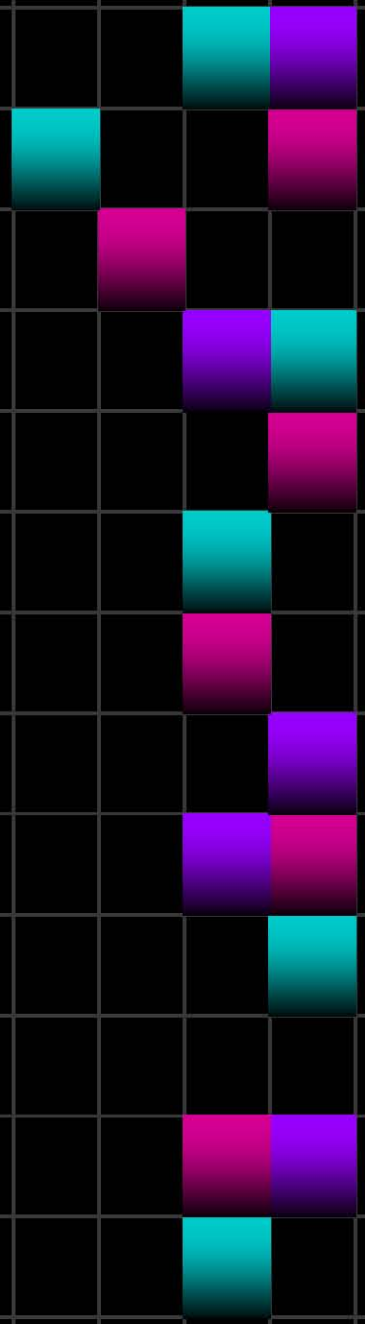
Borland C++ Graphics

(Turbo C++ v. 3.0 for DOS
or
Borland v. 5.02 for Windows)

Mr. Dave Clausen

La Cañada High School



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

- There are five steps that you need to follow to use graphics in Turbo C++ 3.0 DOS:
  1. Tell the compiler that graphics commands will be used.
  2. Have C++ find out what kind of graphics card your computer uses.
  3. Initialize the Graphics Screen
  4. Tell the system where to find Borland's Graphics routines.
  5. Close the graphics screen after you have finished drawing your graphics.

Graphics Setup 2

- 1) To tell the compiler that graphics commands will be used, include the preprocessor directive:

  #include <graphics.h>

- 2) To have C++ find out what kind of graphics card your computer uses.
  - declare two variables of type Integer

    int graphdriver = DETECT, graphmode;

I require that you use the following command:

int graphdriver = VGA, graphmode= VGAHI;

Graphics Setup 3

- 3 & 4) To initialize the graphics screen and tell the system where to find Borland's Graphics routines (BGI) use the following command in the int main( ) function:

initgraph(&graphdriver, &graphmode, "C:\\bc5\\bgi");

After you are finished drawing, you need to use the getch( ); command to leave the picture on the screen (Press any key to continue...).
This requires: #include <conio.h>

- 5) Then close the graphics screen, using:

closegraph( );

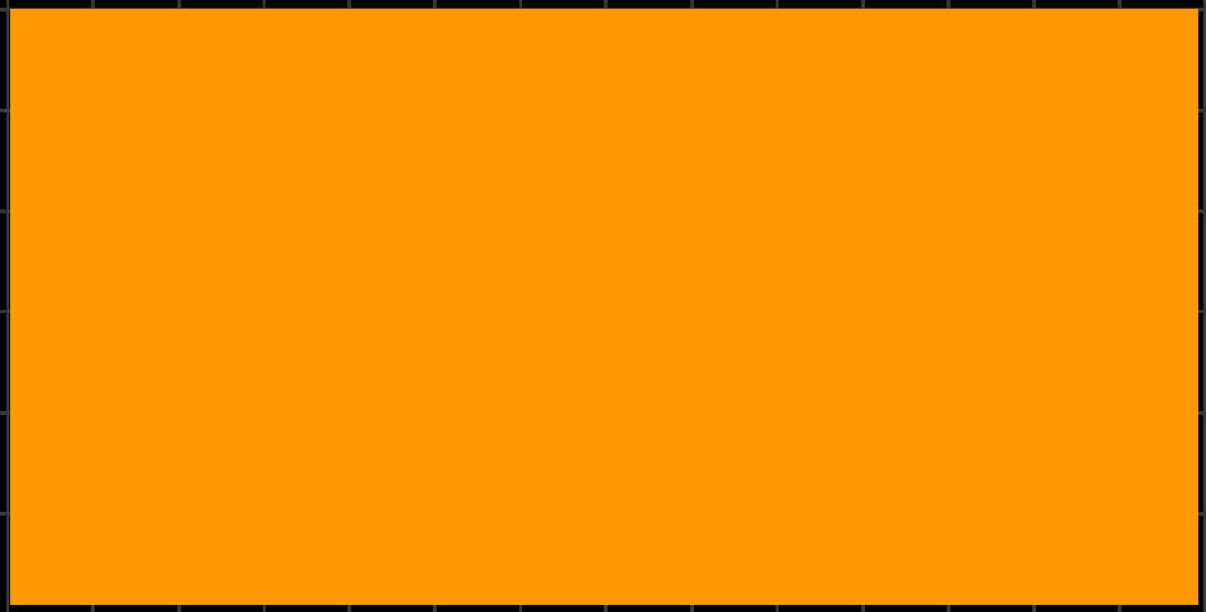
Fundamentals of Graphics

- 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

- The number of colors depend on the graphics mode you select using Turbo C++.
  - The default settings allow for 16 color choices.
- 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.

graphinfo.cpp

Color Names

Here are the color numbers and names:

0 = BLACK	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.

Drawing Lines 2

- 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)

grtmplte.cpp

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 (x_1, y_1, x_2, y_2);

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.

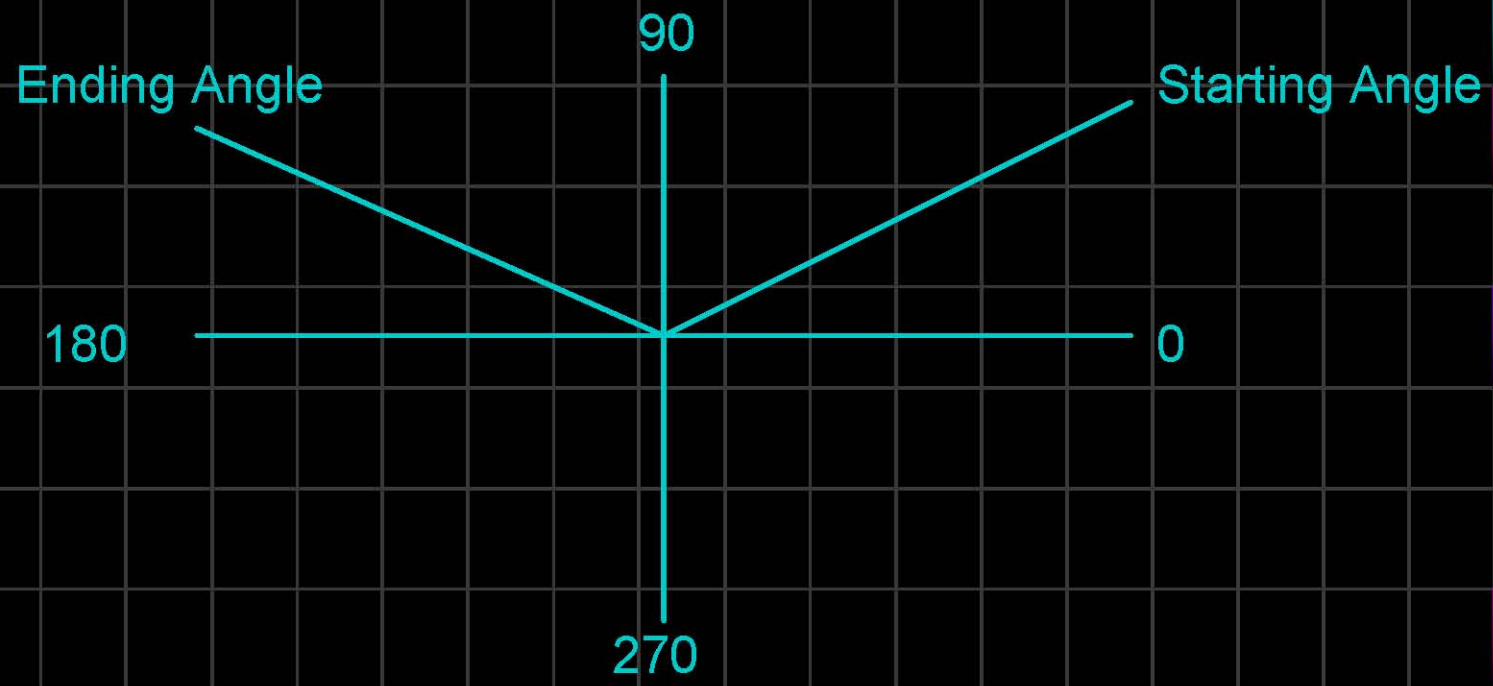
```
circle (x, y, radius);
```

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 StartAngle and follows the circle to EndAngle. The angles are measured in degrees from 0 to 360 counter-clockwise 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 counter-clockwise 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.

Objects.cpp

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);

Line Style and Thickness Names

Here are the names of the line styles and thickness:

Line Style	Thickness
SOLID_LINE	NORM_WIDTH
DOTTED_LINE	
CENTER_LINE	THICK_WIDTH
DASHED_LINE	
USERBIT_LINE	

Line Style Patterns

- The names of the line patterns are:

```
SOLID_LINE    = 0
DOTTED_LINE   = 1
CENTER_LINE   = 2
DASHED_LINE   = 3
```

Filling Patterns

- Selecting Pattern and Color
- Filling Regions
- Getting a Pixel

Selecting Pattern and Color

Use the command SetFillStyle for setting the pattern and color for the object that you wish to fill.

setfillstyle (pattern, color);

Pattern Names

Here are the name of available patterns:

Values	Causing filling with
EMPTY_FILL	Background Color
SOLID_FILL	Solid Color
LINE_FILL	Horizontal Lines
LTSLASH_FILL	Thin diagonal lines
SLASH_FILL	Thick diagonal lines
BKSLASH_FILL	Thick diagonal backslashes
LTBKSLASH_FILL	Light backslashes
HATCH_FILL	Thin cross hatching
XHATCH_FILL	Thick cross hatching
INTERLEAVE_FILL	Interleaving lines
WIDE_DOT_FILL	Widely spaced dots
CLOSE_DOT_FILL	Closely spaced dots

Filling Regions

- After selecting a color and pattern, floodfill is used to fill the desired area.
- floodfill (x, y, border_color);
- This "paints out" the desired color until it reaches border color.
- ***Note: The border color must be the same color as the color used to draw the shape.***
- **Also, you can only fill completely "closed" shapes.**

Program10_4.cpp

Filling "Special" Regions

- To draw a filled ellipse:

fillellipse (xcoordinate, ycoordinate, xradius, yradius);

- To draw a filled rectangle:

bar (x1, y1, x2, y2);

- To draw a filled 3D rectangle:

bar3d(x1, y1, x2, y2, depth, topflag); //depth is width of the 3D rectangle, if topflag is non-0 a top is added to the bar

- To draw a filled section of a circle:

pieslice (x, y, startangle, endangle, xradius);

Text Output on the Graphics Screen

- To write a literal expression on the graphics screen using the location specified by (x, y) use the command:

outtextxy (x, y, “literal expression”);

outtextxy (x, y, string_variable.c_str( ));

Note: string_variable represents a “C-style” string.

When using an apstring variable use the c_str( ) member function to convert the string.

Converting Int to apstring

- The Marine Biology Case Study includes a function in the "utils" class to convert an integer to apstring.
- This function can be found in the "Part 2" folder.
- The filename is:

utils.cpp

Text Styles

- To set the values for the text characteristics, use:

settextstyle (font, direction, charsize);

Font

DEFAULT_FONT
TRIPLEX_FONT
SMALL_FONT
SANS_SERIF_FONT
GOTHIC_FONT
SCRIPT_FONT
SIMPLEX_FONT
TRIPLEX_SCR_FONT

Direction

HORIZ_DIR = Left to right
VERT_DIR = Bottom to top

Fonts continued

COMPLEX_FONT
EUROPEAN_FONT
BOLD_FONT

Text Styles
Font Sizes

CharSize

1 = Default (normal)
2 = Double Size
3 = Triple Size
4 = 4 Times the normal
5 = 5 Times the normal
….
10 = 10 Times the normal

Text Justification

- To set the way that text is located around the point specified use the command:

  settextjustify (horizontal,vertical);

Horizontal	**Vertical**
LEFT_TEXT	TOP_TEXT
CENTER_TEXT	BOTTOM_TEXT
RIGHT_TEXT	

Program10_2.cpp

Clearing the Screen

- There are two ways to clear the screen.
- When in graphics mode use:

  cleardevice( ); //#include <graphics.h>

- ***When not in graphics mode use***:

  clrscr( ); //#include <conio.h>
  - This **only clears the text screen**, not a graphics screen.

Text
Height & Width

- Returns the height, in pixels, of string S if it were to be written on the graphics screen using the current defaults.

  textheight (S string);
- Returns the width, in pixels, of string S if it were to be written on the graphics screen using the current defaults.

  textwidth (S string);

Getting a Pixel

- To return the color number corresponding to the color located at the point: X, Y use the command:

  getpixel (x, y);

Useful Non Graphic Commands

- kbhit()
  - checks to see if a keystroke is currently available
  - If a keystroke is available, returns a nonzero integer.
  - If a keystroke is not available, returns a zero.
- Any available keystrokes can be retrieved with getch().
  - Both kbhit( ) and getch( ) belong to <conio.h>

Using Borland C++ 5.02

- Create a project with the following settings: