# MM804 GRAPHICS AND ANIMATION Assignment 2 Solution

**Question 1:**

The file used to Clip the polygonal data is **Hulk.STL** is a 3d model of a Hulk.

The size of the model is 44MB and the model can be found here.

**https://www.thingiverse.com/thing:993933/files.**

**Question 2:**

Vertices of the original Model: 2708574

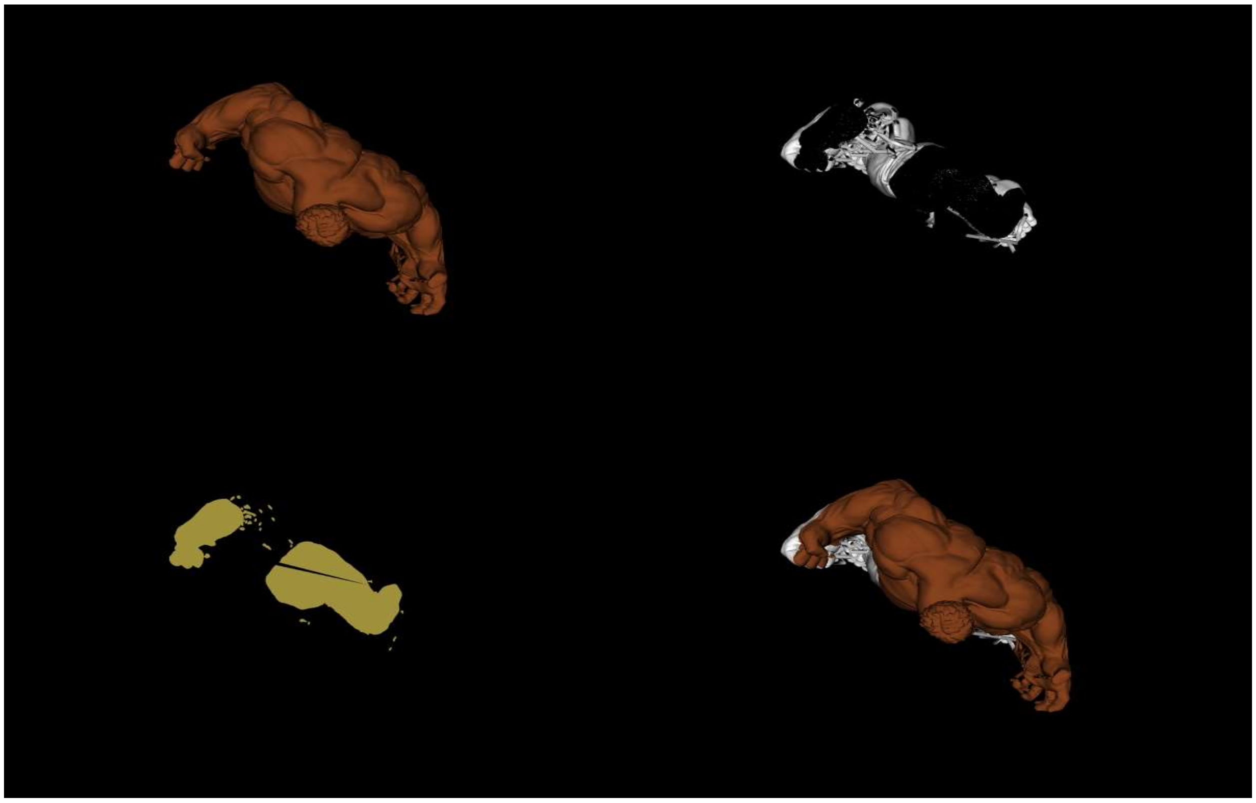
Vertices of the Clipped-out part of the Model: 1803089

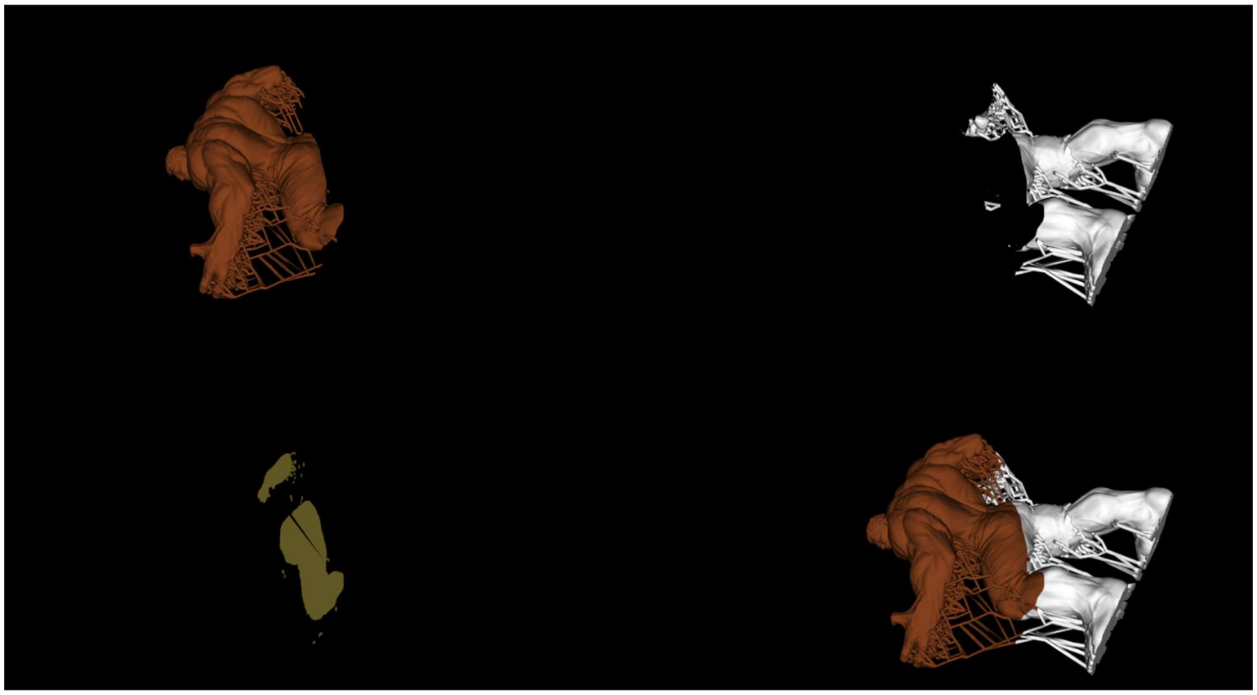
Vertices of the Remaining part of the Model: 901544

Vertices to the Intersection part of the Model: 3940

**Question 3:**

Sample Screenshots of some of the output images





**Question 4:**

Source Code with comments in Human Readable Format

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

# # Initial model vertices using pyvista
# import pyvista as pv
# import numpy as np
# mesh = pv.read('Hulk.stl')
# print(mesh.points.size)

# https://www.thingiverse.com/thing:993933/files

# Question 1. Read the 3d  model input stl  file using vtkSTLReader
inputFile = vtk.vtkSTLReader()
inputFile.SetFileName("Hulk.stl")

# Create Mapper for the input file using vtkPolyDataMapper
fileMapper = vtk.vtkPolyDataMapper()
fileMapper.SetInputConnection(inputFile.GetOutputPort())

# Get and store center and normals of the 3d- model using GetCenter()
getCenter = fileMapper.GetCenter()
inputNormals = vtk.vtkPolyDataNormals()
inputNormals.SetInputConnection(inputFile.GetOutputPort())

# Question 2. Create plane for the model with origin as center of 3d object data
and set normal using vtkPlane()
objectPlane = vtk.vtkPlane()
objectPlane.SetOrigin(getCenter)
objectPlane.SetNormal(1,0,1) # Assigning the normal vector to [1,0,1] as
mentioned

# Question 3. Create a clipper to clip the object/data using vtkClipPolyData
objectClipper = vtk.vtkClipPolyData()
objectClipper.SetInputConnection(inputNormals.GetOutputPort())
objectClipper.SetClipFunction(objectPlane) #setup the plane to clip the data.
objectClipper.GenerateClipScalarsOn() #represent output scalar values
objectClipper.GenerateClippedOutputOn() #Generate clipped out data
objectClipper.SetValue(0) #Clipping values to 0.
# clip the data mapper using vtkPolyDataMapper
objectClipMapper = vtk.vtkPolyDataMapper()
objectClipMapper.SetInputConnection(objectClipper.GetOutputPort())
objectClipMapper.ScalarVisibilityOff()
```

```python
backProp = vtk.vtkProperty()
backProp.SetDiffuseColor(tomato)

# Create clip actor and gettin the meta data of the clipped model
objectClipActor = vtk.vtkActor()
objectClipActor.SetMapper(objectClipMapper)
objectClipActor.GetProperty().SetColor(brown_ochre) # adding color to the clipped data
objectClipActor.SetBackfaceProperty(backProp)
print(objectClipActor.GetProperty())

# Question 4. Show the intersection area between the plane and polygonal data.
# creating a VTK Cutter to display intersection area
objectCutEdges = vtk.vtkCutter()
objectCutEdges.SetInputConnection(inputNormals.GetOutputPort())
objectCutEdges.SetCutFunction(objectPlane)
objectCutEdges.GenerateCutScalarsOn()
objectCutEdges.SetValue(0, 0)
# creating a vtkStripper
objectCutStrips = vtk.vtkStripper()
objectCutStrips.SetInputConnection(objectCutEdges.GetOutputPort())
objectCutStrips.Update()
objectCutPoly = vtk.vtkPolyData()
objectCutPoly.SetPoints(objectCutStrips.GetOutput().GetPoints()) # Get points from strips
objectCutPoly.SetPolys(objectCutStrips.GetOutput().GetLines()) # Create polygonal data to be displayed
# Create Triangle Filter
objectCutTriangles = vtk.vtkTriangleFilter()
objectCutTriangles.SetInputData(objectCutPoly)
objectCutMapper = vtk.vtkPolyDataMapper()
objectCutMapper.SetInputData(objectCutPoly)
objectCutMapper.SetInputConnection(objectCutTriangles.GetOutputPort())
objectCutActor = vtk.vtkActor()
objectCutActor.SetMapper(objectCutMapper)
objectCutActor.GetProperty().SetColor(banana)

# Create mapper and actor for remaining un-clipped data
objectRestMapper = vtk.vtkPolyDataMapper()
objectRestMapper.SetInputData(objectClipper.GetClippedOutput())
objectRestMapper.ScalarVisibilityOff()
objectRestActor = vtk.vtkActor()
objectRestActor.SetMapper(objectRestMapper)
objectRestActor.GetProperty().SetRepresentationToWireframe()
```

```python
# Initialize output render window
renderingWindow = vtk.vtkRenderWindow()
renderingWindow.SetSize(1200, 1000) #Set render window size.
windowInteractor = vtk.vtkRenderWindowInteractor()
windowInteractor.SetRenderWindow(renderingWindow)

# Locations of the viewports
minX=[0,.5,0,.5]
maxX=[0.5,1,0.5,1]
minY=[0,0,.5,.5]
maxY=[0.5,0.5,1,1]

# Question 5: Initialize view ports and set location
# Bottom left

objectBottomLeft = vtk.vtkRenderer()
renderingWindow.AddRenderer(objectBottomLeft)
objectBottomLeft.SetViewport(minX[0],minY[0],maxX[0],maxY[0])


# Bottom right
objectBottomRight = vtk.vtkRenderer()
renderingWindow.AddRenderer(objectBottomRight)
objectBottomRight.SetViewport(minX[1],minY[1],maxX[1],maxY[1])
# Top left
objectTopLeft = vtk.vtkRenderer()
renderingWindow.AddRenderer(objectTopLeft)
objectTopLeft.SetViewport(minX[2],minY[2],maxX[2],maxY[2])
# Top right
objectTopRight = vtk.vtkRenderer()
renderingWindow.AddRenderer(objectTopRight)
objectTopRight.SetViewport(minX[3],minY[3],maxX[3],maxY[3])


# Add actors to viewports
objectTopLeft.AddActor(objectClipActor)
objectBottomLeft.AddActor(objectCutActor)
objectTopRight.AddActor(objectRestActor)

objectBottomRight.AddActor(objectClipActor)
objectBottomRight.AddActor(objectCutActor)
objectBottomRight.AddActor(objectRestActor)

# SetActiveCameras the current active camera of the rendering display
# This allows the visualization to be viewed from same angel in all four
viewports
objectTopRight.SetActiveCamera(objectTopLeft.GetActiveCamera());
objectBottomRight.SetActiveCamera(objectTopLeft.GetActiveCamera());
objectBottomLeft.SetActiveCamera(objectTopLeft.GetActiveCamera());
objectTopLeft.ResetCamera()

# Display the output window with all the representations
renderingWindow.Render()
renderingWindow.SetWindowName('MM 804 Assignment2 output') # Set the windows name

#Finding the number of vertices
# objectModel = vtk.vtkOBJExporter()
# objectModel.SetRenderWindow(renderingWindow)
# renderingWindow.AddRenderer(objectTopLeft)
# objectModel.SetFilePrefix('topLeft')
# objectModel.Write()

# objectModel = vtk.vtkOBJExporter()
# objectModel.SetRenderWindow(renderingWindow)
# renderingWindow.AddRenderer(objectTopRight)
# objectModel.SetFilePrefix('topRight')
# objectModel.Write()

# objectModel = vtk.vtkOBJExporter()
# objectModel.SetRenderWindow(renderingWindow)
# renderingWindow.AddRenderer(objectBottomLeft)
# objectModel.SetFilePrefix('bottomLeft')
# objectModel.Write()

# objectModel = vtk.vtkOBJExporter()
# objectModel.SetRenderWindow(renderingWindow)
# renderingWindow.AddRenderer(objectBottomRight)
# objectModel.SetFilePrefix('bottomRight')
# objectModel.Write()

# Converting the rendered scene to JPEG image format of a single view
windowImageFilter = vtk.vtkWindowToImageFilter()
windowImageFilter.SetInput(renderingWindow)
windowImageFilter.ReadFrontBufferOff()
windowImageFilter.Update()
# Writing it to a image
windowImageWriter = vtk.vtkJPEGWriter() # Create a jpeg file writer
windowImageWriter.SetFileName('finalOutput.jpg') # output jpeg filename.
windowImageWriter.SetInputConnection(windowImageFilter.GetOutputPort()) # Get
render window scene
windowImageWriter.Write()
windowInteractor.Start()

# Alternate wat to Find number of vertices of
# writer = vtk.vtkOBJExporter()
# writer.SetFilePrefix('bottom_left')
# writer.SetInput(renderingWindow)
# writer.Write()
```

**Question 5:**

Readme File

**Development Environment**

- Python - 3.10.0

- VTK - 9.1.0

- OS - Windows 11

**How to run:**

Open file Assignment2.ipynb either in Google Colab or Jupyter-Notebook and execute cell wise for the output window to render