The Render Model

SteamVR™ Tracking

Introduction

A SteamVR™ render model is the 3D model that represents a controller in VR. By default, the JSON file of an object references a render model that looks like the controller object. Having a render model that looks like the object is very useful for end users, because it makes picking up the object and handling it in VR much easier. Once an application loads, it may override the render model with a more appropriate model. The render model for a game may appear as a hand, gun, bow, paint brush, or anything else the application developer imagines. Because third parties are also developing models for controllers, it is important that all controllers agree on a coordinate system. If all render models are created and aligned on the same coordinate system, replacing one render model with another should not change the fundamental relationship between the model and the user's hand.

This document outlines the requirements and process for creating a simple render model to evaluate tracking. OpenSCAD and Blender are two freely available software programs used to in this example. Designers with access to their own solid modeling and 3D graphics software may follow a similar procedure using the tools of their choice. Ultimately, designing a production render model requires more detail than described in this document. However, the process presented below should give engineers the basic understanding required to create a render model for troubleshooting and proof of concept purposes.

Creating a Render Model

The render model is a 3D image that represents the controller in VR. At a minimum, a render model is comprised of three files: a Wavefront (OBJ) object with a material (MTL) and texture (TGA or PNG). The process for creating a render model has several steps. First, an STL file of the model is exported from the design CAD. Then, the STL model must be aligned with existing controller render models. The aligned STL model is then imported into a 3D graphics program to add a material, UV map, and texture. Finally, the finished render model is exported as an OBJ file.

Export the Controller Object STL

The object shape is likely defined by a mechanical engineer, using a 3D solid modeling software package. Exporting this shape as an STL is the first step to creating the render model. However, how the STL is exported greatly affects the rest of the process.

When exporting the STL file for the render model, it is important to keep these rules in mind:

1) The render model OBJ file cannot contain more than 65,000 vertices.
2) The render model OBJ file must contain only one object

Keeping the number of vertices to a minimum requires exporting an STL file that only describes the outer surface of the object. The surface should include buttons, trackpads, and other external features is to give the render model a complete look and feel. However, internal features like mounting bosses, ribs, and alignment posts add unnecessary detail that results in far too many vertices for a final render model. Also, pay attention to the export properties in the CAD program. There is likely a way to reduce the resolution of the meshes to reduce the number of vertices created. Experiment to find the best trade-off between resolution and aesthetics.

Also, when exporting the STL, export all surfaces as a single STL file. When the single STL object is converted to an OBJ, the OBJ should contain only one object.

Once the STL model of the controller is exported from CAD, the next step is to align it to an existing controller render model.

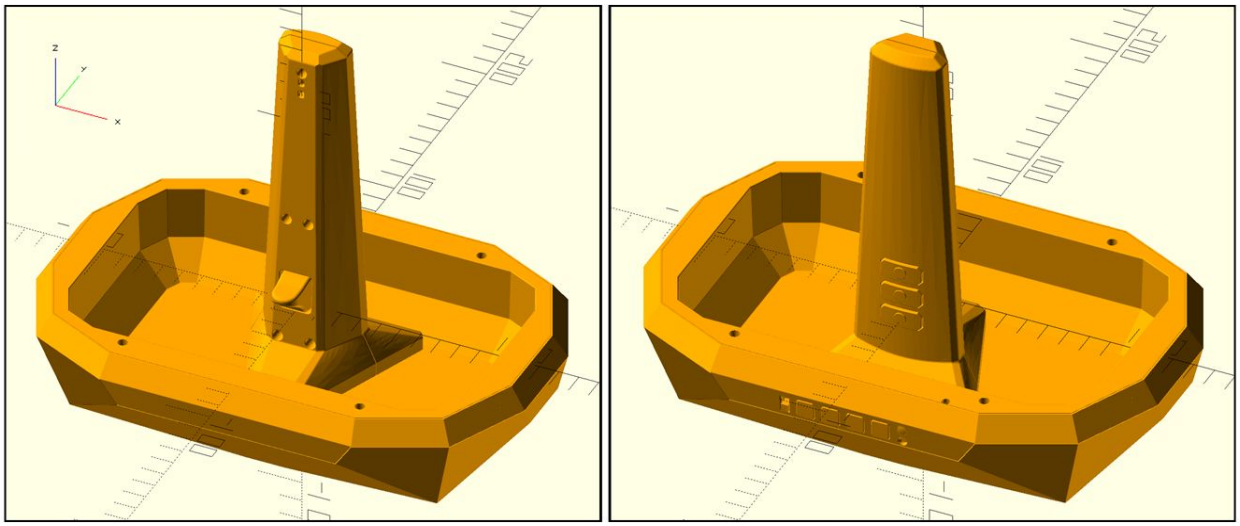
Import the Controller Object STL File

Aligning the render model of the new controller object to an existing controller very important. Different applications override the render model with their own. When the render models are aligned, the new controller object still feels correct when replaced with another render model.

In this document, OpenSCAD is used to import and align the controller object with the HTC Vive controller render model. OpenSCAD is only used for convenience, because it is free and the commands are easily documented. These steps could also be performed in other solid modeling software packages.

First, the STL file that defines the controller object is imported into OpenSCAD.

```
color("orange")
   import("ref_controller_not_aligned.stl", convexity = 4);
```



The STL file shown above was exported with units of millimeters, but the target units for the render model is meters. The scale command is used to convert the millimeters units to meters.

```
scale([0.001, 0.001, 0.001])
   color("orange")
      import("ref_controller_not_aligned.stl", convexity = 4);
```

Convert the Reference Controller Render Model to STL

The goal is to align the new object with a reference render model from a highly supported controller. The controller developed for the HTC Vive is the controller most applications are developed against. Therefore, it is highly recommended to align new controller render models to the render model of the HTC Vive controller. To import the Vive controller model into OpenSCAD, it must be in STL format. Blender may be used to create an STL file from the Vive Controller's OBJ render model.

Render models are stored in this directory, referred to as <rendermodels> elsewhere in this document.

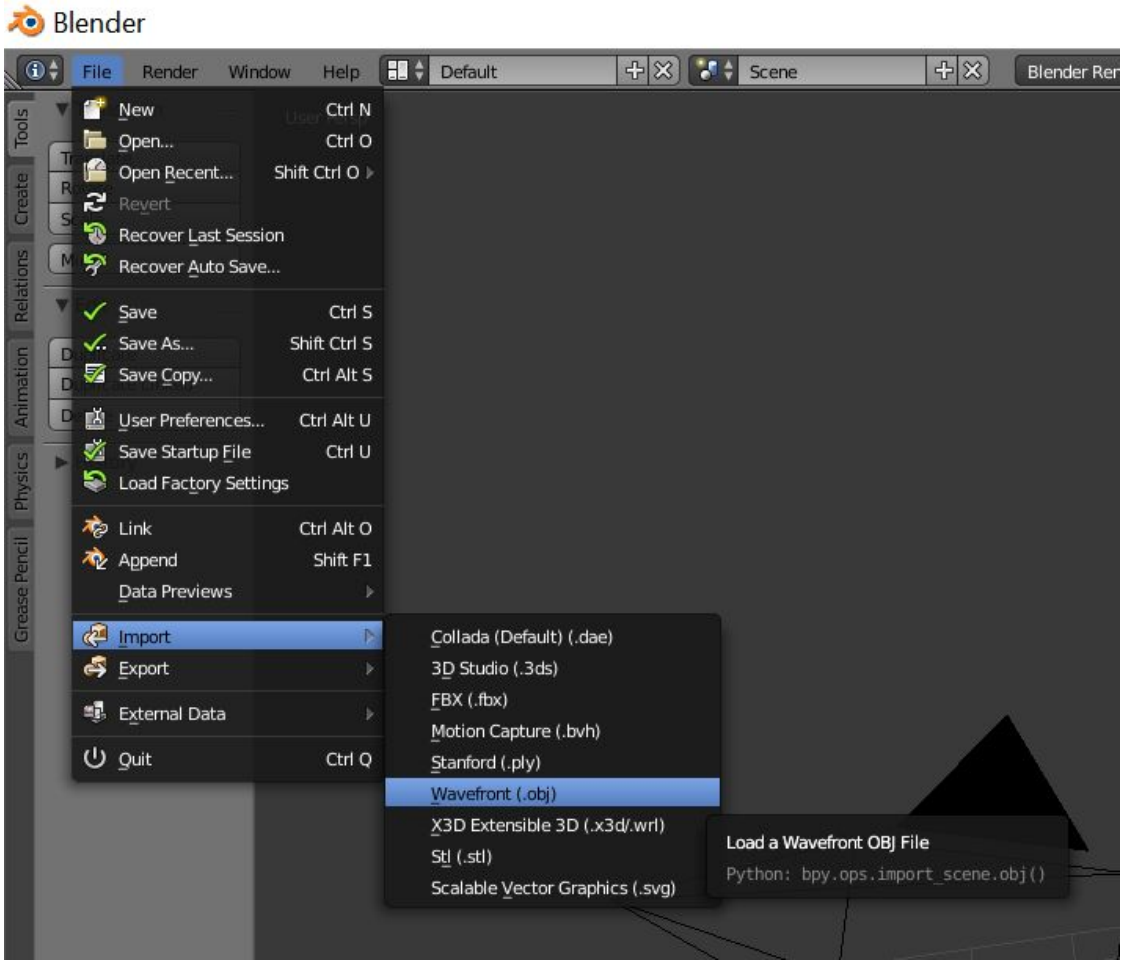
```
C:\Program Files (x86)\Steam\steamapps\common\SteamVR™\resources\rendermodels
```

The HTC Vive render model is found in this subdirectory:

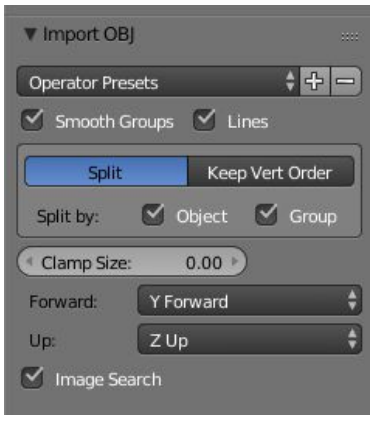
```
<rendermodels>\vr_controller_vive_1_5
```

The top level OBJ file for the controller is named vr_controller_vive_1_5.obj.

Import the Vive render model into Blender by selecting File > Import… > Wavefront (.obj) and browse to vr_controller_vive_1_5.obj. Be certain to configure the import settings for Y Forward and Z Up.



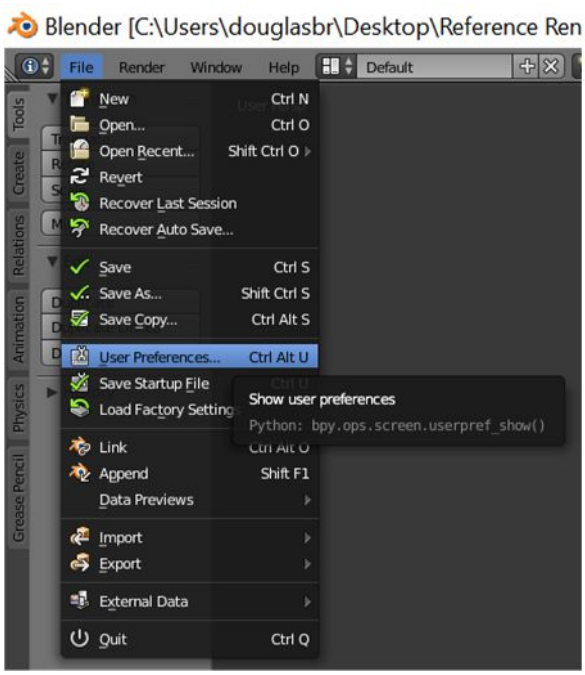
When browsing for the file, the import settings are displayed in the lower left of the window. Set the import axes to Y Forward and Z Up. Otherwise, Blender rotates the OBJ file on import.

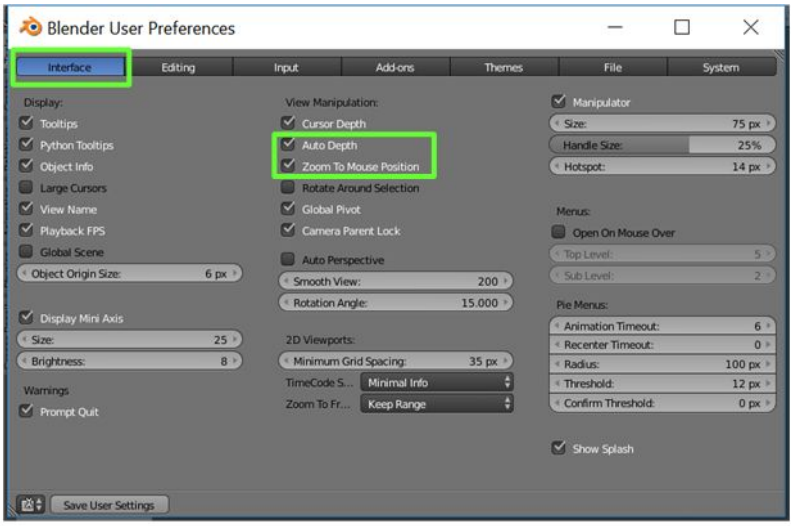


Note: Anytime an OBJ is imported or exported from Blender it is critical to verify that the import or export settings are: Forward = Y Forward, Up = Z Up. These settings are not the default for OBJ files in Blender.

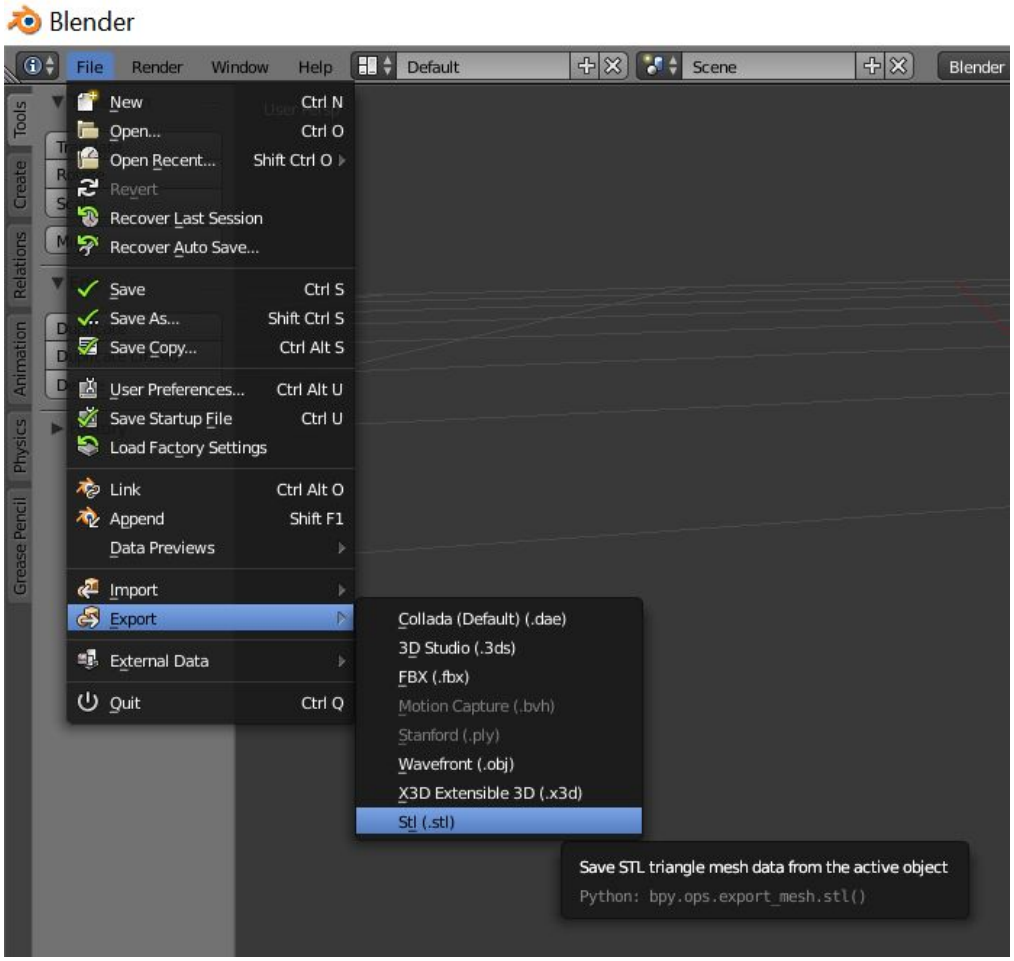
Once the OBJ file is imported, the render model appears at the origin.

Tip: Blender has many options for zooming. You may find it helpful to turn on “Auto Depth” and “Zoom to Mouse Position” in the “Blender User Preferences” dialog on the “Interface” tab.





Export the render model for the Vive controller as an STL file by selecting File > Export… > Stl (.stl). Save the file as Vive_Controller.stl.



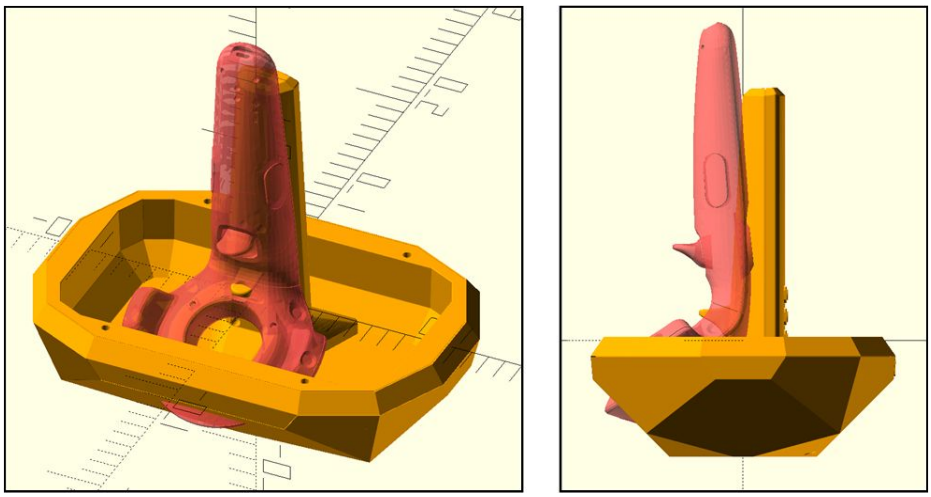
Now that an STL file of the Vive controller exists, it may be imported into OpenSCAD as a reference for aligning the position of the controller object's render model.

Import the Reference Controller Render Model STL File

The new Vive controller STL file can be imported into OpenSCAD along with the STL of the controller object, by adding the following code:

```
#color("grey") import("Vive_Controller.stl", convexity = 4);
```

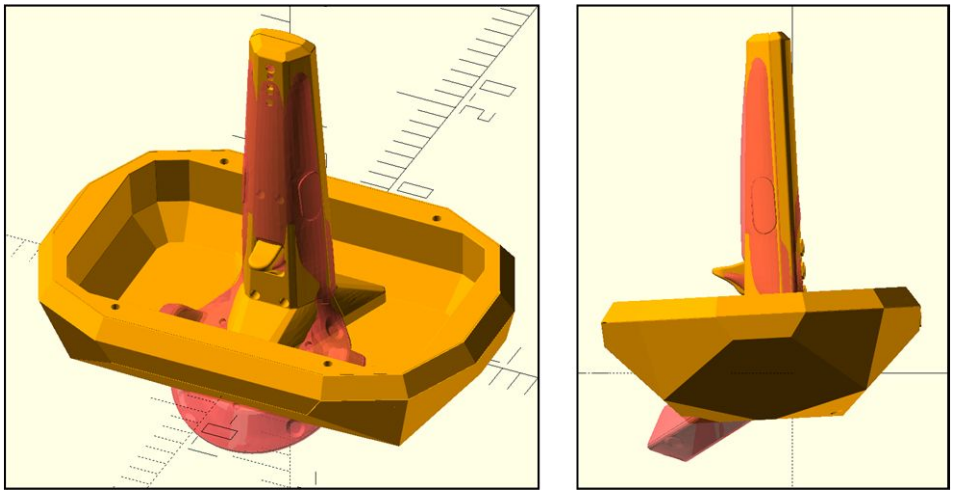
Tip: Use the # character in front of the Vive render model to display it with transparency. Seeing through the object makes it easier to align the two intersecting objects.



Align the Object and Reference Controller Models

It is clear from the above illustration that the new object is already very close to the Vive controller. Applying a translation to the new object aligns the triggers and a slight rotation around the X axis aligns the handles.

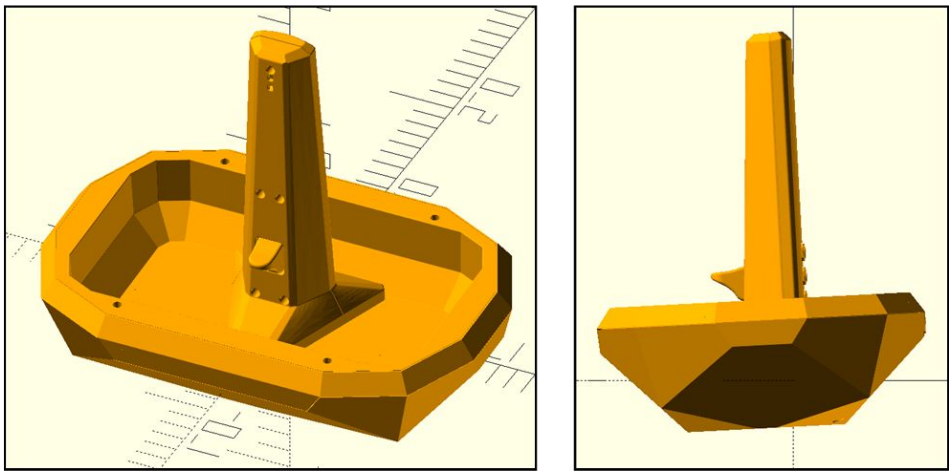
```
rotate ([3, 0, 0])
   translate ([0, -0.015, 0.040])
      scale([0.001, 0.001, 0.001])
         color("orange")
            import("ref_controller_not_aligned.stl", convexity = 4);
```



Export the Object Render Model STL File

Once the new object is aligned with the Vive render model, the STL file for the render model of the new object may be exported. In OpenSCAD, it is easy to single out a geometry by marking it with an exclamation point.

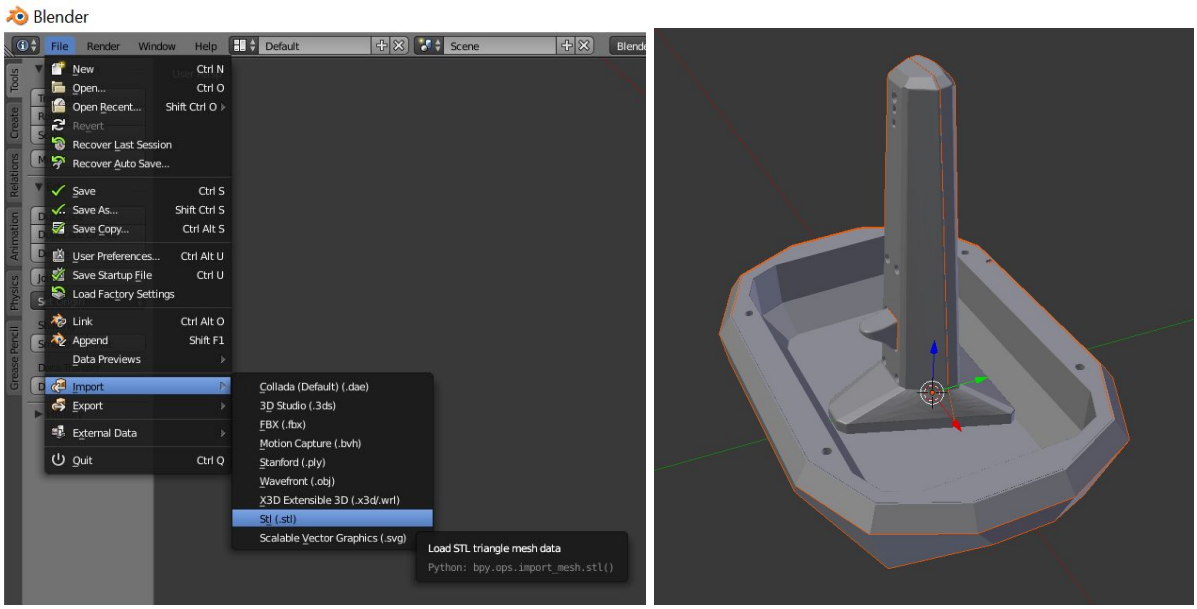
```
!rotate ([3, 0, 0])
   translate ([0, -0.015, 0.040])
      scale([0.001, 0.001, 0.001])
         color("orange")
            import("ref_controller_not_aligned.stl", convexity = 4);
```



Press F6 to build the object. The object may disappear for some time during the build process. When the build completes, the object reappears, colored yellow. Select File > Export > Export as STL… Use a filename that indicates the new STL file is aligned with the reference.

Import the Aligned Controller Object STL File into Blender

In Blender, select File > Import > Stl (.stl).



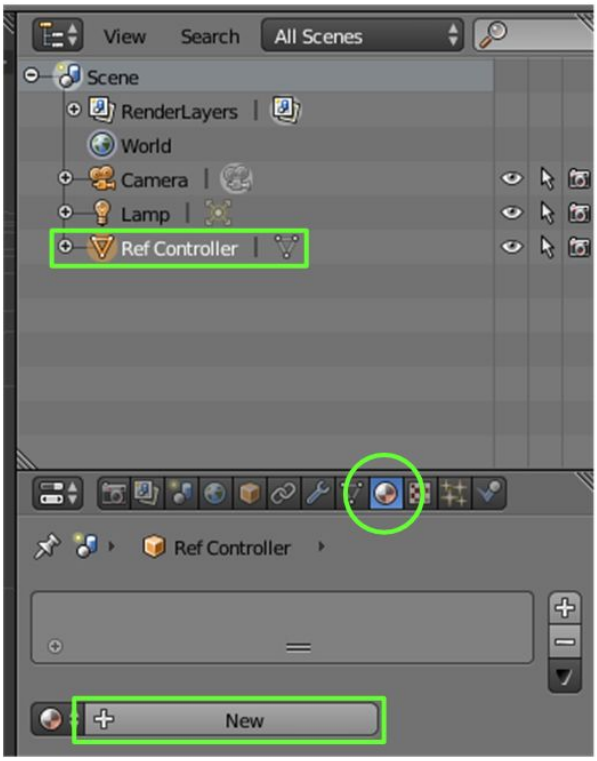
The imported STL file defines the render model shape, but a material and texture must be added.

Add a Material

Render models require materials and textures to appear in SteamVR™. The OBJ file must reference a material (MTL) file.

Add a material:

1. Find the imported STL file in the upper right of the Blender window.
2. Select the STL file.
3. Click the material button.
4. Click +New and give the material a name.

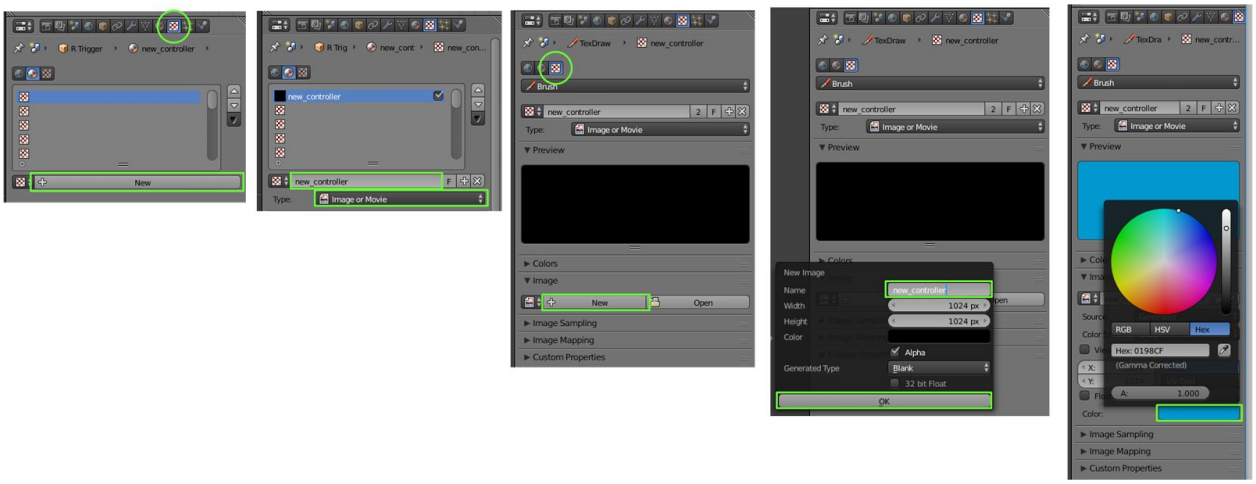


Add a Texture

Materials require textures to appear in SteamVR™. Textures are image files (TGA or PNG), that are applied to the surface of an object. The material file references a texture file.

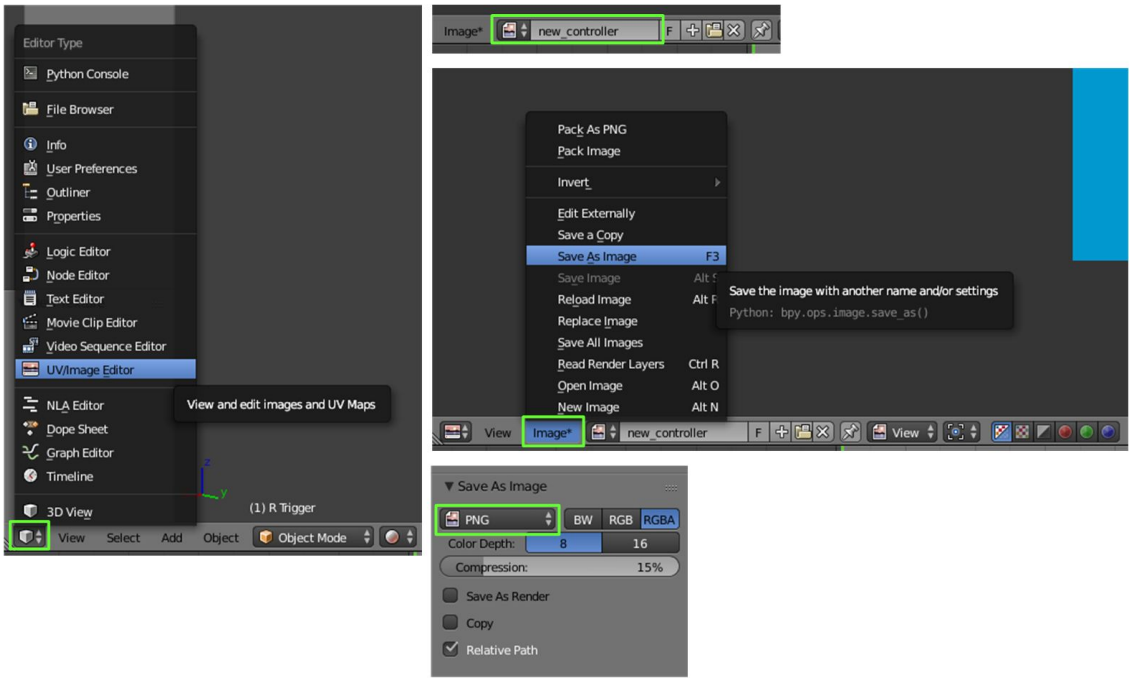
Create a texture for the material:

1. Select the texture button.
2. In the Image pane click New.
3. Give the image a name and click OK.
4. Select a color for the image.



Save the simple texture that was just created as an image file:

1. Change to the UV/Image Editor View.
2. Select the new texture.
3. Click Image and Save As…
4. Select the desired file type in the lower left corner and save the image file.



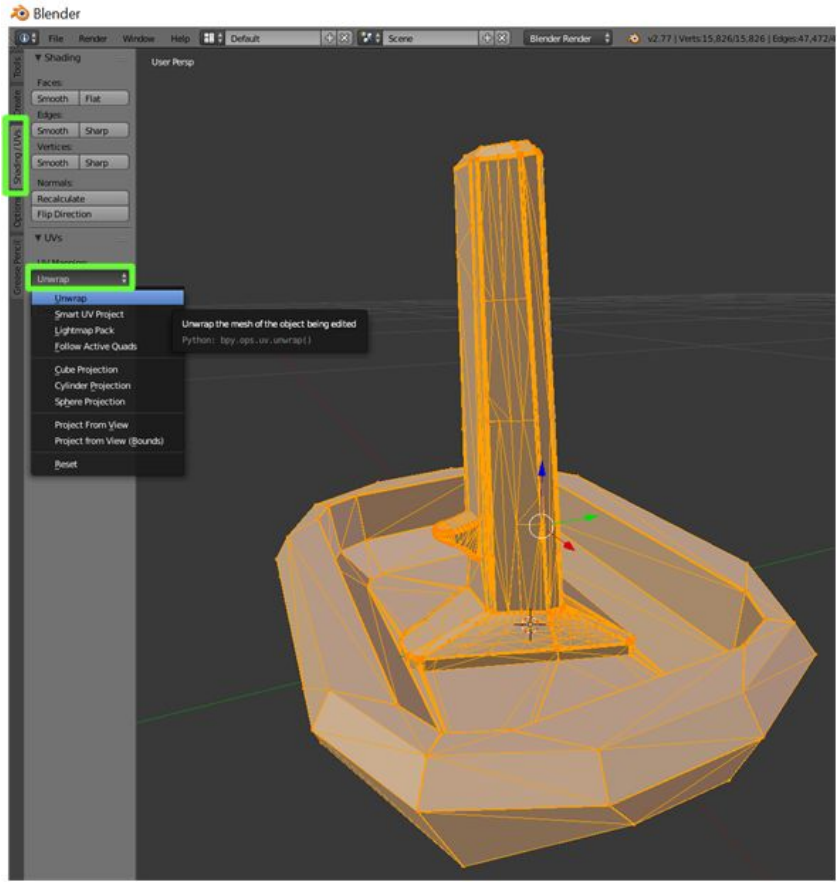
Create the UV Map

The texture is a 2D image that must be mapped onto a 3D shape, which requires a UV map. Blender can create a UV map from the existing STL by unwrapping the object. Unwrapping is an attempt to flatten the 3D image. Imagine taking a cardboard box and trying to unfold and flatten it out onto the floor. When doing this with a complex shape, the results can be messy, but it is possible to tell Blender where to cut the shape to unfold and flatten it onto a two dimensional surface.

UV maps are created in Edit Mode:

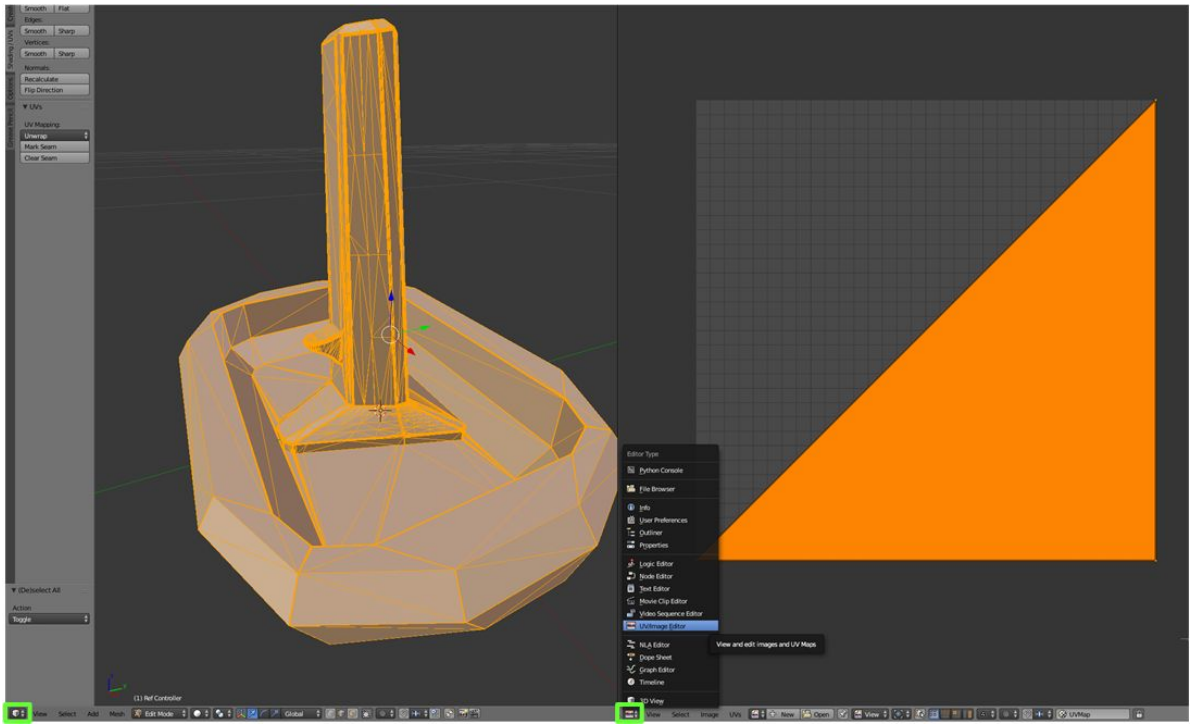
After entering edit mode:

1) Press ‘A’ to select the entire object
2) Select Shading / UVs > Unwrap > Unwrap

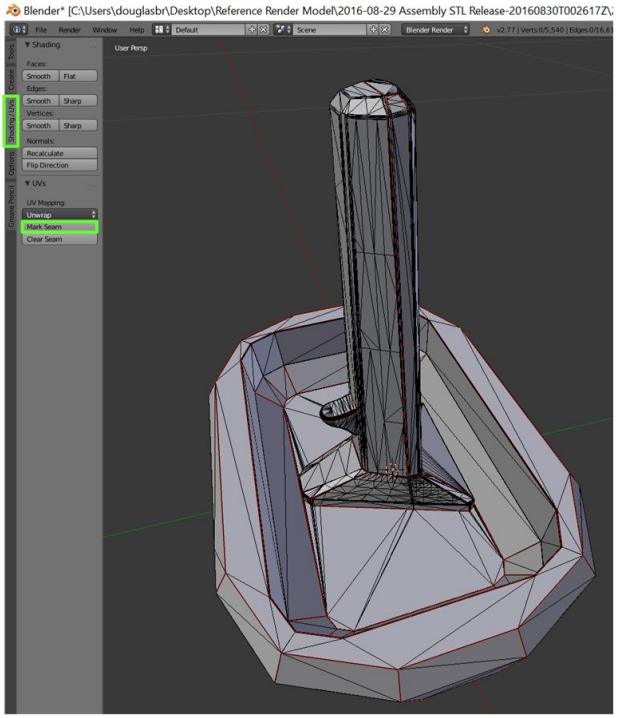


It is possible to see the render map side by side with the object, in edit mode, when the object is selected and the UV/Image Editor window is open. It is clear from the UV map displayed below (on the right), that Blender did not do a good job flattening the object.

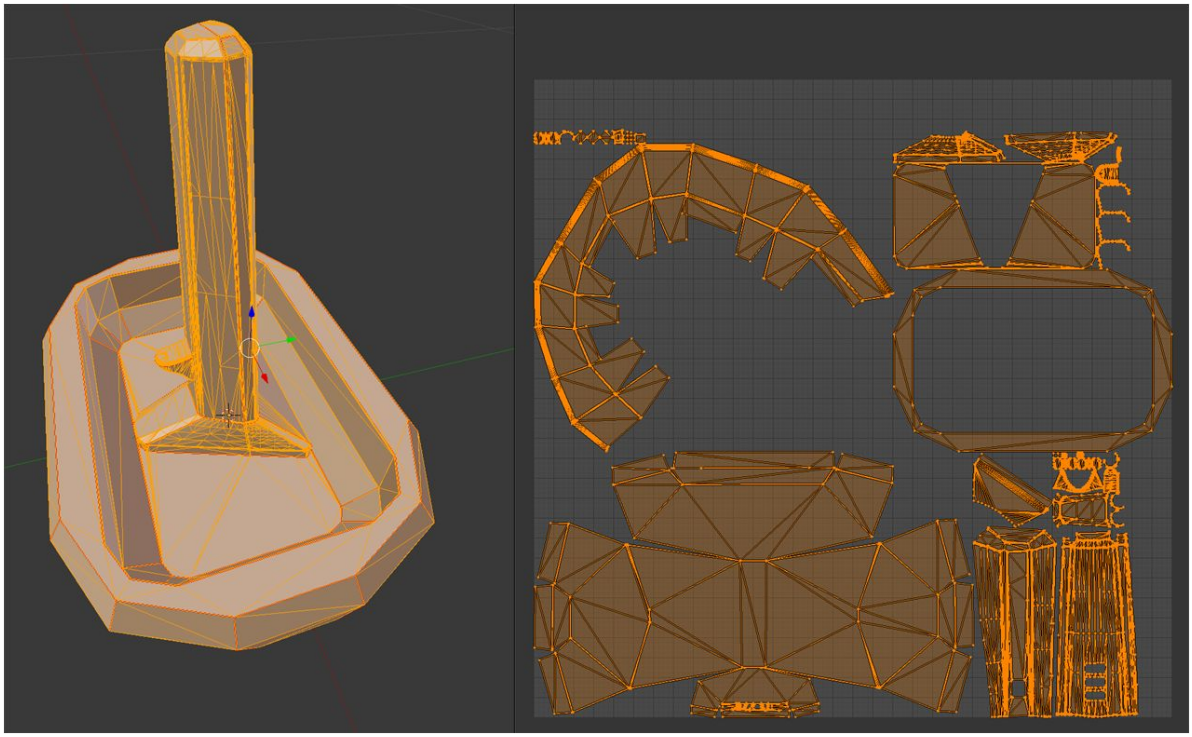
Tip: It is possible to apply a texture at this point, but it would not map onto the object in any discernable way. However, the process of creating a useful UV map is time consuming. Consider skipping to the next step to export the OBJ file and test it with a solid texture. Debug the OBJ file and test tracking, then come back to this point to continue refining the UV map and texture.



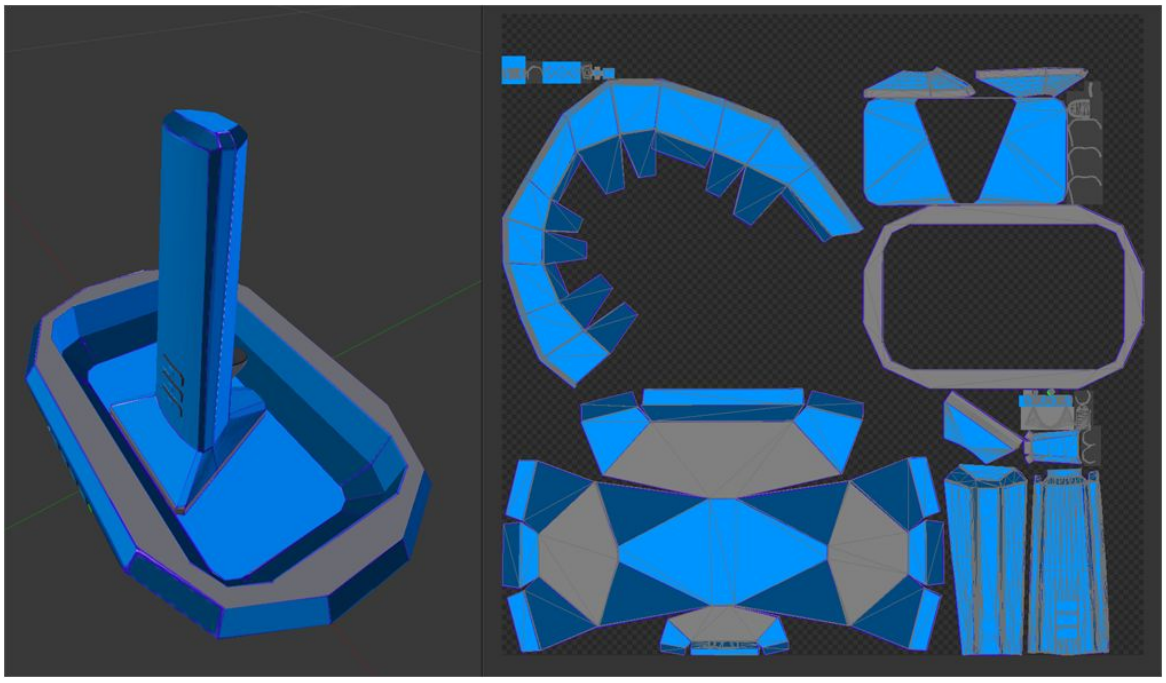
The reason that Blender could not effectively unwrap the object, is that it did not know where to cut the shape to unfold it. Mark seams on the object to tell Blender where it can cut the shape while unwrapping. Right click edges where Blender should cut the object to unfold it, and click the Mark Seam button. Edges marked as seams appear red.



After marking the seams, Blender can effectively unwrap the object as shown below.

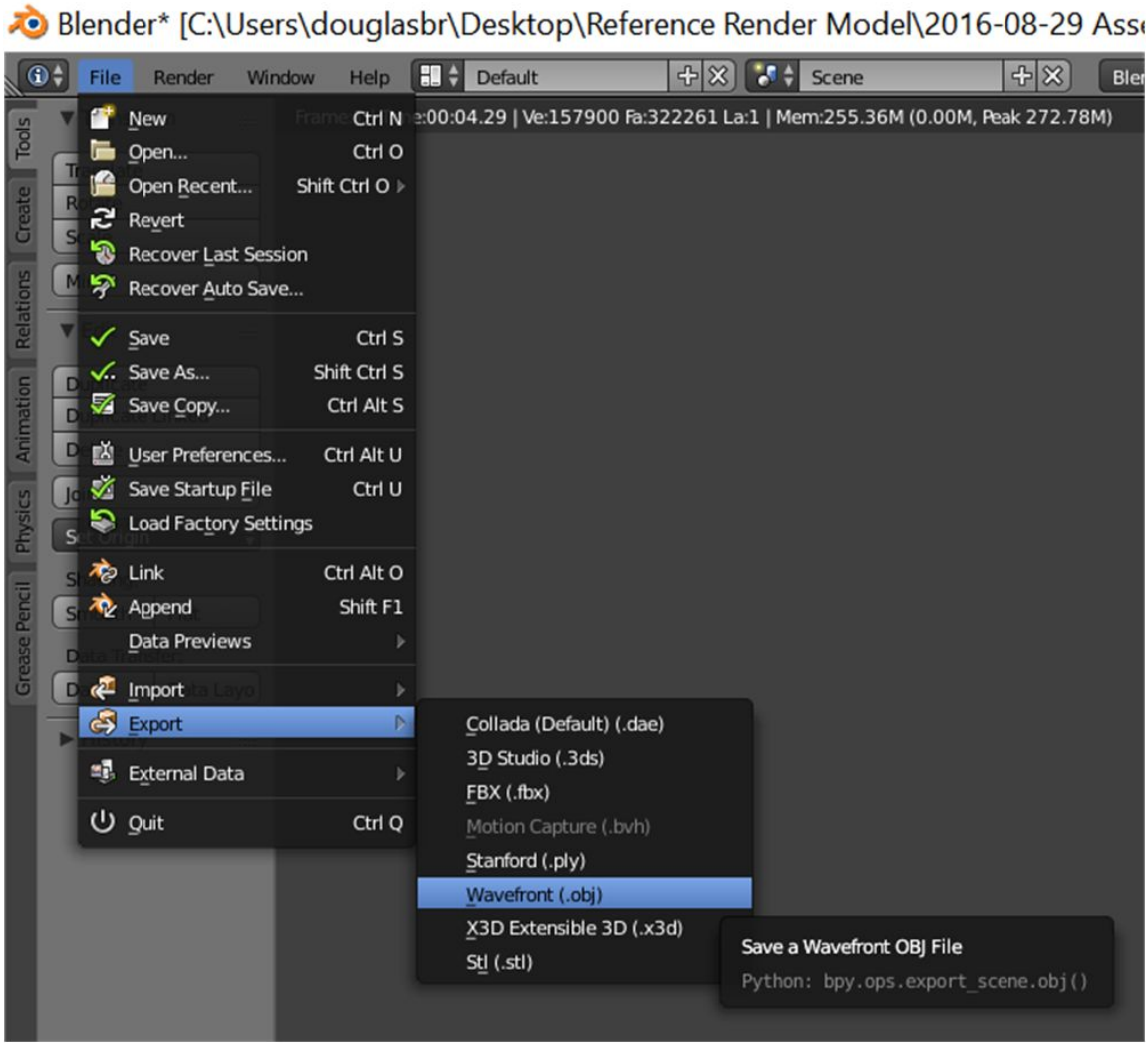


Painting over the UV map on the right, paints the corresponding region on the 3D object to the left. It is possible to paint the texture file directly in Blender, or export it to another paint program. Exporting the UV map as a PNG file, opening it in Paint.Net, adding a new layer, tracing the object contours, and filling in the polygons creates a new texture file that matches the facets of the object. It is not a pretty render model, but is effective at defining the shape.



Export the Render Model as an OBJ File

Export the Blender project as an OBJ file by selecting File > Export > Wavefront (.obj). Again, do not forget to check the export properties and set the Forward and Up options to Y Forward and Z Up.



There are now three files comprising the render model.

1. The OBJ file that defines the shape and references the material file and contains the UV map.
2. The MTL material file that defines the surface finish and references the texture file.
3. The PNG or TGA image file defining the texture for the material.

A complete render model has far more colors, textures, and capabilities. The full procedure for creating a production render model is not covered here. However, this process should be sufficient to create a render model for evaluating tracking performance or proof of concept demonstrations.

Deploying the Render Model

Adding the render model to SteamVR™ is as simple as creating a folder and copying the render model files into that directory.

Render models are stored in the directory:

```
C:\Program Files (x86)\Steam\steamapps\common\SteamVR\resources\rendermodels
```

Add a subfolder with the same name as the render model for the new controller:

```
<rendermodels>\ref_controller
```

Copy the render model files into the new directory:

```
<rendermodels>\ref_controller\ref_controller.obj
<rendermodels>\ref_controller\ref_controller.mtl
<rendermodels>\ref_controller\ref_controller.png
```

Now that SteamVR™ can find the render model, it is possible to reference the render model from the object's JSON file.

Referencing the Render Model

SteamVR™ uses the render model specified in an object's JSON file as the default render model for the object. Set the "render_model" member of the JSON file to the new render model.

If the controller already contains a calibrated JSON file, download that file from the object using lighthouse_console.

Connect to the object using lighthouse_console and download the JSON file.

```
lh>downloadconfig myController.json
```

Change the "render_model" entry in the JSON file to specify the new render model.

```
"render_model": "ref_controller",
```

Upload the new JSON file to the object using lighthouse_console.

```
lh>uploadconfig myController.json
```

Restart SteamVR™ to load the new JSON and render model. The controller should appear in VR as the new render model. However, the render model could have a problem that prevents it from appearing in VR. If the object does not appear, follow the steps described below to find the problem.

Debugging the Render Model

There are several reasons that a render model may not appear in SteamVR™. Follow these steps to identify some of the most common causes.

1) Verify that the object is booting.

If the object does not boot, the render model is never loaded. Verify that the object boots by changing the JSON file to reference a known-good render model. The render model may not look like the object, but it should boot and track. If the object boots and tracks with other render models, but fails to appear with the new render model, there may be a problem with the new render model.

2) Verify that the OBJ, MTL, and PNG/TGA files are all in the render models directory.

All three files are required before the object is rendered in SteamVR™.

3) Verify that the OBJ references the MTL file.

The OBJ file is a text file, and contains a reference to the MTL file. If the MTL file was renamed, any references in the OBJ file should be updated to match.

4) Verify that the MTL file references the texture file (PNG or TGA).

The MTL file is a text file, and contains a reference to the PNG or TGA file used as a texture. If the texture file was renamed, the reference in the MTL file should be updated to match.

5) Check the log file: vrclient_vrcompositor.txt.

Errors opening render models are reported in the vrclient_vrcompositor.txt log file. Open the log file and search for the name of the render model. Below are some common errors and their causes.

a) OBJ contained more than one shape

```
<date> - Render model from <rendermodels>\ref_controller\ref_controller.obj contained
more than one shape. We only support one.
```

This error occurs if the OBJ file was exported from a set of STL files. Try exporting the entire shape as a single STL. Then, use the single STL file to create the OBJ for the render model.

b) Only 65k vertices are supported

```
<date> - Render model <rendermodels>\ref_controller\ref_controller.obj has 313189
vertices. Only 65k are supported
```

This error occurs if the OBJ file contains too many vertices. Try starting from an STL file exported at a lower resolution, remove internal or unseen features from the model before exporting, or use the Decimate Mesh feature in Blender to reduce the mesh count. Then, export a new OBJ file with fewer vertices.

c) OBJ has 0 texture coordinates

```
<date> - Render model <rendermodels>\ref_controller\ref_controller.obj has 0 texture
coordinates, expected 82994
```

This error occurs if the OBJ file does not include a UV map. Import the OBJ file into Blender and unwrap it. Then, export a new OBJ file. Follow the process outlined above under Create the UV Map, and do not forget to check the OBJ import and export settings for Y Forward and Z up.

Once the render model appears in SteamVR™, it is possible that it may appear in the wrong orientation or position. Improper orientation or position is usually the result of the “head” variable in the JSON file.

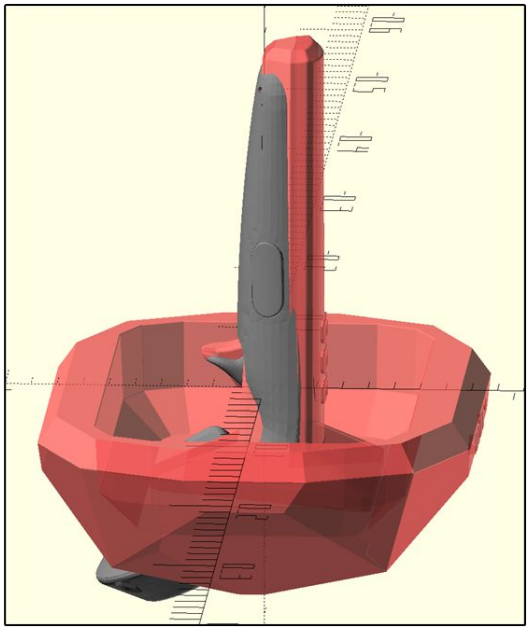
Setting the Head Variable

The “head” variable in the JSON file describes the relationship between the object’s coordinate system and the reference model’s coordinate system. Remember that the first step in the render model process was aligning the object’s STL with an existing render model. Therefore, the coordinate system of the render model and the sensors are no longer the same. The “head” variable tells SteamVR™ how the render model coordinate system relates the the coordinate system of the sensors.

To complete the head variable, we could import the render model into the original CAD and measure the relative difference between the two origins. However, that information already exists in the OpenSCAD we used to align the

render models. We could ask the question, what rotation and translation would be required to align the reference render model to the controller object? Reversing the rotation and translation above provides the answer.

```
#scale([0.001, 0.001, 0.001])
   color("orange")
      import("ref_controller_not_aligned.stl", convexity = 4);
rotate ([-3, 0, 0])
   translate ([0, 0.015, -0.040])
      #color("grey") import("Vive_Controller.stl", convexity = 4);
```



Converting these values to the head variable is simple.

1) The translation may be entered directly into the “position” value as [0, 0.015, -0.040].
2) Rotation about only the X axis means that the direction of the +X axis has not changed. The head member’s “plus_x” equals [1, 0, 0].
3) The rotation does change the direction of the +Z axis. The render model’s +Z axis is now pointing slightly in the object’s +Y direction. The value for “plus_z” equals [0, sin(3), cos(3)], which is entered into the JSON as [0, 0.05233595624, 0.99862953475].

The final head variable for the JSON file:

```
"head": {
        "plus_x": [1, 0, 0],
        "plus_z": [0, 0.05233595624, 0.99862953475],
        "position": [0, 0.015, -0.040]
    }
```

Once the “head” variable is set, upload the new JSON file to the object using lighthouse_console.

Summary

The process of creating a render model is lengthy and requires some initial planning. Remember the rules for render models when generating the original STL file. It should be a single object with a minimum of meshes. Aligning the new render model to the Vive controller's render model is the best way to create a great experience in all SteamVR™ applications. Once the STL is aligned, adding the material and texture is a simple process. However, marking all the seams and exporting a useful UV map is a time consuming process. Try creating a render model OBJ with a solid texture to verify that the OBJ meets SteamVR™'s criteria before taking the time to mark seams and unwrap the UV map. Once the render model is functional, marking the seams and exporting a useful UV map helps define the shape of the object with a more appropriate texture. Create a subdirectory for the render model in SteamVR™'s render model directory and copy the object, material, and texture files into that directory. Also, add a reference to that directory in the object's JSON file. Restart SteamVR™ to load the new JSON and render model and debug any problems. Finally, set the "head" variable in the JSON file to orient object to the render model. Once these steps are complete it should be easy to manipulate the controller in VR, evaluate tracking performance and test the controller in different applications.

Complete OpenSCAD Files

The following OpenSCAD file renders the STL file of a new controller object in the same space as the Vive controller's render model, and aligns the new controller's render model to the Vive controller's render model.

```
rotate ([3, 0, 0])
   translate ([0, -0.015, 0.040])
      scale([0.001, 0.001, 0.001])
         color("orange")
            import("ref_controller_not_aligned.stl", convexity = 4);

#color("grey")
   import("Vive_Controller.stl", convexity = 4);
```

The following OpenSCAD file renders the STL file of a new controller object in the same space as the Vive controller's render model, and aligns the Vive controller's render model with the new controller's render model. The values from this alignment are used to set the "head" variable in the JSON.

```
#scale([0.001, 0.001, 0.001])
   color("orange")
      import("ref_controller_not_aligned.stl", convexity = 4);

rotate ([-3, 0, 0])
   translate ([0, 0.015, -0.040])
      color("grey")
         import("Vive_Controller.stl", convexity = 4);
```