**Assignment 6**

/\*Problem Statement :

Implement following 2D transformations on the object with respect to axis :

i) Scaling

ii) Rotation about arbitrary point

iii) Reflection

iv) Translation \*/

#include<windows.h>

#include<iostream>

#include<GL/glut.h>

#include<math.h>

#include<bits/stdc++.h>

using namespace std ;

int m[20][3], n = 0 ;

void setpixel(GLint x, GLint y)

{

glColor3f(0.0,0.0,1.0);

glBegin(GL\_POINTS);

glVertex2f(x,y);

glEnd();

glFlush();

}void choice()

{

int i;

glPointSize(2.0);

for(i=-700; i<700; i++)

{

setpixel(0,i);

setpixel(i,0);

}}

void setpcolor(double r1, double b1, double g1 )

{

glColor3f(r1,b1,g1);

}

void conect(int x, int y, int px, int py)

{

glPointSize(2);

glBegin(GL\_LINE\_STRIP);

glVertex2i(x,y);

glVertex2i(px,py);

glEnd();

glFlush();

}

void translation(int tx,int ty)

{

int tm[3][3] = {{1,0,tx},{0,1,ty},{0,0,1} },ne[3]= {} ;

for(int i=0 ; i<n; i++)

{

ne [0] = tm[0][0]\*m[i][0] + tm[0][1]\*m[i][1] + tm[0][2]\*m[i][2] ;

ne [1] = tm[1][0]\*m[i][0] + tm[1][1]\*m[i][1] + tm[1][2]\*m[i][2] ;

ne [2] = tm[2][0]\*m[i][0] + tm[2][1]\*m[i][1] + tm[2][2]\*m[i][2] ;

m[i][0] = ne[0] ;

m[i][1] = ne[1] ;

m[i][2] = ne[2] ;

}

for(int i=0 ; i<n; i++)

{

int ni = (i+1)%n;

setpcolor(1,1,0) ;

conect(m[i][0],m[i][1],m[ni][0],m[ni][1]) ;

}}

void rotation(double rot, int xm, int ym )

{

double pi = 3.14159265 ;

double rad = (pi/180.00) ;

rad \*= rot ;

double rm[3][3] = {{cos(rad),sin(rad),0},{-sin(rad),cos(rad),0},{-xm\*cos(rad)+ym\*sin(rad)+xm,-xm\*sin(rad)-ym\*cos(rad)+ym,1} } ;

int ne[3]= {};

for(int i=0 ; i<n; i++)

{

ne [0] = rm[0][0]\*m[i][0] + rm[0][1]\*m[i][1] + rm[0][2]\*m[i][2] ;

ne [1] = rm[1][0]\*m[i][0] + rm[1][1]\*m[i][1] + rm[1][2]\*m[i][2] ;

ne [2] = rm[2][0]\*m[i][0] + rm[2][1]\*m[i][1] + rm[2][2]\*m[i][2] ;

m[i][0] = ne[0] ;

m[i][1] = ne[1] ;

m[i][2] = ne[2] ;

}

for(int i=0 ; i<n; i++)

{

int ni = (i+1)%n;

setpcolor(1,1,0);

conect(m[i][0],m[i][1],m[ni][0],m[ni][1]) ;

}}

void scale(int sx, int sy )

{

int sm[3][3] = {{sx,0,0},{0,sy,0},{0,0,1} } ;

int ne[3]= {} ;

for(int i=0 ; i<n; i++)

{

ne [0] = sm[0][0]\*m[i][0] + sm[0][1]\*m[i][1] + sm[0][2]\*m[i][2] ;

ne [1] = sm[1][0]\*m[i][0] + sm[1][1]\*m[i][1] + sm[1][2]\*m[i][2] ;

ne [2] = sm[2][0]\*m[i][0] + sm[2][1]\*m[i][1] + sm[2][2]\*m[i][2] ;

m[i][0] = ne[0] ;

m[i][1] = ne[1] ;

m[i][2] = ne[2] ;

}

for(int i=0 ; i<n; i++)

{

int ni = (i+1)%n;

setpcolor(1,1,0) ;

conect(m[i][0],m[i][1],m[ni][0],m[ni][1]) ;

}}

void reflect(char c)

{

int sm[3][3] = {{1,0,0},{0,1,0},{0,0,1} } ;

if(c=='x'||c=='X')

{

sm[1][1]=-1 ;

}

else

{

sm[0][0] = - 1 ;

}

int ne[3]= {} ;

for(int i=0 ; i<n; i++)

{

ne [0] = sm[0][0]\*m[i][0] + sm[0][1]\*m[i][1] + sm[0][2]\*m[i][2] ;

ne [1] = sm[1][0]\*m[i][0] + sm[1][1]\*m[i][1] + sm[1][2]\*m[i][2] ;

ne [2] = sm[2][0]\*m[i][0] + sm[2][1]\*m[i][1] + sm[2][2]\*m[i][2] ;

m[i][0] = ne[0] ;

m[i][1] = ne[1] ;

m[i][2] = ne[2] ;

}

for(int i=0 ; i<n; i++)

{

int ni = (i+1)%n;

setpcolor(1,1,0) ;

conect(m[i][0],m[i][1],m[ni][0],m[ni][1]) ;

}}

void init()

{

glClearColor(0.0,0.0,0.0,0.0);

glClear (GL\_COLOR\_BUFFER\_BIT);

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3f(1.0,0.0,0.0);

glPointSize(2.0);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(-350,350,-350,350) ;

}

void menu(int c)

{

for(int i=0 ; i<n; i++)

{

int ni = (i+1)%n;

setpcolor(1,1,1) ;

conect(m[i][0],m[i][1],m[ni][0],m[ni][1]) ;

}

glFlush() ;

if (c==1)

{

for(int i=0; i<n; i++)

{

cout<<m[i][0]<<" "<<m[i][1]<<endl;

} }

else if(c==2)

{

int tx = 0, ty = 0 ;

cout <<"Enter x-translation factor : " ;

cin >> tx ;

cout <<"Enter y-translation factor : " ;

cin >> ty ;

translation(tx,ty) ;

}

else if(c==3)

{

double rot ;

int flg = 1,ym,xm ;

cout <<"Enter the arbitrary point x :" ;

cin >> xm ;

cout <<"Enter the arbitrary point y :" ;

cin >> ym ;

cout <<"Enter 1 for clockwise else enter 0 for anti-clock wise : " ;

cin >> flg ;

cout <<"Enter by how much degree the object is to be rotated : " ;

cin >> rot ;

if(flg)

{

rot = -rot ;

}

rotation(rot,xm,ym) ;

}

else if(c==4)

{

int sx = 1, sy = 1 ;

cout <<"Enter the horizontal scaling factor : " ;

cin >> sx ;

cout <<"Enter the vertical scaling factor : " ;

cin >> sy ;

scale(sx,sy) ;

}

else if(c==5)

{

char c ;

cout <<"Enter the axis of reflection : (X | Y |)" ;

cin >> c ;

reflect(c) ; }}

void mouse(int button, int state, int cx, int cy )

{

cx -= 350 ;

cy -= 350 ;

cy = - cy ;

if(state==GLUT\_DOWN)

{

if(button==GLUT\_LEFT\_BUTTON)

{

m[n][0] = cx ;

m[n][1] = cy ;

m[n][2] = 1 ;

n++;

if(n>1)

{

glColor3f(1.0,0.0,0.0);

glBegin(GL\_LINE\_STRIP);

glVertex2i(m[n-2][0],m[n-2][1]);

glVertex2i(m[n-1][0],m[n-1][1]);

glEnd();

glFlush();

} } }}

int main(int argc, char \*argv[])

{

glutInit(&argc,argv);

glutInitWindowSize(700,700);

glutInitWindowPosition(500,50);

glutCreateWindow(" 2D TRANSFORMATION ");

cout<<"PLEASE FOLLOW THESE STEPS:"<<endl;

cout<<"MAKE POLYGON by USING LEFT BUTTON CLICK"<<endl;

cout<<"FOR MENU, use the RIGHT button of the mouse"<<endl;

init();

glutDisplayFunc(choice);

glutMouseFunc(mouse);

glutCreateMenu(menu);

glutAddMenuEntry("DISPLAY AXES OF POLYGON",1);

glutAddMenuEntry("TRANSLATION",2);

glutAddMenuEntry("ROTATION",3);

glutAddMenuEntry("SCALING",4);

glutAddMenuEntry("REFLECTION",5);

glutAttachMenu(GLUT\_RIGHT\_BUTTON);

glutMainLoop();

return 0;

}