**Pune Institute Of Computer Technology Dhankawadi,**

**Pune – 43.**

Assignment No. 6

Computer Graphics

**SE-IT-10 ACADEMIC YEAR :- 2020-2021**

**Name :- Diptesh Ravindra Varule Roll No :- 23277**

**Topic Name**:

|  |
| --- |
| Implement following 2D transformations on the object with respect to axis : –   1. Scaling 2. Rotation about arbitrary point 3. Reflection |

Source Code:

Rotation:

#include <GL/freeglut.h>

#include <GL/gl.h>

#include <iostream>

using namespace std;

struct Color//declare color stucture

{

float r,g,b;

};

Color getPixelcolor(float x,float y)//get pixelcolor

{

Color c;

glReadPixels(x,y,1,1,GL\_RGB,GL\_FLOAT,&c);//get color in 'c'

return c;//return c

}//end

void setPixelcolor(float x,float y)

{

glBegin(GL\_POINTS);//draw point

glColor3f(1.0,0.0,0.0);//set point color to red

glVertex2f(x,y);

glEnd();

}//end

void floodfill(float x,float y)

{

Color c = getPixelcolor(x,y);//gets color of current pixel

Color old = {1.0,1.0,1.0};

if(c.r == old.r && c.g == old.g && c.b == old.b)//if color of current pixel if white

{

setPixelcolor(x,y);//set pixel color to red

floodfill(x+1,y);//call floodfill recursively for four-connected points

floodfill(x,y+1);

floodfill(x-1,y);

floodfill(x,y-1);

}

return;

}//end

void render()

{

glClearColor(1.0,1.0,1.0,0.0);//clear color to white

glClear(GL\_COLOR\_BUFFER\_BIT);//set color

glMatrixMode(GL\_PROJECTION);//set matrix mode

glLoadIdentity();//load identity matrix

gluOrtho2D(0,400,0,400);//sets axis length

glFlush();//flush buffer and execute all command

}//end

void draw()//draw '+' Diagram

{

glBegin(GL\_LINES);

glColor3f(0.0,0.0,0.0);//sets black color

glVertex2d(100,150);

glVertex2d(100,200);

glVertex2d(100,200);

glVertex2d(50,200);

glVertex2d(50,200);

glVertex2d(50,220);

glVertex2d(50,220);

glVertex2d(100,220);

glVertex2d(100,220);

glVertex2d(100,270);

glVertex2d(100,270);

glVertex2d(120,270);

glVertex2d(120,270);

glVertex2d(120,220);

glVertex2d(120,220);

glVertex2d(170,220);

glVertex2d(170,220);

glVertex2d(170,200);

glVertex2d(170,200);

glVertex2d(120,200);

glVertex2d(120,200);

glVertex2d(120,150);

glVertex2d(120,150);

glVertex2d(100,150);

glEnd();//end

floodfill(110,210);//fill '+' Diagram

glFlush();//flush buffer and execute all command

}//end

void rotate()//draw '+' Diagram

{

float y\_tra = 210 - (320/1.41);//y-translation to have same level

glBegin(GL\_LINES);

glColor3f(0.0,0.0,0.0);//sets black color

glVertex2d(400+(-50/1.41),250/1.41 + y\_tra);

glVertex2d(400+(-100/1.41),300/1.41 + y\_tra);

glVertex2d(400+(-100/1.41),300/1.41 + y\_tra);

glVertex2d(400+(-150/1.41),(250/1.41) + y\_tra);

glVertex2d(400+(-150/1.41),(250/1.41) + y\_tra);

glVertex2d(400+(-170/1.41),270/1.41 + y\_tra);

glVertex2d(400+(-170/1.41),270/1.41 + y\_tra);

glVertex2d(400 + (-120/1.41),320/1.41 + y\_tra);

glVertex2d(400 + (-120/1.41),320/1.41 + y\_tra);

glVertex2d(400 +(-170/1.41),370/1.41 + y\_tra);

glVertex2d(400 +(-170/1.41),370/1.41 + y\_tra);

glVertex2d(400+(-150/1.41),390/1.41 + y\_tra);

glVertex2d(400+(-150/1.41),390/1.41 + y\_tra);

glVertex2d(400+(-100/1.41),340/1.41 + y\_tra);

glVertex2d(400+(-100/1.41),340/1.41 + y\_tra);

glVertex2d(400+(-50/1.41),390/1.41 + y\_tra);

glVertex2d(400+(-50/1.41),390/1.41 + y\_tra);

glVertex2d(400+(-30/1.41),370/1.41 + y\_tra);

glVertex2d(400+(-30/1.41),370/1.41 + y\_tra);

glVertex2d(400+(-80/1.41),320/1.41 + y\_tra);

glVertex2d(400+(-80/1.41),320/1.41 + y\_tra);

glVertex2d(400+(-30/1.41),270/1.41 + y\_tra);

glVertex2d(400+(-30/1.41),270/1.41 + y\_tra);

glVertex2d(400+(-50/1.41),250/1.41 + y\_tra);

glEnd();//end

floodfill(400+(-100/1.41),320/1.41 + y\_tra);//fill rotated Diagram

glFlush();//flush buffer and execute all command

}//end

void mouse(int button,int state,int x,int y)//On only when menu is commented

{

if(button == GLUT\_LEFT\_BUTTON && state == GLUT\_UP)//if left button and up

{

render();//draw '+' diagram

draw();

}

else if(button == GLUT\_RIGHT\_BUTTON && state == GLUT\_UP)//if right button and up

{

render();//draw '+' and rotated diagram

draw();

rotate();

}

}//end

int main(int argc,char \*\*argv)//taking command line orguments

{

int e;

glutInit(&argc,argv);//initialise glut with libraries

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);//initalise mode

glutInitWindowPosition(1000,200);//sets position of window

glutInitWindowSize(400,400);//sets size of window

glutCreateWindow("Practical");//create window

glutMouseFunc(mouse);//activate mouse function(activated only when menu is commented)

render();//call to function

do

{

cout<<"\*Menu = \n1 : Given Diagram\n2 : Rotated Diagram\n3 : Exit\n";//create menu

cout<<"Enter Your Choice = ";

cin>>e;

switch(e)

{

case 1://Draw '+' Diagram

render();//clear screen

draw();

break;

case 2://Draw rotated '+' Diagram

rotate();

break;

case 3://exit from program

exit(1);

}

}while(e);

glutMainLoop();//infinite loop untill user closes window

return 0;

}//end of program

Output:

A picture containing graphical user interface

Description automatically generated

Scaling and Reflection :

#include <GL/freeglut.h>

#include <GL/gl.h>

#include <math.h>

#include <stdio.h>

struct Pt

{

int x,y;

};

int numv,cnt;

bool inp;

Pt points[10];

void initGlobalVars(){

inp=false;

cnt=0;

numv=0;

}

/\* ----------------------- DDA LINE ALGORITHM ----------------------- \*/

void LineDDA(int x1,int y1,int x2,int y2)

{

float dx,dy,incx,incy;

float x,y;

int steps,i;

dx=x2-x1;

dy=y2-y1;

steps=(abs(dx)>abs(dy))?abs(dx):abs(dy);

incx=dx/float(steps);

incy=dy/float(steps);

x=x1;y=y1;

glBegin(GL\_POINTS);

glVertex2f(x,y);

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

{

x+=incx;

y+=incy;

glVertex2f(x,y);

}

glFlush();

glEnd();

}

/\*-------------- INITIALISE DRAWING WINDOW --------------\*/

void init()

{

glClearColor(0.0, 0.0, 0.0, 0.0);

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3f(1.0, 1.0, 1.0);

gluOrtho2D(0,500,0,500);

//glOrtho(-1.0, 1.0, -1.0, 1.0, -1.0, 1.0);

LineDDA(0,250,500,250);

LineDDA(250,0,250,500);

glRasterPos3f(0,246,1);

glutBitmapCharacter(GLUT\_BITMAP\_HELVETICA\_18,'<');

glRasterPos3f(490,246,1);

glutBitmapCharacter(GLUT\_BITMAP\_HELVETICA\_18,'>');

glRasterPos3f(246,0,1);

glutBitmapCharacter(GLUT\_BITMAP\_HELVETICA\_18,'v');

glRasterPos3f(246,487,1);

glutBitmapCharacter(GLUT\_BITMAP\_HELVETICA\_18,'^');

glColor3f(1.0,1.0,0.0);

}

void clrScr()

{

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3f(1.0, 1.0, 1.0);

LineDDA(0,250,500,250);

LineDDA(250,0,250,500);

glRasterPos3f(0,246,1);

glutBitmapCharacter(GLUT\_BITMAP\_HELVETICA\_18,'<');

glRasterPos3f(490,246,1);

glutBitmapCharacter(GLUT\_BITMAP\_HELVETICA\_18,'>');

glRasterPos3f(246,0,1);

glutBitmapCharacter(GLUT\_BITMAP\_HELVETICA\_18,'v');

glRasterPos3f(246,487,1);

glutBitmapCharacter(GLUT\_BITMAP\_HELVETICA\_18,'^');

glColor3f(1.0,1.0,0.0);

}

void drawFig(){

int i;

//clrScr();

for(i=0;i<numv-1;i++)

{

LineDDA(points[i].x,points[i].y,points[i+1].x,points[i+1].y);

}

LineDDA(points[0].x,points[0].y,points[i].x,points[i].y);

}

/\*--------------------------- INPUT VERTICES ---------------------------\*/

void input(){

int ch,i,x,y;

if(inp==true){

printf("\nAlready Input\n");

return;

}

printf("\nENTER NUMBER OF VERTICES :: ");

scanf("%d",&numv);

printf("Input points using :-\n1.Keyboard\n2.Mouse\nENTER CHOICE ::");

scanf("%d",&ch);

if(ch!=1)

return;

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

{

printf("\nENTER POINT (X Y) :: ");

scanf("%d %d",&x,&y);

points[i].x=250+x;

points[i].y=250+y;

}

inp=true;

}

/\*--------------------------- MOUSE FUNCTION ---------------------------\*/

void mouseinp(int button,int action,int xMouse,int yMouse){

if(inp==false)

{

if(cnt<numv)

{

if(button==GLUT\_LEFT\_BUTTON && action==GLUT\_DOWN)

{

//printf("%d %d",xMouse,yMouse);

points[cnt].x=xMouse;

points[cnt].y=500-yMouse;

cnt++;

}

}

else

{

inp=true;

drawFig();

}

}

}

/\*--------------------- TRANSLATE -------------------------\*/

void translate(float tx,float ty)

{ int i;

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

{

points[i].x+=tx;

points[i].y+=ty;

}

}

/\*--------------------- ROTATE -------------------------\*/

void rotate(){

float ang,angsin,angcos,x,y;

int i;

printf("\nENTER ANGLE OF ROTATION :: ");

scanf("%f",&ang);

ang=(ang\*3.141)/180;

angsin=sin(ang);

angcos=cos(ang);

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

{

x=points[i].x-250;

y=points[i].y-250;

//printf("\n%f %f\n",x,y);

x=(x\*angcos)-(y\*angsin);

y=((points[i].x-250)\*angsin)+(y\*angcos);

points[i].x=x+250;

points[i].y=y+250;

//printf("\n%d %d\n",points[i].x,points[i].y);

}

glColor3f(1.0,0.0,0.0);

}

/\*--------------------- SHEAR -------------------------\*/

void shear(){

int ch,i;

float x,y;

printf("\n1.X - SHEAR\n2.Y - SHEAR\nENTER CHOICE :: ");

scanf("%d",&ch);

if(ch==1)

{

printf("\nENTER X - SHEAR FACTOR :: ");

scanf("%d",&ch);

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

{

x=points[i].x-250;

y=points[i].y-250;

x=x+(y\*ch)-((points[0].y-250)\*ch);

points[i].x=x+250;

points[i].y=y+250;

}

}

else if(ch==2)

{

printf("\nENTER Y - SHEAR FACTOR :: ");

scanf("%d",&ch);

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

{

x=points[i].x-250;

y=points[i].y-250;

y=y+(x\*ch)-((points[0].x-250)\*ch);

points[i].x=x+250;

points[i].y=y+250;

}

}

else

{

printf("\n!! INVALID INPUT !!\n");

}

}

/\*--------------------- SCALE -------------------------\*/

void scale(float sx,float sy)

{

float x,y,xi,yi;

int i;

xi=points[0].x;

yi=points[0].y;

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

{

x=points[i].x-250;

y=points[i].y-250;

x=(x\*sx);

y=(y\*sy);

points[i].x=x+250;

points[i].y=y+250;

}

translate(xi-points[0].x,yi-points[0].y);

glColor3f(0.0,1.0,0.0);

}

/\*--------------------- REFLECT -------------------------\*/

void reflect(){

int ch,i;

float x,y;

printf("REFLECTION ABOUT :-\n1.X - AXIS\n2.Y - AXIS\n3.ORIGIN\nENTER CHOICE :: ");

scanf("%d",&ch);

switch(ch)

{

case 1:

for(i=0;i<numv;i++){

y=(-1)\*(points[i].y-250);

points[i].y=250+y;

}

break;

case 2:

for(i=0;i<numv;i++){

x=(-1)\*(points[i].x-250);

points[i].x=250+x;

}

break;

case 3:

for(i=0;i<numv;i++){

x=(-1)\*(points[i].x-250);

y=(-1)\*(points[i].y-250);

points[i].y=250+y;

points[i].x=250+x;

}

break;

}

glColor3f(1.0,.38,.01);

}

void menu(GLint ch)

{

float sx,sy;

switch(ch)

{

case 1:// Input

input();

break;

case 2:// Translate

printf("\nENTER TRANSLATION FACTORS (X Y) :: ");

scanf("%f %f",&sx,&sy);

translate(sx,sy);

break;

case 3:// Rotate

sx=250.0-points[1].x;

sy=250.0-points[1].y;

translate(sx,sy);

rotate();

translate(-sx,-sy);

break;

case 4:// Shear

shear();

break;

case 5:// Scale

printf("\nENTER SCALING FACTORS (X Y) :: ");

scanf("%f %f",&sx,&sy);

scale(sx,sy);

break;

case 6:// Reflect

reflect();

break;

}

drawFig();

}

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

{

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE);

glutInitWindowSize(500,500); // Define Window Size

glutInitWindowPosition(100,100);// Define Window Position

glutCreateWindow("2D Transformations");

init(); // Initialise drawing window

initGlobalVars(); // Initilaise Global Variables

glutDisplayFunc(drawFig); // Declare Drawing Function

glutMouseFunc(mouseinp); // Declare Mouse Function

glutCreateMenu(menu); // Define Menu

glutAddMenuEntry("Input",1);

glutAddMenuEntry("Translate",2);

glutAddMenuEntry("Rotate",3);

glutAddMenuEntry("Shear",4);

glutAddMenuEntry("Scale",5);

glutAddMenuEntry("Reflect",6);

glutAttachMenu(GLUT\_RIGHT\_BUTTON); // Attack menu to right mouse button

glutMainLoop();

return 0;

}

Output: Text

Description automatically generated

Shape

Description automatically generated