MM 804 - Graphics and Animations

Link to GitHub:

https://github.com/Har-Pan/MM-804---Assignment-2.git

Downloaded the 3D model from https://www.thingiverse.com/thing:5167431

Information on the stl file

Name: House_Tower_Gate.stl

Size: 50.7 MB or 50,672,934 bytes

Type: STL

Vertex Count for Original Model: 505,873

Vertex Count for Clipped Out Part of Model: 308,009

Vertex Count for Remaining Part of Model: 203,324

Vertex Count for Intersection part of the Model: 2775

VTK version

vtk==9.1.0

Setup on Mac and Windows

1. Install Python3 version 3.8.8

   Refer the webiste https://docs.python-guide.org/starting/install3/osx/)

2. Install the requirements package

pip install -r requirements.txt

3. Run the Script

   python assignment.py

Code

```
import vtk
from vtk.util.colors import brown_ochre, tomato, banana


import vtk
from vtk.util.colors import brown_ochre, tomato, banana


read_stl = vtk.vtkSTLReader()   # Here we will Read input file vtkSTLReader()
read_stl.SetFileName("House_Tower_Gate.stl")
import vtk
from vtk.util.colors import brown_ochre, tomato, banana


read_stl = vtk.vtkSTLReader()   # Here we will Read input file vtkSTLReader()
read_stl.SetFileName("House_Tower_Gate.stl")
read_stl_mapper = vtk.vtkPolyDataMapper()  # We Create PolyMapper for the stl and send
to output port.
read_stl_mapper.SetInputConnection(read_stl.GetOutputPort())


stl_Center = read_stl_mapper.GetCenter() # We now will get and store the center and
normals for stl
stl_Normals = vtk.vtkPolyDataNormals()
stl_Normals.SetInputConnection(read_stl.GetOutputPort())


stl_Plane = vtk.vtkPlane() # Now we create a plane with origin as center of stl data
and set normal
stl_Plane.SetOrigin(stl_Center)
stl_Plane.SetNormal(1, 0, 1) # Setting the normal vector to [1, 0, 1]T
stl_Clipper = vtk.vtkClipPolyData() # Create a clipper to clip the object/data using
ClipPolyData()
stl_Clipper.SetInputConnection(stl_Normals.GetOutputPort())
stl_Clipper.SetClipFunction(stl_Plane) #Set  the plane to clip the data.
stl_Clipper.GenerateClipScalarsOn()
stl_Clipper.GenerateClippedOutputOn() #Generate clipped out data
stl_Clipper.SetValue(0) #Clipping value of the implicit function set to 0.

stl_ClipMapper = vtk.vtkPolyDataMapper() # Create a PolyDataMapper for the stl
stl_ClipMapper.SetInputConnection(stl_Clipper.GetOutputPort())
stl_ClipMapper.ScalarVisibilityOff()
backProp = vtk.vtkProperty()
backProp.SetDiffuseColor(tomato)


stl_ClipActor = vtk.vtkActor()  # Create clip actor
stl_ClipActor.SetMapper(stl_ClipMapper)
stl_ClipActor.GetProperty().SetColor(brown_ochre) # Now we will set clipped data
colour to brown_ochre
stl_ClipActor.SetBackfaceProperty(backProp)

stlCutEdges = vtk.vtkCutter() # VTK Cutter to display intersection area
stlCutEdges.SetInputConnection(stl_Normals.GetOutputPort())
stlCutEdges.SetCutFunction(stl_Plane)
stlCutEdges.GenerateCutScalarsOn()
stlCutEdges.SetValue(0, 0)

stl_CutStrips = vtk.vtkStripper()
stl_CutStrips.SetInputConnection(stlCutEdges.GetOutputPort())
stl_CutStrips.Update()
stl_CutPoly = vtk.vtkPolyData()
stl_CutPoly.SetPoints(stl_CutStrips.GetOutput().GetPoints()) # Get points from strips
stl_CutPoly.SetPolys(stl_CutStrips.GetOutput().GetLines()) # Create polygonal data to
be displayed


stl_CutTriangles = vtk.vtkTriangleFilter() # Set vtkTriangleFilter for the stl
stl_CutTriangles.SetInputData(stl_CutPoly)
stl_CutMapper = vtk.vtkPolyDataMapper()
stl_CutMapper.SetInputData(stl_CutPoly)
stl_CutMapper.SetInputConnection(stl_CutTriangles.GetOutputPort())
stl_CutActor = vtk.vtkActor()
stl_CutActor.SetMapper(stl_CutMapper)
stl_CutActor.GetProperty().SetColor(banana)


stl_RestMapper = vtk.vtkPolyDataMapper() # Create mapper and actor for remaining data
stl_RestMapper.SetInputData(stl_Clipper.GetClippedOutput())
stl_RestMapper.ScalarVisibilityOff()
stl_RestActor = vtk.vtkActor()
stl_RestActor.SetMapper(stl_RestMapper)
stl_RestActor.GetProperty().SetRepresentationToWireframe()
```

```
stl_list = [] # Initialize render list for the window
rw = vtk.vtkRenderWindow()
rw.SetSize(1300, 950) # Now we set the size for the render window
stl = vtk.vtkRenderWindowInteractor()
stl.SetRenderWindow(rw)


xmins=[0,.5,0,.5] # Now we set locations of all the viewports
xmaxs=[0.5,1,0.5,1]
ymins=[0,0,.5,.5]
ymaxs=[0.5,0.5,1,1]



renBL = vtk.vtkRenderer() # Now we initialize view ports and set location for bottom
left
rw.AddRenderer(renBL)
renBL.SetViewport(xmins[0],ymins[0],xmaxs[0],ymaxs[0])
renBR = vtk.vtkRenderer() # Now we initialize view ports and set location for bottom
right
rw.AddRenderer(renBR)
renBR.SetViewport(xmins[1],ymins[1],xmaxs[1],ymaxs[1])
renTL = vtk.vtkRenderer() # Now we initialize view ports and set location for top left
rw.AddRenderer(renTL)
renTL.SetViewport(xmins[2],ymins[2],xmaxs[2],ymaxs[2])
renTR = vtk.vtkRenderer() # Now we initialize view ports and set location for top
right
rw.AddRenderer(renTR)
renTR.SetViewport(xmins[3],ymins[3],xmaxs[3],ymaxs[3])



renTL.AddActor(stl_ClipActor) # Add actors to viewports
renBL.AddActor(stl_CutActor)
renTR.AddActor(stl_RestActor)
renBR.AddActor(stl_ClipActor)
renBR.AddActor(stl_CutActor)
renBR.AddActor(stl_RestActor)




renTR.SetActiveCamera(renTL.GetActiveCamera()); # SetActiveCameras to the ActiveCamera
of the first renderer
```

```
renBR.SetActiveCamera(renTL.GetActiveCamera()); # This allows the visualization to be
viewed from same angel in all four viewports
renBL.SetActiveCamera(renTL.GetActiveCamera());
renTL.ResetCamera()



rw.Render() # Render the window with all the reprsentations
rw.SetWindowName('MM 802')


win2Im = vtk.vtkWindowToImageFilter() # Writing the rendered scene to JPEG
vtkWindowtoImageFilter()
win2Im.SetInput(rw)
win2Im.ReadFrontBufferOff()
win2Im.Update()

imWriter = vtk.vtkJPEGWriter() # Create a jpeg file writer
imWriter.SetFileName('Result.jpg') # output jpeg filename.
imWriter.SetInputConnection(win2Im.GetOutputPort())
imWriter.Write()


stl.Start()
```

OutPut (From all different angles)