

Copilot- OpenSceneGraph



Centre for Computational Technologies

Transforming human life by democratization of technology

<https://www.cctech.co.in>

Content Reviewer	: Vinayak Sutar
Revision Date	: 7 - 5 - 2024
Version	: 1.0Revision

1. Introduction

- Purpose

The purpose of our application, serving as a copilot for OpenSceneGraph, is to streamline the creation of primitive shapes by providing an intuitive user interface and seamless integration with the OpenSceneGraph framework. By acting as a guiding companion, our tool empowers users to effortlessly generate and visualize geometric objects within their scenes. Through collaborative assistance and efficient workflows, our application enhances the productivity and creativity of users, facilitating the exploration and realization of their spatial visions.

- Scope

The scope of our application includes developing a user-friendly copilot tool for OpenSceneGraph, enabling users to easily generate and customize primitive shapes within the framework. Key features include an intuitive user interface, shape generation and customization options, integration with OpenSceneGraph, copilot functionality for guidance, and export capabilities. Platform compatibility will be considered for major operating systems. The focus is on delivering a functional prototype with potential for future enhancements.

2. System Overview

The application, developed using C++ with Qt for the user interface and OpenSceneGraph (OSG) for graphics, will also integrate OpenAI capabilities. Qt Widgets will facilitate user input for shape parameters, while C++ logic manages shape generation, copilot features, OSG integration, and interaction with OpenAI. Qt signals will handle communication between the UI and backend components. The application will run on Windows, macOS, and Linux. Development will be conducted in Visual Studio, with Git for version control.

3. Functional Requirements

1. Interface:

- Develop an intuitive user interface for specifying parameters required for generating primitive shapes.
- Ensure interactive controls and visual aids are available to streamline the input process.

2. Shape Generation:

- Implement algorithms to generate various primitive shapes based on user-defined parameters.
- Support common primitive shapes such as squares, circles, lines, and arcs.

3. ShapeVisualization:

- Display the generated shapes in a graphical format within the application's interface.
- Enable users to view shapes from different perspectives and angles to assess their geometric properties.

4. Performance Optimization:

- Optimize algorithms and rendering techniques to achieve efficient computation and visualization of shapes.
- Implement strategies to enhance performance, particularly for complex shapes or large datasets.

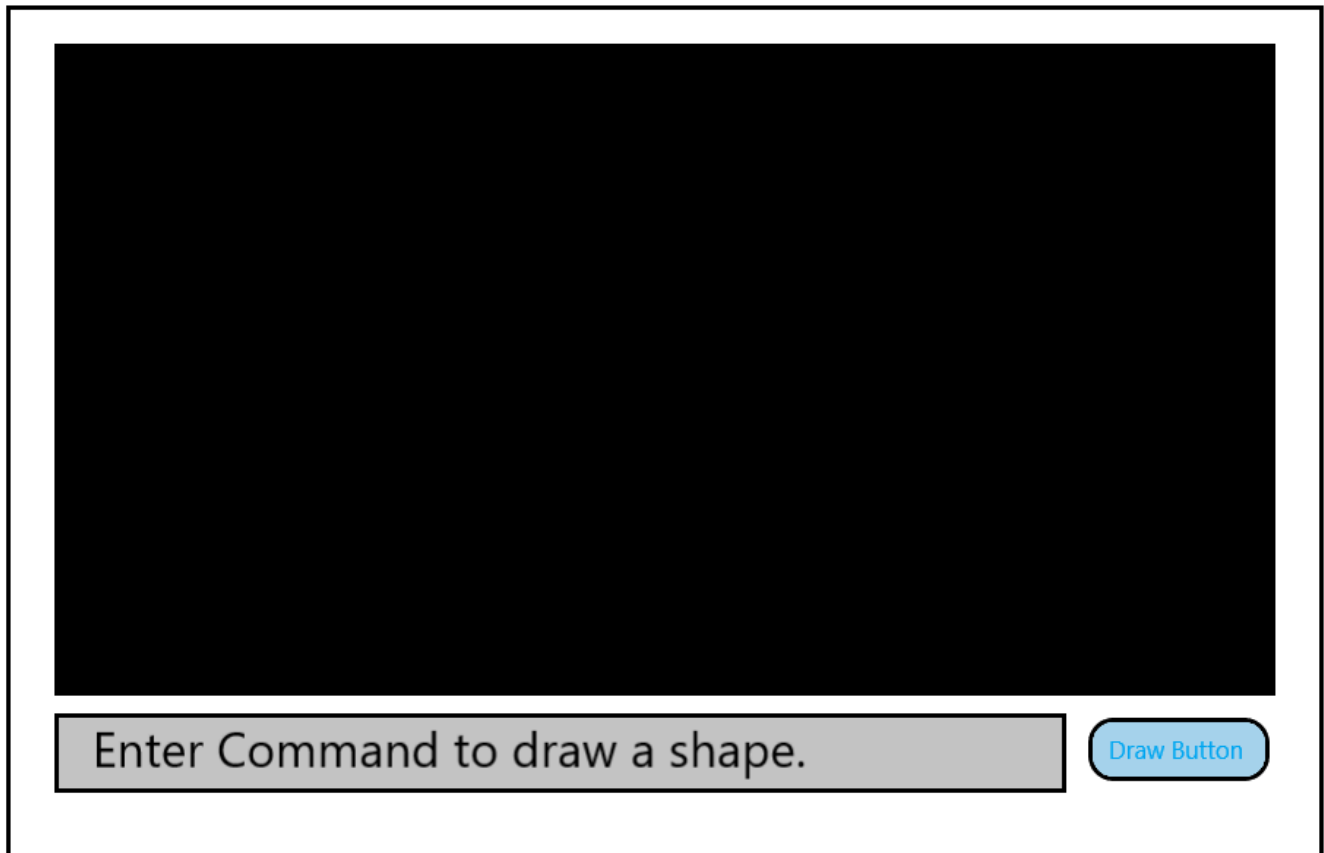
4. Tools

- C++ programming language for application development
- Visual Studio for the structuring part
- Open Scene Graph API
- Qt framework for GUI development
- OpenGL for real-time 3D rendering
- OpenAI API

5. Milestones and Timeline

Sr.No.	Milestones	Date and Time
1	Project Problem Definition	6-04-2024 2.30 PM
2	SRS Presentation & Approval	7-04-2024 2.00 PM
3	Discussion on User Interface & DS	8-04-2024 11.30 AM
4	Implementation of User Interface	9-04-2024 11.30 AM
5	Integration of OpenAI	10-04-2024 4.30 PM
6	Final Presentation and Demonstration	13-04-2024 1.30 PM
7	Project Completion and Submission	13-04-2024 2.30 PM

6. UI



The image displays a user interface for a drawing application. It features a large, solid black rectangular area intended for drawing. Below this area is a light gray rectangular input field containing the placeholder text "Enter Command to draw a shape.". To the right of the input field is a blue button with rounded corners and the text "Draw Button" in white.

7. Conclusion

The proposed application aims to streamline the process of generating and visualizing primitive shapes within the OpenSceneGraph environment. By providing an intuitive user interface and robust shape generation algorithms, users will be able to effortlessly create common primitive shapes such as circles, lines, ellipses and arcs. The application prioritizes efficient performance and accurate visualization, ensuring that users can assess the geometric properties of shapes from various perspectives. Through optimization techniques and interactive controls, the application seeks to enhance the user experience and productivity in scene creation tasks.