A Simple Wolf-Sheep-Grass Simulation

Ruben R. Puente, Ph.D.

The Model

• Wolves eat sheep, reproduce, and (if they don’t eat enough) die.

• Sheep eat grass, reproduce, and (if they don’t eat enough, or are eaten by a wolf) die.

• Grass is eaten by sheep, but regrows after a fixed amount of time.

The Program Outline

- Define the entities involved and their attributes
- Set things up:
  - Define the starting conditions
  - Put in the grass, the sheep, and the wolves
- Run the simulation:
  - The sheep:
    - Move (costs them some energy)
    - Eat grass (gains them some energy)
    - Reproduce (makes a new sheep, and splits the energy with them)
    - Die, if they run out of energy
  - The wolves:
    - Move (costs them some energy)
    - Eat sheep (gains them some energy, and kills the sheep)
    - Reproduce (makes a new wolf, and splits the energy with them)
    - Die, if they run out of energy
  - The grass:
    - Can be eaten, but regrows after a certain amount of time

Running the Simulation

```lisp
(to go
  ask sheep [
    move
    set energy energy - 1 ;; sheep lose energy as they move
    eat-grass
    reproduce-sheep
    death
  ]
  ask wolves [
    move
    set energy energy - 1 ;; wolves lose energy as they move
    catch-sheep
    reproduce-wolves
    death
  ]
  ask patches [ grow-grass ]
  set ticks ticks + 1
end)
```
Defining Entities and Attributes

globals [ ticks ; How much time has elapsed? initial-number-sheep ; How many animals do we start with? initial-number-wolves sheep-gain-from-food ; How much energy do they gain from a meal? wolf-gain-from-food sheep-reproduce ; How likely are they to reproduce? wolf-reproduce ]
breed [ sheep a-sheep ] ; The sheep breed [ wolves wolf ] ; The wolves turtles-own [ energy ] ; Both wolves and sheep have energy sheep-own [ grabbed? ] ; used to prevent two wolves from eating the same sheep patches-own [ countdown ] ; Patches represent the grass

Setting Things Up - The Starting Conditions

to setup
  ca
  set ticks 0
  set initial-number-sheep 100
  set initial-number-wolves 50
  set sheep-gain-from-food 4.0
  set wolf-gain-from-food 20.0
  set sheep-reproduce 4.0
  set wolf-reproduce 5.0
Setting Things Up - Putting in the Grass

ask patches [ set pcolor green ]
ask patches [
  set countdown random grass-regrowth-time ;; initialize grass grow clocks randomly
  if (random 2) = 0 ;; half the patches start out with grass
    [ set pcolor brown ]
]

Setting Things Up - Adding the Sheep

set-default-shape sheep "sheep"
create-custom-sheep initial-number-sheep ;; create the sheep
[
  set color white
  set size 1.5 ;; easier to see
  set energy random (2 * sheep-gain-from-food)
  setxy random-xcor random-ycor
  set grabbed? false
]
Setting Things Up - Adding the Wolves

```
set-default-shape wolves "wolf"
cREATE-CUSTOM-WOLVES initial-number-wolves ;; create the wolves
[
  set color black
  set size 1.5 ;; easier to see
  set energy random (2 * wolf-gain-from-food)
  setxy random-xcor random-ycor
]
end
```

Moving (Sheep and Wolves)

```
to move ;; turtle procedure
  rt random-float 50 - random-float 50
  fd 1
end
```
Eating and Reproducing As a Sheep

```plaintext
to eat-grass  ;; sheep procedure
    ;; sheep eat grass, turn the patch brown
    if pcolor = green [ 
        set pcolor brown
        set energy energy + sheep-gain-from-food  ;; sheep gain energy by eating
    ]
end

to reproduce-sheep  ;; sheep procedure
    if random-float 100 < sheep-reproduce [  ;; throw "dice" to see if you will reproduce
        set energy (energy / 2)  ;; divide energy between parent and offspring
        hatch 1 [ rt random-float 360 fd 1 ]  ;; hatch an offspring, move it forward 1 step
    ]
end
```

Eating and Reproducing As a Wolf

```plaintext
to catch-sheep  ;; wolf procedure
    let prey one-of (sheep-here  ;; grab a random sheep
        with [not grabbed?])  ;; that no one else is grabbing
    if prey != nobody [ 
        set grabbed? of prey true
        ask prey [ die ] 
        set energy energy + wolf-gain-from-food ]  ;; get energy from eating
end

to reproduce-wolves  ;; wolf procedure
    if random-float 100 < wolf-reproduce [  ;; throw "dice" to see if you will reproduce
        set energy (energy / 2)  ;; divide energy between parent and offspring
        hatch 1 [ rt random-float 360 fd 1 ]  ;; hatch an offspring, move it forward 1 step
    ]
end
```
Dying (Sheep and Wolves)

to death ;; turtle procedure
  ;; when energy dips below zero, die
  if energy < 0 [ die ]
end

Growing Grass

to grow-grass ;; patch procedure
  ;; countdown on brown patches, if reach 0, grow some grass
  if pcolor = brown [
    elseif countdown <= 0
      [ set pcolor green
        set countdown grass-regrowth-time ]
    [ set countdown (countdown - 1) ]
  ]
end
The Simulation

Hippasus

http://hippasus.com/rrpweblog/
rubenrp@hippasus.com

This work is licensed under a Creative Commons Attribution-Noncommercial-Share Alike 3.0 License.