Graphics Text Tutorial

-
Alyce

TEXT GRAPHICS COMMANDS

Text can be displayed in a graphics window or graphicbox. Make sure the pen is DOWN so that the text will be visible. It will appear with its lower left corner at the pen location. It will display in the currently selected COLOR and it will be backed by the currently selected BACKCOLOR. It will appear in the currently selected FONT, including face, width, height and attributes. The default COLOR is black. The default BACKCOLOR is white. These can be changed as often as needed, so many colors of text can be displayed in a single graphics area.

Tell Liberty BASIC that you are displaying graphic text by using the backslash or pipes character. The pipes is the uppercase character for the backslash key. Here is a graphic text command.

```
print #1, "\Some text here!"
```

PLACEMENT

Remember that the pen begins at position 0, 0. The text will be placed with the current pen position as its LOWER LEFT corner. If the pen is at 0, 0 the text will not show because it will actually be located above the visible graphics area. Use the PLACE command to position the pen for graphic text:

```
print #1, "place 20 130;\Hello World!"
```

CARRIAGE RETURNS

Each backslash character signals a carriage return. The command here will produce text on two lines:

```
print #1, "\Hello\World!"
```

Produces
Hello
World!

If there will be backslash characters in the text string, such as you would find in a path and filename, then use the pipes character to signal graphic text. Liberty BASIC will then interpret backslash characters in a text command to be part of the text, and not commands for carriage returns. The command here will produce text on a single line:

```
print #1, "|c:\myfiles\mypic.bmp"
```

Produces
c:\myfiles\mypic.bmp

If the backslash had been used to signal graphictext, this would be the result:

```
print #1, "\c:\myfiles\mypic.bmp"
```

Produces
c:
myfiles
mypic.bmp

Do not attempt to place other commands after graphic text. We can chain graphics commands together with semi-colons, but when using graphic text, the semi-colons are interpreted as characters within the text, not as dividers for commands.

```
'wrong:
print #1, "\Hello World!;color red"
```

Produces:
Hello World!;color red

```
'correct:
print #1, "\Hello World!"
print #1, "color red"
```

FONT

Graphics windows and graphicboxes understand the FONT command. The command can be issued many times during graphics drawing, so multiple fonts may be displayed in a single graphics area. You can specify the facename desired, and the width, height, and attributes, such as bold and italic. Generically, a font command looks like this:

Syntax: print #handle, "FONT FACENAME [W] H [ATTRIBUTES]"

The minimum information needed by a FONT command is the facename and height.

FONT SIZE

If only one number is given as a dimension, it is assumed to be point size. If two numbers are given, they are interpreted as the width and height in pixels. Pixel size does not correspond with point size. A font that is 14-points is not the same size as one that is 14 pixels high. Pixel size is dependent upon the resolution on the individual computer. A 15-inch monitor with a resolution of 800x600 will display a font that is 14 pixels high to appear smaller than the same monitor if it is set to 640x480 resolution. There are 72 points in an inch. A font sized by points will be exactly the same size, no matter what screen resolution is in effect on a computer. Since all other graphic commands are issued in pixels, it is suggested that you use pixels to size fonts for graphic text, rather than points. If the width is given as 0, then the default width for that font will be used. Examples:

```
'font point size used:
print #1, "font arial 14"    'a 14-point font size
'font pixel size  used:
print #1, "font arial 8 16"  'chars are 8 pixels wide, 16 high
'font pixel height used:
print #1, "font arial 0 16"  'chars are 16 pixels high, default width
```

All fonts are not created equal! The width designated will be the width of an average character as displayed in the font face. If the font is a fixed-width font, such as Courier New, then all characters will be the same width. Most fonts have variable-width characters.

FONT FACENAME

You can specify any facename you would like. If that font face is not available, Windows will use the best alternative. (It will usually substitute Arial for unknown font faces.) It is important to have the font facename as all one word for Liberty BASIC font commands. This is done by replacing spaces with underscores, like this:

Courier New
becomes

Courier_New

You can count on any Windows computer having the following True Type fonts installed:

- Courier New
- Times New Roman
- Arial
- Ms Sans Serif

If you need to have text that lines up in columns, then be sure to use Courier New for the font, because each character is the same width. This is not true of the other fonts.

The font facename is case insensitive, so you can type ARIAL, Arial or arial.

FONT ATTRIBUTES

After specifying the facename and size, you may specify attributes. The attributes are optional. You may have any or all of the following attributes:

- italic
- bold
- strikeout
- underscore

Here are some sample FONT commands, as sent to a graphics window:

```
print #1, "font arial 10 bold"
print #1, "font courier_new 0 14"
print #1, "font ARIAL 8 16 italic bold"
print #1, "font Times_New_Roman 0 12 underscore"
```

STRINGWIDTH?

You can obtain the width in pixels of a string of text easily with the STRINGWIDTH? command. It will return the width in pixels of the string variable in the current font in use in the graphicbox or graphics window.

Syntax: print #handle, "stringwidth? var$ pixelwidth"

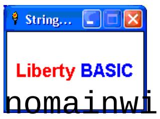
As it would appear in a program:

```
name$ = "Liberty BASIC"
print #1, "stringwidth? name$ pwide"
```

After this command is issued, the width of the text in the variable name$ will be contained in the receiver variable pwide. Notice that this use of receiver variables is the only time in Liberty BASIC programming where variables should be placed inside of the quote marks.

Why would we need this command? It allows us to place a string of text on the window in relation to an existing string. You might want to do this if the text switches colors, or if text needs to be placed in relationship to existing text at a later spot in the program. Here is a small demo that uses STRINGWIDTH?

Demo



```
nomainwin

WindowWidth=160:WindowHeight=120
open "Stringwidth Test" for graphics_nsb_nf as #1
print #1, "trapclose [quit]"
print #1, "down; color red"
print #1, "font arial 0 24 bold"
print #1, "place 10 50"
'set up string variables:
name1$ = "Liberty " 'notice the space after the word
name2$ = "BASIC"
'print the first variable on the window:
print #1, "\";name1$    'don't forget the backslash!
'get the width of that string:
print #1, "stringwidth? name1$ pwide"
'change color:
print #1, "color blue"
'now locate the pen after the first text
newX = 10 + pwide   'starting point, plus text width
print #1, "place ";newX;" 50"
'and print more text:
print #1, "\";name2$
```

```
wait
[quit]

close #1:end
```

Graphics Text Tutorial | TEXT GRAPHICS COMMANDS | PLACEMENT | CARRIAGE RETURNS | FONT | FONT SIZE | FONT FACENAME | FONT ATTRIBUTES | STRINGWIDTH? | Demo