Code for DDA Line Algorithm:

#include <graphics.h>

#include <stdio.h>

#include <math.h>

#include <dos.h>

int main ( )

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

inti;intgd = DETECT;intgm ;

initgraph (&gd , &gm ,"c:\\turbo");

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 ;

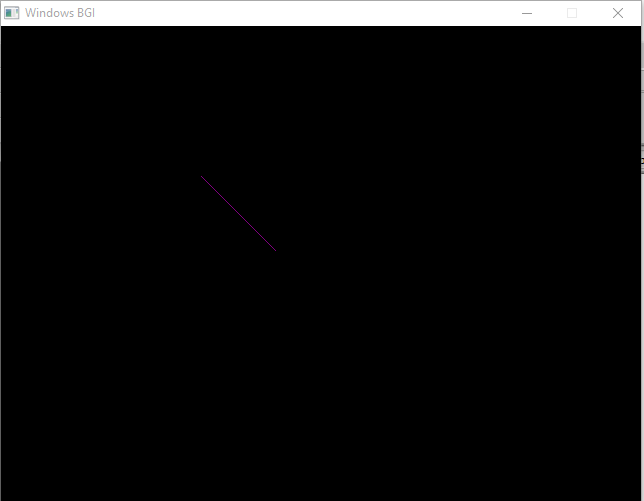
y =y +dy ;

i =i +1 ;

delay( 100);

}closegraph();

}

******

***C CODE FOR BRESENHAM’S LINE:***

#include<stdio.h>

#include<graphics.h>

void drawline( int x0 , int y0 , int x1,int y1)

{

int dx , dy , p , x , y ;

dx =x1 -x0 ;

dy =y1 -y0 ;

x= x0 ;

y= y0 ;

p= 2\* dy - dx ;

while(x < x1 )

{

if ( p>= 0)

{

putpixel ( x, y, 7 );

y = y+ 1;

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

}

else

{

putpixel ( x, y, 7 );

p = p+ 2\* dy ;

}

x =x +1 ;

}

}

int main ()

{

int gdriver=DETECT , gmode, x0=100,y0=100,x1=200,y1=200;

initgraph (& gdriver, & gmode," ");

printf( "Enter co-ordinates of first point: ");

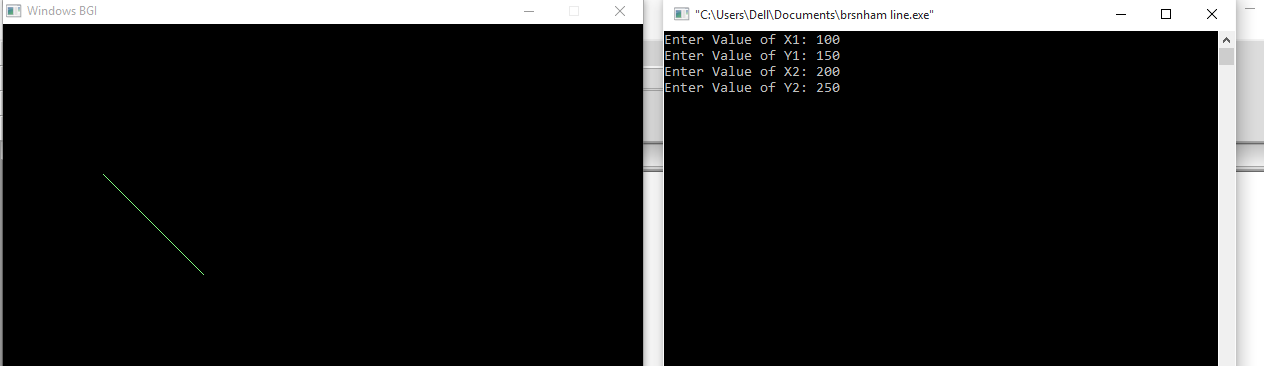
printf( "Enter co-ordinates of second point : ");

drawline( x0 , y0 , x1 , y1 );

return 0 ;

}

OUTPUT:



***C CODE FOR BRESENHAM’S ALGORITHM:(CIRCLE)***

#include<stdio.h>

#include<conio.h>

#include<graphics.h>

#include<math.h>

void cplot(int,int,int,int);

int main()

{

int gd=DETECT,gm;

int x,y,p,xc,yc,r;

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

cleardevice();

printf("x,y,r : ");

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

x=0;y=r;

p=1-r;

cplot(xc,yc,x,y);

while(x<y)

{

x++;

if(p<0)

p+=2\*x+1;

else

{y--;

p+=2\*(x-y)+1;

}cplot(xc,yc,x,y);

}

getch();

}

void cplot(int xc,int yc,int x,int y)

{

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

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

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

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

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

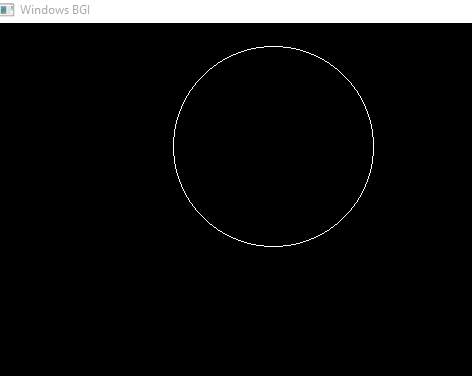
putpixel(xc-y,yc+x,15);

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

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

}

OUTPUT:

******

***C CODE FOR MIDPOINT ALGORITHM:(CIRCLE)***

*#include<stdio.h>*

*#include<conio.h>*

*#include<graphics.h>*

*void pixel(int xc,int yc,int x,int y);*

*int main()*

*{*

*int gd=DETECT,gm,xc,yc,r,x,y,Pk;*

*initgraph(&gd,&gm,"c:\\turboc3\\bgi ");*

*printf("\*\*\* Mid-Point Subdivision algorithm of circle \*\*\*\n");*

*printf("Enter the value of Xc\t");*

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

*printf("Enter the value of Yc \t");*

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

*printf("Enter the Radius of circle\t");*

*scanf("%d",&r);*

*x=0;*

*y=r;*

*Pk=1-r;*

*pixel(xc,yc,x,y);*

*while(x<y)*

*{*

*if(Pk<0)*

*{*

*x=x+1;*

*Pk=Pk+(2\*x)+1;*

*}*

*else*

*{*

*x=x+1;*

*y=y-1;*

*Pk=Pk+(2\*x)-(2\*y)+1;*

*}*

*pixel(xc,yc,x,y);*

*}*

*getch();*

*closegraph();*

*}*

*void pixel(int xc,int yc,int x,int y)*

*{*

*putpixel(xc+x,yc+y,7);*

*putpixel(xc+y,yc+x,7);*

*putpixel(xc-y,yc+x,7);*

*putpixel(xc-x,yc+y,7);*

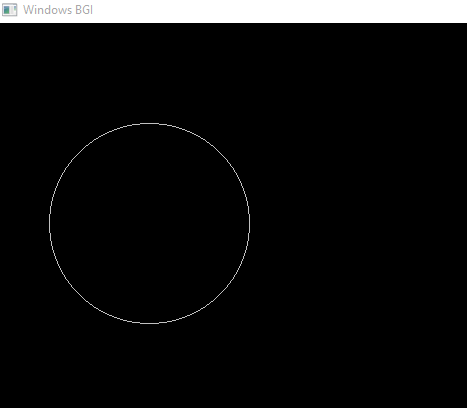
*putpixel(xc-x,yc-y,7);*

*putpixel(xc-y,yc-x,7);*

*putpixel(xc+y,yc-x,7);*

*putpixel(xc+x,yc-y,7)*

***Output:***

**

Code for Ellipse:

#include<stdio.h>

#include<conio.h>

#include<graphics.h>

void ellipse(intxc,intyc,intrx,intry)

{intgm=DETECT,gd;

int x, y, p;

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

x=0;

y=ry;

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

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

{

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

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

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

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

if(p<0)

{

x=x+1;

p=p+(2\*ry\*ry\*x)+(ry\*ry);

}

else {

x=x+1;

y=y-1;

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

}

}

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

while(y>=0)

{

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

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

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

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

if(p>0) {

y=y-1;

p=p-(2\*rx\*rx\*y)+(rx\*rx); }

else

{

y=y-1;

x=x+1;

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

}

}

getch();

closegraph();

}int main()

{intxc,yc,rx,ry;

printf("Enter Xc=");

scanf("%d",&xc);

printf("Enter Yc=");

scanf("%d",&yc);

printf("Enter Rx=");

scanf("%d",&rx);

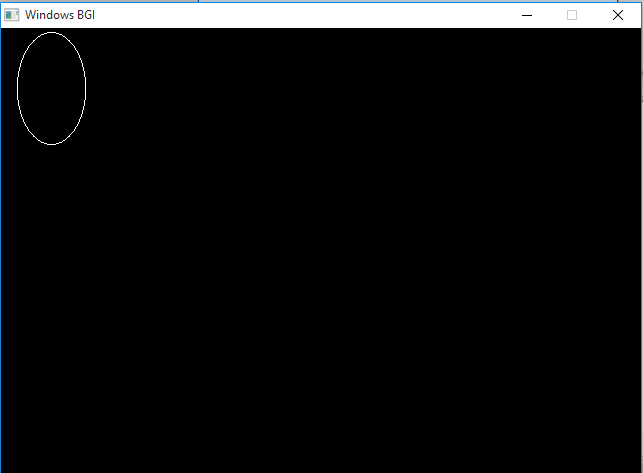
printf("Enter Ry=");

scanf("%d",&ry);

ellipse(xc,yc,rx,ry);

getch();

//output=20,50,20,30



***Code for boundary fill algorithm:***

#include<stdio.h>

#include<graphics.h>

#include<dos.h>

voidboundaryfill(intx,inty,intf\_color,intb\_color)

{

if(getpixel(x,y)!=b\_color&&getpixel(x,y)!=f\_color)

{

putpixel(x,y,f\_color);

boundaryfill(x+1,y,f\_color,b\_color);

boundaryfill(x,y+1,f\_color,b\_color);

boundaryfill(x-1,y,f\_color,b\_color);

boundaryfill(x,y-1,f\_color,b\_color);

}

}

//getpixel(x,y) gives the color of specified pixel

int main()

{

intgm,gd=DETECT,radius;

intx,y;

printf("Enter x and y positions for circle\n");

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

printf("Enter radius of circle\n");

scanf("%d",&radius);

initgraph(&gd,&gm,"c:\\turboc3\\bgi");

circle(x,y,radius);

boundaryfill(x,y,4,15);

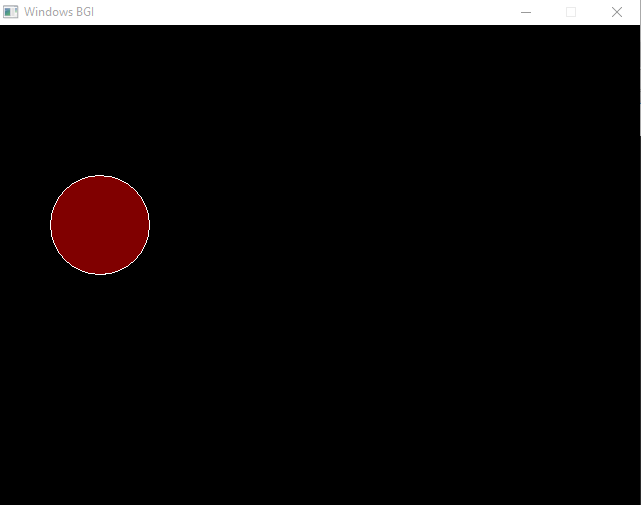
delay(5000);

closegraph();

return 0;

}

Output:



***Code for FloodFill Algorithm:***

#include<stdio.h>

#include<graphics.h>

#include<dos.h>

voidfloodFill(intx,inty,intoldcolor,intnewcolor)

{

if(getpixel(x,y) == oldcolor)

{

putpixel(x,y,newcolor);

floodFill(x+1,y,oldcolor,newcolor);

floodFill(x,y+1,oldcolor,newcolor);

floodFill(x-1,y,oldcolor,newcolor);

floodFill(x,y-1,oldcolor,newcolor);

}

}

//getpixel(x,y) gives the color of specified pixel

int main()

{

intgm,gd=DETECT,radius;

intx,y;

printf("Enter x and y positions for circle\n");

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

printf("Enter radius of circle\n");

scanf("%d",&radius);

initgraph(&gd,&gm,"c:\\turboc3\\bgi");

circle(x,y,radius);

floodFill(x,y,0,15);

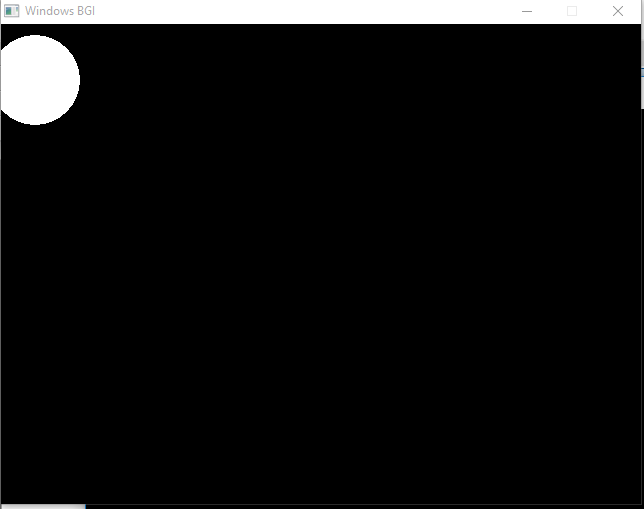
delay(5000);

closegraph();

return 0;

}

OUTPUT:



***Code for Cohen Sutherland :***

#include<stdio.h>

#include<stdlib.h>

#include<math.h>

#include<graphics.h>

#include<dos.h>

typedefstruct coordinate

{

intx,y;

char code[4];

}PT;

voiddrawwindow();

voiddrawline(PT p1,PT p2);

PT setcode(PT p);

int visibility(PT p1,PT p2);

PT resetendpt(PT p1,PT p2);

int main()

{

intgd=DETECT,v,gm;

PT p1,p2,p3,p4,ptemp;

printf("\nEnter x1 and y1\n");

scanf("%d %d",&p1.x,&p1.y);

printf("\nEnter x2 and y2\n");

scanf("%d %d",&p2.x,&p2.y);

initgraph(&gd,&gm,"c:\\turboc3\\bgi");

drawwindow();

delay(500);

drawline(p1,p2);

delay(500);

cleardevice();

delay(500);

p1=setcode(p1);

p2=setcode(p2);

v=visibility(p1,p2);

delay(500);

switch(v) {

case 0: drawwindow();

delay(500);

drawline(p1,p2);

break;

case 1: drawwindow();

delay(500);

break;

case 2: p3=resetendpt(p1,p2);

p4=resetendpt(p2,p1);

drawwindow();

delay(500);

drawline(p3,p4);

break;

}

delay(5000);

closegraph();

}

voiddrawwindow()

{

line(150,100,450,100);

line(450,100,450,350);

line(450,350,150,350);

line(150,350,150,100);

}

voiddrawline(PT p1,PT p2)

{

line(p1.x,p1.y,p2.x,p2.y);

}PT setcode(PT p) //for setting the 4 bit code

{ PT ptemp;

if(p.y<100)

ptemp.code[0]='1'; //Top

else

ptemp.code[0]='0';

if(p.y>350)

ptemp.code[1]='1'; //Bottom

else

ptemp.code[1]='0';

if(p.x>450)

ptemp.code[2]='1'; //Right

else

ptemp.code[2]='0';

if(p.x<150)

ptemp.code[3]='1'; //Left

else

ptemp.code[3]='0';

ptemp.x=p.x;

ptemp.y=p.y;

return(ptemp);

}

int visibility(PT p1,PT p2){

inti,flag=0;

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

{

if((p1.code[i]!='0') || (p2.code[i]!='0'))

flag=1;

}

if(flag==0)

return(0);

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

{

if((p1.code[i]==p2.code[i]) && (p1.code[i]=='1'))

flag='0';

}

if(flag==0)

return(1);

return(2);

}

PT resetendpt(PT p1,PT p2)

{

PT temp;

intx,y,i;

floatm,k;

if(p1.code[3]=='1')

x=150;

if(p1.code[2]=='1')

x=450;

if((p1.code[3]=='1') || (p1.code[2]=='1'))

{

m=(float)(p2.y-p1.y)/(p2.x-p1.x);

k=(p1.y+(m\*(x-p1.x)));

temp.y=k;

temp.x=x;

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

temp.code[i]=p1.code[i];

if(temp.y<=350 &&temp.y>=100)

return (temp);

}

if(p1.code[0]=='1')

y=100;

if(p1.code[1]=='1')

y=350;

if((p1.code[0]=='1') || (p1.code[1]=='1'))

{

m=(float)(p2.y-p1.y)/(p2.x-p1.x);

k=(float)p1.x+(float)(y-p1.y)/m;

temp.x=k;

temp.y=y;

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

temp.code[i]=p1.code[i];

return(temp);

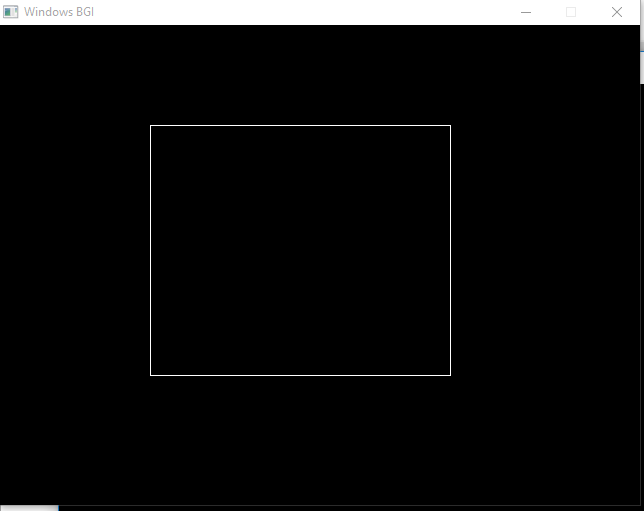
}

else

return(p1);

}

*Output:*



***C CODE FOR DRAWING EMOJI:( SAD)***

#include<stdio.h>

#include<graphics.h>

int main()

{

int gd=DETECT,gm;

initgraph(&gd,&gm,"C://TC//BGI ");

int x=getmaxx();

int y=getmaxy();

setfillstyle(1,YELLOW);

fillellipse(x/2,y/2,70,70);

setfillstyle(1,BLACK);

fillellipse(x/2-30,y/2-25,10,12);

fillellipse(x/2+30,y/2-25,10,12);

setcolor(BLACK);

line(x/2,y/2-10,x/2,y/2+20);

fillellipse(x/2-30,y/2-25,10,12);

arc(x/2,y/2+75,45,135,45);//sad

getch();

closegraph();

}

***Output***



***C CODE FOR DRAWING EMOJI:(HAPPY)***

#include<stdio.h>

#include<graphics.h>

int main()

{

int gd=DETECT,gm;

initgraph(&gd,&gm,"C://TC//BGI ");

int x=getmaxx();

int y=getmaxy();

setfillstyle(1,YELLOW);

fillellipse(x/2,y/2,70,70);

setfillstyle(1,BLACK);

fillellipse(x/2-30,y/2-25,10,12);

fillellipse(x/2+30,y/2-25,10,12);

setcolor(BLACK);

line(x/2,y/2-10,x/2,y/2+20);

fillellipse(x/2-30,y/2-25,10,12);

arc(x/2,y/2,220,320,50);//happy

getch();

closegraph();

}

OUTPUT:



***C CODE FOR DRAWING EMOJI:(CONFUSED)***

#include<stdio.h>

#include<graphics.h>

int main()

{

int gd=DETECT,gm;

initgraph(&gd,&gm,"C://TC//BGI ");

int x=getmaxx();

int y=getmaxy();

setfillstyle(1,YELLOW);

fillellipse(x/2,y/2,70,70);

setfillstyle(1,BLACK);

fillellipse(x/2-30,y/2-25,10,12);

fillellipse(x/2+30,y/2-25,10,12);

setcolor(BLACK);

line(x/2,y/2-10,x/2,y/2+20);

fillellipse(x/2-30,y/2-25,10,12);

line(x/2-30,y/2+40,x/2+30,y/2+40);

getch();

closegraph();

}

******

***C CODE FOR 2D SCALLING:***

*#include<graphics.h>*

*#include<stdlib.h>*

*#include<stdio.h>*

*#include<math.h>*

*int main()*

*{*

*int graphdriver=DETECT,graphmode,errorcode;*

*int i;*

*int x2,y2,x1,y1,x,y;*

*printf("Enter the 2 line end points:");*

*printf("x1,y1,x2,y2");*

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

*initgraph(&graphdriver,&graphmode,"c:\\tc\\bgi");*

*line(x1,y1,x2,y2);*

*printf("Enter scaling co-ordinates ");*

*printf("x,y");*

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

*x1=(x1\*x);*

*y1=(y1\*y);*

*x2=(x2\*x);*

*y2=(y2\*y);*

*printf("Line after scaling");*

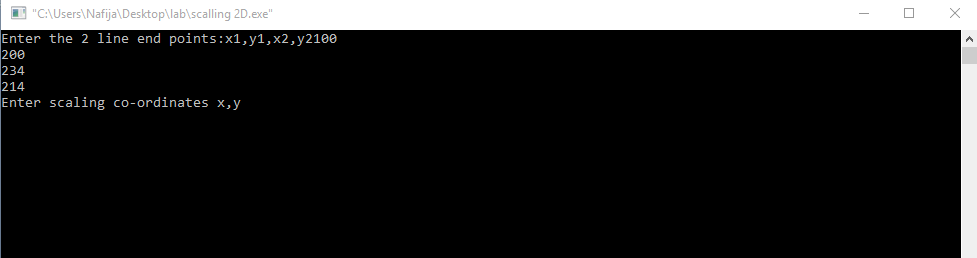
*line(x1,y1,x2,y2);*

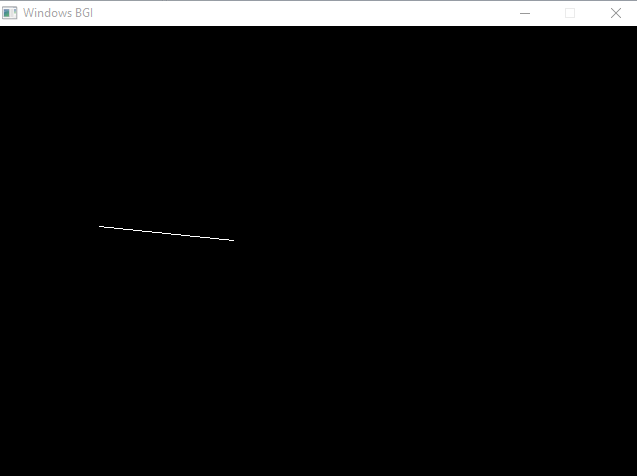
*getch();*

*closegraph();*

*}*

*Input:*

******

******

***C CODE FOR 2D ROTATION****:*

*#include<graphics.h>*

*#include<stdlib.h>*

*#include<stdio.h>*

*#include<math.h>*

*int main()*

*{*

*int graphdriver=DETECT,graphmode,errorcode;*

*int i;*

*int x2,y2,x1,y1,x,y,xn,yn;*

*double r11,r12,r21,r22,th;*

*//clrscr();*

*printf("Enter the 2 line end points:");*

*printf("x1,y1,x2,y2");*

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

*initgraph(&graphdriver,&graphmode,"c:\\tc\\bgi");*

*line(x1,y1,x2,y2);*

*printf("\n\n\n[ Enter the angle");*

*scanf("%lf",&th);*

*r11=cos((th\*3.1428)/180);*

*r12=sin((th\*3.1428)/180);*

*r21=(-sin((th\*3.1428)/180));*

*r22=cos((th\*3.1428)/180);*

*//printf("%lf %lf %lf %lf",r11,r12,r21,r22);*

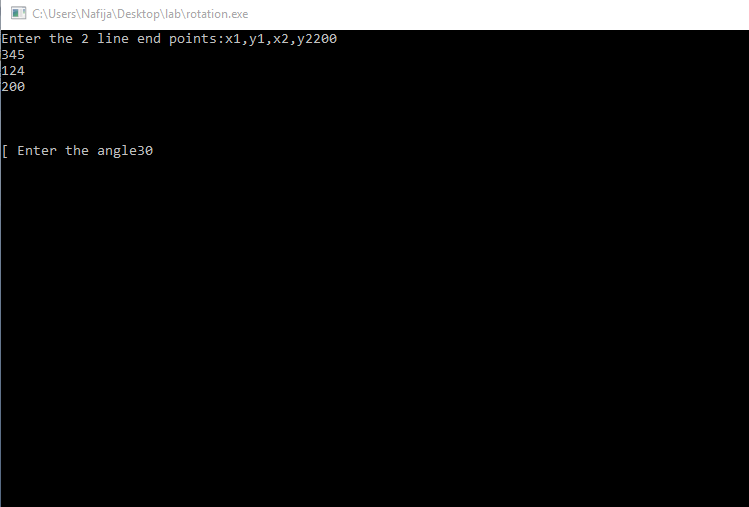
*xn=((x2\*r11)-(y2\*r12));*

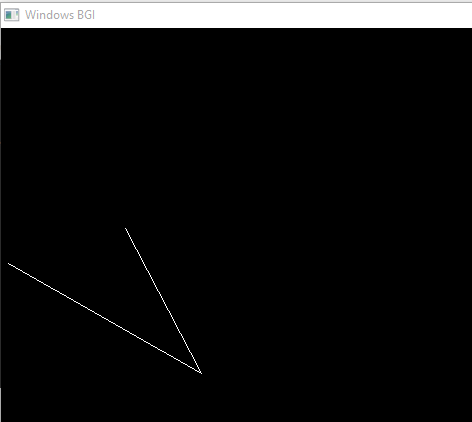
*yn=((x2\*r12)+(y2\*r11));*

*line(x1,y1,xn,yn);*

*getch();*

*closegraph();}*

******

******

***C CODE FOR 2D SHEARING:***

#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;int gd = DETECT;int gm ;

initgraph (& gd , &gm ,"c:\\turbo");

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();

}

***C CODE FOR MOVING OBJECT:***

*#include <stdio.h>*

*#include <conio.h>*

*#include <graphics.h>*

*#include <dos.h>*

*int main() {*

*int gd = DETECT, gm;*

*int i, x, y, flag=0;*

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

*/\* get mid positions in x and y-axis \*/*

*x = getmaxx()/2;*

*y = 30;*

*while (!kbhit()) {*

*if(y >= getmaxy()-30 || y <= 30)*

*flag = !flag;*

*/\* draws thy board \*/*

*setcolor(RED);*

*setfillstyle(SOLID\_FILL, RED);*

*circle(x, y, 30);*

*floodfill(x, y, RED)*

*delay(50);*

*/\* clears screen \*/*

*cleardevice();*

*if(flag){*

*y = y + 5;*

*} else {*

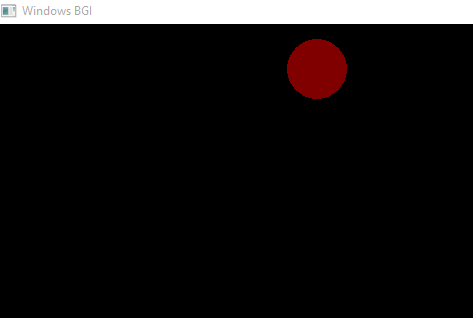
*y = y }*

*getch();*

*closegraph();*

*return 0;*

*}*

******

*}*

***Code for 3D rotation:***

#include <stdio.h>

#include <graphics.h>

#include <math.h>

#include <stdlib.h>

#include <dos.h>

#include <conio.h>

#define ORG -50

double face1[5][2] = {

{ 250, 125 },

{ 350, 125 },

{ 350, 225 },

{ 250, 225 },

{ 250, 125 }

};

double face2[5][2] = {

{ 250+ORG, 125-ORG },

{ 350+ORG, 125-ORG },

{ 350+ORG, 225-ORG },

{ 250+ORG, 225-ORG },

{ 250+ORG, 125-ORG }

};

double angle = 5.0 \* M\_PI / 180;

double midx1, midy1, midx2, midy2;

void rotate (void)

{int i;

for (i=0; i<5; i++)

{

doublexnew, ynew;

xnew = midx1 + (face1[i][0] - midx1) \* cos (angle) -

(face1[i][1] - midy1) \* sin (angle);

ynew = midy1 + (face1[i][0] - midx1) \* sin (angle) +

(face1[i][1] - midy1) \* cos (angle);

face1[i][0] = xnew;

face1[i][1] = ynew;

xnew = midx2 + (face2[i][0] - midx2) \* cos (angle) -

(face2[i][1] - midy2) \* sin (angle);

ynew = midy2 + (face2[i][0] - midx2) \* sin (angle) +

(face2[i][1] - midy2) \* cos (angle);

face2[i][0] = xnew;

face2[i][1] = ynew;

}

cleardevice();

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

{

setcolor(7);

line (face1[i][0], face1[i][1], face1[i+1][0], face1[i+1][1]);

setcolor(8);

line (face2[i][0], face2[i][1], face2[i+1][0], face2[i+1][1]);

setcolor(9);

line (face1[i][0], face1[i][1], face2[i][0], face2[i][1]);

}

delay (125);

}

int main()

{

intgd = DETECT, gm;

midx1 = (face1[0][0] + face1[1][0]) / 2.0;

midy1 = (face1[1][1] + face1[2][1]) / 2.0;

midx2 = (face2[0][0] + face2[1][0]) / 2.0;

midy2 = (face2[1][1] + face2[2][1]) / 2.0;

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

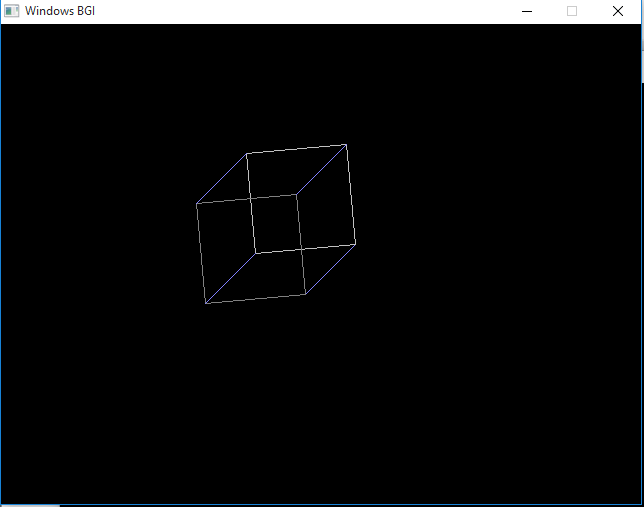
while (!kbhit())

rotate();

closegraph();

}

***Output:***



***C CODE FOR 3D SHEARING:***

#include<stdio.h>

#include<graphics.h>

#include<conio.h>

int main()

{

int gd=DETECT,gm,option,xref,yref;

int i,maxx,maxy,x1,y1,x2,y2,x3,y3,x4,y4,gap=50;

float shx=0.0,shy=0.0;

char str[5];

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

printf("enter the endpoints of the top of the rectangle (x1,y1) & (x2,y2):");

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

printf("enter the endpoints of bottom of the rectangle (x3,y3) & (x4,y4)");

scanf("%d%d%d%d",&x3,&y3,&x4,&y4);

printf(" Enter the axis to shear\n");

printf(" 1 - X axis Shear\n");

printf(" 2 - Y axis shear\n");

scanf("%d",&option );

if(option==1)

{

printf("enter the value for x-axis shear( can be fraction too):");

scanf("%f",&shx);

}

else

{

printf("enter the value for y-axis shear( can be fraction too):");

scanf("%f",&shy);

}

clearviewport();

maxx= getmaxx();

maxy=getmaxy();

line(3,maxy-1,maxx-5,maxy-1);

line(5,5,5,maxy-3);

for( i= 0;i<maxx-5;i=i+gap) // code to display co-ordinates

{

outtextxy(i+3,maxy-7,"|");

itoa(i,str,10);

outtextxy(i,maxy-10,str);

}

for( i= maxy;i>0;i=i-gap)

{

outtextxy(3,i,"-");

itoa(maxy-i,str,10);

outtextxy(9,i,str);

}

setcolor(50); // drawing rectangle using endpoints

line(x1,maxy-y1,x2,maxy-y2);

line(x3,maxy-y3,x4,maxy-y4);

line(x1,maxy-y1,x3,maxy-y3);

line(x2,maxy-y2,x4,maxy-y4);

outtextxy(10,10,"hit any key to see the shearing effect" );

getch();

setcolor(0); // to hide the rectangle drawn

line(x1,maxy-y1,x2,maxy-y2);

line(x3,maxy-y3,x4,maxy-y4);

line(x1,maxy-y1,x3,maxy-y3);

line(x2,maxy-y2,x4,maxy-y4);

setcolor(0); // to redraw the rectangle

if(option==1)

{

// shearing about x axis so only points x1 and x2 need to be recomputed

line(x1+shx\*y1,maxy-y1,x2+shx\*y2,maxy-y2);

line(x3,maxy-y3,x4,maxy-y4);

line(x1+shx\*y1,maxy-y1,x3,maxy-y3);

line(x2+shx\*y2,maxy-y2,x4,maxy-y4);

}else

{ line(x1,maxy-y1,x2,maxy-(y2+shy\*x2));

line(x3,maxy-y3,x4,maxy-(y4+shy\*x4));

line(x1,maxy-y1,x3,maxy-y3);

line(x2,maxy-(y2+shy\*x2),x4,maxy-(y4+shy\*x4));

}

getch();

closegraph();}

INPUT:

