Report: 3D Grid with Cube Visualization

Team Members

1. **Supraja** - 112101049
2. **Sunil** - 112101046

Working Principle of the Codebase

The codebase is designed to create a 3D grid environment where a cube can be moved around using keyboard inputs. The cube can change its color, and the grid cells can be filled or cleared based on user interactions. The project uses OpenGL for rendering and GLFW for window management and input handling.

Key Features:

- **3D Grid**: A 5x5x5 grid is rendered in 3D space.
- **Movable Cube**: A cube can be moved along the X, Y, and Z axes using keyboard inputs.
- **Color Change**: The cube's color can be changed by entering RGB values.
- **Fill/Clear Cells**: Grid cells can be filled with the cube's current color or cleared.
- **Rotation**: The entire scene can be rotated around the X and Y axes.

Code Structure:

- **Shaders**: Vertex and fragment shaders are used to handle the rendering of the grid and cube.
- **Grid Initialization**: The grid is initialized with vertices and colors.
- **Cube Initialization**: The cube is initialized with vertices, colors, and indices for rendering.
- **Rendering**: The grid, filled cells, and cube are rendered in each frame.
- **Input Handling**: Keyboard inputs are handled to move the cube, change its color, fill/clear cells, and rotate the scene.

Implementation Details

Shaders

- **Vertex Shader**: Transforms vertex positions using model, view, and projection matrices. It also passes the color to the fragment shader.
- **Fragment Shader**: Outputs the color passed from the vertex shader.

Grid Initialization

- The grid is created by generating vertices for lines parallel to the X, Y, and Z axes.
- The vertices are stored in a vector and uploaded to the GPU using a VBO and VAO.

Cube Initialization

- The cube is defined by 8 vertices, each with a position and color.
- Indices are used to define the cube's faces.
- The vertices and indices are uploaded to the GPU using VBO, VAO, and EBO.

Rendering

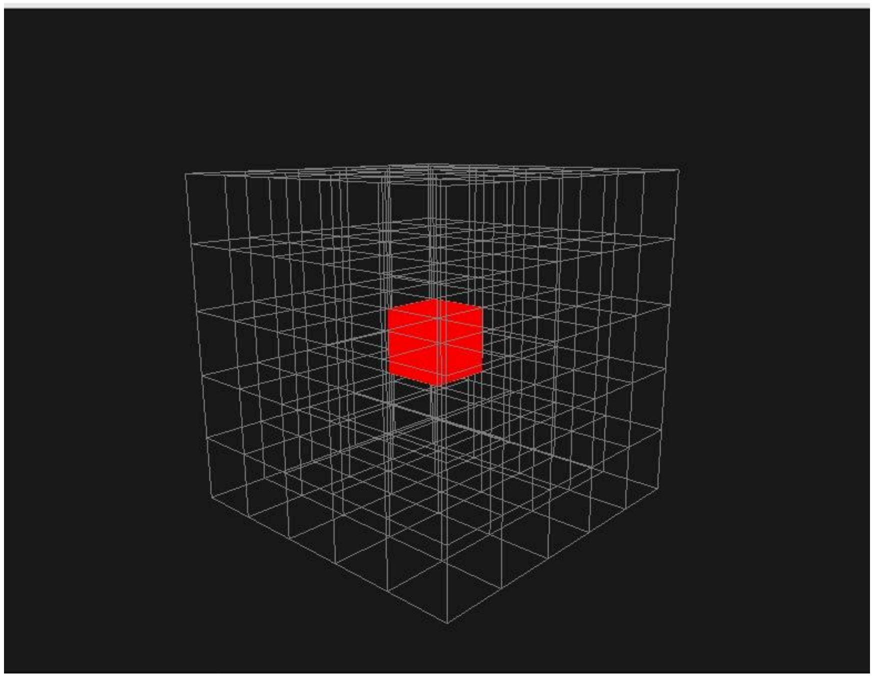
- The grid is rendered using `glDrawArrays` with `GL_LINES`.
- The cube and filled cells are rendered using `glDrawElements` with `GL_TRIANGLES`.
- The model matrix is updated for each object to apply transformations like translation and rotation.

Input Handling

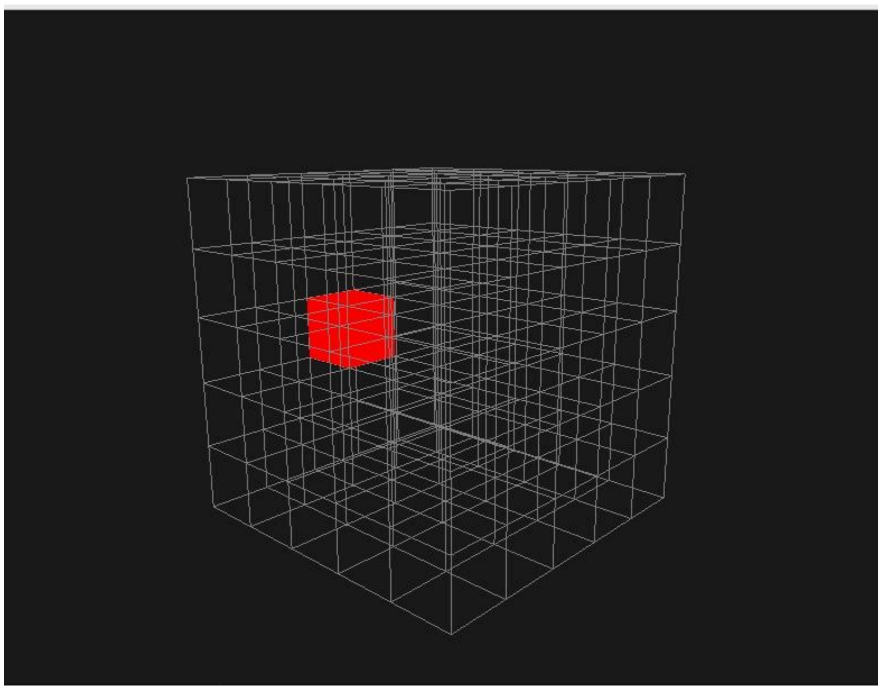
- Keyboard inputs are handled using GLFW's key callback function.
- The cube's position is updated based on the key pressed.
- The cube's color can be changed by entering RGB values.
- Grid cells can be filled or cleared using specific keys.
- The scene can be rotated around the X and Y axes.

Screenshots of Output Results

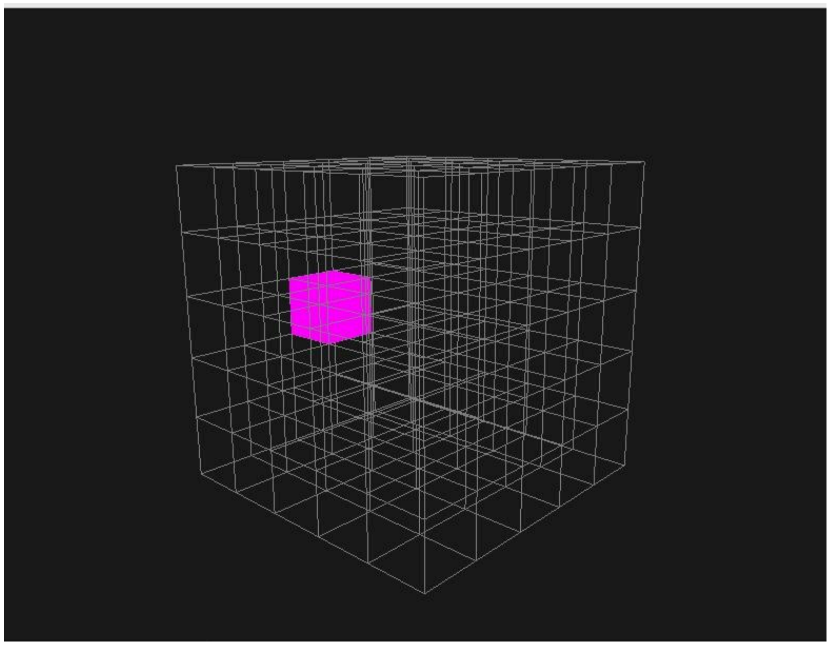
1. Initial View



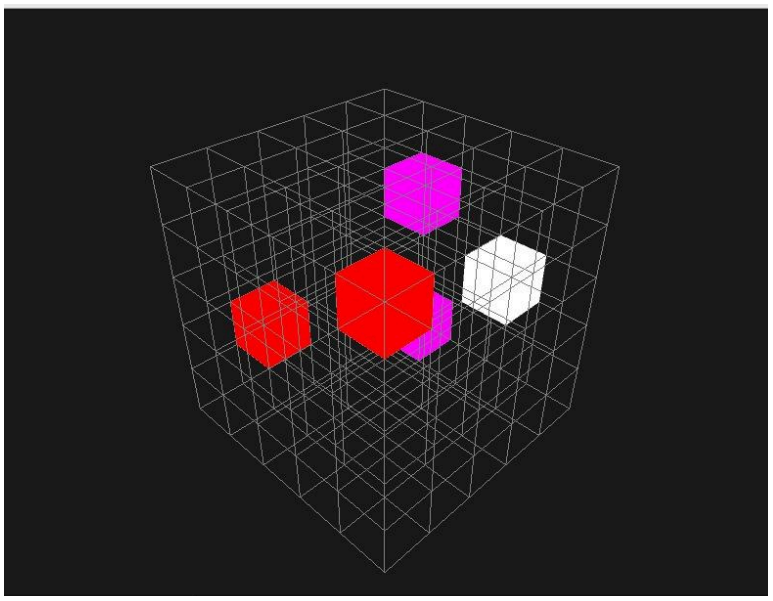
2. Cube Movement



3. Color Change



4. Filled Cells



Conclusion

This project demonstrates the use of OpenGL and GLFW to create an interactive 3D environment. The grid and cube are rendered using shaders, and user interactions are handled to move the cube, change its color, fill/clear grid cells, and rotate the scene. The code is structured to be modular and easy to extend for additional features.