Assignment 2 (Graphics and Animation)

Tasks under this assignment –

- First of all, I downloaded the CT dataset [1] provided by professor Kumar in Lecture 08 (Volume Rendering) that comprised of DICOM images of a heart.
- Next, I began with the code provided in the same lecture to read the DICOM image into vtk using the vtkDICOMReader to read the images and display it using spyder IDE.
- Second task was to change the color transfer function as well as the opacity transfer function under the code as predefined.
- Furthermore, I divided the display into three viewports by providing the x_min, y_min, x_max and y_max values for the display. On the first viewport, I displayed the volume rendering data of the CT dataset. Next, on the second viewport, I displayed the mhd file of the heart provided under Lecture 05 [2] given by professor Kumar on eclass. Finally, on the last viewport, I displayed a combination of both volume rendered data as well as iso surface rendered data.
- Finally, I synced all three viewports using the SetActiveCamera() function of vtk.

1. Information about the **CT dataset**:

The CT dataset comprises of ***188 DICOM images*** of the heart.

The dimensions of the DICOM images under CT dataset are – ***(512 * 512 * 188).***

The size of the dataset is ***49283072 bytes.***

The ***voxel resolution*** is denoted by the following array –

(x_min, x_max)	(0, 249.512)
(y_min, y_max)	(0, 249.512)
(z_min, z_max)	(0, 130.899)

2. Source Code

```python
import vtk

# path to the stored images
path = r'CT'

# Read data
reader = vtk.vtkDICOMImageReader()
reader.SetDirectoryName(path)
reader.Update()

# Create colour transfer function
colorFunc = vtk.vtkColorTransferFunction()
colorFunc.AddRGBPoint(-1024, 0.0, 0.0, 0.0)
colorFunc.AddRGBPoint(-77, 0.5, 0.2, 0.1)
colorFunc.AddRGBPoint(100, 0.9, 0.6, 0.3)
colorFunc.AddRGBPoint(180, 1, 0.8, 0.9)
colorFunc.AddRGBPoint(260, 0.6, 0.1, 0)
colorFunc.AddRGBPoint(3071, 0.7, 0.8, 1)

# Create opacity transfer function
alphaChannelFunc = vtk.vtkPiecewiseFunction()
alphaChannelFunc.AddPoint(-1024, 0.0)
alphaChannelFunc.AddPoint(-77, 0.0)
alphaChannelFunc.AddPoint(179, 0.1)
alphaChannelFunc.AddPoint(260, 0.4)
alphaChannelFunc.AddPoint(3071, 0.7)

```

Figure 1 Reading the data, CT function, Opacity function

```python
# View 1 for volume
volume1 = vtk.vtkVolume()
ren1 = vtk.vtkRenderer()
txt1 = vtk.vtkTextActor()
txt1.SetInput("Volume rendering!")
txtprop=txt1.GetTextProperty()
txtprop.SetFontFamilyToArial()
txtprop.SetFontSize(18)
txtprop.SetColor(1,1,1)

# Define volume mapper
volumeMapper1 = vtk.vtkSmartVolumeMapper()
volumeMapper1.SetInputConnection(reader.GetOutputPort())

# Define volume properties
volumeProperty1 = vtk.vtkVolumeProperty()
volumeProperty1.SetScalarOpacity(alphaChannelFunc)
volumeProperty1.SetColor(colorFunc)
volumeProperty1.ShadeOn()

# Set the mapper and volume properties
volume1.SetMapper(volumeMapper1)
volume1.SetProperty(volumeProperty1)

# create volume renderer
ren1.AddVolume(volume1)
ren1.AddActor(txt1)

```

Figure 2 Volume rendering

```
# View 2 for ISO mapper
ren2 = vtk.vtkRenderer()
txt2 = vtk.vtkTextActor()
txt2.SetInput("ISO rendering!")
txtprop=txt2.GetTextProperty()
txtprop.SetFontFamilyToArial()
txtprop.SetFontSize(18)
txtprop.SetColor(1,1,1)

# Apply Marching Cubes algorithm
march_cubes = vtk.vtkMarchingCubes()
march_cubes.SetInputConnection(reader.GetOutputPort())
march_cubes.ComputeNormalsOn()
march_cubes.ComputeGradientsOn()
march_cubes.SetValue(0,260)
march_cubes.Update()

# Polydata mapper for the iso-surface
isoMapper = vtk.vtkPolyDataMapper()
isoMapper.SetInputConnection(march_cubes.GetOutputPort())
isoMapper.ScalarVisibilityOff()

# Actor for the iso surface
isoActor = vtk.vtkActor()
isoActor.SetMapper(isoMapper)
isoActor.GetProperty().SetColor(1.0,1.0,1.0)

# create iso metric renderer
ren2.AddActor(isoActor)
ren2.AddActor(txt2)
```

Figure 3 ISO metric rendering

```
##########################################################################################
# View 3 for Volume+Iso mapping
ren3 = vtk.vtkRenderer()
txt3 = vtk.vtkTextActor()
txt3.SetInput("ISO & Volume rendering!")
txtprop=txt3.GetTextProperty()
txtprop.SetFontFamilyToArial()
txtprop.SetFontSize(18)
txtprop.SetColor(1,1,1)

# create a renderer for both volume as well as iso mapper
ren3.AddVolume(volume1)
ren3.AddActor(isoActor)
ren3.AddActor(txt3)


##########################################################################################
```

Figure 4 Volume and ISO rendering

```
# Render the scenes into 3 different viewports
ren1.SetViewport(0, 0, 0.33, 1)    # viewport 1
ren2.SetViewport(0.33, 0, 0.66, 1) # viewport 2
ren3.SetViewport(0.66, 0, 1, 1)    # viewport 3

# Rendering
renWin = vtk.vtkRenderWindow()
renWin.AddRenderer(ren1) # renders the volume mapper
renWin.AddRenderer(ren2) # renders the iso mapper
renWin.AddRenderer(ren3) # renders both volume and iso mapper

# the render window size for 16:9 display
renWin.SetSize(1920,1080)
renWin.Render()

# The window interactor
iren = vtk.vtkRenderWindowInteractor()
iren.SetRenderWindow(renWin)

# syncing the viewports to first renderer
camera = ren1.GetActiveCamera()
ren1.ResetCamera()
ren2.SetActiveCamera(camera)
ren3.SetActiveCamera(camera)
```

Figure 5 Viewport division and rendering window

```
# pixel intensities
maximum_minimum = reader.GetOutput().GetScalarRange()
print('the intensities are: ', maximum_minimum)
# Dimension and voxel resolution
dimension_voxel = reader.GetOutput(0)
print(dimension_voxel)

iren.Initialize()
iren.Start()
```

Figure 6 Obtain the intensities, dimensions, size and voxel resolution

3. Output

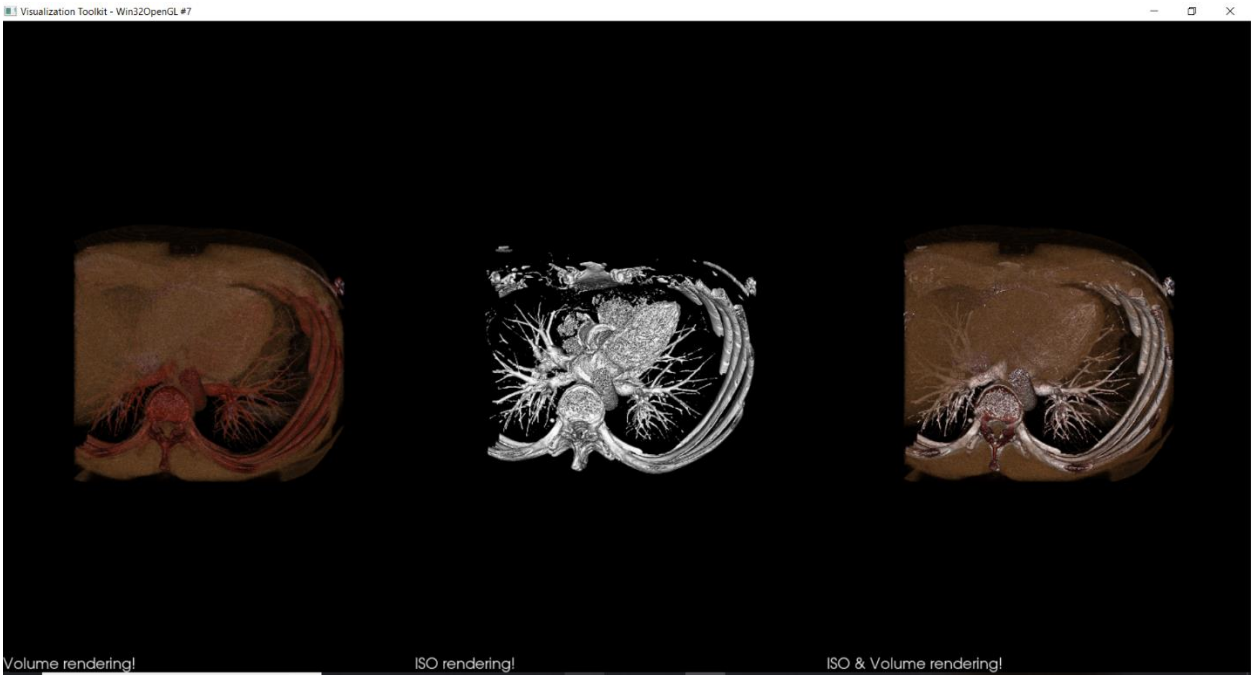

Figure 7 Default view

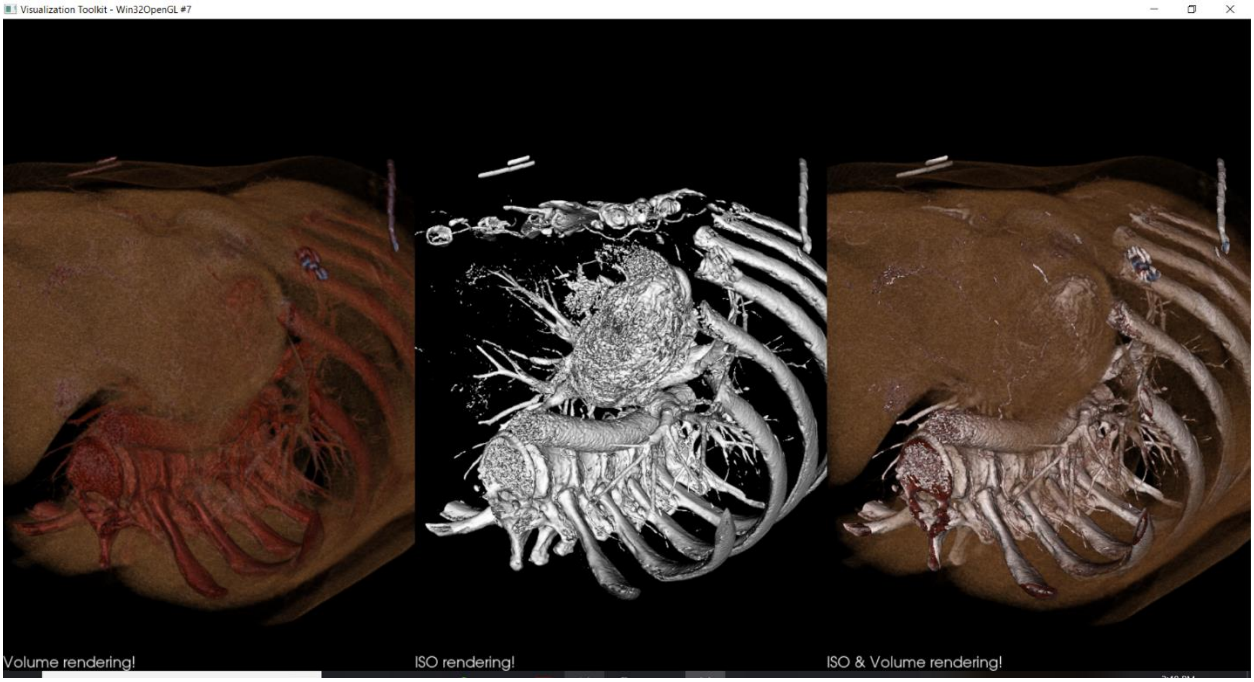

Figure 8 Enlarged Right side view

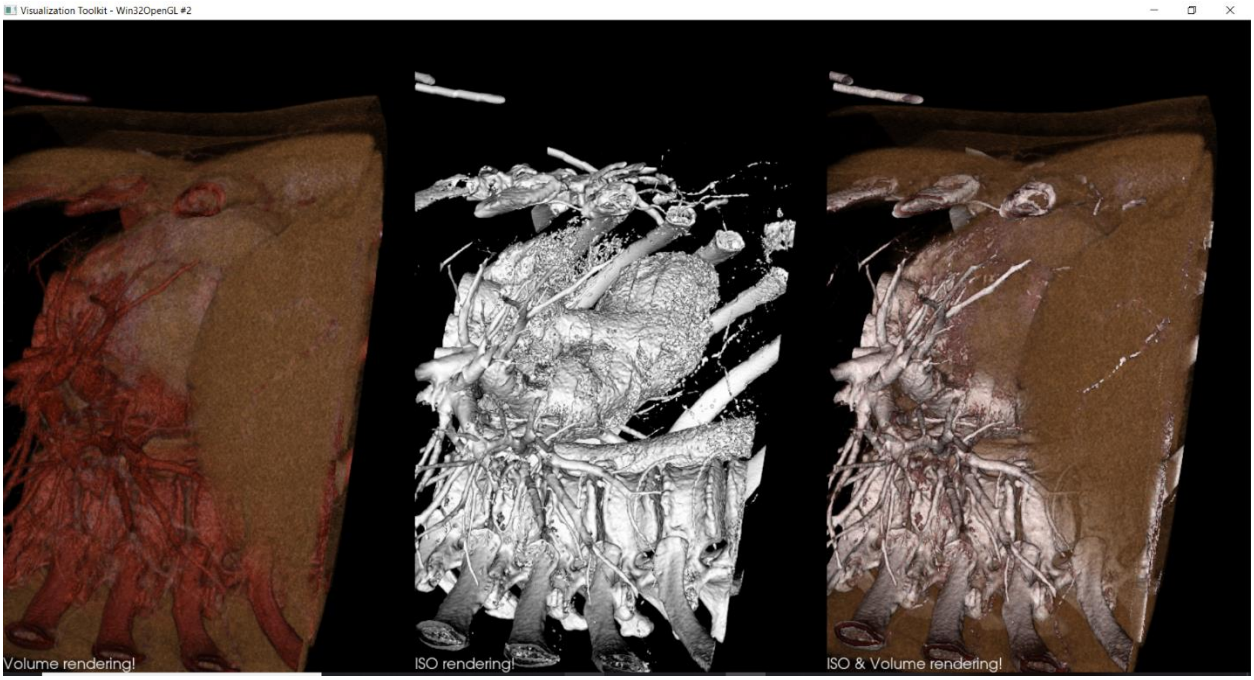

Figure 9 Enlarged Side view

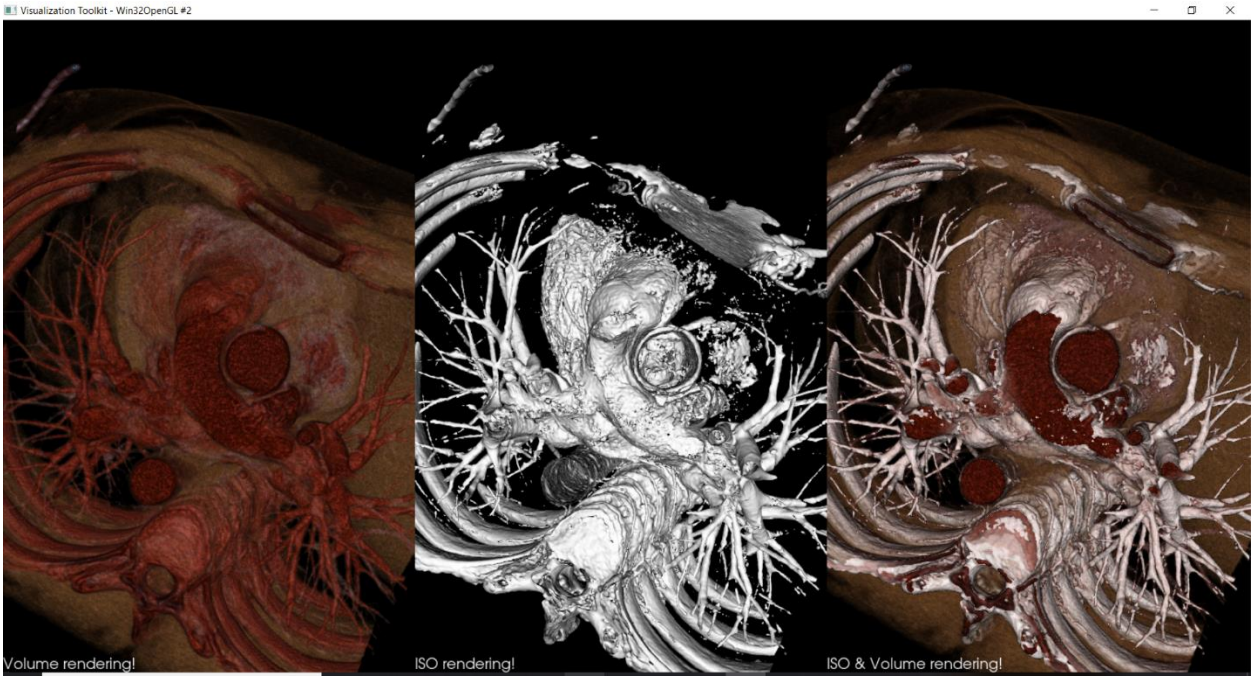

Figure 10 Enlarged Left side view

NOTE: The vtk version used for performing this program is ***8.2.0.***

4. References

1) CT: https://eclass.srv.ualberta.ca/mod/folder/view.php?id=3991376
2) ISO: https://eclass.srv.ualberta.ca/mod/folder/view.php?id=3991373