**Homework 3 Report**

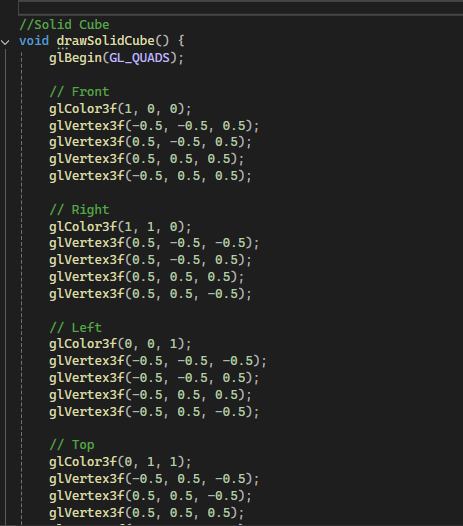
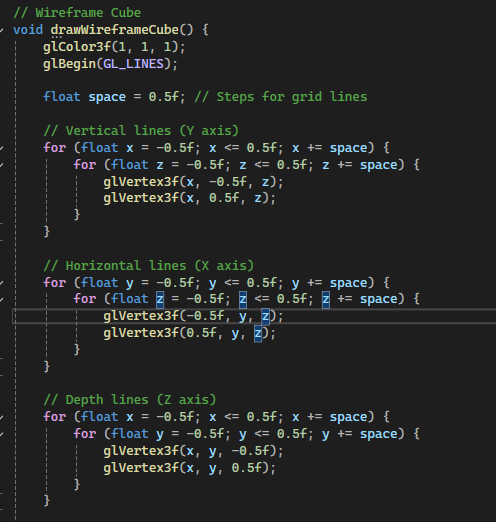
D11315807

Ardiawan Bagus Harisa

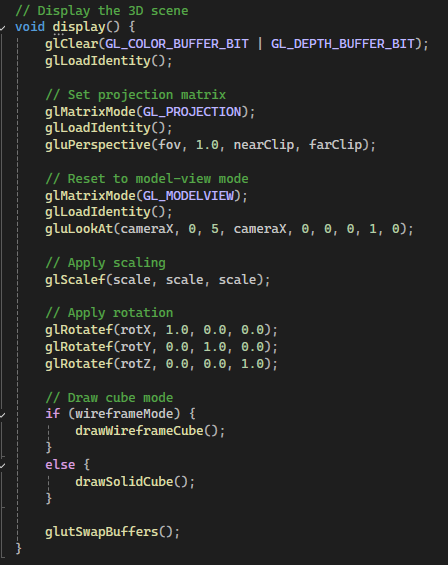
Department of CSIE

In this homework, we are asked to do the basic projection in a 3D scene.

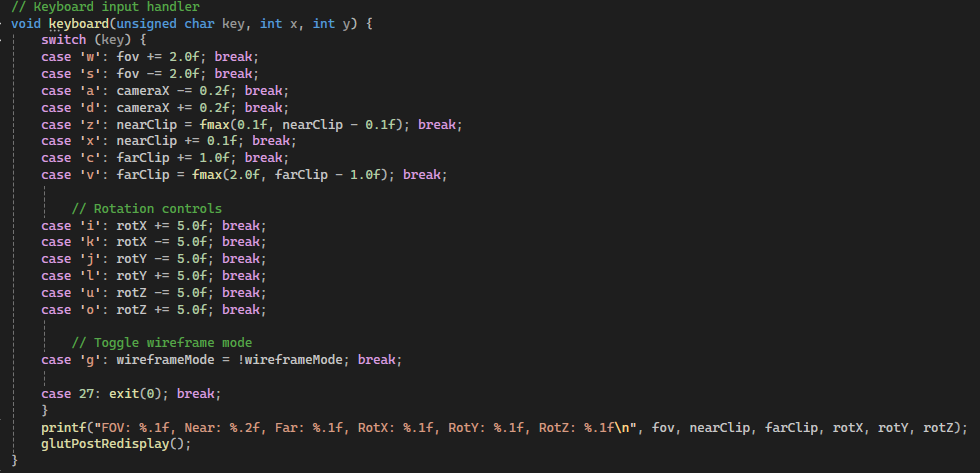
What I do first is creating a cube using GL\_QUADS. Basically, I just draw six rectangles. I want the user to be able to switch between the solid mode or wireframe mode. So, instead of GL\_QUADS, I use GL\_LINES to draw the line in wireframe mode. Specifically, in drawWireframeCube(), first I determine the space or the steps to draw the line. Then, for each destined steps on Y axis (from bottom to top), I draw a vertical line. Then, I also draw the horizontal line (x axis). Finally, I draw the z axis line as well. These two drawing functions will be called on display().

On the display(), we will display the previously drawing functions and also apply some transformation. The first thing I do is configure the projection matrix, including the perspective projection, aspect ratio, near and far clipping plane. gluLookA() set the camera view positioned at (cameraX, 0, 5) and looks at (cameraX, 0, 0), with the up vector pointing along the positive Y-axis (0, 1, 0). Right after, I apply the transformation of scale and rotation. Finally, according whether the user press the “g” button, the drawing cube mode is changed accordingly.



To do the projection operations, I implement some keyboard keys as the control. Finally, just call the display(), keyboard() functions.



|  |  |
| --- | --- |
| Initial display (front view) | Near clipping |
| Far clipping | Zoom |
| Rotation | Wireframe |

Notes:

* I push my code here: <https://github.com/ardiawanbagusharisa/cgopengl>

The complete code:

#

#include <GL/glew.h>

#include <GL/freeglut.h>

#include <cmath>

#include <stdio.h>

// Parameters for perspective projection

float fov = 60.0f; // Field of view

float nearClip = 0.1f; // Near clipping plane

float farClip = 10.0f; // Far clipping plane

float scale = 1.0f; // Scaling factor

float cameraX = 0.0f; // Camera's X position

float rotX = 0.0f, rotY = 0.0f, rotZ = 0.0f;

bool wireframeMode = false;

void initGL() {

glEnable(GL\_DEPTH\_TEST);

glClearColor(0.0f, 0.0f, 0.0f, 1.0f);

}

//Solid Cube

void drawSolidCube() {

glBegin(GL\_QUADS);

// Front

glColor3f(1, 0, 0);

glVertex3f(-0.5, -0.5, 0.5);

glVertex3f(0.5, -0.5, 0.5);

glVertex3f(0.5, 0.5, 0.5);

glVertex3f(-0.5, 0.5, 0.5);

// Right

glColor3f(1, 1, 0);

glVertex3f(0.5, -0.5, -0.5);

glVertex3f(0.5, -0.5, 0.5);

glVertex3f(0.5, 0.5, 0.5);

glVertex3f(0.5, 0.5, -0.5);

// Left

glColor3f(0, 0, 1);

glVertex3f(-0.5, -0.5, -0.5);

glVertex3f(-0.5, -0.5, 0.5);

glVertex3f(-0.5, 0.5, 0.5);

glVertex3f(-0.5, 0.5, -0.5);

// Top

glColor3f(0, 1, 1);

glVertex3f(-0.5, 0.5, -0.5);

glVertex3f(0.5, 0.5, -0.5);

glVertex3f(0.5, 0.5, 0.5);

glVertex3f(-0.5, 0.5, 0.5);

// Bottom

glColor3f(1, 0, 1);

glVertex3f(-0.5, -0.5, -0.5);

glVertex3f(0.5, -0.5, -0.5);

glVertex3f(0.5, -0.5, 0.5);

glVertex3f(-0.5, -0.5, 0.5);

// Back

glColor3f(0, 1, 0);

glVertex3f(-0.5, -0.5, -0.5);

glVertex3f(0.5, -0.5, -0.5);

glVertex3f(0.5, 0.5, -0.5);

glVertex3f(-0.5, 0.5, -0.5);

glEnd();

}

// Wireframe Cube

void drawWireframeCube() {

glColor3f(1, 1, 1);

glBegin(GL\_LINES);

float space = 0.5f; // Steps for grid lines

// Vertical lines (Y axis)

for (float x = -0.5f; x <= 0.5f; x += space) {

for (float z = -0.5f; z <= 0.5f; z += space) {

glVertex3f(x, -0.5f, z);

glVertex3f(x, 0.5f, z);

}

}

// Horizontal lines (X axis)

for (float y = -0.5f; y <= 0.5f; y += space) {

for (float z = -0.5f; z <= 0.5f; z += space) {

glVertex3f(-0.5f, y, z);

glVertex3f(0.5f, y, z);

}

}

// Depth lines (Z axis)

for (float x = -0.5f; x <= 0.5f; x += space) {

for (float y = -0.5f; y <= 0.5f; y += space) {

glVertex3f(x, y, -0.5f);

glVertex3f(x, y, 0.5f);

}

}

glEnd();

}

// Display the 3D scene

void display() {

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

glLoadIdentity();

// Set projection matrix

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluPerspective(fov, 1.0, nearClip, farClip);

// Reset to model-view mode

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

gluLookAt(cameraX, 0, 5, cameraX, 0, 0, 0, 1, 0);

// Apply scaling

glScalef(scale, scale, scale);

// Apply rotation

glRotatef(rotX, 1.0, 0.0, 0.0);

glRotatef(rotY, 0.0, 1.0, 0.0);

glRotatef(rotZ, 0.0, 0.0, 1.0);

// Draw cube mode

if (wireframeMode) {

drawWireframeCube();

}

else {

drawSolidCube();

}

glutSwapBuffers();

}

// Keyboard input handler

void keyboard(unsigned char key, int x, int y) {

switch (key) {

case 'w': fov += 2.0f; break;

case 's': fov -= 2.0f; break;

case 'a': cameraX -= 0.2f; break;

case 'd': cameraX += 0.2f; break;

case 'z': nearClip = fmax(0.1f, nearClip - 0.1f); break;

case 'x': nearClip += 0.1f; break;

case 'c': farClip += 1.0f; break;

case 'v': farClip = fmax(2.0f, farClip - 1.0f); break;

// Rotation controls

case 'i': rotX += 5.0f; break;

case 'k': rotX -= 5.0f; break;

case 'j': rotY -= 5.0f; break;

case 'l': rotY += 5.0f; break;

case 'u': rotZ -= 5.0f; break;

case 'o': rotZ += 5.0f; break;

// Toggle wireframe mode

case 'g': wireframeMode = !wireframeMode; break;

case 27: exit(0); break;

}

printf("FOV: %.1f, Near: %.2f, Far: %.1f, RotX: %.1f, RotY: %.1f, RotZ: %.1f\n", fov, nearClip, farClip, rotX, rotY, rotZ);

glutPostRedisplay();

}

// Special keys handler (for scaling)

void specialKeys(int key, int x, int y) {

if (key == GLUT\_KEY\_UP) scale += 0.1f;

if (key == GLUT\_KEY\_DOWN) scale = fmax(0.1f, scale - 0.1f);

glutPostRedisplay();

}

// Main function

int main(int argc, char\*\* argv) {

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB | GLUT\_DEPTH);

glutInitWindowSize(600, 600);

glutCreateWindow("3D Projection with Wireframe Cube");

glewInit();

initGL();

glutDisplayFunc(display);

glutKeyboardFunc(keyboard);

glutSpecialFunc(specialKeys);

glutMainLoop();

return 0;

}