CS352 Assignment-7

Krishanu Saini
190001029

Write a program in OpenGL for the following:

1) Create a 3-D Box
2) Create a provision to view the house using a mouse and keyboard.
3) Zoom function
4) Change Intensity function
5) Add Image and Text

Program

We will use GlutMouseFunc to find out angles to rotate and use mainloop to rotate given the angle.
We will use glTexImage2D for texture

Code

```
#include <GL/glu.h>
#include <GL/glut.h>
#include <bits/stdc++.h>
#include <sys/unistd.h>
#include <stdlib.h>
#include "imageio.h"

using namespace std;

#include <chrono>
#include <thread>

using namespace std::this_thread; // sleep_for, sleep_until
using namespace std::chrono;      // nanoseconds, system_clock, seconds

/*
   Keys
   Rotate - mouse
   Rotate - arrow keys - Left, Right, Up, Down
   Zoom - F1, F2
   intensity - F3, F4
*/

GLfloat d = 0, dy = 0;
GLfloat zoom = 1;
GLfloat intensity = 0;
GLubyte *textureImage;

int a = 0;
float x = 0.0, y = 0.0, z = 0.0;
float angleX = 0.0, angleY = 0.0;
float xOrigin = -1;
```

```
float yOrigin = -1;
vector<GLfloat> tx(3);

void keyboard(int, int, int);
void shapeTranslate(GLfloat[32][3]);
void MyInit();
void Spin();
void Face(GLfloat[], GLfloat[], GLfloat[], GLfloat[]);
void Cube(GLfloat V0[], GLfloat V1[], GLfloat V2[], GLfloat V3[], GLfloat
V4[], GLfloat V5[], GLfloat V6[], GLfloat V7[]);
void Rotate(GLfloat[32][3], int, GLfloat[32][3]);
void copyMatrix(GLfloat[32][3], GLfloat[32][3]);
void Draw();
void mouseMove(int, int);
void mouseButton(int, int, int, int);

GLuint texture[2];

void init_texture(void)
{
   glClearColor(0.0, 0.0, 0.0, 0.0);
   glEnable(GL_DEPTH_TEST);
   // The following two lines enable semi transparent
   glEnable(GL_BLEND);
   glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);

   int width, height;
   int width1, height1;
   bool hasAlpha;
   char filename[] = "text.png";
   char filename1[] = "car.png";
   // bool success = loadPngImage(filename, width, height, hasAlpha,
&textureImage);
   unsigned char *ibuffer = loadImageRGBA(filename, &width, &height);
   std::cout << "Image loaded " << width << " " << height << " alpha " <<
hasAlpha << std::endl;
   std::cout << "Image loaded " << width1 << " " << height1 << " alpha "
<< hasAlpha << std::endl;
   glPixelStorei(GL_UNPACK_ALIGNMENT, 1);
   glGenTextures(2, texture);
```

```
   cout << texture[0] << " " << texture[1] << endl;
   glBindTexture(GL_TEXTURE_2D, texture[0]);
   glTexImage2D(GL_TEXTURE_2D, 0, 2, width,
                height, 0, GL_RGBA, GL_UNSIGNED_BYTE,
                ibuffer);
   glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
   glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);

   ibuffer = loadImageRGBA(filename1, &width, &height);
   glPixelStorei(GL_UNPACK_ALIGNMENT, 1);
   glBindTexture(GL_TEXTURE_2D, texture[1]);
   glTexImage2D(GL_TEXTURE_2D, 0, 2, width,
                height, 0, GL_RGBA, GL_UNSIGNED_BYTE,
                ibuffer);
   glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
   glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);

   glEnable(GL_TEXTURE_2D);
   glShadeModel(GL_FLAT);
}

void output(GLfloat x, GLfloat y, char *text)
{
   glPushMatrix();
   glTranslatef(x, y, 0);
   glScalef(1 / 152.38, 1 / 152.38, 1 / 152.38);
   for (char *p = text; *p; p++)
   {
       glutStrokeCharacter(GLUT_STROKE_ROMAN, *p);
   }
   glPopMatrix();
}

GLfloat rec1[32][3] = {
   {-0.5, 0.5, 0.5},
   {0.5, 0.5, 0.5},
   {0.5, -0.5, 0.5},
   {-0.5, -0.5, 0.5},
   {-0.5, 0.5, -0.5},
   {0.5, 0.5, -0.5},
```

```
    {0.5, -0.5, -0.5},
    {-0.5, -0.5, -0.5},
};

void keyboard(int button, int x1, int y1)
{
    int rotation = 0;
    switch (button)
    {
    case GLUT_KEY_LEFT:
        x--;
        rotation = 1;
        break;
    case GLUT_KEY_RIGHT:
        x++;
        rotation = 1;
        break;
    case GLUT_KEY_UP:
        y++;
        rotation = 1;
        break;
    case GLUT_KEY_DOWN:
        y--;
        rotation = 1;
        break;
    case GLUT_KEY_F1:
        zoom += 0.1;
        break;
    case GLUT_KEY_F2:
        zoom -= 0.1;
        break;
    case GLUT_KEY_F3:
        intensity += 0.1;
        break;
    case GLUT_KEY_F4:
        intensity -= 0.1;
        break;
    }
    glutPostRedisplay();
    if (rotation == 1)
```

```
    {
        angleX = (x - xOrigin) * 0.0001f;
        angleY = (y - yOrigin) * 0.0001f;
    }
}

void shapeTranslate(GLfloat V[32][3])
{
    for (int i = 0; i < 32; i++)
    {
        for (int j = 0; j < 3; j++)
        {
            V[i][j] += tx[j];
        }
    }
}

void updateLight(void)
{
    glShadeModel(GL_SMOOTH);
    float a = 0.5 + intensity;
    GLfloat light_ambient[] = {a, a, a, 1.0};
    glLightfv(GL_LIGHT0, GL_DIFFUSE, light_ambient);

    glEnable(GL_LIGHTING);
    glEnable(GL_LIGHT0);
    glEnable(GL_DEPTH_TEST);
}

void MyInit()
{

    // shapes and translate
    tx = {0.0, 0.0, 0.0};
    shapeTranslate(rec1);
    glClearColor(0, 0, 0, 1);
    glEnable(GL_DEPTH_TEST);
}

void Spin()
```

```
{
   sleep_for(nanoseconds(1000));
   sleep_until(system_clock::now() + nanoseconds(1000000));
   d = angleX * 180 / 3.14159;
   dy = angleY * 180 / 3.14159;
   // if (d > 360)
   //     d = 0;
   glutPostRedisplay();
}

void FaceTexture(GLfloat A[], GLfloat B[], GLfloat C[], GLfloat D[])
{
   glActiveTexture(GL_TEXTURE0);
   glBindTexture(GL_TEXTURE_2D, texture[1]);
   glBegin(GL_POLYGON);
   glTexCoord2f(0.0, 0.0);
   glVertex3fv(A);
   glTexCoord2f(0.0, 1.0);
   glVertex3fv(B);
   glTexCoord2f(1.0, 1.0);
   glVertex3fv(C);
   glTexCoord2f(1.0, 0.0);
   glVertex3fv(D);
   glEnd();
}

void FaceText(GLfloat A[], GLfloat B[], GLfloat C[], GLfloat D[])
{
   glActiveTexture(GL_TEXTURE0);
   glBindTexture(GL_TEXTURE_2D, texture[0]);
   glBegin(GL_POLYGON);
   glTexCoord2f(0.0, 0.0);
   glVertex3fv(A);
   glTexCoord2f(0.0, 1.0);
   glVertex3fv(B);
   glTexCoord2f(1.0, 1.0);
   glVertex3fv(C);
   glTexCoord2f(1.0, 0.0);
   glVertex3fv(D);
   glEnd();
```

```
}

void Cube(GLfloat V0[], GLfloat V1[], GLfloat V2[], GLfloat V3[], GLfloat
V4[], GLfloat V5[], GLfloat V6[], GLfloat V7[])
{
   glColor3f(1, 0, 0);
   FaceTexture(V0, V1, V2, V3); // Front
   glColor3f(0, 1, 0);
   FaceTexture(V4, V5, V6, V7); // Back
   glColor3f(1, 0, 1);
   FaceTexture(V2, V3, V7, V6); // Bot
   glColor3f(0, 1, 1);
   FaceTexture(V0, V1, V5, V4); // Top
   glColor3f(0, 0, 1);
   FaceText(V0, V4, V7, V3); // Left
   glColor3f(1, 1, 0);
   FaceText(V1, V5, V6, V2); // Right
}

void Rotate(GLfloat V[32][3], int points, GLfloat rV[32][3])
{
   GLfloat r, ry;
   r = d * 3.14 / 180;
   ry = dy * 3.14 / 180;
   if (a == 1)
   {
       for (int i = 0; i < points; i++)
       {
           rV[i][0] = V[i][0];
           rV[i][1] = V[i][1] * cos(ry) - V[i][2] * sin(ry);
           rV[i][2] = V[i][1] * sin(ry) + V[i][2] * cos(ry);

           rV[i][0] = rV[i][2] * sin(r) + rV[i][0] * cos(r);
           rV[i][1] = rV[i][1];
           rV[i][2] = rV[i][2] * cos(r) - rV[i][0] * sin(r);
       }
   }
}

void copyMatrix(GLfloat V[32][3], GLfloat rV[32][3])
```

```
{
   for (int i = 0; i < 32; i++)
   {
       for (int j = 0; j < 3; j++)
       {
           V[i][j] = rV[i][j];
       }
   }
}

void Draw()
{
   glLoadIdentity();
   GLfloat V[32][3];
   copyMatrix(V, rec1);
   Rotate(V, 8, rec1);
   glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
   glMatrixMode(GL_PROJECTION);
   glLoadIdentity();
   gluPerspective(90, 1, 1, 3);
   // Do this every time the user wants to zoom
   glOrtho(-zoom, zoom, -zoom, zoom, 1, 10);
   glMatrixMode(GL_MODELVIEW);
   // update brightness
   updateLight();
   // glTranslatef(0, 0, zoom);

   Cube(rec1[0], rec1[1], rec1[2], rec1[3], rec1[4], rec1[5], rec1[6],
rec1[7]);

   glutSwapBuffers();
}

void mouseMove(int x, int y)
{

   // this will only be true when the left button is down
   if (xOrigin >= 0)
   {
```

```
        // update angleX
        angleX = (x - xOrigin) * 0.0001f;
        angleY = (y - yOrigin) * 0.0001f;
    }
}

void mouseButton(int button, int state, int x, int y)
{

    // only start motion if the left button is pressed
    if (button == GLUT_LEFT_BUTTON)
    {

        // when the button is released
        if (state == GLUT_UP)
        {
            xOrigin = -1;
            yOrigin = -1;
        }
        else
        { // state = GLUT_DOWN
            xOrigin = x;
            yOrigin = y;
        }
    }
    angleX = 0;
    angleY = 0;
}

int main(int argc, char *argv[])
{
    a = 1;
    glutInit(&argc, argv);
    glutInitWindowSize(600, 600);
    glutInitWindowPosition(50, 150);
    glutInitDisplayMode(GLUT_RGB | GLUT_DOUBLE | GLUT_DEPTH);
    glutCreateWindow("Cube Spin with Matrices");
    glOrtho(-1, 1, -1.0, 1, 1, 3);
    MyInit();
    init_texture();
```

```
    updateLight();

    glutDisplayFunc(Draw);
    glutIdleFunc(Spin);

    // here are the two new functions
    glutSpecialFunc(keyboard);
    glutMouseFunc(mouseButton);
    glutMotionFunc(mouseMove);

    glutMainLoop();
    return 0;
}
```

Compile

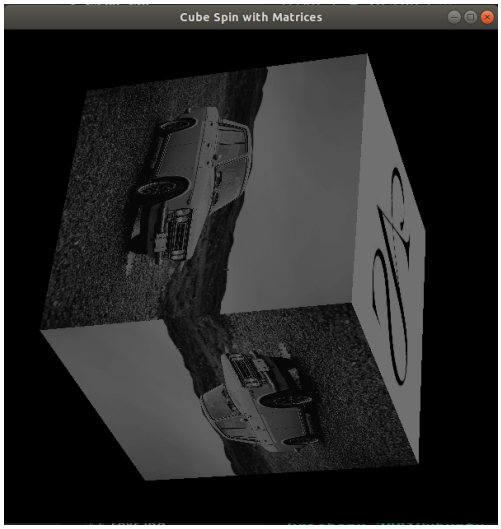
g++ Q2.cpp -o Q2 -lglut -lGLU -lGL imageio.o -ltiff -lpng

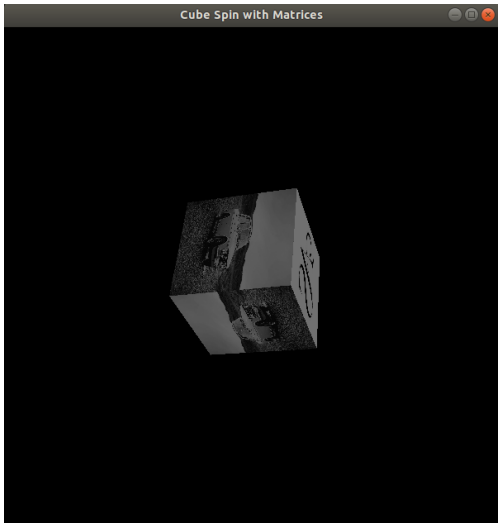
Output

./Q2


Cube Spin with Matrices

Zoom in





Brightness

Image and text