**TRIBHUVAN UNIVERSITY**

Faculty of Humanities and

Social Science



Ambikeshwari Campus

Lab Report of Graphic design [and Animation](https://www.itcollegenepal.com/wp-content/uploads/2021/03/CSCS-305-Computer-Graphics-and-Animation-5th-SEM.pdf)

Submitted By : Submitted To:

Apar Singh Chaudhary Mr. Nabraj Basel

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Index Sheet |  |
|  |  |  | |
| **Experiment** |  | **Title** | |
| **No :** |  |  |  |
| **1** | Write a C program to draw a Line by using build in function line( ). | | |
|  |  | | |
| **2** | Write a C program for DDA Line Drawing Algorithm. | | |
|  |  | | |
| **3** | Write a C program to implement Bresenham's Line Drawing Algorithm for drawing | | |
|  | a line segment between two given endpoints A(x1, y1) and B(x2, y2). | | |
|  |  | | |
| **4** | Write a C program to draw a circle using Bresenham's Circle Drawing Algorithm. | | |
|  |  | | |
| **5** | Write a C program for Mid-Point Ellipse Drawing Algorithm. | | |
|  |  | | |
| **6** | Write a C program for scan line polygon fill algorithm. | | |
|  |  | | |
| **7** | Write a C program to implement 2D translation of a line. | | |
|  |  | | |
| **8** | Write a C program to implement 2D rotation of a line. | | |
|  |  | | |
| **9** | Write a C program for 2D scaling of a line. | | |
|  |  | | |
| **10** | Write a C program for 2D reflection of a triangle about x-axis. | | |
|  |  | | |
| **11** | Write a C program to implement 3D translation. | | |
|  |  | | |
| **12** | Write a C program to implement 3D scaling. | | |
|  |  |  |  |

**Experiment No. 1**

**Objective:**

Write a C program to draw a Line by using build in function line( ).

**Technology:**

Windows OS, DEV C++

**Theory and concept:**

Line function is used to draw a line from a point(x1,y1) to point(x2,y2) i.e. (x1,y1) and (x2,y2) are end points of the line.

**Program:**

#include <stdio.h>

#include <graphics.h>

#include <conio.h>

#include <dos.h>

main( ){

printf("Mr. Arya Singh");

int gd = DETECT, gm;

initgraph(&gd, &gm, "");

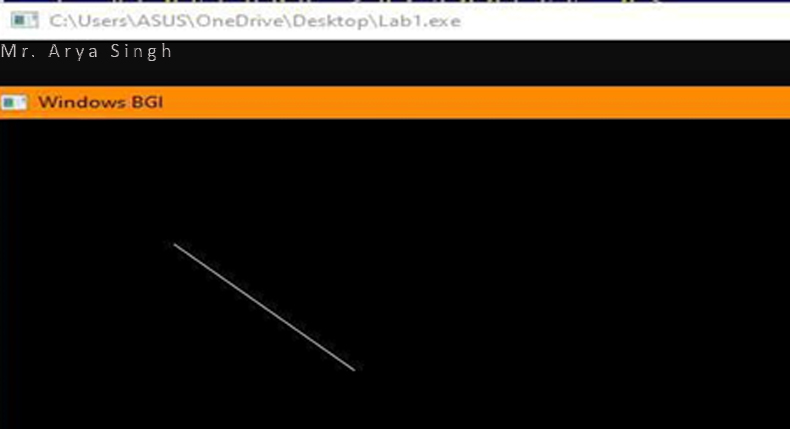
line(100,100,200,200);

getch( );

closegraph( );

}

**Input/Output:**



**Conclusion:** So after doing the above study we have understood the concept of build in function line( ).

**Experiment No. 2**

**Objective:** Write a C program for DDA Line Drawing Algorithm.

**Technology:** Windows OS, DEV C++

**Theory and concept:** DDA stands for Digital Differential Analyzer. It is an incremental method of scan conversion of line. The DDA generates lines from their differential equations. The equation of a straight line is



The DDA works on the principle that we simultaneously increment x and y by small steps proportional to the first derivatives of x and y. In this case of a straight line, the first derivatives are constant and are proportional to ∆x and ∆y. Therefore, we could generate a line by incrementing x and y by ϵ ∆x and ϵ ∆y, where ϵ is some small quantity.

**Program:**

#include <graphics.h>

#include <stdio.h>

#include <math.h>

#include <dos.h>

int main( ){

float x,y,x1,y1,x2,y2,dx,dy,step;

int i, gd=DETECT, gm;

initgraph(&gd,&gm,"");

printf("Mr. Arya Singh\n");

printf("Enter the value of x1 and y1 : ");

scanf("%f%f",&x1,&y1);

printf("Enter the value of x2 and y2: ");

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

dx=abs(x2-x1);

dy=abs(y2-y1);

if(dx>=dy)

step=dx;

else

step=dy;

dx=dx/step;

dy=dy/step;

x=x1;

y=y1;

i=1;

while(i<=step){

putpixel(x,y,5);

x=x+dx;

y=y+dy;

i=i+1;

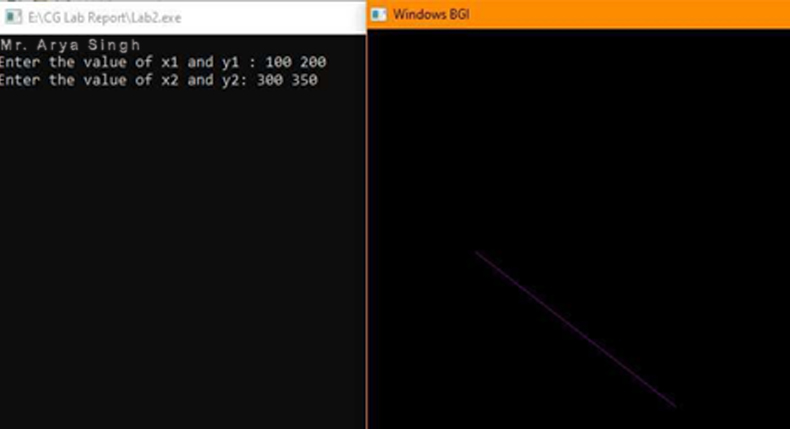
delay(100);

}

closegraph( );

return 0;

}

**Input/Output:**

**Conclusion:**

So after doing the above study we have understood the concept of DDA Line Drawing Algorithm.

**Experiment No. 3**

**Objective:** Write a C program to implement Bresenham's Line Drawing Algorithm for drawing a line segment between two given endpoints A(x1, y1) and B(x2, y2).

**Technology:** Windows OS, DEV C++

**Theory and concept:** This algorithm is used for scan converting a line. It was developed by Bresenham. It is an efficient method because it involves only integer addition, subtractions, and multiplication operations. These operations can be performed very rapidly so lines can be generated quickly. In this method, next pixel selected is that one who has the least distance from true line.

**Program:**

#include <graphics.h>

#include <stdio.h>

#include <math.h>

int main( ){

int x,y,x1,y1,x2,y2,p,dx,dy;

int gd=DETECT,gm;

initgraph(&gd,&gm,"");

printf("Mr. Arya Singh\n");

printf("\nEnter the x-coordinate of the first point ::"); scanf("%d",&x1);

printf("\nEnter the y-coordinate of the first point ::"); scanf("%d",&y1);

printf("\nEnter the x-coordinate of the second point ::"); scanf("%d",&x2);

printf("\nEnter the y-coordinate of the second point ::");

scanf("%d",&y2);

x=x1;

y=y1;

dx=x2-x1;

dy=y2-y1;

putpixel(x,y,2);

p=(2\*dy-dx);

while(x<=x2){

if(p<0){

x=x+1;

p=p+2\*dy;

}else{

x=x+1;

y=y+1;

p=p+(2\*dy) - (2\*dx);

}

putpixel(x,y,7);

}

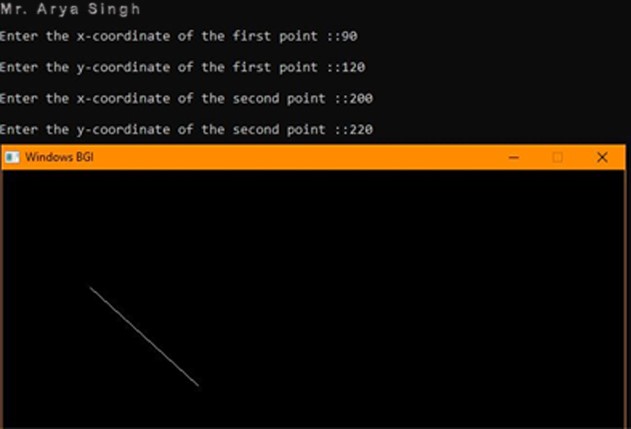
getch( );

return 0;

closegraph( );

}

**Input/Output:**



**Conclusion:** So after doing the above study we have understood the concept of Bresenham's Line Drawing Algorithm for drawing a line segment between two given endpoints.

**Experiment No. 4**

**Objective:** Write a C program to draw a circle using Bresenham's Circle Drawing Algorithm.

**Technology:** Windows OS, DEV C++

**Theory and concept:** Bresenham’s Circle Drawing Algorithm **is a** circle drawing algorithm that selects the nearest pixel position to complete the arc. The unique part of this algorithm is that is uses only integer arithmetic which makes it, significantly, faster than other algorithms using floating point arithmetic in classical processors.

**Program:**

#include <graphics.h>

#include <stdlib.h>

#include <stdio.h>

#include <conio.h>

#include <math.h>

void EightWaySymmetricPlot(int xc,int yc,int x,int y){

putpixel(x+xc,y+yc,WHITE);

putpixel(x+xc,-y+yc,WHITE);

putpixel(-x+xc,-y+yc,WHITE);

putpixel(-x+xc,y+yc,WHITE);

putpixel(y+xc,x+yc,12);

putpixel(y+xc,-x+yc,14);

putpixel(-y+xc,-x+yc,15);

putpixel(-y+xc,x+yc,6);

}void BresenhamCircle(int xc,int yc,int r){

int x=0,y=r,d=3-(2\*r);

EightWaySymmetricPlot(xc,yc,x,y);

while(x<=y){

if(d<=0){

d=d+(4\*x)+6;

}

else{

d=d+(4\*x)-(4\*y)+10;

y=y-1;

}

x=x+1;

EightWaySymmetricPlot(xc,yc,x,y);

}}

int main(void){

printf("Mr. Arya Singh\n");

int xc,yc,r,gdriver = DETECT, gmode, errorcode; initgraph(&gdriver, &gmode, ""); errorcode = graphresult();

if (errorcode != grOk)

{

printf("Graphics error: %s\n", grapherrormsg(errorcode));

printf("Press any key to halt:");

getch( );

exit(1);

}

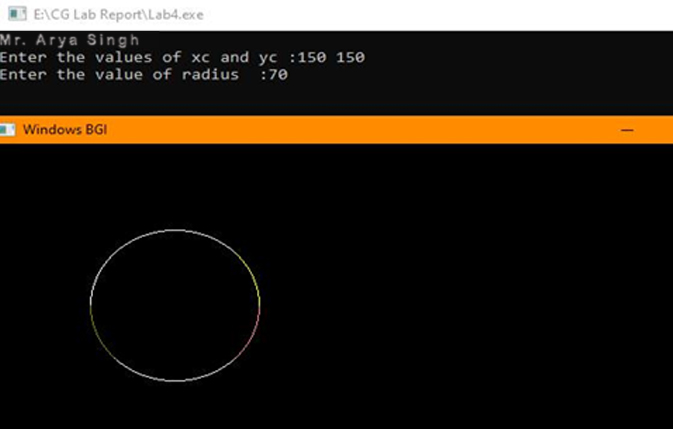
printf("Enter the values of xc and yc :");

scanf("%d%d",&xc,&yc);

printf("Enter the value of radius :");

scanf("%d",&r);

BresenhamCircle(xc,yc,r);

getch( );

closegraph( );

return 0;

}

**Input/Output:**

**Conclusion:** So after doing the above

study we have understood the

concept of Bresenham's

Circle Drawing Algorithm.

**Experiment No. 5**

**Objective:** Write a C program for Mid-Point Ellipse Drawing Algorithm.

**Technology:** Windows OS, DEV C++

**Theory and concept:** Mid-point Ellipse algorithm is used to draw an ellipse in computer graphics. Midpoint ellipse algorithm plots(finds) points of an ellipse on the first quadrant by dividing the quadrant into two regions. Each point(x, y) is then projected into other three quadrants (-x, y), (x, -y), (-x, -y) i.e. it uses 4-way symmetry.

**Program:**

#include<conio.h>

#include<stdio.h>

#include<graphics.h>

int main( ){

int gd=DETECT,gm;

float x,y,xc,yc,rx,ry,pk,pk1;

initgraph(&gd,&gm,"..\\bgi");

printf("Mr. Arya Singh\n");

printf("Enter Center for ellipse : ");

scanf("%f %f",&xc ,&yc);

printf("Enter x-radius and y-radius : ");

scanf("%f %f",&rx ,&ry);

x=0;

y=ry;

pk=(ry\*ry)-(rx\*rx\*ry)+((rx\*rx)/4);

while((2\*x\*ry\*ry)<(2\*y\*rx\*rx)){

if(pk<=0){

x=x+1;

pk1=pk+(2\*ry\*ry\*x)+(ry\*ry);

}else{

x=x+1;

y=y-1;

pk1=pk+(2\*ry\*ry\*x)-(2\*rx\*rx\*y)+(ry\*ry);

}

pk=pk1;

putpixel(xc+x,yc+y,2);

putpixel(xc-x,yc+y,2);

putpixel(xc+x,yc-y,2);

putpixel(xc-x,yc-y,2);

}

pk=((x+0.5)\*(x+0.5)\*ry\*ry)+((y-1)\*(y-1)\*rx\*rx)-(rx\*rx\*ry\*ry);

while(y>0){

if(pk>0){

y=y-1;

pk1=pk-(2\*rx\*rx\*y)+(rx\*rx);

}else{

x=x+1;

y=y-1;

pk1=pk+(2\*ry\*ry\*x)-(2\*rx\*rx\*y)+(rx\*rx);

}

pk=pk1;

putpixel(xc+x,yc+y,2);

putpixel(xc-x,yc+y,2);

putpixel(xc+x,yc-y,2);

putpixel(xc-x,yc-y,2);

}

line(xc+rx,yc,xc-rx,yc);

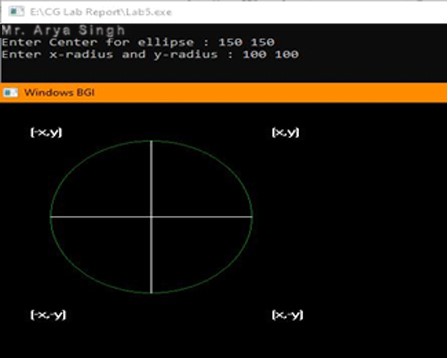
line(xc,yc+ry,xc,yc-ry);

outtextxy(xc+(1.2\*rx),yc-(1.2\*ry),"(x,y)");

outtextxy(xc-(1.2\*rx),yc+(1.2\*ry),"(-x,-y)");

outtextxy(xc+(1.2\*rx),yc+(1.2\*ry),"(x,-y)");

outtextxy(xc-(1.2\*rx),yc-(1.2\*ry),"(-x,y)");

getch( );

**Input/Output:**

**Conclusion:** So after doing the above study we have understood the concept of Mid-Point Ellipse Drawing Algorithm.

**Experiment No. 6**

**Objective:** Write a C program for scan line polygon fill algorithm.

**Technology:** Windows OS, DEV C++

**Theory and concept:** This algorithm lines interior points of a polygon on the scan line and these points are done on or off according to requirement. The polygon is filled with various colors by coloring various pixels.

**Program:**

#include <stdio.h>

#include <conio.h>

#include <graphics.h>

main( ){

int n,i,j,k,gd,gm,dy,dx;

int x,y,temp;

int a[20][2],xi[20];

float slope[20];

printf("\n\n\tMr. Arya Singh");

printf("\n\n\tEnter the no. of edges of polygon : ");

scanf("%d",&n);

printf("\n\n\tEnter the cordinates of polygon :\n\n\n ");

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

printf("\tX%d Y%d : ",i,i);

scanf("%d %d",&a[i][0],&a[i][1]);

}

a[n][0]=a[0][0];

a[n][1]=a[0][1];

detectgraph(&gd,&gm);

initgraph(&gd,&gm," ");

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

line(a[i][0],a[i][1],a[i+1][0],a[i+1][1]);

}

getch( );

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

dy=a[i+1][1]-a[i][1];

dx=a[i+1][0]-a[i][0];

if(dy==0) slope[i]=1.0;

if(dx==0) slope[i]=0.0;

if((dy!=0)&&(dx!=0)){

slope[i]=(float) dx/dy;

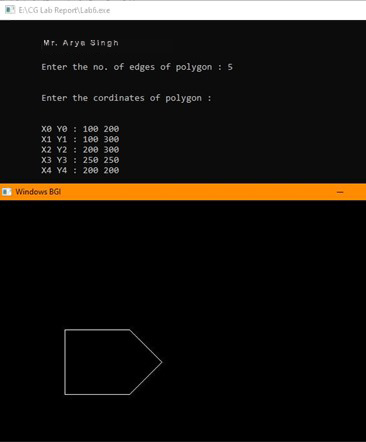
}}

for(y=0;y< 480;y++){

k=0;

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

if( ((a[i][1]<=y)&&(a[i+1][1]>y))||

((a[i][1]>y)&&(a[i+1][1]<=y))){

xi[k]=(int)(a[i][0]+slope[i]\*(y-a[i][1]));

k++;

}

}

for(j=0;j<k-1;j++) for(i=0;i<k-1;i++){

if(xi[i]>xi[i+1]){

temp=xi[i];

xi[i]=xi[i+1];

xi[i+1]=temp;

}

}

setcolor(35);

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

line(xi[i],y,xi[i+1]+1,y);

getch( );

}}}

**Input/Output:**

**Conclusion:**

So after doing the above study we have understood the concept of scan line polygon fill algorithm.

**Experiment No. 7**

**Objective:** Write a C program to implement 2D translation of a line.

**Technology:** Windows OS, DEV C++

**Theory and concept:** 2D Translation is a process of moving an object from one position to another in a two dimensional plane.

**Program:**

#include<stdio.h>

#include <graphics.h> #include <stdlib.h> #include <conio.h> int main( ){

int gdriver = DETECT, gmode, errorcode;

int xmax, ymax,x1,y1,x2,y2,tx,ty; initgraph(&gdriver, &gmode, " "); printf("Mr. Arya Singh\n");

printf("Enter the X1 coordinate:\n"); scanf("%d",&x1);

printf("Enter the Y1 coordinate:\n"); scanf("%d",&y1);

printf("Enter the X2 coordinate:\n"); scanf("%d",&x2);

printf("Enter the y2 coordinate:\n"); scanf("%d",&y2);

line(x1,y1,x2,y2);

printf("Enter the translation vector:\n");

printf("tx:");

scanf("%d",&tx);

printf("ty:");

scanf("%d",&ty);

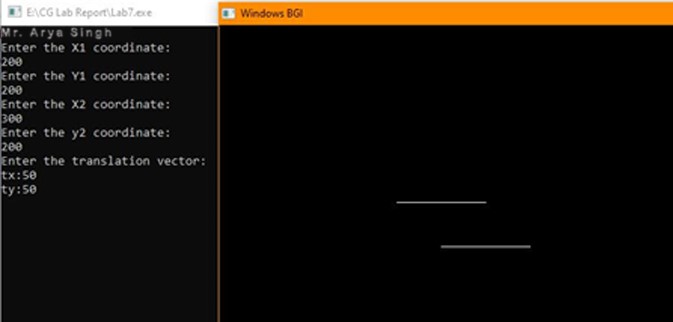
line(x1+tx,y1+ty,x2+tx,y2+ty);

getch( );

closegraph( );

return 0;

}



**Conclusion:**

So after doing the above study we have understood the concept of 2D Translation.

**Experiment No. 8**

**Objective:** Write a C program to implement 2D rotation of a line.

**Technology:** Windows OS, DEV C++

**Theory and concept:** 2D Rotation is a process of rotating an object with respect to an angle in a two dimensional plane. Consider a point object O has to be rotated from one angle to another in a 2D plane.

**Program:**

#include<stdio.h>

#include<graphics.h>

#include<math.h>

int main( ){

int gd=DETECT, r, gm, d, x1, y1, x2, y2, xn1, yn1, xn2, yn2;

float ra, si, co;

initgraph(&gd, &gm, " ");

printf("Mr. Arya Singh\n");

printf("Enter the value of X1 and Y1: ");

scanf("%d %d", &x1, &y1);

printf("Enter the value of X2 and Y2: ");

scanf("%d %d", &x2, &y2);

line(x1, y1, x2, y2);

printf("Enter the degree of rotation: ");

scanf("%d", &d);

//Starting point would be same

xn1 = x1;

yn1 = y1;

r = x2-x1;

ra = 0.0175 \* d;

si = sin(ra);

co = cos(ra);

xn2 = x1 + r\*co + 1;

yn2 = y1 + r\*si + 1;

line(xn1, yn1, xn2, yn2);

getch( );

closegraph( );

return 0;

}

**Input/Output:**

**Conclusion:**

So after doing the above study we have understood the concept of 2D rotation.

**Experiment No. 9**

**Objective:** Write a C program for 2D scaling of a line.

**Technology:** Windows OS, DEV C++

**Theory and concept:** Scaling is the transformation that is used to change the object's size. The operation is carried out for polygon by multiplying the coordinate value(X, Y) of each vertex with the scaling factors.

**Program:**

#include<stdio.h>

#include <conio.h>

#include <graphics.h>

#include <math.h>

int main( ){

int gd=DETECT,gm;

float x1,y1,x2,y2,sx,sy,x3,y3,x4,y4;

initgraph(&gd,&gm," ");

printf("Mr. Arya Singh\n");

printf("Enter the starting point coordinates:");

scanf("%f %f",&x1,&y1);

printf("Enter the ending point coordinates:");

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

printf("Enter scaling factors sx,sy: ");

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

setcolor(5);

line(x1,y1,x2,y2);

outtextxy(x2+2,y2+2,"Original line");

x3=x1\*sx;

y3=y1\*sy;

x4=x2\*sx;

y4=y2\*sy;

setcolor(7);

line(x3,y3,x4,y4);

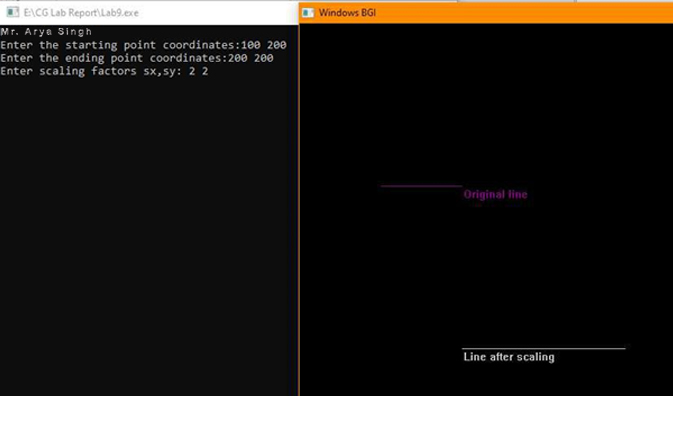
outtextxy(x3+2,y3+2,"Line after scaling");

getch( );

closegraph( );

return 0;

}

 **Input/Output:**

**Conclusion:**

So after doing the above study we have understood the concept of 2D scaling.

**Experiment No. 1****0**

**Objective:** Write a C program for 2D reflection of a triangle about x-axis.

**Technology:** Windows OS, DEV C++

**Theory and concept:** Reflection is a kind of rotation where the angle of rotation is 180 degree. The reflected object is always formed on the other side of mirror. The size of reflected object is same as the size of original object.

**Program:**

#include<stdio.h>

#include<conio.h>

#include<graphics.h>

#include<math.h>

void disp(float v[][3]){

float xmax, ymax ;

xmax = getmaxx()/2 ;

ymax = getmaxy()/2 ;

int i = 0;

while(i<2){

line(xmax+v[i][0], ymax-v[i][1], xmax+v[i+1][0], ymax-v[i+1][1]);

i++ ;

}

i = 2 ;

line(xmax+v[i][0],ymax-v[i][1],xmax+v[0][0],ymax-v[0][1]);

setcolor(BLUE);

line(0,ymax,xmax\*2,ymax);

line(xmax,0,xmax,ymax\*2);

setcolor(WHITE);

}

void multiply (float b[][3],float v[][3],float a[][3]){

int i,j,k;

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

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

a[i][j]=0;

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

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

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

a[i][j] = a[i][j] + (v[i][k] \* b[k][j]);

}}

void reflect(float v[][3]){

float b[10][3],a[10][3];

int i=0, j;

cleardevice();

disp(v);

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

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

b[i][j]=0;

if(i==j)

b[i][j]=1;

}

b[1][1]=-1;

multiply(b,v,a);

setcolor(YELLOW);

disp(a);

}

int main( ){

int gd=DETECT, gm ;

float v[10][3] ;

initgraph(&gd, &gm,"" ) ;

printf("Mr. Arya Singh\n");

printf("Enter the vertex coordinate of triangle : \n");

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

printf("Enter the coordinate v%d :\n", i+1);

scanf("%f%f", &v[i][0], &v[i][1]);

v[i][2] = 1;

}

cleardevice();

setcolor(BLACK);

disp(v);

reflect(v);

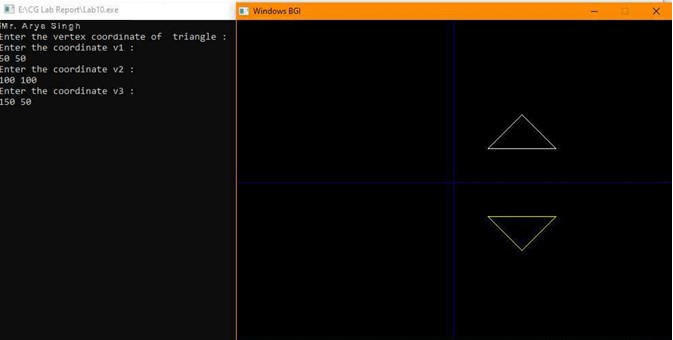
getch( );

closegraph( );

return 0;

}

**Input/Output:**



**Conclusion:**

So after doing the above study we have understood the concept of 2D reflection.

**Expriment No: 11**

**Objective:** Write a C program to implement 3D translation.

**Technology:** Windows OS, DEV C++

**Theory and concept:** 3D Translation is a process of moving an object from one position to another in a three dimensional plane.

**Program:**

#include<stdio.h>

#include<conio.h>

#include<graphics.h>

#include<math.h>

int maxx, maxy, midx, midy;

void axis( ){

getch( );

cleardevice( );

line(midx,0,midx,maxy);

line(0,midy,maxx,midy);

}

int main( ){

int gd,gm,x,y,z,ang,x1,x2,y1,y2;

detectgraph(&gd,&gm);

initgraph(&gd,&gm,"");

maxx=getmaxx( );

maxy=getmaxy( );

midx=maxx/2;

midy=maxy/2;

outtextxy(100,100,"ORIGINAL OBJECT");

line(midx,0,midx,maxy);

line(0,midy,maxx,midy);

bar3d(midx+100,midy-20,midx+60,midy-90,20,5);

axis( );

outtextxy(100,20,"TRANSLATION");

printf("Mr. Arya Singh\n");

printf("\n\n Enter the Translation vector: ");

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

bar3d(midx+100,midy-20,midx+60,midy-90,20,5);

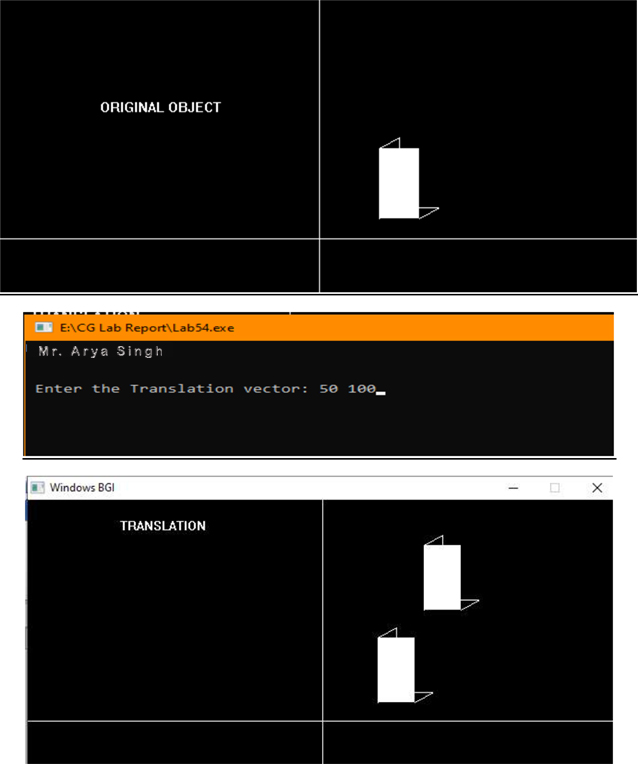
bar3d(midx+(x+100),midy-(y+20),midx+(x+60),midy-(y+90),20,5);

axis( );

closegraph( );

return 0;

}

****

**Conclusion:**

So after doing the above study we have understood the concept of 3D translation.

**Experiment No. 12**

**Objective:** Write a C program to implement 3D scaling.

**Technology:** Windows OS, DEV C++

**Theory and concept:** Scaling is a process of modifying or altering the size of objects. Scaling may be used to increase or reduce the size of object. Scaling subjects the coordinate points of the original object to change.

**Program:**

#include<stdio.h>

#include<conio.h>

#include<graphics.h>

#include<math.h>

int maxx,maxy,midx,midy;

void axis( ){

getch( );

cleardevice( );

line(midx,0,midx,maxy);

line(0,midy,maxx,midy);

}

int main( ){

int gd,gm,x,y,z,ang,x1,x2,y1,y2;

detectgraph(&gd,&gm);

initgraph(&gd,&gm,"");

maxx=getmaxx( );

maxy=getmaxy( );

midx=maxx/2;

midy=maxy/2;

outtextxy(100,100,"ORIGINAL OBJECT");

line(midx,0,midx,maxy);

line(0,midy,maxx,midy);

bar3d(midx+100,midy-20,midx+60,midy-90,20,5);

axis( );

outtextxy(100,20,"SCALING");

printf("Mr. Arya Singh\n");

printf("\n Enter the Scaling Factor: ");

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

bar3d(midx+100,midy-20,midx+60,midy-90,20,5);

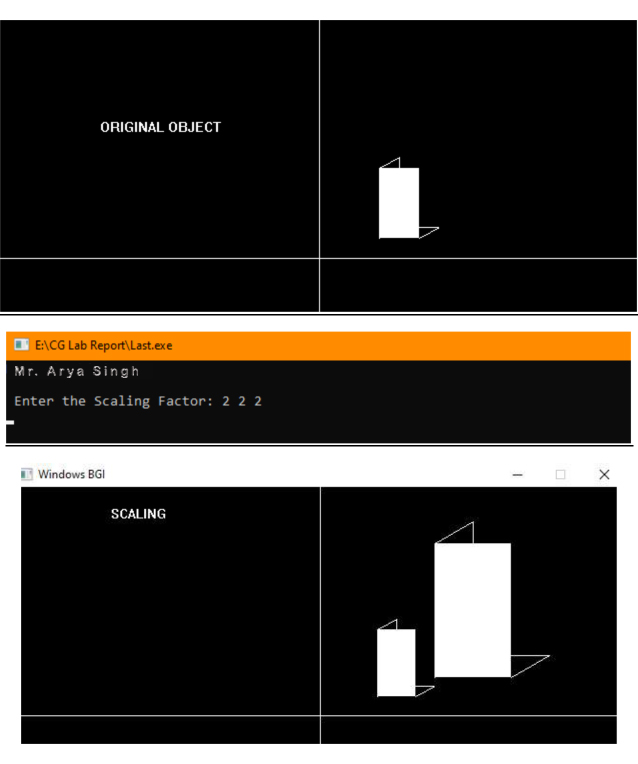
bar3d(midx+(x\*100),midy-(y\*20),midx+(x\*60),midy-(y\*90),20\*z,5);

axis( );

closegraph( );

return 0;

}



**Conclusion:**

So after doing the above study we have understood the concept of 3D scaling.