# Graphics in C++

## The Text Screen

- The text screen contains 25 lines with a capacity of holding 80 columns of textual characters.
- 80 X 25 = 2,000 positions
- But there are actually over 2,000 positions on a display screen.
- The screen consists of pixels (picture elements) that it uses to represent the textual characters and symbols.

## Graphics Setup

- Here are the steps that you need to follow to use “Borland Style Graphics” source code in Dev C++:
  1. Tell the compiler that graphics commands will be used.
  2. Initialize the Graphics Screen
  3. Close the graphics screen after you have finished drawing your graphics.

## Graphics Setup 2

- To tell the compiler that graphics commands will be used, include the preprocessor directive: #include <graphics.h>
- To initialize the graphics screen initwindow(640,480);
- After you are finished drawing, you need to use the while(!kbhit()); command to leave the picture on the screen, or use cin.get();
- The last choice requires: #include <iostream.h>
- Then close the graphics screen, using: closegraph( );

## Fundamentals of Graphics

- The Graphics Screen.
- Color Options.
- Graphics Mode.
- Drawing Lines
- Line Style
- Clearing the Screen.
- Plotting Points.

## The Graphics Screen

- If you have a VGA graphics card or better in your computer, then the graphics screen has 640 pixels across and 480 pixels down.
- 640 X 480 = 307,200 pixels
- The upper left corner is position (0, 0)
- The lower right corner is position (639, 479)

• Remember, the computer starts counting with zero.

## The Graphics Screen Dimensions


## Background Color Options

- You can select the color of the background.
- This is done before drawing anything in the foreground (otherwise your drawing will disappear.)
- To select the background color use the command: setbkcolor(number);
    - Where (number) is a numeric constant from 0 through 15, or the symbolic constant that represents the color.

## Color Options

- You select a foreground or "drawing" color by using the following command: setcolor(number);
    - Where (number) is a numeric constant from 0 through 15, or the symbolic constant that represents the color.

## Color Names

Here are the color numbers and names:

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

## Drawing Lines

- The Current Pointer: The current pointer is an invisible pointer that keeps track of the current pixel position. It is the equivalent of the visible cursor in text mode.

- To move the pointer to a location on the graph without drawing anything, use the command:
  moveto (X,Y);
    - This is like PenUp (PU) in LOGO

- To draw lines from the current pointer's position to another point on the graph, use the command:
  lineto (X,Y);
  - This is like PenDown (PD) in LOGO or SetXY (x, y)

## Graphics Figures

- Lines
- Rectangles
- Circles
- Arcs
- Ellipses
- Points

## Lines, The Easy Way

- Instead of using the commands: moveto and lineto, we can draw a line using one command:

  line(x1, y1, x2, y2);
- The points (x1, y1) describe the beginning of the line, while (x2, y2) describes the endpoint of the line.
- The numbers x1, y1, x2, y2 are integers.

## Rectangles

Rectangles can be drawn in different ways using lineto, moveto, moverel, and linerel. But an easier and faster way is using the Rectangle procedure which draws a rectangle in the default color and line style with the upper left at X1, Y1 and lower right X2, Y2.

rectangle (x1, y1, x2, y2);

## Circles

Circles can be drawn using the circle procedure. This draws a circle in the default color and line style with center at X, Y, radius in the X direction of Xradius, and corresponding Y radius.

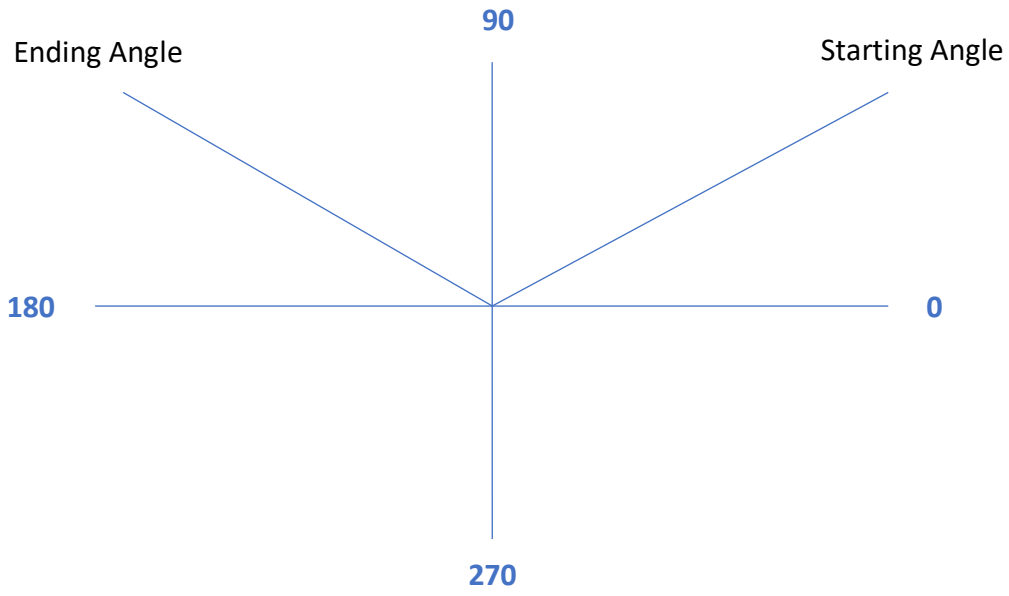
circle (x, y, radius);

## Arcs

This procedure draws a circular arc in the default color and line style based upon a circle with center X, Y and given X radius. The arc begins at an angle of Start Angle and follows the circle to End Angle. The angles are measured in degrees from 0 to 360 counterclockwise where 0 degrees is directly right.

arc ( x, y, startangle, endangle, radius);

## Visualizing Arcs Starting & Ending Angles




## Ellipses

Draws an elliptical arc in the default color and line style based upon an ellipse with center X, Y and given radii. The arc begins at an angle to Start Angle and follows the ellipse to End Angle. The angles are measured in degrees from 0 to 360 counterclockwise where 0 degrees is directly right.

ellipse ( x, y, startangle , endangle, x_radius, y_radius);

## Plotting Points

- The Maximum value for X can be found using:
  getmaxx( )
- The Maximum value for Y can be found using:
  getmaxy( )
- To Plot a point:
  putpixel ( x_value, y_value, color);
  - For example: putpixel (100, 100, WHITE);

## Sample Program

Let's look at a program with a line, rectangle, circle, arc, ellipse, and a point.

## Line Style

- Setting the line style.
  All lines have a default line mode, but Turbo C++ allows the user to specify three characteristics of a line: style, pattern, and thickness.

- Use the command:
  setlinestyle (style, pattern, thickness);