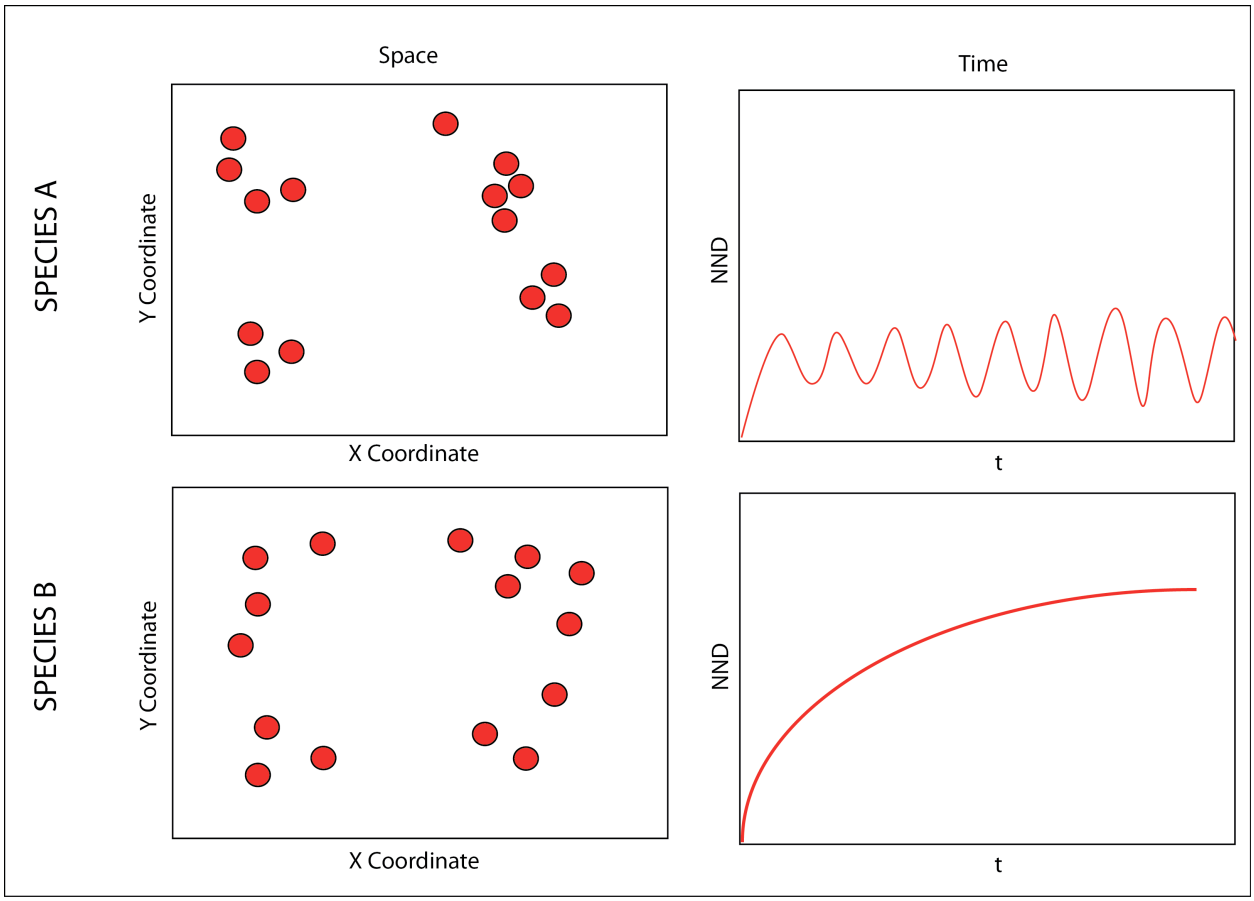
GEOG 490/590 I SPATIAL MODELING I SPRING 2015

ASSIGNMENT 3: PATTERN-ORIENTED MODELING WITH AGENTS

Objective: To determine a process that produces a particular spatial pattern.

Description: An ecologist studying the movement patterns of two species (A and B) is interested to know what underlying processes are driving observable locational patterns. Below are descriptions of the observed spatial and temporal patterns of both species. You are responsible for evaluating three models of movement to determine if they describe the observed patterns. The models are (1) flocking, (2) foraging, and (3) random movement. You will follow the instructions below to create a single NetLogo model that provides options to test these three models. Using videos and text, you will provide an answer to the question of which of the three movement theories best describes the spatial and temporal patterns of both species.

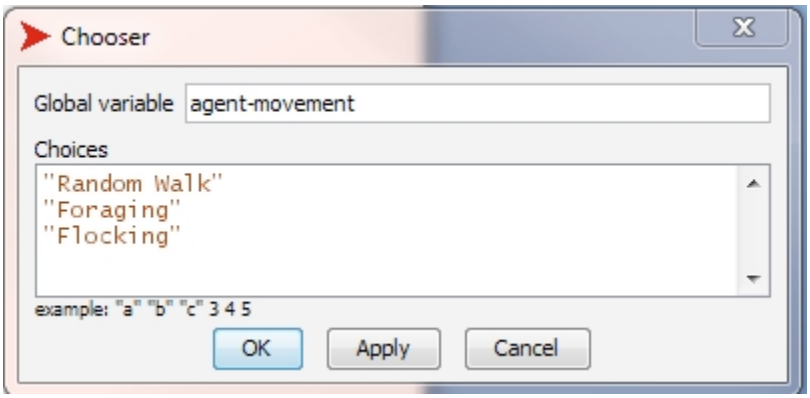


INSTRUCTIONS

PART 1: SETTING UP YOUR MODEL

1. Create new model.

2. Create two buttons: `go` and `setup`. Create a chooser that looks like this:



3. Initialize the model using the `setup` button. Under the setup code, type the following:

```
to setup
  clear-all
  setup-patches
  setup-turtles
  reset-ticks
end
```

4. In the `setup-patches` method, create 5 resources patches in random locations in your landscape (HINT: use the "`ask n-of` *x* `patches`" function). Set the color of your random resources green.

5. In the `setup-turtles` method, use the command "`crt 20`" to put 20 turtles on the patch with coordinates 0,0. Set the turtle color to blue. Don't change the shape of your turtles as it is important to have a shape that will indicate the direction the turtles are facing.

6. Create a button called `turtle-movement`. Under `to go` method, use if statements to call a specific type of turtle movement theory:

```
to go
  if turtle-movement = "Random Walk" [random-walk]
  if turtle-movement = "Foraging" [forage]
  if turtle-movement = "Flocking" [flock]
  tick
end
```

PART 2: CREATE RANDOM WALK MODEL

7. Create a new method in your model called `random-walk`.

8. Create a switch called `Directed-Walk?` that will allow you to toggle between a random and a directed walk. Next, in the `random-walk` method, insert the following code:

```
ask turtles [
  ifelse (Directed-Walk?)
    [rt random 90 lt random 90]
    [rt random 360]
  forward .25
]
```

Can you explain what this code accomplishes?

PART 3: CREATE FORAGING MODEL

9. Create a new method called `forage` and insert the following code:

```
ask turtles [
  ifelse (Directed-Walk?)
    [rt random 90 lt random 90]
    [rt random 360]
  forward .25
]
```

Can you explain what this code accomplishes?

10. In this model the turtles will gain energy from the food as represented by the green patches. To do this, the turtles need to own a variable called energy and the patches need to own a variable named food (this code should be inserted at the top of the code panel):

```
turtles-own [
  energy
  ]
patches-own [
  food
  ]
```

11. In the `setup-turtles` method, set the turtles' initial energy to 500 using the following code.

```
ask turtles [set energy 500]
```

12. The turtles will walk around the landscape looking for food. If the turtle is on a green "food resource" turtle energy will return to 500. The code needs to be within the ask turtles block of code.

```
if pcolor = green[
  set energy 500
```

13. As of now, nothing happens if the turtles run out of energy. Rather than have them die, you will code the turtles to make them move directly to a food resource when energy = 0.

```
set energy energy - 1
if energy = 0 [
  move-to min-one-of patches
    with [pcolor = green] [distance myself]
  ]
```

PART 4: CREATE FLOCKING MODEL

14. Add two more turtles-own variables at the beginning of the model called `flockmates` and `nearest-neighbor` .

```
turtles-own [
  energy
  flockmates
  nearest-neighbor
  ]
```

15. Paste the following at the end of your code

```
to flock
  ask turtles [ flocking ]
  repeat 5 [ ask turtles [ fd 0.2 ] display ]
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;
  to flocking
    find-flockmates
    if any? flockmates
      [ find-nearest-neighbor
        ifelse distance nearest-neighbor < 1
          [ separate ]
          [ align
            cohere ]
          ]
  end

  to find-flockmates
    set flockmates other turtles in-radius 3
  end

  to find-nearest-neighbor
    set nearest-neighbor min-one-of flockmates [distance
  myself]
  end

  to separate
    turn-away ([heading] of nearest-neighbor) 1.5
  end

  to align
    turn-towards average-flockmate-heading 5
  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

  to cohere
    turn-towards average-heading-towards-flockmates 3
  end

  to-report average-heading-towards-flockmates  ;;
  turtle procedure
    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

  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

  to turn-at-most [turn max-turn]
    ifelse abs turn > max-turn
      [ ifelse turn > 0
          [ rt max-turn ]
          [ lt max-turn ] ]
      [ rt turn ]
  end
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

PART 5: CREATE MONITOR FOR MEASURING NEAREST NEIGHBORS

16. Add a monitor and plot to capture the mean nearest neighbor of the turtles in the graph. This plot will show the distance to the closest turtle, and can be a powerful way to link the emergent pattern of the model to the underlying processes. First, create a new turtle variable called `nearest-neighbor-distance`. You will update this variable at the end of each timestep. Make the final line of the `random-walk`, `forage`, and `flock` methods `update-plot`.

17. At the end of the model, copy the following code:

```
to update-plot
  ask turtles [
    let nd min-one-of other turtles [distance
myself]
    set nearest-neighbor-distance distance nd
  ]
end
```

NOTE: "`nd`" is a temporary variable. It's value is the name of the turtle which is closest to it in the model environment. The variable `nearest-neighbor-distance` is simply the distance to the closest turtle.

18. Finally create a plot and monitor to show the mean value of the `nearest-neighbor-distance` variable. Refer to the traffic grid model in the model library for the code syntax for plotting a turtle variable.

PART 6: DISCUSSION

19. Create an Assignment 3 page. Provide videos and text that address the following questions:

    i. Describe in 4-5 sentences the utility of using a pattern oriented modeling approach for understanding the observed patterns of both species.
    ii. For each movement model, describe how agent interactions lead to emergent patterns in both space and time.
    iii. Which movement model best describes the patterns observed for species A? Why?
    iv. Which movement model best describes the patterns observed for species B? Why?
    v. How did a pattern oriented modeling approach allow you to determine the answers to (iii) and (iv)?

GRADING

Your answer to each question in Part 6	**5 POINTS**
TOTAL	**25 POINTS**

DUE DATE:
Tuesday, April 21th at 11:59pm
*Late submissions will be penalized 5% per day.