Visvesvaraya Technological University
Belagavi, Karnataka-590 018

A MINI PROJECT REPORT
On

RUBIK'S CUBE GAME

Submitted
in partial fulfilment requirements for the credit of the Course
on
COMPUTER GRAPHICS AND VISUALIZATION LABORATORY
(18CSL67)

by

P PADMAPRASAD SHENOY 4CB19CS064
S SREENIVASA SHENOY 4CB19CS087

Under the Guidance of Mrs.
Sukshma Shetty
Assistant Professor

Department of Computer Science and Engineering
Canara Engineering College , Benjanapadavu 2021-2022

CANARA ENGINEERING COLLEGE

BENJANAPADAVU, BANTWAL- 574219 – KARNATAKA

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CERTIFICATE

This is to certify that P PADMAPRASAD SHENOY (4CB19CS064) and S SREENIVASA SHENOY (4CB19CS087) have successfully completed the project work on 'RUBIK'S CUBE GAME' and submitted in partial fulfillment of the requirements of 6th Semester B.E., Computer Science and Engineering, prescribed by the VISVESVARAYA TECHNOLOGICAL UNIVERSITY during the academic year 2021-2022. It is verified that all corrections/suggestions indicated for internal assessment have been incorporated in the report deposited in the department library. The project report has been approved as it satisfies the academic requirements in respect of mini project work prescribed by Bachelor of Engineering Degree.

Course Instructor	**H.O.D**
Mrs. Sukshma Shetty **Assistant Professor**	**Dr. Demian Antony D' Mello** **Professor**

Name of the Examiners **Signature with Date**

1.

2.

ACKNOWLEDGEMENT

We are indebted to our **Principal, Dr. Ganesh V. Bhat** and management of Canara Engineering College for providing an environment with all facilities that helped us in completing our mini project.

We are extremely grateful to **Dr. Demian Antony D' Mello, Head of Computer Science & Engineering Department** for his moral support and encouragement.

We wish to express our sincere gratitude to our professor **Mrs. Sukshma Shetty**, from Computer Science & Engineering Department, for the guidance and suggestions.

We thank all the teaching and non-teaching staff of Department of Computer Science & Engineering for their kind help.

Last but not the least, we would like to add some personal note. If there is a driving force that kept us going, and what has not changed it is the constant support and blessing of our parents, family and friends. There is no doubt, in spite of my strenuous efforts error might remain in mini project. Naturally, we alone take full responsibility for any lack of clarity, occasional erratum or inexactness that may occur.

TEAM MEMBERS:

P PADMAPRASAD SHENOY(4CB19CS064)

S SREENIVASA SHENOY(4CB19CS087)

ABSTRACT

Computer Graphics is the creation, manipulation, and storage of models and images of pictures objects by the aid of computers. This was started with the display of data on plotters and CRT. Computer Graphics is also defined as the study of techniques to improve the communication between user and machine, Thus Computer Graphics is one of the effective media of communication between machine and user.

Computer Graphics is one of the hottest buzzword in the Information Technology today, with graphics assuming importance in every field. There is a wide scope for graphics based software. Some of the areas in which graphics has made in roads are destroyed publishing, image processing, animation, movies etc.

In our package, we are clearly going to demonstrate the working of "Rubik's Cube Game". Viewers can use mouse interface and navigate between menu and actual output stream through keyboard interface.

We have developed this package using OpenGL. OpenGL is a software interface graphics hardware. The OpenGL Utility Library(GLU) provides many of the modelling functions GLU is a standard part of every OpenGL implementation

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 Introduction to Computer Graphics

The term computer graphics includes almost everything on computers that is not text or sound. Today almost every computer can do some graphics, and people have even come to expect to control their computer through icons and pictures rather than just by typing.

Computer graphics is nothing but drawing pictures on computers also called *rendering*. The pictures can be photographs, drawings, movies, or simulations -- pictures of things which do not yet exist and maybe could never exist. Or they may be pictures from places we cannot see directly, such as medical images from inside your body.

We spend much of our time improving the way computer pictures can simulate real world scenes. We want images on computers to not just look more realistic, but also to be more realistic in their colors, the way objects and rooms are lighted, and the way different materials appear.

Computer graphics is widespread today. Computer imagery is found on television, in newspapers, for example in weather reports, or for example in all kinds of medical investigation and surgical procedures. A well-constructed graph can present complex statistics in a form that is easier to understand and interpret. In the media "such graphs are used to illustrate papers, reports, thesis", and other presentation material.

Many powerful tools have been developed to visualize data. Computer generated imagery can be categorized into several different types: 2D, 3D, and animated graphics. As technology has improved, 3D computer graphics have become more common, but 2D computer graphics are still widely used. Computer graphics has emerged as a sub-field of computer science which studies methods for digitally synthesizing and manipulating visual content.

1.2 Introduction to OpenGL

OpenGL is designed as a streamlined, hardware-independent interface to be implemented on many different hardware platforms. To achieve these qualities, no commands for performing windowing tasks or obtaining user input are included in OpenGL; instead, you must work through whatever windowing system controls the particular hardware you're using

OpenGL is designed to work efficiently even if the computer that displays the graphics you create isn't the computer that runs your graphics program. This might be the case if you work in a networked computer environment where many computers are connected to one another by wires capable of carrying digital data. In this situation, the computer on which your program runs and issues OpenGL drawing commands is called the client, and the computer that receives those commands and performs the drawing is called the server. The format for transmitting OpenGL commands (called the *protocol*) from the client to the server is always the same, so OpenGL programs can work across a network even if the client and server are different kinds of computers. If an OpenGL program isn't running across a network, then there's only one computer, and it is both the client and the server. OpenGL is a software interface to graphics hardware. This interface consists of about 120 distinct commands, which you use to specify the objects and operations needed to produce interactive three-dimensional applications.

1.2.1 OpenGL Architecture

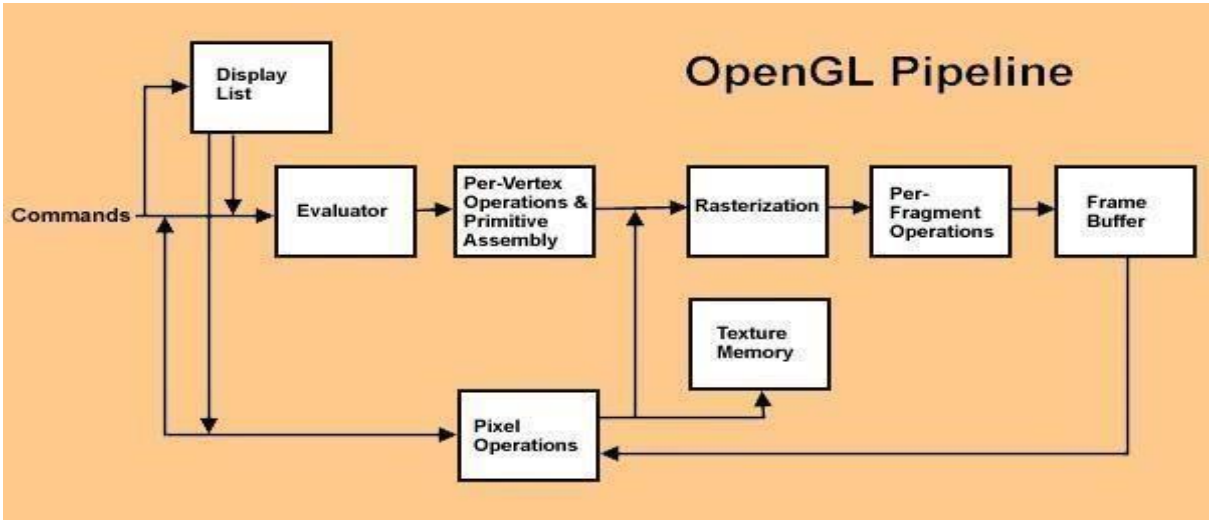

Figure 1.1 OpenGL Architecture

This is the most important diagram, representing the flow of graphical information, as it is processed from CPU to the frame buffer. There are two pipelines of data flow. The upper pipeline is for geometric, vertexbased primitives. The lower pipeline is for pixel-based, image primitives.
Texturing combines the two types of primitives together.

Related APIs

• GLU (OpenGL Utility Library)

- 2D image scaling
- Rendering 3D objects including spheres, cylinders, etc.

1.2.2 Libraries

Enumerated Types

- Libraries are the operating system dependent implementation of OpenGL on the system you're using. Each operating system has its own set of libraries. For UNIX systems, the OpenGL library is commonly named libGL. so and for Microsoft Windows, it's named opengl32.lib. • OpenGL defines numerous types for compatibility GLfloat, GLint, GLenum, etc.

- All of our discussions today will be presented in the C++ computer language. For C++, there are a few required elements which an application must do:
- Header files describe all of the function calls, their parameters and defined constant values to the compiler. OpenGL has header files for GL (the core library), GLU (the utility library), and GLUT (freeware windowing toolkit).
- Finally, enumerated types are definitions for the basic types (i.e. float, double, int, etc.) which your program uses to store variables. To simplify platform independence for OpenGL programs, a complete set of enumerated types are defined. Use them to simplify transferring your programs to other operating systems.

1.2.3 Glut Basics

Application Structure

- Configure and open window
- Initialize OpenGL state
- Register input callback functions
  - 1.2.3..1 render
  - 1.2.3..2 resize
- Enter event processing loop

Built-in functions used:

- glutInit(&argc, argv);
- glutInitDisplayMode();
- glutInitWindowSize();
- glutCreateWindow();
- glutReshapeFunc();
- glutIdleFunc();
- glutMouseFunc();
- glutMotionFunc();
- glutCreateMenu();
- glutAddMenuEntry();
- glutAttachMenu(GLUT_RIGHT_BUTTON);
- glutKeyboardFunc(keyboard);
- glutDisplayFunc(display);
- glutMainLoop();
- glClear();
- glLoadIdentity();
- speedmeter();
- glColor3fv();
- glPushMatrix();
- glRotatef();

User-defined Functions used:

- void polygon();
- void colorcube();
- void speedmeter();
- void display();
- void transpose();
- void topc()
- void frontc()
- void rightc();
- void leftc();
- void bottomc();
- void backc();
- void spincube();
- void spincube();
- void mouse();
- void keyboard();
- void myreshape();
- void mymenu();

GLUT Callback Functions

Routine to call when something happens

- window resize or redraw
- user input
- animation

"Register" callbacks with GLUT

- glutDisplayFunc();
- glutIdleFunc();
- glutKeyboardFunc();

GLUT uses a callback mechanism to do its event processing. Callbacks simplify event processing for the application developer.

GLUT supports many different callback actions, including:

- **glutDisplayFunc()** - called when pixels in the window need to be refreshed
- **glutReshapeFunc ()** - called when the window changes size
- **glutKeyboardFunc ()** - called when a key is struck on the keyboard
- **glutMouseFunc()** - called when the user presses a mouse button on the mouse
- **glutIdleFunc ()** - a callback function called when nothing else is going on.

1.2.4 OpenGL Command

Formats

Suffix	Data Type	C-language Type	Open GL Type
b	8-bit int	signed char	GLbyte
s	16-bit int	Short	GLshort
i	32-bit int	Long	GLint, GLsizei
f	32-bit float	Float	GLfloat, GLcalmpf
d	64bit float	Double	GLdouble, GLclamped
ub	8-bit uns int	unsigned char	GLbyte, GLboolean
us	16-bit uns int	unsigned float	GLushort
ui	32-bit uns int	Unsigned long	GLuint, GLenum

Table 1: Open GL Command Formats

The OpenGL API calls are designed to accept almost any basic data type, which is reflected in the calls name. Vertices from most commercial models are stored as three component floating point vectors. As such, the appropriate OpenGL command to use is glVertex3fv (coords). As mentioned before, OpenGL uses homogenous coordinates to specify vertices. For glVertex*() calls which don't specify all the coordinates (i.e. glVertex2f()), OpenGL will default $z = 0.0$, and $w = 1.0$.

1.2.5 Buffers and their Uses

An OpenGL system can manipulate the following buffers:Color buffers: front-left, front- right, backleft, back-right, and any number of auxiliary color.

1.2.6 Color Buffers

- The color buffers are usually the ones you draw to. They contain either color-index or
- RGB color data and may also contain alpha values. An OpenGL implementation supports stereoscopic viewing has left and right color buffers for the left and right stereo images.
- If stereo isn't supported, only the left buffers are used. Similarly, double-buffered systems have front and back buffers, and a single-buffered system has the front buffers only. Every OpenGL implementation must provide a front-left color buffer.

1.2.7 Depth Buffer

The depth buffer stores a depth value for each pixel. As described in "Hidden-Surface Removal Survival Kit" depth is usually measured in terms of distance to the eye, so pixels with larger depth-buffer values are overwritten by pixels with smaller values. This is just a useful convention, however, and the depth buffer's behavior can be modified as described in "Depth Test." The depth buffer is sometimes called the *z buffer* (the z comes from the fact that *x* and *y* values measure horizontal and vertical displacement on the screen, and the *z* value measures distance perpendicular to the screen).

1.2.8 Clearing Buffers

The OpenGL clearing commands are structured to take advantage of architectures like this. First, you specify the values to be written into each buffer to be cleared. Then you issue a single command to perform the clear operation, passing in a list of all the buffers to be cleared. If the hardware is capable of simultaneous clears, they all occur at once; otherwise, each buffer is cleared sequentially. The following commands set the clearing values for each buffer. void glClearColor(GLclampf red, GLclampf green, GLclampf blue, GLclampf alpha);

1.2.9 Depth Test

For each pixel on the screen, the depth buffer keeps track of the distance between the viewpoint and the object occupying that pixel. Then, if the specified depth test passes, the incoming depth value replaces the one already in the depth buffer. The depth buffer is usually used for hidden-surface elimination. To use the depth buffer, you simply have to enable it by passing GL_DEPTH_TEST to glEnable() and remember to clear the depth buffer before you redraw each frame (see "Clearing Buffers"). You can also choose a different comparison function for the depth test with glDepthFunc().void glDepthFunc(GLenum func);

1.2.10 Problem Statement

To design and develop the working of Rubik"s Cube using openGL.

1.2.11 Scope

This project illustrates the use of various OpenGL features such as keyboard interaction, mouse interaction and Menus.

1.3 Mini Project Description

In our package we are clearly going to demonstrate the working of "Rubik's Cube Game ". The project improvised on this basic idea to convert a normal 2 Dimensional game into a graphical 3 Dimensional game. The original game is a 2 Dimensional game that involves swapping of any 2 tiles and arranging the tiles in the correct sequence. We modified this idea to include a blank position. This now increases level of difficulty as only the tiles adjacent to the blank position can now be translated, unlike the Puzzle Slider game.

CHAPTER 2

REQUIREMENT SPECIFICATION

2.1 Project Requirements

The project is designed such that users with a computer having minimum configuration can also use it. It doesn't require many complex graphics packages. The project requires simple built-in functions found in the <glut.h> library

2.2 Hardware Requirements

The physical components required are:

- Processor - Pentium Pro
- Memory - 128MB RAM
- 40GB Hard Disk Drive

2.3 Software Requirements

The software used in building this program are:

- Operating system - Windows
- Tools : Codeblocks
- Graphics Library – glut.h
- OpenGL 2.0

CHAPTER 3

DESIGN

ALGORITHM

STEP 1: Define vertices for 27 cubes which are used to compose one whole cube known as "Rubik's cube".

STEP 2: Define colors for each cube of the Rubiks cube to distinguish one cube from the other and as well as color for the speed meter, which is used to control the speed of rotation.

Output( ) function:
STEP 3: This function is used to display the message using the Commands glutBitmapCharacter( ) and glRasterpos*( ).

Polygon( ) function:
STEP 4: Draw a polygon via list of vertices with the line of 3 pixels wide.

Colorcube( ) function:
STEP 5: Map vertices to faces. Hence the use of colorcube function has done 27 times with different prefixes for all the 27 cubes amongst which one with no color, 6 with one color, 12 with two colors and 8 with three colors on it.

Speedmeter( ) function:
STEP 6: This function is used to define vertices for a speedmeter, which is used to control the speed of rotation.

Display( ) function:
STEP 7: Clear the frame buffers and the z-buffer.

STEP 8: Invoke the function speedmeter and output.

STEP 9: Using the variables rotation and inverse the rotation for the faces are defined, where if rotation is one and the inverse flag is zero then the top face of the cube will be rotated in the clockwise direction and incase if rotation is one and also is the inverse flag then the top face of the cube will be rotated in the anti-clockwise direction.
STEP 10: The same procedure that has been specified in the step 9 is adopted with different values for the rotation, such as rotation with the value two is implemented for right three for front, four for left, five for back and six for bottom rotations respectively with the implementation of inverse variable being same.

STEP 11: Invoke the output function and swap the buffers.

Transpose( ) function:
STEP 12: This function is used to define the transpose for all the six faces.

Topc( ) function:
STEP 13: This function is used to assign the values when the operation is rotation of the top face.

Frontc( ) function:
STEP 14: This function is used to assign the values when the operation is rotation of the front face.

Rightc( ) function:
STEP 15: This function is used to assign the values when the operation is rotation of the right face.

Leftc( ) function:
STEP 16: This function is used to assign the values when the operation is rotation of the left face.

Backc( ) function:
STEP 17: This function is used to assign the values when the operation is rotation of the back face.

Bottomc( ) function:
STEP 18: This function is used to assign the values when the operation is rotation of the bottom face.

Spincube( ) function:
STEP 19: This is function is an idle callback which rotates the cube accordingly, if rotation is one and inverse is zero then rotate the top face of the cube by 90 degrees in the clockwise direction and if inverse is one then rotate the same face by 90 degree in the anti-clockwise direction.

STEP 20: The same procedure is implemented for the other faces of the cube with different rotation values.

STEP 21: The cube is redisplayed after the rotation.

Motion( ) function:
STEP 22: This is used to rotate the cube about the selected axis.

Mouse( ) function:
STEP 23: Mouse callback function. This allows user to give input through mouse buttons.

Keyboard( ) function:
STEP 24: use the keys 'a','s','d','f','g','h' to rotate the faces of the accordingly in the clockwise direction and the keys 'q','w','e','r','t','y' to rotate in the anti-clockwise direction.

STEP 25: use the keys '1','2','4','5','6','8','9' are used to move the viewer along the axis.

STEP 26: use the keys 'm' and 'n' to alter the value of the rotation meter which controls the speed of the rotation.

STEP 27: use the key 'o' for the automatic solving of the cube.

Myreshape( ) function:
STEP 28: Define a viewport and set the matrix to projection and modelview matrices.

Mymenu( ) function:
STEP 29: Define the actions corresponding to each entry in the menu.

Main( ) function:
STEP 30: Both double and z-buffer is enabled. Invoke the start function.

STEP 31: Add entries to the menu and link the menu to the right mouse button.

STEP 32: Stop.

CHAPTER 4

IMPLEMENTATION

4.1 Few functions used:

- **file_gen** is used to check whether high score file already exists. If it does not, it will create. If it does, it will read data from existing file.
- **render_header** to display the player's name along with a running timer. The timer gets its values from the clock class, which maintains the time elapsed since the game started in hour, minutes and seconds.
- **hiscore_check** to determine the position of the player in the highscores list. If he completes the game in a time lesser than the top 10 players, it implements an insertion sort to insert the player's name and time into the existing high score list.
- **write_file** is used to write a text file of the current player class. The player class contains details of the player name and the time he took to complete the game. All these details, along with the pre-existing records of the highscores are arranged in ascending order of time and are written into a text file.
- **sub** is the display function for the sub window created to display the highscores. This window is activated when the user chooses the option "highscores" from the menu obtained on right click of the mouse.

4.2 Mouse events:

- When mouse event occurs, the ASCII code for the corresponding coordinates that generate the event and the location of mouse are returned.
- Mouse callback function is glutMouseFunc (mouse);
- Void mouse (int btn, int state, int x, int y)

```
{
        if (button == GLUT_LEFT_BUTTON && state == GLUT_DOWN)
                begin = x;
}
```

4.3 Menu Entry:

GLUT provides one additional feature, pop_up menus, which we can use with the mouse to create sophisticated interactive application glutCreateMenu (); ,glutAddMenuEntry (); and glutAttachMenu (GLUT_RIGHT_BUTTON);

CHAPTER 5

RESULTS

5.1 SCREENSHOTS

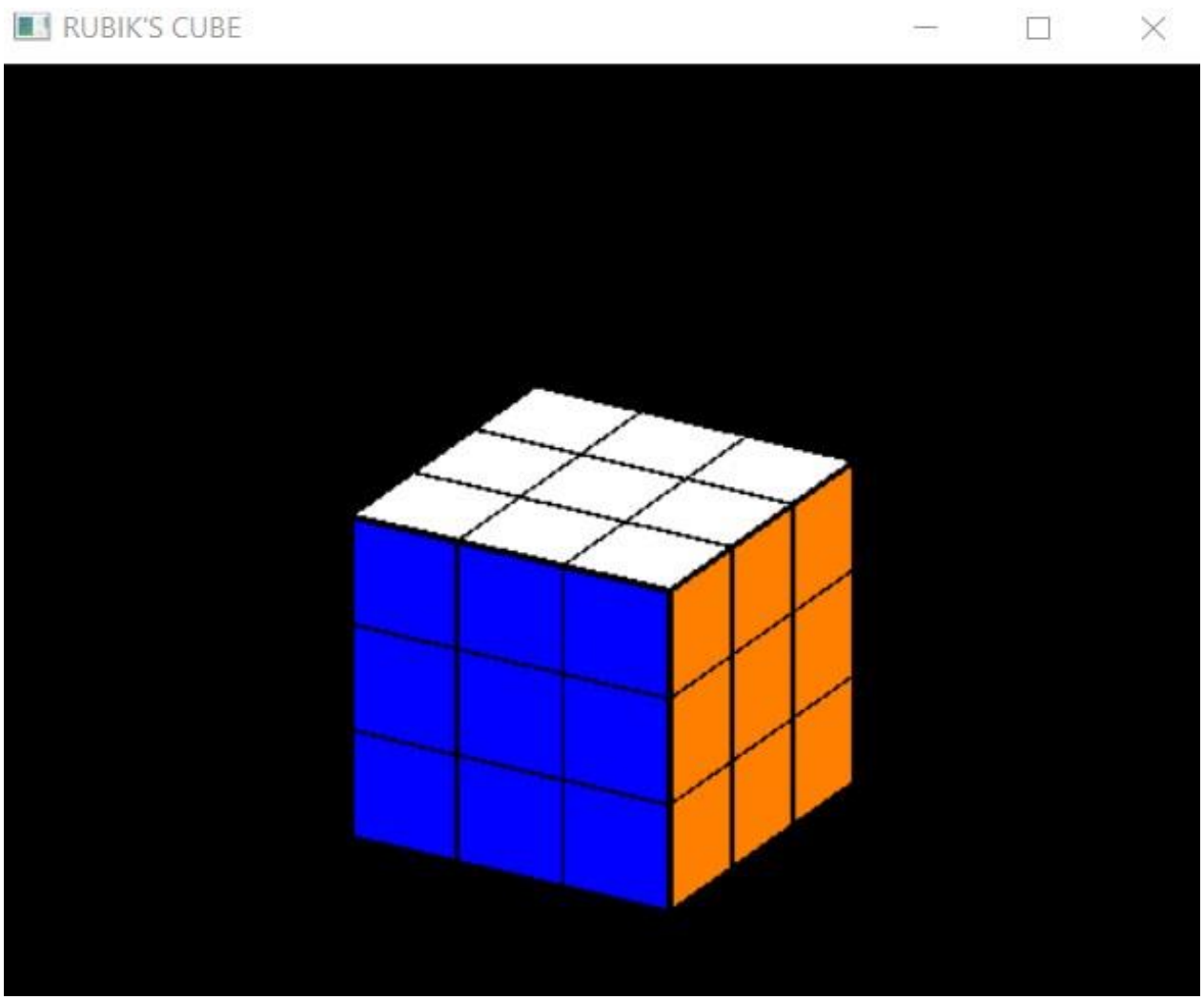

Figure 5.1 Display Page

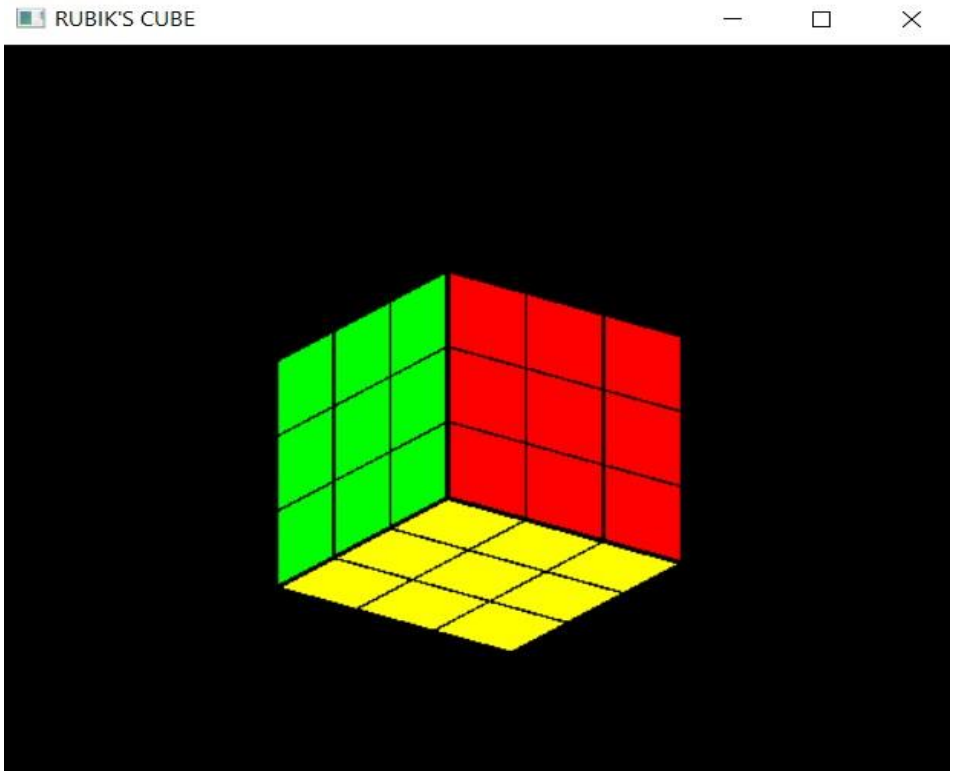

Figure 5.2: Other Side Of Cube

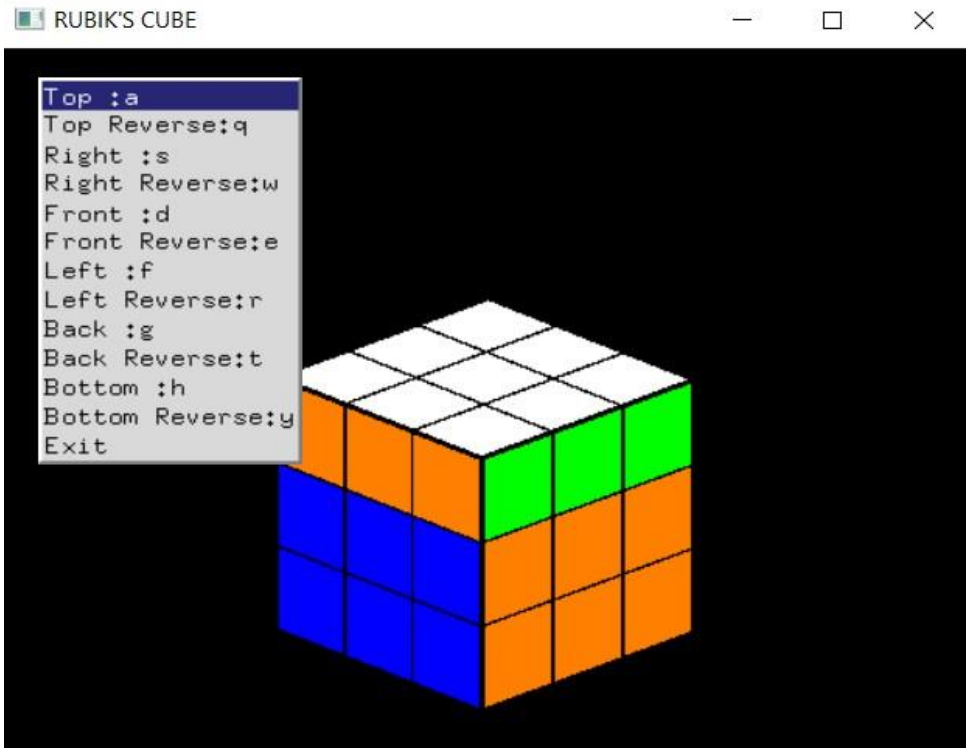

Figure 5.3: Top Rotation (Clockwise)

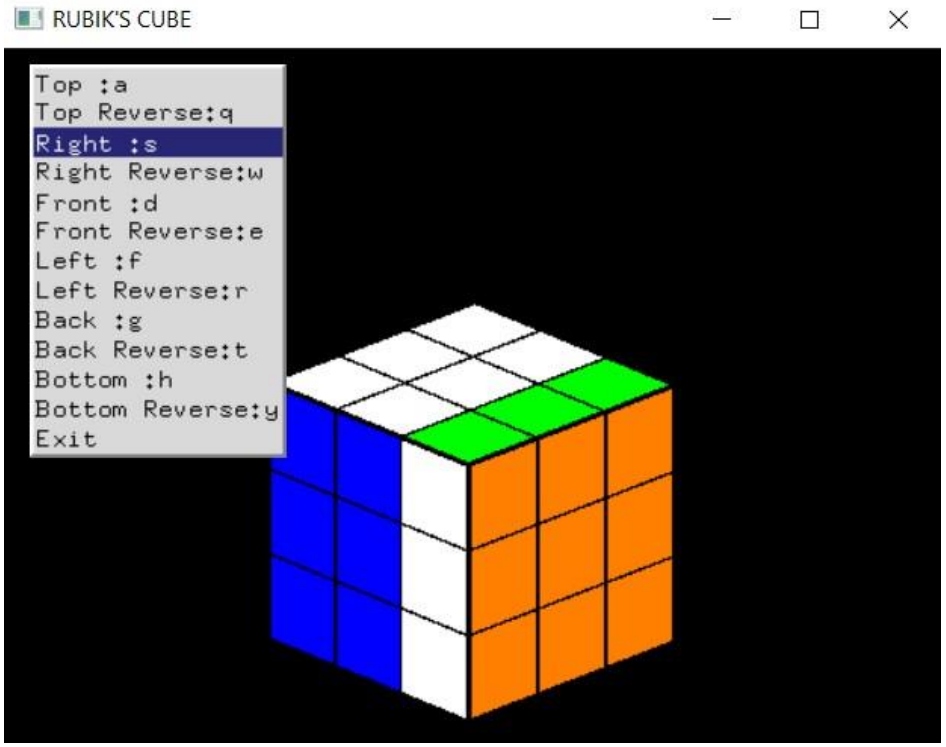

Figure 5.4: Right Rotation (Clockwise)

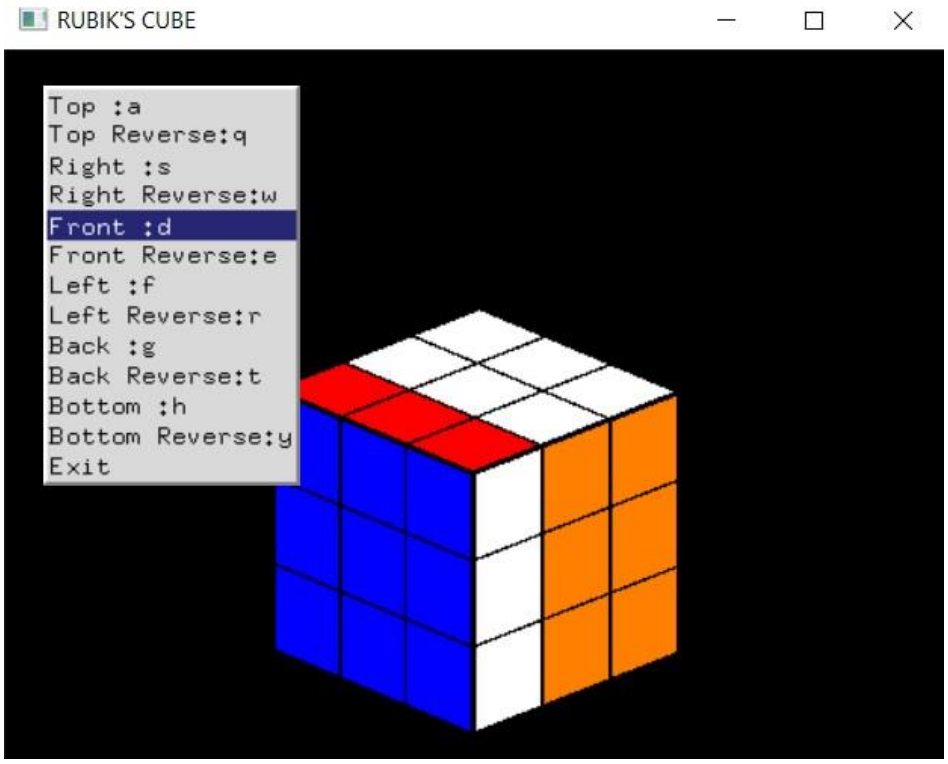

Figure 5.5: Front Rotation (Clockwise)

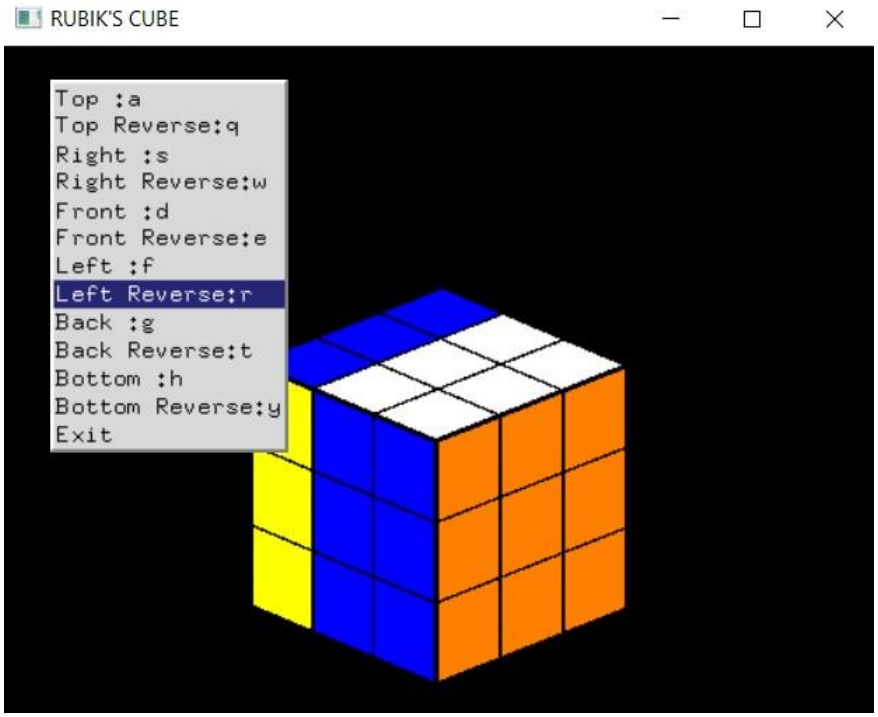

Figure 5.6: Left Rotation (Anti-clockwise)

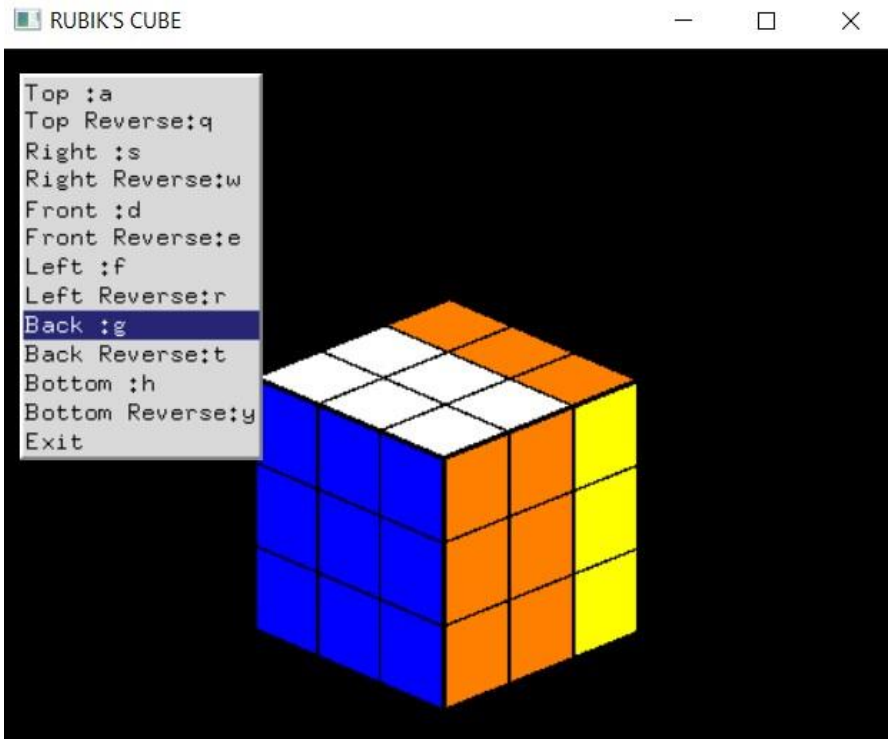

Figure 5.7: Back Rotation (Clockwise)

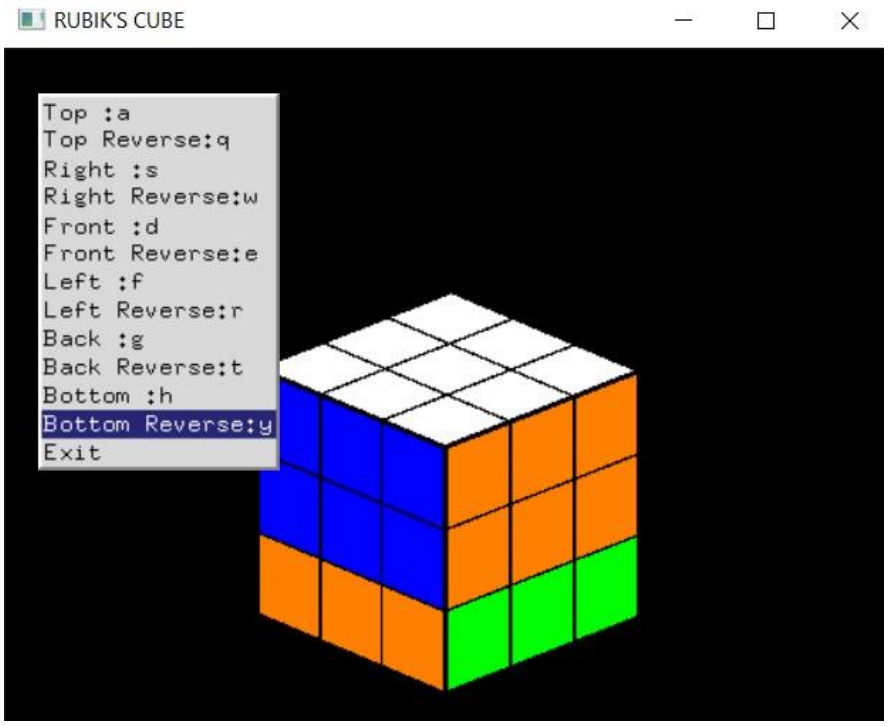

Figure 5.8: Bottom Rotation (Anti-clockwise)

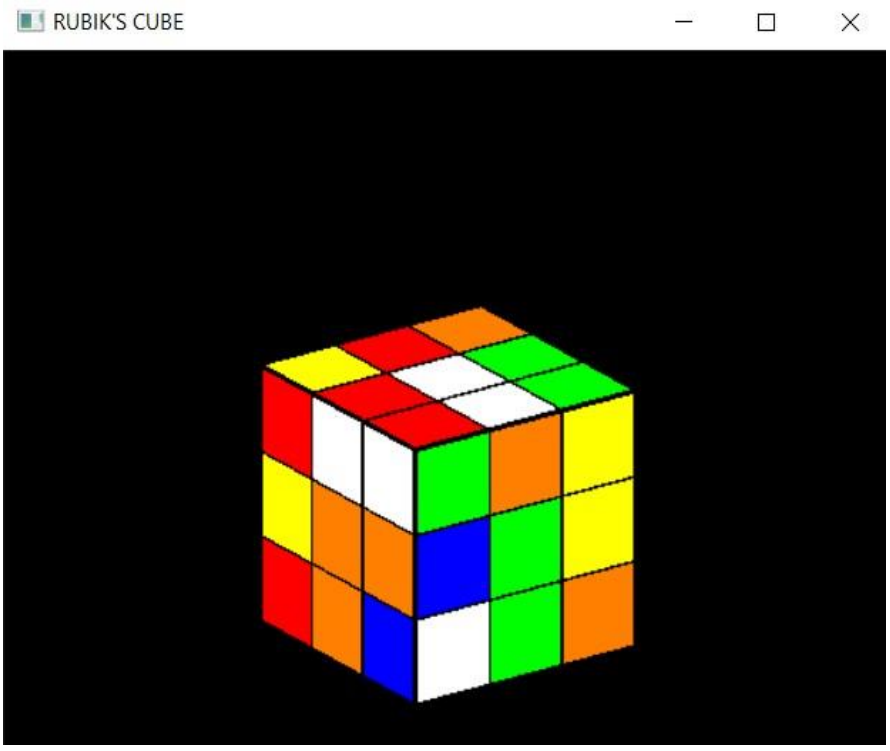

Figure 5.9: Cube to be solved!!

CHAPTER 6

CONCLUSION & FUTURE ENHANCEMENTS

We have successfully implemented the Rubik's Cube game with tiles and frames. We have also provided features so that the user can view and solve the cube as per his requirements.

FUTURE IDEAS

- Adding cheat codes like most commercial games, so that the player can enter them and the fully solved puzzle is displayed. This can also be used to view the effects when a player makes it to the Hall of Fame.
- Give the option of allowing the user to put any image of his choice onto the tiles. This again involves Texture mapping and image processing.
- Implementing different levels with increasing levels of difficulty.
- The extra features can include Player's name and a clock that starts at the beginning of the game.
- To make the high score list permanent, a text file is running at the back end.

REFERENCES

Websites Referred

- www.opengl.com
- www.sourcecode.com