**ABM – Week 4 – Seminar – LVL1**

**Purpose**

This task will allow you to code simple interactions between turtles and patches. It will also prepare you for Coursework 2, in which you will use a similar model.

**Model**

Open the model tutorial\_4\_starting\_model.nlogo. This model provides a basic framework as a starting point for this task.

**Task**

Create a model where…

* Patches contain food that turtles can eat, with each patch having a maximum amount of food that it can contain (between 0 and 100). Food starts at its maximum value and grows back at a rate of 1 unit per tick if it is depleted.
* Each turtle (initial population 100) has an energy level (initially 100), which reduces by a fixed amount (between 2 and 10, varying between turtles) at every tick. Turtles can consume the food on their patch at a rate of 5 units per tick to replenish their energy. Turtles with zero energy will die.
* If a neighbouring patch contains more food than their current patch, turtles move to the patch with more available food.

Create graphs to monitor how key quantities change as the simulation runs.

**Simple Extensions**

1. There are several fixed values in the model (e.g. turtles can eat a maximum of 5 units of food per tick; food grows back at a rate of 1 unit per tick; etc.). Replace some of these fixed values with variables and create sliders in the Interface for these variables. Investigate how the behaviour of the model changes as these variables are changed.
2. Rather than distribute food randomly, create “hills” of food, like the geography used in the butterflying hilltopping model. How does this affect the behaviour of the model?

**Step-By-Step Guide**

*After each change you make, check that the code runs before you move on to the next step…*

Creating the environment

1. Add two patch variables to patches-own to record the *maximum* food that a patch can contain and the *current* amount of food at a patch.
2. In the setup procedure:
   1. Set the maximum food of each patch to a value between 0 and 100

*[Useful primitives: ask, patches, random, set]*

* 1. Set the current amount of food at each patch to be equal to the maximum.

1. Colour the patches by their current amount of food.

*[Useful primitives: pcolor, scale-color, set]*

Now, when you click the setup button, the world should turn into a patchwork of different shades of colour. If you inspect a patch, you should see how much food it has and the maximum food that it can store.

Creating the agents

1. Add two turtle variables to turtles-own to record the current energy level of a turtle and the amount of energy that the turtle uses at each tick.
2. In the setup procedure, create 100 turtles.  
   *[Useful primitives: crt]*
3. In the create turtles block:
   1. Set the current energy of each turtle to 100 and the energy use per tick to a random number between 2 and 10.  
      *[Useful primitives: set, random]*
   2. Place each turtle on a random patch.  
      *[Useful primitives: random-pxcor, random-pycor, setxy]*
   3. Set the colour and size of the turtles to make them easy to see against the coloured patches.  
      *[Useful primitives: color, set, size]*

Now, when you click setup, you should see the turtles spread across the patches.

Adding a behaviour: Consuming food

1. Create a new procedure to model turtles consuming food from their patch. The procedure should be empty for the moment. Add the name of this procedure to the go procedure, so that it runs every tick.
2. In the new procedure, create an “ask turtles” block to move into turtle context.

Inside the “ask turtles” block…

1. … write a line that creates a temporary variable called meal\_size equal to the minimum of (i) 5 and (ii) the current food on the turtle’s patch.

*[Hint: A turtle can read and set the variables of the patch it is on.  
Useful primitives: let, list, min]*

1. … write lines that increase the turtle’s energy by meal\_size and decrease the food available on the turtle’s current patch by the same amount.

1. At the end of the go procedure, add a line to recolour the patches, based on their current food available.  
   *[Useful primitives: ask, patches, pcolor, scale-color, set.]*

When you run your simulation, turtles should eat all the food on their patches, which will turn black as their food is exhausted.

Adding a behaviour: Food growback

1. Create a new procedure to model the food on the patches growing back. The procedure should be empty for the moment. Add the name of this procedure to the go procedure, so that it runs every tick
2. In the new procedure, create an “ask patches” block to move into patch context.

Inside the “ask patches” block…

1. … write a line that sets the available food on the patch to the minimum of (i) one more than the current amount of food and (ii) the maximum food that the patch can store.  
   *[Useful primitives: let, list, min]*

There will be no obvious change in the simulation if you run it now, though, if you inspect a turtle, you will notice that their energy increases indefinitely, because the food on their patch is continuously growing back.

Adding a behaviour: Using energy

1. Create a new procedure to model the energy use of turtles. The procedure should be empty for the moment. Add the name of this procedure to the go procedure, so that it runs every tick.
2. In the new procedure, create an “ask turtles” block to move into turtle context.

Inside the “ask turtles” block…

1. … write a line to reduce a turtle’s current energy by the amount that they use each tick.
2. … write a line that kills a turtle whose energy drops to zero or below.  
   *[Useful primitives: die, if]*

Now, when the simulation runs, the turtles will gradually die as they exhaust the food on their patch. The patches will then regrow their food to their maximum values.

Adding a behaviour: Moving to the most abundant food source

1. Create a new procedure to model the movement of turtles to find the most abundant food source in their neighbourhood. The procedure should be empty for the moment. Add the name of this procedure to the go procedure, so that it runs every tick.
2. In the new procedure, create an “ask turtles” block to move into turtle context.

Inside the “ask turtles” block…

1. … write a line to move a turtle to the patch in its neighbourhood with the most available food.

*[Useful primitives: uphill]*

Now, when your simulation runs, turtles will move around to find more food, with food growing back on the patches behind them. Some turtles will die when they cannot find enough food.

Creating graphs to monitor the population of turtles and their mean energy use per tick

1. Create two plots in the Interface. Edit the plots to monitor (i) the number of turtles and (ii) the mean energy use per tick of the surviving turtles.

*[Useful primitives for the first plot: count, plot, turtles*

*Useful primitives for the second plot: mean, of, plot, turtles]*

1. Give the plots appropriate titles and axis labels. Set the axes to appropriate scales.

Your simulation should now run as before, but the plots will update as time passes.