Rubik's Cube Visualization and Interaction in C++ using OpenGL

Arfaz Hussain

September 4, 2024

1 Introduction

This document provides an in-depth explanation of a C++ program designed to visualize and interact with a Rubik's Cube using OpenGL. The code demonstrates advanced 3D graphics programming concepts, including rotations, transformations, and event handling for keyboard and mouse input. The program allows users to interact with a simulated Rubik's Cube, manipulating its layers using various keyboard commands.

2 Dependencies and Libraries

The program utilizes the following libraries:

- `<GL/gl.h>` - The core OpenGL library for rendering.
- `<GL/glut.h>` - The GLUT library for handling windowing, input, and event processing.
- `<vector>` - The C++ Standard Library for dynamic array handling.

3 Code Structure

The code is structured into several functions, each responsible for specific aspects of visualization and interaction:

- Global variable declarations and structures.
- Functions for loading visualization parameters and setting up the camera.
- Functions for applying rotations and drawing individual cubes and the entire Rubik's Cube.
- Functions for handling user input via keyboard and mouse.
- Main function to initialize the OpenGL context and enter the event loop.

4 Function Descriptions

4.1 `apply_rotation(GLfloat angle)`

This function is responsible for applying a rotation to a specific face of the Rubik's Cube. It first identifies which face (X, Y, or Z) is selected based on global variables `x0`, `xK`, `y0`, `yK`, `z0`, `zK`. The function uses these indices to modify the `cube_rotations` array that keeps track of all the rotations applied to each cube.

4.2 `reset_selected_face()`

Resets the selected face for rotation to the default entire cube selection.

4.3 `errand(int x, int y, int z)`

This function is an example of memory management. It creates a temporary integer pointer, allocates memory, and then deallocates it. This could serve as an example or a placeholder for error handling or other computational tasks.

4.4 `set_camera()`

Sets up the camera view using `gluLookAt()` to provide a perspective for rendering the 3D cube.

4.5 `draw_cube(int x, int y, int z)`

Responsible for drawing an individual cube in the Rubik's Cube grid. It applies all stored rotations and translates the cube to its correct position in the 3D space.

4.6 `draw_func()`

The core display function that renders the entire Rubik's Cube. It clears the screen, resets transformations, applies camera settings, and draws all 27 subcubes in their respective positions.

4.7 `init_func()`

Initializes OpenGL settings such as lighting, shading, and material properties. This function sets up the environment for 3D rendering.

4.8 `load_visualization_parameters()`

Loads perspective projection parameters to define how the 3D scene is projected onto the 2D screen.

4.9 `reshape_func(GLsizei w, GLsizei h)`

Handles window resizing events to adjust the aspect ratio and projection settings.

4.10 `keyboard_func(unsigned char key, int x, int y)`

Handles keyboard input for rotating the entire Rubik's Cube or its individual faces. It maps various keys to specific transformations and updates.

4.11 `mouse_func(int b, int s, int x, int y)`

Handles mouse input for zooming in and out of the 3D scene by adjusting the viewing angle.

4.12 `main(int argc, char **argv)`

The entry point of the program. It initializes GLUT, sets up the display mode and window size, registers callback functions, and starts the GLUT event processing loop.

5 Compilation and Execution Instructions

To compile and execute the program, follow these steps:

1. Ensure you have OpenGL and GLUT installed on your system.
2. Use the following command to compile the program:

   ```
   g++ -o rubiks_cube rubiks_cube.cpp -lGL -lGLU -lglut
   ```

3. Run the program using texttmake or using the following command:

   ```
   ./rubiks_cube
   ```

4. Use keyboard and mouse inputs (yet to be implemented) to interact with the Rubik's Cube.

6 Source Code

The complete source code for this Rubik's Cube visualization can be found on GitHub at: `https://github.com/arfazhxss/rubiks-cube-cpp`.