11.4 Modeling an "Able" Invader—the "Cane" Toad

NetLogo Quick Review Questions

Introduction to Computational Science:
Modeling and Simulation for the Sciences, 2nd Edition

Angela B. Shiflet and George W. Shiflet
Wofford College

***Compose all the following answers in NetLogo*:**

Phase 0: Initialization

Quick Review Question 1 Declare a breed of *toads*. Start the function *initToads* to set the shape of *toads* to be "toad".

Quick Review Question 2 Declare that toads own *energy*, *water*, *state*, *numTimeSteps*, *lastx*, and *lasty*. Write a statement for *initToads* to create *NUM_TOADS* number of initialized toad agents.

Quick Review Question 3 Suppose *PERCENT_AWPS* is 0.3, *PERCENT_AWPS_FENCED* is 25, and *PERCENT_MOIST_AREAS* is 0.1 and the grid is 100-by-40 cells. On the average, after the initialization phase how many of the following would we expect on the grid:

a. *Awp* agents before initialization of *FencedAwp* agents
b. *FencedAwp* agents
c. *Awp* agents after initialization of *FencedAwp* agents
d. *MoistArea* agents
e. If there are 5 *Awp*, 2 *FencedAwp*, and 3 *MoistArea* agents, none of which are next to a border or each other, how many *AwpAdjacent* agents are there?

Quick Review Question 4 Assume *SIDE* is the length of one side of the square landscape. Suppose the function *initMoisture* calls *initDesert* and *initAwps*.

a. Write *initFood* to initialize the food value for each patch.
b. Write *initDesert*, as follows: If a patch is in the interior, make its class be *DESERT* and have its color be grey and its *scale-color* be proportional to its amount of food, *food*, on a scale from (*FOOD_CELL* * 2) down to 0. Otherwise, make its class be *BORDER* and its color be *grey* – 3. For a patch on the east border, set its *plabel* to "-". For a patch on one of the other borders, set its *plabel* to "!".
c. Write *initAwps* to initialize the AWPs, fenced AWPs, and surrounding areas.

Phase 1: Consumption

Quick Review Question 5 Suppose *AMT_EAT* = 0.01 and *FRACTION_WATER* = 0.6. Assume a toad is on top of a desert cell. Give the values of a toad's *energy* and *water* and a desert cell's *food* after execution of *eat* and *updateFood* for each of the following situations:

a. *energy* = 0.9, *water* = 0.8, and *availableFood* = 0.03
b. *energy* = 0.9, *water* = 0.8, and *availableFood* = 0.005
c. *energy* = 0.999, *water* = 0.8, and *availableFood* = 0.03
d. *energy* = 0.9, *water* = 0.999, and *availableFood* = 0.03

Quick Review Question 6 Write the following consumption functions:

a. *Toad* function *toadMayEat*
b. *Toad* function *eat*, which also updates the amount of food in the cell
c. *Toad* function *toadMayDrink*
d. *Toad* function *drink*

Phase 2: Movement

Quick Review Question 7 Write the toad function *toadMove*.

Quick Review Question 8 Write the following functions related to movement for moisture:

a. *thirsty*
b. *lookForMoisture*
c. *moveW*
d. *useWaterEnergyHopping*

Quick Review Question 9 Write the *Toad* method *lookForFood*.

Quick Review Question 10 Write the *Toad* functions *hopForFun*. Note that *hopHere* is implemented with *useWaterEnergyHopping*.

Phase 3: Complete Cycle

Quick Review Question 11 Write the following:

a. *changeCounts*
b. *checkTerminate* implemented as *terminate?*, a function that returns *true* or *false* depending on whether the simulation should terminate or not
c. A statement in *go* that terminates the simulation if *terminate?* returns *true*

Answers to Quick Review Questions

1.

```
breed [ toads toad ]

to initToads
  set-default-shape toads "toad"
```

2.

```
toads-own [ energy water state numTimeSteps lastx lasty ]

  create-toads NUM_TOADS [
    set size 1.5
    set state ALIVE
    set lastx -1
    set lasty -1
    set energy min list max list 0 random-normal MEAN_ENERGY STD_ENERGY
1
    set water min list max list 0 random-normal MEAN_WATER STD_WATER 1
    set numTimeSteps 0
    set color cyan
    set xcor SIDE - 1
    set ycor SIDE - 1 - random (SIDE * 2 - 1)
    set heading -90
  ]
```

3. **a.** 12 = (0.003)(100)(40)

b. 3 = (0.25)(12), where 12 is obtained from Part a

c. 9 = 12 - 3

d. A little less than 4 = (0.001)(100)(40), because immediately before initialization of moist areas, some of the (100)(40) = 4000 *Desert* agents have likely been converted to *Awp* and/or *FencedAwp* agents

e. 56 = (8)(5 + 2) because each *Awp* and *FencedAwp* agent is surrounded by 8 *AwpAdjacent* agents.

4. **a.**

```
to initFood
  ask patches [
    ifelse (max list (abs pxcor) (abs pycor) = SIDE)
    [ set food -1
      set moisture -1 ]
    [ set food FOOD_CELL ]
  ]
end
```

4. **b.**

```
to initDesert
    ask patches [
    ifelse ( (abs pxcor) < SIDE and (abs pycor) < SIDE ) [
      set class DESERT
      set pcolor scale-color grey food (MAX_FOOD * 2) 0
           ;Have the color of the patch reflect the amount of food
    ] [
      set class BORDER
      set pcolor grey - 3
      ifelse ( (abs pycor) = SIDE )
      [set plabel "-"] [set plabel "|"]
    ]
  ]
```

```
end
```

4. c.

```
to initAwps
    ask patches [
    if ((abs pxcor) < SIDE - 2 and (abs pycor) < SIDE - 2 and
           8 = count neighbors with [class = DESERT] and
           random-float 1 < (PERCENT_AWPS / 100)) [
      set moisture AWP_MOISTURE
      ifelse (random-float 1 < (PERCENT_AWPS_FENCED / 100))
        [ set class FENCED_AWP
          set pcolor black
          set plabel "♦ " ]
        [ set class AWP
          set pcolor black ]
      ask neighbors [
        set moisture AWP_R1
        set class AWP_ADJACENT
        set plabel-color grey
        set pcolor black
        set plabel "#"
      ]
      ask neighbors [
        ask neighbors with [class = DESERT] [
          set moisture AWP_R2
          set class AWP_OVER2
          set pcolor white
          set plabel-color black
          set plabel "//"
        ]
      ]
    ]
  ]
end
```

5. a. *energy* = 0.91, *water* = 0.806, and *food* = 0.02 because *amtEat* = 0.01, so *energy* = 0.9 + 0.01, *water* = 0.8 + 0.6*0.01, and *food* = 0.03 - 0.01

b. *energy* = 0.905, *water* = 0.803, and *food* = 0.0 because *amtEat* = *availableFood* = 0.005, so *energy* = 0.9 + 0.005, *water* = 0.8 + 0.6*0.005, and *food*= 0.005 - 0.005

c. *energy* = 1.0, *water* = 0.8006, and *food* = 0.029 because *amtEat* = 1 - *energy* = 0.001, so *energy* = 0.9 + 0.001, *water* = 0.8 + 0.6*0.001, and *food* = 0.03 - 0.001

d. *energy* = 0.91, *water* = 1.0, and *food* = 0.02 because *amtEat* = 0.01, so *energy* = 0.9 + 0.01, *water* = the minimum of 0.999 + 0.6*0.01 = 1.005 and 1.0, and *food* = 0.03 - 0.01

6. a.

```
to toadMayEat
  ask toads with [state = ALIVE] [
    if (energy < WOULD_LIKE_EAT)
      [ eat ]
  ]
end
```

6. **b.** Note that *availableFood* is just *food*.

```
to eat
  let amtEat min (list AMT_EAT food (1 - energy))
  set energy energy + amtEat
  set water min list (water + FRACTION_WATER * amtEat) 1.0
  set food food - amtEat
end
```

6. **c.**

```
to toadMayDrink
  ask toads with [state = ALIVE] [
    if (moisture >= AWP_MOISTURE and water < WOULD_LIKE_DRINK)
      [ drink ]
  ]
end
```

6. **d.**

```
to drink
  set water min list (AMT_DRINK + water) 1
end
```

7.

```
to toadMove
  ask toads with [state = ALIVE] [
    ifelse (water < WOULD_LIKE_DRINK) [
        thirsty
      ][
        ifelse (energy < WOULD_LIKE_EAT) [
          lookForFood
        ] [
          ifelse (random-float 1 < MAY_HOP) [
            hopForFun
          ] [
            stayHere
          ]
        ]
      ]
    ]
end
```

8. **a.** Note that *stayHere* is implemented with *useWaterEnergySitting*.

```
to thirsty
  ifelse (moisture >= AMT_AWP) [
    useWaterEnergySitting
  ][
    ifelse (moisture >= 0) [
      lookForMoisture
    ][
      ifelse (plabel = "|") and ((count turtles-at -1 0) = 0) [
        moveW
      ][
        useWaterEnergySitting
       ]
     ]
   ]
```

```
end
```

8. b.

```
to lookForMoisture
  let matLst getNbrsLst
  let next max-one-of matLst [moisture]
  face next
  move-to next
  useWaterEnergyHopping
end

; Function to return a list of neighbors not including fenced AWPs
; and borders
to-report getNbrsLst
  let matLst neighbors4 with [class != FENCED_AWP and class != BORDER]
  report matLst
end
```

8. c.

```
to moveW
  set pxcor (pxcor - 1)
  useWaterEnergyHopping
end
```

8. d.

```
to useWaterEnergyHopping
  if moisture < AMT_AWP [ set water water - WATER_HOPPING ]
  set energy energy - ENERGY_HOPPING
end
```

9.

```
to lookForFood
  set lastx pxcor
  set lasty pycor
  let matLst (patch-set getNbrsLst patch-here)
  let next max-one-of matLst [food]
  face next
  move-to next
  ifelse [pxcor] of next = lastx and [pycor] of next = lasty [
    useWaterEnergySitting
  ][
    useWaterEnergyHopping
  ]
end
```

10.

```
to hopForFun
  let loc one-of getNbrsLst
  face loc
  move-to loc
  useWaterEnergyHopping
end
```

11. a.

```
to changeCounts
  ask toads with [state = ALIVE] [
```

```
        ifelse water < DESICCATE or energy < STARVE [
        set state DEAD
        move-to patch (SIDE + 1) (SIDE + 1)
        ] [
        if xcor = 1 - SIDE [
          set state MIGRATED
        ]
      ]
    ]
end
```

11. b.

```
to-report terminate?
  report (count toads with [ state = ALIVE ] < 1)
end
```

11. c.

```
if terminate? [ stop ]
```