Lab Taks-5

Submission Guidelines-

- Rename the file to your id only. If your id is 18-XXXXX-1, then the file name must be 18-XXXXX-1.docx.
- Must submit within the announced time.
- Must include resources for all the section in the table

Question-1

Create an animation using two box that will move in the opposite direction.

Graph Plot (Picture)-

[Not needed]

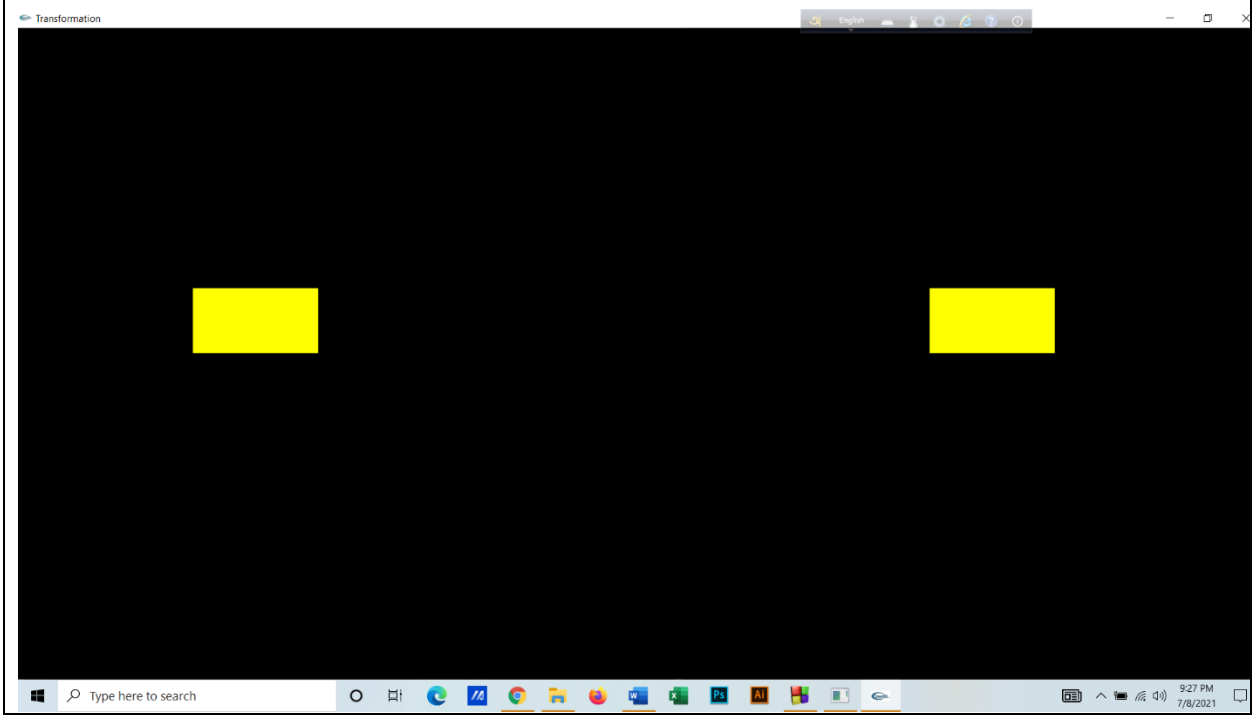
Code-

```
#include <iostream>
#include<GL/gl.h>
#include <GL/glut.h>
using namespace std;

float _move = 0.0f;
float _move1 = -.03f;
void drawScene() {

glClear(GL_COLOR_BUFFER_BIT);
glColor3d(1,1,0);
glLoadIdentity(); //Reset the drawing perspective
glMatrixMode(GL_MODELVIEW);
 glPushMatrix();
glTranslatef(_move, 0.0f, 0.0f);
glBegin(GL_QUADS);
glVertex2f(0.1f, 0.0f);
glVertex2f(0.3f, 0.0f);
glVertex2f(0.3f, 0.2f);
glVertex2f(0.1f, 0.2);
glEnd();


glPushMatrix();
glTranslatef(_move1, 0.0f, 0.0f);
```

```
glBegin(GL_QUADS);
glVertex2f(-0.1f, 0.0f);
glVertex2f(-0.3f, 0.0f);
glVertex2f(-0.3f, 0.2f);
glVertex2f(-0.1f, 0.2f);
glEnd();
glPopMatrix();
glutSwapBuffers();
}

void update(int value) {


 _move += .03;
if(_move > 1.3)
{
_move =0.0;
}


glutPostRedisplay();
glutTimerFunc(20, update, 0);

}

void updatee(int value) {


 _move1 += -.05;
if(_move1 < -1.3)
{
_move1 =-0.03;
}

glutPostRedisplay();
glutTimerFunc(18, updatee, 0);

}

int main(int argc, char** argv) {
glutInit(&argc, argv);
glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);
glutInitWindowSize(800, 800);
glutCreateWindow("Transformation");
```

```
glutDisplayFunc(drawScene);
gluOrtho2D(-2,2,-2,2);
glutTimerFunc(20, update, 0); //Add a timer
glutTimerFunc(20, updatee, 0); //Add a timer

glutMainLoop();
return 0;
}
```

Output Screenshot (Full Screen)-



Question-2

Design a car which will have rotating wheels.

Graph Plot (Picture)-

[Not needed]

Code-

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

```
#include <math.h>

using namespace std;



float _move = 0.0f;
float _move1 = 0.23f;
float _move2 = 0.34f;
float _angle1 = 0.0f;
float _angle2 = 0.0f;
float _angle3 = 0.0f;
float _angle4 = 0.0f;
float _angle5 = 0.0f;
float _angle6 = 0.0f;

void drawScene() {
glClear(GL_COLOR_BUFFER_BIT);
glColor3d(1,1,0);
glLoadIdentity(); //Reset the drawing perspective
glMatrixMode(GL_MODELVIEW);



 glPushMatrix();
glTranslatef(_move, 0.0f, 0.0f);
glBegin(GL_QUADS);
glVertex2f(0.1f, 0.0f);
glVertex2f(0.7f, 0.0f);
glVertex2f(0.7f, 0.2f);
glVertex2f(0.1f, 0.2);
glColor3d(0,1,0);
glVertex2f(0.29f, 0.37f);
glVertex2f(0.18f, 0.2f);
glVertex2f(0.6f, 0.2f);
glVertex2f(0.5f, 0.37);
glEnd();

          glLineWidth(7.5);
          glTranslatef(_move1, 0.0f, 0.0f);
          glBegin(GL_LINES);// Draw a Red 1x1 Square centered at origin
          for(int i=0;i<700;i++)
      {
        glColor3ub(255, 170, 0 );
```

```
        float pi=3.1416;
        float A=(i*2*pi)/700;
        float r=0.1;
        float x = r * cos(A);
        float y = r * sin(A);
        glVertex2f(x,y );
     }

  //glVertex2f(0.3f,0.4f);
  //glVertex2f(0.1f,0.4f);

         glEnd();



glLineWidth(7.5);
         glBegin(GL_LINES);// Draw a Red 1x1 Square centered at origin
         for(int i=0;i<700;i++)
     {
        glColor3f(1,1,1);
        float pi=3.1416;
        float A=(i*2*pi)/700;
        float r=0.02;
        float x = r * cos(A);
        float y = r * sin(A);
        glVertex2f(x,y );
     }

  //glVertex2f(0.3f,0.4f);
  //glVertex2f(0.1f,0.4f);

         glEnd();

         glLineWidth(7.5);
         glTranslatef(_move2, 0.0f, 0.0f);
         glBegin(GL_LINES);// Draw a Red 1x1 Square centered at origin
         for(int i=0;i<700;i++)
     {
        glColor3ub(255, 170, 0 );
        float pi=3.1416;
        float A=(i*2*pi)/700;
        float r=0.1;
        float x = r * cos(A);
        float y = r * sin(A);
```

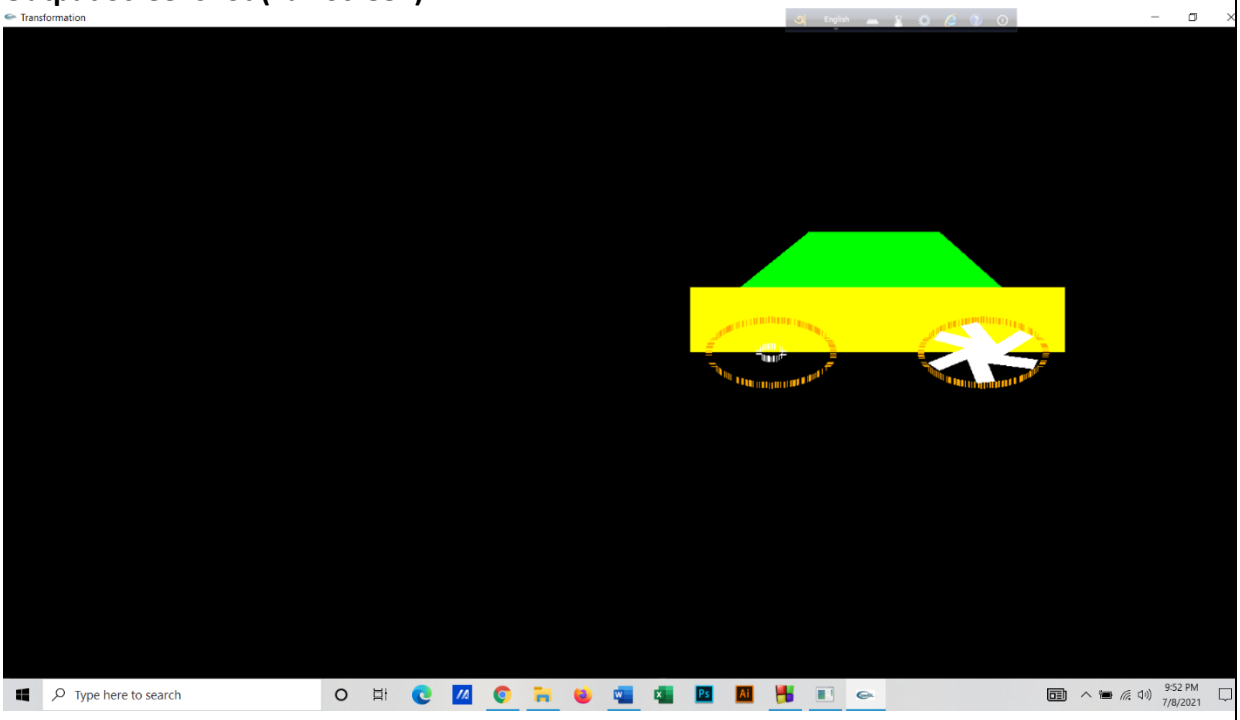
```
        glVertex2f(x,y );
      }

   //glVertex2f(0.3f,0.4f);
   //glVertex2f(0.1f,0.4f);

         glEnd();

glLineWidth(7.5);
         glBegin(GL_LINES);// Draw a Red 1x1 Square centered at origin
         for(int i=0;i<700;i++)
      {
        glColor3f(1,1,1);
        float pi=3.1416;
        float A=(i*2*pi)/700;
        float r=0.02;
        float x = r * cos(A);
        float y = r * sin(A);
        glVertex2f(x,y );
      }

   //glVertex2f(0.3f,0.4f);
   //glVertex2f(0.1f,0.4f);

         glEnd();

  glRotatef(_angle1, 0.0f, 0.0f,1.0f);
   glBegin(GL_QUADS);
      glVertex2f(0.0f, 0.0f);
      glVertex2f(0.03f, 0.0f);
      glVertex2f(0.03f, 0.09f);
      glVertex2f(0.0f, 0.09f);
         glEnd();

         glRotatef(_angle2, 0.0f, 0.0f,1.0f);
   glBegin(GL_QUADS);
      glVertex2f(0.0f, 0.0f);
      glVertex2f(0.03f, 0.0f);
      glVertex2f(0.03f, 0.09f);
      glVertex2f(0.0f, 0.09f);
         glEnd();

         glRotatef(_angle3, 0.0f, 0.0f,1.0f);
   glBegin(GL_QUADS);
```

```
      glVertex2f(0.0f, 0.0f);
      glVertex2f(0.03f, 0.0f);
      glVertex2f(0.03f, 0.09f);
      glVertex2f(0.0f, 0.09f);
          glEnd();




          glRotatef(_angle4, 0.0f, 0.0f,1.0f);
   glBegin(GL_QUADS);
      glVertex2f(0.0f, 0.0f);
      glVertex2f(0.03f, 0.0f);
      glVertex2f(0.03f, 0.09f);
      glVertex2f(0.0f, 0.09f);
          glEnd();

          glRotatef(_angle5, 0.0f, 0.0f,1.0f);
   glBegin(GL_QUADS);
      glVertex2f(0.0f, 0.0f);
      glVertex2f(0.03f, 0.0f);
      glVertex2f(0.03f, 0.09f);
      glVertex2f(0.0f, 0.09f);
          glEnd();

          glRotatef(_angle6, 0.0f, 0.0f,1.0f);
   glBegin(GL_QUADS);
      glVertex2f(0.0f, 0.0f);
      glVertex2f(0.03f, 0.0f);
      glVertex2f(0.03f, 0.09f);
      glVertex2f(0.0f, 0.09f);
          glEnd();



glPopMatrix();
glutSwapBuffers();



}
```

```
void update(int value) {


    _angle1+=1.0f;
   if(_angle1 > 360.0)
   {
      _angle1-=360;
   }

   _angle2+=1.0f;
   if(_angle2 > 360.0)
   {
      _angle2-=360;
   }

   _angle3+=1.0f;
   if(_angle3 > 360.0)
   {
      _angle3-=360;
   }

  _angle4+=1.0f;
   if(_angle4 > 360.0)
   {
      _angle4-=360;
   }

   _angle5+=1.0f;
   if(_angle2 >360.0)
   {
      _angle5-=360;
   }

   _angle6+=1.0f;
   if(_angle6 > 360.0)
   {
      _angle6-=360;
   }


glutPostRedisplay();
glutTimerFunc(20, update, 0);
}
```

```
int main(int argc, char** argv) {
glutInit(&argc, argv);
glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);

glutInitWindowSize(800, 800);
glutCreateWindow("Transformation");
glutDisplayFunc(drawScene);
gluOrtho2D(-2,2,-2,2);
glutTimerFunc(20, update, 0); //Add a timer
glutMainLoop();
return 0;
}
```

Output Screenshot (Full Screen)-



Question-3

Now move your car of question-2 from left to right in a loop.

Graph Plot (Picture)-

[Not needed]

Code-

```
#include <iostream>
#include<GL/gl.h>
#include <GL/glut.h>
#include <math.h>

using namespace std;



float _move = 0.0f;
float _move1 = 0.23f;
float _move2 = 0.34f;
float _angle1 = 0.0f;
float _angle2 = 0.0f;
float _angle3 = 0.0f;
float _angle4 = 0.0f;
float _angle5 = 0.0f;
float _angle6 = 0.0f;

void drawScene() {
glClear(GL_COLOR_BUFFER_BIT);
glColor3d(1,1,0);
glLoadIdentity(); //Reset the drawing perspective
glMatrixMode(GL_MODELVIEW);



 glPushMatrix();
glTranslatef(_move, 0.0f, 0.0f);
glBegin(GL_QUADS);
glVertex2f(0.1f, 0.0f);
glVertex2f(0.7f, 0.0f);
glVertex2f(0.7f, 0.2f);
glVertex2f(0.1f, 0.2);
glColor3d(0,1,0);
glVertex2f(0.29f, 0.37f);
glVertex2f(0.18f, 0.2f);
glVertex2f(0.6f, 0.2f);
```

```
glVertex2f(0.5f, 0.37);
glEnd();

          glLineWidth(7.5);
          glTranslatef(_move1, 0.0f, 0.0f);
          glBegin(GL_LINES);// Draw a Red 1x1 Square centered at origin
          for(int i=0;i<700;i++)
      {
         glColor3ub(255, 170, 0 );
         float pi=3.1416;
         float A=(i*2*pi)/700;
         float r=0.1;
         float x = r * cos(A);
         float y = r * sin(A);
         glVertex2f(x,y );
      }

   //glVertex2f(0.3f,0.4f);
   //glVertex2f(0.1f,0.4f);

          glEnd();



glLineWidth(7.5);
          glBegin(GL_LINES);// Draw a Red 1x1 Square centered at origin
          for(int i=0;i<700;i++)
      {
          glColor3f(1,1,1);
         float pi=3.1416;
         float A=(i*2*pi)/700;
         float r=0.02;
         float x = r * cos(A);
         float y = r * sin(A);
         glVertex2f(x,y );
      }

   //glVertex2f(0.3f,0.4f);
   //glVertex2f(0.1f,0.4f);

          glEnd();

          glLineWidth(7.5);
          glTranslatef(_move2, 0.0f, 0.0f);
```

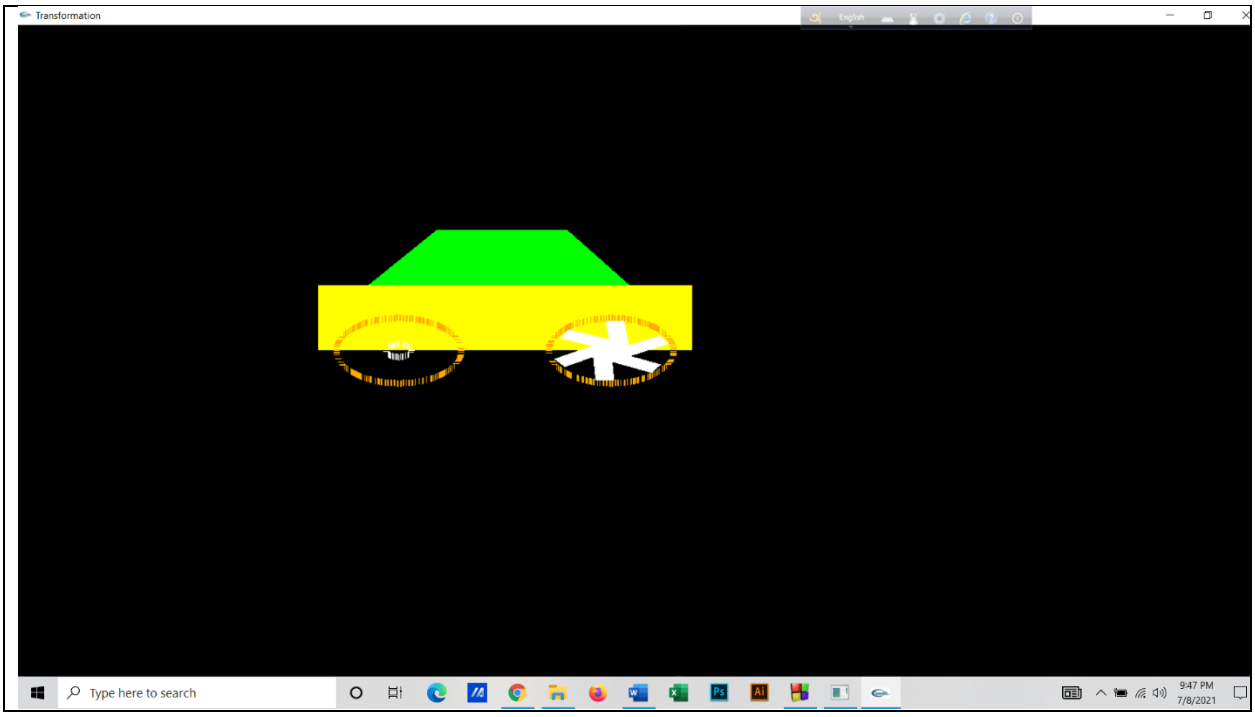
```
          glBegin(GL_LINES);// Draw a Red 1x1 Square centered at origin
          for(int i=0;i<700;i++)
      {
          glColor3ub(255, 170, 0 );
         float pi=3.1416;
         float A=(i*2*pi)/700;
         float r=0.1;
         float x = r * cos(A);
         float y = r * sin(A);
         glVertex2f(x,y );
      }

   //glVertex2f(0.3f,0.4f);
   //glVertex2f(0.1f,0.4f);

          glEnd();

glLineWidth(7.5);
          glBegin(GL_LINES);// Draw a Red 1x1 Square centered at origin
          for(int i=0;i<700;i++)
      {
         glColor3f(1,1,1);
         float pi=3.1416;
         float A=(i*2*pi)/700;
         float r=0.02;
         float x = r * cos(A);
         float y = r * sin(A);
         glVertex2f(x,y );
      }

   //glVertex2f(0.3f,0.4f);
   //glVertex2f(0.1f,0.4f);

          glEnd();

   glRotatef(_angle1, 0.0f, 0.0f,1.0f);
    glBegin(GL_QUADS);
      glVertex2f(0.0f, 0.0f);
      glVertex2f(0.03f, 0.0f);
      glVertex2f(0.03f, 0.09f);
      glVertex2f(0.0f, 0.09f);
          glEnd();

          glRotatef(_angle2, 0.0f, 0.0f,1.0f);
```

```
glBegin(GL_QUADS);
   glVertex2f(0.0f, 0.0f);
   glVertex2f(0.03f, 0.0f);
   glVertex2f(0.03f, 0.09f);
   glVertex2f(0.0f, 0.09f);
       glEnd();

       glRotatef(_angle3, 0.0f, 0.0f,1.0f);
glBegin(GL_QUADS);
   glVertex2f(0.0f, 0.0f);
   glVertex2f(0.03f, 0.0f);
   glVertex2f(0.03f, 0.09f);
   glVertex2f(0.0f, 0.09f);
       glEnd();




       glRotatef(_angle4, 0.0f, 0.0f,1.0f);
glBegin(GL_QUADS);
   glVertex2f(0.0f, 0.0f);
   glVertex2f(0.03f, 0.0f);
   glVertex2f(0.03f, 0.09f);
   glVertex2f(0.0f, 0.09f);
       glEnd();

       glRotatef(_angle5, 0.0f, 0.0f,1.0f);
glBegin(GL_QUADS);
   glVertex2f(0.0f, 0.0f);
   glVertex2f(0.03f, 0.0f);
   glVertex2f(0.03f, 0.09f);
   glVertex2f(0.0f, 0.09f);
       glEnd();

       glRotatef(_angle6, 0.0f, 0.0f,1.0f);
glBegin(GL_QUADS);
   glVertex2f(0.0f, 0.0f);
   glVertex2f(0.03f, 0.0f);
   glVertex2f(0.03f, 0.09f);
   glVertex2f(0.0f, 0.09f);
       glEnd();
```

```
glPopMatrix();
glutSwapBuffers();



}


void update(int value) {


 _move += .02;
if(_move > 1.3)
{
_move =-1.0;
}



    _angle1+=1.0f;
   if(_angle1 > 360.0)
   {
      _angle1-=360;
   }

   _angle2+=1.0f;
   if(_angle2 > 360.0)
   {
      _angle2-=360;
   }

   _angle3+=1.0f;
   if(_angle3 > 360.0)
   {
      _angle3-=360;
   }

  _angle4+=1.0f;
   if(_angle4 > 360.0)
   {
      _angle4-=360;
   }
```

```
   _angle5+=1.0f;
   if(_angle2 >360.0)
   {
      _angle5-=360;
   }

   _angle6+=1.0f;
   if(_angle6 > 360.0)
   {
      _angle6-=360;
   }


glutPostRedisplay();
glutTimerFunc(20, update, 0);
}


int main(int argc, char** argv) {
glutInit(&argc, argv);
glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);

glutInitWindowSize(800, 800);
glutCreateWindow("Transformation");
glutDisplayFunc(drawScene);
gluOrtho2D(-2,2,-2,2);
glutTimerFunc(20, update, 0); //Add a timer
glutMainLoop();
return 0;
}
```

Output Screenshot (Full Screen)-



Question-4

Design a windmill with rotating blades

Graph Plot (Picture)-

[Not needed]

Code-

```
#include <iostream>
#include<GL/gl.h>
#include <GL/glut.h>
#include <math.h>

using namespace std;


float _angle1 = 0.0f;
float _angle2 = 0.0f;
float _angle3 = 0.0f;
void drawScene() {
glClear(GL_COLOR_BUFFER_BIT);
glLoadIdentity(); //Reset the drawing perspective
glMatrixMode(GL_MODELVIEW);
```

```
 glPushMatrix();
 glColor3ub(171, 102, 52);
 glBegin(GL_QUADS);
      glVertex2f(0.02f, 0.0f);
      glVertex2f(-0.02f, 0.0f);
      glVertex2f(-0.03f,- 0.9f);
      glVertex2f(0.03f, -0.9);
          glEnd();
glRotatef(_angle1, 0.0f, 0.0f,1.0f);
glColor3d(1,1,0);
   glBegin(GL_QUADS);
      glVertex2f(0.0f, 0.0f);
      glVertex2f(0.5f, 0.0f);
      glVertex2f(0.5f, 0.1f);
      glVertex2f(0.1f, 0.1);
          glEnd();
glRotatef(_angle2, 0.0f, 0.0f,1.0f);

glBegin(GL_QUADS);
      glVertex2f(0.0f, 0.0f);
      glVertex2f(0.5f, 0.0f);
      glVertex2f(0.5f, 0.1f);
      glVertex2f(0.1f, 0.1);
          glEnd();

          glRotatef(_angle3, 0.0f, 0.0f,1.0f);
   glBegin(GL_QUADS);
      glVertex2f(0.0f, 0.0f);
      glVertex2f(0.5f, 0.0f);
      glVertex2f(0.5f, 0.1f);
      glVertex2f(0.1f, 0.1);
          glEnd();

          glLineWidth(7.5);
          glBegin(GL_POLYGON);// Draw a Red 1x1 Square centered at origin
          for(int i=0;i<700;i++)
      {
         glColor3f(1,1,1);
         float pi=3.1416;
         float A=(i*2*pi)/700;
         float r=0.1;
         float x = r * cos(A);
         float y = r * sin(A);
```

```
        glVertex2f(x,y );
      }


        glEnd();


glPopMatrix();
glutSwapBuffers();


}


void update(int value) {

_angle1+=1.0f;
   if(_angle1 > 360.0)
   {
      _angle1-=360;
   }

   _angle2+=1.0f;
   if(_angle2 > 360.0)
   {
      _angle2-=360;
   }

   _angle3+=1.0f;
   if(_angle3 > 360.0)
   {
      _angle3-=360;
   }


glutPostRedisplay();
glutTimerFunc(20, update, 0);
}

int main(int argc, char** argv) {
glutInit(&argc, argv);
glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);
glutInitWindowSize(800, 800);
glutCreateWindow("Transformation");
```

```
glutDisplayFunc(drawScene);
gluOrtho2D(-2,2,-2,2);
glutTimerFunc(20, update, 0); //Add a timer
glutMainLoop();
return 0;
}
```

Output Screenshot (Full Screen)-