Assignment 6

Roll no 23148

\*

Implement translation, sheer, rotation and scaling transformations on equilateral triangle and

rhombus.

\*/

#include<iostream>

#include<stdio.h>

using namespace std;

#include<math.h>

#include<GL/glut.h>

#define PI 3.142857143

double P1[3][3],R1[4][4],P2[3][3],R2[4][4],T[3][3],T1[4][4];

int length,xi,yi;

int choice;

double angle;

void autoInit();

void Display();

void Display1();

void menu(int item);

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

{

glutInit(&argc,argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowPosition(0,0);

glutInitWindowSize(640,480);

cout<<"PLEASE SELECT:\n1.EQUILATERAL \n2.RHOMBUS \n3.EXIT";

cin>>choice;

switch(choice)

{

case 1:

printf("\nEnter co-ordinates of TRIANGLE => ");

scanf("%d%d",&xi,&yi);

printf("\nEnter length of TRIANGLE => ");

scanf("%d",&length);

int i,j;

for(i=0;i<3;i++)

for(j=0;j<3;j++)

P1[i][j]=1;

P1[0][0]=xi;

P1[0][1]=yi;

P1[1][0]=xi+length;

P1[1][1]=yi;

P1[2][1]=(sqrt(3)/2\*length)+yi;

P1[2][0]=length/2+xi;

break;

case 2:

printf("\nEnter co-ordinates of RHOMBUS => ");

scanf("%d%d",&xi,&yi);

printf("\nEnter length of RHOMBUS => ");

scanf("%d",&length);

printf("\nEnter angle of RHOMBUS =>" );

scanf("%lf",&angle);

for(i=0;i<4;i++)

for(j=0;j<4;j++)

R1[i][j]=1;

R1[0][0]=xi;

R1[0][1]=yi;

R1[1][0]=xi+length;

R1[1][1]=yi;

R1[2][1]=yi+length\*sin(angle);

R1[2][0]=length+xi+length\*cos(angle);

R1[3][0]=xi+length\*cos(angle);

R1[3][1]=yi+length\*sin(angle);

break;

}

/\*P1[0][0]=0;

P1[0][1]=0;

P1[0][2]=1;

P1[1][0]=100;

P1[1][1]=0;

P1[1][2]=1;

P1[2][0]=50;

P1[2][1]=86.6025;

P1[2][2]=1;

\*/

glutCreateWindow("TRANSFORMATIONS");

autoInit();

if(choice==1)

glutDisplayFunc(Display);

else

glutDisplayFunc(Display1);

glutCreateMenu(menu);

glutAddMenuEntry("1.Scaling",1);

glutAddMenuEntry("2.Translation",2);

glutAddMenuEntry("3.Rotation",3);

glutAddMenuEntry("4.Reflection",4);

glutAddMenuEntry("5.Shear",5);

glutAddMenuEntry("0.EXIT",0);

glutAttachMenu(GLUT\_RIGHT\_BUTTON);

glutMainLoop();

}

void autoInit()

{

glClearColor(0.0,0.0,0.0,0);

glColor3f(1.0,1.0,1.0);

glPointSize(4.0);

gluOrtho2D( -320 , 320 , -240 , 240 );

}

void poly(double P[3][3])

{

int i;

glBegin(GL\_LINE\_LOOP);

for(i=0;i<3;i++)

glVertex2d(P[i][0],P[i][1]);

glEnd();

glFlush();

}

void poly1(double R[4][4])

{

int i;

glBegin(GL\_LINE\_LOOP);

for(i=0;i<4;i++)

glVertex2d(R[i][0],R[i][1]);

glEnd();

glFlush();

}

void mult(double P[3][3],double T[3][3])

{

double sum;

int i,j,k;

for(i=0;i<3;i++)

{

for(j=0;j<3;j++)

{

sum=0;

for(k=0;k<3;k++)

sum=sum+P[i][k]\*T[k][j];

P2[i][j]=sum;

}

}

}

void mult1(double R[4][4],double T[4][4])

{

double sum;

int i,j,k;

for(i=0;i<4;i++)

{

for(j=0;j<4;j++)

{

sum=0;

for(k=0;k<4;k++)

sum=sum+R[i][k]\*T[k][j];

R2[i][j]=sum;

}

}

}

void scaling()

{

double sx,sy;

cout<<"\n\t\*\*SCALING\*\*\n";

cout<<"\n\tSx : ";

cin>>sx;

cout<<"\n\tSy : ";

cin>>sy;

T[0][0]=sx;

T[0][1]=0;

T[0][2]=0;

T[1][0]=0;

T[1][1]=sy;

T[1][2]=0;

T[2][0]=0;

T[2][1]=0;

T[2][2]=1;

mult(P1,T);

glColor3f(1.0,1.0,0.0);

poly(P2);

}

void scaling1()

{

double sx,sy;

cout<<"\n\t\*\*SCALING\*\*\n";

cout<<"\n\tSx : ";

cin>>sx;

cout<<"\n\tSy : ";

cin>>sy;

T1[0][0]=sx;

T1[0][1]=0;

T1[0][2]=0;

T1[0][3]=0;

T1[1][0]=0;

T1[1][1]=sy;

T1[1][2]=0;

T1[1][3]=0;

T1[2][0]=0;

T1[2][1]=0;

T1[2][2]=1;

T1[2][3]=0;

T1[3][0]=0;

T1[3][1]=0;

T1[3][2]=0;

T1[3][3]=1;

mult1(R1,T1);

glColor3f(1.0,1.0,0.0);

poly1(R2);

}

void translation()

{

double tx,ty;

cout<<"\n\t\*\*TRANSLATION\*\*\n";

cout<<"\n\tTx : ";

cin>>tx;

cout<<"\n\tTy : ";

cin>>ty;

T[0][0]=1;

T[0][1]=0;

T[0][2]=0;

T[1][0]=0;

T[1][1]=1;

T[1][2]=0;

T[2][0]=tx;

T[2][1]=ty;

T[2][2]=1;

mult(P1,T);

glColor3f(0.0,1.0,0.0);

poly(P2);

}

void translation1()

{

double tx,ty;

cout<<"\n\t\*\*TRANSLATION\*\*\n";

cout<<"\n\tTx : ";

cin>>tx;

cout<<"\n\tTy : ";

cin>>ty;

T1[0][0]=1;

T1[0][1]=0;

T1[0][2]=0;

T1[0][3]=0;

T1[1][0]=0;

T1[1][1]=1;

T1[1][2]=0;

T1[1][3]=0;

T1[2][0]=0;

T1[2][1]=0;

T1[2][2]=1;

T1[2][3]=0;

T1[3][0]=tx;

T1[3][1]=ty;

T1[3][2]=0;

T1[3][3]=1;

mult1(R1,T1);

glColor3f(0.0,1.0,0.0);

poly1(R2);

}

void rotation()

{

double rx,ry,angle,rad;

cout<<"\n\t\*\*ROTATION\*\*\n";

cout<<"\n\tArbitary Point (x,y) : ";

cin>>rx>>ry;

cout<<"\n\tAngle (in degree) : ";

cin>>angle;

rad=angle\*(PI/180);

T[0][0]=cos(rad);

T[0][1]=sin(rad);

T[0][2]=0;

T[1][0]=-sin(rad);

T[1][1]=cos(rad);

T[1][2]=0;

T[2][0]=(-(rx\*cos(rad))+(ry\*sin(rad))+rx);

T[2][1]=(-(rx\*sin(rad))-(ry\*cos(rad))+ry);

T[2][2]=1;

mult(P1,T);

glColor3f(0.0,1.0,0.0);

poly(P2);

}

void rotation1()

{

double rx,ry,angle,rad;

cout<<"\n\t\*\*ROTATION\*\*\n";

cout<<"\n\tArbitary Point (x,y) : ";

cin>>rx>>ry;

cout<<"\n\tAngle (in degree) : ";

cin>>angle;

rad=angle\*(PI/180);

T1[0][0]=cos(rad);

T1[0][1]=sin(rad);

T1[0][2]=0;

T1[0][3]=0;

T1[1][0]=-sin(rad);

T1[1][1]=cos(rad);

T1[1][2]=0;

T1[1][3]=0;

T1[2][0]=0;

T1[2][1]=0;

T1[2][2]=1;

T1[2][3]=0;

T1[3][0]=(-(rx\*cos(rad))+(ry\*sin(rad))+rx);

T1[3][1]=(-(rx\*sin(rad))-(ry\*cos(rad))+ry);

T1[3][2]=0;

T1[3][3]=1;

mult1(R1,T1);

glColor3f(0.0,1.0,0.0);

poly1(R2);

}

void reflection()

{

int choice;

cout<<"\n\t\*\*REFLECTION\*\*\n\t1.X-Axis\n\t2.Y-Axis\n\t3.ORIGIN\n\t4.about Y = X\n\t5.about Y = -X \n\tCHOICE>>";

cin>>choice;

switch(choice)

{

case 1:

T[0][0]=1;

T[0][1]=0;

T[0][2]=0;

T[1][0]=0;

T[1][1]=-1;

T[1][2]=0;

T[2][0]=0;

T[2][1]=0;

T[2][2]=1;

break;

case 2:

T[0][0]=-1;

T[0][1]=0;

T[0][2]=0;

T[1][0]=0;

T[1][1]=1;

T[1][2]=0;

T[2][0]=0;

T[2][1]=0;

T[2][2]=1;

break;

case 3:

T[0][0]=-1;

T[0][1]=0;

T[0][2]=0;

T[1][0]=0;

T[1][1]=-1;

T[1][2]=0;

T[2][0]=0;

T[2][1]=0;

T[2][2]=1;

break;

case 4:

T[0][0]=0;

T[0][1]=1;

T[0][2]=0;

T[1][0]=1;

T[1][1]=0;

T[1][2]=0;

T[2][0]=0;

T[2][1]=0;

T[2][2]=1;

break;

case 5:

T[0][0]=0;

T[0][1]=-1;

T[0][2]=0;

T[1][0]=-1;

T[1][1]=0;

T[1][2]=0;

T[2][0]=0;

T[2][1]=0;

T[2][2]=1;

break;

}

mult(P1,T);

glColor3f(0.0,1.0,0.0);

poly(P2);

}

void reflection1()

{

int choice;

cout<<"\n\t\*\*REFLECTION\*\*\n\t1.X-Axis\n\t2.Y-Axis\n\t3.ORIGIN\n\t4.about Y = X\n\t5.about Y = -X \n\tCHOICE>>";

cin>>choice;

switch(choice)

{

case 1:

T1[0][0]=1;

T1[0][1]=0;

T1[0][2]=0;

T1[0][3]=0;

T1[1][0]=0;

T1[1][1]=-1;

T1[1][2]=0;

T1[1][3]=0;

T1[2][0]=0;

T1[2][1]=0;

T1[2][2]=1;

T1[2][3]=0;

T1[3][0]=0;

T1[3][1]=0;

T1[3][2]=0;

T1[3][3]=1;

break;

case 2:

T1[0][0]=-1;

T1[0][1]=0;

T1[0][2]=0;

T1[0][3]=0;

T1[1][0]=0;

T1[1][1]=1;

T1[1][2]=0;

T1[1][3]=0;

T1[2][0]=0;

T1[2][1]=0;

T1[2][2]=1;

T1[2][3]=0;

T1[3][0]=0;

T1[3][1]=0;

T1[3][2]=0;

T1[3][3]=1;

break;

case 3:

T1[0][0]=-1;

T1[0][1]=0;

T1[0][2]=0;

T1[0][3]=0;

T1[1][0]=0;

T1[1][1]=-1;

T1[1][2]=0;

T1[1][3]=0;

T1[2][0]=0;

T1[2][1]=0;

T1[2][2]=1;

T1[2][3]=0;

T1[3][0]=0;

T1[3][1]=0;

T1[3][2]=0;

T1[3][3]=1;

break;

case 4:

T1[0][0]=0;

T1[0][1]=1;

T1[0][2]=0;

T1[0][3]=0;

T1[1][0]=1;

T1[1][1]=0;

T1[1][2]=0;

T1[1][3]=0;

T1[2][0]=0;

T1[2][1]=0;

T1[2][2]=1;

T1[2][3]=0;

T1[3][0]=0;

T1[3][1]=0;

T1[3][2]=0;

T1[3][3]=1;

break;

case 5:

T1[0][0]=0;

T1[0][1]=-1;

T1[0][2]=0;

T1[0][3]=0;

T1[1][0]=-1;

T1[1][1]=0;

T1[1][2]=0;

T1[1][3]=0;

T1[2][0]=0;

T1[2][1]=0;

T1[2][2]=1;

T1[2][3]=0;

T1[3][0]=0;

T1[3][1]=0;

T1[3][2]=0;

T1[3][3]=1;

break;

}

mult1(R1,T1);

glColor3f(0.0,1.0,0.0);

poly1(R2);

}

void shear1()

{

double xs,ys;

int choice;

cout<<"\n\t\*\*SHEAR\*\*";

cout<<"\n\t1.X - Shear\n\t2.Y - Shear\n\tCHOICE >>";

cin>>choice;

switch(choice)

{

case 1:

cout<<"\n\tX-shear value : ";

cin>>xs;

T1[0][0]=1;

T1[0][1]=0;

T1[0][2]=0;

T1[0][3]=0;

T1[1][0]=xs;

T1[1][1]=1;

T1[1][2]=0;

T1[1][3]=0;

T1[2][0]=0;

T1[2][1]=0;

T1[2][2]=1;

T1[2][3]=0;

T1[3][0]=0;

T1[3][1]=0;

T1[3][2]=0;

T1[3][3]=1;

break;

case 2:

cout<<"\n\tY-shear value : ";

cin>>ys;

T1[0][0]=1;

T1[0][1]=ys;

T1[0][2]=0;

T1[0][3]=0;

T1[1][0]=0;

T1[1][1]=1;

T1[1][2]=0;

T1[1][3]=0;

T1[2][0]=0;

T1[2][1]=0;

T1[2][2]=1;

T1[2][3]=0;

T1[3][0]=0;

T1[3][1]=0;

T1[3][2]=0;

T1[3][3]=1;

break;

}

mult1(R1,T1);

glColor3f(0.0,1.0,0.0);

poly1(R2);

}

void shear()

{

double xs,ys;

int choice;

cout<<"\n\t\*\*SHEAR\*\*";

cout<<"\n\t1.X - Shear\n\t2.Y - Shear\n\tCHOICE >>";

cin>>choice;

switch(choice)

{

case 1:

cout<<"\n\tX-shear value : ";

cin>>xs;

T[0][0]=1;

T[0][1]=0;

T[0][2]=0;

T[1][0]=xs;

T[1][1]=1;

T[1][2]=0;

T[2][0]=0;

T[2][1]=0;

T[2][2]=1;

break;

case 2:

cout<<"\n\tY-shear value : ";

cin>>ys;

T[0][0]=1;

T[0][1]=ys;

T[0][2]=0;

T[1][0]=0;

T[1][1]=1;

T[1][2]=0;

T[2][0]=0;

T[2][1]=0;

T[2][2]=1;

break;

}

mult(P1,T);

glColor3f(0.0,1.0,0.0);

poly(P2);

}

void Display()

{

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

glBegin(GL\_LINES);

glVertex2d(-320,0); glVertex2d(320,0);

glVertex2d(0,-240); glVertex2d(0,240);

glEnd();

glColor3f(1.0,0.0,0.0);

poly(P1);

glFlush();

}

void Display1()

{

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

glBegin(GL\_LINES);

glVertex2d(-320,0); glVertex2d(320,0);

glVertex2d(0,-240); glVertex2d(0,240);

glEnd();

glColor3f(1.0,0.0,0.0);

poly1(R1);

glFlush();

}

void menu(int item)

{

switch(item)

{

case 1:

if(choice==1)

scaling();

else

scaling1();

break;

case 2:

if(choice==1)

translation();

else

translation1();

break;

case 3:

if(choice==1)

rotation();

else

rotation1();

break;

case 4:

if(choice==1)

reflection();

else

reflection1();

break;

case 5:

if(choice==1)

shear();

else

shear1();

break;

case 0:

exit(0);

}

}



