

Mr. Giansante



C++ Programming
Graphics

August 2018

C++ Graphics

Setting Up Graphics With Dev C++

In order to use Graphics with Dev C++, you must follow the instructions in the booklet entitled "Setting Up Graphics With Dev C++".

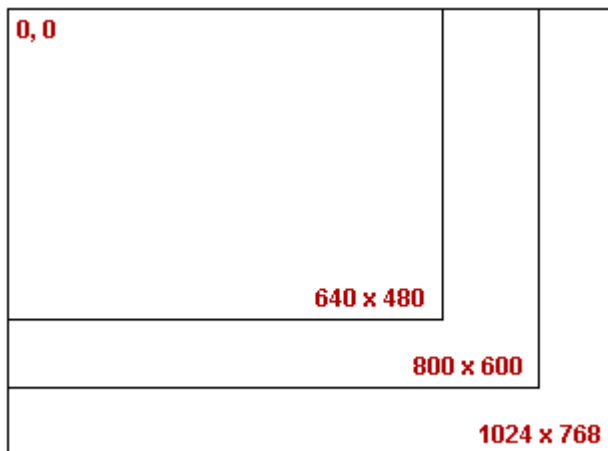
Colours Codes

Colour	Value
Black	0
Blue	1
Green	2
Cyan	3
Red	4
Magenta	5
Brown	6
Light Gray	7
Dark Gray	8
Light Blue	9
Light Green	10
Light Cyan	11
Light Red	12
Light Magenta	13
Yellow	14
White	15

Coordinate System

When programming with C++, points can be specified using the x and y coordinate system.

The point (0, 0) is found in the upper left-hand corner.



initwindow() / initgraph()

`initgraph()` initializes the graphics system by loading a graphics driver (or validating a registered driver), and putting the system into graphics mode.

```
initwindow(vertical size, horizontal size);
```

```
example. initwindow(300, 300);
```

closegraph()

`closegraph()` closes the graphics mode, deallocates all memory allocated by graphics system and restores the screen to the mode it was in before you called `initgraph()`.

```
example. closegraph();
```

setcolor()

`setcolor()` sets the current drawing color for lines, circles, ellipses, etc.

The parameter is a number between 0 and 15. Refer to the chart at the top left of this page.

```
example. setcolor(15); // Sets color to white
```

setbgcolor()

`setbgcolor()` sets the background color of the graphics area.

The parameter is a number between 0 and 15. Refer to the chart at the top left of this page.

```
example. setbgcolor(4); // Sets background to Red
```

putpixel()

`putpixel()` plots a pixel at location (x, y) of specified color.

```
example. putpixel(35,35,2); // Green at (35, 35)
```

getpixel()

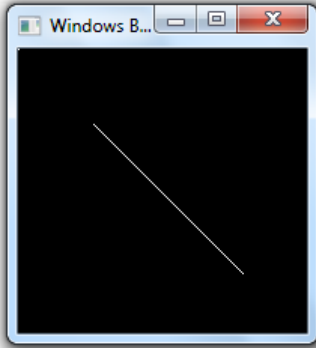
`getpixel()` returns the color of pixel present at location(x, y).

C++ Graphics

Drawing Lines

The `line(x1, y1, x2, y2);` function is used to draw a line from a point $(x1, y1)$ to point $(x2, y2)$

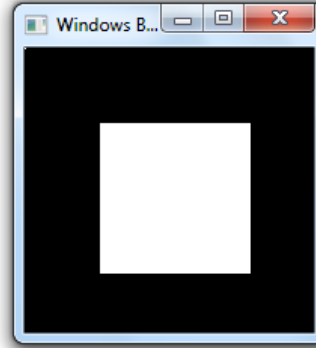
example. `line(20, 20, 150, 150);`



Drawing a Filled-in Rectangle

The `bar(x1, y1, x2, y2);` function is used to draw a filled-in rectangle. Coordinates of left top and right bottom corner are required to draw the rectangle.

example. `bar(50, 50, 150, 150);`

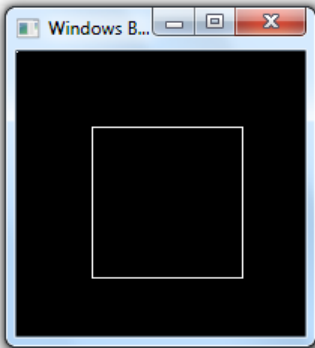


Drawing Rectangles

The `rectangle(x1, y1, x2, y2);` function is used to draw a rectangle. Coordinates of left top and right bottom corner are required to draw the rectangle.

example. `rectangle(50, 50, 150, 150);`

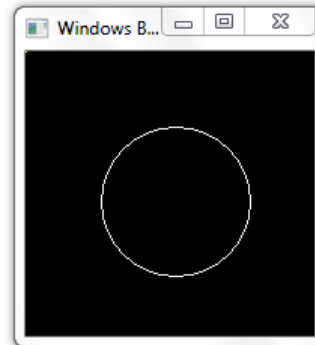
Draws a rectangle with top left corner at $(50, 50)$ and bottom right corner at $(150, 150)$.



Drawing a Circle

The `circle(x, y, r);` function is used to draw a circle with center at (x, y) and radius $= r$.

example. `circle(100,100,50);`



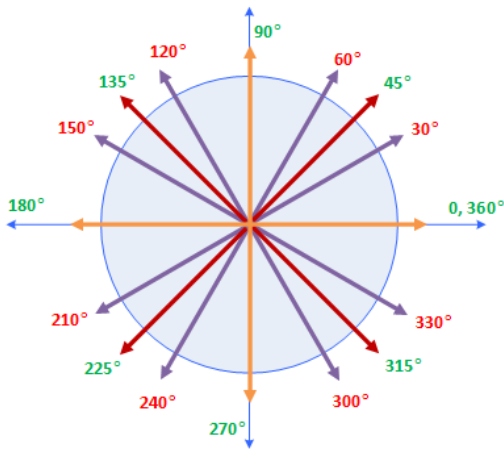
C++ Graphics

Drawing an Ellipse

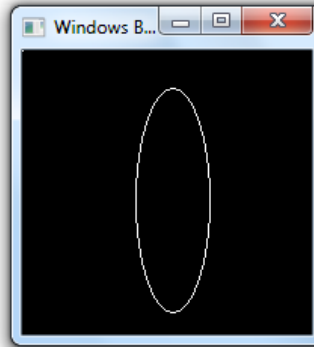
The `ellipse(x, y, start angle, end angle, x radius, y radius);` function is used to draw an ellipse.

(x,y) are coordinates of center of the ellipse
start angle is the starting angle (see diagram below)
end angle is the ending angle (see diagram below)
x radius is the horizontal radius
y radius is the vertical radius

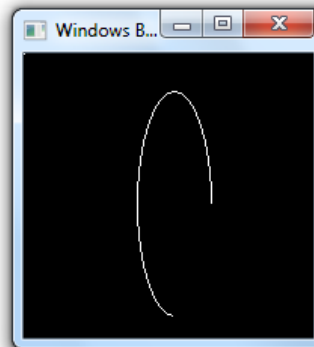
To draw a complete ellipse start angle and end angle should be 0 and 360 respectively.



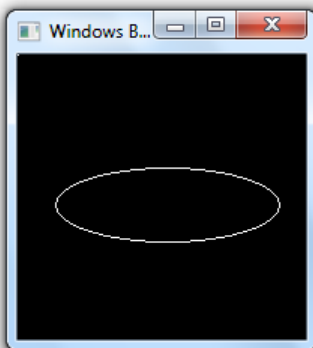
example. `ellipse(100, 100, 0, 360, 25, 75);`



example. `ellipse(100, 100, 0, 270, 25, 75);`



example. `ellipse(100, 100, 0, 360, 75, 25);`

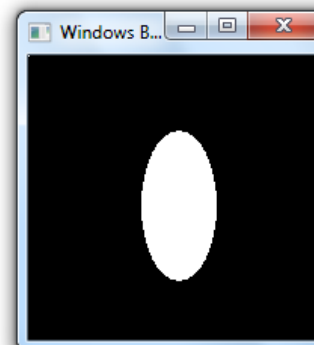


Drawing a Filled-in Ellipse

The `fillellipse(x, y, x radius, y radius);` is used to draw a filled-in ellipse.

(x,y) are coordinates of center of the ellipse
x radius is the horizontal radius
y radius is the vertical radius

example. `fillellipse(100, 100, 25, 50);`

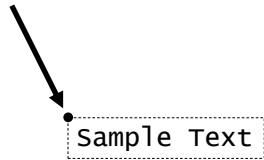


C++ Graphics

outtextxy(x, y, "Sample Text");

The `outtext()`; function is used to place text in the graphics window.

(x, y) are the top, left coordinates of the text



settextstyle(font, direction, size);

`settextstyle()` sets the text font, the direction in which text is displayed, and the size of the characters. A call to `settextstyle` affects all text output by `outtext` and `outtextxy`.

settextstyle - font

One 8x8 bit-mapped font and several "stroked" fonts are available. The 8x8 bit-mapped font is the default.

8x8 bit-mapped Font:

DEFAULT_FONT

Stroked Fonts:

TRIPLEX_FONT
SMALL_FONT
SANS_SERIF_FONT
GOTHIC_FONT
SCRIPT_FONT
SIMPLEX_FONT
TRIPLEX_SCR_FONT
COMPLEX_FONT
EUROPEAN_FONT
BOLD_FONT

The default bit-mapped font is built into the graphics system. Stroked fonts are stored in *.CHR disk files, and only one at a time is kept in memory. Therefore, when you select a stroked font the corresponding *.CHR file must be loaded from disk.

settextstyle - direction

Font directions supported are horizontal text (left to right) and vertical text (rotated 90 degrees counterclockwise). Horizontal text is the default.

HORIZ_DIR
VERT_DIR

settextstyle - size

The size of each character can be magnified using the `charsize` factor.

If `charsize` equals 1, `outtext` and `outtextxy` displays characters from the 8x8 bit-mapped font in an 8x8 pixel rectangle onscreen.

If `charsize` equals 2, these output functions display characters from the 8x8 bit-mapped font in a 16*16 pixel rectangle, and so on (up to a limit of ten times the normal size).