# Introduction to Graphic functions:

A general purpose graphic package provide user with a variety of functions for creating and manipulating pictures. The basic building blocks for pictures are referred to as output primitives. They include points for straight lines, curve, filled area an d other general purpose functions for polygon, circle etc.

**Most commonly used functions:**

1. **Initgraph ():**

The function initializes the graphic system by loading a graphics driver from disk. Then the system into graphics mode. Initgraph() also resets all graphics settings like color, current position etc. to their defaults.

**Syntax:**

void initgraph(int \*driver, int \*mode, char \*path);

\***mode:**  is an integer that specifies the initials graphics mode.

\***drive:** is an integer that specifies the graphics driver to be used.

If the **\*driver= DETECT**, initgraph sets **\*mode** to be highest resolution a variable for the detective driver.

**\*path:** specifies the directory where initgraph function looks for graphics driver**(\*.BGI)** file**.** It first looks in the path specified, then if they are not there, it looks in the current directory. If path is null, the driver files must be in current directory.

1. **Detectgraph() :**

This function determines graphics driver and graphics mode to use by checking the hardware. **Detectgraph()**  detects the system’s graphics adapter and choose the mode that provide the highest resolution for that adapter.

**Syntax:**

void detectgraph (int \*driver, int \*mode);

**\*driver:** is an integer that specifies the graphics driver to be used.

**\*mode:** is an integer that specifies the initial graphics mode.

1. **line():**

This function draws a line in graphics mode between two specified end points. It draws the line using the current color, line setting and style.

**Syntax:**

void line(int x1,int y1, int x2, int y2);

whereas x1, y1 are starting points of line and x2, y2 are ending points of line.

1. **Circle():**

This function draws the circle on the graphics screen with a center given by (x, y) co-ordinates and the radius. This function uses the current drawing color & line style for drawing a circle.

**Syntax:**

void circle(int x, int y, ,int radius);

1. **Rectangle():**

This function draws a rectangle in graphics mode using current line style and color.

**Syntax:**

void rectangle (int left, int top, int right, int bottom);

These first two parameters specify the coordinates of one corner of the rectangle and other two parameters gives the opposite corner.

1. **Ellipse():**

This function draws an elliptical arc on the graphics screen.

**Syntax:**

void ellipse(int x, int y, int stangle, int endangle, int xradius, int yradius);

This ellipse function draws an elliptical arc in the current drawing color with its center (x, y) and horizontal & vertical axes given by (xradius & yradius). The stangle & endangle parameters give the starting & ending angles of arc in decrease.

1. **Arc( ):**

This function draws a circular arc on the graphics screen using current drawing color and line styles.

**Syntax:**

void arc(int x, int y, int stangle, int endangle, int radius);

Where the center of the arc is given by x & y co-ordinates and the radius are provided by the radius parameter. The parameter stangle & endangle specify the starting 7 stopping angle in decrease.

1. **Setcolor():**

This function sets the current drawing color to color, which can range from 0 to getmax color.

**Syntax:**

void setcolor(int color);

1. **Program to make a flag**

#include<iostream.h>

#include<conio.h>

#include<graphics.h>

#include<dos.h>

#include<stdlib.h>

void main()

{

int gd=DETECT,gm;

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

setcolor(5);

line(20,50,20,180);

line(20,50,100,50);

line(20,70,100,70);

line(20,90,100,90);

line(20,110,100,110);

line(100,50,100,110);

circle(60,80,10);

line(52,80,68,80);

line(60,72,60,88);

setfillstyle(SOLID\_FILL,22);

floodfill(21,51,5);

setfillstyle(SOLID\_FILL,7);

floodfill(21,75,5);

setfillstyle(SOLID\_FILL,2);

floodfill(21,95,5);

setfillstyle(SOLID\_FILL,7);

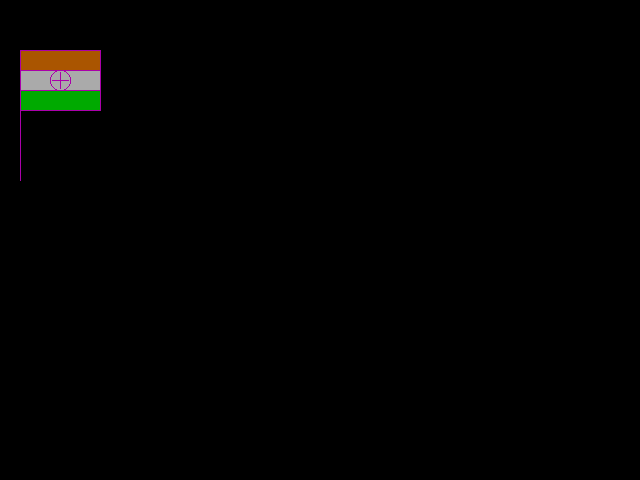
floodfill(81,75,5);

setfillstyle(SOLID\_FILL,7);

floodfill(61,81,5);

}

**Output:**



1. **Program of DDA line algorithm**

#include<iostream.h>

#include<conio.h>

#include<graphics.h>

#include<math.h>

#define round(a)((int)(a+0.5))

void main()

{

int gd=DETECT,gm;

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

float x,y;

int xa,ya,yb,xb,steps,k;

setbkcolor(5);

cout<<"enter starting coordinate";

cin>>xa>>ya;

textcolor(10);

cout<<"enter the ending coordinate";

cin>>xb>>yb;

int dx=xb-xa;

int dy=yb-ya;

float xinc,yinc;

x=xa;

y=ya;

if(abs(dx)>abs(dy))

steps=abs(dx);

else

steps=abs(dy);

xinc=dx/float(steps);

yinc=dy/float(steps);

putpixel(round(x),round(y),10);

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

{

x=x+xinc;

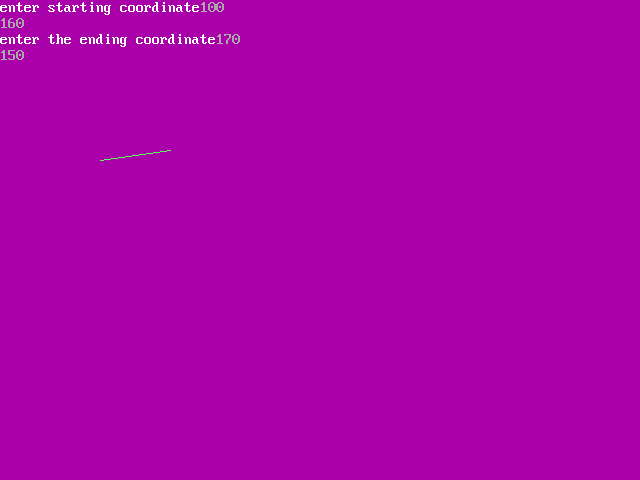
y=y+yinc;

putpixel(round(x),round(y),10);

}

}

**Output:-**

****

1. **Program of Bresenham’s Line algorithm**

#include<iostream.h>

#include<conio.h>

#include<graphics.h>

void main()

{

int gd=DETECT,gm;

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

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

cout<<"\n enter first coordinate(x1,y1)";

cin>>x1>>y1;

cout<<"\n enter second coordinate(x2,y2)";

cin>>x2>>y2;

dx=x2-x1;

dy=y2-y1;

length=dy/dx;

x=x1;

y=y1;

dx=1;

dy=1;

while((x<x2)||(y<=y2))

{

if(length<1)

{

x=x+dx;

y=y+length;

putpixel(x,y,GREEN);

}

if(length>1)

{

y=y+dy;

x=(x/(1/length));

putpixel(x,y,WHITE);

}

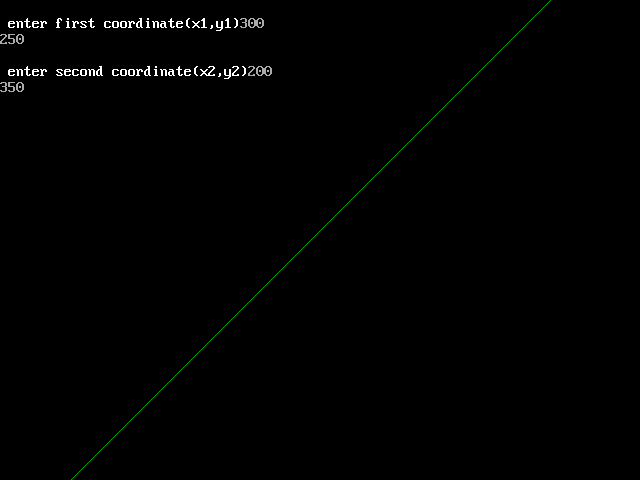
}

getch();

closegraph();

}

**Output:**

****

1. **Program of Bresenham’s circle algorithm**

#include<iostream.h>

#include<conio.h>

#include<graphics.h>

#include<math.h>

#include<dos.h>

void circlebres(int,int,int);

void drawcircle(int,int,int,int);

void main()

{

int gd=DETECT,gm;

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

int xc,yc,r;

cout<<"Enter center coordinates of circle:";

cin>>xc>>yc;

cout<<"Enter the Radius";

cin>>r;

circlebres(xc,yc,r);

getch();

}

void circlebres(int xc,int yc,int r)

{

int x=0,y=r;

int d=3-2\*r;

while(x<y)

{

drawcircle(xc,yc,x,y);

x++;

if(d<0)

{

d=d+4\*x+6;

}

else

{

y--;

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

}

drawcircle(xc,yc,x,y);

delay(50);

}

}

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

{

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

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

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

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

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

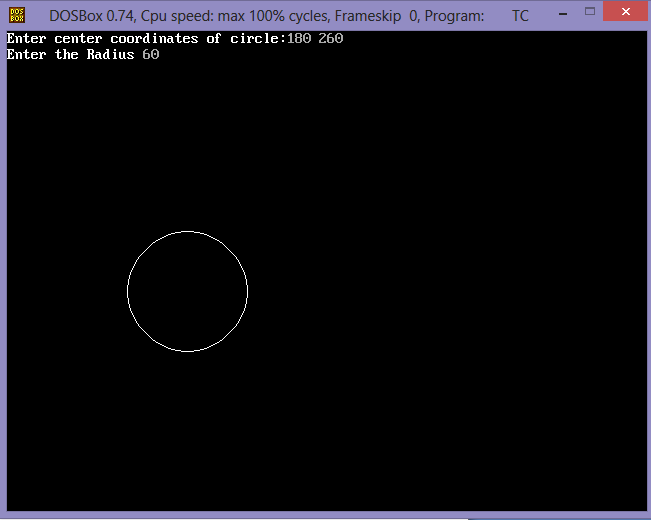
putpixel(xc+y,yc+x,15);

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

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

}

**Output:**



1. **Program of mid point**

#include<iostream.h>

#include<conio.h>

#include<graphics.h>

void main()

{

int gd=DETECT,gm;

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

int x,y,xc,yc,r;

float p;

cout<<"enter values";

cin>>xc>>yc>>r;

x=0;

y=r;

p=(5/4)-r;

putpixel(xc,yc+r,RED);

putpixel(xc,yc-r,RED);

putpixel(xc-r,yc,RED);

putpixel(xc+r,yc,RED);

for(x=0;x<=y;x++)

{

if(p<0)

{

p=p+(2\*x)+3;

}

else

{

p=p+(2\*x)-(2\*y)+5;

y=y-1;

}

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

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

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

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

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

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

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

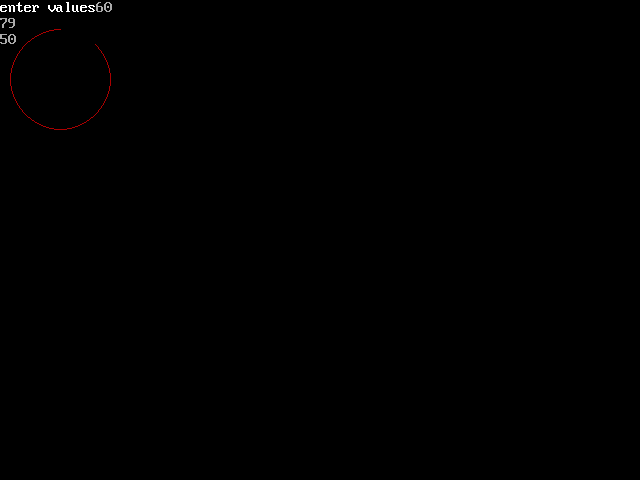
putpixel(xc-y,yc-x,RED);

}

getch();

}

**Output:**



1. **Program of brehseman ellipse algoritham**

#include<iostream.h>

#include<conio.h>

#include<graphics.h>

#include<dos.h>

#include<math.h>

int xc,yc,rx,ry;

void main()

{

int gd=DETECT,gm;

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

void plotpoints(int,int);

int round(float);

cout<<"enter the center point";

cin>>xc>>yc;

cout<<"enter major radius";

cin>>rx;

cout<<"enter minor radius";

cin>>ry;

long int p,px,py,x,y;

long int ry2=ry\*ry;

long int rx2=rx\*rx;

long int ry22=ry2\*2;

long int rx22=rx2\*2;

x=0;

y=ry;

plotpoints(x,y);

p=round(ry2-rx2\*ry+(0.25\*rx2));

px=0;

py=rx22\*y;

while(px<py)

{

x++;

px=px+ry22;

if(p>=0)

{

y--;

py=py-rx22;

}

if(p<0)

p=p+px+ry2;

else

p=p+px+ry2-py;

plotpoints(x,y);

}

p=round(ry2\*(x+0.5)\*(x+0.5)+rx2\*(y-1)\*(y-1)-rx2\*ry2);

while(y>0)

{

y--;

py=py-rx22;

if(p<=0)

{

x++;

px=px+ry22;

}

if(p>=0)

p=p-py+rx2;

plotpoints(x,y);

}

getch();

closegraph();

}

void plotpoints(int x,int y)

{

setlinestyle(0,0,3);

line(xc+x,yc+y,xc+x,yc+y);

line(xc-x,yc+y,xc-x,yc+y);

line(xc+x,yc-y,xc+x,yc-y);

line(xc-x,yc-y,xc-x,yc-y);

}

int round(float x)

{

int y=abs(x);

if(x>(y+0.5))

return(y+1);

else

return(y);

}

Output:



1. **Program of translation**

#include<iostream.h>

#include<conio.h>

#include<graphics.h>

void main()

{

int gd=DETECT,gm;

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

int tx,ty;

int x1=40;

int x2=50;

int y1=60;

int y2=70;

rectangle(x1,x2,y1,y2);

cout<<"enter translation coordinate";

cin>>tx>>ty;

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

getch();

}

**Output:**



1. **Program of 2D Rotation**

#include<iostream.h>

#include<conio.h>

#include<graphics.h>

#include<dos.h>

#include<math.h>

#include<stdlib.h>

void main()

{

int xc=300,yc=300;

int r=100;

int x1,x2,y1,y2;

x1=xc-r;

x2=xc+r;

y1=yc;

int driver=DETECT;

int mode;

initgraph(&driver,&mode,"C:\\TC\\BGI");

while(x1<x2)

{

outtextxy(x1,y1,"hello");

x1++;

y1=yc+sqrt(r\*r-(x1-xc)\*(x1-xc));

delay(100);

}

x1=xc-r;

x2=xc+r;

y1=yc;

while(x1<x2)

{

outtextxy(x1,y1,"Hello");

x1++;

y1=yc-sqrt(r\*r-(x1-xc)\*(x1-xc));

delay(100);

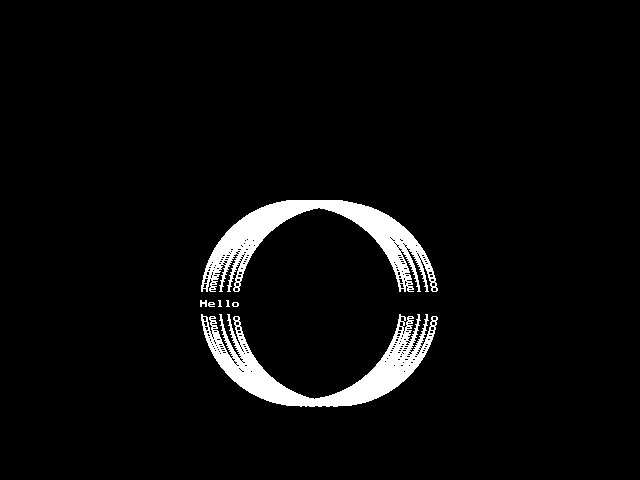
}

getch();

closegraph();

}

**Output:**



1. **Program of scaling**

#include<iostream.h>

#include<conio.h>

#include<graphics.h>

void main()

{

int gd=DETECT,gm;

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

int tx,ty;

int x1=20;

int x2=30;

int y1=10;

int y2=40;

rectangle(x1,x2,y1,y2);

cout<<"enter translation coordinate";

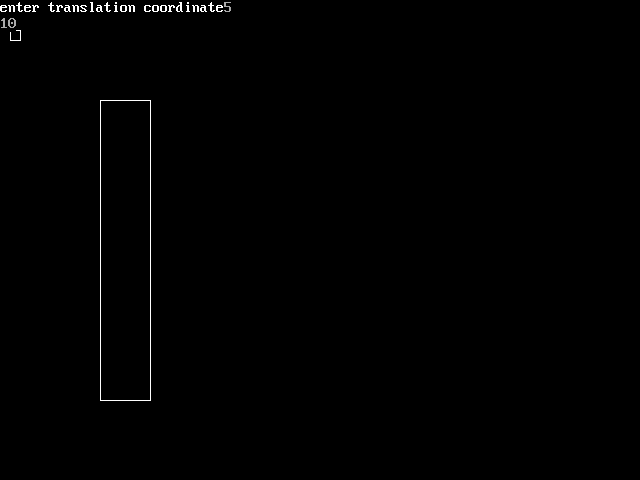
cin>>tx>>ty;

rectangle(x1\*tx,y1\*ty,x2\*tx,y2\*ty);

getch();

}

**Output:**



1. **Program of parallel projection**

#include<iostream.h>

#include<conio.h>

#include<graphics.h>

void main()

{

int gd=DETECT,gm;

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

line(50,100,100,50);

line(120,170,170,120);

line(50,100,120,170);

line(100,50,170,120);

setcolor(RED);

line(400,20,400,400);

setcolor(YELLOW);

line(100,50,400,50);

line(50,100,400,100);

line(120,170,400,170);

line(170,120,400,120);

setcolor(GREEN);

outtextxy(420,70,"orthographic view");

line(20,400,400,20);

setcolor(CYAN);

line(50,100,50,370);

line(120,170,120,300);

line(170,120,170,250);

line(100,50,100,320);

setcolor(BROWN);

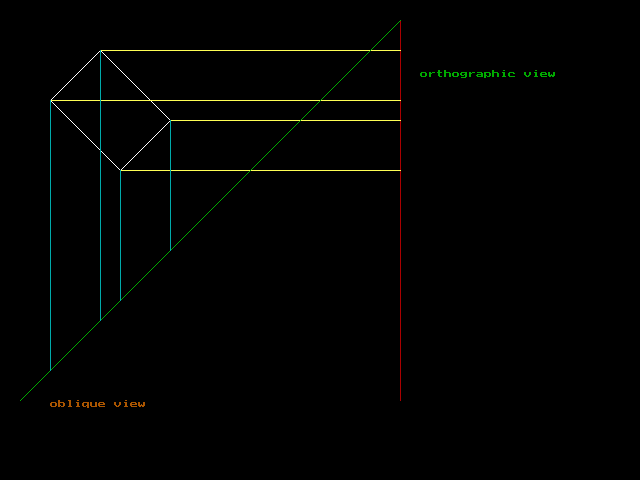
outtextxy(50,400,"oblique view");

getch();

closegraph();

}

**Output:**

****

1. **Program of perceptive view**

#include<iostream.h>

#include<conio.h>

#include<graphics.h>

void main()

{

int gd=DETECT,gm;

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

setbkcolor(WHITE);

setcolor(BLUE);

rectangle(40,80,80,120);

rectangle(20,60,60,100);

line(20,60,40,80);

line(60,60,80,80);

line(60,100,80,120);

line(20,100,40,120);

setcolor(MAGENTA);

line(60,60,300,90);

line(80,80,300,90);

line(80,120,300,90);

line(60,100,300,90);

line(20,60,300,90);

line(20,100,300,90);

line(40,120,300,90);

line(90,80,300,90);

setcolor(GREEN);

line(250,89.257,250,96.8180);

line(230,87.3076,250,87.727);

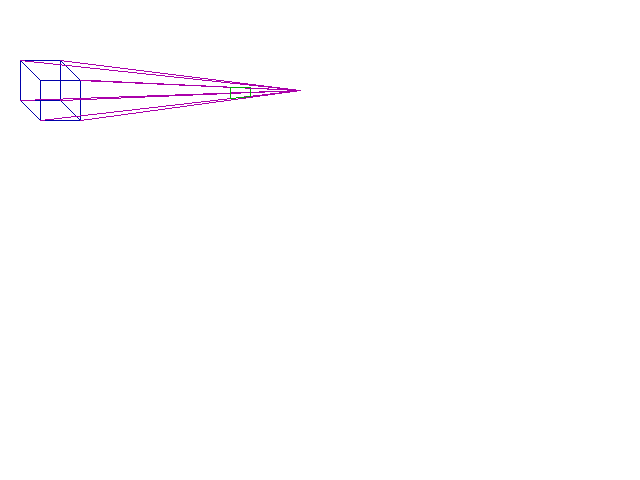
line(230,87.3076,230,98.0769);

line(230,98.0769,250,96.8180);

getch();

}

**Output:**



1. **Program of flood fill**

#include<iostream.h>

#include<conio.h>

#include<graphics.h>

void main()

{

int gd=DETECT,gm;

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

int i,maxx,maxy;

int poly[8];

maxx=getmaxx();

maxy=getmaxy();

poly[0]=20;

poly[1]=maxy/2;

poly[2]=maxx-20;

poly[3]=20;

poly[4]=maxx-50;

poly[5]=maxy-20;

poly[6]=maxx/2;

poly[7]=maxy/2;

for(i=EMPTY\_FILL;i<USER\_FILL;i++)

{

setfillstyle(i,getmaxcolor());

fillpoly(4,poly);

getch();

}

closegraph();

}

**Output:**

