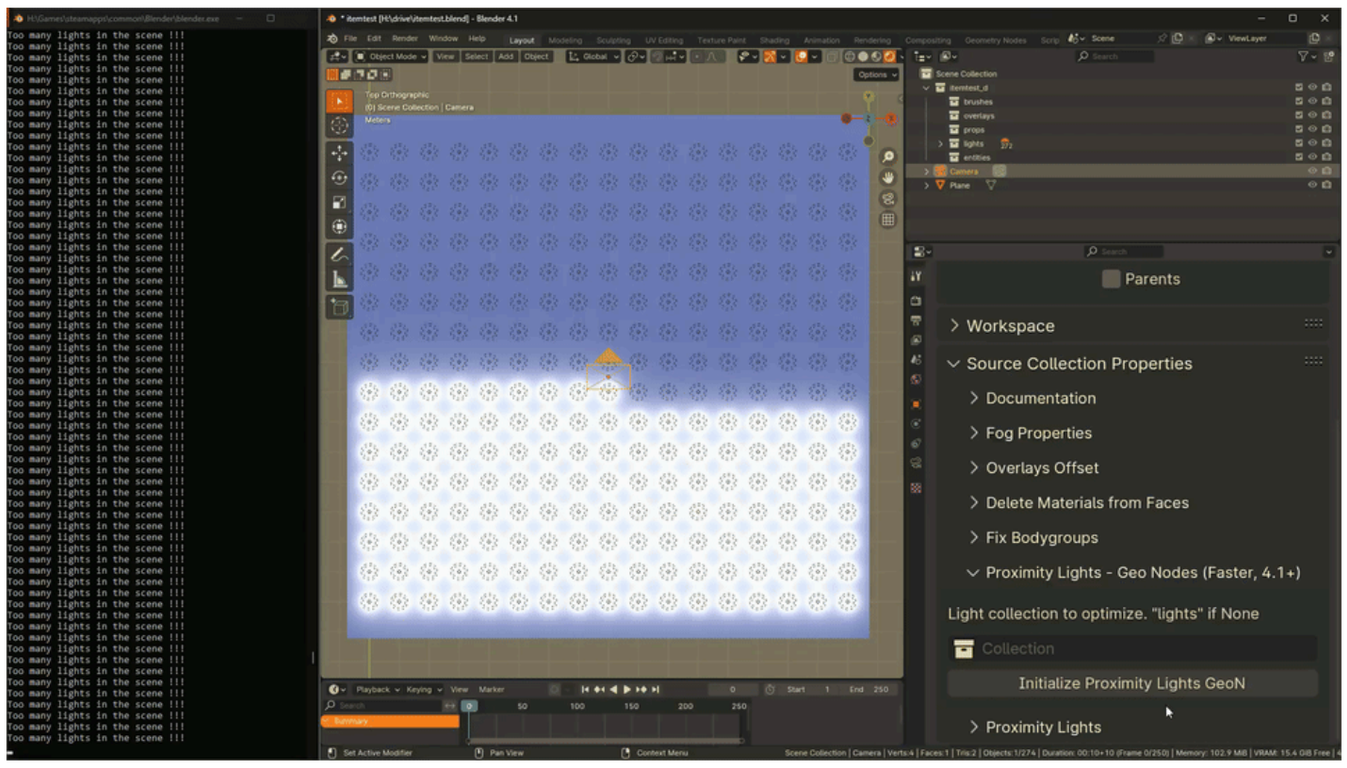
What's included?

- 40 Gb of content
- Models and materials from games, including ones embedded into maps
- Asset library support for models and materials
- Each model should have their included skins and animations
- Self-shadowing bump maps converted to Normal Maps
- Maps that have been thoroughly optimized
- Source-style ropes on maps
- HDRi's of skyboxes
- Correct skins for props on maps

▶ Comparison vs. Plumber & SourceIO

Map Extras

Proximity Lights



This feature is more useful to EEVEE.

EEVEE Legacy's light limit is hardcapped at 127 due to technical limitations. This is a problem, as most of the maps exceed that light limit. The solution to this is a node group that disables lights when too far from the active camera, a distance threshold determined by the user. This is updated real-time.

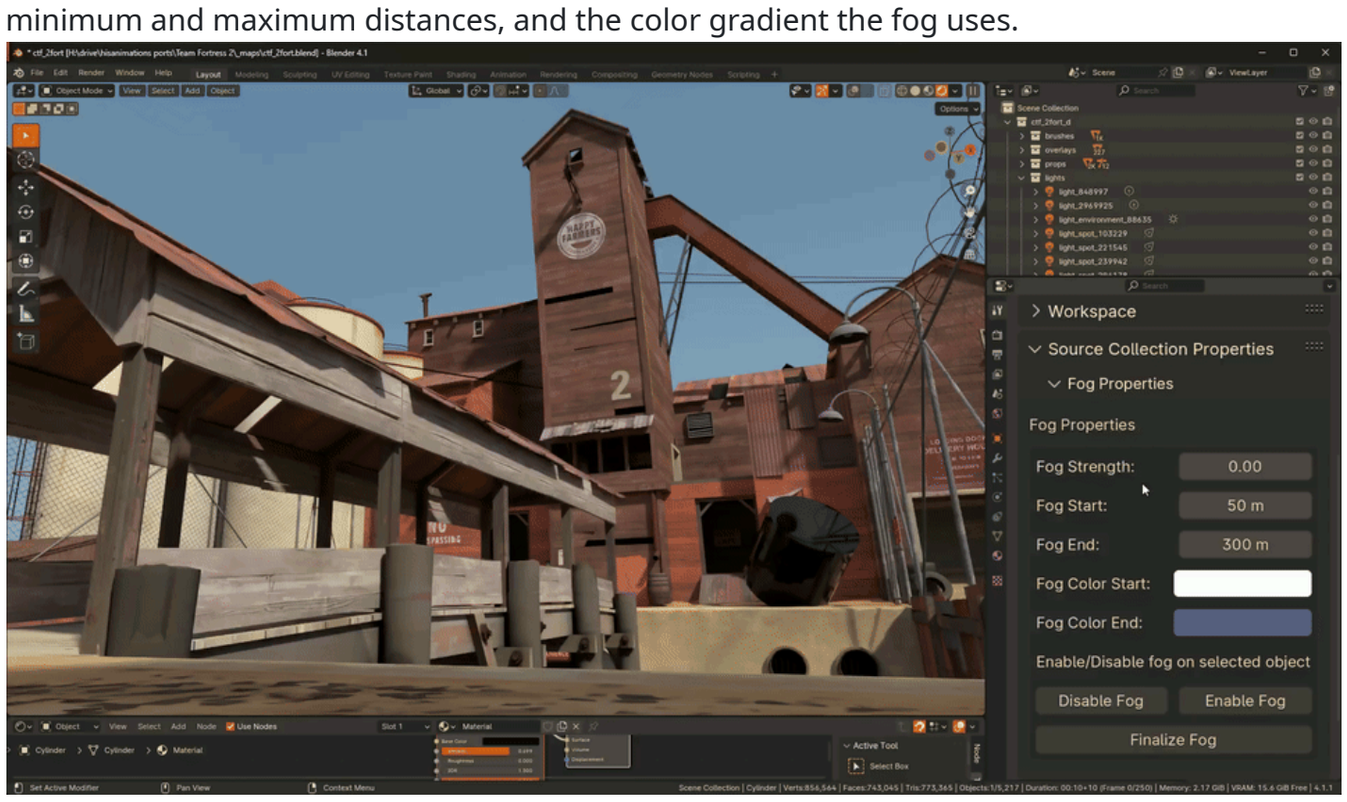
Method 1 (Faster, less flexible)

You can initialize the Proximity Lights node group by going to `Proximity Lights - Geo Nodes` in the Tools tab. Upon initializing, any sun lights in the light collection will be moved to a new collection to isolate them, and a proxy object for the optimized lights will be created. In the modifier panel of this proxy object, you can edit the parameters of the Proximity Lights node group, such as `Camera Frustrum` and `Light Culling`. Enabling `Camera Frustrum` will delete any lights not in view of the camera, and enabling `Light Culling` will set a hard cap on how many lights can be visible at once.

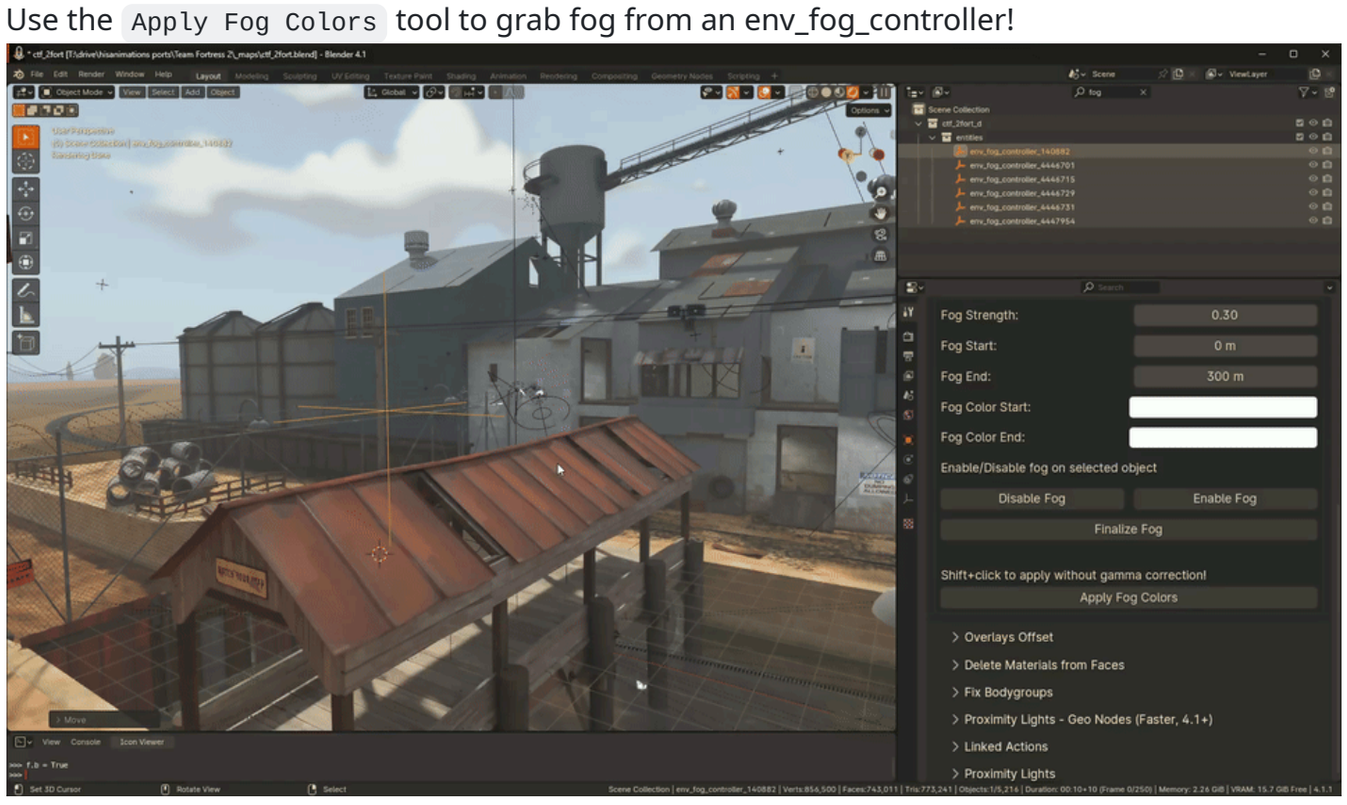
> **Note**
>
> It is worth noting that in the optimized state of Proximity Lights, you can not edit the lights in any way. To make changes, you will need to swap back to the unoptimized state of the lights in the Tools tab. Once changes are ready, swap back to the optimized state of the lights.

Fog

Tons of Source games use a fog effect in their gameplay, and fortunately that shader is easy to recreate. To mess around with the fog settings in a map, head over to the `Tools` tab and open up the `Fog Properties`. From there, you can adjust the fog strength, minimum and maximum distances, and the color gradient the fog uses.

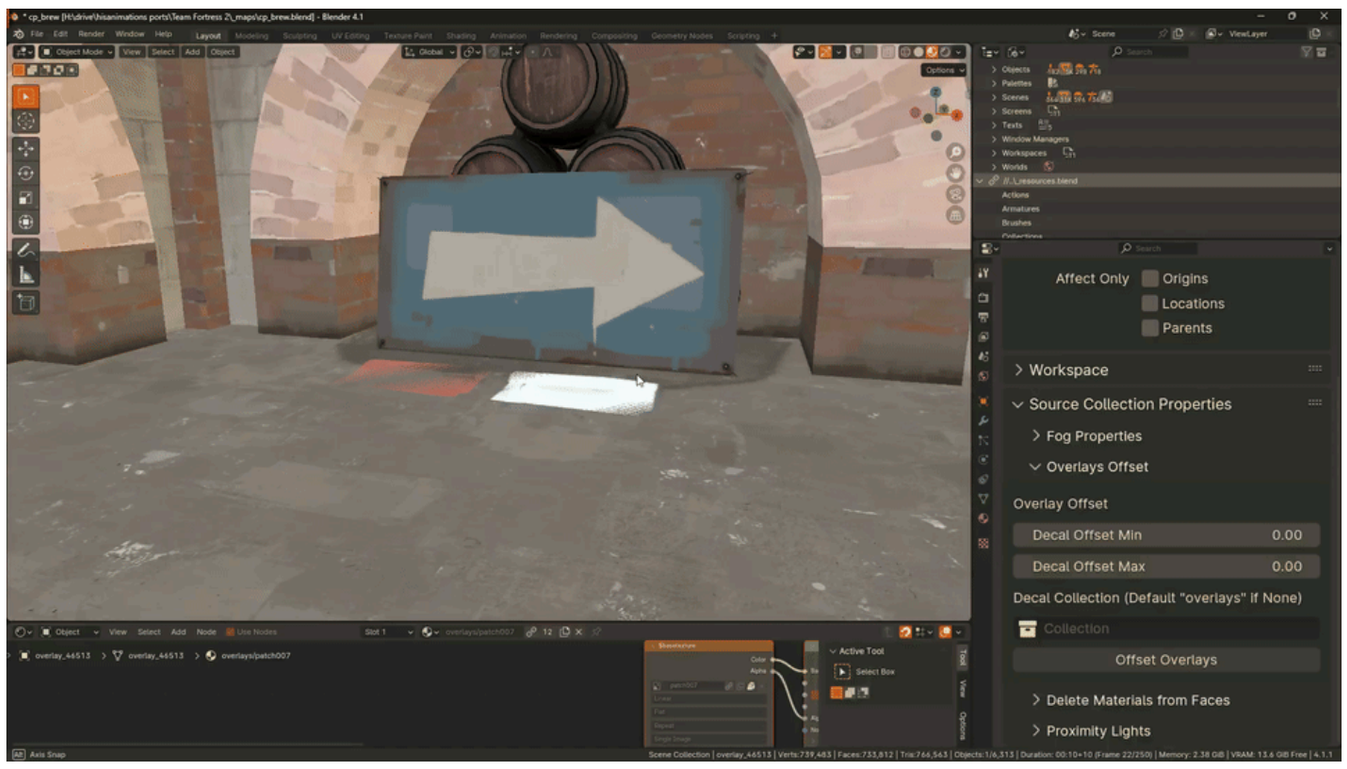


Use the `Apply Fog Colors` tool to grab fog from an env_fog_controller!



Overlays Offset

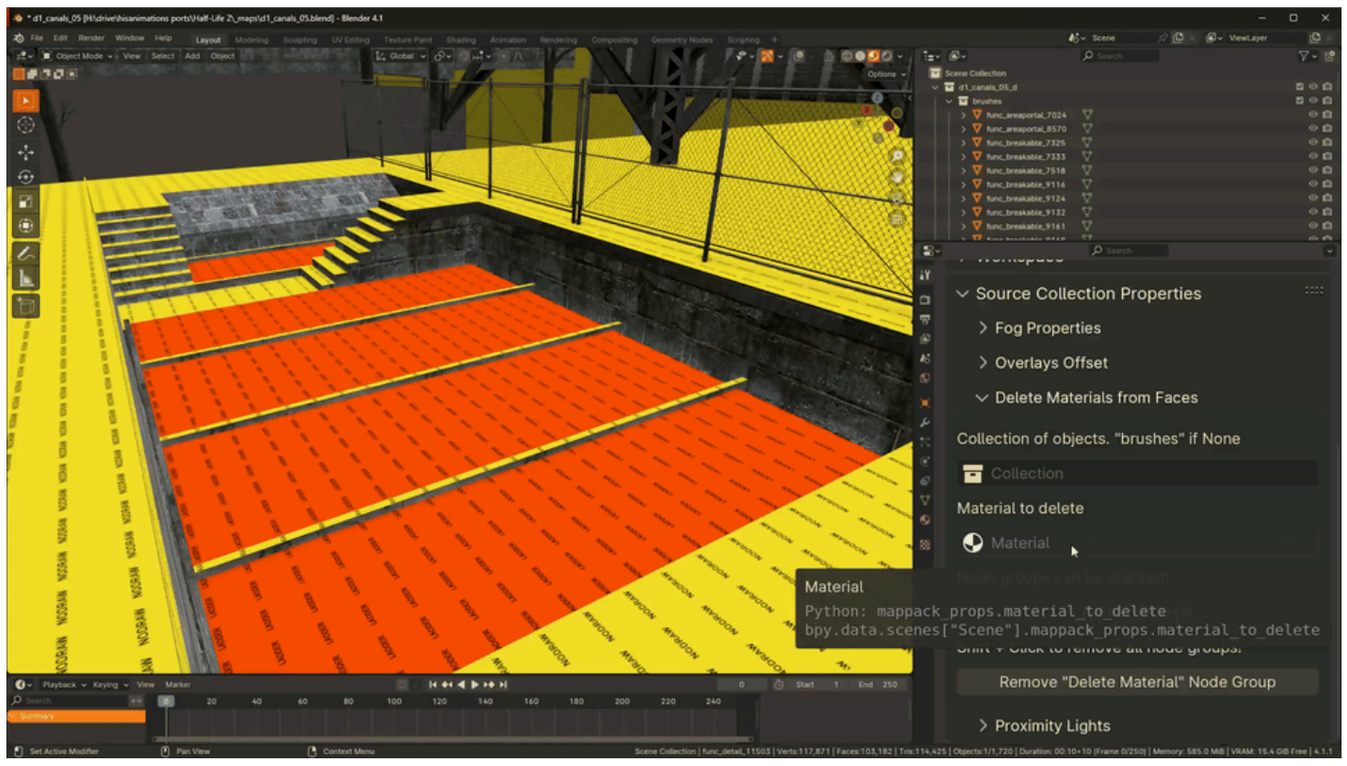
Overlays in the map usually have clipping issues, which can make a render look really ugly upon a render. In the `Map Extras` tab lies a panel named `Overlays Offset`. You can randomize the offset in case of overlays overlapping each other, or in case the overlays are clipping into a map. `Decal Collection` can be set for a custom set of meshes to fix. If left empty, it will fix all overlays in the `overlays` collection by default. When ready, click `Offset Overlays`. All overlays will be offset by their normal scaled by the inputted sizes.



This can also be used on the brushes collection to unclip brush geometry!

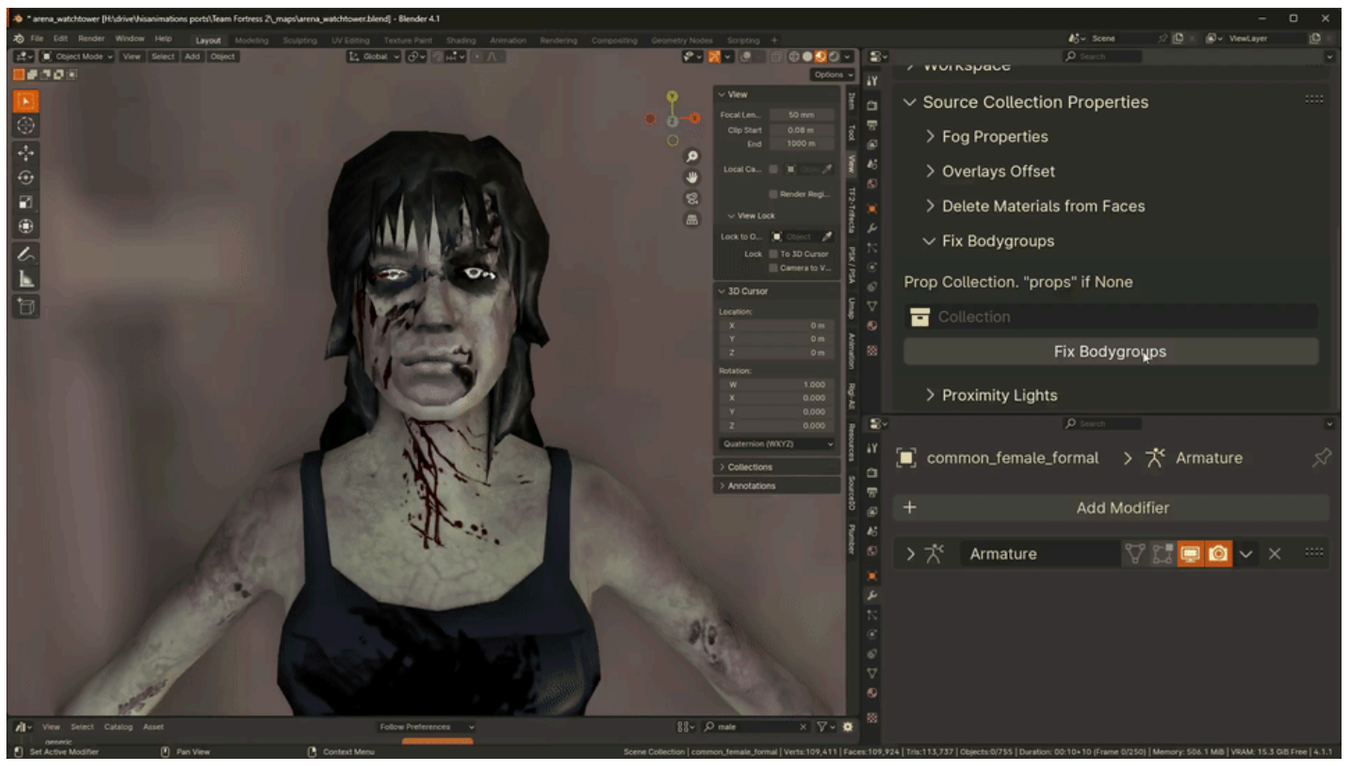
Delete Materials from Faces

Tons of unneeded materials such as `toolsnodraw` and `toolsblack` can pollute a file with unneeded geometry. Despite not being visible in EEVEE, they can produce visual errors in Cycles through Z-fighting. Deleting this geometry through a node group is a fast solution, and is applied to every object of a collection by batch. Set the material you wish to be deleted and click `Delete Material`. `Remove "Delete Material" Node Group` will delete every node group targeted the selected material. Shift clicking this operator will delete every node group regardless of material.



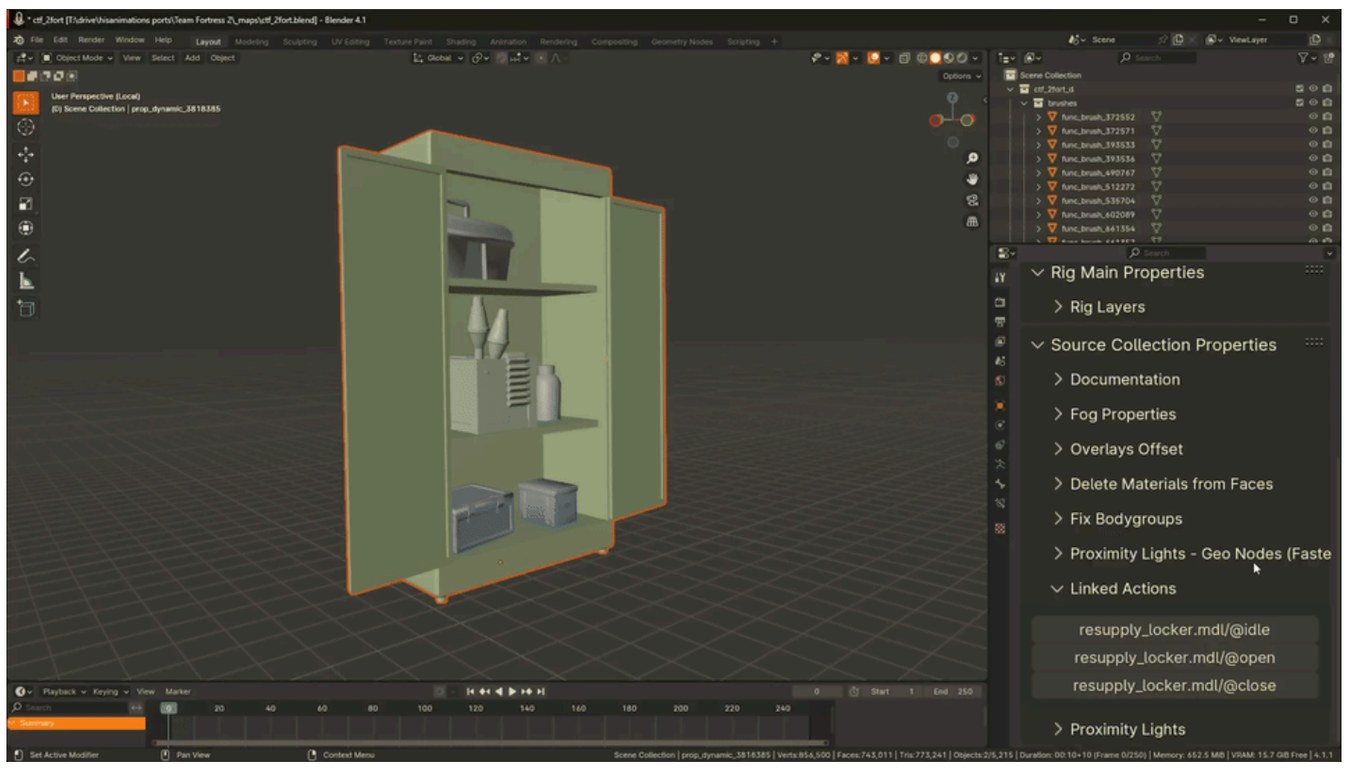
Fix Bodygroups

Some models have swappable bodygroups. If left untreated, these bodygroups can stack on top of each other and cause the model to appear not how it should. To fix this, you can use the `Fix Bodygroups` operator to adjust the model. The geometry nodes modifier that appears will not have a title for body groups, only indices. You'll have to go through trial and error to make it look how you want.



Linked Actions

If supported animations were able to be imported, they can be viewed and added as an NLA strip to the prop.



Extensions

Every good project should have a way for users to mod it, and so I introduce extension loader! A non-destructive way for users to add their own scripts. Upon loading a map, the .blend file will check if a folder named `_extensions` exists in the archive root directory. If it does, it will go through all .py files under the folder and load them. It should be structured like a regular Blender addon, sans any code executing a register function.

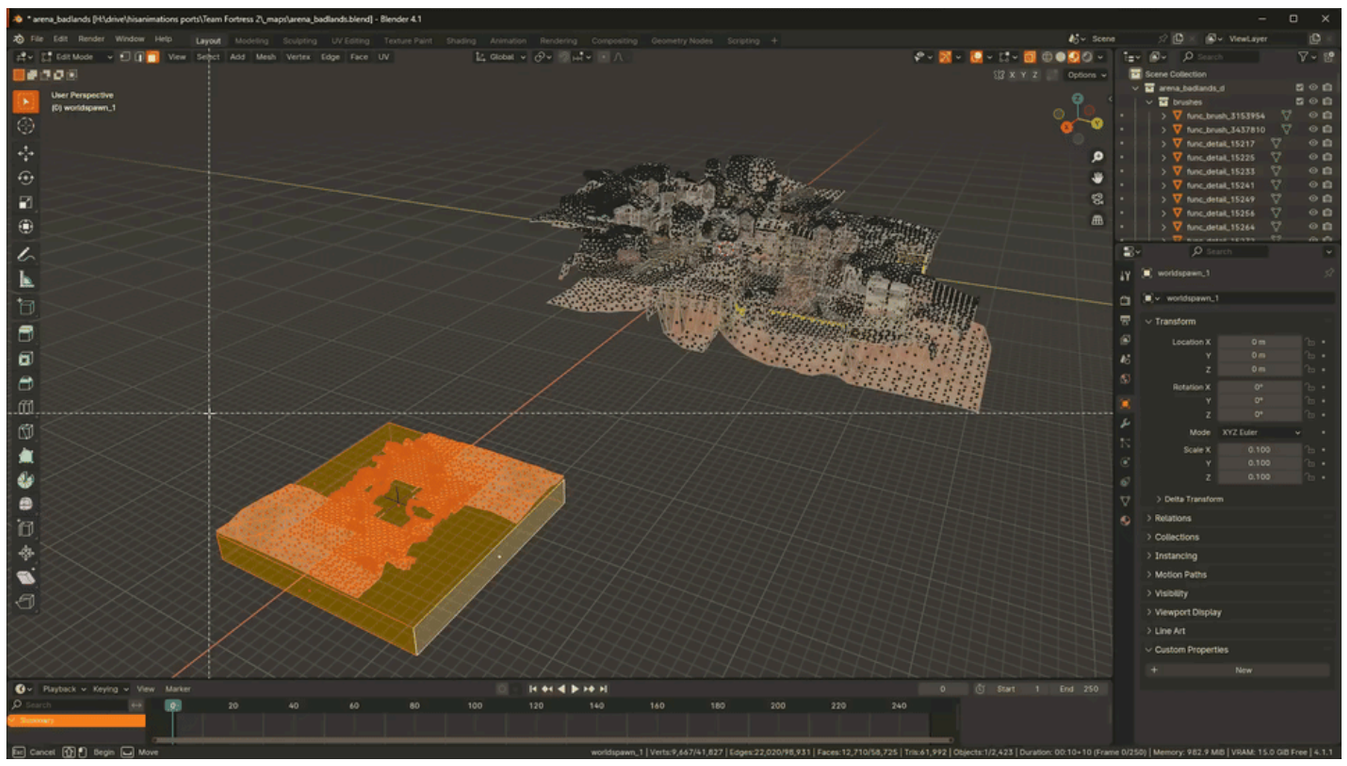
Example:

▶ Extension example

Tips

Fixing Skybox

Look for an empty object labelled "sky_camera." Around this empty should be lots of little pieces of meshes. Select the `sky_camera` object first, then box select (B) the surrounding objects. However, if you see the enitre map mesh is highlighted, then you need to enter `Edit Mode` on the world mesh labelled "worldspawn". Select the mesh surrounding the empty with the box select tool then separate it (P, Separate selection). Enter `Object Mode`, select the `sky_camera`, box select the surrounding mesh, then go to the `Object tab > Transform VMF 3D Sky`. If the entire map moves with the `sky_camera` object, that means you did not separate all of the world mesh around the empty, or you accidentally had it selected while transforming the 3D sky.



Here, I am separating the world mesh around the skybox by pressing P. Once separated, I select the `sky_camera` empty, then the surrounding mesh, then I go to `Object > Transform VMF 3D Sky`

Injecting Data

`_resources.blend` contains two major node groups named `Resources-ShaderN Container` and `Resources-GeoN Container`. These two main node groups contain every node group used by the materials, models, and maps. The main node groups can have more node groups added inside them. That way, any new node group will be available in any .blend file you open. This makes for extremely easy editing across .blend files, minimizing the amount that would alternatively be needed.

Adjusting Ropes

By default, the ropes are shown in Source-style, meaning they are flat but will always face the camera (4.1+ ONLY). To adjust the thickness and texture length of the rope, head to the modifier panel and adjust the effect there.

Editing Linked Assets

In the ported map files, you can open the .blend file of any linked model or material. You can find this under the `Linked Properties` panel of any mesh or material tab. Clicking the operator will open the .blend file, and Shift clicking will reload the .blend file to apply any changes. Make sure you save the file first!

If you wish to apply a temporary fix, then you may be trying to **localize** an asset. On the mesh or material, look for a chain icon next to its name. Clicking this chain will create a local copy of the asset, allowing you to make changes that won't be saved to the original file.

If you are unable to localize a material, node group, image, etc., that means it is only being used by linked assets. If you wish to make a local copy, it needs to have at least one user that is not linked. For example, if you are failing to localize a material, it is most likely due to its user (model/prop in most cases) not being localized. Localize the model first, then you should have no issues localizing the material.

Embedded Animations

Skeletal meshes will have a custom property named `["actions"]` in its mesh data containing a list of all animations that were able to be imported with the models.

Fixing Animations

When using animations outside of maps, skeletons and meshes will appear deformed. This is because the animations are not the same scale as the models outside of the maps,

Adjusting Sky Material

Some environment textures may appear super dark and saturated, appearing miscolored. The remedy is to adjust the sky texture's gamma with a `Gamma` node, and adjusting the brightness on the `Background` node. On the left, the gamma is unadjusted. On the right, the gamma is set to `0.196`. Adjust it to what feels right!

Lighting isn't the same

This mostly applies to EEVEE Legacy, as it has no native support for real-time global illumination. In general, the ambience of the map will never be 1:1 between Blender and TF2 no matter what render engine you use. Despite that, Cycles will always appear closer.

When baking a map, the HAMMER editor performs an effect like global illumination onto areas of the map to give the ambience a semi-realistic feel. It will then save that data to use in-game. For example, a light will light up most parts of the room because the light bounces everywhere. This data is not recoverable when porting a map. Therefore, using a map in EEVEE Legacy can make the ambience feel lackluster. There are two remedies for this solution:

Using Irradiance Volumes

Irradiance volumes will bake the ambience of its surroundings to then project off of its surroundings. This can be considered global illumination, but it is not real-time compared to cycles.

Altering World Shader

Depending where the camera is in a map, you can brighten or darken the world shader through the `TF2 Ambience` node through the `Color` and `Strength` values to whatever feels right at the time. These values can be keyframed.

Known Issues

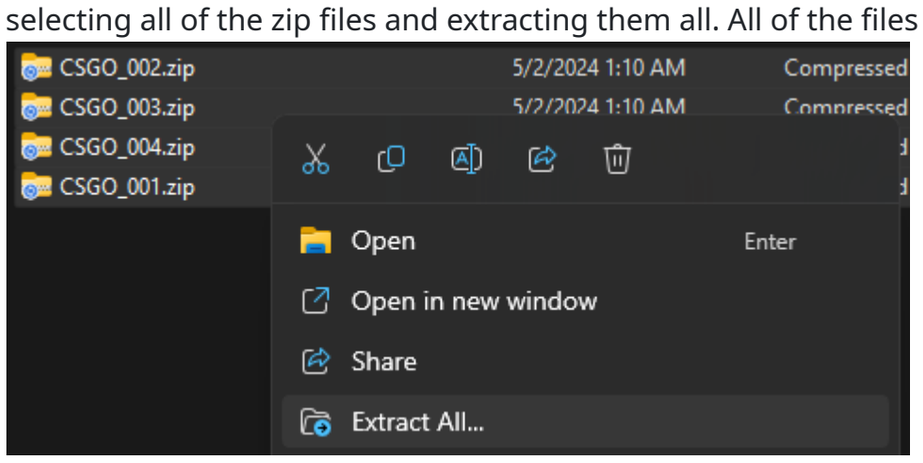
- No animations from `.ani` files
- Water materials are partially broken
- L4D2's infected shader is not perfect
- SolidEnergy materials are not supported

Installation Instructions

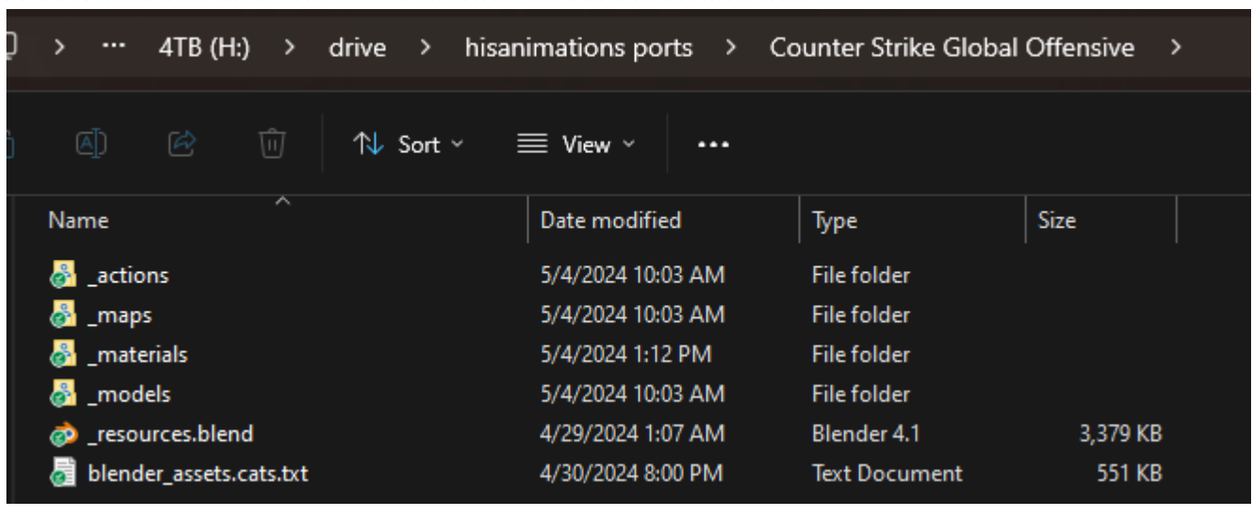
To install an archive of the Source Engine Blender Collection, you must first allocate a folder for a game collection. Then, download `_resources.blend` and `blender_assets.cats.txt`. `_resources.blend` is required regardless of what you decide to download. It serves as a resource pack for the models and materials to reuse data from. `blender_assets.cats.txt` is required for Blender's asset library functionality, allowing you to add the allocated folder as an asset library.

To download the actual content, download the `_materials` and `_models` folders. Download the `_actions` folder if you wish for animations. I don't recommend downloading the entirety of the maps at once, so just make a folder named `_maps` and download the maps you need or want when you need or want them.

If you plan to download all of the files at once, chances are the .zip files will be split into multiple pieces. Extracting it is a matter of selecting all of the zip files and extracting them all. All of the files will be merged into a complete piece.



A downloaded and extracted archive should look like this:



Depending on how you choose to download the archive, your installation may be different.

To add the archive as an asset library for Blender, head to `Edit > Preferences > File Paths > Asset Libraries` and add the archive folder.

Counter Strike: Global Offensive

A ported archive of CS:GO's assets, resulting in **23,504** models, **28,928** materials (19,078 usable out of asset library) and 89 maps. Totalling 18.2 GB. If you download all of this, I salute you.

Counter Strike: Source

A ported archive of CS:S's assets, resulting in 3,519 models, 9,034 materials (6,817 usable out of asset library) and 20 maps. Totaling 2.66 GB/

Day of Defeat: Source

A ported archive of Day of Defeat: Source's assets, resulting in 691 models, 1,673 materials (602 usable out of asset library) and nine maps. Totalling 839.84 MB.

Half-Life: 2

A ported archive of Half-Life: 2's assets, resulting in 3,217 models, 6,791 materials (4968 usable out of asset library) and 118 maps. Totalling 2.09 GB. Half-Life 2: Episode One and Two are included.

Left 4 Dead 1/2

A ported archive of Left 4 Dead 2's assets, resulting in 5,497 models, 8,165 materials (4,999 usable out of asset library) and 60 maps. Totalling 4.45 GB.

▶ Recreating L4D2's infected Shader

Credits

Syborg64 - Helped with infected shader
мяFunreal - Infected Shader guide

Portal

A ported archive of Portal's assets, resulting in 316 models, 504 materials (269 usable out of asset library) and 26 maps. Totalling 220.07 MB.

Portal 2

A ported archive of Portal 2's assets, resulting in 2,216 models, 3,629 materials (2,521 usable out of asset library) and 116 maps. Totalling 1.99 GB.

▶ Tips

▶ Effects

Issues

- Map's water materials are broken

Credits

Lil' Boneless Pizza - Helped with the Flowmap shader

Team Fortress 2

A ported archive of Team Fortress 2's assets, resulting in 8,297 models, 17,799 materials (10,472 usable out of asset library) and 185 maps. Totalling 9.94 GB

As per the request of a TF2Maps.net moderator, the maps and assets have been given credit. On each map with a credits section, a credits panel has been created to quote the names of the people who collaberated on the map. Also in the archive, there's a credits.txt file listing the .blend files used by the maps with credits details.

These creators should be credited.

Source to Blender Porting Tools - hisanimations

Porting Tools .zip
Modified Addons

This is a set of porting tools to build an asset library of all models and materials of a source game for Blender. This asset library will then be used for the maps to reuse data, significantly lowering the size of the maps. When all is said and done, you should have a complete archive of the game you are trying to port.

All tools for the .blend files reside in the `TOOLS` tab in Blender. Any tweaking done should only have to be done in there.

This process is only compatible with Blender 4.0. Don't worry, you can use this in later versions with no problems. Make sure you port EVERYTHING in Blender 4.0, and install the modified addons on 4.0. You can install Blender 4.0 Here

To start, you must first prepare a folder for the ports. Put all of the .blend files in the porting tools into this folder.

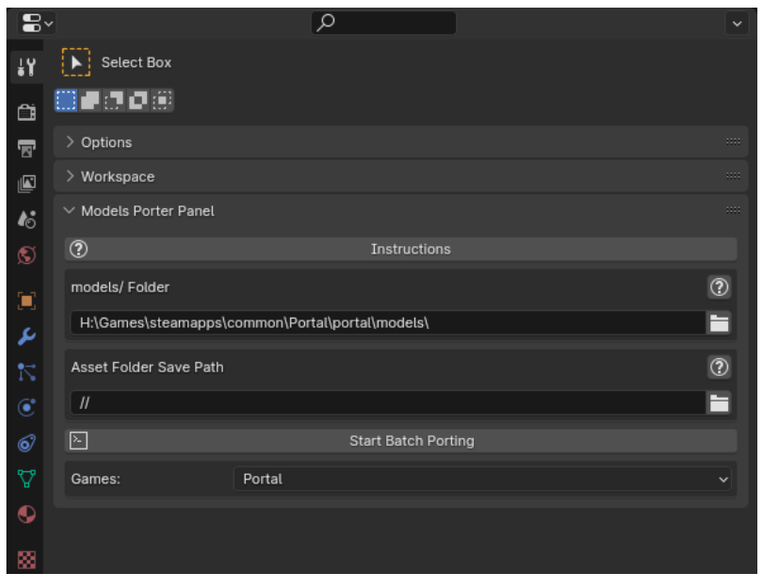
Prepare for all the assets you wish to port. Extract the models and materials from a game's `.vpk` archives and place them in the game folder. For example, to extract HL2's assets, you would extract the folders named `models` or `materials` and place them all under the `hl2` folder. You should have a folder structure that looks like `..\steamapps\common\Half-Life 2\hl2\materials` and `..\steamapps\common\Half-Life 2\hl2\models`, containing all of the extracted assets. For something like garry's mod, you should have something that looks like `..\steamapps\common\GarrysMod\garrysmod\materials` and `..\steamapps\common\GarrysMod\garrysmod\models`. This is where SourceIO and Plumber are going to look for assets.

Gather all the `.bsp` files you want to port and decompile them with bspsrc into `.vmf` files. They can be placed anywhere. In the `Other` tab of `bspsrc`, enable `Extract embedded files`. If you wish to add the embedded files as assets for the asset library, run this script. Set the `game` variable of the script to the game folder you are porting from, and set `root_dir` to the folder containing the `.vmf` files. Any detected model or material will be moved over to the game folder.

Once you have prepared all the assets you want to port, install hisanimations' versions of SourceIO and Plumber. If these addons already exist in your Blender's `addons` folder, rename them to keep a backup. Once you have installed the modified version of Plumber, make sure the game you want to port from is listed as a game directory.

Before you actually start porting any of the tools, I **HIGHLY** recommend that you keep the console open to view progress and check for any errors, such as whether textures or materials can be found or not.

_models_porter.blend (Step 1)



Open `_models_porter.blend` and set the `models/ Folder` path to the `models` folder you want to port from. Set `Asset Folder Save Path` to where you want the models to be saved. `//` means it will be saved alongside `_models_porter.blend` (recommended). Hit `Create Catalogs` for creating the asset library, then click `Start Batch Job` to port the models. Two files and two folders will be created. `_mapper_models.json`, `_mapper_materials.json`, `_models` and `_actions`. The two .json files serve as directions for porting the maps. They are used to check if an asset already exists, and if it does, it will return the `.blend` file it is located in and what it is known as. Hopefully within an hour, the porting tool will have ported all of the models and saved them as `.blend` files.

_materials_porter.blend (Step 2)

Open `_materials_porter.blend` and set the `materials/ Folder` path to the `materials` folder you want to port from. Set `Asset Folder Save Path` to where you want the models to be saved. `//` means it will be saved alongside `_materials_porter.blend` (recommended). Hit `Create Catalogs` for creating the asset library, then click `Start Batch Job` to port the materials. A folder named `_materials` will be created to store all of the materials. Eventually, the porting tool will have ported all of the materials and saved them as `.blend` files.

_maps_porter.blend (Step 3)

Set `.VMF Maps Folder` to the folder containing all of the decompiled maps. Set `.BSP Maps Path` to the folder containing all of the `.bsp` versions of the maps. Set `Asset Folder Save Path` to where you want the models to be saved. `//` means it will be saved alongside `_maps_porter.blend`. Set `Game Folder` to the game folder. This folder should have the `models` and `materials` folder under it. Set `Games:` to the game you are attempting to port from. Once ready, hit `Start Batch Porting`.

To-Do

- Port and rig SFM version of L4D2 survivors
- Port and rig Chell

Process

Going over my entire learning process this past month is just too much to write about, so I will summarize what I learned in the end.

Everything was ported in Blender 4.0 due to 4.1 causing massive breakage by changing the auto-smooth method. However, this grants native support for 4.1, and possibly support for 3.6.

Prerequisites

- Create a resource library containing every shader node group or geometry node group that they could need
  - These node groups are stored inside of two containers - one for Shader nodes and one for Geometry nodes. These two containers are linked to any ported material or model, that way I can add more inside of the container *later* and have it show up any time I open a .blend file using this resource library. This is known as indirect linking.
- Modify SourceIO
  - REDxEYE/SourceIO#287
- Modify Plumber
  - Make it export prop properties in the form of dictionaries
  - Make it export paths for models and materials

Porting Models

A special .blend file is created, having the two containers mentioned earlier already linked.

The tasks and process for porting the models are as follows in the form of steps:

Step 1 - Create catalogs

Recursively search through folders in which at least one .mdl file exists in. Catalogs are generated for these folders along with a unique UUID.

Step 2 - Porting `.mdl`s

I port all the .mdl files in the current folder of the recursion iteration one by one, twice! Once with SourceIO then once with Plumber, for animations. The end goal was to have one single mesh in the end, but because the imported .mdl could have multiple body groups, I had to mark them by their bodygroup index and their model index. This was necessary to have a way to separate them through a geometry node group. Once that's done, I merge them all into one object (if they weren't one object already) then link them to a new collection.

Next, I had to save the skin groups. Luckily, SourceIO already does that, but I need to update the skin group dictionary to make it more reliable. The dictionary only stores names of the skin groups, but it's better to have their actual data blocks instead.

Then I port the same .mdl file with Plumber, grab any animations and delete any new objects, then save the animations in a dictionary with the model it is for.

Finally, I store its true path as a custom property (directory + name) for identification purposes.

Step 3 - Mark as Asset

A dictionary is created to help find models, acting like a map. their TRUE path (directory + name), the .blend file they will be saved in, and their true name (how they are visually named in the .blend file). A second dictionary is also created, but only to help find models. That way when porting materials later, we can avoid porting the same one because it was already ported through the models.

The two containers mentioned earlier are attached to the mesh data of the model as a custom property.

Finally, they are marked as assets. Previews will be automatically generated, and their catalog ID will be set to the unique UUID generated for the folder that is being ported from.

Step 4 - Export

The two dictionaries that were created are dumped into a .json file. Then, any imported animations are dumped into another folder, then deleted in the current porting session. They are linked back to the session, then re-attached to the models via custom properties. This makes downloading animations optional, which can help you save space.

Once the mappers (dictionaries) and animations are dumped, the models are 100% ready to be dumped to another file. And so they are. We move onto the next folder, and the process begins anew.

Porting Materials

A special .blend file is created, having the two containers mentioned earlier already linked.

Step 1 - Create catalogs

Recursively search through folders in which at least one .vmt file exists in. Catalogs are generated for these folders along with a unique UUID.

Step 2 - Porting `.vmt`s

I port all the .vmt files in the current folder of the recursion iteration one by one.

Step 3 - Mark as Asset

Using the pre-existing materials mapper created by the models porter, I store the material's true path, the .blend file it will be saved in, and its true name (how they are visually named in the .blend file).

The two containers mentioned earlier are attached the material as a custom property.

Finally, they are marked as assets. Previews will be automatically generated, and their catalog ID will be set to the unique UUID generated for the folder that is being ported from.

Step 4 - Export

The materials mapper dictionary is dumped to a .json file. Then, the materials are dumped to a .blend file. We move onto the next folder, and the process begins anew.

Porting Maps

Step 1 - Port the Maps

Import the map with Plumber in `.vmf` format, then again with SourceIO in `.bsp` format. Plumber is better in general, as it cleans up the map a way a user would generally want it. It also has better support for lighting. SourceIO is used to import the ropes. The ropes are modified with a geometry node group to make them source-styled.

Step 2 - Link from Existing Assets

Once ported, the linking/reusing process begins. It will first go through every object under the `brushes` and `overlays` collection to compile a list of materials that need to be ported using the materials mapper. Once the list is complete, it will go through every mentioned .blend file and grab every material in the list under the .blend file. The objects' current materials are then replaced with the linked materials.

The same thing happens again, this time with the models. Once they have all been ported, they are adjusted for skins.

Step 3 - Remove "rendermode" "10" Brushes

Brushes set to "rendermode" "10" are invisible in-game, so it makes no sense to keep them in the final port. The `.VMF` file is open and read in reversed to create a list of brush entities to delete if their rendermode is set to 10.

Step 4 - Final Touches

- Ambient occlusion is added.
- The world shader is adjusted to use a solid color of ambience instead of relying on the skybox, while still showing the skybox.
- The skybox is exported as an .exr file

Step 5 - Save

Once the map has been properly ported, a collection of data is saved as a .json file before finally saving the map:

- Start Size
- Final Size
- Object Count
- Light Count
- Port Time
- Used Assets

The map is finally saved, the .blend file template is re-opened, and we move onto the next map.

Donate

If you'd like to donate through Ko-Fi that'd be great! The money would go towards expanding my drive storage, allowing me to do more ports to share.

Credits

Plumber - Maps, animations
SourceIO - Models, materials, ropes
Syborg64 - Helped with infected shader
мяFunreal - Infected Shader guide
Lil' Boneless Pizza - Helped with the Flowmap shader, made a few renders for the background

To the authors of SourceIO and Plumber:
Working on this project has allowed me greater insight to what goes on in your addons, and it just blows my mind how much dedication you all bring to the table. I am so grateful for your contributions. Without you, none of this would have been possible.

Plumber by lasa01, SourceIO by REDxEYE

Changelog

▶ Changelog

Disclaimer

I do not own, nor do I claim to own any ported asset that is hosted on my Google Drive. I am simply a middle man to help people get what they want. Everything is non-profit, and not blocked by any form of paywall.

All rights go to the creators of the original models.