# Learning to Draw Basic Graphics in C++

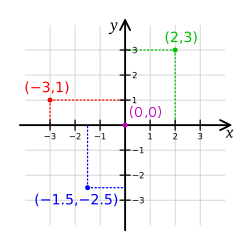
## **Graphics Essentials**

Before jumping into the details of how graphics work in Windows and how they are applied to games, it's important to establish some ground rules and gain an understanding of how computer graphics work in general. More specifically, you need to have a solid grasp on what a graphics coordinate system is, as well as how color is represented in computer graphics. The next couple of sections provide you with this knowledge, which you'll put to practical use a little later in the chapter.

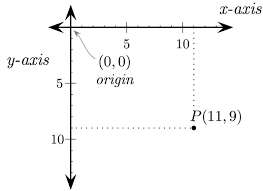
### **Understanding the Graphics Coordinate System**

graphics coordinate system to specify how points are arranged in a window or on the screen. Graphics coordinate systems typically spell out the origin (0, 0) of the system, as well as the axes and directions of increasing value for each of the axes.

**The traditional XY coordinate system is commonly used in math**



**The Windows XY coordinate system**



### **Learning the Basics of Color**

#### Numeric RGB Color Component Values for Commonly Used Colors

|  |  |  |  |
| --- | --- | --- | --- |
| Color | Red | Green | Blue |
| White | 255 | 255 | 255 |
| Black | 0 | 0 | 0 |
| Light Gray | 192 | 192 | 192 |
| Medium Gray | 128 | 128 | 128 |
| Dark Gray | 64 | 64 | 64 |
| Red | 255 | 0 | 0 |
| Green | 0 | 255 | 0 |
| Blue | 0 | 0 | 255 |
| Yellow | 255 | 255 | 0 |
| Purple | 255 | 0 | 255 |

/\* Simple example to draw circle \*/

#include”graphics.h”

#include”conio.h”

void main()

{

int gd=DETECT,gm;//graphics driver and mode

//Switch txt mode to GM

initgraph(&gd, &gm, “c:/tc/bgi “);//bgi= Borland graphics interface-access operating system lib

circle(330,180,100);

getch();

closegraph();

restorecrtmode();

}

Code explanation

The first line to look at is: GRAPHICS.H ,this file contains definitions and explaination of all the graphic functions and constants. While GRAPHICS.LIB file contains standard graphic functions.

Turbo C++ graphic functions have two categaries :Text mode graphic functions and graphic mode functions. Here we are dealing with graphic mode function.so just forget about text mode function right now. To switch from text mode to graphic mode,we have function called as ” initgraph ” .

initgraph : This function initialises the graphic mode. It selects the best resolution and direct that value to mode in variable gm.The two int variables gd, gm are graphic driver and graphic mode respectively. The gm handles value that tells us which resolution and monitor we are using. The gd specifies the graphic driver to be used. In our program we have gd=DETECT means we have passed the highest possible value available for the detected driver. If you don’t want that value then you have to assign the constant value for gd,gm.The ” &” symbol is used for initgraph to pass address of the constants.

Path ( ” C:\\tc\\bgi”) : It specifies the directory path where initgraph looks for graphics drivers (\*.BGI) first. If files are not there then initgraph will look for the current directory of your program. If it unable to find wihtin current working directory then it will parse an error. You can leave it blank ( ” ” ) if the \*.BGI files are within the working directory.

Circle( ) : Circle function takes X and Y values with respect to top left corner of the screen and third co-ordinate is nothing but radius of circle. In our example we have passed X=330,Y=180 and radius equal to 100 in terms of pixels as arguments.

Closegraph( ) : The closegraph() swithces back the screen from grpahics mode to text mode. If you don’t use this function then you may have undesirable effects.Here this function is called afer the getch() function as screen shouldn’t switch to text mode till user hits any key.

Restorcrtmode( ) : This mode will restore the original video mode detected by initgraph function.

getch( ) : getch( ) function gets a character from console but does not echo it on screen. This is used to pause the screen till user hits any key.

Note:

1) Make sure you have entered the correct path for the include & library directories. You can change the path by pointing your mouse to : Options > Directories. Enter the valid path for the include directory and libraries,and output directories.

2) After installation of Turbo C,you have to adjust the settings of linker. Go to Options>Linker > Libraries> and then check the ” Graphics Library“. This will help to solve the linker errors for the graphics programs. Please do not uncheck any other option already selected by compiler.

3) Graphic initialisation depends on the path mentioned in initgraph path. Be sure to enter slash between c,tc,bgi. The path C & TC depends on user if he installed TC in d: drive then it will be d,tc. Read the above code’s path carefully.

4) If you want help on specific function then point your mouse to “Help> Contents“,and then browse the content for the function you want. If you want fast-help then put the cursor on the first letter of the function or term and press CTRL+F1,it will point you to the help file of that term/function.

**Draw a line in C++ graphics**

**Examples:**

// C++ Implementation for drawing line

#include <graphics.h>

// driver code

int main()

{

    // gm is Graphics mode which is a computer display

    // mode that generates image using pixels.

    // DETECT is a macro defined in "graphics.h" header file

    int gd = DETECT, gm;

    // initgraph initializes the graphics system

    // by loading a graphics driver from disk

    initgraph(&gd, &gm, "");

    // line for x1, y1, x2, y2

    line(150, 150, 450, 150);

    // line for x1, y1, x2, y2

    line(150, 200, 450, 200);

    // line for x1, y1, x2, y2

    line(150, 250, 450, 250);

   getch();

  // closegraph function closes the graphics

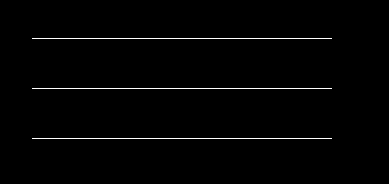
    // mode and deallocates all memory allocated

    // by graphics system .

    closegraph();

}

**Output:**



**functions of graphics.h**

* C graphics using graphics.h functions or WinBGIM (Windows 7) can be used to draw different shapes, display text in different fonts, change colors and many more.
* Using functions of graphics.h in Turbo C compiler you can make graphics programs, animations, projects, and games. You can draw circles, lines, rectangles, bars and many other geometrical figures.
* You can change their colors using the available functions and fill them. Following is a list of functions of graphics.h header file.

**C graphics examples**

1. Drawing concentric circles

#include <graphics.h>

int main()

{

int gd = DETECT, gm;

int x = 320, y = 240, radius;

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

for ( radius = 25; radius <= 125 ; radius = radius + 20)

circle(x, y, radius);

getch();

closegraph();

return 0;}

**C graphics functions**

arc

bar

bar3d

circle

cleardevice

closegraph

drawpoly

ellipse

fillellipse

fillpoly

floodfill

getarccords

getbkcolor

getcolor

getdrivername

getimage

getmaxcolor

getmaxx

getmaxy

getpixel

getx

gety

graphdefaults

grapherrormsg

imagesize

line

lineto

linerel

moveto

moverel

outtext

outtextxy

pieslice

putimage

putpixel

rectangle

sector

setbkcolor

setcolor

setfillstyle

setlinestyle

settextstyle

setviewport

textheight

textwidth

**bar3d() function in C graphics**

void bar3d(int left, int top, int right,

int bottom, int depth, int topflag);

Input : left = 150, top = 250,

right = 190, bottom = 350,

depth = 20, topflag = 1

**Draw shapes using C++ graphics**

This C graphics program draws basic shapes such as circle, line, rectangle, ellipse and display text on screen using C graphics. This can be a first graphics program for a beginner.

**C++ programming code**

#include<graphics.h>

#include<conio.h>

main()

{ int gd = DETECT,gm,left=100,top=100,right=200,bottom=200,x= 300,y=150,radius=50;

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

rectangle(left, top, right, bottom);

circle(x, y, radius);

bar(left + 300, top, right + 300, bottom);

line(left - 10, top + 150, left + 410, top + 150);

ellipse(x, y + 200, 0, 360, 100, 50);

outtextxy(left + 100, top + 325, "My first C graphics program");

getch();

closegraph();

return 0;

}

**Set Bk color:**

|  |  |
| --- | --- |
| BLACK | 0 |
| BLUE | 1 |
| GREEN | 2 |
| CYAN | 3 |
| RED | 4 |
| MAGENTA | 5 |
| BROWN | 6 |
| LIGHTGRAY | 7 |
| DARKGRAY | 8 |
| LIGHTBLUE | 9 |
| LIGHTGREEN | 10 |
| LIGHTCYAN | 11 |
| LIGHTRED | 12 |
| LIGHTMAGENTA | 13 |
| YELLOW | 14 |
| WHITE | 15 |

setbkcolor(WHITE) /\* or \*/ setbkcolor(15)

**setfillstyle() and floodfill() in C**

void setfillstyle(int pattern, int color)

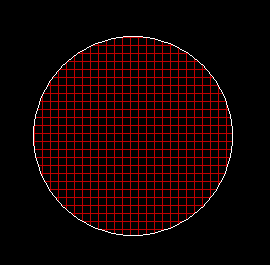
void floodfill(int x, int y, int border\_color)

Input : pattern = HATCH\_FILL, Color = RED

circle : x = 250, y = 250, radius = 100

floodfill : x = 250, y = 250, border color =15

Output :

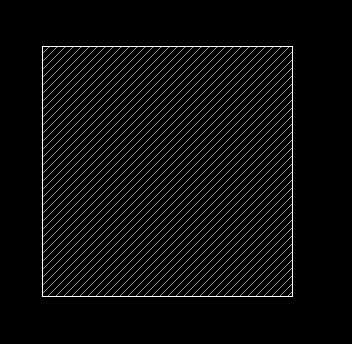


Input : pattern = LTSLASH\_FILL, Color = RED

rectangle : left = 200, top = 200, right = 450, bottom = 450

floodfill : x = 201, y = 201, border\_color = 15

Output :



**PATTERN** **INT VALUES**

-------------------------------

EMPTY\_FILL 0

SOLID\_FILL 1

LINE\_FILL 2

LTSLASH\_FILL 3

SLASH\_FILL 4

BKSLASH\_FILL 5

LTBKSLASH\_FILL 6

HATCH\_FILL 7

XHATCH\_FILL 8

INTERLEAVE\_FILL 9

WIDE\_DOT\_FILL 10

CLOSE\_DOT\_FILL 11

USER\_FILL 12

Below is the implementation for setfillstyle() and floodfill() function :

|  |
| --- |
| // C Implementation for setfillstyle  // and floodfill function  #include <graphics.h>    // driver code  int main()  {      // gm is Graphics mode which is      // a computer display mode that      // generates image using pixels.      // DETECT is a macro defined in      // "graphics.h" header file      int gd = DETECT, gm;        // initgraph initializes the      // graphics system by loading      // a graphics driver from disk      initgraph(&gd, &gm, " ");        // center and radius of circle      int x\_circle = 250;      int y\_circle = 250;      int radius=100;        // setting border color      int border\_color = WHITE;          // set color and pattern      setfillstyle(HATCH\_FILL,RED);        // x and y is a position and      // radius is for radius of circle      circle(x\_circle,y\_circle,radius);        // fill the color at location      // (x, y) with in border color      floodfill(x\_circle,y\_circle,border\_color);        getch();        // closegraph function closes the      // graphics mode and deallocates      // all memory allocated by      // graphics system      closegraph();        return 0;  } |

**C++ smiling face animation**

C programming code

#include<graphics.h>

#include<conio.h>

#include<stdlib.h>

main()

{

int gd = DETECT, gm, area, temp1, temp2, left = 25, top = 75;

void \*p;

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

setcolor(YELLOW);

circle(50, 100, 25);

setfillstyle(SOLID\_FILL, YELLOW);

floodfill(50, 100, YELLOW);

setcolor(BLACK);

setfillstyle(SOLID\_FILL, BLACK);

fillellipse(44, 85, 2, 6);

fillellipse(56, 85, 2, 6);

ellipse(50, 100, 205, 335, 20, 9);

ellipse(50, 100, 205, 335, 20, 10);

ellipse(50, 100, 205, 335, 20, 11);

area = imagesize(left, top, left + 50, top + 50);

p = malloc(area);

setcolor(WHITE);

settextstyle(SANS\_SERIF\_FONT, HORIZ\_DIR, 2);

outtextxy(155, 451, "Smiling Face Animation");

setcolor(BLUE);

rectangle(0, 0, 639, 449);

while(!kbhit())

{

temp1 = 1 + random (588);

temp2 = 1 + random (380);

getimage(left, top, left + 50, top + 50, p);

putimage(left, top, p, XOR\_PUT);

putimage(temp1 , temp2, p, XOR\_PUT);

delay(100);

left = temp1;

top = temp2;

}

getch();

closegraph();

return 0;

}