

Scheduling Model Behavior

EES 4760/5760

Agent-Based and Individual-Based Computational Modeling

Jonathan Gilligan

Class #17: Monday, October 21 2024

Mousetrap model

On “downloads” page

https://ees4760.jgilligan.org/models/class_17/Mousetrap_Ch14.nlogo

https://ees4760.jgilligan.org/models/class_17/Mousetrap_Ch14_v2.nlogo

Scheduling Actions:

Scheduling Actions:

- Representing time:
 - Discrete (`tick`)
 - Continuous (`tick-advance x`)
- Execution order
 - Synchronous
 - Asynchronous
 - Random order
 - Determined order

Repeating actions

- **repeat** repeats a certain number of times

```
repeat 5 [ wander ]
```

or

```
repeat (random 10) [ wander ]
```

- **while** repeats as long as a condition is true

```
while not any? other turtles-here [  
  wander ]
```

- **loop** repeats forever (until **stop** or **report**)

```
loop [  
  wander  
  if any? other turtles-here [ stop ]  
]
```

Discrete vs. Continuous Time

Discrete vs. continuous time

- Almost all models use discrete time:
 - `tick` advances tick counter by 1.
 - `ticks` is always an integer.
- Continuous time
 - `tick-advance 2.3`
 - `ticks` can have fractional values
- Things to think about:
 - When to tick?

```
to go
  ask patches [ do-patch-stuff ]
  ask turtles [ do-turtle-stuff ]

  tick
  if ticks > run-duration [stop]
end
```

```
to go
  tick
  if ticks > run-duration [stop]

  ask patches [ do-patch-stuff ]
  ask turtles [ do-turtle-stuff ]
end
```

- BehaviorSpace only writes to the file when it gets to the end of `go`,
 - so when you use `stop`, that iteration of `go` will not write the values from that step to the file.

Order of Execution

Order of execution

- `ask`: Asks turtles in a random order.

```
ask turtles [do-sales]
```

- Suppose we wanted bigger turtles to act before the smaller ones?

```
foreach (sort-on [size] turtles) [ x -> ask x [do-sales] ]
```

- `x ->` means that:
 - NetLogo creates a local variable `x`
 - For each turtle in the list or agent-set, it sets `x` to that turtle, and executes whatever's to the right of `->`

Order of execution

```
ask patches [ set patch-value 0 ]
ask turtles [turtle-action]

to turtle-action
  ask one-of patches with [pcolor =
blue]
  [
    set patch-value patch-value + 1
    set pcolor red
  ]
end
```

- Each turtle finishes everything in brackets before the next turtle starts
 1. turtle 7 checks [pcolor] of patch 20 20: it's blue
 2. turtle 7 increments patch-value
 3. turtle 7 sets pcolor to red
 4. turtle 3 checks [pcolor] of patch 20 20: it's red
 5. turtle 3 checks another patch
 6. ...
- [patch-value] of patch 20 20 is 1
 - [pcolor] of patch 20 20 is red

Order of execution

```
ask patches [ set patch-value 0 ]
ask turtles [ turtle-action-1 ]
ask turtles [ turtle-action-2 ]

to turtle-action-1
  ask one-of patches with [pcolor =
blue]
  [
    set patch-value patch-value + 1
  ]
end

to turtle-action-2
  ask one-of patches with [pcolor =
blue]
  [
    set pcolor red
  ]
end
```

- Different order of execution
 1. turtle 4 checks [pcolor] of patch 20 20: it's blue
 2. turtle 4 increments patch-value of patch 20 20
 3. turtle 13 checks [pcolor] of patch 20 20: it's blue
 4. turtle 13 increments patch-value of patch 20 20
 5. turtle 6 checks [pcolor] of patch 8 32: it's blue
 6. turtle 6 sets pcolor to red
 7. turtle 9 checks [pcolor] of patch 17 3: it's blue
 8. turtle 9 sets pcolor to red
- [patch-value] of patch 20 20 is 2
 - [pcolor] of patch 20 20 is blue

Synchronous vs. asynchronous updating

- What is the difference?
- When would you want to use one or the other?
 - Business investor model?
 - Telemarketer model?
- How would you do *asynchronous* updating?
- How would you do *synchronous* updating?
 - Hidden state-variables (variables you choose not to let other turtles see)
 - Two ways:
 1. Break submodel into two parts:
 1. Turtles sense environment, update hidden variables that others can't sense
 2. Update environment (including state-variables that others can sense)
 2. Make *shadow copy* of all state variables:

```
turtles-own [ wealth new-wealth ]
```

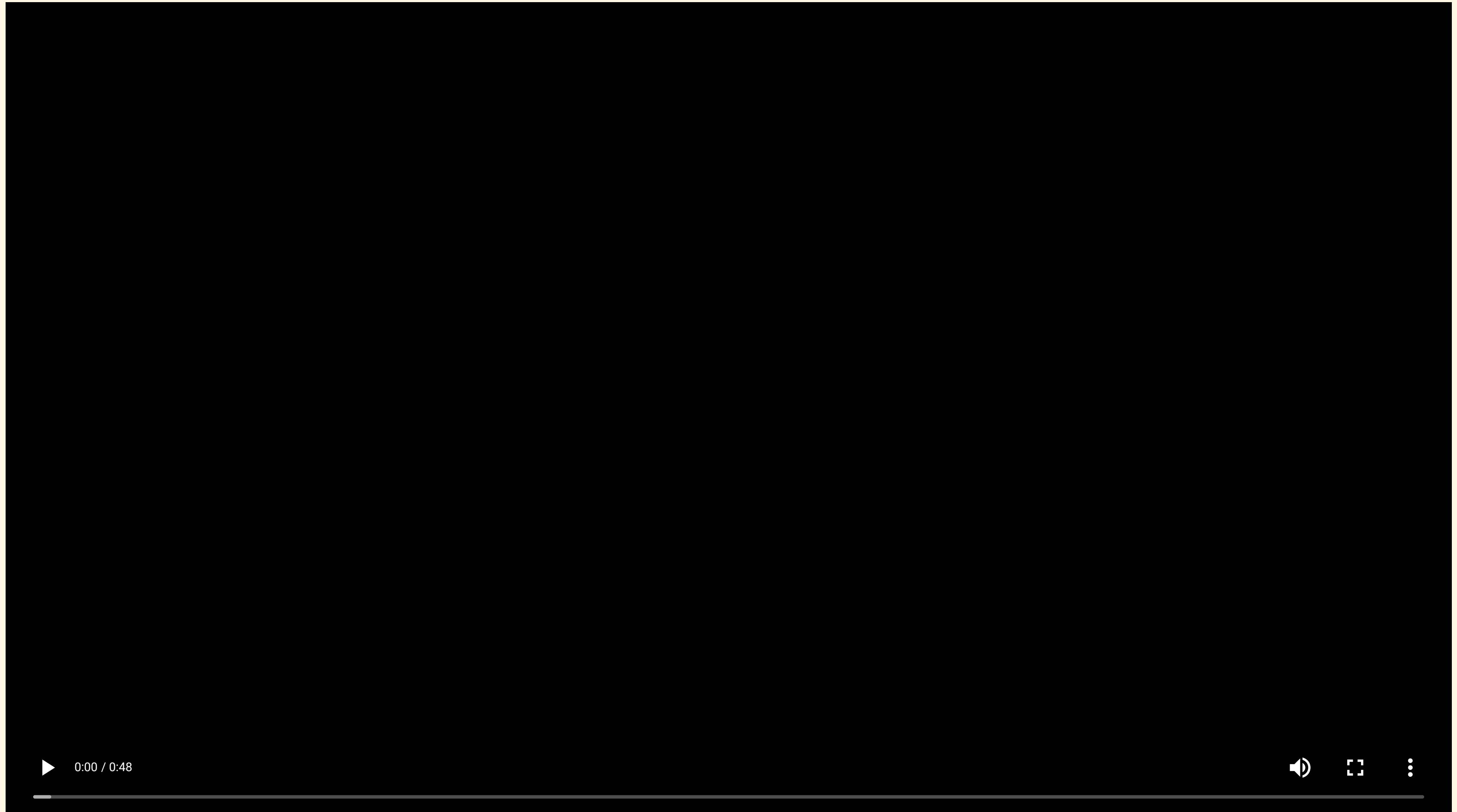
 1. Sensing sees originals, updates change shadow-copies (*new-wealth*)
 2. Update the original (*set wealth new-wealth*)

One procedure or two?

- What is the difference?
- When would you want to use one or the other?
 - Business investor model?
 - Telemarketer model?
- The book mentions the `ask-concurrent` primitive.
 - **Don't use it!**
 - It is very unpredictable and makes it hard to understand your model.

Mousetrap Model

Mousetrap model



Mousetrap model

https://ees4760.jgilligan.org/models/class_17/Mousetrap_Ch14.nlogo

https://ees4760.jgilligan.org/models/class_17/Mousetrap_Ch14_v2.nlogo

- Play with models
- Compare continuous updating with updating on ticks

