VTK LAB02

Miguel Matos 103341, Raquel Paradinha 102491

I. MULTIPLE ACTORS

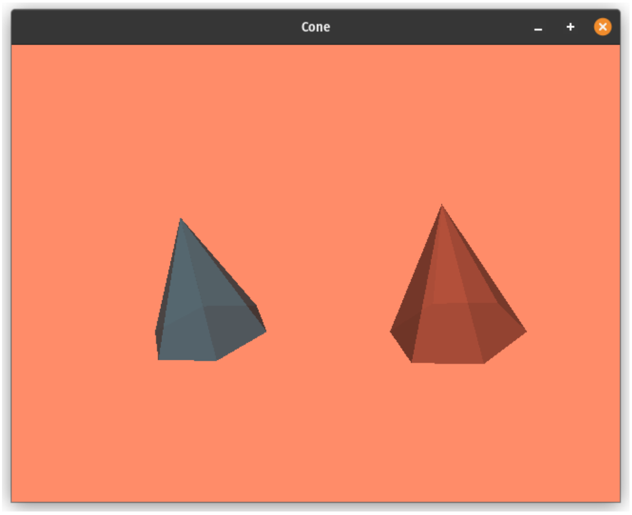
The goal of this exercise is to modify the properties of the cone given in the previous lesson, such as colour, diffuse and specular levels, and opacity, and creating a new cone with shared properties.

A. Implementation

To start the exercise, we changed the properties of the first cone using direct methods like **SetColor** and **SetDiffuse**, among others.

Then, a second cone was created using a shared **vtkProperty** object for efficiency. This approach is more memory-efficient as it allows multiple actors to reuse the same set of properties.

Additionally, the task involves manipulating the position and opacity of the cones, specifically by translating the second cone and setting both cones' opacity to a certain level (e.g., 0.5).



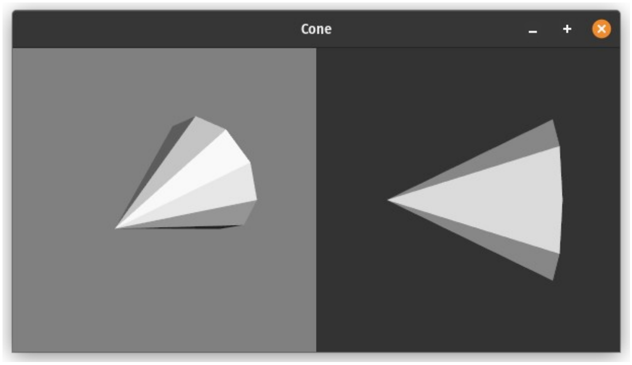
II. MULTIPLE RENDERERS

The second lab exercise consisted in modifying the cone file from the first lesson to include two renderers in the same window, with different viewport settings, camera positions and background colours.

A. Implementation

For this task, the window should be divided to accommodate two renderers, with each renderer occupying half of the window. These renderers must had different viewport settings. This included different camera positions, orientations, and different background colours for clear distinction.

One specific requirement that was implemented, was to rotate the camera by 90 degrees in azimuth in the second renderer to offer a different perspective of the scene.



III. SHADING OPTIONS

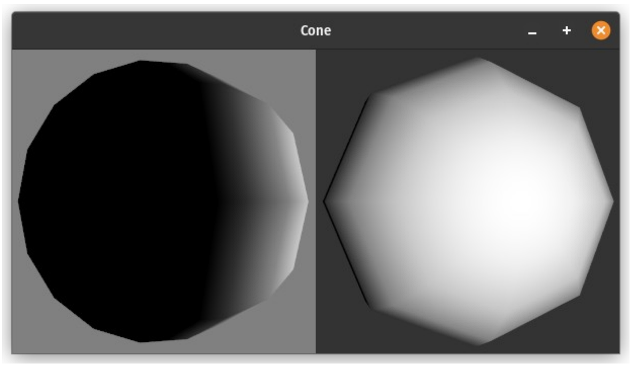
To complete this task it was necessary to display two spheres with contrasting shading techniques to understand the visual differences.

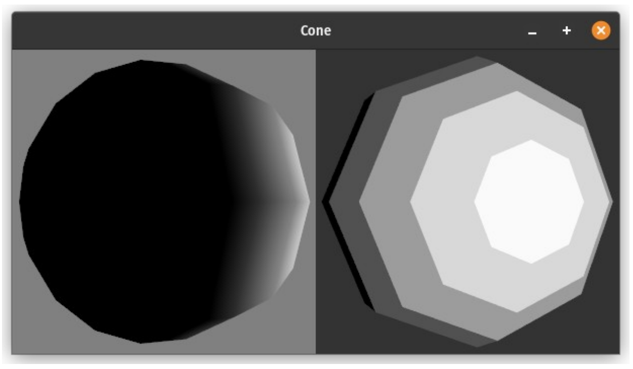
A. Implementation

For this exercise, in a new script, we created one sphere rendered with flat shading (**SetInterpolationToFlat**), which provides a basic, non-smoothed visual representation.

For comparison, other shading techniques, such as **Gouraud** and **Phong** shading, were experimented on the second sphere. Both of these are types of smooth shading techniques, which are used to create a more natural transition between different lighting conditions on a surface.

These methods offer more realistic and smoother shading effects due to their advanced light and shadow interpolation techniques.





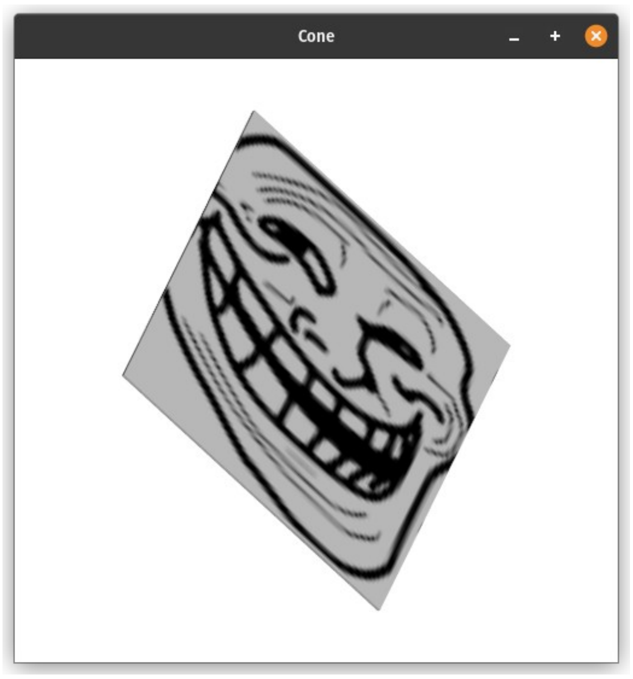
IV. TEXTURES

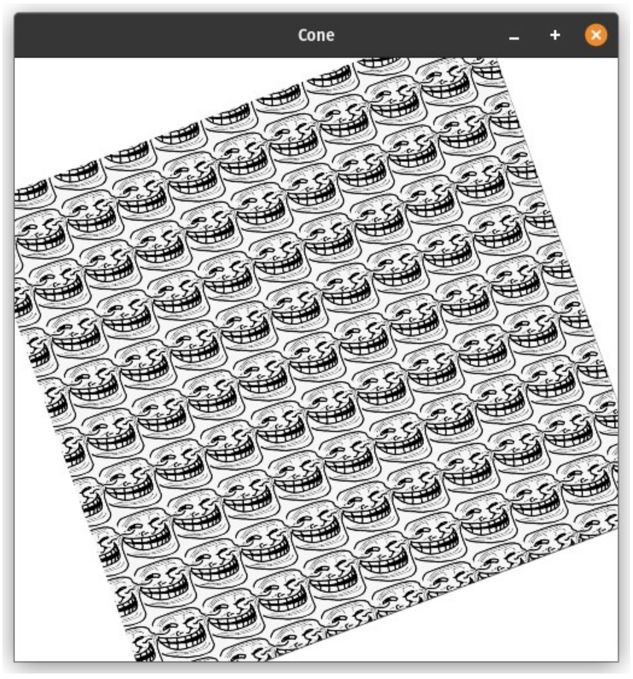
The tasks for this exercise involve creating a plane and applying a texture to it using **vtkTexture** and **vtkJPEGReader**.

A. Implementation

We started by creating the plane and then textured it with an image chosen by us. This image was loaded using the **vtkJPEGReader** class.

An essential part of this task was to modify the size of the plane and observe how the texture adapts to these changes.



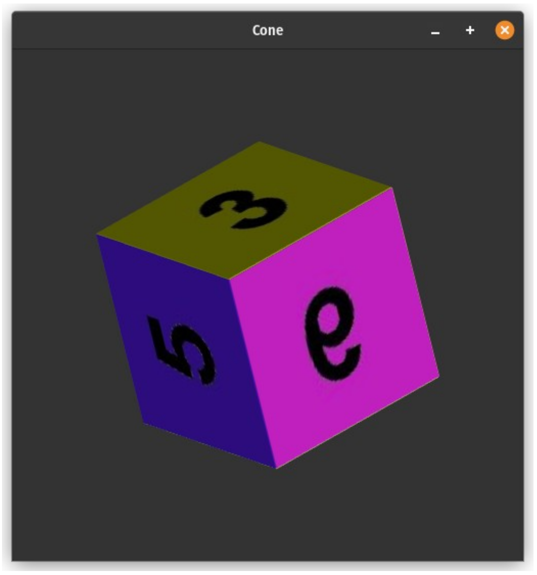


V. TRANSFORMATION

This exercise aims to use transformations to position objects in VTK, particularly focusing on creating six textured planes to form a cube.

A. Implementation

The tasks to this exercise involved using **vtkTransform** and **vtkTransformPolyDataFilter** to accurately position each plane.
Also, the challenge was to apply the correct transformations to the planes so that they collectively form a cube-like structure.
The exercise also included texturing the planes, which aded complexity in managing both the texture application and the transformations. So, we created a function, **make_face**, to simplify the process.



VI. CALLBACKS FOR INTERACTION

In the sixth exercise, we returned to the **cone.py** file with the objective of implementing a callback function for interaction events.

A. Implementation

This task involved defining a new class, **vtkMyCallback**, which was responsible for handling user interaction events.
This callback class provided feedback about interaction events, such as printing the camera's position and other relevant details, which helpd us in understanding how the scene responds to user inputs.