Unit 4 – Solving Complex Problems

\*\*Instructions:  Please change the text color of your responses to red text.  Please organize the endings to each page.

ACTIVITY 4.1.1 – Simulations in Science

GOALS:

* Use simulations to learn about complex systems that are too difficult to observe in the real world.
* Gain insight and knowledge using the iterative and interactive nature of models and simulations.
* Derive meaning knowing the inclusion or exclusion of certain elements is an important part of the model/simulation.

You will be using NetLogo for this assignment.  Please make sure that you have it installed on your laptop.    
  
Complete the following:

Step 9: Using a faster speed, run the simulation a few times at each of the following densities and note how often the fire reaches the right side of the world and the percent burned at each density.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Density | How Often Fire Reaches Right Side of World | | | | % Burned |
| Never | Sometimes | Often | Always |
| 57% | Yes |  |  |  | 10% |
| 58% | Yes |  |  |  | 25% |
| 59% |  |  | Yes |  | 60% |
| 60% |  |  |  | Yes | 70% |

Based on your data, what conclusions can you make about the relationship between tree density and the amount of the forest that would burn?

|  |
| --- |
| A higher tree density results in a larger proportion of the forest that is burned. |

Step 12: Run the simulation until you see a blank (green) world where both sheep and wolves have died out.  Using the plot of data, describe any correlations between the sheep population and the wolf population.

|  |
| --- |
| The sheep population and wolf population are inversely correlated, as when wolf populations increase sheep populations decrease, while when sheep populations increase, it is because wolf populations have decreased. However, when the sheep population increases, the wolf population increases soon after. |

Why do you suppose the wolf population eventually dies out, even with an abundance of sheep?

|  |
| --- |
| The wolf population always grows too large to support itself, growing faster than the number of sheep. |

Step 15: In the NetLogo control bar to the right of the ticks slider, uncheck view updates. The world will stop showing changes, but the simulation is still running. Allow the simulation to run for about 20,000 ticks and then stop the simulation by clicking the go button (the button is a toggle.)  What correlations can you make between the sheep population and the wolf population now?

|  |
| --- |
| The sheep population and wolf population still inversely fluctuate, in an almost-sinusoidal pattern. |

Make a hypothesis: Why do you think the system is no longer unstable?

|  |
| --- |
| The system is no longer unstable because the sheep’s dependence on grass stops them from growing uncontrollably, which also limits the population of the wolves, which stops the system from being so unstable. |

With a hypothesis, make a prediction: Predict what will happen when you increase the grass regrowth time.

|  |
| --- |
| Increasing the grass regrowth time will cause the maximum population of all species to decrease. |

Step 16: Experiment with grass-regrowth-time to support or contradict your hypothesis.  How does changing the grass regrowth time support or contradict your hypothesis and prediction?

|  |
| --- |
| Increasing the grass regrowth time decreases the population size of all species, which supports my hypothesis. However, it also causes the wolf population to go extinct, and makes the fluctuations in population size smaller for all species. |

Step 17:  Other variables can affect the stability of the simulation. While the simulation is running, experiment with both Sheep and Wolf settings sliders to see how those variables affect the simulation.  How do the Sheep settings and Wolf settings variables affect the stability of this system?

|  |
| --- |
| Increasing the initial sheep setting causes the wolf population to overshoot its carrying capacity and almost die out, but then return to equilibrium where all 3 populations fluctuate.  Increasing the initial wolf setting has the same effect, except the simulation starts on the overshoot, instead of before it. |

Step 25:  Insert your code below and run the simulation, confirming the simulation now has three turtles of varying color.

|  |
| --- |
| 3 different turtles are created, with different colors, moving in random directions.  to setup  clear-all  create-turtles 3 ; create one turtle  [  set color one-of base-colors  pen-down  ]  reset-ticks  end  to go  ask turtles [  rt random 360 ; set random heading  forward 1 ; advance one step  ]  tick  end  ; Public Domain:  ; To the extent possible under law, Uri Wilensky has waived all  ; copyright and related or neighboring rights to this model. |

Step 27:  Change your code so that the turtle turns less than 360 degrees and moves more or less than one step. Decimal values are acceptable.

|  |
| --- |
| The turtles now move in circular patterns.  to setup  clear-all  create-turtles 3 ; create one turtle  [  set color one-of base-colors  pen-down  ]  reset-ticks  end  to go  ask turtles [  rt random 30 ; set random heading  forward 5 ; advance one step  ]  tick  end  ; Public Domain:  ; To the extent possible under law, Uri Wilensky has waived all  ; copyright and related or neighboring rights to this model. |

Step 31:  Insert your code and run your simulation, experimenting with turn-amount. Be sure to test the largest and smallest values for turn-amount.

|  |
| --- |
| to setup  clear-all  create-turtles 3 ; create one turtle  [  set color one-of base-colors  pen-down  ]  reset-ticks  end  to go  ask turtles [  rt random turn-amount ; set random heading  forward 1 ; advance one step  ]  tick  end  ; Public Domain:  ; To the extent possible under law, Uri Wilensky has waived all  ; copyright and related or neighboring rights to this model. |

Step 41: Insert your code and test your simulation to see your new shape in action.

|  |
| --- |
| turtles-own [  flockmates ;; agentset of nearby turtles  nearest-neighbor ;; closest one of our flockmates  ]  to setup  clear-all  create-turtles bird-population  [ set color green - 2 + random 7 ;; random shades look nice  set size 1.5 ;; easier to see  setxy random-xcor random-ycor  set flockmates no-turtles  set shape "bird side"  ]  reset-ticks  end  to go  ask turtles [ flock ]  ;; the following line is used to make the turtles  ;; animate more smoothly.  repeat 5 [ ask turtles [ fd 0.2 ] display ]  ;; for greater efficiency, at the expense of smooth  ;; animation, substitute the following line instead:  ;; ask turtles [ fd 1 ]  tick  end  to flock ;; turtle procedure  find-flockmates  if any? flockmates  [ find-nearest-neighbor  ifelse distance nearest-neighbor < minimum-separation  [ separate ]  [ align  cohere ] ]  end  to find-flockmates ;; turtle procedure  set flockmates other turtles in-radius vision  end  to find-nearest-neighbor ;; turtle procedure  set nearest-neighbor min-one-of flockmates [distance myself]  end  ;;; SEPARATE  to separate ;; turtle procedure  turn-away ([heading] of nearest-neighbor) max-separate-turn  end  ;;; ALIGN  to align ;; turtle procedure  turn-towards average-flockmate-heading max-align-turn  end  to-report average-flockmate-heading ;; turtle procedure  ;; We can't just average the heading variables here.  ;; For example, the average of 1 and 359 should be 0,  ;; not 180. So we have to use trigonometry.  let x-component sum [dx] of flockmates  let y-component sum [dy] of flockmates  ifelse x-component = 0 and y-component = 0  [ report heading ]  [ report atan x-component y-component ]  end  ;;; COHERE  to cohere ;; turtle procedure  turn-towards average-heading-towards-flockmates max-cohere-turn  end  to-report average-heading-towards-flockmates ;; turtle procedure  ;; "towards myself" gives us the heading from the other turtle  ;; to me, but we want the heading from me to the other turtle,  ;; so we add 180  let x-component mean [sin (towards myself + 180)] of flockmates  let y-component mean [cos (towards myself + 180)] of flockmates  ifelse x-component = 0 and y-component = 0  [ report heading ]  [ report atan x-component y-component ]  end  ;;; HELPER PROCEDURES  to turn-towards [new-heading max-turn] ;; turtle procedure  turn-at-most (subtract-headings new-heading heading) max-turn  end  to turn-away [new-heading max-turn] ;; turtle procedure  turn-at-most (subtract-headings heading new-heading) max-turn  end  ;; turn right by "turn" degrees (or left if "turn" is negative),  ;; but never turn more than "max-turn" degrees  to turn-at-most [turn max-turn] ;; turtle procedure  ifelse abs turn > max-turn  [ ifelse turn > 0  [ rt max-turn ]  [ lt max-turn ] ]  [ rt turn ]  end  ; Copyright 1998 Uri Wilensky.  ; See Info tab for full copyright and license. |

Step 50:  Insert your code and test the simulation to confirm that remaining raindrops evaporate when the rainfall is 0.

|  |
| --- |
| breed [waters water]  breed [raindrops raindrop]  patches-own [  elevation  ]  globals [  color-min  color-max  old-show-water?  water-height ;; how many feet tall one unit of water is  border ;; keep the patches around the edge in a global  ;; so we don't ever have to ask patches in go  ]  ;;  ;; Setup Procedures  ;;  ;; reading the external file is startup rather  ;; than setup so we only do it once in the model  ;; running the model does not change the elevations  to startup  ;; read the elevations from an external file  ;; note that the file is formatted as a list  ;; so we only have to read once into a local variable.  file-open "Grand Canyon data.txt"  let patch-elevations file-read  file-close  ;; put a little padding on the upper bound so we don't get too much  ;; white, and higher elevations have a little more variation.  set color-max max patch-elevations + 200  let min-elevation min patch-elevations  ;; adjust the color-min a little so patches don't end up black  set color-min min-elevation - ((color-max - min-elevation) / 10)  ;; transfer the data from the file into the sorted patches  (foreach sort patches patch-elevations [ [the-patch the-elevation] ->  ask the-patch [ set elevation the-elevation ]  ])  set-default-shape turtles "circle"  setup  end  ;; just clean up the marks that the raindrops have made  ;; and set some global variable to defaults  to setup  clear-drawing  clear-turtles  ask patches  [ set pcolor scale-color brown elevation color-min color-max ]  set water-height 10  set border patches with [ count neighbors != 8 ]  reset-ticks  end  ;;  ;; Runtime Procedures  ;;  to go  ;; check for mouse clicks on empty patches.  ;; if we've got a winner make a manual raindrop that  ;; is red.  if mouse-down? and not any? turtles-on patch mouse-xcor mouse-ycor  [  ;; even when raindrops are hidden  ;; newly created manual drops will  ;; be visible  create-raindrops 1  [ setxy mouse-xcor mouse-ycor  set size 2  set color red  ]  ]  ;; make rain-rate drops randomly  create-raindrops rain-rate  [ move-to one-of patches  set size 2  set color blue ]  ifelse draw?  [ ask turtles [ pen-down ] ]  [ ask turtles [ pen-up ] ]  ask raindrops [ flow ]  ask border  [  ;; when raindrops reach the edge of the world  ;; kill them so they exit the system and we  ;; don't get pooling at the edges  ask turtles-here [ die ]  ]  ;; evaporation  ask n-of (evaporation \* count turtles) turtles [ die ]  tick  end  to flow ;; turtle procedure  ;; get the lowest neighboring patch taking into account  ;; how much water is on each patch.  let target min-one-of neighbors [ elevation + (count turtles-here \* water-height) ]  ;; if the elevation + water on the neighboring patch is  ;; lower than here move to that patch.  ifelse [elevation + (count turtles-here \* water-height)] of target  < (elevation + (count turtles-here \* water-height))  [ move-to target ]  [ set breed waters ]  end  ; Copyright 2006 Uri Wilensky.  ; See Info tab for full copyright and license. |

Assuming that rain rate and evaporation rate reflect the correct real-world rates, how well does the evaporation algorithm fit the model?

|  |
| --- |
| The evaporation algorithm does not fit the model very well, as sometimes, water gets stuck on top of a mountain, where it would normally evaporate quickly in the real world, but takes a long time to evaporate in the model. |