

Introduction:

In the first NetLogo lab you learned how to control turtles and patches using the Command Center. In this lab you will learn how to control the interactions of patches and turtles by writing procedures. Procedures are groups of NetLogo *commands* or *reporters*. *Commands* are actions that patches or agents carry out. *Reporters* are commands that return a value. For example `random 40` returns a random number between 0 and 39. Together these are referred to as NetLogo *primitives*. An example of a simple procedure is given below:

```
to circle
  cct 1 [ pendown ]           ; create a turtle for drawing
  repeat 36 [
    ask turtles[ fd 5
                  rt 10 ] ] ; draw a circle
  clear-turtles             ; remove turtles
end
```

There are a couple of structural aspects of a procedure that you should notice. First, a procedure starts with a `to` and end with an `end`. Second, when you are programming it is important to indent, so that the code is easier to read. Please also get into the habit early of adding comments to your procedures so that it is clear to others what each component of your program does. Comments are added using semi-colons.

If you look at the commands in the procedure carefully you will see that this procedure actually draws a 36 sided polygon. Fortunately this is indistinguishable from a circle.

First Procedure:

Open up the file `Classic_Logo.nlogo`, which is in the handouts folder on Masu. This is simply the normal Netlogo interface, with the screen-size set to 200x200 and a button that will clear the screen and paint it white. Save your file with the naming convention

`NetLogo_Procedures_lastname_firstname.nlogo`

Remember to save your file often as you work. Click on the procedures tab. The `clear` procedure is already there. Notice what it does. Now type in the circle procedure given above. I recommend that you type it line by line, as this will help you become aware of syntax subtleties.

Return to the interface and type `circle` in the Command Center. Low and behold a circle appears. Try changing the number of turtles you create and see what happens.

To save having to type `circle` each time lets add a button to the interface that will call the circle procedure when pressed. Here's how to make the button:

1. Click on the button icon in the interface Toolbar
2. Click where you want the button to be in the empty white area of the Interface tab.
3. When the dialog box for editing the properties of the button opens, type `circle` in the box labeled "Commands".
4. Click OK and now you have a button that will draw a circle.

We can also add a slider to let us change the size of the circle we draw. Click on the slider icon on the interface Toolbar. And click on the empty white area where you would like to place it. Then enter `step-size` in the Global Variable box, set the minimum to be 0 and the maximum to be 10 and the increment to be 0.1 and the value to be 5. This defines a new variable called `step-size`, which we will use to change the size of the circle.

In the `circle` procedure replace the line `fd 5` with `fd step-size`. Now with your slider you can change the size of your circles easily.

Spiral Procedure:

There are a few more enhancement we could do with the `circle` procedure, but instead let's adapt it to draw a spiral. A spiral is really a circle that "misses", because its radius grows. We need to find some way to make the turtle move a little bit more each time it takes a step.

We can do this as follows.

Define a new `turtles-own` variable called `step` by putting the command

```
turtles-own [ step ]
```

at the top of the procedures window. This defines a variable `step` for each turtle. Each turtle will have its own value for the variable `step`. The variable `step-size` we defined earlier is a Global variable, which means that it doesn't belong to any particular turtle.

Click on the procedures tab, copy and paste the `circle` procedure to create a new procedure and rename it `spiral` and then make the following changes:

```
to spiral
  cct 1 [ set step step-size
          pendown ] ; create a turtle for drawing
  repeat 36 [
    ask turtles[ fd step
                 set step step * 1.03 ; draw a spiral which grows
                 rt 10 ] ] ; with factor 1.03 each repeat
  clear-turtles ; remove turtles
end
```

The line `set step step-size` initializes the variable `step` to be equal to the global variable `step-size` that is defined by the slider on the interface.

The line `step step * 1.03` makes the value of `step` grow by 3% once each repeat.

Add a `spiral` button and try it out. You can modify the procedure to make a longer spiral by increasing the number of repeats. You can change the growth rate by change the factor 1.03. If the spiral grows off the screen, you may want to set the initial `step-size` to be quite small.

There is a nice way of getting NetLogo to draw a spiral whenever and wherever you click the mouse. This is how you modify the `spiral` procedure:

```

to spiral
  if mouse-down?[                               ; draw a spiral if the mouse
                                                    ; is down
    cct 1 [ setxy mouse-xcor mouse-ycor         ; move the turtle to the
                                                    ; location of the pointer
            set step step-size
            pendown]                             ; create a turtle for drawing
  repeat 36 [
    ask turtles[ fd step
                 set step step * 1.03         ; draw a spiral which grows
                 rt 10 ] ]                     ; with factor 1.03 each repeat
  clear-turtles                                 ; remove turtles
]                                               ; close the if statement
end

```

Now back in the interface right click on the `spiral` button and check the box marked `forever` and press ok. This will make sure the procedure keeps checking to see if you have clicked the mouse. Press the spiral button to start the procedure and then click somewhere on the screen.

Finally we can enhance the beauty of this spiral by letting the pen-size and the color change as the spiral grows. After the `fd step` command inside the repeat loop of your procedure add the lines

```

set pen-size pen-size * 1.03
set color color + 10

```

Try it out. If you want to give specific initial values for the color and pen-size you can do this in the part of the procedure where you create your turtle. You can also change the growth factors for these variables with sliders.

Homework Assignment

Enhancing Spiral:

Add sliders for your spiral procedure that will allow the number of repeats (currently 36) to range from 0 to 200 in increments of 5. Add a slider that will allow the growth factor (currently 1.03) to change between 0.95 and 1.05 in increments of 0.01.

Branching:

We have talked about branching structures in the past few weeks. We can write a branching procedure that is remarkably similar to the spiral program you wrote in the tutorial.

1. Copy and paste the procedure above for `spiral` and rename the new copy `branch`. Here are the key differences:
 - a. Instead of turning right after each step have the turtle produce a bud (at some angle (say 45°). You can do this by replacing the `rt 10` command with a `hatch 1 [rt 45]` command.
 - b. Instead of new branches getting longer with each step they get smaller (say by a decay factor of a half). Adjust the `set step` and `set pen-size` command in your repeat loop accordingly.
 - c. Instead of repeating steps 36 times it will be enough to repeat 6 or 7 times. (Warning: your computer may freeze if you forget to do this.)

2. Make the changes suggested above and add a `branch` button to your interface. Try it out (you may need to make `step-size` larger). Happy?
3. To make your branch a bit more symmetric make the turtle hatch another bud on the other side. Try it out. Happy?
4. You will actually get a somewhat more realistic branch if you let the turtle take an additional step immediately after producing its left and right buds. Do this. And try it out? Happy?
5. Add the following enhancements to your model
 - a. Branches are usually brown. When you first create your turtle make it brown and also set the `pen-size` to 5.
 - b. Branches usually have green leaves at the end. Before clearing the turtles at the end of the procedure ask the turtles to change their color to green and get them to stamp their shape on the canvas.
 - c. Add variables with sliders for the `decay-factor` (range between 0.1 and 1), `repeat-number` (range from 1 to 10 repeats) and the `branching-angle` (range from 0 to 90).
6. After adding the enhancements try them out for a variety of settings. You should get remarkably different branches with small changes in these variables.
7. To make your branch procedure even more realistic instead of branching at a fixed angle branch at a random angle. You can do this by replacing `rt branching-angle` with `rt random branching-angle` (and like wise with the left branch).

Phyllotaxis

We also discussed the phenomenon of phyllotaxis in class. The basic process is quite simple. A seed is produced at the center and then moves a short distance away. Then another seed is produced at the center rotated by some divergence-angle relative to the previous one. Each new seed moves slightly further than the previous seed. You should be able to make a modified copy of your spiral procedure to do this. Here are some suggested modifications.

1. In this procedure you will not be drawing but stamping the shape of the turtle on the canvas to represent the seeds, so remove any `pendown` commands and replace instances of `pen-size` with `size`
2. In your `repeat` loop make the turtle `stamp` its shape after moving forward.
3. Instead of turning `rt 10` after taking a step have the turtle `hatch` a new turtle at the current position of the mouse (which is the center of the phyllotaxis pattern) with its turtles-own variable `heading` set to `heading + divergence-angle` (where you define `divergence-angle` with a new slider on the interface).
4. After a turtle has stamped its location and hatched a new seed have the turtle `die`.

Try out these changes in a procedure called `phyllotaxis`. Play around with different `step-size` (small works best) `growth-factor` (between 1 and 1.03 works best) and divergence angles. Try out different colors and shapes for your turtles.

When you have finished make a copy of your program and paste it into the drop box.