Converting a Textured Unreal Engine '.uasset' Mesh to a Source Engine '.mdl' Model

About this Guide and Disclaimer

This guide was written mostly for myself so that I can repeat this process more easily in the future. I am no 3D artist and this conversion was my most extensive use of Blender ever. I know very little about the process and assume a lot so some things I explain may be unnecessary or just plain wrong.

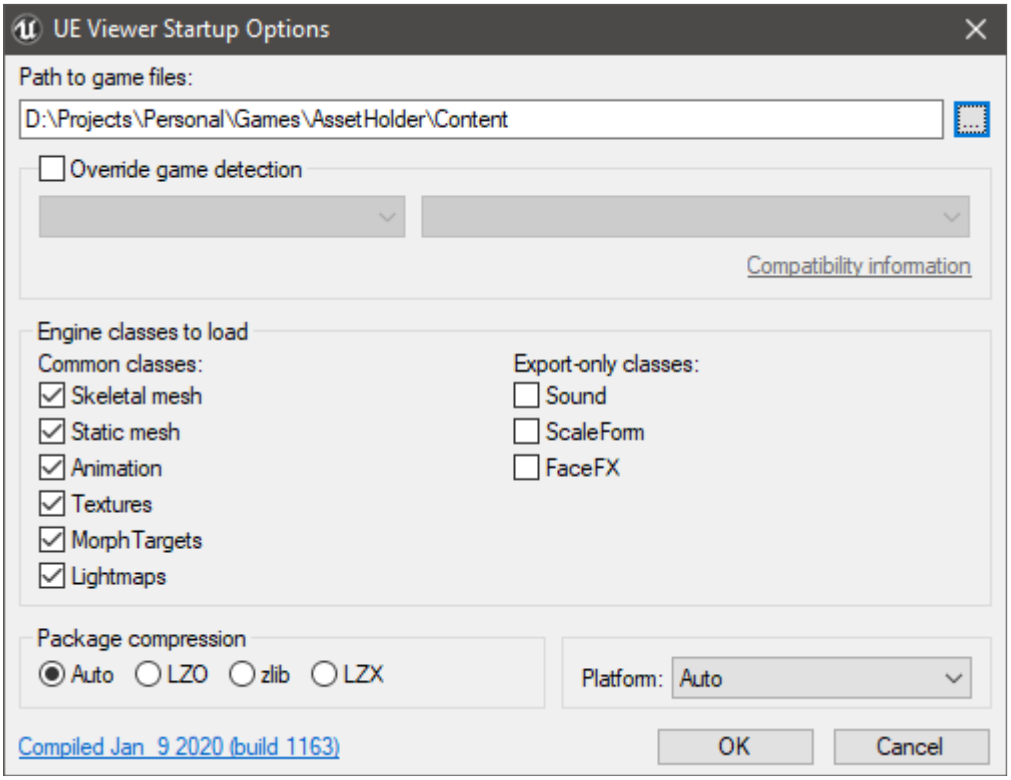
Author: Lutt.online (GitHub/luttje)

Table of Contents

Author: Lutt.online (GitHub/luttje)

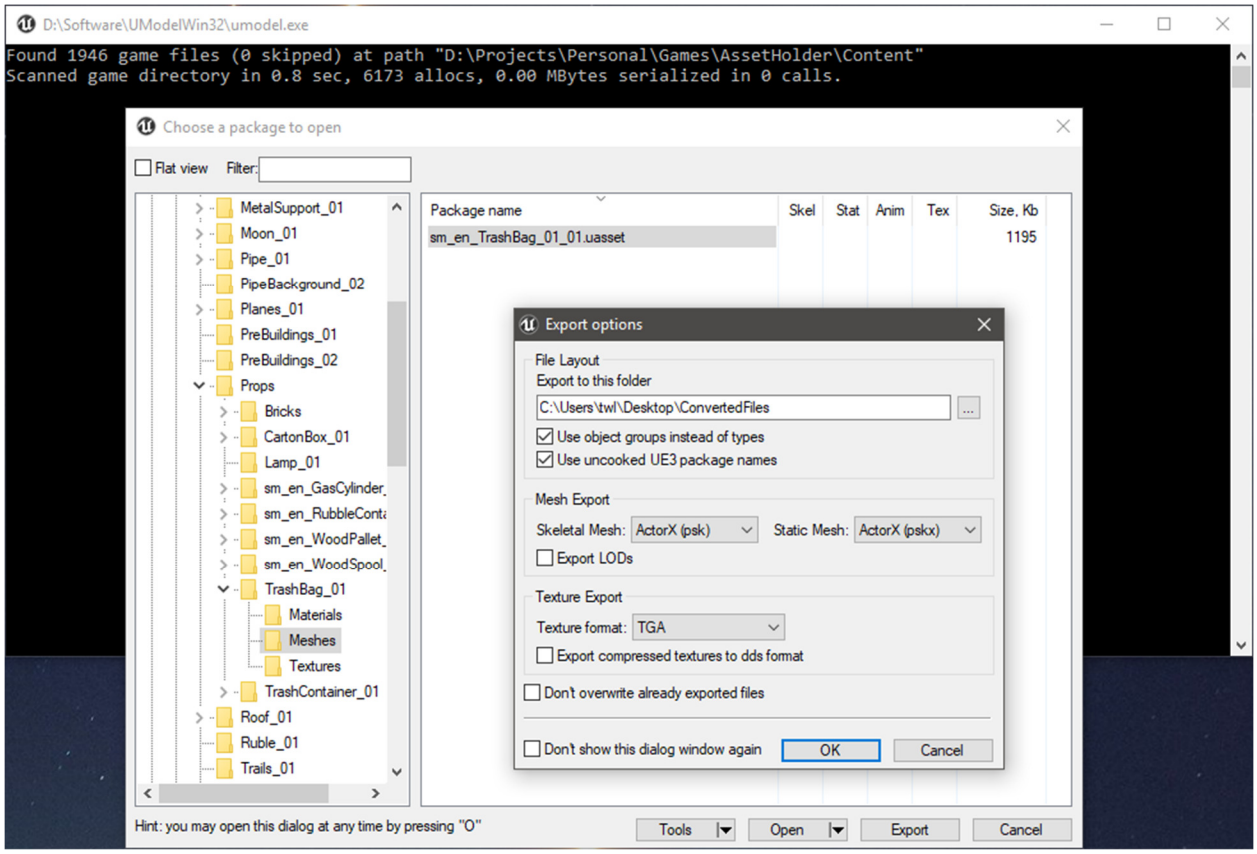
First Step: Exporting the uasset Meshes to psk(x) and Textures to tga

1. Download UE Viewer/UModel from here: https://www.gildor.org/en/projects/umodel#files
   (Files also included next to this document as 'UModelWin32.zip'
2. Extract to a folder on your PC
3. Open the umodel.exe (a Console Window and Configuration will open:)



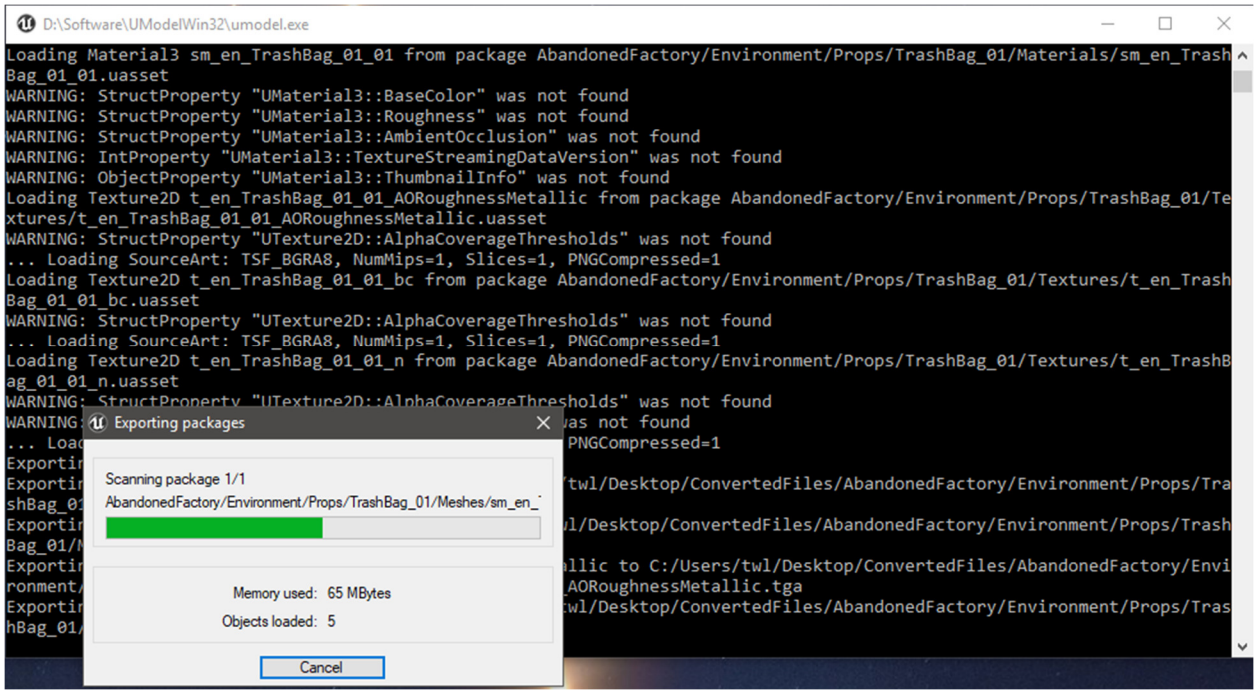
4. Under 'Path to game files' select where your uassets are.
   (I have loaded my assets into an empty project and select that projects' Content-folder)
5. Leave all default settings and click 'OK'
6. In the left File Tree navigate to the file you would like to convert
7. Click on 'Export' and put the settings to this:

Author: Lutt.online (GitHub/luttje)



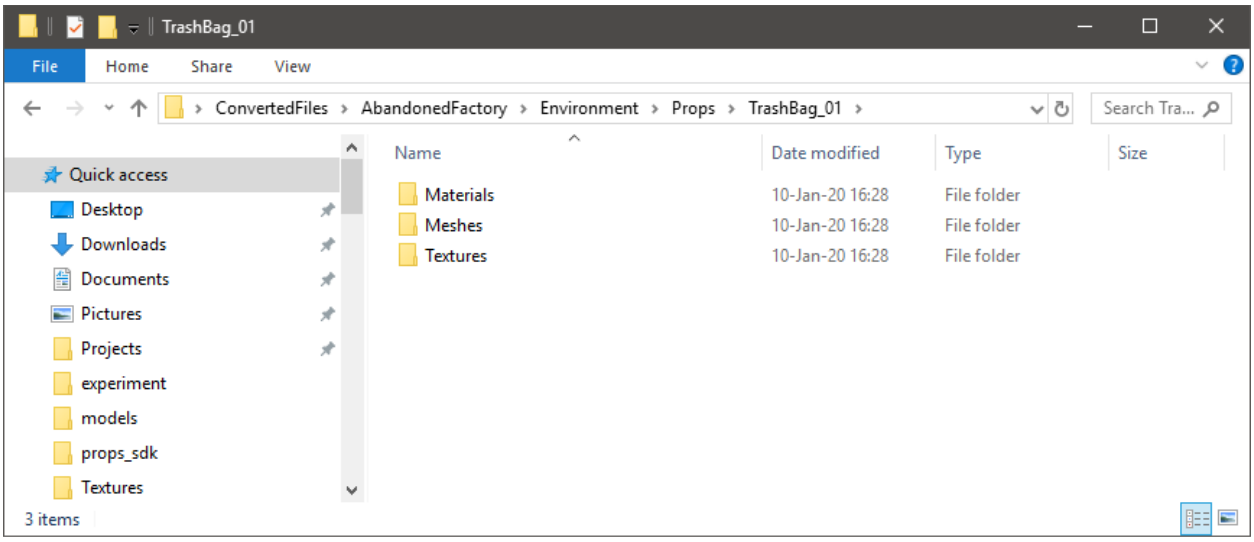
(I am unsure what the 'Use object groups instead of types' and 'Use uncooked UE3 package names' settings do, but they worked for me)

8. Click 'OK' to start the export process



9. Open Windows File Explorer and you will see that the export has given you a structure of folders:

Author: Lutt.online (GitHub/luttje)



Step Two: Import the .pskx Mesh in Blender

(Required only once) Download and install Blender

Blender is a tool for 3D Artists. We will need Blender to import the psk file, apply the textures and then export it for Source Engine.

1. Download and install Blender from: https://www.blender.org/download/
   (The contents of this guide are based on testing with Blender v2.80.75)
2. Start Blender
3. On the right-hand side delete the default 'Cube' that is placed into the scene.

(Required only once) Installing the .psk importer for Blender

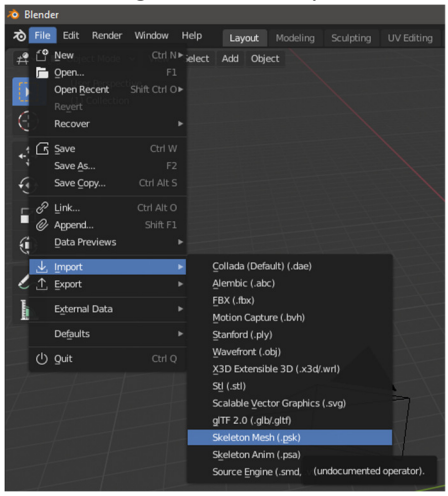
Befzz @ GitHub has created an Add-on for Blender that allows you to import the earlier .pskx file.

1. Download the latest version of the Add-on here:
   https://github.com/Befzz/blender3d_import_psk_psa
   (Make sure you download it for your version of Blender, you can find your Blender version in the bottom right corner of Blender)
   For Blender 2.80 versions the Add-on script has been included with this guide as 'io_import_scene_unreal_psa_psk_280.py'
2. Install the Add-on by:
   a. In Blender go to Edit -> Preferences... -> Add-ons -> Install...
   b. Navigate to the just downloaded scripts' .py file and double-click it
   c. Select the checkbox in front of 'Import-Export: Import Unreal Skeleton (.psk)/Animation Set (.psa)'
   d. Give it some time to load the Add-on

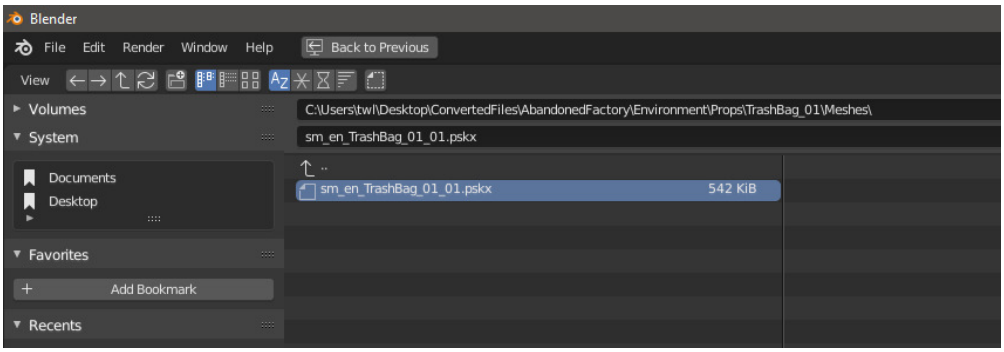
Author: Lutt.online (GitHub/luttje)

Importing the .pskx Mesh into Blender

1. In Blender go to File -> Import -> Skeleton Mesh (.psk)

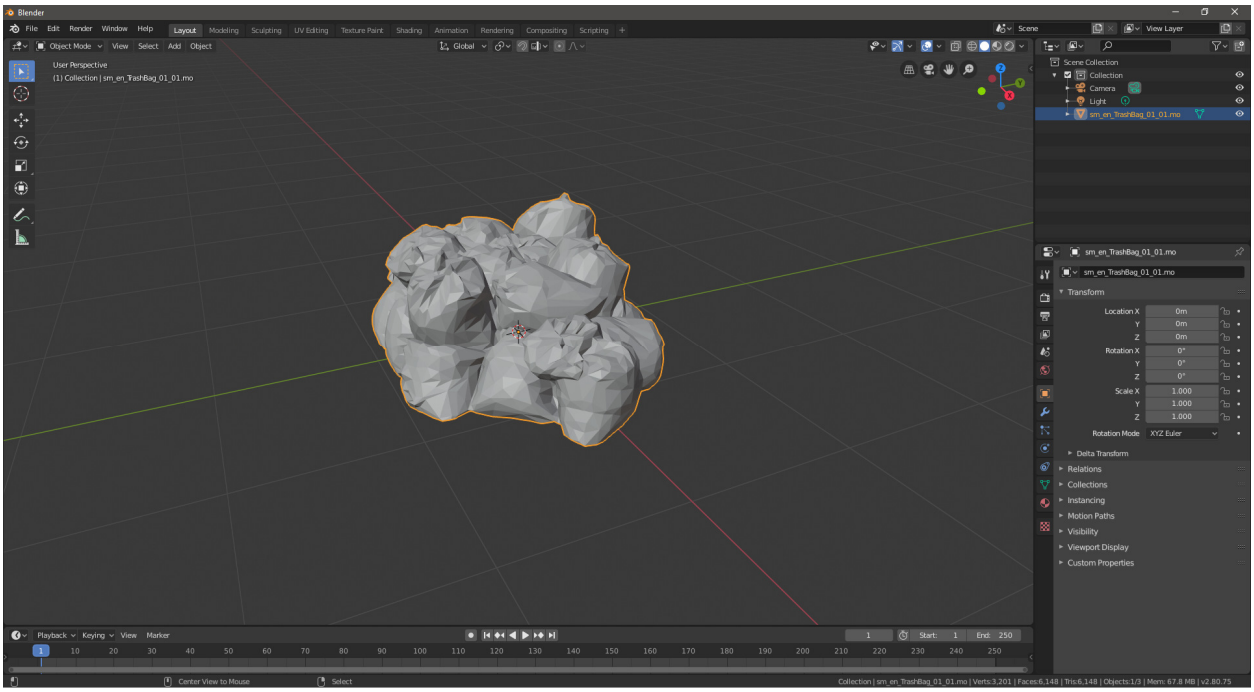


2. Navigate to where the .pskx-file is and double-click it:



3. You should see the Mesh (from now on we'll call it a 'model') you want to convert to .mdl appear in the ViewPort:
   (In this case we're converting a bunch of trashbags)

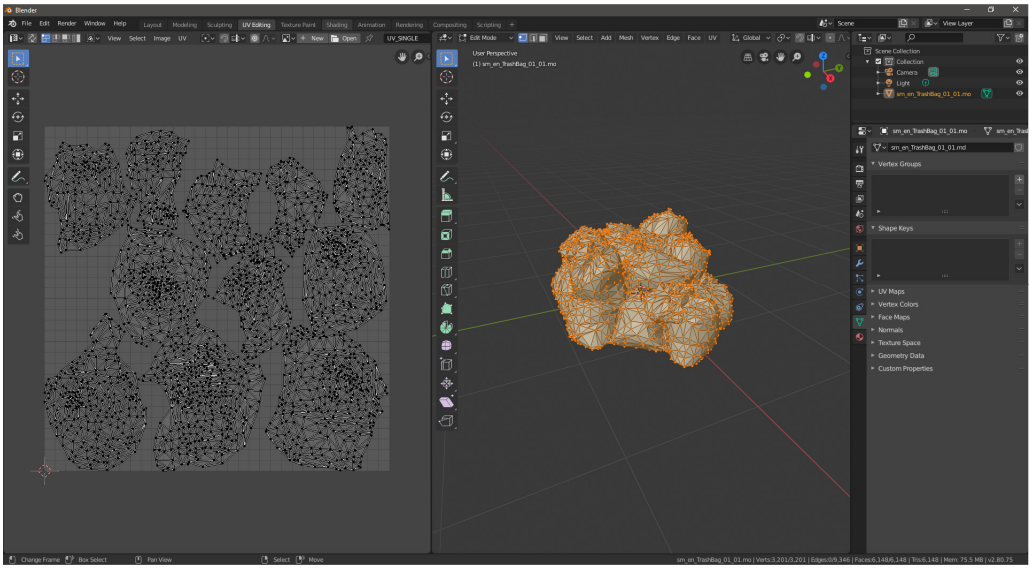
Author: Lutt.online (GitHub/luttje)



Step Three: Apply and preview the Texture using the existing UV maps

When applying a Texture to a model you normally need to explain to the 3D software how the texture needs to be placed on the model. Luckily during conversion with umodel all this information has stayed from it's time in Unreal Engine. Hence applying the texture will be relatively simple.

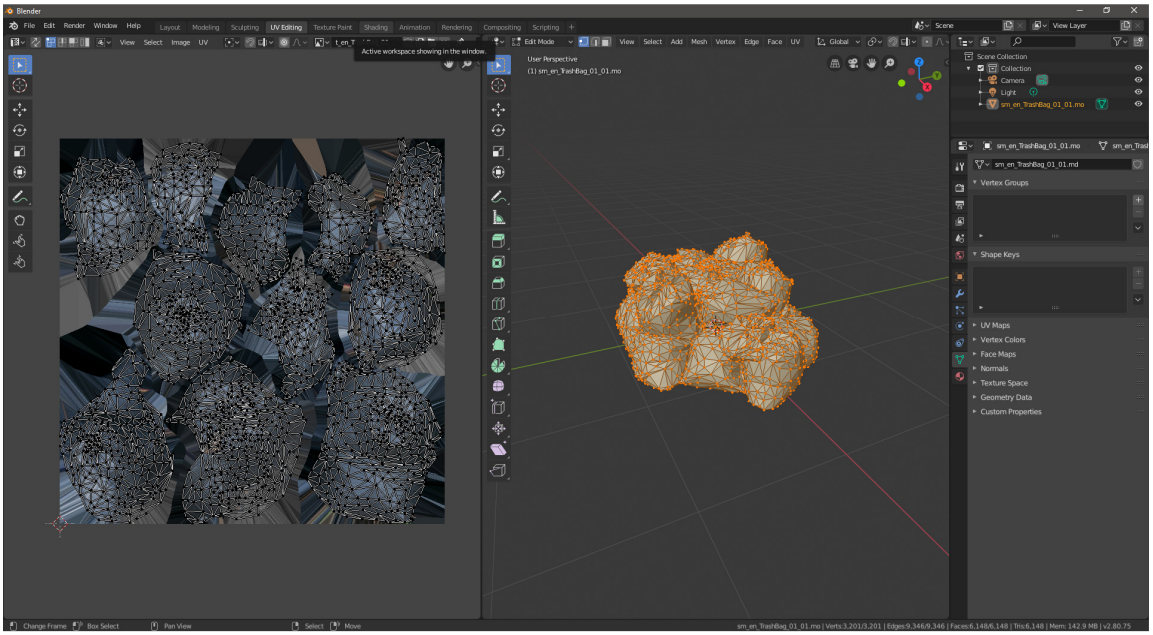
1. Select the 'UV Editing'-workspace at the top of Blender:



(What you see here is the UV-map on the left and the Model on the right. The UV-map is basically the model, but then layed-out on a 2D plane. This way we can easily align the Texture with the Model)

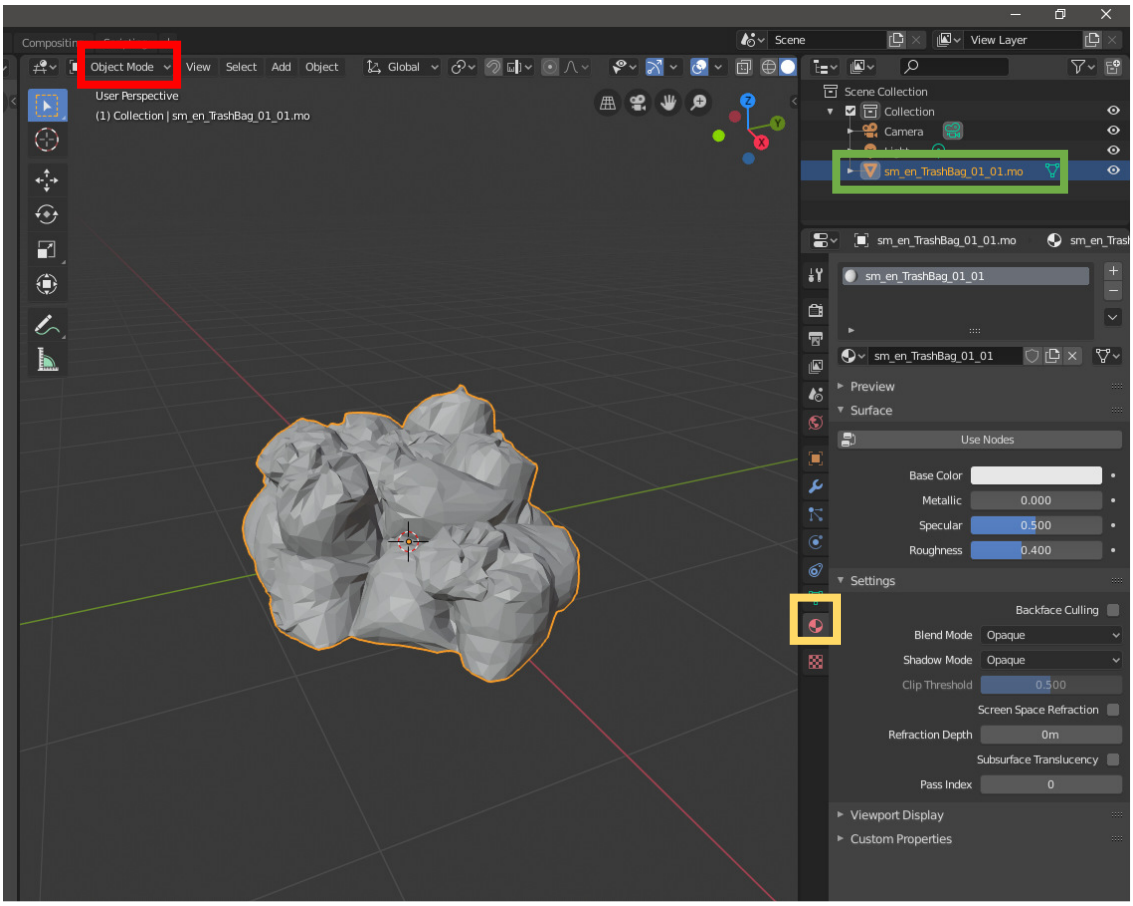
Author: Lutt.online (GitHub/luttje)

2. At the top of the UV-map view (left) there is an ‘Open’-button. You can use this to load the .tga image into the UV-map. *There is likely to be multiple .tga images, so look for the one that looks most like it should to you. There can also be ‘normal’ images that look blue/purple-ish. These can be ignored for now as they describe the depth to the model.*
3. Navigate to the .tga image (that shows nice grey Trashbags in my case) and double-click it:

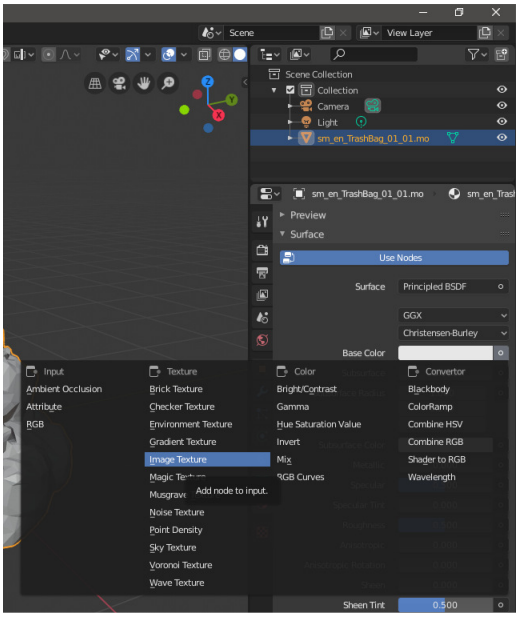


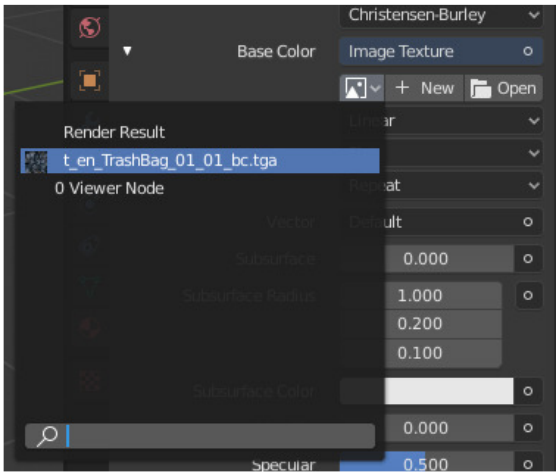
4. At this point we will not yet see the texture on our models on the right-hand side. To show them we need to apply the Texture to the model.
5. Make sure you have the model selected (in the Outlining top-right shown in green) and make sure you are doing this in ‘Object Mode’ (shown in red):

Author: Lutt.online (GitHub/luttje)



6. Click the ‘Material’-tab shown in yellow in the above image.
7. Click ‘Use Nodes’ and more options will unfold. We are looking for a circle behind ‘Base Color’. Click the circle and choose ‘Image Texture’.

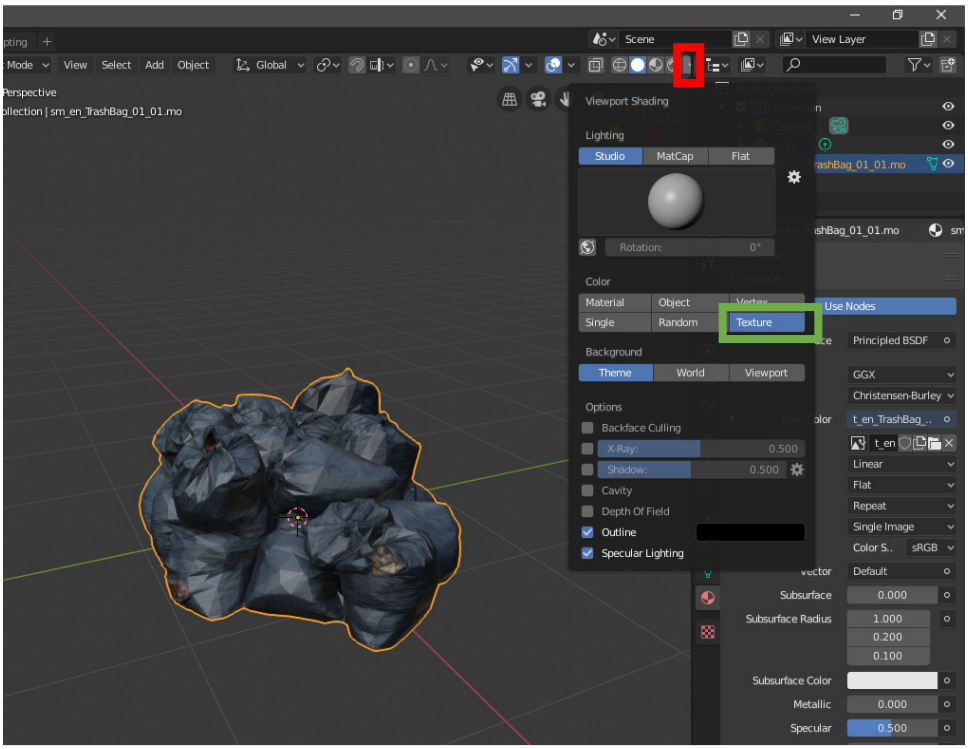




8. Now click the little image dropdown to select our texture.

Author: Lutt.online (GitHub/luttje)

9. The texture is applied, but our Viewport will likely still not show it. To fix this click the Viewport shading dropdown (downward arrow to the right of a bunch of circles, outlined red in the following image). With the shading dropdown open, now click ‘Texture’ under color (outlined green). *Beware, the dropdown button may fall off-screen if you do not have enough space. Widen the viewport to have it appear.*



Have a break, stretch your legs and watch a video: [Garbage Day!]

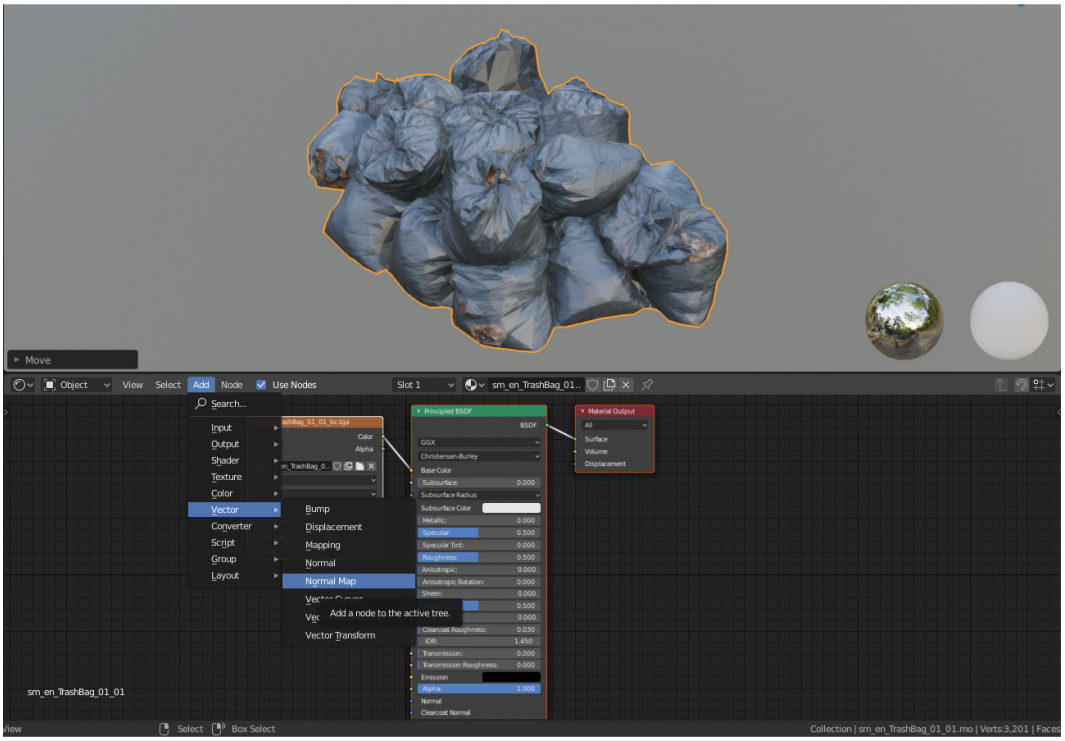
(Optional) Step: Apply and preview the normal map if there is one

A normal map describes depth to a texture. This is used by a game engine to decide how light should fall on the object.

1. In Blender change to the ‘Shading’-workspace using the top tab:

Author: Lutt.online ([GitHub/luttje])

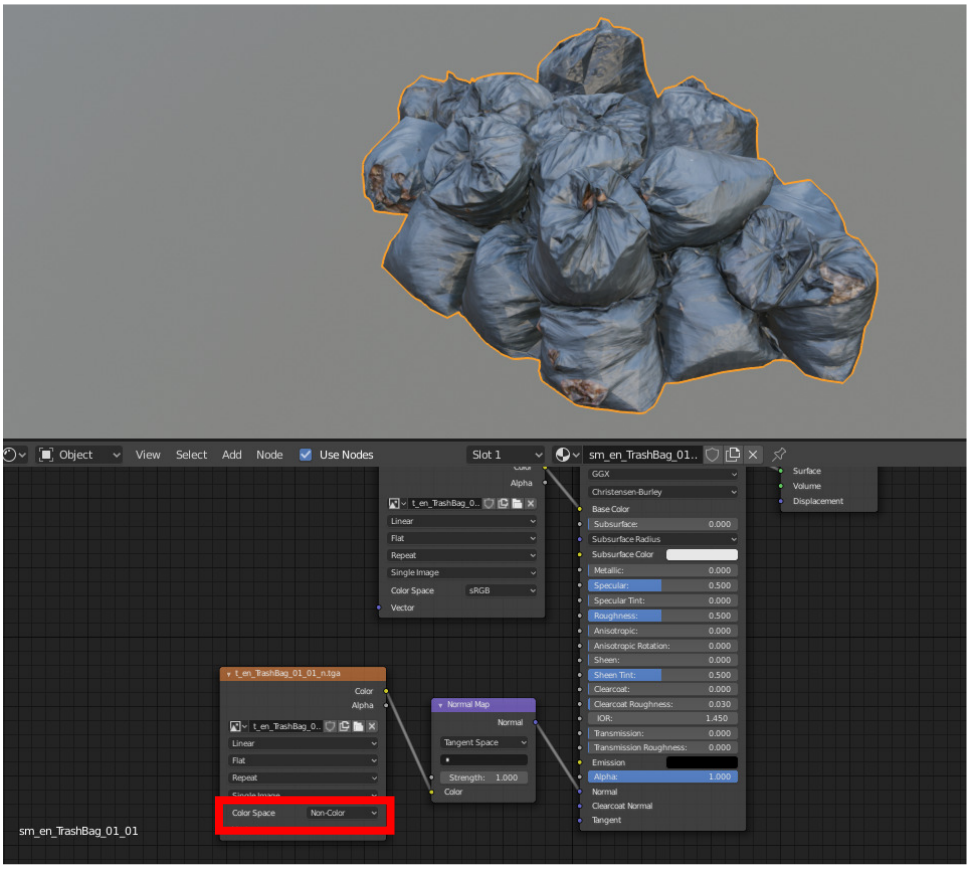
2. Here you can see part of what we just did. We just added a texture to the ‘Base Color’, which can be seen in the bottom node view. *Our image texture node with orange heading directs it’s ‘Color’ to ‘Base Color’ on the material*
3. To also have a normal map we should click the Add-button in the node view -> Go to Vector -> Normal Map. Click anywhere on the node view grid to place this node:



4. We’re going to add another Image Texture, this time with our Normal .tga. Click Add -> Texture -> Image Texture and again click somewhere to place the Image Texture node.

Author: Lutt.online (GitHub/luttje)

5. Now in the newly place Image Texture node click ‘Open’ and navigate to the location of the ‘normal map’ .tga that we converted with umodel.exe earlier.
6. Now configure the ‘Color Space’ in the Image Texture node for our normal map to be ‘Non-Color’ (Outlined in red below). It only provides data on depth after all.
7. Connect the dot from our Image Texture node ‘Color’ to the Normal Map ‘Color’ by dragging it
8. Connect the dot from our Normal Map nodes’ ‘Normal’ to the ‘Normal’ connection on the Principled BSDF. *You should immediately see the lighting of our model become more detailed. If not, play around with the ‘Strength’-value.*



If you haven’t saved a couple times by now, you should really save your Blender project through File -> Save and then selecting a logical name and location on your PC. *Blender could crash you know?!*

Step Four: Exporting this model to an .smd and consequently an .mdl for Source Engine

Source Engine models (.mdl) are built from an .smd with vertex information and a .qc file with model information. Luckily someone has made us some tools to take a lot out of our hands, namely: ‘Blender Source Tools’

Author: Lutt.online (GitHub/luttje)

(Required only once) Installing the Blender Source Tools for blender

1. Download Blender Source Tools from here: http://steamreview.org/BlenderSourceTools/
   This guide comes with version 3.0.3 of the Blender Source Tools, which you can find in the 'blender_source_tools_3.0.3.zip'-file
2. The installation process will be the same as earlier with the .py file, but now we will install the whole .zip-file **(do not unzip the files!):**
   a. In Blender go to Edit -> Preferences -> Add-ons -> Install…
   b. Navigate to the just downloaded Blender Source Tools .zip-file and double-click it.
   c. Now click the checkbox to enable this Add-on and give it some time to load

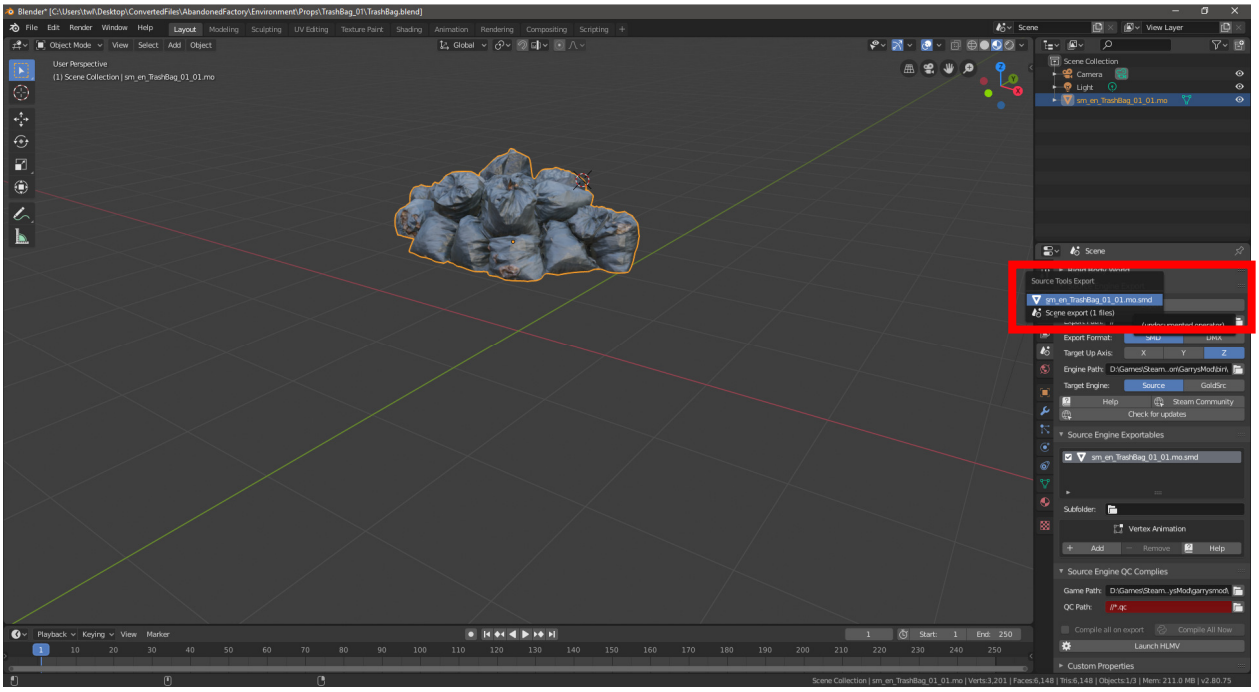
Export the .smd-file

1. By default Blender has our model in a 'Collection' along with the Camera and Light objects. Click 'Collection' in the Outliner (top right) and right click -> Delete
2. The Blender Source Tools options can be found in the 'Scene'-tab. Select the 'Scene'-tab (outlined red below) and scroll down to the 'Source Engine Export' and 'Source Engine Exportables' options (unfold them if needed):

3. Under the 'Source Engine Export'-category type '//' as the 'Export Path' to just have it be the current folder (where your Blender project is saved).
   This is where the .smd file will be exported to.
4. Select 'SMD' as the 'Export Format' to create a model for old Source Engine games (like Counter-Strike: Source, Garry's Mod, Half-Life 2, etc)
5. For 'Engine Path' I selected the 'GarrysMod/bin'-folder that contains 'studiomdl.exe'. By default it's location is at: 'C:\Program Files\Steam\steamapps\common\GarrysMod\bin\'
   For games like Team Fortress the path is similar. Just make sure you select the 'bin'-folder in which a studiomdl.exe exists. The studiomdl tool is used to compile the model.
6. Keep 'Target Engine' on 'Source'
7. If you didn't forget to do step 1 of this chapter, you should see the model under 'Source Engine Exportables'. Select it.

Author: Lutt.online (GitHub/luttje)

8. With the model file selected, click Export under ‘Source Engine Export’ and click the models’ name again:



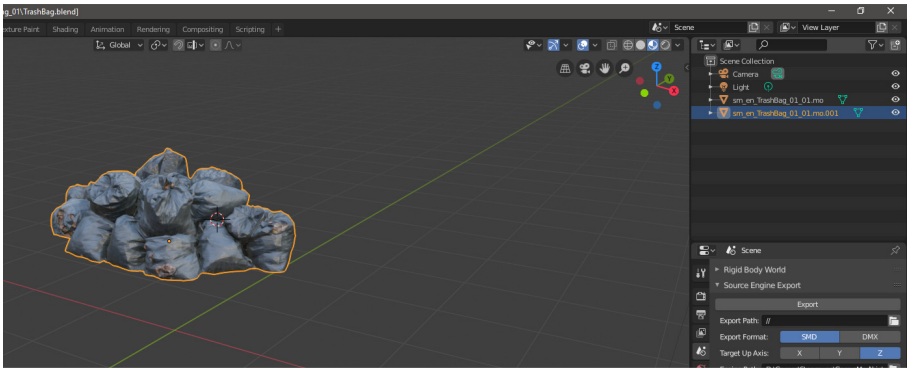
At the bottom of Blender a short message will appear like ‘1 files exported in 0.5 seconds’. You will now have an .smd file at the configured ‘Export Path’.

Getting errors? You can open the Blender Console to view all errors through Window -> Toggle System Console.

Creating a rough collision model

The collision model will be a low-poly (few details) version of the model that describes how players collide with it.

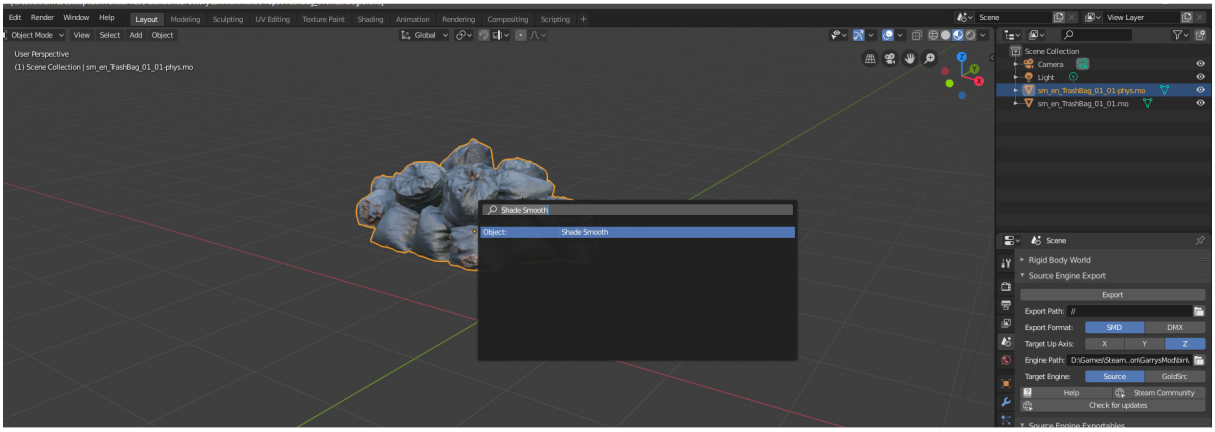
1. In the Outliner copy the model by right-clicking -> Copy
2. Then immediately paste the model (not changing the position, so it sits exactly over the copied model):



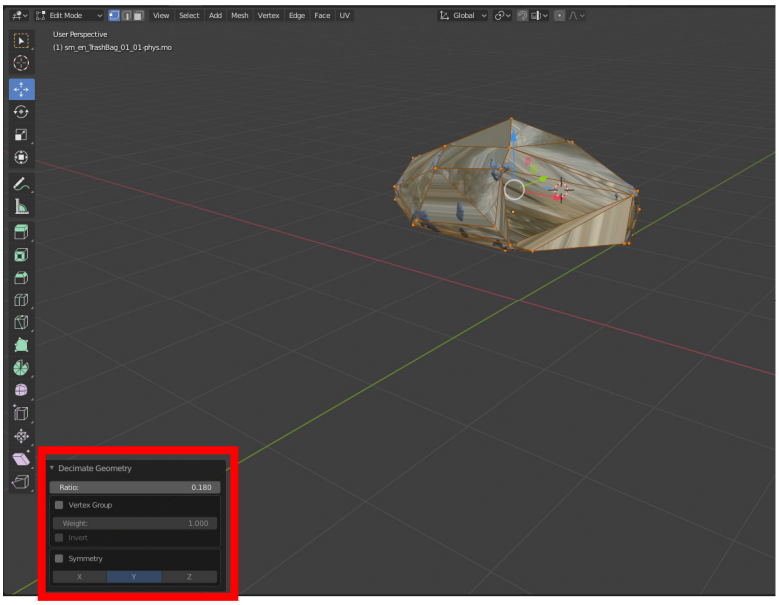
3. I renamed my model to sm_en_TrashBag_01_01-phys.mo by double-clicking its name in the Outliner
4. Select the copy in the Outliner and press

Author: Lutt.online (GitHub/luttje)

5. (In ‘Object Mode’) press space and type Shade Smooth, then press enter.



6. Now go to ‘Edit Mode’ and press space again, this time type ‘Convex Hull’.
   Now you will have a very rough and low-poly version of the model.
7. With the object still selected look at the bottom right of blender to see how many ‘Tris’ there are. A good number to aim for with a collision model is at most 32 convex pieces.
   I am unsure if Tris are the Blender equivalent of [what Valve describes here as ‘convex pieces’](#).
8. If you find that your collision model is still too complex you can manually remove vertices *or* use a faster process of ‘Decimating the Geometry’: again press space (in Edit Mode) and type ‘Decimate Geometry’
9. Play around with the settings in the bottom left of Blender and keep an eye on the Tris count in the bottom right bar:



10. Now export the model the same way you did earlier: in the Scene-tab select the collision model under ‘Source Engine Exportables’, click Export under ‘Source Engine Export’ and click the name of the collision model again.
11. Now a second .smd file has been created next to the other one.

Author: Lutt.online (GitHub/luttje)

Step Five: Compiling the .smd to an .mdl using a .qc

1. Under 'Source Engine QC Complies' *(I think that's a typo, should be compiles?)* I selected my Garry's Mod game folder. By default that is located at 'C:\Program Files\Steam\steamapps\common\GarrysMod \garrysmod\'.
   This is where QC will compile the .mdl files to.
2. Now comes the writing of a .qc-file. We will use this example to start off from:

```
$modelname     "props_custom\trashbags.mdl"
$scale  2
$body mybody   "sm_en_TrashBag_01_01.mo.smd"
$staticprop
$surfaceprop   plastic
$cdmaterials   "models\props_custom\trashbags"

$sequence idle "sm_en_TrashBag_01_01.mo.smd"

$collisionmodel       "sm_en_TrashBag_01_01-phys.mo.smd" {
   $mass 2
   $concave
}
```

   A bit of information

   - *$modelname can be changed to anything you want. After compiling you can't change the model, so be sure to set it correctly here.*
   - *$scale is useful, because Unreal Engine uses different units of Source Engine. Play around with it's values to get the right sized model.*
   - *$body describes what bodygroups there are. We only have one for this model and call it 'mybody'.*
   - *$staticprop indicates this is not a ragdoll or anything special*
   - *$surfaceprop describes to the game what sounds your objects makes (when dropping or walking over it). You can change the surfaceprop to anything listed here:* https://developer.valvesoftware.com/wiki/Material_surface_properties
   - *$cdmaterials describes where the game should look for the .vmt accompanying the model. A vmt is a Source Engine Material and we will generate it later using VTFEdit.*
   - *$sequence describes the animations. In this case we don't have any, since it's just a bunch of garbage bags. Therefor we set the 'idle' animation to the same thing as $body*
   - *$collisionmodel describes how players collide with it. This should be a low-poly (very little details) model that nicely fits around our model.*

   You can read more about QC-files here:
   https://developer.valvesoftware.com/wiki/QC

3. Copy that qc-code into a new file named 'solid.qc'. Place the file in the same folder as the .smd-file.
4. Modify the settings to your liking.

Author: Lutt.online (GitHub/luttje)

5. Now under ‘Source Engine QC Complies’ click on the ‘solid.qc’-button which should show if you have placed the .qc-file in the same folder as your .blend-file:
   Strangely I got an error on first-try, but without changes it worked fine the second time I pressed the ‘solid.qc’-button. Again check the Console (Window -> Toggle System Console) for other errors.
6. If no errors (shown in red text) occurred, the model files will be compiled to your game folder. For me I found them here: ‘C:\Program Files\Steam\steamapps\common\GarrysMod \garrysmod\models/props_custom\trashbags.*’
   There will be these file extensions all with the same model name: .mdl, .dx80.vtx, .dx90.vtx, .phy, .sw.vtx, .vvd

Step Six: Converting the .tga textures to .vtf’s with an accompanying .vmt

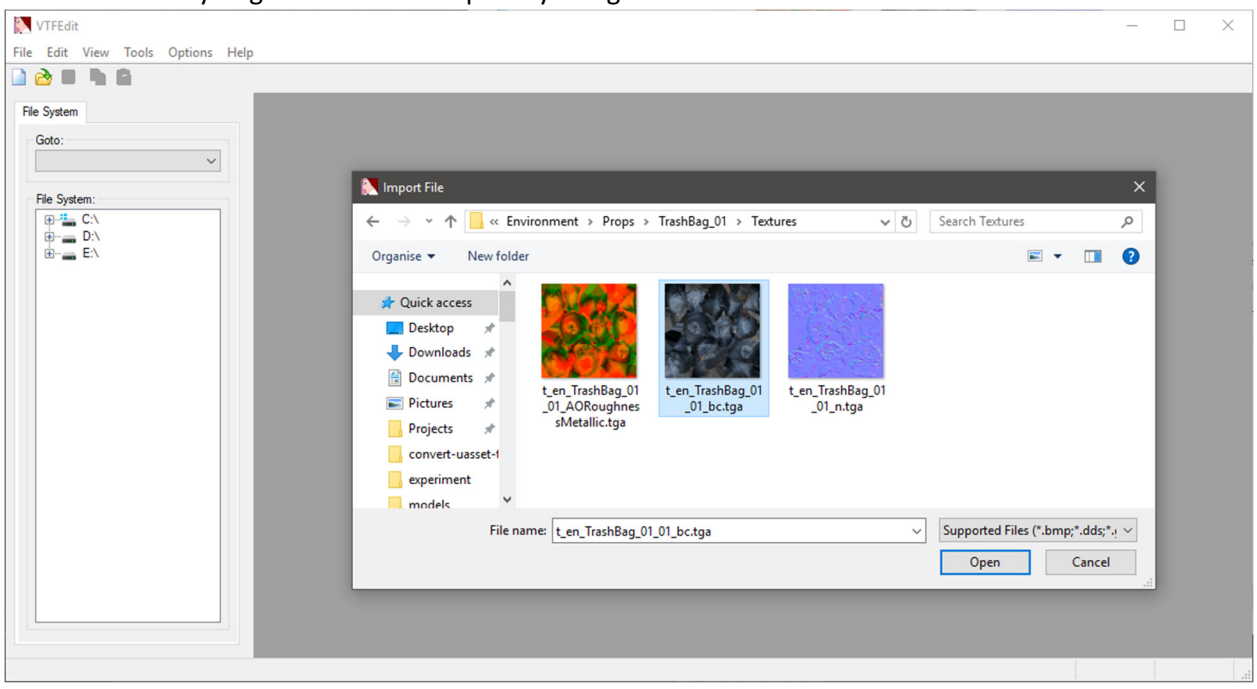
Finally, the final step before testing! We need to convert our .tga textures to .vtf-files. Also we will need to write a .vmt file that points to them.

(Required only once) Install VTFEdit

1. Go to Nem’s Tools and download VTFEdit: http://nemesis.thewavelength.net/index.php?c=178
   The installer for version 125 for .NET Framework 1.1 is also included with this guide as ‘vtfedit125-11.exe’
2. Install and open VTFEdit

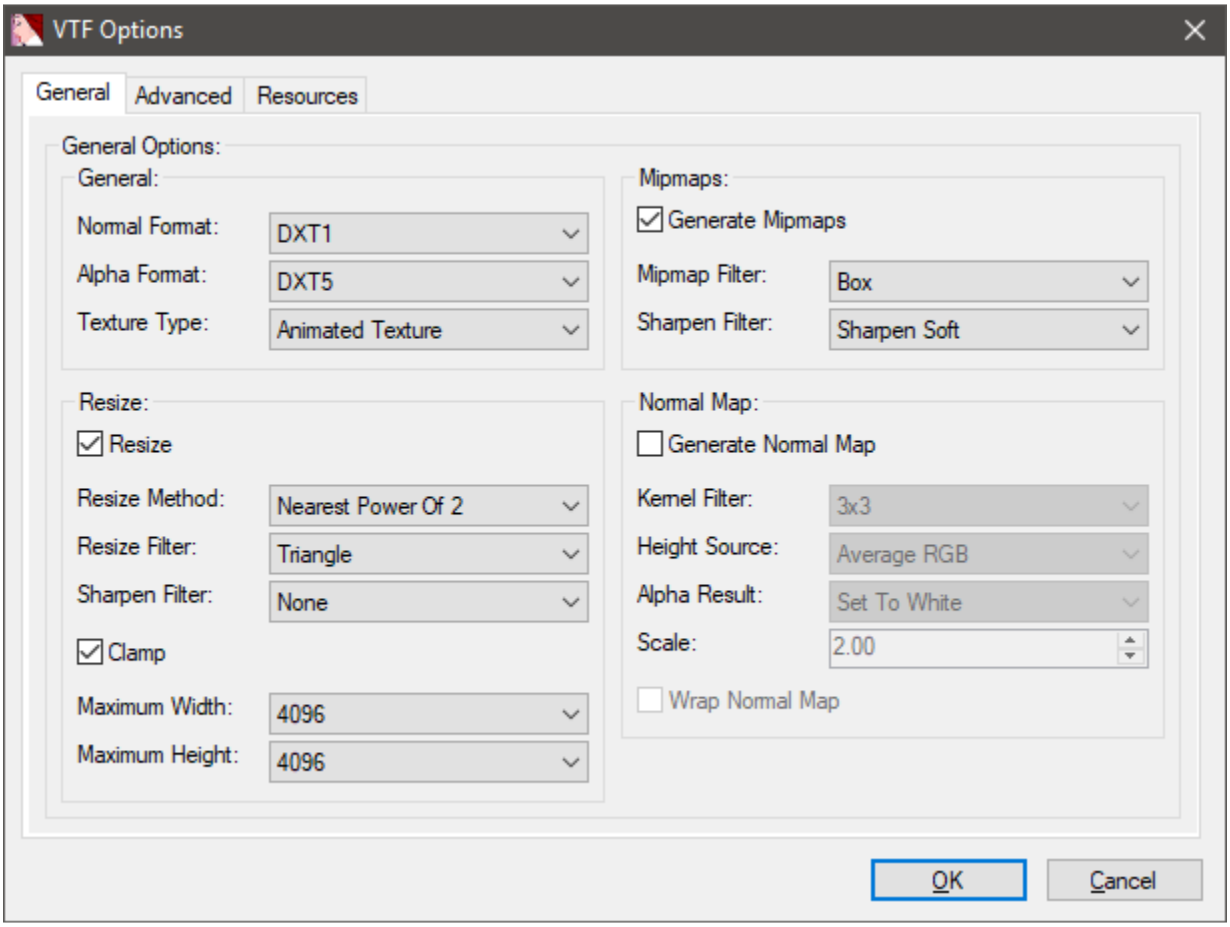
Convert the .tga to .vtf

1. In VTFEdit: for both the texture and normal-texture you need to go File -> Import -> Select the texture file that you got in the first chapter by using the umodel.exe tool:

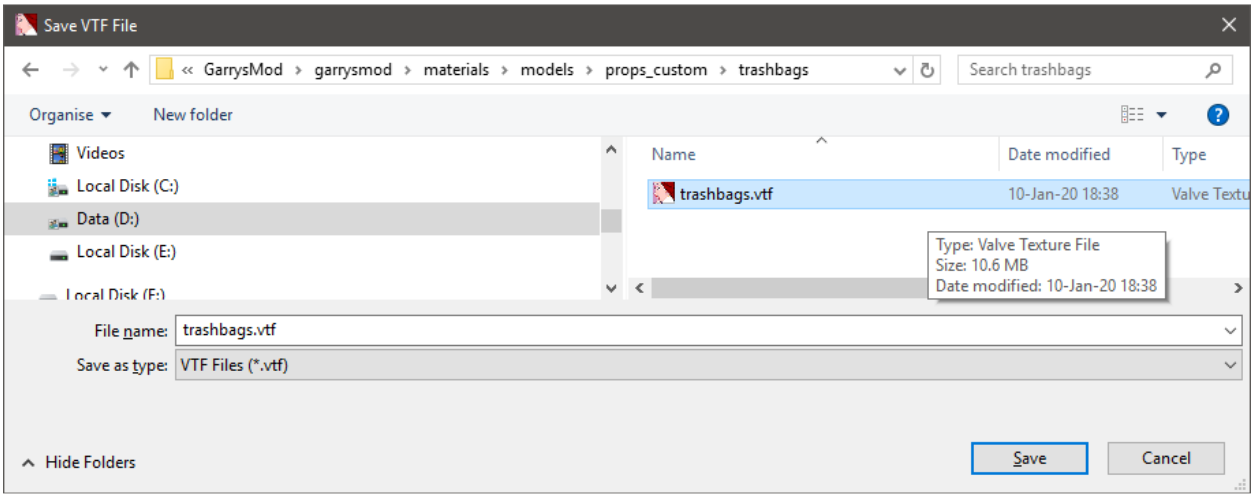


Author: Lutt.online (GitHub/luttje)

2. An import window will open. Leave all settings default and press ‘OK’. Give it some time to load the texture:



3. After it’s done loading, click File -> Save As and save it to the folder configured in the .qc-file earlier:



For me it was in (the garrysmod folder)/materials/models/props_custom/trashbags/trashbags.vtf

4. Save the normal-texture as well, but in (the garrysmod folder) /materials/models/props_custom/trashbags/trashbags_normal.vtf

Author: Lutt.online (GitHub/luttje)

Create a .vmt file

To tie the model and textures together you will need to make a Source Engine Material (.vmt) that has the same name as the model had in Blender (in this guide that was sm_en_TrashBag_01_01).

1. In the same folder as the .vtf files (materials/models/props_custom/trashbags/) create a file named 'sm_en_TrashBag_01_01.vmt'
2. The contents will look like this example:

```
"VertexLitGeneric"
{
        "$basetexture" "models/props_custom/trashbags/trashbags"
        "$bumpmap" "models/props_custom/trashbags/trashbags_normal"
}
```

A bit of info:

- *VertixLitGeneric must be specified for models that you want to have lit realisticly*
- *$basetexture points to the main texture (what the model surface looks like)*
- *$bumpmap points to the texture that specifies the bumps in the surface so that light falls onto it with higher detail (normal map)*

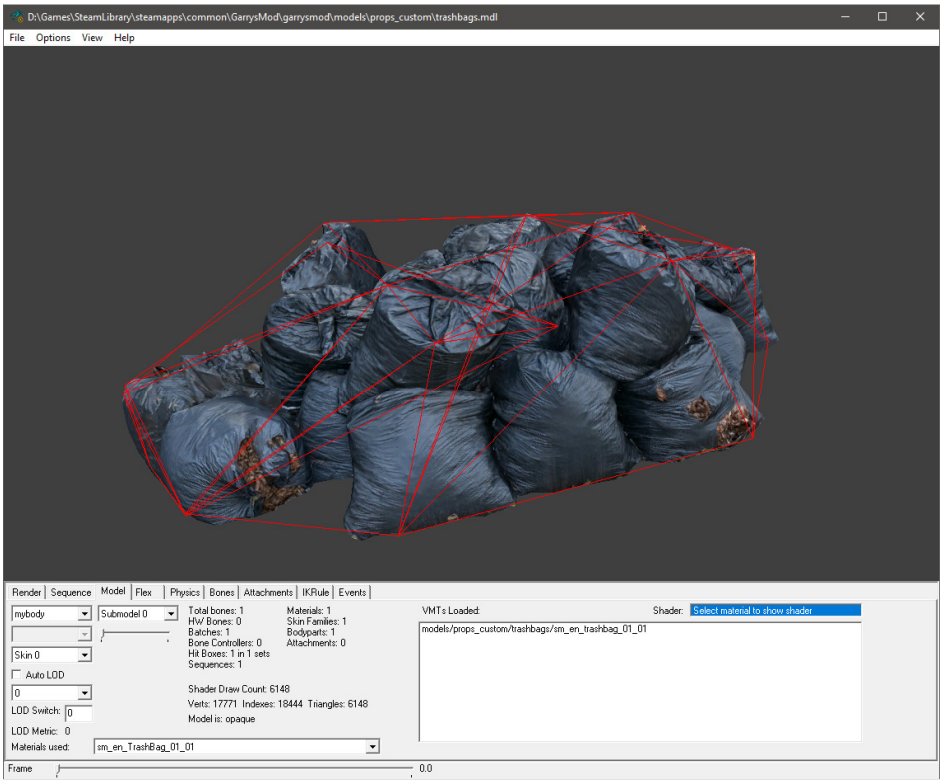
You should be done! Test the model by spawning it in your favorite game or using HLMV (Half-Life Model Viewer). You can open the Model Viewer from the Blender Source Tools in Blender.`

Continue reading, because if you followed the guide perfectly something may still have gone wrong.

Author: Lutt.online (GitHub/luttje)

Extra: Checking for problems

Using the Half-Life Model Viewer (HLMV) you can check the model for problems.



Finding the right Source Engine Material (.vmt) name

1. Go to the ‘Model’-tab and look behind ‘Materials used:’
2. The .vmt should be named exactly like this

Showing the collision model

1. Go to the ‘Physics’-tab and check the box in front of ‘Highlight’
2. You will see (with red wireframe lines) what the collision model looks like

Author: Lutt.online (GitHub/luttje)

(After testing in-game) my model is way too small!

After testing in gmod I saw that the model was quite small… I want it about 12 times bigger! Go back to the solid.qc-file and change $scale to be 12 times bigger, so '$scale 24'. In Blender click the 'solid.qc'-button again. The models will be recompiled to the game folder. Restart your game to see the effects. After testing a bit more I landed on a $scale of 30 that looked like I wanted it to. I also changed the $surfaceprop to 'slime' so it made a disgusting sound when walking over it (plastic sounded too much like walking over a plastic table).

Author: Lutt.online (GitHub/luttje)

Sources used

https://steamcommunity.com/app/1840/discussions/0/622954747289620976/

https://github.com/Befzz/blender3d_import_psk_psa

http://steamreview.org/BlenderSourceTools/

https://developer.valvesoftware.com/wiki/Blender_Modelling_Walkthrough

https://developer.valvesoftware.com/wiki/QC

https://developer.valvesoftware.com/wiki/$scale

https://developer.valvesoftware.com/wiki/Collision_mesh#Caveats

https://developer.valvesoftware.com/wiki/$collisionmodel

https://developer.valvesoftware.com/wiki/Material_surface_properties

https://developer.valvesoftware.com/wiki/Bump_map

https://developer.valvesoftware.com/wiki/$bumpmap

https://all3dp.com/2/blender-how-to-reduce-polygons/

https://blender.stackexchange.com/questions/121778/convex-hull-as-mesh

https://blender.stackexchange.com/questions/38970/reset-object-origin-without-moving-object

https://www.katsbits.com/codex/bsdf-normal-maps/

https://blender.stackexchange.com/questions/131034/textures-not-showing-in-viewport-blender-2-8

Author: Lutt.online (GitHub/luttje)