Turtle Graphics Tutorial

-
Alyce

What are Turtle Graphics?

In Liberty BASIC, we don't have a robotic turtle walking around on the floor. When we issue turtle graphics commands, we see the output in a graphics window or in a graphicbox. We can imagine that we have a turtle that holds a drawing pen. If the drawing pen is down, then when the turtle travels, he leaves a trail on the display. If the pen is up, the turtle travels, but no graphics are drawn.

Note that Liberty BASIC has many drawing commands that will draw boxes, circles, etc., but we will not discuss these in this article. We will focus only on the turtle graphics commands.

IMPORTANT NOTE
Drawing commands are issued inside of quote marks. The commands are case insensitive, which means that it doesn't matter if you type them in all uppercase, all lowercase, or in a combination.

POSITIONING THE DRAWING PEN

When a graphics window is opened, the drawing pen is located at point 0, 0, which is the upper left corner of the drawing area (another name for this area is the client area.)

Graphics are measured in PIXELS, which is short for PICTURE ELEMENTS, or dots on your screen. Point 0, 0 is in the upper left corner of the drawing area. As x positions get larger they move to the right. As y positions get larger they move down. This might be confusing at first for those of us who were used to drawing graphs in math where the 0, 0 point was positioned at the LOWER left corner. Don't worry, after a while the computer's method of positioning will seem natural!

When we first open a graphics window or graphicbox, the pen is up. To draw, we must first put the pen down, like this:

```
print #gr, "down"
```

All graphics commands will be visible until we issue the UP command, like this:

```
print #gr, "up"
```

When the pen is UP, the drawing pen will move around in the drawing area, but no graphics will be drawn.

We can locate the drawing pen in several ways. If we want to center the pen in the drawing area, we can issue a HOME command, like this:

```
print #gr, "home"
```

We can locate the drawing pen with the GOTO command. The pen will move to the x, y position indicated, and it will draw if it is down. If the pen is up, it will move to the position indicated, but no line will be drawn.

```
print #gr, "goto 100 20"
```

Before we continue, let's consider an important point!

USING VARIABLES IN DRAWING COMMANDS

In the example above, the value for x is 100, and the value for y is 20. These are "hard coded" values. This means that the actual values appear in the command. It may at times be necessary to give a command when values are variables or expressions. This is easy, but it requires you to remember that values that are not literals must be located OUTSIDE of the quote marks, and that BLANK SPACES MUST BE PRESERVED.

Correct usage:

```
    x=100 : y = 20
    print #gr, "goto ";x;" ";y
```

In the above example, Liberty BASIC sees this:

```
    print #gr, "goto 100 20"
```

Incorrect usages:

```
     x = 100 : y = 20
     print #gr, "goto x y" 'variables are not outside of quotes!
     print #gr, "goto";x;y 'no blank spaces!
```

In the above examples, Liberty BASIC sees this:

```
     print #gr, "goto x y"
     print #gr, "goto10020"
```

There is yet another way to locate the drawing pen in Liberty BASIC turtle graphics. The pen will move the desired distance in the current direction when a GO command is issued, like this:

```
     print #gr, "go 10"
'or

     d=10
     print #gr, "go ";d
```

If the pen is DOWN, a line will be drawn from the current pen position in the current direction, and it will be the length specified. If the pen is UP, the pen will move to the new position, but no line will be drawn.

We may also locate the pen in the desired x, y position with the PLACE command, which moves the pen to the position indicated without drawing. This isn't really a part of turtle graphics, even though it is a part of Liberty BASIC graphics, so we won't use it here.

We can also move the drawing pen with a SET X Y command. This command draws a point in the x, y location given.

```
     print #gr, "set 100 20"

'or
     x = 100 : y = 20
     print #gr, "set ";x;" ";y
```

SETTING AND CHANGING DIRECTION

In the GO command that was mentioned earlier, we stated that the pen would travel the distance indicated in the CURRENT DIRECTION. When a graphicbox or graphics window is first opened, the direction is NORTH. We can return the direction to the default NORTH direction whenever we want by issuing a NORTH command:

```
     print #gr, "north"
```

The direction of travel can be changed with a **TURN A** command. **A** will be the number of degrees in the angle used to turn the pen direction. An angle of 180 will reverse the direction. An angle of 90 will cause the pen to begin traveling at a right angle to the current direction. When a TURN command is issued, the pen does not move. It turns while staying at the same x, y location.

If the angle specified is positive, the direction will change in a clockwise direction. If it is negative, the change will be in a counter-clockwise direction.

```
     print #gr, "turn 90"

'or
     a = 90
     print #gr, "turn ";a
```

There are 360 degrees in a circle. You can specify an angle that is greater than 360 degrees. In that case, the drawing direction will make a full rotation for each multiple of 360 degrees, and the remainder is the actual angle of change. For example, if you specify an angle of 450, the first 360 degrees will simply place the drawing pen so that it is heading in the same direction as it was before the command, and then it will turn 90 degrees. 450 - 360 = 90. This means that specifying an angle of 450 has exactly the same result as specifying an angle of 90.

If you want the pen to travel towards the "east" or towards the right side of the window, then issue a NORTH command, followed by a TURN 90 command, like this:

```
     print #gr, "north"
     print #gr, "turn 90"
```

CHAINING COMMANDS

The example above could be written on a single line, like this:

```
     print #gr, "north; turn 90"
```

Graphics commands can be written on a single line within a pair of double quote marks if they are separated by a semi-colon, as above. If graphics text is used, no commands can be added after a semi-colon, but we are not discussing graphics text in this article, so we don't need to worry about this now.

SIZE AND COLOR

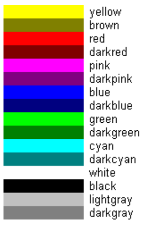
The drawing pen is one pixel wide by default. We can make the pen as wide as we would like by issuing a SIZE command, like this:

```
     print #gr, "size 5"
'or
      s = 5
     print #gr, "size ";s
```

We can draw in the color of our choice as well. Liberty BASIC allows us to choose from 16 named colors, or to make an RGB color choice. The RGB choice will be discussed in a future graphics tutorial.

Named Colors

The 16 named colors are:



Issue a COLOR command like this:

```
     print #gr, "color green"

'or
     color$ = "green"
     print #gr "color ";color$
```

The colornames are case insensitive, so RED, Red, and red are all acceptable.

LOOPING

Because we can use variables in our turtle graphics commands, we can achieve some very fancy graphics in very few lines of code. Look at the following tiny program. It uses a loop to draw the graphics, and the counter variable for the loop determines the distance the pen travels in each step.

```
nomainwin
     WindowWidth=400 : WindowHeight=400
     open "Test" for graphics_nsb as #gr
     print #gr, "down; home; color red; size 2"
for d = 1 to 30
     print #gr, "turn 60; go ";d
     next d
wait
```

Do you see that we have used the counter variable, d, as the distance variable as well? And because it is a variable, we've placed it outside of the quote marks. We've hard-coded the turning angle of 60 degrees, so it is placed inside of the quote marks.

Let's examine the steps and see what Liberty BASIC "sees" when it executes the program.

On the first iteration (that's a fancy word for step) Liberty BASIC sees this:

```
print #gr, "turn 60; go 1"
```

It turns 60 degrees clockwise and travels a distance of 1 pixel in that direction.

On the second iteration, Liberty BASIC "sees" this:

```
print #gr, "turn 60; go 2"
```

It now turns another 60 degrees clockwise and travels two pixels in that direction. Liberty BASIC repeats this procedure, incrementing the counter variable by one each time, until the counter variable reaches 30.

We can achieve a different figure by changing the step increment from the default value of 1. Let's change it to 2, like this:

```
nomainwin
     WindowWidth=400 : WindowHeight=400
     open "Test" for graphics_nsb as #gr
     print #gr, "down; home; color red; size 2"
for d = 1 to 30 step 2
     print #gr, "turn 60; go ";d
     next d
wait
```

Here is a more complicated example. We can make both the angle and the distance dependent on the value of the counter variable.

```
nomainwin
     WindowWidth=400 : WindowHeight=400
     open "Test" for graphics_nsb as #gr
     print #gr, "down; home; color red; size 2"
for d = 1 to 30
     print #gr, "turn " ;d+40; " ; go ";d
     next d
wait
```

In the example above, the value of the angle will be equal to the counter variable plus 40.

DEMO

For an impressive display of turtle graphics in Liberty BASIC, look at the mandala.bas program that comes in the Liberty BASIC distribution.

TURTLE GRAPHICS COMMANDS:

print #gr, "up" 'lift the pen - don't draw
print #gr, "down" 'lower the pen - begin drawing
print #gr, "go D" 'go D pixels in the current direction
print #gr, "home" 'center the pen
print #gr, "north" 'set direction to North - 270 degrees
print #gr, "turn A" 'turn angle A degrees

```
print #gr, "goto X Y" 'go to point XY, drawing if the pen is down
print #gr, "set" 'draw one point at current pen position
print #gr, "set X Y" 'draw one point at location XY
print #gr, "color colorname" 'set the color to be colorname
print #gr, "size n" 'set the pen size to be n
```