Assignment-06

Cube

```
#include <GL/glut.h>

#include <iostream>


typedef int index[4];

typedef GLfloat color[3];


using namespace std;



static GLfloat verts[][4] = {

{ 1.0,  1.0,  1.0},

{-1.0,  1.0,  1.0},

{-1.0, -1.0,  1.0},

{ 1.0, -1.0,  1.0},

{ 1.0,  1.0, -1.0},

{-1.0,  1.0, -1.0},

{-1.0, -1.0, -1.0},

{ 1.0, -1.0, -1.0},

};
```

```
int rotx = 0, roty = 0, rotz = 0;
float transx = 0, transy = 0, transz = 0;
float sfactor = 1.0;
unsigned int primtype = GL_POLYGON;

void keyboardHandler(unsigned char c, int x, int y) {
switch (c)
{
case '\e':
exit(0);
break;
case '8':
rotx -= 5;
break;
case '2':
rotx += 5;
break;

case '6':
roty += 5;
```

```
break;

case '4':
roty -= 5;
break;

case '7':
rotz += 5;
break;

case '9':
rotz -= 5;
break;

case '5':
rotx = 0;
roty = 0;
rotz = 0;
break;

case 'w':
```

```
transx -= 0.5;

break;

case 's':

transx += 0.5;

break;


case 'a':

transy += 0.5;

break;


case 'd':

transy -= 0.5;

break;


case 'e':

transz += 0.5;

break;


case 'q':

transz -= 0.5;

break;
```

```
case 'r':
transx = 0;
transy = 0;
transz = 0;
break;

case '+':
sfactor+=0.2;
break;
case '-':
sfactor-=0.2;
break;

default:
break;
}

rotx = rotx % 360;
roty = roty % 360;
rotz = rotz % 360;
```

```
}

void drawface(const index &indices, const color& clr) {
glBegin(primtype);
glColor3fv(clr);
for (unsigned int i = 0; i < sizeof(indices) / sizeof(unsigned int);
i++)
{
glVertex3fv(verts[indices[i]]);
}
glEnd();
}

void display(void)
{
glMatrixMode(GL_MODELVIEW);
glPushMatrix();
// rot=0.1*(GLfloat)glutGet(GLUT_ELAPSED_TIME);

glRotatef(rotx, 1.0f, 0.0f, 0.0f);
glRotatef(roty, 0.0f, 1.0f, 0.0f);
glRotatef(rotz, 0.0f, 0.0f, 1.0f);

glRotatef(-45, 0.0f, 1.0f, 0.0f);
glRotatef(30, 1.0f, 0.0f, 0.0f);
glTranslatef(transx, transy, transz);
glScalef(sfactor, sfactor, sfactor);

glClear(GL_COLOR_BUFFER_BIT |
GL_DEPTH_BUFFER_BIT);
glColor3f(0.0, 1.0, 1.0);

drawface({0,1,2,3},{1.0,0.0,0.0});
drawface({4,5,6,7},{1.0,0.0,1.0});

drawface({0,4,7,3},{0.0,0.0,1.0});
drawface({1,5,6,2},{0.0,1.0,1.0});

drawface({0,1,5,4},{0.0,1.0,0.0});
drawface({3,2,6,7},{1.0,1.0,0.0});

glPopMatrix();
glutSwapBuffers();
```

```
}

int main(int argc, char** argv) {
glutInit(&argc, argv);                    // initialises glut
glutInitDisplayMode(                      // used to set display modes
GLUT_DOUBLE |                             // has a single buffer; double
buffers not required as static scene
GLUT_RGB |                                // sets color mode to RGB
GLUT_DEPTH                                // enables depth, used for
Z-buffer(for glut)
);
glutInitWindowPosition(100, 100);         // sets window position
on screen
glutInitWindowSize(500, 500);             // sets window size
glutCreateWindow("GLTUT: cube");          // creates and sets the
title of window
glClearColor(0.0, 0.0, 0.0, 0.0);         // black background, used
everytime glClear(); is called
glMatrixMode(GL_PROJECTION);              // setup viewing
projection
glLoadIdentity();                         // start with identity matrix
```

```
glOrtho(-5.0, 5.0, -5.0, 5.0, -5.0, 5.0);   // setup a 10x10x2
viewing world
glEnable(GL_DEPTH_TEST);                    // enables depth in
opengl

glutDisplayFunc(display);
glutIdleFunc(display);
glutKeyboardFunc(keyboardHandler);
glutMainLoop();
return 0;
}
```

Explanation of the OpenGL Cube Rotation Program

This C++ program uses OpenGL and the GLUT library to create a 3D cube that can be rotated, translated, and scaled using keyboard inputs. The user can control the cube's transformations via specific keys, allowing for interactive 3D visualization.

Key Components of the Code

1.OpenGL Initialization:

•The program initializes GLUT and sets up a window for rendering.

•It specifies the display mode, including double buffering and RGB color mode.

•It sets the background color to black and defines the orthographic projection matrix for viewing the cube.

2.Global Variables:

•rotx, roty, rotz: Angles for rotation around the x, y, and z axes.

•transx, transy, transz: Translation offsets along the x, y, and z axes.

•sfactor: Scaling factor for the cube.

•primtype: Primitive type used for drawing (polygons in this case).

3.Keyboard Input Handling:

•The keyboardHandler function captures user inputs to modify rotation, translation, and scaling of the cube:

•Arrow keys (8, 2, 4, 6, 7, 9) adjust rotation.

•w, s, a, d, e, q control translation along the axes.

•+ and - adjust the scaling factor.

4.Drawing the Cube:

•The drawface function takes vertex indices and colors to draw each face of the cube.

•The display function sets up the transformation matrix, applies the rotations, translations, and scaling, and clears the screen for redrawing.

5.Main Function:

•Initializes GLUT and registers the display, idle, and keyboard functions.

•Enters the GLUT main loop to keep the window open and responsive to user inputs.