Kathmandu University

Department of Computer Science and Engineering

Dhulikhel, Kavre

A Mini-Project Report on

"Color Cube Flyby-C++"

[COMP 306]

(For partial fulfillment of 3nd Year/2nd Semester in Computer Engineering)

Submitted by:

Prajwol Manandhar(32)

Submitted to:

Dhiraj Shrestha

Assistant Professor

Department of Computer Science & Engineering (DoCSE)

Submission Date:

May 21, 2022

Table of Contents

Chapter1: Introduction

ColorCube Flyby as stated in the name is a project in which we rotate the perspective of the vision about a 3d multicolored cube in a closed orbit. Here the Cube is in the fixed space with fixed color in its faces and vertices. Then we look in the direction of the center of the cube while we are moving about an orbit around the cube.

1.1 Introduction of Library Used(OpenGL)

We have used GLUT library and C++ language along with their respective libraries to complete the project.

GLUT is the OpenGL Utility Toolkit, a window system independent toolkit for writing OpenGL programs. It implements a simple windowing application programming interface (API) for OpenGL. GLUT makes it considerably easier to learn about and explore OpenGL Programming.

C++ is a general-purpose programming language and is widely used nowadays for competitive programming. It has imperative, object-oriented and generic programming features. C++ runs on lots of platforms like Windows, Linux, Unix, Mac etc.

Chapter2: Code Breakdown:

2.1 Libraries

```
1
2     #ifdef __APPLE__
3     #include <GLUT/glut.h>
4     #else
5     #include <windows.h>
6     #include <GL/glut.h>
7     #endif
8     #include <cmath>
```

We have used <windows.h>, <glut.h>, <cmath> libraries:

<Windows.h> : Helps the code to work with windows OS

<glut.h> : Library for all the OpenGL Codes

<cmath> : C++ library for common mathematical operations and tarnsformations

2.2 Defining Cube Properties

```
namespace Cube {

const int NUM_VERTICES = 8;
const int NUM_FACES = 6;
```

2.3 Defining Cube Vertex Positions

```
18    GLint vertices[NUM_VERTICES][3] = {
19        {0, 0, 0}, {0, 0, 1}, {0, 1, 0}, {0, 1, 1},
20        {1, 0, 0}, {1, 0, 1}, {1, 1, 0}, {1, 1, 1}};
```

2.4 Defining Cube Faces

```
22    GLint faces[NUM_FACES][4] = {
23        {1, 5, 7, 3}, {5, 4, 6, 7}, {4, 0, 2, 6},
24        {3, 7, 6, 2}, {0, 1, 3, 2}, {0, 4, 5, 1}};
```

2.5 Defining Vertex Colors (R, G, B)

```
GLfloat vertexColors[NUM_VERTICES][3] = {
    {0.0, 0.0, 0.0}, {0.0, 0.0, 1.0}, {0.0, 1.0, 0.0}, {0.0, 1.0, 1.0},
    {1.0, 0.0, 0.0}, {1.0, 0.0, 1.0}, {1.0, 1.0, 0.0}, {1.0, 1.0, 1.0}};
```

2.6 Defining a Cube Drawing Function

```
void draw() {
    glBegin(GL_QUADS);
    for (int i = 0; i < NUM_FACES; i++) {
        for (int j = 0; j < 4; j++) {
            glColor3fv((GLfloat*)&vertexColors[faces[i][j]]);
            glVertex3iv((GLint*)&vertices[faces[i][j]]);
        }
    }
    glEnd();
}
}
```

2.7 Clearing Buffers

```
void display() {
    glClear(GL_COLOR_BUFFER_BIT);
    Cube::draw();
    glFlush();
    glutSwapBuffers();
}
```

This function clears the window to draw the cube and is used to achieve the animation by successively drawing presenting the view as in frames based on the movement of the camera.

2.8 Setting timer

```
void timer(int v) {
    static GLfloat u = 0.0;
    u += 0.01;
    glLoadIdentity();
    gluLookAt(8*cos(u), 7*cos(u)-1, 4*cos(u/3)+2, .5, .5, .5, cos(u), 1, 0);
    glutPostRedisplay();
    glutTimerFunc(1000/60.0, timer, v);
}
```

2.9 Setting Perspective of Camera

```
60  void reshape(int w, int h) {
61      glViewport(0, 0, w, h);
62      glMatrixMode(GL_PROJECTION);
63      glLoadIdentity();
64      gluPerspective(60.0, GLfloat(w) / GLfloat(h), 0.5, 40.0);
65      glMatrixMode(GL_MODELVIEW);
66  }
```

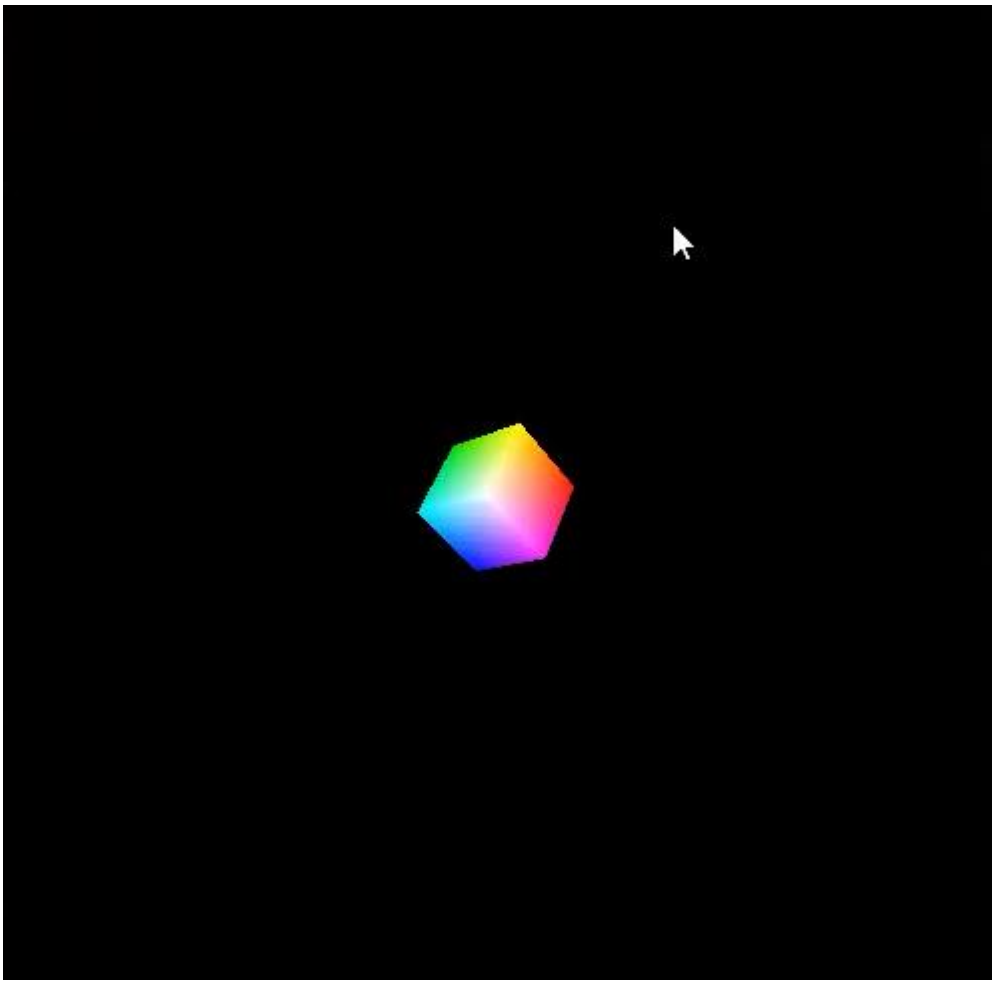
2.10 Backface Culling

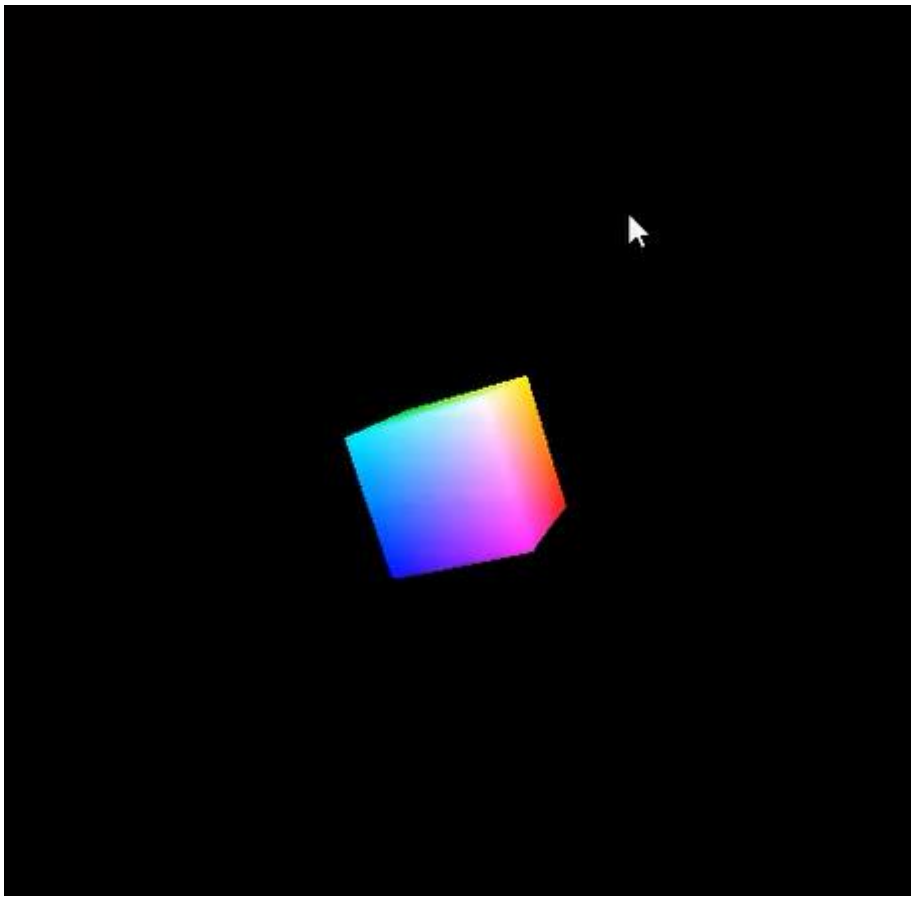
```
69  void init() {
70      glEnable(GL_CULL_FACE);
71      glCullFace(GL_BACK);
72  }
```

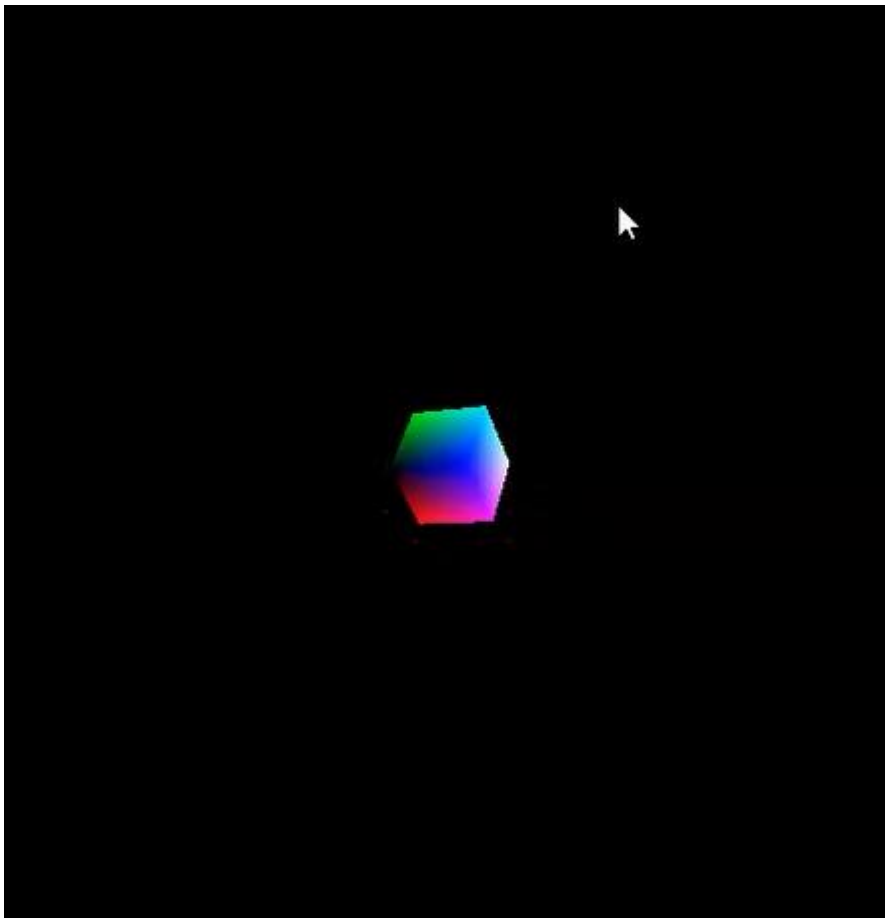
2.11 Defining the main() Funciton

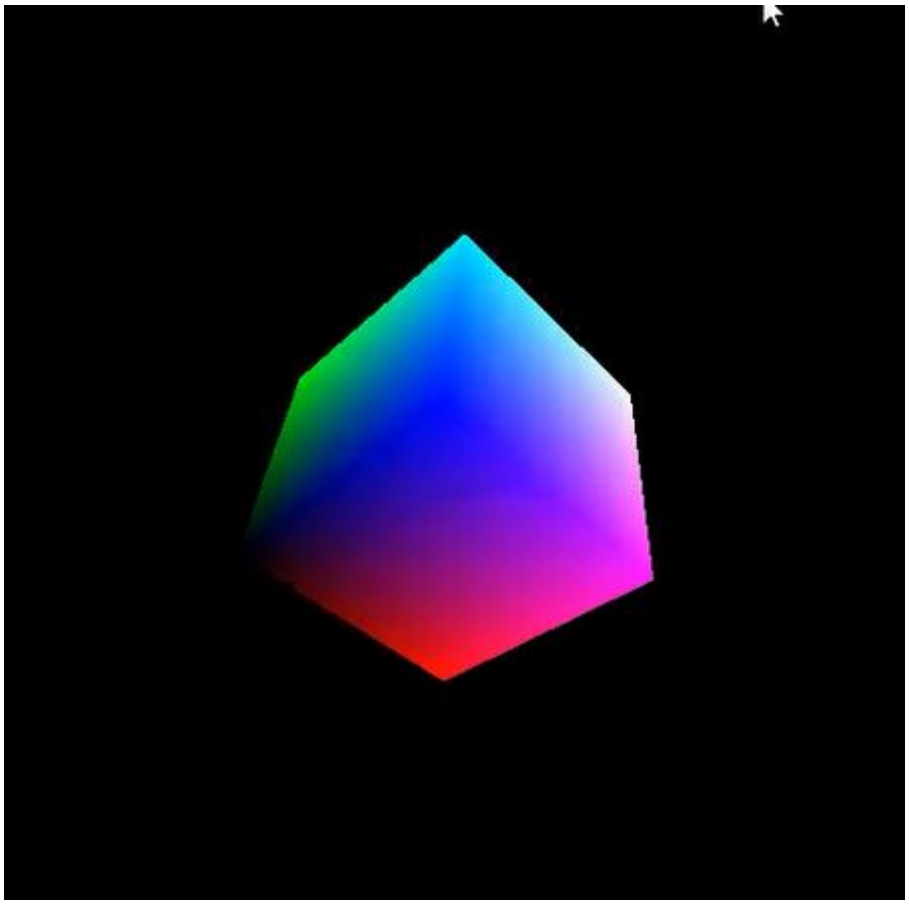
```
74  int main(int argc, char** argv) {
75      glutInit(&argc, argv);
76      glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);
77      glutInitWindowSize(500, 500);
78      glutCreateWindow("The RGB Color Cube");
79      glutReshapeFunc(reshape);
80      glutTimerFunc(100, timer, 0);
81      glutDisplayFunc(display);
82      init();
83      glutMainLoop();
84  }
```

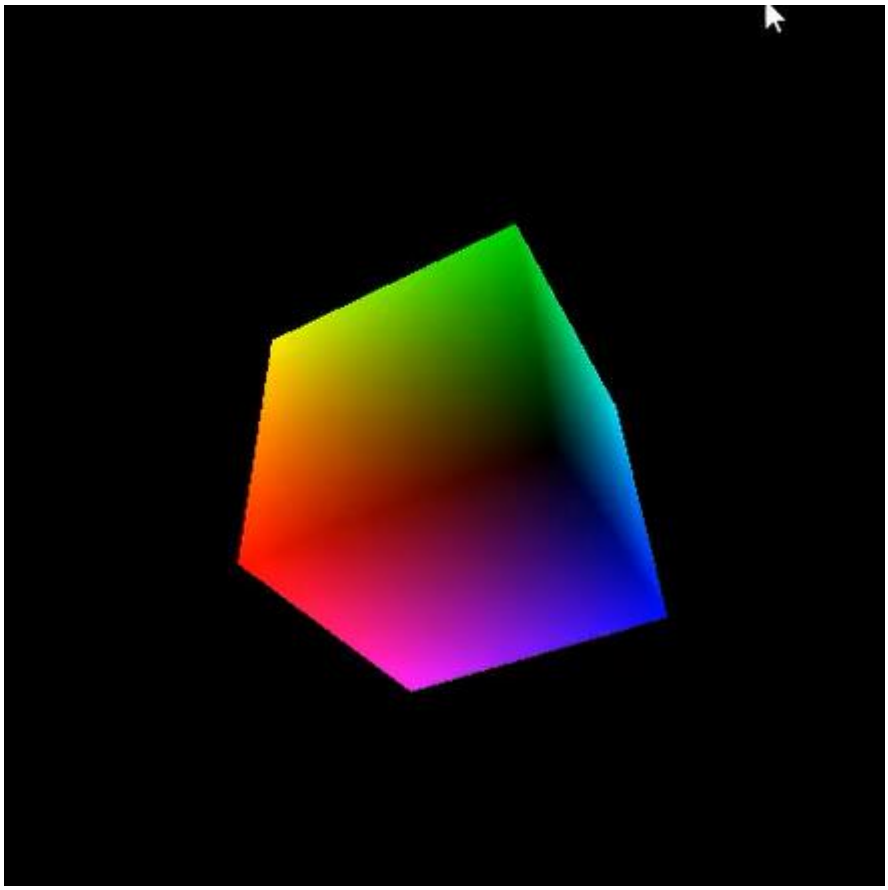
Output:











screen-capture.webm

Additional Materials

The video of the code explanation and the output running is in the given git hub link:

https://github.com/PrajwolM/Colocube-Flyby.git

Appendices:

```
1
2     #ifdef __APPLE__
3     #include <GLUT/glut.h>
4     #else
5     #include <windows.h>
6     #include <GL/glut.h>
7     #endif
8     #include <cmath>
9     #include <stdlib.h>
10
11
12
13    namespace Cube {
14
15    const int NUM_VERTICES = 8;
16    const int NUM_FACES = 6;
17
18    GLint vertices[NUM_VERTICES][3] = {
19      {0, 0, 0}, {0, 0, 1}, {0, 1, 0}, {0, 1, 1},
20      {1, 0, 0}, {1, 0, 1}, {1, 1, 0}, {1, 1, 1}};
21
22    GLint faces[NUM_FACES][4] = {
23      {1, 5, 7, 3}, {5, 4, 6, 7}, {4, 0, 2, 6},
24      {3, 7, 6, 2}, {0, 1, 3, 2}, {0, 4, 5, 1}};
25
26    GLfloat vertexColors[NUM_VERTICES][3] = {
27      {0.0, 0.0, 0.0}, {0.0, 0.0, 1.0}, {0.0, 1.0, 0.0}, {0.0, 1.0, 1.0},
28      {1.0, 0.0, 0.0}, {1.0, 0.0, 1.0}, {1.0, 1.0, 0.0}, {1.0, 1.0, 1.0}};
29
```

```
31  void draw() {
32    glBegin(GL_QUADS);
33    for (int i = 0; i < NUM_FACES; i++) {
34      for (int j = 0; j < 4; j++) {
35        glColor3fv((GLfloat*)&vertexColors[faces[i][j]]);
36        glVertex3iv((GLint*)&vertices[faces[i][j]]);
37      }
38    }
39    glEnd();
40  }
41  }
42
43  void display() {
44    glClear(GL_COLOR_BUFFER_BIT);
45    Cube::draw();
46    glFlush();
47    glutSwapBuffers();
48  }
49
50
51  void timer(int v) {
52    static GLfloat u = 0.0;
53    u += 0.01;
54    glLoadIdentity();
55    gluLookAt(8*cos(u), 7*cos(u)-1, 4*cos(u/3)+2, .5, .5, .5, cos(u), 1, 0);
56    glutPostRedisplay();
57    glutTimerFunc(1000/60.0, timer, v);
58  }
```

```
59
60  void reshape(int w, int h) {
61    glViewport(0, 0, w, h);
62    glMatrixMode(GL_PROJECTION);
63    glLoadIdentity();
64    gluPerspective(60.0, GLfloat(w) / GLfloat(h), 0.5, 40.0);
65    glMatrixMode(GL_MODELVIEW);
66  }
67
68
69  void init() {
70    glEnable(GL_CULL_FACE);
71    glCullFace(GL_BACK);
72  }
73
74  int main(int argc, char** argv) {
75    glutInit(&argc, argv);
76    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);
77    glutInitWindowSize(500, 500);
78    glutCreateWindow("The RGB Color Cube");
79    glutReshapeFunc(reshape);
80    glutTimerFunc(100, timer, 0);
81    glutDisplayFunc(display);
82    init();
83    glutMainLoop();
84  }
85
```