Exercise 2.1 – A first NetLogo program

Open NetLogo

Click on the “Code” tab at the top-middle-right of the tool bar.

This is where you can type and edit NetLogo programs. Programs are composed of procedures. Most NetLogo programs contain at least two procedures called “setup” and “go”.

Type in the following program which defines these two procedures:

```
to setup
  clear-all
  create-turtles 100 [
    setxy random-pxcor random-pycor
    set heading random 360
  ]
  reset-ticks
end

to go
  ask turtles [
    forward 0.1
    if count turtles-here > 1 [
      right random 360
      forward 1
    ]
  ]
  tick
end
```

The setup procedure clears the environment and creates 100 turtles placing each in a random location and pointing them in random directions.

The go procedure asks each turtle to move forward a short distance (one tenth of a patch) and then check if any other turtles are in the same patch as itself. If this is so then the turtle changes to a random direction and moves forward the length of a patch.

Go back to the Interface screen by clicking on the “Interface” tab at the top-middle-left of the toolbar.

Since both setup and go are now defined you could type them directly into the command centre. However it is convenient to create two buttons to execute them.

Task: Create two buttons. Make one execute the “setup” procedure and the other execute the “go” procedure. When creating the go button check the “forever” checkbox at the top-left of the button dialogue. This means that when you press the go button it will repeatedly execute the go procedure until it is pressed again.

Click the setup button you just created.

You should see a set of randomly generated agents in the environment.

Click the go button you just created.

You should see the agents moving about very quickly.

To stop the program click the go button again.

In order to see the movement more clearly we can force display updates to happen on each tick of the model. Note that in the procedure “go” we put a “tick” command at the end.

On the toolbar under the “view updates” checkbox select “on ticks” from the dropdown.

Click the go button to run the program.

Now you should be able to see the turtles moving in straight lines until they hit each other (enter an occupied patch) and then they change to a random direction and move forward one patch distance.

To stop the program click the go button again.

Task: Create a slider that ranges from 1 to 1000 assigned to the global variable “popsize”. Go to the Code screen and modify the setup function so that it creates “popsize” turtles rather than 100.

Go back to the Interface screen and test different popsize values by changing the popsize slider and clicking on setup.

Exercise 2.2 – Birth and death of turtles

We will modify the program so that instead of turtles changing direction when they move into an occupied patch they die.

Go to the Code screen and modify the go procedure in the following way:

```
to go
  ask turtles [
    forward 0.1
    if count turtles-here > 1 [die]
  ]
  tick
end
```

The die command immediately deletes the turtle that executes it.

Go back to the Interface screen and execute the program to verify that turtles now die when they enter an occupied patch.

We will modify the program so that each turtle “gives birth” with some probability.

Task: Create a slider that ranges from 0 to 1 in increments of 0.01 assigned to the global variable “birthprob”.

Go to the Code screen and modify the go procedure in the following way and add the new function “to-report prob[x]”:

```
to go
  ask turtles [
    forward 0.1
    if count turtles-here > 1 [die]
  ]
  ask turtles [
    if prob birthprob [
      hatch 1 [
        set heading random 360
        forward 1]
      ]
    ]
  tick
end

to-report prob[x]
  report (random-float 1 < x)
end
```

Note: In the go procedure we have added another ask turtles command which makes each turtle hatch one new turtle (which is a clone of itself inheriting all the same values other than the who value – remember who is always a unique number for each turtle) with birthprob probability.

The commands after the hatch command are immediately executed by the newly hatched turtle. In this case the turtle points in a random direction and moves forward one patch length.

We have also defined a new procedure called “prob” using the “to-report” form. This defines a “reporter procedure” that takes an argument “x” and returns a single value (in this case a true / false value) using the report command. Here then we see how to pass arguments and how to return values. Note: procedures that do not return values, such as setup and go, are called “command procedures”.

The prob reporter procedure returns true with probability equal to its argument. Hence “prob 1” is always true and “prob 0” is always false. The random-float command produces a random float between 0 but strictly less than it’s argument (here being 1).

Go back to the interface screen and experiment with different birthprob settings. Note, you can change this value while the problem is running by moving the slider. Global variables change immediately when they are changed on the interface screen.

Exercise 2.3 – Adding outputs to the interface

So far we have added inputs in the form of sliders and buttons to the interface. However we can also add outputs such as charts and values.

We will add a dynamic chart that displays the population size over time.

From the Interface screen:

Select “Plot” from “+Add” dropdown.

Click on a blank area. A plot dialogue should appear. In the name field type:

```
population size
```

In the “pen update commands” list make sure the following is entered:

```
plot count turtles
```

Note: this may already appear as the default value.

Click OK

Task: Experiment with different initial population sizes and birthprob values notice how the plot changes. Try starting the population size at 1 and the birthprob at 0 and then slowly increasing birthprob as the program runs to see the effect on the population size.

We will add an output that prints the population size.

Select “Monitor” from the “+Add” dropdown.

Click on a blank area. A Monitor dialogue should appear. In the “Reporter” field enter:

```
count turtles
```

Click OK

Try running the program again. The monitor should display the dynamic population size.

Exercise 2.4 – Adding variables to turtles and global variables to programs

In the Code screen add the following at the very top of the program:

```
globals [deaths]

turtles-own [age]
```

Note: the globals command specifies a list (in square brackets) of global variables for the entire program. The turtles-own command specifies a list of variables local to each turtle.

We will update the program so that for each turtle age variable is initialised to 0 and incremented for each tick. In addition we will initialise deaths to 0 at the start of each go procedure call and increment it for each death that occurs.

Modify the setup and go procedures as below:

```
to setup
  clear-all
  create-turtles popsize [
    setxy random-pxcor random-pycor
    set heading random 360
    set age 0
  ]
  reset-ticks
end

to go
```

```
  set deaths 0
  ask turtles [
    set age age + 1
    forward 0.1
    if count turtles-here > 1 [
      set deaths deaths + 1
      die]
  ]
  ask turtles [
    if prob birthprob [
      hatch 1 [
        set age 0
        set heading random 360
        forward 1]
      ]
    ]
  tick
end
```

Go to the Interface screen. We will add a new Monitor to show the mean (average) age of the population.

Select “Monitor” from the “+Add” dropdown.

Click on a blank area of the screen.

In the Monitor dialogue type the following in the “Reporter” field

```
mean [age] of turtles
```

In the “Display Name” field type:

```
age
```

Click OK.

Note: the “of” reporter returns a list containing each age turtle variable and the mean reporter calculates the mean of a list.

Try running and the program and varying birthprob to see how it changes the mean turtle age.

We can also create a histogram showing the distribution of a variable. We will create one for the age turtle variable. Create a new plot output on a blank part of the screen.

In the plot dialogue in the “X max” field type:

```
200
```

Under “Pen update commands” type:

histogram [age] of turtles

Click on the pen icon to the left of what you just typed. This will open a pen dialogue. From the “Mode” dropdown select “Bar”. In the Interval field type:

```
10
```

Click OK on both dialogues. You should now see histogram with 10 bins. Try running the program to see the output. Try setting popsize at 1 and birthprob at 0.01. Run the program and then gradually increase birthprob to see the effect.

Task: Create a new Monitor that displays the deaths global variable. Create two new plots for the age and deaths. Test them by running the program.

Task: Add a new global variable called “births” to the program and modify the go procedure so it records how many births happen in each cycle. Create a new plot that displays births over time. Create a new monitor that displays the births value. Test it by running the program.