VTK Assignment - Lesson 1

Mateus Almeida - 118146, Filipe Silveira - 97981

Information Visualization, 2023 (MSc Informatics Engineering, University of Aveiro)

1 Abstract

The Visualization Toolkit (VTK) is an open-source software system for 3D computer graphics, image processing and scientific visualization.

This report will detail the exploration and section guidelines of the first lesson of the VTK technology in the Information Visualization class. All of the exercises were done using the python wrapper for the VTK library.

2 Primitives

Through the first example, we learnt of the different available primitives in VTK, such as Cone, Cube, Sphere and Cylinder. For this report we choose to add a $vtkSphereSource$ and four $vtkCubeSource$ as seen in Figure 1.

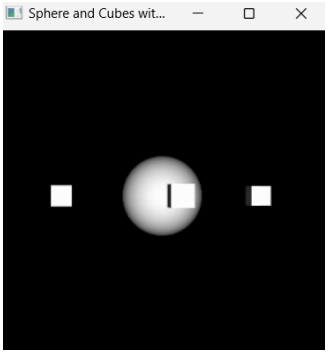

Figure 1: Demonstration of the primitives.

It was also learnt that it is possible to change the render's background through the method $SetBackground$ function, which receives RGB values as a parameter.

However, to better demonstrate the properties of this report the background was not changed from the default. The render windows has default values of 300x300 but its size can be changed through the $SetSize$ function.

3 Interaction

In the previous section we had to use a for loop that would render the windows 1000 times. To remove the unnecessary loop and add interactivity we added a render window interactor. Through the Interactor, it's possible to interact with the rendered source. These were the observed results for keys:

- **Mouse left button (Rotate):** Rotates the scene when the left mouse button is clicked and dragged.
- **Mouse right button (Pan):** Pans the scene when the right mouse button is clicked and dragged.
- **Mouse wheel (Zoom):** Zooms in or out of the scene based on the direction of the mouse wheel.
- **Key F (Fly-To):** Moves the camera to focus on a specific point or object in the scene.
- **Key P (Pick):** Allows the user to interactively pick/select an object in the scene.
- **Key R (Reset):** Resets the view to the initial state.
- **Key S (Surfaces):** Switches the rendering mode to display surfaces.
- **Key W (Wireframe):** Switches the rendering mode to display wireframes as seen in Figure.
- **Key J/Key T (Joystick or Trackball Style):** Changes the interaction style to either Joystick or Trackball.
- **Key E (Exit):** Exits the application or closes the VTK window.

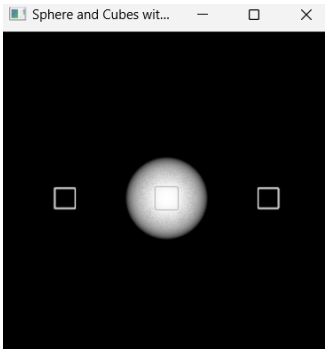

Figure 2: Cubes and Sphere in Wireframe

4 Camera Control

To add a top-down view of our primitives, we had to change the orientation of the camera.

In VTK, a camera can be initialized through the $vtkCamera$ method. or it's possible to control the camera of each renderer through the $GetActiveCamera()$GetActiveCamera method therefore adding more simplicity to the code.

To aid our necessity we also added the property $ParallelProjectio$ to provide a clear and undistorted view of the objects in relation to the light sources, crucial for understanding the

impact of lighting in a 3D environment as we will see in the next section.

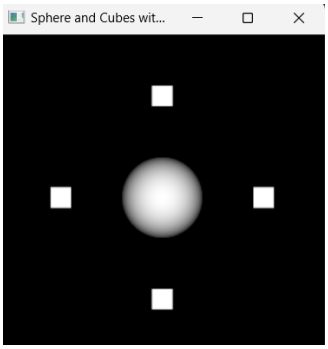

Figure 3: Camera with a top-down view

5 Lighting

To showcase VTK's lighting capabilities, we introduce multiple light sources into our scene.

By creating a function called $activateLight$ we are able to activate lights at designated positions, each emitting a distinct color that corresponds to a cube in the scene and allowed us to observe their individual and collective effects on the central sphere.

However due to the lighting effects on the cubes, as seen in Figure 4, the light sources aren't easily identifiable.

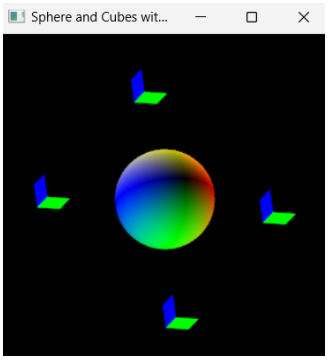

Figure 4: Primitives with the added lighting.

6 Actor properties

In order to correct the lighting effect on the cubes, it was learnt that it's possible to change the color and opacity, as well as other features of the primitive actor, through the $GetProperty$ method.

In addition, it is also possible to change the actor's representation. Here are some of the found properties:

- To set the representation to points:
  - $GetProperty().SetRepresentationToPoints()$
- To set the representation to wireframe:
  - $GetProperty().SetRepresentationToWireframe()$
- To set the representation to surface:
  - $GetProperty().SetRepresentationToSurface()$

To highlight the source of the light, we changed the property of the cube actor to set the color the same as the color of the light. In addition we switched the light effect off with the $LightingOff()$ function so that the lights of the cubes don't influence each other.

These changes are demonstrated in Figure 5.s

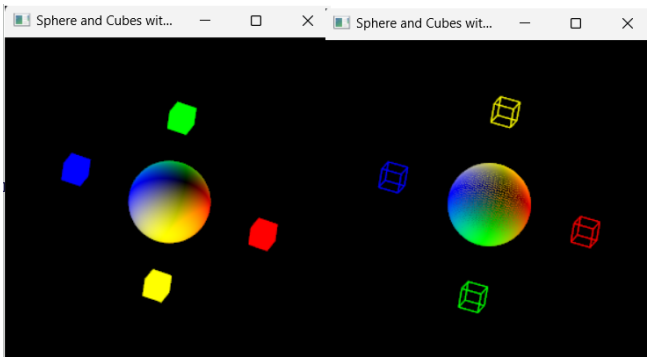

Figure 5: Surfaces/Wireframe primitives with properties.

7 Conclusion

In conclusion, the exploration of the Visualization Toolkit (VTK) during the first lesson of the CG module, has provided an introduction to the possibilities of this open-source software system. The exercises, spanning primitives, interaction, camera control, lighting, actor properties, and the connection between properties and lighting, have provided an understanding of key concepts of the VTK library.

References

[1] *VTK Documentation* `https://vtk.org/documentation/`

[2] *Github Repository for Lessons* `https://github.com/pmdjdias/ua_infovis/tree/master/VTK`