渲染模型

SteamVR™ Tracking

简介

SteamVR™ 渲染模型是用于在 VR 中代表控制器的 3D 模型。默认情况下，对象的 JSON 文件会参考一个与控制器对象相似的渲染模型。拥有一个与对象相似的渲染模型对于最终用户来说非常有用，因为这在 VR 中能够大大简化对象的拾取与处理。应用程序加载后，它会用更合适的模型覆盖该渲染模型。一个游戏的渲染模型可能会显示为手臂、枪械、弓、漆刷或应用程序开发人员所想象的任何其他东西。因为第三方也在开发控制器的模型，因此所有控制器务必要在坐标系上保持一致。如果所有渲染模型都在同一坐标系上创建并对齐，替换渲染模型时就不用更改模型与用户手臂之间的基本关系。

本文档将介绍创建一个简单渲染模型以评估定位时的要求和流程。本例用到的 OpenSCAD 和 Blender 是两款免费软件程序。自己拥有实体建模和 3D 图形软件的设计师可使用自己选择的工具执行类似的程序。最后，设计生产渲染模型时，还需要本文未提及的更多详细信息。但是，借助下面所述的流程，工程师应能够地掌握到构建用于故障排除和概念验证目的的渲染模型时所需的基本知识。

创建渲染模型

渲染模型是用于在 VR 中代表控制器的 3D 模型。一个渲染模型至少包含三个文件：Wavefront (OBJ) 对象文件文件以及材料文件 (MTL) 和纹理文件（TGA 或 PNG）。创建渲染模型的流程分为几个步骤。首先，从设计 CAD 导出模型的 STL 文件。然后，该 STL 模型必须与现有控制器渲染模型对齐。然后将对齐后的 STL 模型导入一个 3D 图形程序以添加材料、UV 图和纹理。最后，将完成的渲染模型导出为一个 OBJ 文件。

导出控制器对象 STL

对象形状可由机械工程师使用 3D 实体建模软件包定义。将此形状导出为一个 STL 文件，这是创建渲染模型的第一步。但是，STL 的导出方式对之后的流程有重大的影响。

为渲染模型导出 STL 文件时，务必牢记以下规则：

1) 渲染模型 OBJ 文件所包含的顶点数不得超过 65,000 个。
2) 渲染模型 OBJ 文件必须仅包含一个对象

要将顶点数保持最小值，需要导出一个仅描述对象外表面的 STL 文件。该表面应包括按钮、触控板和其他外部特征，以提供渲染模型的完整外观和感受。但是，装配凸缘、挡边和对齐杆柱等内部功能会增加不必要的细节，导致最终渲染模型中有太多的顶点。另外，注意 CAD 程序中的导出属性。也许有办法能够降低网格的分辨率，以减少创建的顶点数。进行实验以找出分辨率与美学之间的最佳平衡点。

另外，在导出 STL 时，将所有表面导出为单个 STL 文件。当单个 STL 对象被转换为一个 OBJ 时，此 OBJ 应仅包含一个对象。

一旦从 CAD 导出控制器的 STL 模型，下一步就是将其与现有控制器渲染模型进行对齐。

© 2016 Valve Corporation。保留所有权利。

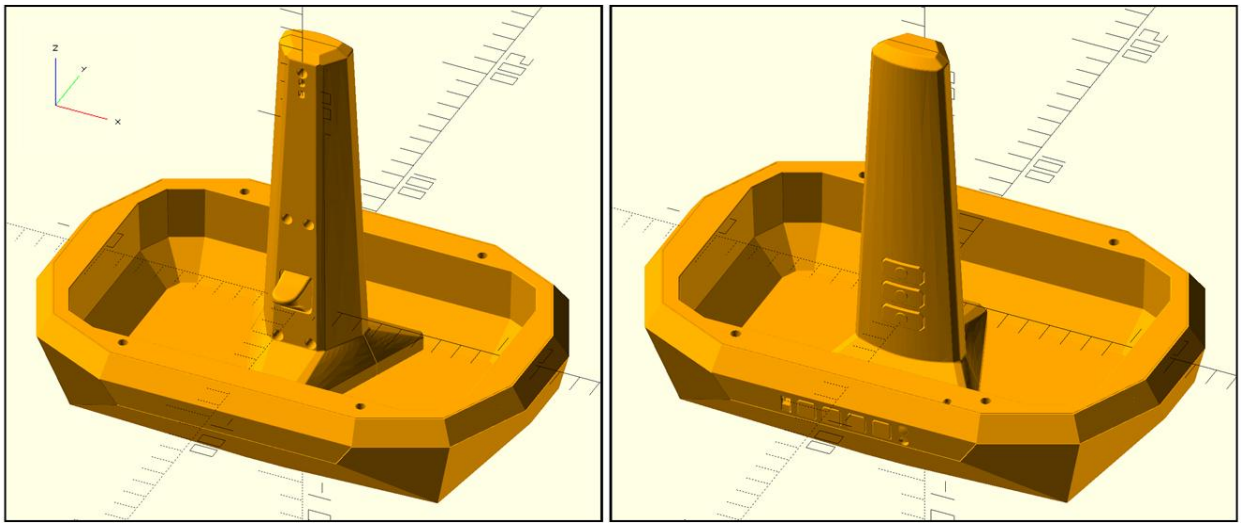
导入控制器对象 STL 文件

将新控制器的渲染模型与现有控制器进行对齐非常重要。不同的应用程序会用自己的渲染模型覆盖现有渲染模型。渲染模型完成对齐后，在替换为另一个渲染模型时，新控制器对象仍会感觉正确。

在本文档中，采用了 OpenSCAD 来导出控制器对象并将其与 HTC Vive 控制器渲染模型进行对齐。使用 OpenSCAD 仅仅是为了方便，因为它是免费的，并且可以轻松将命令存档。这些步骤也可在其他实体建模软件包中执行。

首先，将定义控制器对象的 STL 文件导入到 OpenSCAD 中。

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



上述 STL 文件的导出单位为毫米，但渲染模型的目标单位是米。使用 scale 命令将毫米单位转换为米。

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

将参考控制器渲染模型转换为 STL

目标是将新对象与来自高度支持的控制器的参考渲染模型对齐。大多数应用程序开发控制器时，都会把针对 HTC Vivi 开发的控制器作为依据。因此，强烈建议将新控制器渲染模型与 HTC Vive 控制器的渲染模型相对齐。要将 Vive 控制器模型导入 OpenSCAD，此模型必须为 STL 格式。Blender 可用于根据 Vive 控制器的 OBJ 渲染模型创建 STL 文件。

渲染模型存储在以下目录中，渲染模型在本文档别处称为 <rendermodels>。

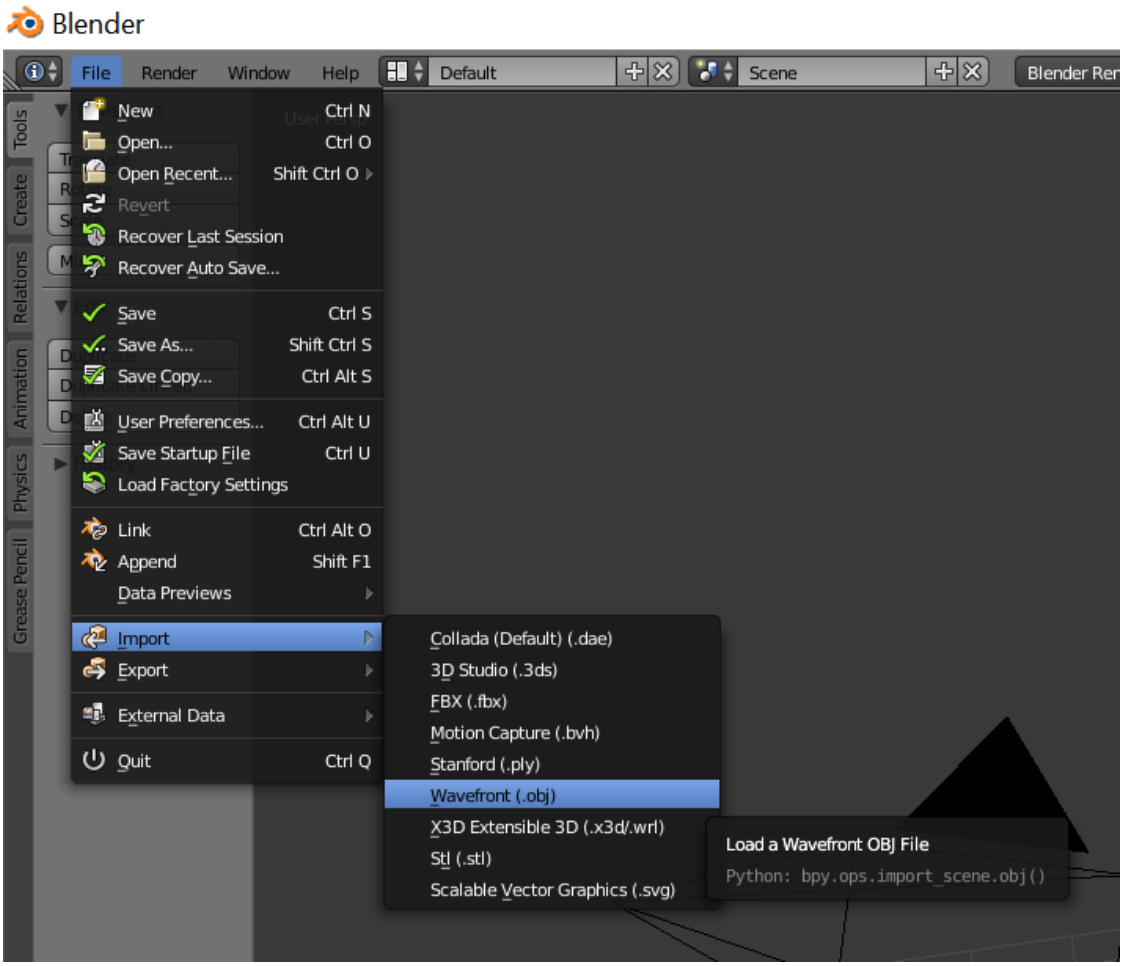
```
C:\Program Files (x86)\Steam\steamapps\common\SteamVR™\resources\rendermodels
```

HTC Vive 渲染模型在以下子目录中：

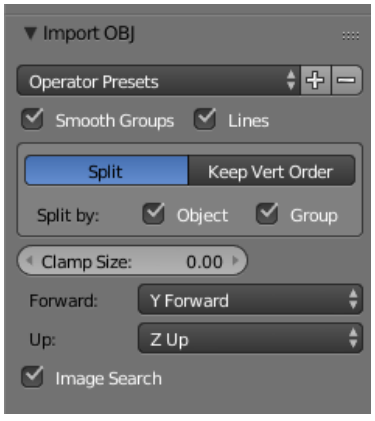
```
<rendermodels>\vr_controller_vive_1_5
```

控制器的顶级 OBJ 文件名为 vr_controller_vive_1_5.obj。

通过选择“文件”>“导入...”>“Wavefront (.obj)”将 Vive 渲染模型导入到 Blender，并浏览至 vr_controller_vive_1_5.obj。一定要将导入设置配置为“Y 向前”和“Z 向上”。



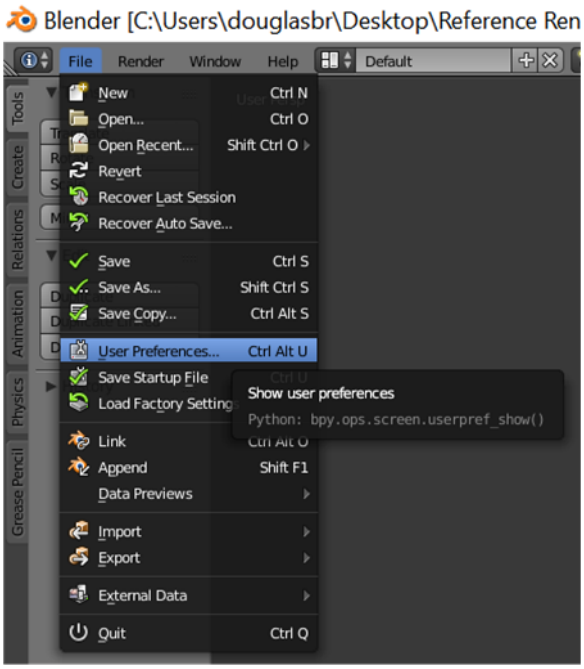
浏览该文件时，导入设置显示在窗口的左下角。将导入轴设置为“Y 向前”和“Z 向上”。否则，Blender 会在导入时旋转 OBJ 文件。

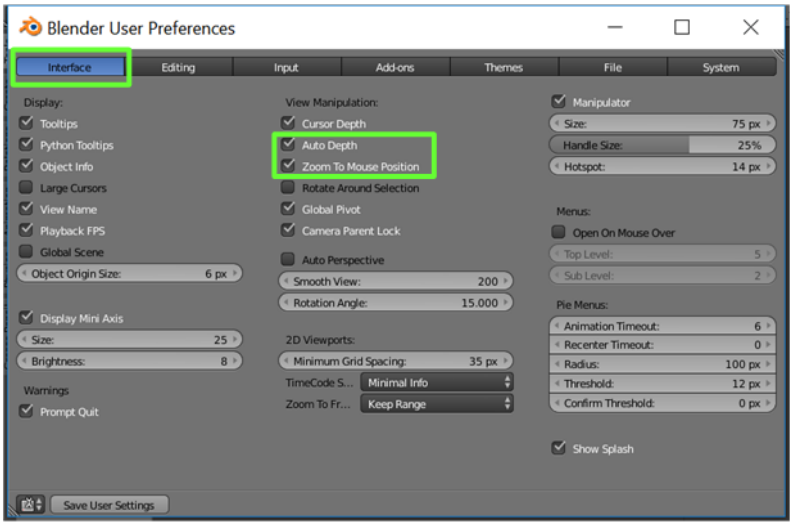


注意：在从 Blender 导入或导出 OBJ 时，务必验证导入或导出设置为：向前 = Y 向前，向上 = Z 向上。这些设置不是 OBJ 文件在 Blender 中的默认设置。

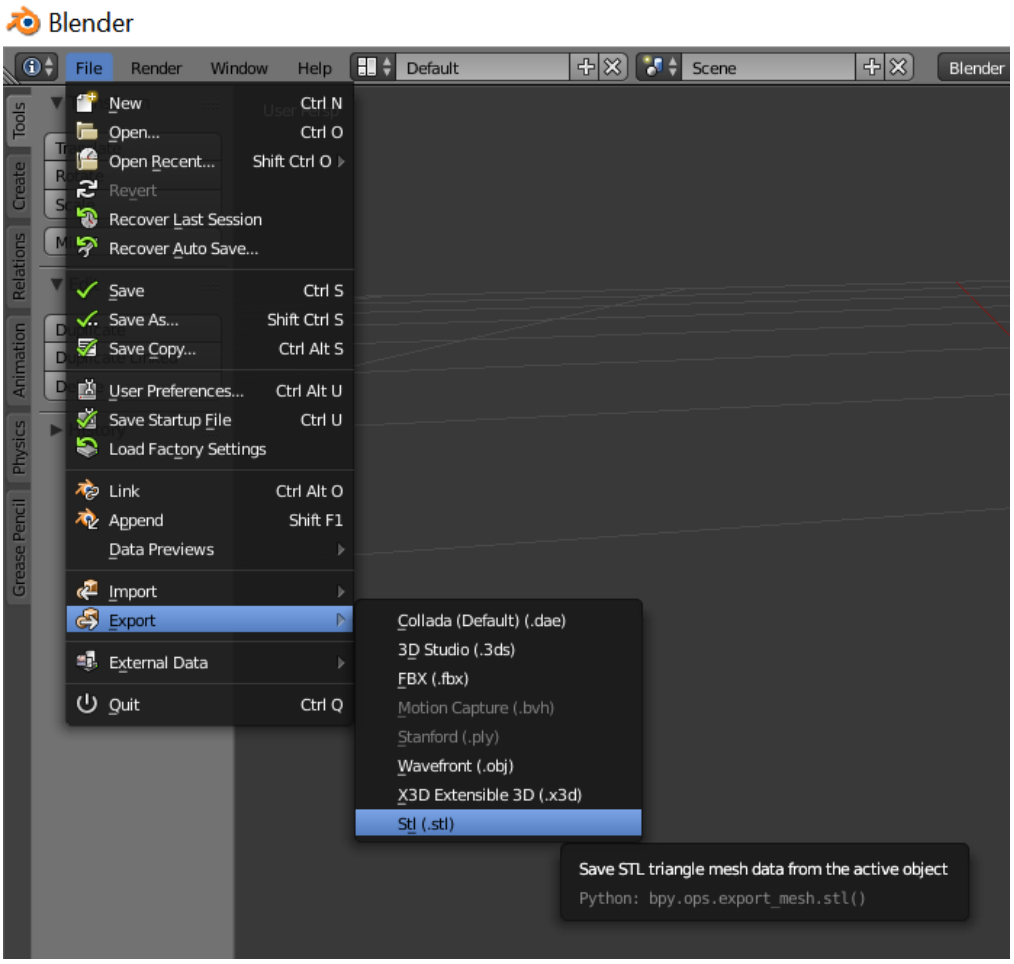
一旦导入 OBJ 文件，渲染模型会显示在原点处。

提示：Blender 有很多缩放选项。这对于在“界面”选项卡上的“Blender 用户首选项”对话框中打开“自动深度”和“缩放至鼠标位置”非常有帮助。





通过选择“文件”>“导出...”>“Stl (.stl)”，将 Vive 控制器的渲染模型导出为一个 STL 文件。将文件另存为 Vive_Controller.stl。



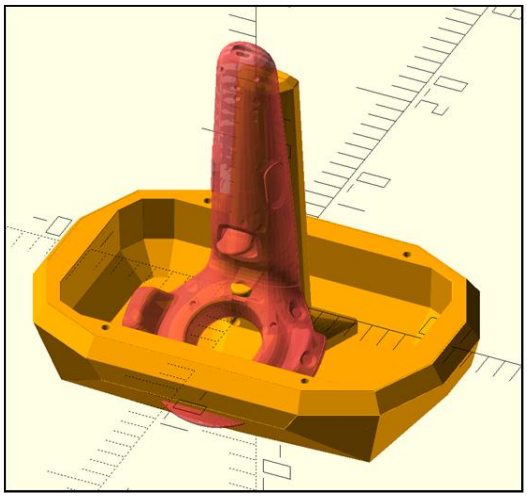
现在已经得到 Vive 控制器的 STL 文件，可将该文件导入 OpenSCAD，用作对齐控制器对象渲染模型位置的基准。

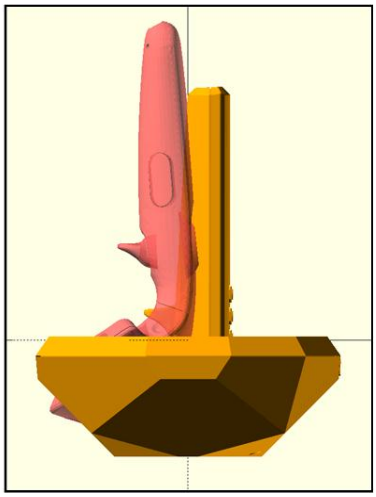
导入参考控制器渲染模型 STL 文件

可通过添加以下代码，将新 Vive 控制器 STL 文件以及控制器对象的 STL 文件导入到 OpenSCAD 中。

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

提示：在 Vive 渲染模型前面使用 # 字符，使其透明显示。能够透视对象意味着更容易对齐两个相交的对象。

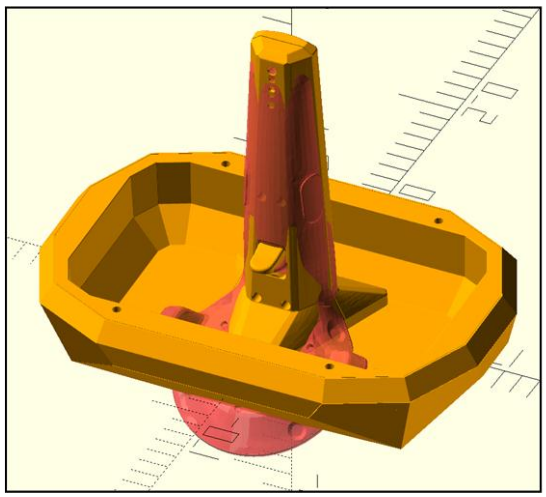


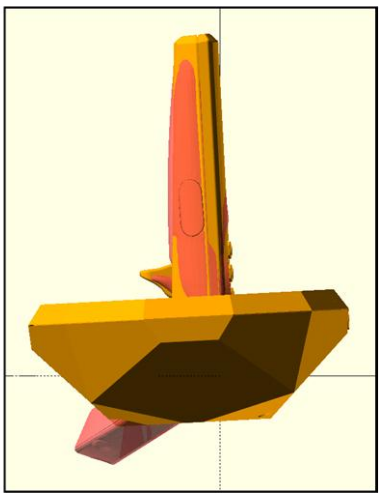


将对象与参考控制器模型对齐

从上述图例中，很明显可以看到新对象已非常接近 Vive 控制器。对新对象进行平移以对齐触发器，绕着 X 轴略微旋转以对齐把手。

```
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);
```

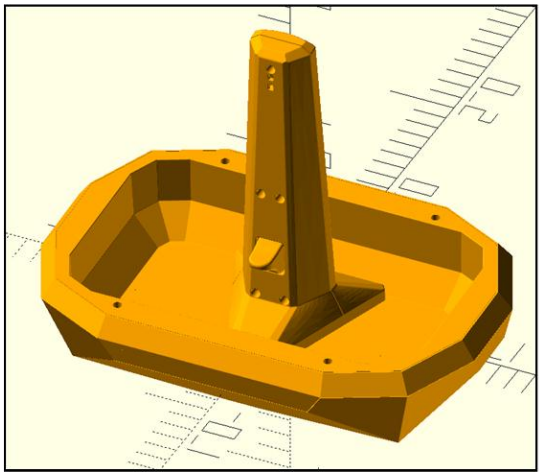


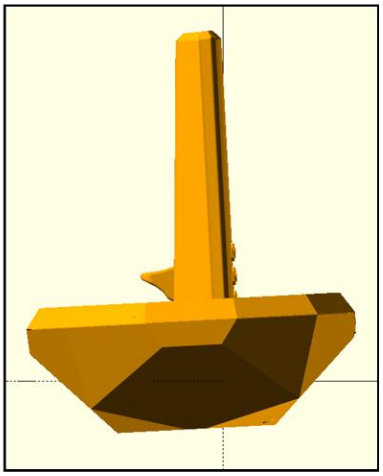


导出对象渲染模型 STL 文件

待新对象与 Vive 渲染模型对齐后，即可导出新对象渲染模型的 STL 文件。在 OpenSCAD 中，可通过标上感叹号来挑选出几何形状。

```
!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);
```

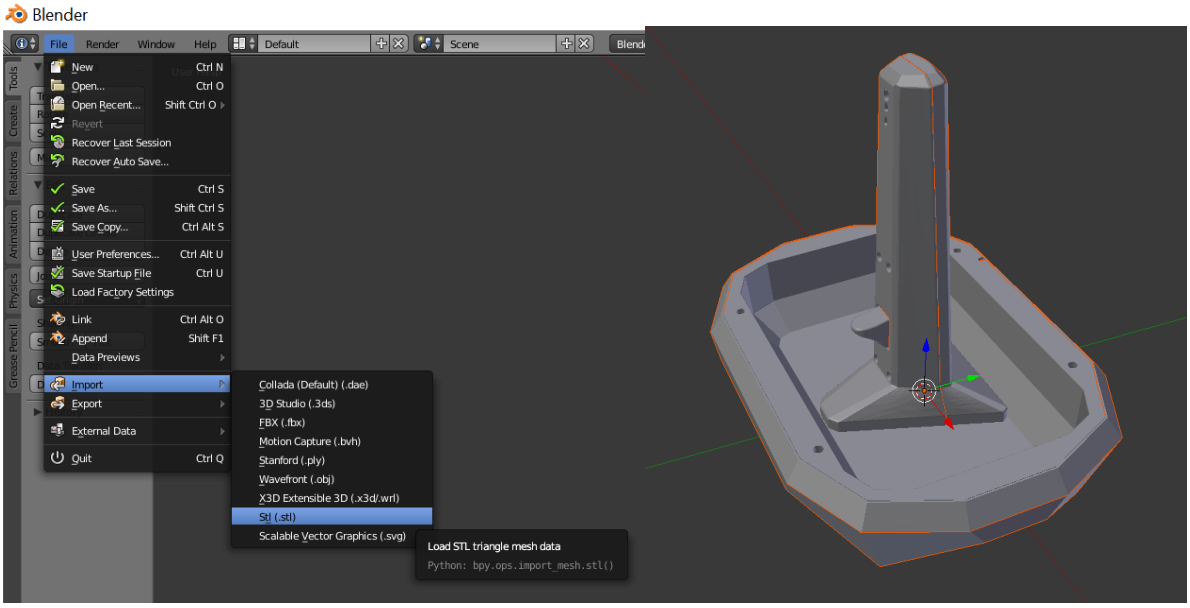




按 F6 构建对象。该对象会在构建过程中消失一段时间。构建完成后，对象会以黄色重新显示。选择“文件”>“导出”>“导出为 STL...” 使用一个可以表示新 STL 文件与参考对齐的文件名。

将匹配的控制器对象 STL 文件导入 Blender

在 Blender 中，选择“文件”>“导入”>“Stl (.stl)”。



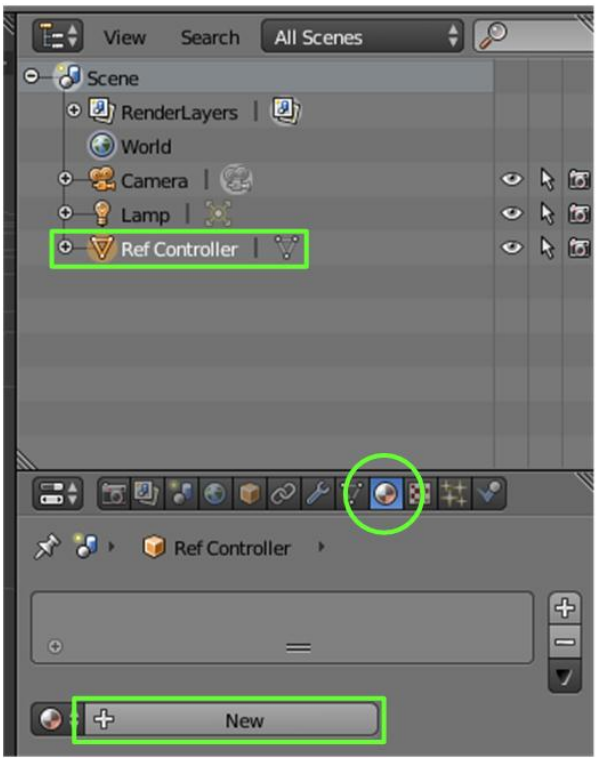
导入的 STL 文件定义了渲染模型形状，但还必须添加材料和纹理。

添加材料

渲染模型需要材料和纹理，以在 SteamVR™ 中显示。OBJ 文件必须参考一个材料 (MTL) 文件。

添加材料：

1. 在 Blender 窗口的右上角找出已导入的 STL 文件。
2. 选择该 STL 文件。
3. 单击材料按钮。
4. 单击“新建”并为材料命名。

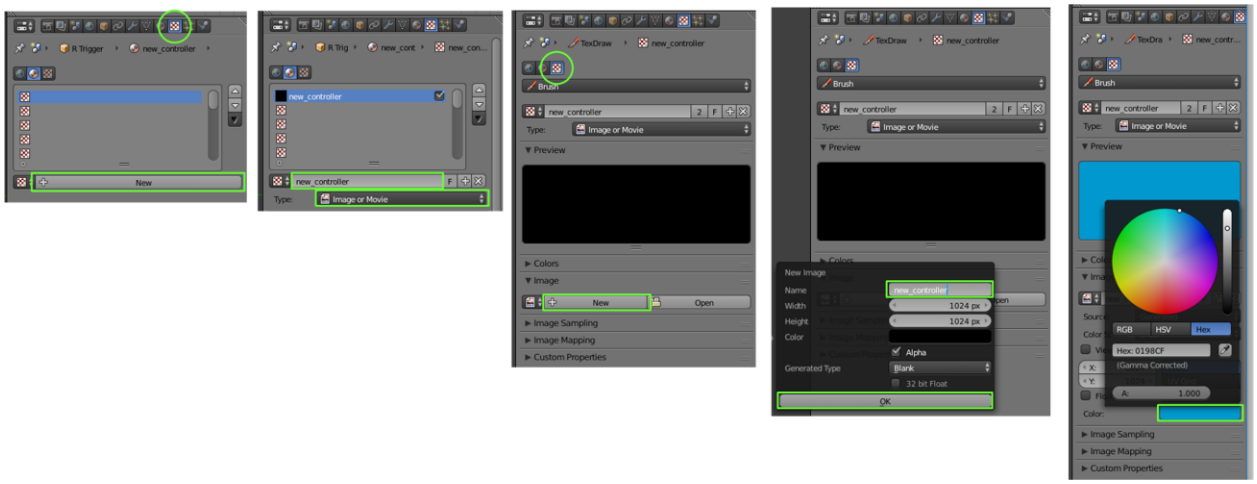


添加纹理

材料需要纹理，方可在 SteamVR™ 中显示。纹理是图像文件（TGA 或 PNG），应用于对象表面。材料文件会引用纹理文件。

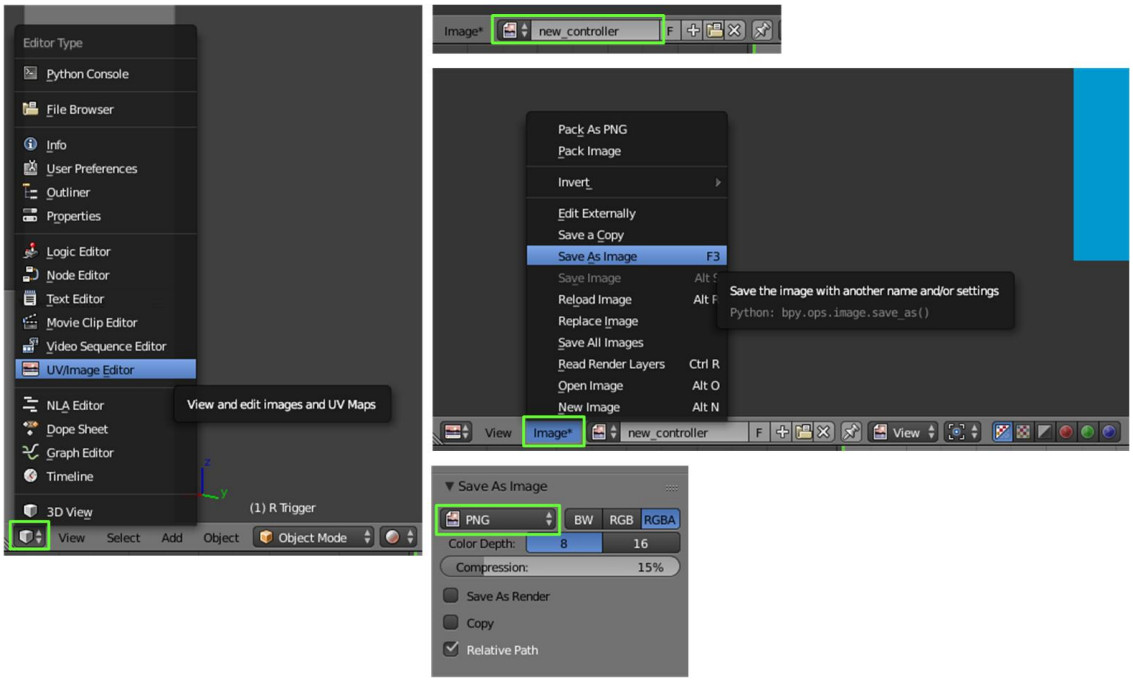
为材料创建纹理：

1. 选择纹理按钮。
2. 在“图像”面板中，单击“新建”。
3. 为图像命名，并单击“确定”。
4. 为图像选择颜色。



将刚创建的简单纹理另存为一个图像文件：

1. 更改为 UV/图像编辑器视图。
2. 选择该新纹理。
3. 单击“图像”和“另存为...”
4. 在左下角选择所需的文件类型，并保存该图像文件。



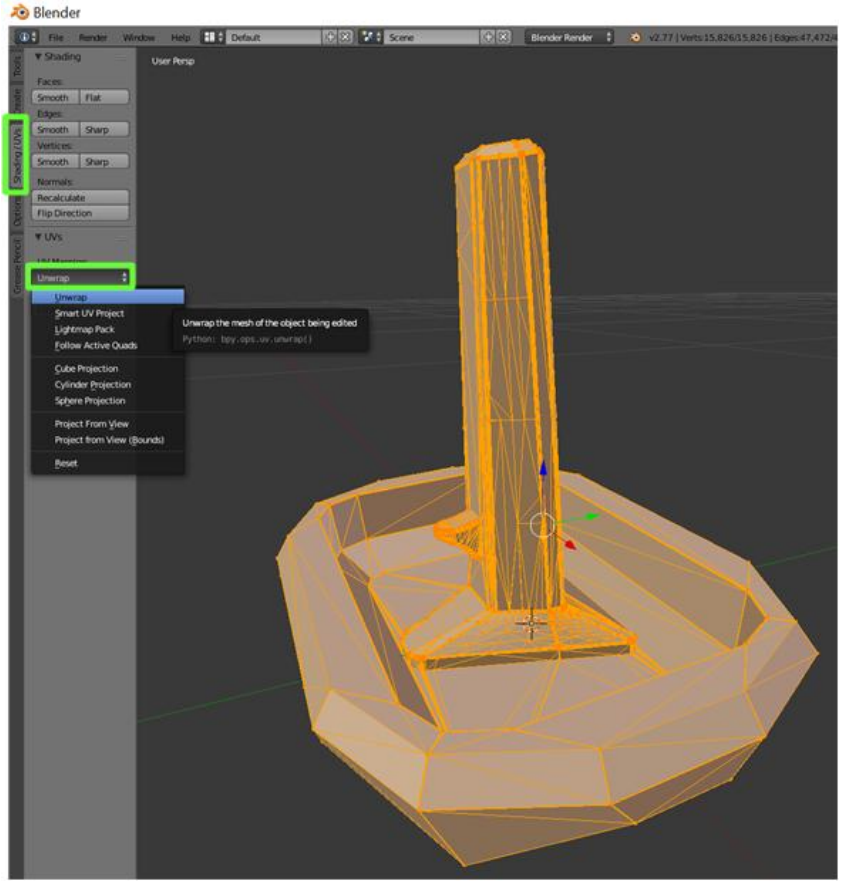
创建 UV 图

纹理是一个必须映射至 3D 形状的 2D 图像，这就需要一个 UV 图。Blender 可以通过展开对象，从现有 STL 创建一个 UV 图。展开是尝试将 3D 图像扁平化的操作。想象一下，拿一个纸板盒，并试着将其展开并平铺到地板上。当对一个复杂的形状执行此操作时，结果可能会一团糟，但我们可以告诉 Blender 在何处切割形状以将其展开并平铺至一个二维表面上。

UV 图在编辑模式下创建：

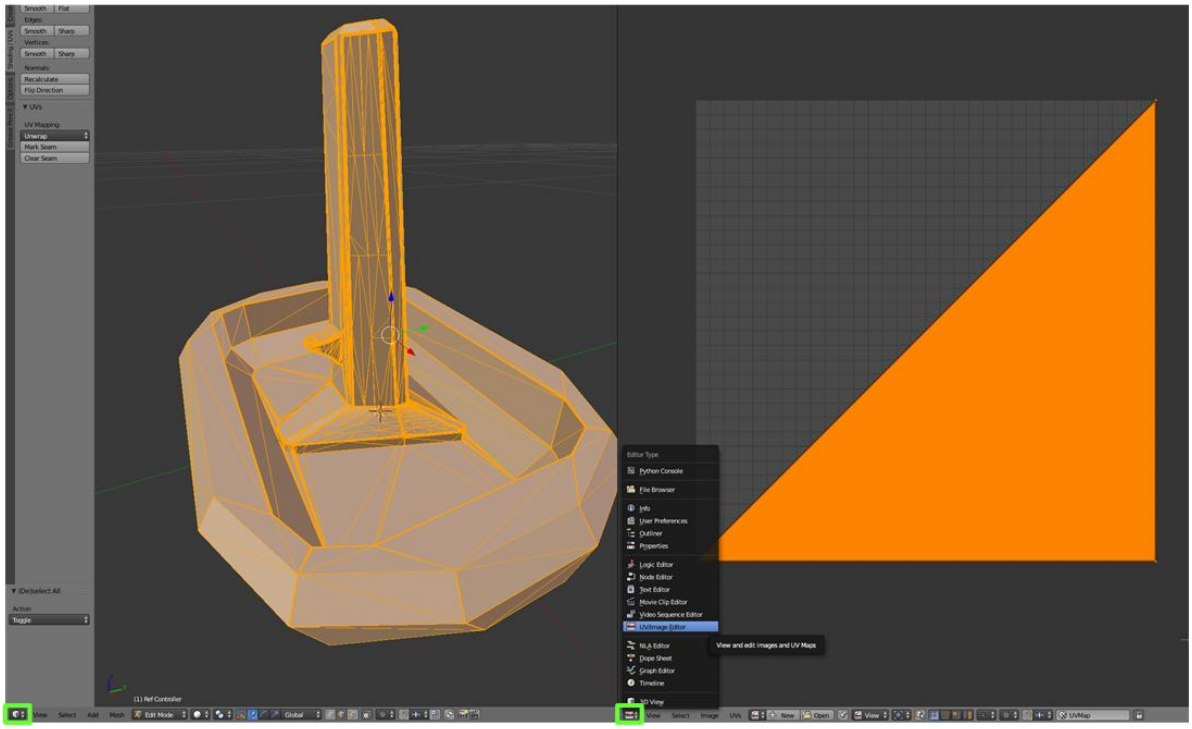
进入编辑模式后：

1) 按“A”以选择整个对象
2) 选择“遮蔽”/“UV”>“展开”>“展开”

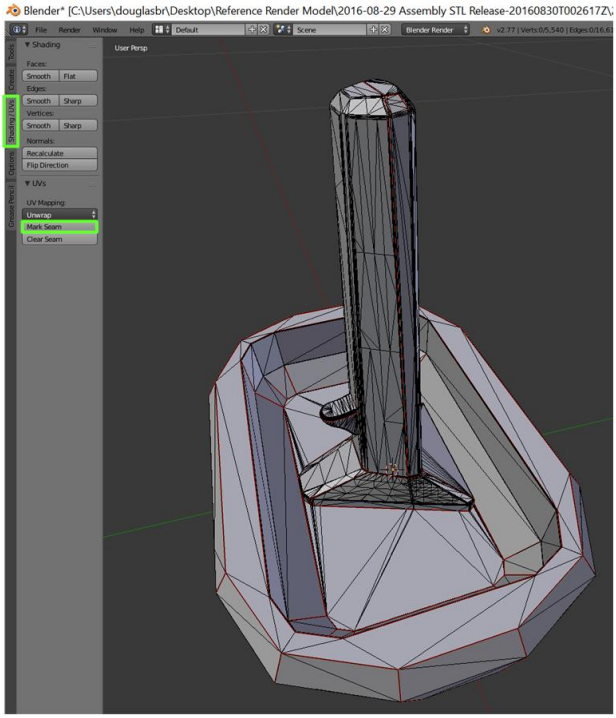


选择了对象并打开了“UV/图像编辑器”窗口后，可在编辑模式下并排查看渲染图和对象。从下方（右侧）显示的 UV 图中，可以很明显的看到 Blender 并未很好地将对象变平。

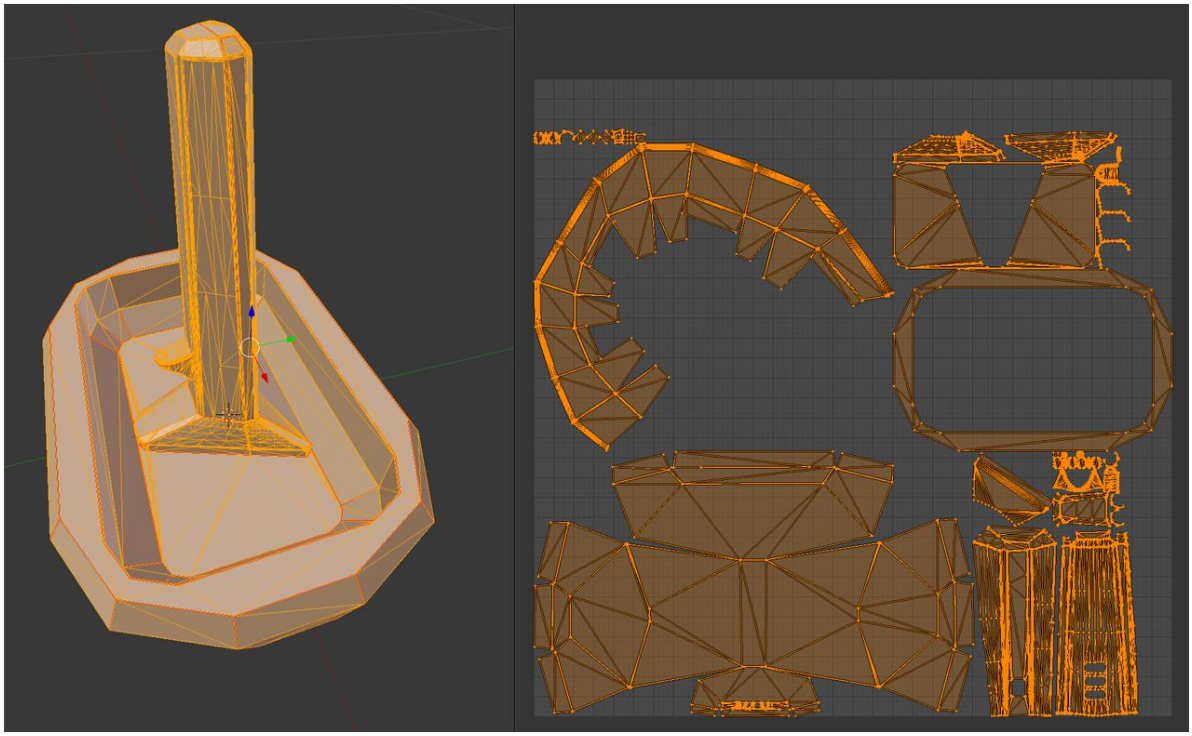
提示：此时可以应用纹理，但应以不会被分辨出的方式将纹理映射到对象上。但是，创建一个行之有效的 UV 图是一个较为耗时的过程。考虑跳到下一步以导出 OBJ 文件，并用实体纹理对其进行测试。调试 OBJ 文件并测试定位，然后返回之前继续改进 UV 图和纹理。



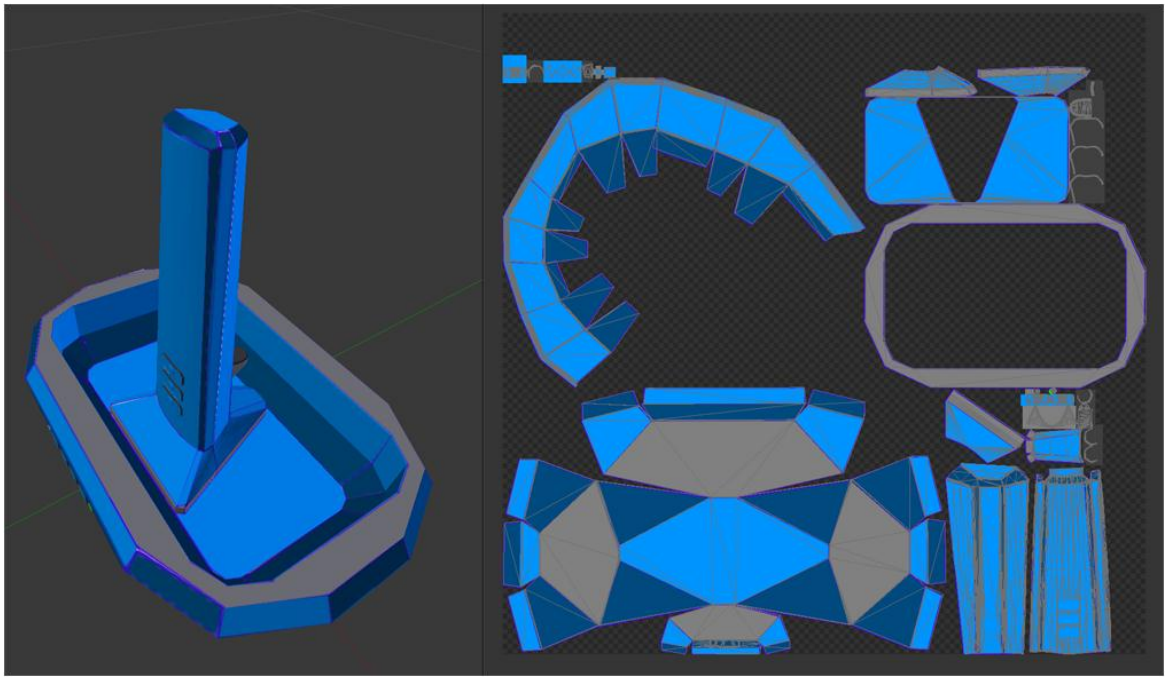
Blender 无法有效展开对象的原因是，它不知道在何处切割形状以将其展开。在对象上标上接缝，以告知 Blender 在展开对象时可在何处切割形状。右键单击 Blender 展开对象时的切割边棱，并单击“标记接缝”按钮。标记为接缝的边缘显示为红色。



标记接缝后，Blender 可以有效展开对象，如下所示。

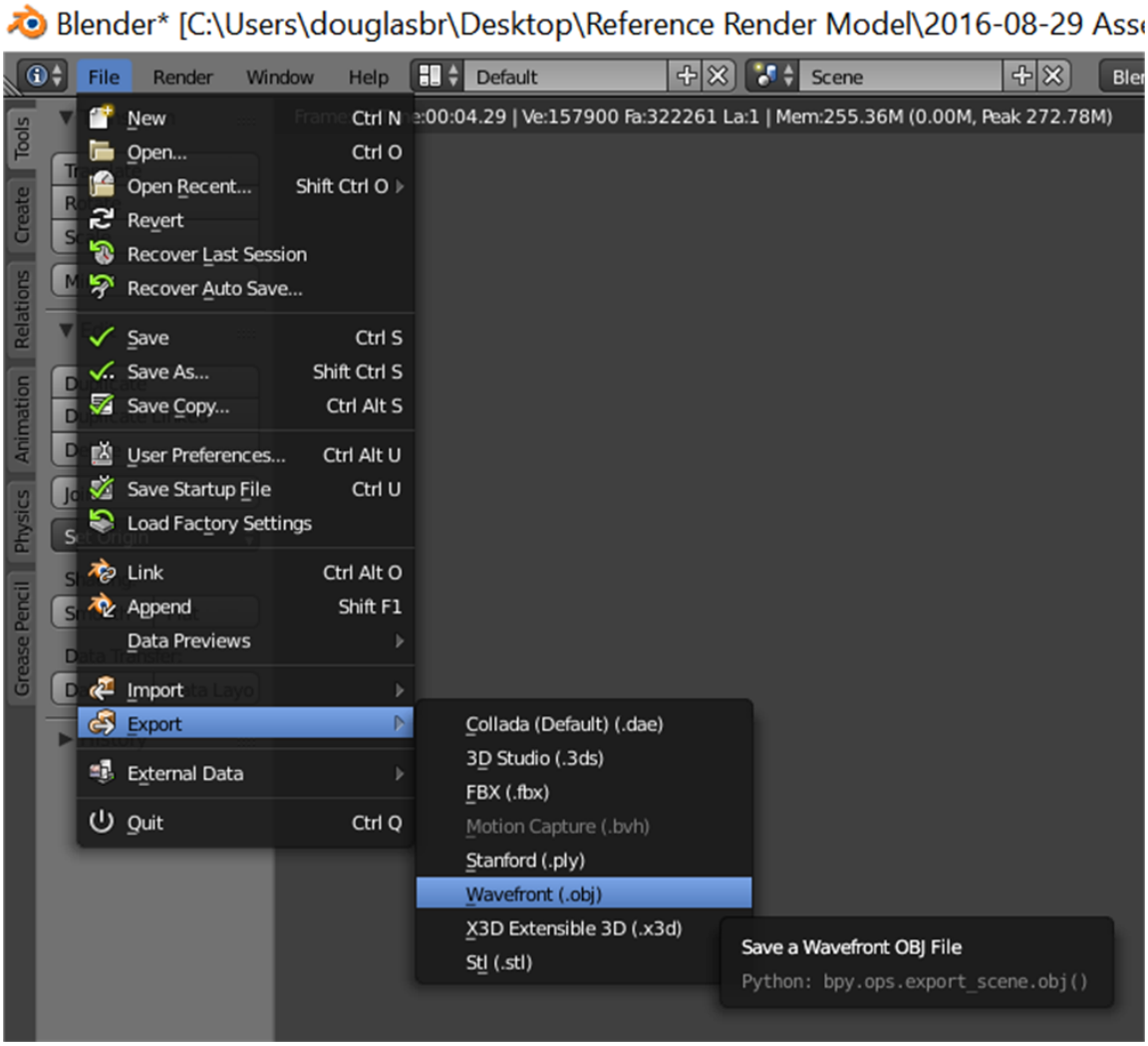


在右侧的 UV 图上喷涂，然后在左侧 3D 对象上的对应区域上喷涂。可以直接在 Blender 中喷涂纹理文件，或将其导出到另一个喷涂程序。将 UV 图导出为一个 PNG 文件，在 Paint.Net 中打开此文件，添加一个新层，跟踪对象轮廓，并填充多边形，从而创建一个与对象的面相匹配的新纹理文件。这个渲染模型不美观，但可以有效定义形状。



将渲染模型导出为 OBJ 文件

选择“文件”>“导出”>“Wavefront (.obj)”，将 Blender 项目导出为 OBJ 文件。再说一次，不要忘记检查导出属性，并将“向前”和“向上”选项设置为“Y 向前”和“Z 向上”。



现在，构成渲染模型的三个文件均已存在。

1. 用于定义形状、引用材料文件且包含 UV 图的 OBJ 文件。
2. 用于定义表面加工并引用纹理文件的 MTL 材料文件。
3. 用于定义材料纹理的 PNG 或 TGA 图像文件。

完整的渲染模型所拥有的颜色、纹理和功能要比这丰富很多。本文并未提供创建生产渲染模型的完整程序。但是，此流程应足以创建一个用于评估定位性能或用作方案验证证据的渲染模型。

部署渲染模型

将渲染模型添加到 SteamVR™ 非常简单，只需创建一个文件夹并将渲染模型文件复制到该目录下。

渲染模型存储在以下目录中：

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

添加一个子文件夹，名称与新控制器的渲染模型相同：

```
<rendermodels>\ref_controller
```

将渲染模型文件复制到新目录下：

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

现在 SteamVR™ 已能找到该渲染模型，我们可从对象的 JSON 文件引用该渲染模型了。

引用渲染模型

SteamVR™ 使用对象的 JSON 文件中指定的渲染模型作为该对象的默认渲染文件。将该 JSON 文件的“render_model”成员设置为新渲染模型。

如果控制器已包含一个已校准 JSON 文件，使用 lighthouse_console 从对象下载该文件。

使用 lighthouse_console 连接到该对象，并下载该 JSON 文件。

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

更改该 JSON 文件中的“render_model”条目以指定新渲染模型。

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

使用 lighthouse_console 将新 JSON 文件上载到对象。

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

重启 SteamVR™ 以加载新 JSON 和渲染模型。控制器应在 VR 中显示为新渲染模型。但是，渲染模型可能会出现问题，以致无法在 VR 中显示。如果对象不显示，请按下述步骤操作以找出问题。

调试渲染模型

渲染模型无法在 SteamVR™ 中显示，可能的原因有几种。请按这些步骤操作，以辨别一些最常见的原因。

1) 验证对象是否启动。

如果对象未启动，就绝不会加载渲染模型。通过更改 JSON 文件以引用已知的良好渲染模型，验证对象能否启动。渲染模型可能与对象不相像，但它应该能够启动和定位。

如果对象启动，并可使用其他渲染模型定位，但未能使用新渲染模型显示，则该新渲染模型可能有问题。

2) 验证 OBJ、MTL 和 PNG/TGA 文件都在渲染模型目录中。

需要拥有所有这三个文件，方可在 SteamVR™ 中渲染对象。

3) 验证 OBJ 参考 MTL 文件。

OBJ 文件是一个文本文件，包含对 MTL 文件的引用。如果 MTL 文件被重命名，应更新 OBJ 文件中的所有引用文件以相互匹配。

4) 验证 MTL 文件是否引用了纹理文件（PNG 或 TGA）。

MTL 文件是一个文本文件，包含对用作纹理的 PNG 或 TGA 文件的引用。如果纹理文件被重命名，应更新 MTL 文件中的所有引用文件以相互匹配。

5) 检查日志文件 vrclient_vrcompositor.txt。

Vrclient_vrcompositor.txt 日志文件中报告打开渲染模型时出现的错误。打开该日志文件并搜索渲染模型的名称。以下是一些常见错误及其导因。

a) OBJ 包含多个形状

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

如果 OBJ 文件是导自一组 STL 文件，将出现此错误。尝试将整个形状导出为单个 STL 文件。然后，使用该 STL 文件创建渲染模型的 OBJ。

b) 最多仅支持 65000 个顶点。

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

如果 OBJ 文件包含太多顶点，将出现此错误。尝试从导出分辨率较低的 STL 文件着手，在导出之前从模型中移除内部或看不见的特征，或使用 Blender 中的网格功能减少网格数。然后，导出具有较少顶点的新 OBJ 文件。

c) OBJ 拥有 0 个纹理坐标

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

如果 OBJ 文件不包括 UV 图，将出现此错误。将 OBJ 文件导入 Blender 并展开。然后，导出新的 OBJ 文件。按照上述“创建 UV 图”中所述流程操作，并且不要忘记检查 OBJ 的“Y 向前”和“Z 向上”导入和导出设置。

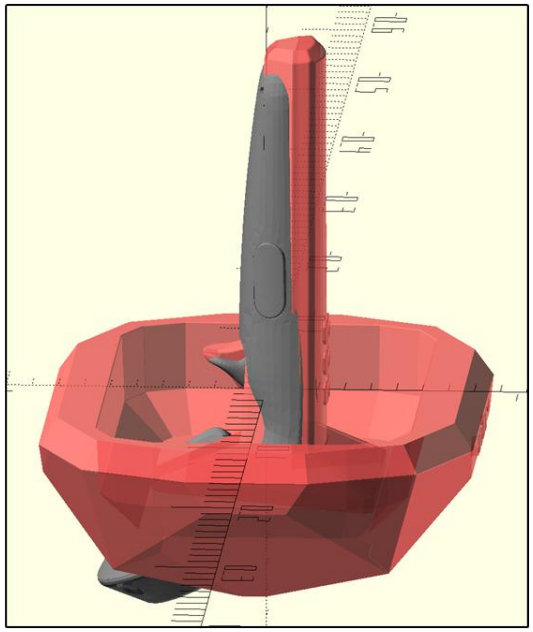
当渲染模式在 SteamVR™ 中出现后，它显示的方位或位置可能是错误的。方位或位置错误通常是 JSON 文件中的“head”变量所致。

设置 Head 变量

JSON 文件中的“head”变量用于描述对象坐标系与参考模型坐标系之间的关系。记住，渲染模型流程中的第一步是将对象的 STL 与现有渲染模型对齐。因此，渲染模型的坐标系不再与传感器相同。“head”变量告知 SteamVR™ 渲染模型坐标系与传感器的坐标系之间的关联。

要完成 head 变量，我们可以将渲染模型导入原始 CAD 并测量两个原点之间的相对差异。但是，该信息已存在于我们用于对齐渲染模型的 OpenSCAD 中。我们可以提出一个问题，即要将参考渲染模型与控制器对象对齐，需要执行怎样的旋转和平移？答案就是将上述的旋转和平移反向执行。

```
#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);
```



将这些值转换为 head 变量是很简单的。

1) 可将平移直接输入到“position”值中，数值为 [0, 0.015, -0.040]。
2) 仅围绕 X 轴旋转意味着 +X 轴的方向不变。Head 成员的“plus_x”等于 [1, 0, 0]。
3) 旋转不会改变 +Z 轴的方向。渲染模型的 +Z 轴现在略微指向对象的 +Y 轴方向。“plus_z”的值等于 [0, sin(3), cos(3)]，输入到 JSON 中时为 [0, 0.05233595624, 0.99862953475]。

JSON 文件的最终 head 变量为：

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

一旦设置了“head”变量，使用 lighthouse_console 将新 JSON 文件上载到对象。

总结

创建渲染模型的流程耗时较长，需要一些初始规划。生成原始 STL 文件时，记住渲染模型的规则。它应是一个具有最少网格数的对象。将新渲染模型与 Vive 控制器的渲染模型对齐，这是在所有 SteamVR™ 应用程序中创建优良体验的最佳方式。一旦匹配了 STL，添加材料和纹理就很简单了。但是，标记所有接缝和导出有用的 UV 图是一个耗时的过程。尝试使用实体纹理创建渲染模型 OBJ，以在花时间标记接缝和展开 UV 图之前，能够验证 OBJ 是否满足 SteamVR™ 的条件。一旦渲染模型可以正常运行，标记接缝并导出有用的 UV 图，以助于利用更合适的纹理定义对象形状。在 SteamVR™ 的渲染模型目录下创建一个用于渲染模型的子目录，并将对象、材料和纹理文件复制都该目录中。同时，在对象的 JSON 文件中添加对该目录的引用。重启 SteamVR™ 以加载新 JSON 和渲染模型，并排除任何问题。最后，设置 JSON 文件中“head”变量以使对象朝向渲染模型。待这些步骤完成后，您应可以在 VR 中轻松地操纵控制器，在不同的应用中评估定位性能并测试控制器。

完成 OpenSCAD 文件

以下 OpenSCAD 文件用于渲染与 Vive 控制器的渲染模型处于同一空间中的新控制器对象的 STL 文件，并将新控制器的渲染模型与 Vive 控制器的渲染模型对齐。

```
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);
```

以下 OpenSCAD 文件用于渲染与 Vive 控制器的渲染模型处于同一空间中的新控制器对象的 STL 文件，并将 Vive 控制器的渲染模型与新控制器的渲染模型对齐。此对齐操作所得到的值将用于设置 JSON 中的“head”变量。

```
#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);
```