Computer Science I — CSCI 141
Turtle Graphics Reference — Lecture 3 of 3

08/29/2016

Getting the turtle graphics window to a desired size and modifying the coordinates used in the window are part of the overall turtle drawing process. These steps can be confusing and are typically not the primary learning objective, but rather just a necessary step in the overall effort. Nonetheless it's worth spending a minute to understand them a little better.

The basic command for setting the size of the window is: `turtle.setup(width, height)`. This can be specified as a floating point fraction of the screen, but most often just uses integers to indicate the number of pixels for the width and height. By default, it is set to be 50% of the width, 75% of the height of the display. Note that this will be a different number of pixels based on the resolution of your display. So a drawing that fits in the default size window on my laptop may or may not fit inside of the default size window on another display. The origin `(0,0)` is in the middle of the window. The bottom-left corner of the window has coordinates `(-width/2, -height/2)` and the top-right corner of the window has coordinates `(width/2, height/2)`. (Somewhere in there is a `-1` to make the total window size exactly `width` × `height` but we're not going to worry about that.) The turtle by default starts at the origin.

If your drawing extends outside of your window, you can stretch the boundaries of the window using the mouse, or redefine the window using `turtle.setup` to possibly make the drawing fit. Based on the way your window is presented on your display, likely the last few pixels of the window are outside of actual view, because of the window border.

Turtle also defines the concept of a canvas. The canvas is the area you intend to draw inside. The command `turtle.screensize(canvas_width, canvas_height)` allows you to set the size of the canvas. You can make the canvas larger than the presented window, in which case scrollbars are provided to let you scroll to different parts of the canvas.

For our usage of turtle graphics, we'll have no need to have our canvas be any larger than the window itself, and we won't need to use the `turtle.screensize` command.

If you know exactly how large your drawing will be, it is usually sufficient to just define your window to be the appropriate size to fit it, or conversely, for simple drawings you can just choose turtle movement commands of an appropriate size to fit conveniently in the default window.

But what if the user provides input that can affect the overall size of the drawing? What if the drawing is coming out much too big or much too small? What if you want the picture to be a certain size (i.e. inches) regardless of the resolution of the display on which it is viewed?

What we will often do for our turtle drawings is to make the window some standard size (perhaps just the default), and then based on how much movement needs to fit inside that window, we will redefine the coordinates inside of our window. The command of interest is: `turtle.setworldcoordinates(llx, lly, urx, ury)`. The four parameters specify the x value for the lower left corner of the window, the y value for the lower left corner of the window, the x value for the upper right corner of the window, and the y value for the upper right corner of the window. Using this command maps the coordinates you provide to the visible window. **The window itself does not change size.** Again, it's not always clear exactly where the border is, so to make sure that our picture does fully appear, and isn't right along the edge of the window, we can just add a few extra pixels for the margin on each side.

So, for example, if you know your drawing will extend no more than `size` units of movement in any direction, you can use a margin of 5 (or something similar) and use the following commands:

```
turtle.setup(600,600) # gives us a square window of specified size

turtle.setworldcoordinates(-size-5,-size-5,size+5,size+5).
```

Once you've set the world coordinates in this manner, if you tell the turtle to move forward by `size` units from the center of the window, it will result in a line that reaches just about the edge of the window, regardless of the specific value of `size`.

Note that this puts the origin in the middle of the window. Sometimes you might like to be able to start from the lower-left corner (or somewhere else other than the center of the window). You can either lift the pen and move there, or you can also define the world coordinates so that the origin is the lower-left corner.

For example,

```
turtle.setup(600,600) # gives us a square window of specified size

turtle.setworldcoordinates(-5,-5,2*size+5,2*size+5).
```

This constructs our window so that the same size drawing will fit in it as before. Now, however, the origin (where the turtle starts out) is down at the lower-left, which may be convenient.

Three final comments. One, when you map your coordinates, it is possible to use different scaling factors in the horizontal and vertical directions, causing your picture to look stretched, or squished in one direction. Be aware that if you use the default window setup (in which the number of pixels may be different for horizontal and vertical dimensions) and then you map it to world coordinates that are an equal number of units in each dimension, this will cause your picture to stretch/squish.

Two, by choosing a fixed window size (say (600×600)) and scaling all drawings to fit inside this window, we mostly eliminate the affect of a `size` parameter. Many programs will prompt for a `size` parameter for the drawing, but this ultimately will have little to no effect as the drawing is scaled so that movements of length `size` look the same, regardless of the value of `size`.

Three, you may still not be completely satisfied with the above approach because it uses a window of size 600×600 pixels, which will appear differently on displays having different resolutions. To write a program that will generate a window of "standard size" for all displays, while simultaneously ensuring that the drawing fits appropriately inside that window, you can use the following sequence:

```
turtle.reset()

turtle.setup(turtle.window_height(), turtle.window_height())

turtle.setworldcoordinates(-size-5,-size-5,size+5,size+5).
```

The first statement resets and gives you a default window of 50% of the display width, 75% of the display height. This allows the program to generate a window that is the same percentage of the display, regardless of the display resolution. When the `turtle.window_height()` function is called without any parameters, it returns the pixel height of the current window. This value is used in the `setup` function to make the window square. Once the window is square, the coordinates are easily mapped without any concern of squishing or stretching.