Practical No.06

Title:- 6.Implement following 2D transformations on the object with respect to axis : – CO5 i) Scaling ii) Rotation about arbitrary point iii) Translation

Name:-Ujawala Sinha
Roll no.S554

CODE:-

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

double parr[8];

void init()
{
   glClear(GL_COLOR_BUFFER_BIT);
   glClearColor(0,0,0,1);
   glColor3f(1,0,1);
   gluOrtho2D(-500,500,-500,500); // Left,right,bottom,top

   // Polygon Defaut
   parr[0] = 60;  //x
   parr[1] = 60;  //y

   parr[2] = 250;
   parr[3] = 60;

   parr[4] = 250;
   parr[5] = 250;

   parr[6] = 60;
   parr[7] = 250;
}

double degreeToRad(double deg)
{
   return 3.14*(deg/180);
}

void polygon()
{
   glColor3f(1,0,0);
   glBegin(GL_LINE_LOOP);
      glVertex2f(parr[0],parr[1]);
      glVertex2f(parr[2],parr[3]);

      glVertex2f(parr[4],parr[5]);
      glVertex2f(parr[6],parr[7]);
   glEnd();
```

```
    glFlush();
}


void drawCorodinates()
{
    glClear(GL_COLOR_BUFFER_BIT);

    glColor3f(1,1,1);
    glPointSize(4);

    glBegin(GL_LINES);
        glVertex2f(-500,0);
        glVertex2f(500,0);

        glVertex2f(0,500);
        glVertex2f(0,-500);
    glEnd();

    glColor3f(1,0,0);

    glBegin(GL_POINTS);
        glVertex2f(0,0);
    glEnd();

    glFlush();
}


// Translation
void translate()
{
   // 40px in x
   // 50px in y

    int i = 0;
    int x = 50, y = 30;

    for(i= 0;i<8;i=i+2)
    {
        parr[i] = parr[i] + x;
    }

    for(i = 1;i<8;i=i+2)
    {
        parr[i] = parr[i] + y;
    }

    polygon();
}
```

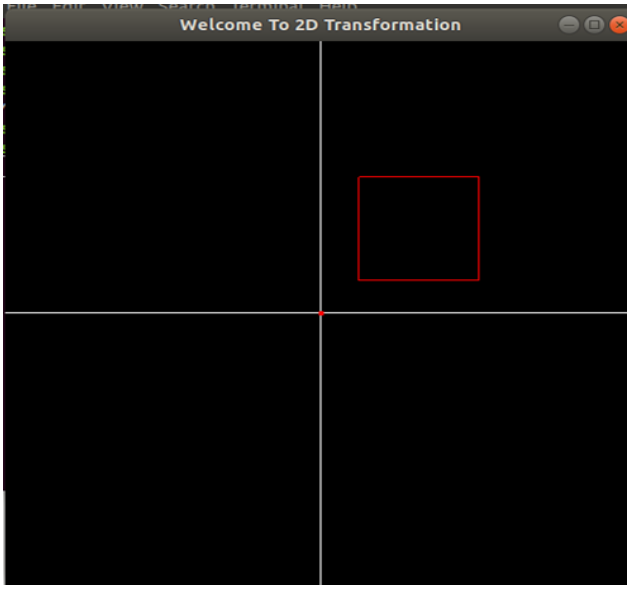
```
// Rotation

 void rotation()
{
   double angle = 45; // You can try 90, 120, etc.
   double rad = degreeToRad(angle);

   // Arbitrary fixed point (example: bottom-left corner of the polygon)
   double xr = parr[0]; // x of 1st vertex
   double yr = parr[1]; // y of 1st vertex

   for (int i = 0; i < 8; i += 2)
   {
      // Step 1: Translate point so that (xr, yr) becomes origin
      double x = parr[i] - xr;
      double y = parr[i + 1] - yr;

      // Step 2: Apply rotation
      double xNew = x * cos(rad) - y * sin(rad);
      double yNew = x * sin(rad) + y * cos(rad);

      // Step 3: Translate point back to original position
      parr[i] = xNew + xr;
      parr[i + 1] = yNew + yr;
   }

   polygon();
}

// Scaling
void scaling()
{
  // 2 unit in x
  // 2 unit in y

   int i = 0;
   double x = 2, y =2;

   for(i= 0;i<7;i=i+2)
   {
      parr[i] = parr[i] * x;
      parr[i+1] = parr[i+1] * y;
   }


   polygon();
}


void menu(int ch)
{
      drawCorodinates();
```

```
   switch(ch)
   {
      case 1: polygon();
         break;

      case 2: translate();
         break;

      case 3: scaling();
         break;

      case 4: rotation();
         break;
   }
}


int main(int argc,char **argv)
{
   glutInit(&argc,argv);
   glutInitWindowSize(500,500);
   glutInitWindowPosition(100,100);

   glutCreateWindow("Welcome To 2D Transformation");
   init();
   glutDisplayFunc(drawCorodinates);

   glutCreateMenu(menu);
      glutAddMenuEntry("1 Display Polygon",1);
      glutAddMenuEntry("2 Translate",2);
      glutAddMenuEntry("3 Scaling",3);
      glutAddMenuEntry("4 Rotate",4);
   glutAttachMenu(GLUT_RIGHT_BUTTON);

   glutMainLoop();
   return 0;
}
```

OUTPUT:-

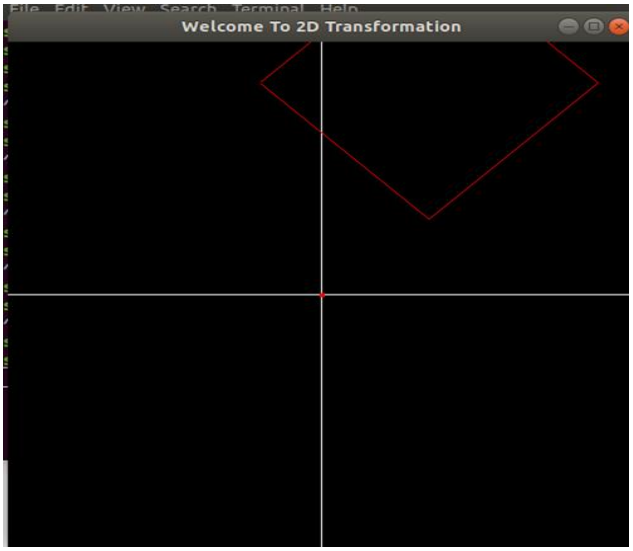
svpm@svpm-HP-EliteDesk-800-G2-SFF:~$ g++ 2DT.cpp -lGL -lGLU -lglut svpm@svpm-HP-EliteDesk-800-G2-SFF:~$ ./a.out

i)DISPLAY



ii)TRANSLATION

iii)SCALING



iv)ROTATION ABOUT ARBITARY POINTS