

# Asset Import / Export

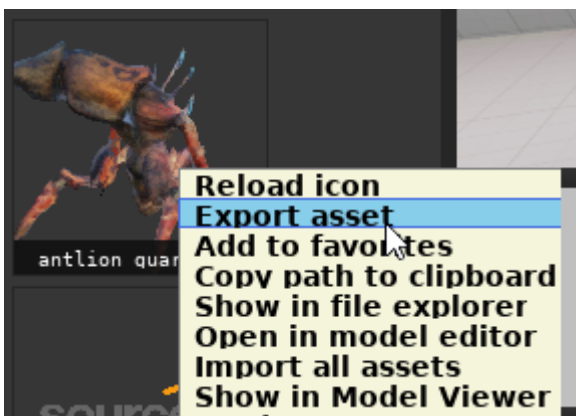
## Export

Pragma comes with the following export capabilities:

- You can use Pragma to export Source Engine and Source 2 Engine assets for use in modelling programs
- Assets in Pragma can be exported in the Source Engine model/material/texture format
  - Fake PBR can be applied automatically for use in SFM
- Automatic generation of ambient occlusion maps (if they don't exist)
- Automatic re-scaling of the models to meters
- Conversion to match Blender's coordinate system
- Includes the rig, all of the animations and morph targets of the model
- Conversion to standard normal maps (e.g. if self-shadowed bumpmaps are used)
- Conversion of Source 1 assets with PBR settings
- Automatic conversion of the textures to png, bmp, tga, jpg, hdr, dds, ktx or vtf
- Automatic conversion of the textures to a RMA format
- Works with maps / level geometry

## Exporting Assets

If you want to export a model or material, the easiest way to do so is via the Filmmaker interface. Simply open the model/material catalog, right click on the model icon, then click "Export asset":



If you want to export a different asset type, or need more control, you can use the export command utility instead. To use it, start a game in Pragma first. You can either load a map (which map doesn't matter), or run "map empty" in the console. Once the map has been loaded, you can use the "util\_export\_asset" console command with the following parameters:

- **-model** <modelName>: Exports the specified model. If a directory is specified, all models within that directory will be exported.
- **-texture** <textureName>: Exports the specified texture(s). If a directory is specified, all textures within that directory will be exported.
- **-material** <materialName>: Exports the specified material(s). If a directory is specified, all materials within that directory will be exported.
- **-map** <mapName>: Exports the specified map(s). If a directory is specified, all maps within that directory will be exported.
- **-export\_images** 1/0: If enabled, all textures of the asset will be converted to the specified image output format and saved alongside the asset. Default: 1
- **-normalize\_texture\_names** 1/0: If enabled, the exported textures will be renamed according to their type (e.g. "textureName\_normal") and will not keep their original name. Default: 0
- **-image\_format** png/bmp/tga/jpg/hdr/dds/ktx: The format to use for exported textures. Default: dds
- **-binary** 1/0: If enabled, the model will be exported in the "glb" format, otherwise "glTF". Default: 0
- **-verbose** 1/0: If enabled, additional information will be printed to the console during the export. Default: 1
- **-recursive** 1/0: If enabled and a directory is specified as the asset, all assets from all sub-directories will be exported as well. Default: 0

Models only:

- **-export\_animations** 1/0: If enabled, all animations of the model will be exported as well. This may drastically increase the file size and loading times when importing the model into a modelling program. Default: 1
- **-export\_morph\_targets** 1/0: If enabled, will export all morph targets (flexes) of the model as well. Default: 1
- **-export\_skinned\_mesh\_data** 1/0: If disabled, no skeleton, vertex weights or animations will be exported. Default: 1
- **-embed\_animations** 1/0: If enabled, all of the animations of the model will be embedded in the exported glTF file. When disabled, each animation will be exported as a separate file instead. Default: 1
- **-full\_export** 1/0: If enabled, body groups of the model will be exported as well. Default: 0
- **-scale** <scale>: The value by which to scale the exported meshes. Use 1 to keep the original scale. Use 0.025 to convert the meshes to meters. Default: 0.025
- **-generate\_ao** 1/0: If enabled, will automatically generate ambient occlusion maps for the model if it doesn't have any. Default: 0
- **-ao\_resolution** <resolution>: The resolution to use for the ambient occlusion maps if "generate\_ao" is enabled. Default: 512
- **-ao\_samples** <samples>: The number of samples to use for generating the ambient occlusion maps if "generate\_ao" is enabled. Default: 40
- **-ao\_device** cpu/gpu: Whether to use the CPU or GPU to generate the ambient occlusion maps if "generate\_ao" is enabled. Default: cpu

- **-merge\_meshes\_by\_material** *1/0*: If enabled, all meshes that use the same material will be merged. Default: 1
- **-enable\_extended\_dds** *1/0*: If enabled and the image format is set to dds, will enable support for BC6 and BC7 compression using the DXT10 version of the format. Default: 0
- **-list\_animations** *1/0*: If enabled, the model will not be exported and all available animations of the model will be listed in the console instead. Default: 0
- **-animation** *<animName>*: If specified, only this animation will be exported.
- **-format** *glTF/mdl*: The output format for the model. If "mdl" is specified TODO
- **-game** *1/0*: Only has an effect if "format" is set to "mdl". TODO
- **-preview** *1/0*: If enabled, the model will be shown in the viewport. Default: 0

If you want to use Pragma to export Source Engine assets, you generally don't have to copy any of the assets to Pragma, the Engine should be able to locate the assets automatically as long as they're part of one of your installed Source Engine games on Steam.

## Usage Examples

Exporting a material (including textures):

```
util_export_asset -material "brick/brick_1a" -verbose -image_format png
```

Exporting a texture:

```
util_export_asset -texture "concrete/concrete_5_floor_3_psd_918fcd2d" -verbose -image_format png
```

Exporting a map:

```
util_export_asset -map "gm_construct" -verbose -export_images 1 -image_format png -generate_ao 0 -  
export_animations 0
```

Exporting a model:

```
util_export_asset -model "props_2fort/tank001" -verbose -image_format png -generate_ao 0
```

(Make sure to remove the ".mdl" extension!)

Exporting a Source 2 model:

```
util_export_asset -model "characters/gman/gman" -verbose -image_format png -generate_ao 0
```

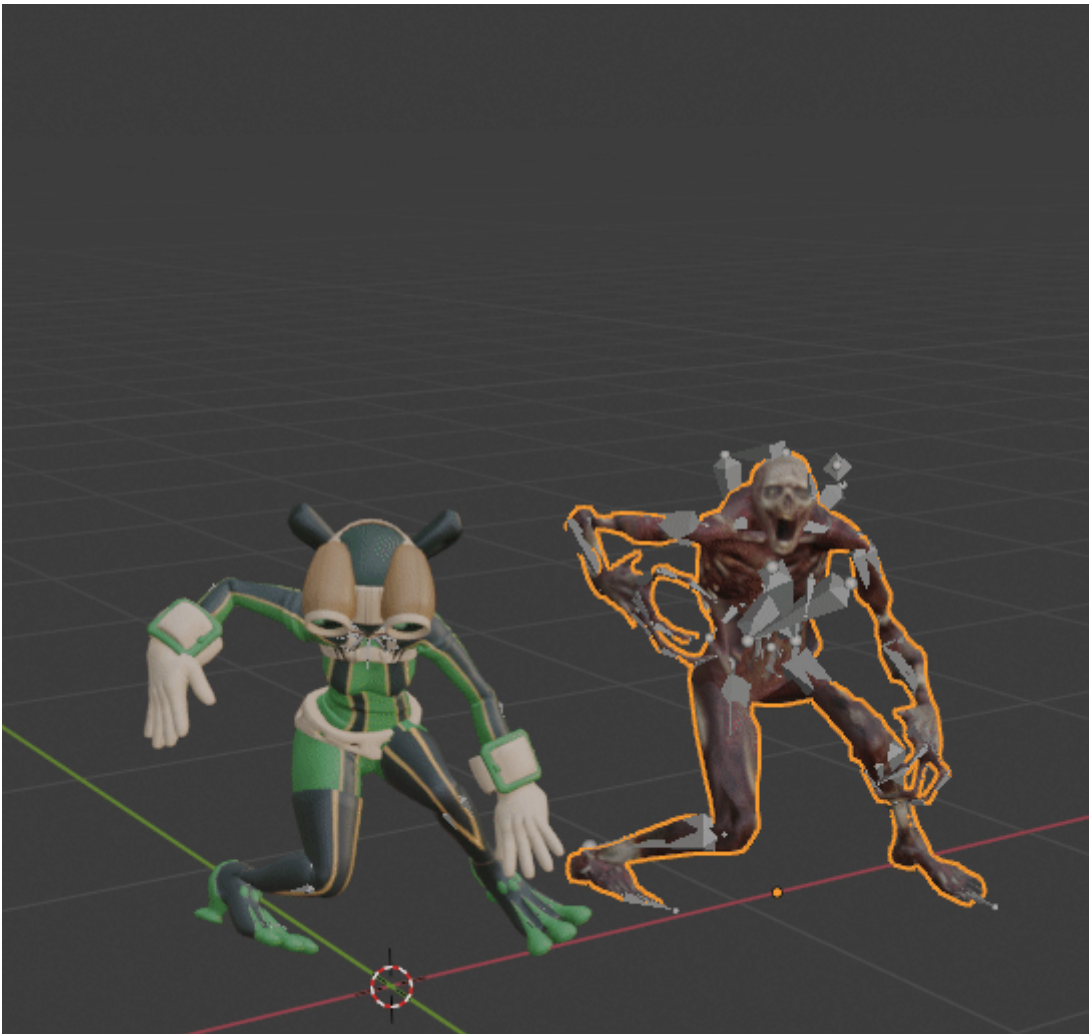
Batch-exporting all models in a directory and all sub-directories:

```
util_export_asset -model "props_2fort" -verbose -image_format png -generate_ao 0 -recursive 1
```

## Exporting retargeted models

This section refers to PFM v0.4.3 and newer and may not be representative of older versions.

It is also possible to export a model with retargeted animations (i.e. export a model with the animations of another model.):



To do so, you will have to set up a retarget rig in PFM's retarget editor first:

<https://www.youtube-nocookie.com/embed/2f-pxcjbanA>

Once you have created the retarget rig, you can use the `-retarget_source` argument in the `util_export_asset` command to export the model with the retargeted animations:

```
util_export_asset -model "e3assassin" -retarget_source "re5/jv1" -verbose -image_format tga -generate_ao 0 -  
embed_animations 1
```

## Exporting to Source Engine

You can also use the command to export a model **to** the Source Engine .mdl format. To do so, simply set the "format" parameter to "mdl":

```
util_export_asset -model "props_2fort/tank001" -format "mdl"
```

Most of the parameters listed above will have no effect when using the Source Engine mdl-exporter!

This will automatically generate the QC and SMD files for the model, convert the textures to vtf, convert the materials to vmt, as well as automatically compile the model using studiomdl. Pragma will attempt to locate studiomdl automatically, but you can also specify which game's studiomdl should be used:

```
util_export_asset -model "props_2fort/tank001" -format "mdl" -game "tf2"
```

This will use the studiomdl from TF2. The game name has to match the entry from "Pragma/cfg/mounted\_games.udm" (You can open the file in a text-editor).

If the model's materials use PBR, the exporter will also automatically apply fake PBR for use in SFM.

All of the model's animations will be included in the Source Engine model as well, however morph targets are currently not supported.

## Using Pragma to convert models from Blender to Source

Converting a model from Blender (or other modelling programs) is very simple. All you have to do is export your model from Blender in either the glTF format, or the glb format and copy the exported file to "Pragma/addons/imported/models". Then use the command above and replace the model name with the name of the glTF/glb file (without the extension).

## Source 2

The approach for Source 2 assets is exactly the same as for Source 1 assets, with the exception of maps. To export a Source 2 map, follow these steps (using Half-Life: Alyx as an example):

1. Open the map's vpk-file (steamapps/common/Half-Life Alyx/game/hlvr/maps) in [GCFscape](#)
2. Extract all of the files from the vpk to "Pragma/addons/imported/"
3. Run the "util\_export\_asset" command with the asset name pointing to the "vwrl\_d\_c"-file.  
For example, for the map "a5\_vault", you should use:

```
util_export_asset -map "a5_vault/world" -verbose -export_images 1 -image_format png -generate_ao 0
```

# Import

If you want to import a model from Blender (or other modelling programs), all you have to do is export the model in the glTF or glb format and copy the exported file to "Pragma/addons/imported/models". The model should automatically show up in the PFM model catalog and it will automatically get converted (along with the textures) when used.

## Source Engine / Source 2

If you want to import a model (or material/texture) from Source or Source 2, you generally don't have to do anything. All of your installed Source Engine games (assuming they're installed via Steam) should be automatically detected and mounted by Pragma. In the rare event that this doesn't work, you can try adding the mount path for the game to "Pragma/cfg/mounted\_games.udm".

Alternatively you can also simply extract the asset files to "Pragma/addons/imported/models" and "Pragma/addons/imported/materials" respectively.

---

Revision #17

Created 6 April 2021 16:32:48 by Silverlan

Updated 16 July 2023 06:17:24 by Silverlan