From Animations To Science

EES 4760/5760

Agent-Based and Individual-Based Computational Modeling

Jonathan Gilligan

Class #6: Tuesday, Sept. 10 2019

Download files

Download files for Butterfly model

Download the following files from the Download page on the class web site:

- A single ZIP file with all the files in it:
  https://ees4760.jgilligan.org/models/class_06/class_06.zip
  - Unzip after downloading.
- Or individual files listed on Download page
  https://ees4760.jgilligan.org/downloads/butterfly_science/
- Start NetLogo and load `butterfly_class_06a.nlogo`

Projects

Planning

- Semester Project:
  - Pick a model from an open-source repository (OpenABM.org), or NetLogo “model library” that you want to work with.
  - Fri. Sept. 27:
    - One-page description of model and thoughts for extending it
  - Tue. Oct. 15: Examine ODD and code.
    - Short write-up of how model works and output from running it
  - Wed. Oct. 23: ODD for extending model
  - Tue–Thu. Dec. 3–5: Presentations on experiments with extended models
  - Fri. Dec. 6: Write-up of research project (around 10 pages)
- Team Project:
  - Each team (2–3 students) will code a model from an ODD in the textbook (Ch. 10 or Ch. 13)
  - Use model to do exercises from book
  - Make presentation about what you learned (Thu. Oct. 10)
- Detailed Assignments on Brightspace and course web site.

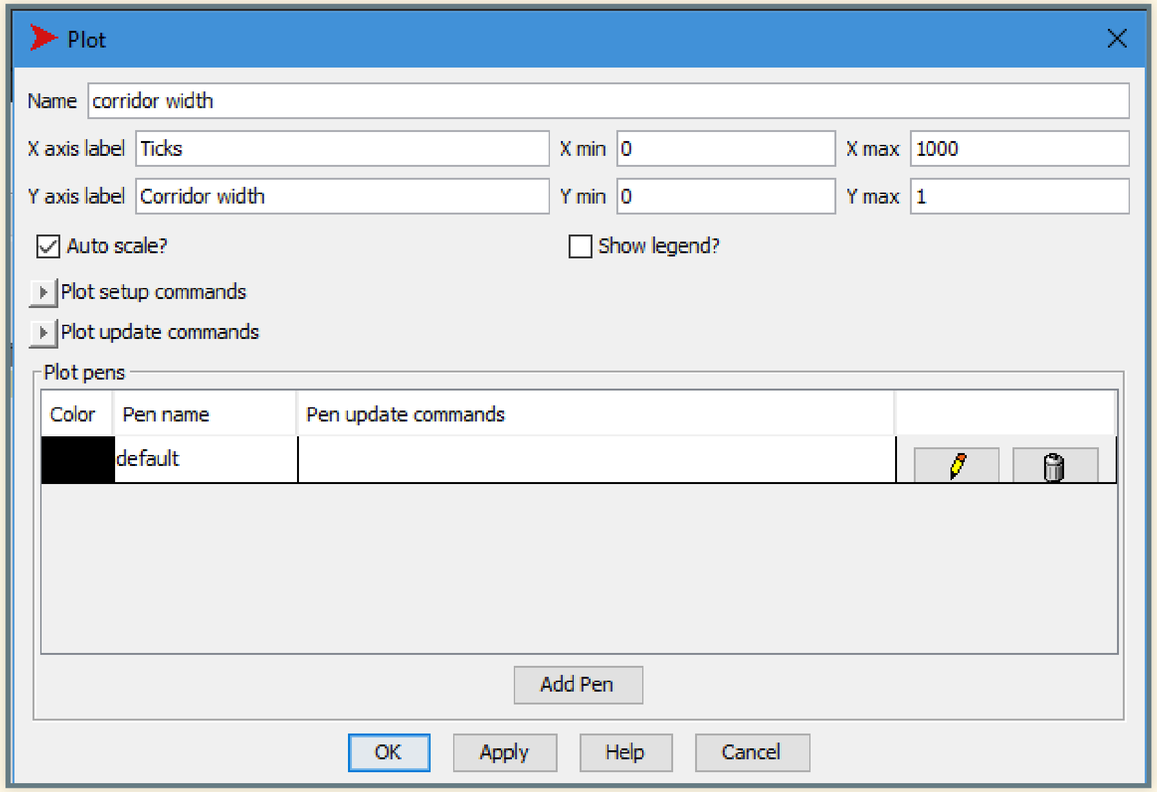
Experiments with the Butterfly Model

Experiments with the Butterfly Mode

- You should have downloaded:
  - Various versions of NetLogo butterfly models.
  - The NetLogo “Testing Is Fun” library https://ees4760.jgilligan.org/models/class_06/jg-tif.nls
  - A digital elevation map of real hills https://ees4760.jgilligan.org/models/class_06/ElevationData.txt
- Start NetLogo and load `butterfly_class_06a.nlogo`

Plot Corridor Width

- On the interface tab, add a plot



- On the code tab, add a line to go to plot the corridor width

```
plot corridor-width
```

Enhance Interface

- Add a button to export the plot to a file:

```
export-plot "Corridor-width" (word "corridor-output-for-q-" (precision q
2) ".csv")
```

  - `precision q 2` rounds `q` off to two decimal places.
- Add a button to increment *q* by 0.1

BehaviorSpace

- If your model is having problems, compare it to `butterfly_class_06b.nlogo`
- Open BehaviorSpace and create an experiment
  - Call it `experiment`
  - Vary `real-terrain` between `false` and `true`
  - Vary `q` from 0 to 1 in steps of 0.2
  - Run 20 repetitions for each value of `q`.
  - Measure `corridor-width` at the last tick only
  - Set time limit to 0 to let model run until it stops
- Run BehaviorSpace experiment
  - Save “table” output
  - Speed things up by unchecking “Update view” and “Update plots and monitors”
- Open the analyzeBehaviorspace app at https://ees4760.jgilligan.org/analyze_behaviorspace and use it to compare the relationship between corridor width and `q` for each terrain

Practice

Practice

- Work together with a partner
- Add a button to erase the tracks of the turtles (Exercise 5.2)
- Using the realistic terrain, play with *q* and see what values do best at helping butterflies find mates near hilltops.

Testing

Leaving trails

Turtles leave trails with the pen (because you told them `pen-down` when you created them in `to-setup`). The pens are the same color as the turtles, so it's hard to tell the difference between turtles and trails.

It would be nice to color patches they visited yellow so you could erase the pen trails and see the red turtles contrasting with the yellow trails of visited patches.

- Open the original "`butterfly_class_06a_testing.nlogo`"
- Add code to color patches yellow when a turtle visits them:
  - At the end of `to move`, add

```
set pcolor yellow
```

- Add a button to the interface to erase the pen trails:
  - Give the button the command `clear-drawing` and the display name "erase trails"

Testing Models

- Using monitors
- Unit testing resource “Testing Is Fun”
  - Open the original “`butterfly_class_06a_testing.nlogo`”
  - At beginning of code:

```
__includes ["jg-tif.nls"]
```

  - In `to_setup` add:

```
initialize-tests
```

  - In `to go`, after the turtles move add:

```
set-context "Testing consistency"
test-that "# visited patches should equal # yellow patches"
expect-that (count patches with [visited?]) equals (count patches with [pcolor =
yellow])
...
if ticks >= 1000 or all? turtles [finished?]
[
  resume-all-tests
  stop
]
```

More testing

- At the beginning `to go`, before the turtles move, add:

```
set-context "Moving turtles"

ask turtles with [not finished?] [move]
```

- In `to move` replace the `ifelse` block with this:

```
ifelse random-float 1 < q
[
  let current-elevation elevation
  move-to max-one-of neighbors [elevation]
  test-that (word "Turtle " self " should not move downhill")
  expect-that elevation is-greater-or-equal current-elevation
] ; move uphill
[ move-to one-of neighbors ] ; move randomly
```

- Why does the turtle sometimes move downhill when it should be moving uphill?