Visual Tool Kit (VTK) - Lesson 1

Miguel Pinto
Student 107449, DETI
Aveiro University
Aveiro, Portugal
miguel.silva48@ua.pt

Tiago Figueiredo
Student 107263, DETI
Aveiro University
Aveiro, Portugal
tiago.a.figueiredo@ua.pt

I. Introduction

The Visualization Toolkit (VTK) is a powerful open-source tool for manipulating and visualizing scientific data. This report documents the exercises completed during the introduction to VTK lessons. Each exercise explores a fundamental feature of VTK, progressing from basic visualization to interactive features, camera control, lighting, and actor properties.

II. Solved Exercises

A. First Example

The first exercise involved creating and visualizing a cone using the ‘vtkConeSource‘. The cone was configured with a height of 2 and a radius of 1. The resolution parameter was explored, showing its effect on the cone’s smoothness (the higher the smoother it became). The background color was changed to white using the ‘SetBackground‘ method. The window size was modified to 500x500 pixels using the ‘SetSize‘ method. The default window size was found to be 300x300.

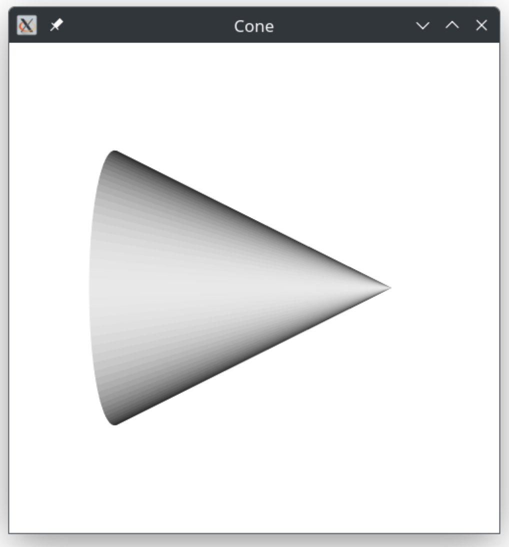

Fig. 1. Visualization of the cone with height 2 and radius 1.

Other primitives, such as cylinders and spheres, were explored to observe similar effects. For the cylinder, reducing the resolution to 1 transformed it into a triangular prism, as shown in Figure 2. This demonstrated how resolution directly influences the geometric fidelity of the shape.

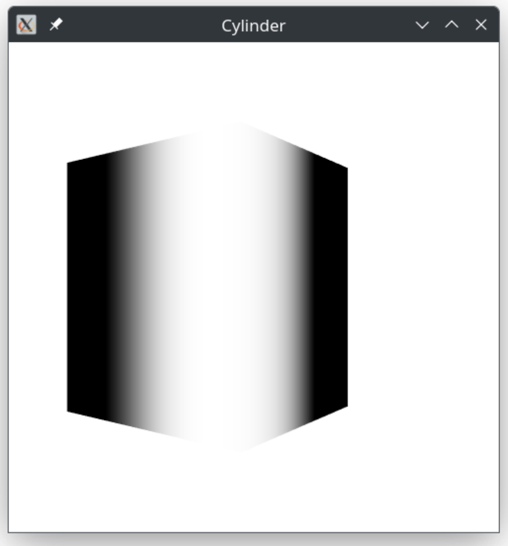

Fig. 2. Visualization of a cylinder with resolution set to 1, resulting in a triangular prism.

The sphere primitive, configured using ‘vtkSphereSource‘, showcased unique properties. Unlike the cone and cylinder, the sphere required two distinct resolution parameters—Theta and Phi resolutions—to control the detail along the longitudinal (z-axis) and latitudinal (y-axis) directions, respectively. In this example, Theta and Phi resolutions were set to 10 and 500, highlighting the distinct differences in smoothness along the two axes, as shown in Figure 3.

B. Interaction

This exercise explored interactivity in VTK using the ‘vtkRenderWindowInteractor‘. By enabling this feature, various mouse controls and key bindings were tested to enhance user interaction and understanding of 3D visualizations.

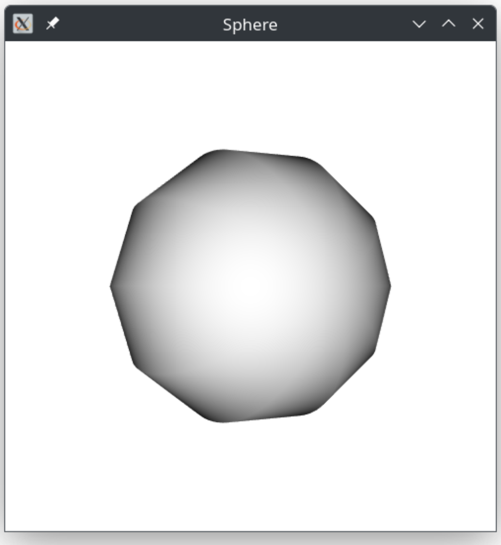

Fig. 3. Visualization of a sphere with Theta resolution set to 10 and Phi resolution set to 500, illustrating axis-specific smoothing.

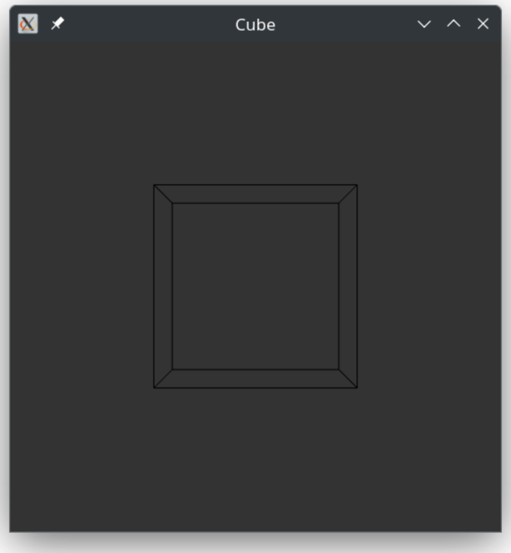

Fig. 4. Cube visualized with perspective projection (the default) and wireframe.

Specific actions included rotation, zooming, switching representations (e.g., wireframe or surface), and object selection, as we found out:

- **Mouse Left Button (MLB):** Rotates the 3D image.
- **Mouse Right Button (MRB):** Adjusts zoom via a hidden slider accessible anywhere on the screen.
- **Mouse Scroll Wheel:** Zooms in and out.
- **Key F:** Moves the camera to a specific focal point, but behaviour is sometimes unpredictable.
- **Key P:** Activates object selection, enclosing the selected object in a bounding box.
- **Key R:** Resets the zoom level and view.
- **Key S:** Switches to surface mode (solid object rendering).
- **Key W:** Switches to wireframe mode (outline rendering).
- **Key J / Key T:** Unresponsive in tests.
- **Key E / Key Q:** Exits the visualization window.

C. Camera Control

The default camera was replaced with a custom camera using the ‘vtkCamera‘ class. The camera’s position was set to (10,10,0), and its view-up vector was set to (0,1,1). Perspective and Parallel projections were compared by visualizing a cube in wireframe mode as it can be seen in Figures 4 and 5.

Note: the background was changed, as the white background did not allow to see the figures when wireframe was toggled on.

D. Lighting

Lighting effects were added to the scene, with a red light positioned at (-5,0,0) and directed towards the origin. The cone’s appearance changed significantly, highlighting the impact of lighting in 3D rendering. The Figure 6 illustrates how the light affects the cone (it was rotated to better visualize the effect).

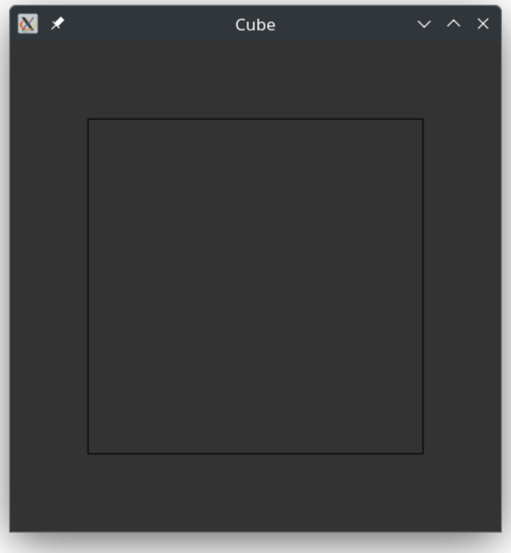

Fig. 5. Cube visualized with parallel projection and wireframe.

E. Actor Properties

In this exercise, the figure’s visual properties were adjusted using the actor’s ‘GetProperty‘ method. Initially, the guide instructed setting the cone’s colour to red using the ‘SetColor‘ method. However, the provided code had it set to blue instead.

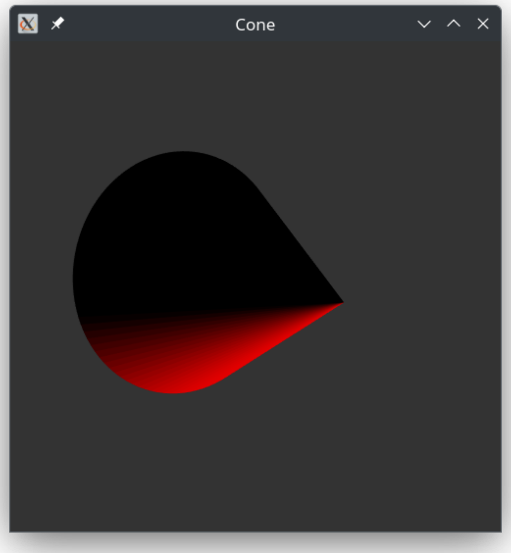

Fig. 6. Cone illuminated by a red light from (-5,0,0).

This discrepancy was corrected, and the cone was rendered with the intended red colour.

Transparency was also tested using the ‘SetOpacity‘ method, which accepts values from 0.0 (completely transparent) to 1.0 (fully opaque). A mid-range value of 0.5 rendered the cylinder translucent, allowing the internal structure to be partially visible, as seen in Figure 7.

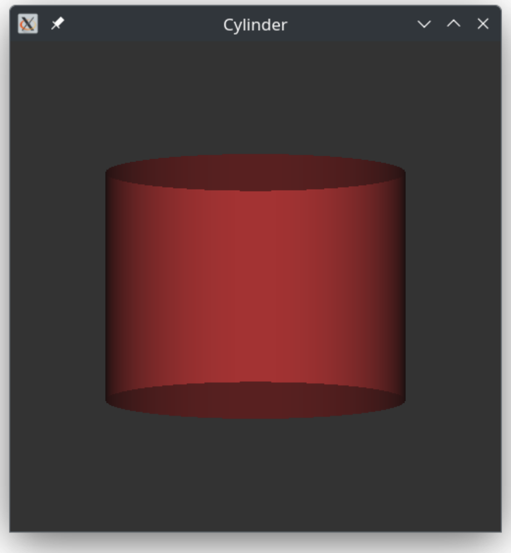

Fig. 7. Cylinder with 0.5 opacity value (translucid).

F. *Properties and Lighting*

Four lights of different colours (red, green, blue, and yellow) were added at specified positions, all pointing towards the origin. Spheres representing each light source were included, with their lighting disabled to prevent interference. A cube primitive was used as the main object to observe the combined lighting effects, as seen in Figure 8.

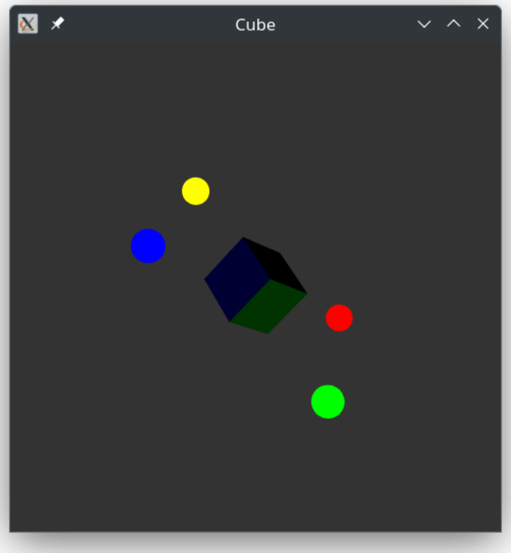

Fig. 8. Cube illuminated by red, green, blue, and yellow lights.

III. Conclusions

Through these exercises, the core concepts of VTK were explored in some detail: object creation, interactivity, camera control, and lighting. These enable the creation of detailed and interactive 3D visualizations, showing VTK’s potential for diverse visualization tasks.

References

[1] VTK Documentation. Available at: https://vtk.org/doc/nightly/html/
[2] Python VTK Wrapper. Available at: https://pypi.org/project/vtk/