|  |  |
| --- | --- |
|  | **DEPARTMENT OF COMPUTER ENGINEERING** |



Semest

er



S.E. Semester III



–



Computer Engineering



Subj



Computer Graphics



Subject Professor In



-



char



Prof. Sachin Deshpande



Assisting

Teachers



Prof. Sachin Deshpande



|  |  |
| --- | --- |
| Student Name | Trisha Shah |
| Roll Number | 20102A0004 |

**EXPERIMENT DETAILS:**

|  |  |
| --- | --- |
| Title | Write a program to implement 2D Transformations |
| Tools/Language Used | Dev C++ |

Transformation means changing some graphics into something else by applying rules. We can have various types of transformations such as translation, scaling up or down, rotation, shearing, etc. When a transformation takes place on a 2D plane, it is called 2D transformation.

Transformations play an important role in computer graphics to reposition the graphics on the screen and change their size or orientation.

To perform a sequence of transformation such as translation followed by rotation and scaling, we need to follow a sequential process −

Translate the coordinates,

Rotate the translated coordinates, and then

Scale the rotated coordinates to complete the composite transformation.

**Program**

#include<stdio.h>

#include<graphics.h>

#include<conio.h>

#include<dos.h>

#include<math.h>

void translate(int x1,int y1, int x2, int y2, int x3, int y3, int xt, int yt)

{

int nx1,nx2,nx3,ny1,ny2,ny3;

nx1=x1+xt;

ny1=y1+yt;

nx2=x2+xt;

ny2=y2+yt;

nx3=x3+xt;

ny3=y3+yt;

line(nx1,ny1,nx2,ny2);

line(nx2,ny2,nx3,ny3);

line(nx3,ny3,nx1,ny1);

}

void Rotate(int x1,int y1, int x2, int y2, int x3, int y3, int r)

{

int t;

t=3.14\*r/180;

int nx1,nx2,nx3,ny1,ny2,ny3;

nx1=floor(x1\*cos(t)-y1\*sin(t));

ny1=floor(x1\*sin(t)+y1\*cos(t));

nx2=floor(x2\*cos(t)-y2\*sin(t));

ny2=floor(x2\*sin(t)+y2\*cos(t));

nx3=floor(x3\*cos(t)-y3\*sin(t));

ny3=floor(x3\*sin(t)+y3\*cos(t));

line(nx1,ny1,nx2,ny2);

line(nx2,ny2,nx3,ny3);

line(nx3,ny3,nx1,ny1);

}

void scaling(int x1,int y1, int x2, int y2, int x3, int y3,int sx,int sy)

{

int nx1,nx2,nx3,ny1,ny2,ny3;

nx1=x1\*sx;

ny1=y1\*sy;

nx2=x2\*sx;

ny2=y2\*sy;

nx3=x3\*sx;

ny3=y3\*sy;

line(nx1,ny1,nx2,ny2);

line(nx2,ny2,nx3,ny3);

line(nx3,ny3,nx1,ny1);

}

void reflection(int x1,int y1,int x2,int y2, int x3, int y3){

// Draw the graph

line(getmaxx() / 2, 0, getmaxx() / 2,

getmaxy());

line(0, getmaxy() / 2, getmaxx(),

getmaxy() / 2);

// Object initially at 2nd quadrant

printf("Before Reflection Object"

" in 2nd Quadrant");

// Set the color

setcolor(14);

line(x1, y1, x2, y2);

line(x2, y2, x3, y3);

line(x3, y3, x1, y1);

getch();

// After reflection

printf("\nAfter Reflection");

// Reflection along origin i.e.,

// in 4th quadrant

setcolor(4);

line(getmaxx() - x1, getmaxy() - y1,

getmaxx() - x2, getmaxy() - y2);

line(getmaxx() - x2, getmaxy() - y2,

getmaxx() - x3, getmaxy() - y3);

line(getmaxx() - x3, getmaxy() - y3,

getmaxx() - x1, getmaxy() - y1);

// Reflection along x-axis i.e.,

// in 1st quadrant

setcolor(3);

line(getmaxx() - x1, y1,

getmaxx() - x2, y2);

line(getmaxx() - x2, y2,

getmaxx() - x3, y3);

line(getmaxx() - x3, y3,

getmaxx() - x1, y1);

// Reflection along y-axis i.e.,

// in 3rd quadrant

setcolor(2);

line(x1, getmaxy() - y1, x2,

getmaxy() - y2);

line(x2, getmaxy() - y2, x3,

getmaxy() - y3);

line(x3, getmaxy() - y3, x1,

getmaxy() - y1);

getch();

}

int main()

{

int x1,y1, x2,y2, x3,y3,ch;

int sx,sy;

int xt,yt;

int r;

int gd=DETECT, gm;

initgraph (&gd, &gm,("C:\\TURBOC3\\BGI"));

printf("Enter triangle vertices: ");

scanf("%d%d%d%d%d%d",&x1,&y1,&x2,&y2,&x3,&y3);

fflush(stdin);

line(x1,y1,x2,y2);

line(x2,y2,x3,y3);

line(x3,y3,x1,y1);

printf("\nEnter your choice for \n1-Translation\n2-Rotation\n3-Scaling\n4-

Reflection\n5-Exit :");

scanf("%d", &ch);

switch(ch)

{

case 1: // for translation

{

printf("\nEnter xt and yt: ");

scanf("%d%d",&xt,&yt);

translate(x1, y1, x2, y2, x3, y3,xt,yt);

}

break;

case 2: //for rotation

{

printf("\nEnter angle of rotation: ");

scanf("%d",&r);

Rotate(x1, y1, x2, y2, x3, y3,r);

}

break;

case 3: //for scaling

{

printf("\nEnter scaling values: ");

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

scaling(x1, y1, x2, y2, x3, y3,sx,sy);

}

break;

case 4:

{

reflection(x1,y1,x2,y2,x3,y3);

}

break;

case 5:

{

break;

}

default:

{

printf("invalid choice...");

}

}

delay(100);

getch();

return 0;

}

**Output:**



