CS352 Assignment-6

Krishanu Saini
190001029

Write a program in OpenGL for the following:

1) Create a 3-D house.
2) Create a provision to view the house using mouse.

Program

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

Code

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

GLfloat d = 0, dy = 0;
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);

GLfloat rec1[32][3] = {
   {-0.5, 0.3, 0.2},
   {0.5, 0.3, 0.2},
   {0.5, -0.3, 0.2},
   {-0.5, -0.3, 0.2},
   {-0.5, 0.3, -0.2},
   {0.5, 0.3, -0.2},
   {0.5, -0.3, -0.2},
   {-0.5, -0.3, -0.2},
};

GLfloat tri1[32][3] = {
   {0, 0.3, 0.2},
```

```
   {0, 0.3, 0.2},
   {0.55, -0.3, 0.2},
   {-0.55, -0.3, 0.2},
   {-0, 0.3, -0.2},
   {0, 0.3, -0.2},
   {0.55, -0.3, -0.2},
   {-0.55, -0.3, -0.2},
};

GLfloat win1[32][3] = {
   {-0.1, 0.1, -0.2},
   {0.1, 0.1, -0.2},
   {0.1, -0.1, -0.2},
   {-0.1, -0.1, -0.2},
};

GLfloat win2[32][3] = {
   {-0.1, 0.1, -0.2},
   {0.1, 0.1, -0.2},
   {0.1, -0.1, -0.2},
   {-0.1, -0.1, -0.2},
};

GLfloat door[32][3] = {
   {-0.1, 0.15, -0.2},
   {0.1, 0.15, -0.2},
   {0.1, -0.15, -0.2},
   {-0.1, -0.15, -0.2},
};

GLfloat handle[32][3] = {
   {-0.01, 0.01, -0.2},
   {0.01, 0.01, -0.2},
   {0.01, -0.01, -0.2},
   {-0.01, -0.01, -0.2},
};

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 MyInit()
{

    // shapes and translate
    tx = {0.0, 0.0, 0.0};
    shapeTranslate(rec1);
    tx = {0.0, 0.6, 0.0};
    shapeTranslate(tri1);
    tx = {-0.3, 0, -0.002};
    shapeTranslate(win1);
    tx = {0.3, 0, -0.002};
    shapeTranslate(win2);
    tx = {0, -0.1, -0.002};
    shapeTranslate(door);
    tx = {-0.05, -0.10, -0.003};
    shapeTranslate(handle);
    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 Face(GLfloat A[], GLfloat B[], GLfloat C[], GLfloat D[])
```

```
{
   glBegin(GL_POLYGON);
   glVertex3fv(A);
   glVertex3fv(B);
   glVertex3fv(C);
   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);
   Face(V0, V1, V2, V3); // Front
   glColor3f(0, 1, 0);
   Face(V4, V5, V6, V7); // Back
   glColor3f(0, 0, 1);
   Face(V0, V4, V7, V3); // Left
   glColor3f(1, 1, 0);
   Face(V1, V5, V6, V2); // Right
   glColor3f(1, 0, 1);
   Face(V2, V3, V7, V6); // Bot
   glColor3f(0, 1, 1);
   Face(V0, V1, V5, V4); // Top
}

void Triangle(GLfloat V0[], GLfloat V1[], GLfloat V2[], GLfloat V3[],
GLfloat V4[], GLfloat V5[], GLfloat V6[], GLfloat V7[])
{
   glColor3f(0.5, 0, 0);
   Face(V0, V1, V2, V3); // Front
   glColor3f(0, 0.5, 0);
   Face(V4, V5, V6, V7); // Back
   glColor3f(0, 0, 0.5);
   Face(V0, V4, V7, V3); // Left
   glColor3f(0.5, 0.5, 0);
   Face(V1, V5, V6, V2); // Right
   glColor3f(0.5, 0, 0.5);
   Face(V2, V3, V7, V6); // Bot
   glColor3f(0.5, 1, 1);
```

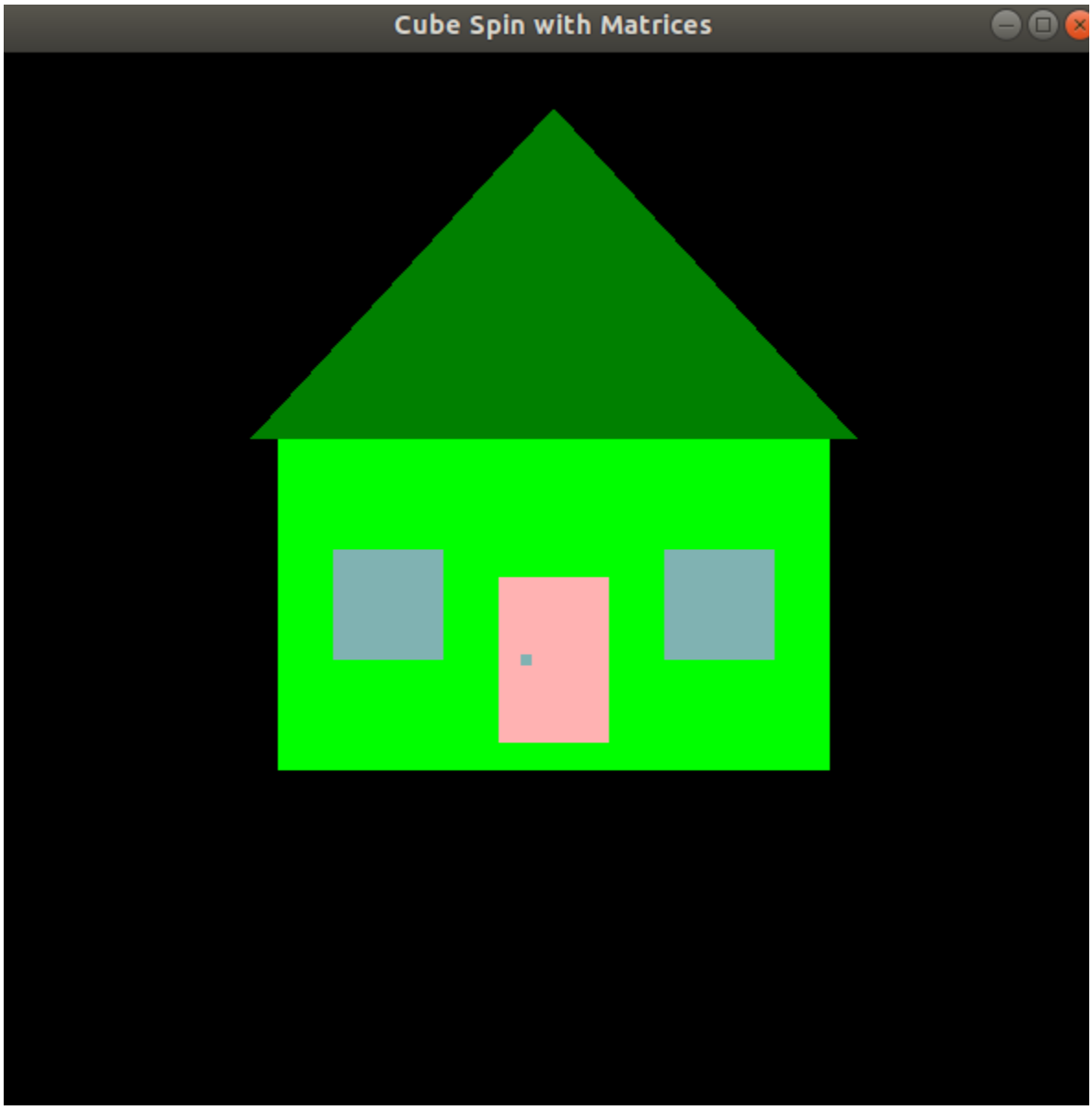
```
   Face(V0, V1, V5, V4); // Top
}

void Windows(GLfloat V0[], GLfloat V1[], GLfloat V2[], GLfloat V3[])
{
   glColor3f(0.5, 0.7, 0.7);
   Face(V0, V1, V2, V3); // Front
}

void Door(GLfloat V0[], GLfloat V1[], GLfloat V2[], GLfloat V3[])
{
   glColor3f(1, 0.7, 0.7);
   Face(V0, V1, V2, V3); // Front
}

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;
   cout << d << " " << dy << endl;
   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++)
   {
```

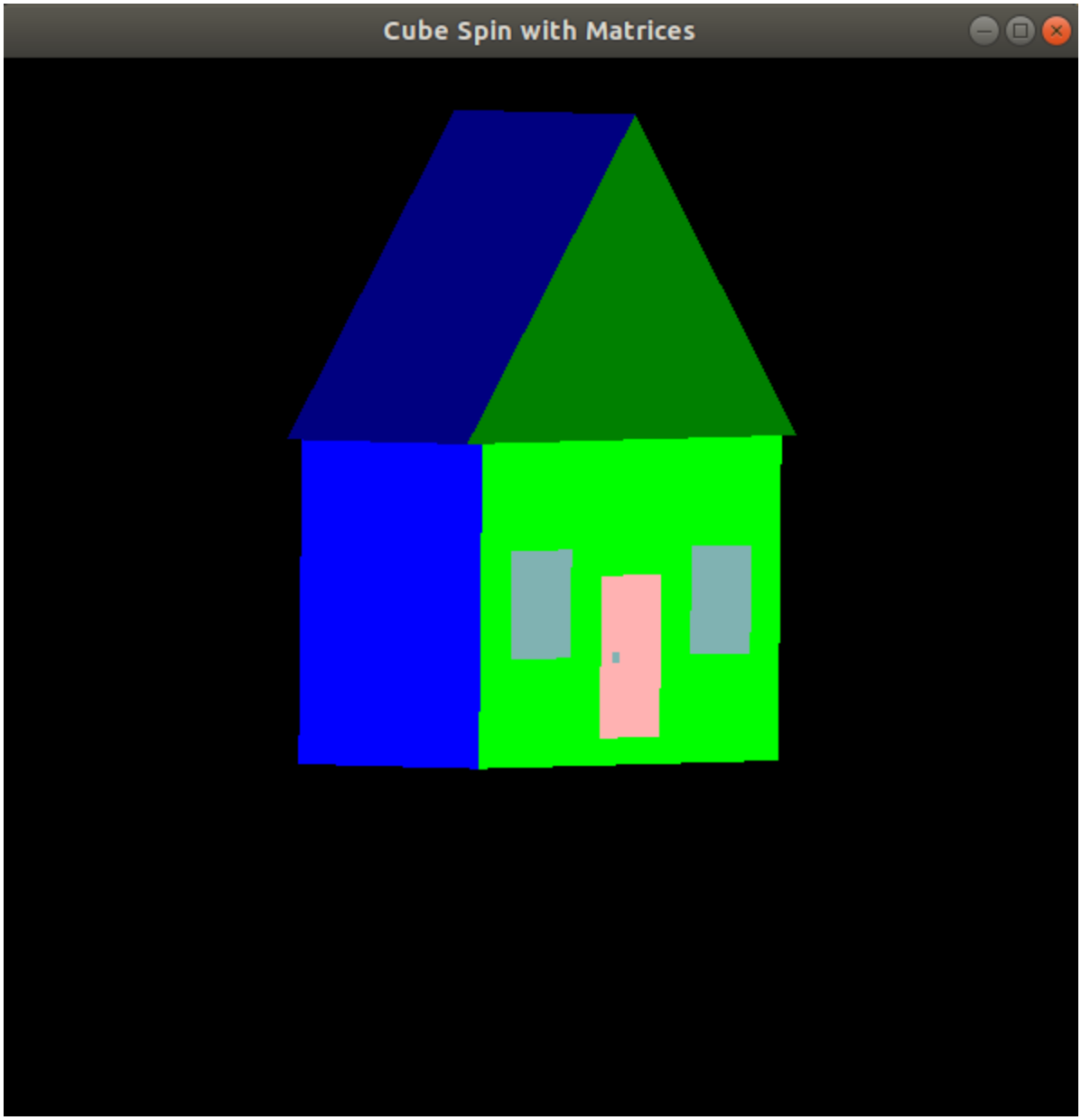
```
        for (int j = 0; j < 3; j++)
        {
            V[i][j] = rV[i][j];
        }
    }
}

void Draw()
{
    GLfloat V[32][3];
    copyMatrix(V, rec1);
    Rotate(V, 8, rec1);
    copyMatrix(V, tri1);
    Rotate(V, 8, tri1);
    copyMatrix(V, win1);
    Rotate(V, 4, win1);
    copyMatrix(V, win2);
    Rotate(V, 4, win2);
    copyMatrix(V, door);
    Rotate(V, 4, door);
    copyMatrix(V, handle);
    Rotate(V, 4, handle);
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

    Cube(rec1[0], rec1[1], rec1[2], rec1[3], rec1[4], rec1[5], rec1[6],
rec1[7]);
    Triangle(tri1[0], tri1[1], tri1[2], tri1[3], tri1[4], tri1[5], tri1[6],
tri1[7]);
    Windows(win1[0], win1[1], win1[2], win1[3]);
    Windows(win2[0], win2[1], win2[2], win2[3]);
    Door(door[0], door[1], door[2], door[3]);
    Windows(handle[0], handle[1], handle[2], handle[3]);

    glutSwapBuffers();
}

void mouseMove(int x, int y)
{

    // this will only be true when the left button is down
```

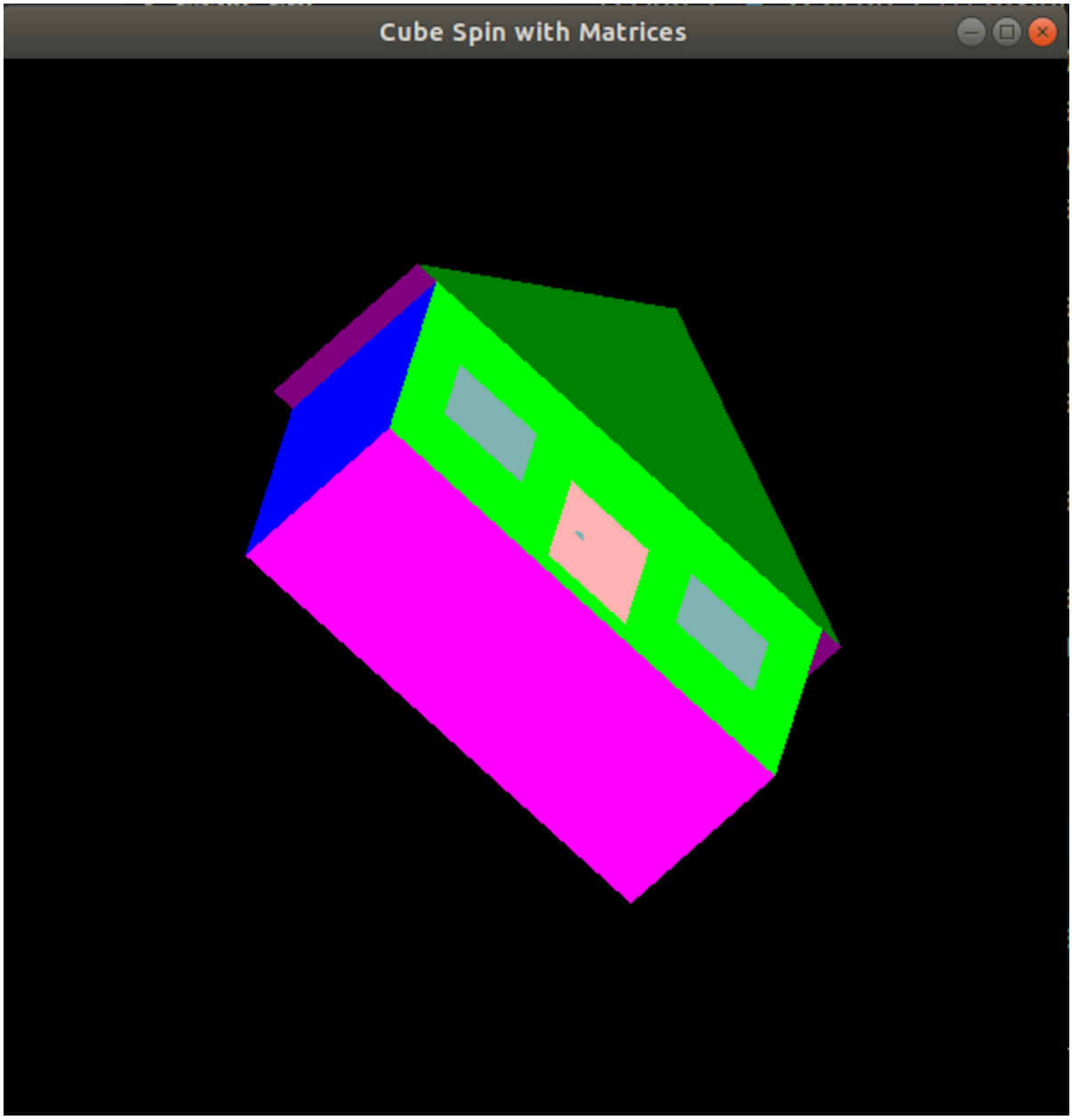
```
    if (xOrigin >= 0)
    {

        // 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");
```

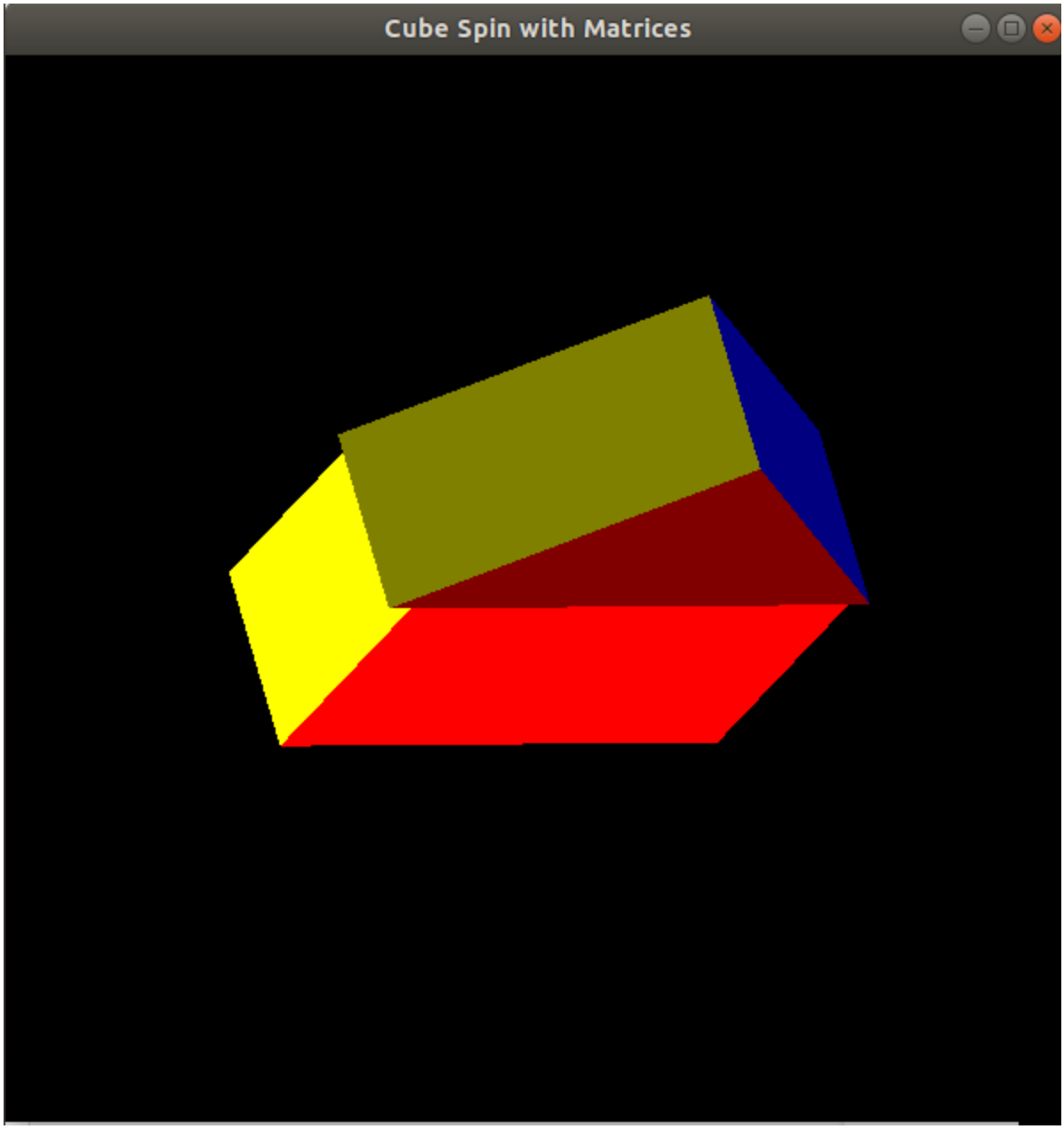
```
   MyInit();
   glutDisplayFunc(Draw);
   glutIdleFunc(Spin);

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

   glutMainLoop();
   return 0;
}
```

Output




Cube Spin with Matrices


Cube Spin with Matrices


Cube Spin with Matrices