#include <math.h>

#include <GL/glut.h>

#include <stdio.h>

#include <stdlib.h>

typedef float Matrix3x3 [3][3];

Matrix3x3 theMatrix;

float ptsIni[4][2]={{80,80},{180,80},{180,180},{80,180}};

//Realign above line while execution

// Initial Co-ordinates of the Cube to be Transformed

float ptsFin[4][2];

//float refptX,refptY; //Reference points

float TransDistX,TransDistY; //Translations along Axes

float ScaleX,ScaleY; //Scaling Factors along Axes

float Alpha; //Rotation angles about Axes

int choice;

void matrixSetIdentity(Matrix3x3 m) // Initialises the matrix as Unit Matrix

{

int i, j;

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

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

m[i][j] = (i == j);

}

void matrixPreMultiply(Matrix3x3 a , Matrix3x3 b)

{// Multiplies matrix a times b, putting result in b

int i,j,k;

Matrix3x3 tmp;

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

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

{

tmp[i][j]=0;

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

{

tmp[i][j]=tmp[i][j]+a[i][k]\*b[k][j];

}

}

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

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

b[i][j] = tmp[i][j];

}

void Translate(int tx, int ty)

{

Matrix3x3 m;

matrixSetIdentity(m);

m[0][2] = tx;

m[1][2] = ty;

matrixPreMultiply(m, theMatrix);

}

void Scale(float sx , float sy)

{

Matrix3x3 m;

matrixSetIdentity(m);

m[0][0] = sx;

//m[0][2] = (1 - sx)\*refptX;

m[1][1] = sy;

//m[1][2] = (1 - sy)\*refptY;

matrixPreMultiply(m , theMatrix);

}

void Rotate(float angle)

{

Matrix3x3 m;

matrixSetIdentity(m);

angle = angle\*22/1260;

m[0][0] = cos(angle);

m[0][1] = -sin(angle);

m[1][0] = sin(angle);

m[1][1] = cos(angle);

matrixPreMultiply(m , theMatrix);

}

void TransformPoints(void)

{

int i,k;

float tmp ;

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

for (i=0 ; i<2 ; i++)

ptsFin[k][i] = theMatrix[i][0]\*ptsIni[k][0] + theMatrix[i][1]\*ptsIni[k][1]+ theMatrix[i][2];

// Realign above line while execution

}

void Axes(void)

{

glColor3f (0.0, 0.0, 0.0); // Set the color to BLACK

glBegin(GL\_LINES); // Plotting X-Axis

glVertex2s(-1000 ,0);

glVertex2s( 1000 ,0);

glEnd();

glBegin(GL\_LINES); // Plotting Y-Axis

glVertex2s(0 ,-1000);

glVertex2s(0 , 1000);

glEnd();

}

void Draw(float a[4][2]) //Display the Figure

{

//glColor3f (1.0, 0.0, 0.0);

glBegin(GL\_POLYGON);

glVertex2f(a[0][0],a[0][1]);

glVertex2f(a[1][0],a[1][1]);

glVertex2f(a[2][0],a[2][1]);

glVertex2f(a[3][0],a[3][1]);

glEnd();

}

void display(void)

{

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

Axes();

glColor3f (1.0, 0.0, 0.0); // Set the color to RED

Draw(ptsIni);

matrixSetIdentity(theMatrix);

switch(choice)

{

case 1: Translate(TransDistX , TransDistY);

Rotate(Alpha);

Translate(-TransDistX , -TransDistY);

break;

case 2:

Translate(TransDistX , TransDistY);

Scale(ScaleX, ScaleY);

Translate(-TransDistX , -TransDistY);

break;

case 3:

Rotate(-Alpha);

Scale(ScaleX, ScaleY);

Rotate(-Alpha);

break;

}

TransformPoints();

glColor3f(0.0,1.0,0.0);

Draw(ptsFin);

glFlush();

}

void init(void)

{

glClearColor (1.0, 1.0, 1.0, 1.0);

// Set the Background color to WHITE

gluOrtho2D(-454.0, 454.0, -250.0, 250.0);

// Set the no. of Co-ordinates along X & Y axes and their gappings

}

int main (int argc, char \*argv)

{

glutInit(&argc, &argv);

glutInitDisplayMode (GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize (1362, 750);

glutInitWindowPosition (0, 0);

glutCreateWindow (" Composite Transformations ");

init ();

printf("Enter your choice number:\n1.General Pivot Point Rotation\n2.General Fixed Point Scaling \n3.General Scaling Direction\=>");

scanf("%d",&choice);

switch(choice)

{

case 1:

printf("Enter Translation along X, Y\n=>");

scanf("%f%f",&TransDistX , &TransDistY);

printf("Enter Rot. Angle Alpha: ");

scanf("%f",&Alpha);

break;

case 2:

printf("Enter Translation along X, Y\n=>");

scanf("%f%f",&TransDistX , &TransDistY);

printf("Enter Scaling ratios along X, Y\n=>");

scanf("%f%f",&ScaleX , &ScaleY);

break;

case 3:

printf("Enter Scaling ratios along X, Y\n=>");

scanf("%f%f",&ScaleX , &ScaleY);

printf("Enter Rot. Angle Alpha: ");

scanf("%f",&Alpha);

break;

}

glutDisplayFunc(display);

glutMainLoop();

return 0;

}

#include <math.h>

#include <GL/glut.h>

#include <stdio.h>

#include <stdlib.h>

typedef float Matrix4x4 [4][4];

Matrix4x4 theMatrix;

float ptsIni[8][3]={{80,80,-100},{180,80,-100},{180,180,-100},{80,180,-100},

{60,60,0},{160,60,0},{160,160,0},{60,160,0}};

//Realign above line while execution

// Initial Co-ordinates of the Cube to be Transformed

float ptsFin[8][3];

float refptX,refptY,refptZ; //Reference points

float TransDistX,TransDistY,TransDistZ; //Translations along Axes

float ScaleX,ScaleY,ScaleZ; //Scaling Factors along Axes

float Alpha,Beta,Gamma,Theta; //Rotation angles about Axes

float A,B,C; //Arbitrary Line Attributes

float aa,bb,cc; //Arbitrary Line Attributes

float x1,y11,z1,x2,y2,z2;

int choice,choiceRot,choiceRef;

void matrixSetIdentity(Matrix4x4 m) // Initialises the matrix as Unit Matrix

{

int i, j;

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

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

m[i][j] = (i == j);

}

void matrixPreMultiply(Matrix4x4 a , Matrix4x4 b)

{// Multiplies matrix a times b, putting result in b

int i,j,k;

Matrix4x4 tmp;

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

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

{

tmp[i][j]=0;

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

{

tmp[i][j]=tmp[i][j]+a[i][k]\*b[k][j];

}

}

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

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

theMatrix[i][j] = tmp[i][j];

}

void Translate(int tx, int ty, int tz)

{

Matrix4x4 m;

matrixSetIdentity(m);

m[0][3] = tx;

m[1][3] = ty;

m[2][3] = tz;

matrixPreMultiply(m, theMatrix);

}

void Scale(float sx , float sy ,float sz)

{

Matrix4x4 m;

matrixSetIdentity(m);

m[0][0] = sx;

m[0][3] = (1 - sx)\*refptX;

m[1][1] = sy;

m[1][3] = (1 - sy)\*refptY;

m[2][2] = sz;

m[2][3] = (1 - sy)\*refptZ;

matrixPreMultiply(m , theMatrix);

}

void RotateX(float angle)

{

Matrix4x4 m;

matrixSetIdentity(m);

angle = angle\*22/1260;

m[1][1] = cos(angle);

m[1][2] = -sin(angle);

m[2][1] = sin(angle);

m[2][2] = cos(angle);

matrixPreMultiply(m , theMatrix);

}

void RotateY(float angle)

{

Matrix4x4 m;

matrixSetIdentity(m);

angle = angle\*22/1260;

m[0][0] = cos(angle);

m[0][2] = sin(angle);

m[2][0] = -sin(angle);

m[2][2] = cos(angle);

matrixPreMultiply(m , theMatrix);

}

void RotateZ(float angle)

{

Matrix4x4 m;

matrixSetIdentity(m);

angle = angle\*22/1260;

m[0][0] = cos(angle);

m[0][1] = -sin(angle);

m[1][0] = sin(angle);

m[1][1] = cos(angle);

matrixPreMultiply(m , theMatrix);

}

void Reflect(void)

{

Matrix4x4 m;

matrixSetIdentity(m);

switch(choiceRef)

{

case 1: m[2][2] = -1;

break;

case 2: m[0][0] = -1;

break;

case 3: m[1][1] = -1;

break;

}

matrixPreMultiply(m , theMatrix);

}

void DrawRotLine(void)

{

switch(choiceRot)

{

case 1: glBegin(GL\_LINES);

glVertex3s(-1000 ,B,C);

glVertex3s( 1000 ,B,C);

glEnd();

break;

case 2: glBegin(GL\_LINES);

glVertex3s(A ,-1000 ,C);

glVertex3s(A ,1000 ,C);

glEnd();

break;

case 3: glBegin(GL\_LINES);

glVertex3s(A ,B ,-1000);

glVertex3s(A ,B ,1000);

glEnd();

break;

case 4: glBegin(GL\_LINES);

glVertex3s(x1-aa\*500 ,y11-bb\*500 , z1-cc\*500);

glVertex3s(x2+aa\*500 ,y2+bb\*500 , z2+cc\*500);

glEnd();

break;

}

}

void TransformPoints(void)

{

int i,k;

float tmp ;

for(k=0 ; k<8 ; k++)

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

ptsFin[k][i] = theMatrix[i][0]\*ptsIni[k][0] + theMatrix[i][1]\*ptsIni[k][1]

+ theMatrix[i][2]\*ptsIni[k][2] + theMatrix[i][3];

// Realign above line while execution

}

void Axes(void)

{

glColor3f (0.0, 0.0, 0.0); // Set the color to BLACK

glBegin(GL\_LINES); // Plotting X-Axis

glVertex2s(-1000 ,0);

glVertex2s( 1000 ,0);

glEnd();

glBegin(GL\_LINES); // Plotting Y-Axis

glVertex2s(0 ,-1000);

glVertex2s(0 , 1000);

glEnd();

}

void Draw(float a[8][3]) //Display the Figure

{

int i;

glColor3f (0.7, 0.4, 0.7);

glBegin(GL\_POLYGON);

glVertex3f(a[0][0],a[0][1],a[0][2]);

glVertex3f(a[1][0],a[1][1],a[1][2]);

glVertex3f(a[2][0],a[2][1],a[2][2]);

glVertex3f(a[3][0],a[3][1],a[3][2]);

glEnd();

i=0;

glColor3f (0.8, 0.6, 0.5);

glBegin(GL\_POLYGON);

glVertex3s(a[0+i][0],a[0+i][1],a[0+i][2]);

glVertex3s(a[1+i][0],a[1+i][1],a[1+i][2]);

glVertex3s(a[5+i][0],a[5+i][1],a[5+i][2]);

glVertex3s(a[4+i][0],a[4+i][1],a[4+i][2]);

glEnd();

glColor3f (0.2, 0.4, 0.7);

glBegin(GL\_POLYGON);

glVertex3f(a[0][0],a[0][1],a[0][2]);

glVertex3f(a[3][0],a[3][1],a[3][2]);

glVertex3f(a[7][0],a[7][1],a[7][2]);

glVertex3f(a[4][0],a[4][1],a[4][2]);

glEnd();

i=1;

glColor3f (0.5, 0.4, 0.3);

glBegin(GL\_POLYGON);

glVertex3s(a[0+i][0],a[0+i][1],a[0+i][2]);

glVertex3s(a[1+i][0],a[1+i][1],a[1+i][2]);

glVertex3s(a[5+i][0],a[5+i][1],a[5+i][2]);

glVertex3s(a[4+i][0],a[4+i][1],a[4+i][2]);

glEnd();

i=2;

glColor3f (0.5, 0.6, 0.2);

glBegin(GL\_POLYGON);

glVertex3s(a[0+i][0],a[0+i][1],a[0+i][2]);

glVertex3s(a[1+i][0],a[1+i][1],a[1+i][2]);

glVertex3s(a[5+i][0],a[5+i][1],a[5+i][2]);

glVertex3s(a[4+i][0],a[4+i][1],a[4+i][2]);

glEnd();

i=4;

glColor3f (0.7, 0.3, 0.4);

glBegin(GL\_POLYGON);

glVertex3f(a[0+i][0],a[0+i][1],a[0+i][2]);

glVertex3f(a[1+i][0],a[1+i][1],a[1+i][2]);

glVertex3f(a[2+i][0],a[2+i][1],a[2+i][2]);

glVertex3f(a[3+i][0],a[3+i][1],a[3+i][2]);

glEnd();

}

void display(void)

{

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

Axes();

glColor3f (1.0, 0.0, 0.0); // Set the color to RED

Draw(ptsIni);

matrixSetIdentity(theMatrix);

switch(choice)

{

case 1: Translate(TransDistX , TransDistY ,TransDistZ);

break;

case 2: Scale(ScaleX, ScaleY, ScaleZ);

break;

case 3: switch(choiceRot)

{

case 1: DrawRotLine();

Translate(0,-B,-C);

RotateX(Alpha);

Translate(0,B,C);

break;

case 2: DrawRotLine();

Translate(-A,0,-C);

RotateY(Beta);

Translate(A,0,C);

break;

case 3: DrawRotLine();

Translate(-A,-B,0);

RotateZ(Gamma);

Translate(A,B,0);

break;

case 4: DrawRotLine();

float MOD =sqrt((x2-x1)\*(x2-x1) + (y2-y11)\*(y2-y11) + (z2-z1)\*(z2-z1));

aa = (x2-x1)/MOD;

bb = (y2-y11)/MOD;

cc = (z2-z1)/MOD;

Translate(-x1,-y11,-z1);

float ThetaDash;

ThetaDash = 1260\*atan(bb/cc)/22;

RotateX(ThetaDash);

RotateY(1260\*asin(-aa)/22);

RotateZ(Theta);

RotateY(1260\*asin(aa)/22);

RotateX(-ThetaDash);

Translate(x1,y11,z1);

break;

}

break;

case 4: Reflect();

break;

}

TransformPoints();

Draw(ptsFin);

glFlush();

}

void init(void)

{

glClearColor (1.0, 1.0, 1.0, 1.0);

// Set the Background color to WHITE

glOrtho(-454.0, 454.0, -250.0, 250.0, -250.0, 250.0);

// Set the no. of Co-ordinates along X & Y axes and their gappings

glEnable(GL\_DEPTH\_TEST);

// To Render the surfaces Properly according to their depths

}

int main (int argc, char \*argv)

{

glutInit(&argc, &argv);

glutInitDisplayMode (GLUT\_SINGLE | GLUT\_RGB | GLUT\_DEPTH);

glutInitWindowSize (1362, 750);

glutInitWindowPosition (0, 0);

glutCreateWindow (" Basic Transformations ");

init ();

printf("Enter your choice number:\n1.Translation\n2.Scaling\n3.Rotation\n4.Reflection\n=>");

scanf("%d",&choice);

switch(choice)

{

case 1:printf("Enter Translation along X, Y & Z\n=>");

scanf("%f%f%f",&TransDistX , &TransDistY , &TransDistZ);

break;

case 2:printf("Enter Scaling ratios along X, Y & Z\n=>");

scanf("%f%f%f",&ScaleX , &ScaleY , &ScaleZ);

break;

case 3:printf("Enter your choice for Rotation about axis:\n1.parallel to X-axis.(y=B & z=C)\n2.parallel to Y-axis.(x=A & z=C)\n3.parallel to Z-axis.(x=A & y=B)\n4.Arbitrary line passing through (x1,y1,z1) & (x2,y2,z2)\n =>");

//Realign above line while execution

scanf("%d",&choiceRot);

switch(choiceRot)

{

case 1: printf("Enter B & C: ");

scanf("%f %f",&B,&C);

printf("Enter Rot. Angle Alpha: ");

scanf("%f",&Alpha);

break;

case 2: printf("Enter A & C: ");

scanf("%f %f",&A,&C);

printf("Enter Rot. Angle Beta: ");

scanf("%f",&Beta);

break;

case 3: printf("Enter A & B: ");

scanf("%f %f",&A,&B);

printf("Enter Rot. Angle Gamma: ");

scanf("%f",&Gamma);

break;

case 4: printf("Enter values of x1 ,y1 & z1:\n");

scanf("%f %f %f",&x1,&y11,&z1);

printf("Enter values of x2 ,y2 & z2:\n");

scanf("%f %f %f",&x2,&y2,&z2);

printf("Enter Rot. Angle Theta: ");

scanf("%f",&Theta);

break;

}

break;

case 4: printf("Enter your choice for reflection about plane:\n1.X-Y\n2.Y-Z\n3.X-Z\n=>");

scanf("%d",&choiceRef);

break;

default: printf("Please enter a valid choice!!!\n");

return 0;

}

glutDisplayFunc(display);

glutMainLoop();

return 0;

}

#include <windows.h>

#include<GL/glut.h>

#include<GL/gl.h>

#include<stdio.h>

float X1,Y1,X2,Y2;

float xmin=-100;

float ymin=-100;

float xmax=100;

float ymax=100;

void init(void)

{

glClearColor(0.0,0,0,0);

gluOrtho2D(-300,300,-300,300);

}

int code(float x,float y)

{

int c=0;

if(y>ymax)c=8;

if(y<ymin)c=4;

if(x>xmax)c=2;

if(x<xmin)c=1;

return c;

}

void cohen\_Line(float x1,float y1,float x2,float y2)

{

int c1=code(x1,y1);

int c2=code(x2,y2);

float m=(y2-y1)/(x2-x1);

while((c1|c2)>0)

{

if((c1 && c2)>0)

{

printf("The Line Will be outside the Clipping Window and it will be completely discarded");

break;

}

float xi=x1;float yi=y1;

int c=c1;

if(c==0)

{

c=c2;

xi=x2;

yi=y2;

}

float x,y;

if((c & 8)>0)

{

y=ymax;

x=xi+ 1.0/m\*(ymax-yi);

}

else

if((c & 4)>0)

{

y=ymin;

x=xi+1.0/m\*(ymin-yi);

}

else

if((c & 2)>0)

{

x=xmax;

y=yi+m\*(xmax-xi);

}

else

if((c & 1)>0)

{

x=xmin;

y=yi+m\*(xmin-xi);

}

if(c==c1)

{

X1=x;

Y1=y;

c1=code(X1,Y1);

}

if(c==c2)

{

X2=x;

Y2=y;

c2=code(X2,Y2);

}

}

glColor3f(0.0,0.0,1.0);

glBegin(GL\_LINES);

glVertex2i(X1,Y1);

glVertex2i(X2,Y2);

glEnd();

}

void display(void)

{

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3f(0.0,1.0,0.0);

glBegin(GL\_LINE\_LOOP);

glVertex2i(xmin,ymin);

glVertex2i(xmin,ymax);

glVertex2i(xmax,ymax);

glVertex2i(xmax,ymin);

glEnd();

glColor3f(1.0,0.0,0.0);

glBegin(GL\_LINES);

glVertex2i(X1,Y1);

glVertex2i(X2,Y2);

glEnd();

cohen\_Line(X1,Y1,X2,Y2);

glFlush();

}

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

{

printf("Enter line Start co-ordinates X1 and Y1\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*:");

scanf("%f%f",&X1,&Y1);

printf("Enter line Start co-ordinates X2 and Y2\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*:");

scanf("%f%f",&X2,&Y2);

glutInit(&argc,argv);

glutInitDisplayMode(GLUT\_SINGLE|GLUT\_RGB);

glutInitWindowSize(600,600);

glutInitWindowPosition(0,0);

glutCreateWindow("Clipping");

glutDisplayFunc(display);

init();

glutMainLoop();

return (0);

}

#include<stdio.h>

#include<GL/glut.h>

#include<GL/gl.h>

#include<windows.h>

#include<time.h>

int xmi,xma,ymi,yma;

int xmin=-100;

int xmax=100;

int ymin=-100;

int ymax=100;

int cut;

void display1()

{

glColor3f(0.0,1.0,0.0);

glClear(GL\_COLOR\_BUFFER\_BIT);

glBegin(GL\_LINE\_LOOP);

glVertex2i(xmin,ymin);

glVertex2i(xmax,ymin);

glVertex2i(xmax,ymax);

glVertex2i(xmin,ymax);

glEnd();

glColor3f(1.0,0.0,0.0);

glBegin(GL\_POLYGON);

glVertex2i(xmi,ymi);

glVertex2i(xma,ymi);

glVertex2i(xma,yma);

glVertex2i(xmi,yma);

glEnd();

glFlush();

}

void clip(void)

{

while(xmi<xmin|xma>xmax|ymi<ymin|yma>ymax)

{

if(xmi<xmin)

{

cut=xmin-xmi;

xmi=xmi+cut;

}

else if(xma>xmax)

{

cut=xma-xmax;

xma=xma-cut;

}

else if(ymi<ymin)

{

cut=ymin-ymi;

ymi=ymi+cut;

}

else if(yma>ymax)

{

cut=yma-ymax;

yma=yma-cut;

}

}

}

void delay(float secs)

{

float end = clock()/CLOCKS\_PER\_SEC + secs;

while((clock()/CLOCKS\_PER\_SEC) < end);

}

void display()

{

display1();

clip();

delay(3);

display1();

}

void init(void)

{

glClearColor(0.0,0,0,0);

gluOrtho2D(-300,300,-300,300);

}

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

{

glutInit(&argc,argv);

glutInitDisplayMode(GLUT\_SINGLE|GLUT\_RGB);

printf("enter the points for left bottom of polygon:");

scanf("%d%d",&xmi,&ymi);

printf("Enter the points for right up of the polygon:");

scanf("%d%d",&xma,&yma);

glutInitWindowSize(600,600);

glutInitWindowPosition(0,0);

glutCreateWindow("Clipping");

glutDisplayFunc(display);

init();

glutMainLoop();

return (0);

}