## Line Style and Thickness Names

Here are the names of the line styles and thickness:

**Line Style**

SOLID_LINE
DOTTED_LINE
CENTER_LINE
DASHED_LINE
USERBIT_LINE

**Thickness**

NORM_WIDTH
THICK_WIDTH

## 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 is 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.

## 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);

- Note: These are not “apstring” type strings. They are C++ standard Strings.

## 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
COMPLEX_FONT
EUROPEAN_FONT
BOLD_FONT

**Direction**

HORIZ_DIR = Left to right
VERT_DIR = Bottom to top

## 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 | |

## Clearing the Screen

- Here is the way to clear the graphics screen.
- When in graphics mode use:

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

## 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().