Politechnika Łódzka
Instytut Elektroniki

Image Processing and Computer Graphics

Laboratory #4

VTK

M. Kociński, J. Blumenfeld

Medical Electronics Division
Institute of Electronics

Introduction to visualization

During this laboratory session you will acquaint with basic capabilities of the Visualization Toolkit (VTK) library. The VTK is an open-source, freely available software system for 3D computer graphics, image processing and visualization. VTK is cross-platform and runs on Linux, Windows, Mac and Unix platforms.

Rendering

Process of generation of 2D and 3D images using the computers is called rendering. At first, a data are transformed into a graphical form and then rendered. Physical generation of an image is based on the reflection of light rays from the surface of objects. The object will be visible only when the reflected rays hit the eye. In reality this may work because light sources are generating enormous amount of light. Simulation of this process is almost impossible to implement, so we use other rendering techniques that can be divided into two types:

- image-order
- object-order.

During the image-order rendering each pixel of object is painted separately (e.g. ray-tracing, where each ray of light is traced separately). In the object-order thype the whole objects (actors) are rendered in the scene at the same time (simultaneously) in a certain order: left to right, top to bottom. We should also pay attention to the objects that don't have surface, such as clouds or fog. In this case we have to consider the changing light properties inside of this objects.

We can distinguish 3 primary rendering process components: sources of light, rendered objects (that we referred to as actors) and camera. The simplest type of light sources is the infinitely distant, point light source that emits parallel light rays in all directions.

The way that the 3D scene can be seen depends on: camera:

- camera position
- camera orientation
- camera focal point
- the method of camera projections
- and the position of the camera back and front clipping planes.

There are two projection methods:

- orthographic
- perspective.

In orthographic method rays of light entering the camera are parallel to the projection vector. In the second case light rays go through a common point. Front clipping plane is used to eliminate objects that are too close to the camera and back clipping plane to eliminate to far objects.

An actor to be shown in scene has to be modeled. In practice, objects are represented by combination of primitives forms like points, lines, polygons, curves and splines of various forms.

There are seven basic objects that we use to render a scene. Documentation of all objects and classes used in vtk library is available on the webpage: http://www.vtk.org/doc/release/5.10/html/classes.html.

I. vtkRenderWindow — manages a window on the display device; it is possible to draw one or

more renderers into an instance of vtkRenderWindow.

II. vtkRenderer — coordinates the rendering process involving lights, cameras, and actors.
III. vtkLight — a source of light to illuminate the scene.
IV. vtkCamera — defines the view position, focal point and other viewing properties of the scene.
V. vtkActor — represents an object rendered in the scene, including its properties (color, shading type, etc.) and position in the words coordinate system. (Note: vtkActor is a subclass of vktProp. vtkProp is a more general form of actor that includes annotation and 2D drawing classes.)
VI. vtkProperty — defines the appearance properties of an actor including color, transparency, and lighting properties such as specular and diffuse. Also representational properties like wireframe and solid surface.
VII. vtkMapper — the geometric representation for an actor. More than one actor may refer to the same mapper.

```
1. Run script cone.py with a cone model and added a light source. Change roll
and azimuth parameters of camera.
2. Run scrypt triangle1.py that draws triangle on the black background. Pay
attention to used pipeline of the basic vtk objects in the graphic model.
3. Run script triangle2.py. Notice how to set colors to each triangle node.
4. Draw a sphere, use vtkSphereSource class (script sphere1.py). Change some of
the parameters:

- PhiResolution,
- ThetaResolution
- Radius
- Position of the sphere in the 3D space.

5. Change some of the surface properties of the sphere with the use of
GetProperty() object:
- SetColor() — RGB color in range (0:0–1:0)
- SetDiffuse() — in range (0:0–1:0)
- SetSpecular() — in range (0:0–1:0)
- SetSpecularPower() — in range (0–255)
- SetBackground(...) method on the renderer object.

6. With the use of vtkCylinderSource object draw cylinder. Use additional
vtkPolyDataMapper and vtkActor for this purpose (script
sphere1_and_cylinder.py)

7. It is possible to divide RenderWindow among few Renderers (script
renderers.py).
(a) create 4 renderers (vtkRenderer class)
(b) set different colors for each of them with the use of SetBackground(...)
function
(c) put every renderer in appropriate position inside RendererWindow
 ren1.SetViewport(0.0,0.0,0.5,0)
 ren2.SetViewport(0.5,0.0,1.0,0.5)
 ren3.SetViewport(0.0,0.5,0.5,1.0)
 ren4.SetViewport(0.5,0.5,1.0,1.0)
(d) add each renderer to renderer window (use AddRenderer(...) function)
(e) create 4 different 3D objects to render in every renderer:
 Cone
    – use: vtkConeSource, SetCenter(...),SetHeight(...), SetRadius(...),
SetResolution(...),SetAngle(...)
 Cube
```

```
    - use: vtkCubeSource, SetXLength(...), SetYLenght(...),
SetZLength(...),SetCenter(...)

 Use other objects e.g.: vtkArrowSource, vtkTextCource,
vtkDiskSource,vtkEarthSource, vtkTexturedSphereSource, vtkPlaneSource,...
(f) for each 3D object create mapper and actor (vtkPolyDataMapper, vtkActor )
(g) add actors to the rendreres (AddActor(...))

8. VTK has implemented many components and methods to image processing. To read
```