Designing and Documenting Models: The ODD Protocol

EES 4760/5760

Agent-Based and Individual-Based Computational Modeling

Jonathan Gilligan

Class #4: Monday, September 1 2025

Getting Started:

- Purpose of Today’s Class:
  - Learn a structure for designing and documenting a model
- Getting started for today:
  - Download and save the Butterfly model from https://ees4760.jgilligan.org/models/class_04/butterfly_odd.nlogo
    - Or go to the “Downloads” page at https://ees4760.jgilligan.org and click on item 4: “Butterfly Model ODD”
  - Open NetLogo and load “butterfly_odd.nlogo”

Designing and Documenting Models

Designing and Documenting Models

ECOLOGICAL MODELLING 198 (2006) 115–126

available at www.sciencedirect.com

ScienceDirect

journal homepage: www.elsevier.com/locate/ecolmodel

A standard protocol for describing individual-based and agent-based models

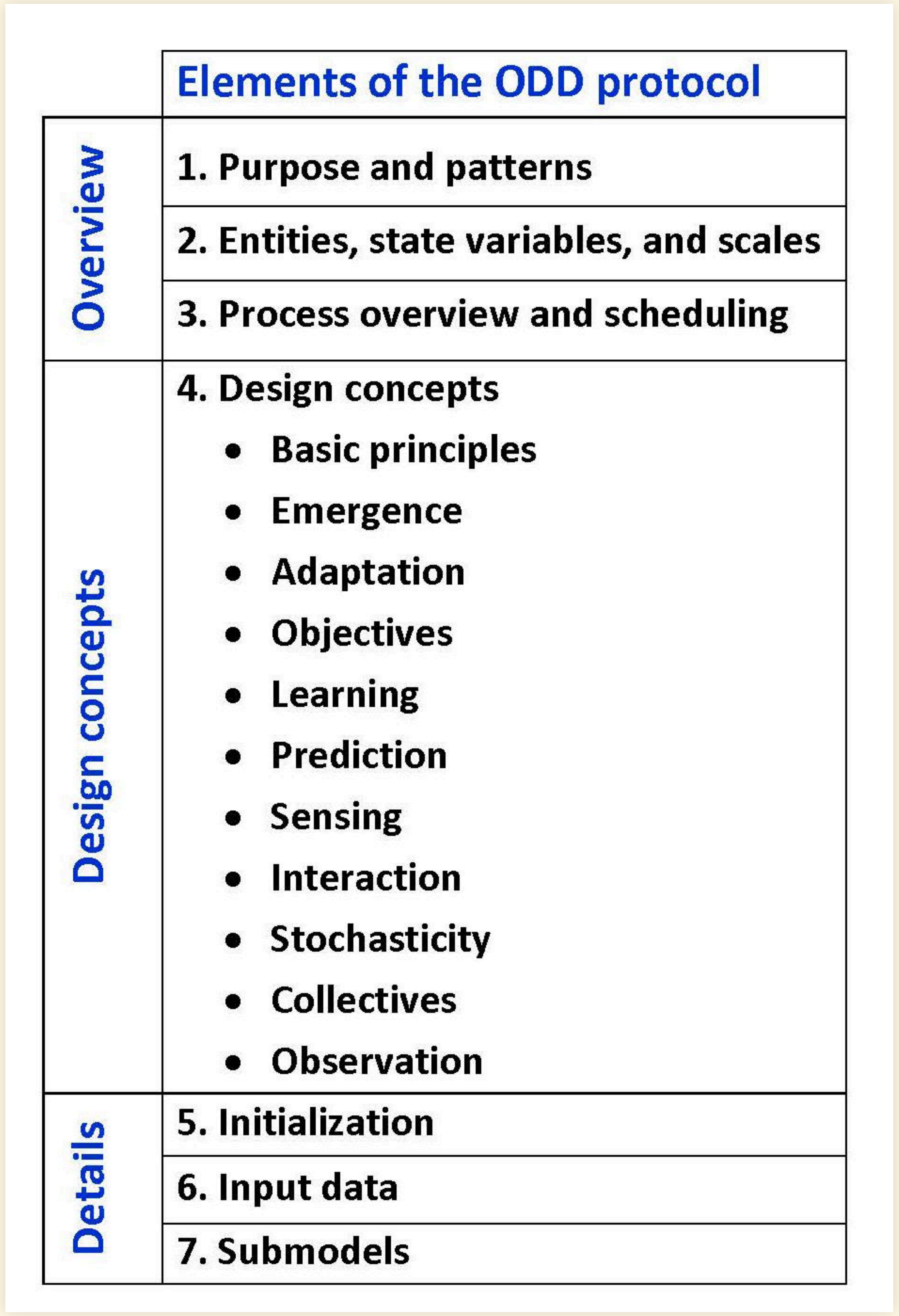
Volker Grimm[a,*], Uta Berger[b], Finn Bastiansen[a], Sigrunn Eliassen[c], Vincent Ginot[d], Jarl Giske[c], John Goss-Custard[e], Tamara Grand[f], Simone K. Heinz[c], Geir Huse[g], Andreas Huth[a], Jane U. Jepsen[a], Christian Jørgensen[c], Wolf M. Mooij[h], Birgit Müller[a], Guy Pe'er[i], Cyril Piou[b], Steven F. Railsback[j], Andrew M. Robbins[k], Martha M. Robbins[k], Eva Rossmanith[l], Nadja Rüger[a], Espen Strand[c], Sami Souissi[m], Richard A. Stillman[e], Rune Vabø[g], Ute Visser[a], Donald L. DeAngelis[n]

[a] UFZ Umweltforschungszentrum Leipzig-Halle GmbH, Department Ökologische Systemanalyse, Permoserstr. 15, 04318 Leipzig, Germany
[b] Zentrum für Marine Tropenökologie, Fahrenheitstr. 6, 28359 Bremen, Germany
[c] University of Bergen, Department of Biology, P.O. Box 7800, N-5020 Bergen, Norway
[d] INRA, Unité de Biométrie, Domaine St.-Paul, 84 814 Avignon Cedex 9, France

Design

- Don’t start writing code until you know what you’re trying to do.
- Big picture
  - What is the **purpose** of your model?
  - What **things** does your model use?
  - How do those things **behave**?
- Design concepts
  - How do you represent the **things** in your model?
  - How do you implement their **behavior**?
  - How will your **things** and **behaviors** realize your **purpose**?
  - What **data** will you collect from your model?
  - How will you **use that data** to achieve your **purpose**?

Overview, Design Concepts, and Details



	Elements of the ODD protocol
Overview	1. Purpose and patterns
	2. Entities, state variables, and scales
	3. Process overview and scheduling
Design concepts	4. Design concepts • Basic principles • Emergence • Adaptation • Objectives • Learning • Prediction • Sensing • Interaction • Stochasticity • Collectives • Observation
Details	5. Initialization
	6. Input data
	7. Submodels

General

Detailed

ODD in perspective:

- Write overview and major parts of design concepts *first*
- As you write the model code, revisit and revise design concepts.
- Much of the details will emerge in the course of programming.
- When you are finished, write a complete ODD. This will be the major documentation for your model.

ODD Outline

1. Purpose & Patterns

Questions:

- What is the **purpose** of the model?
  - What are you trying to do?
  - ***Suggestion***: What will the *key figure* showing your results look like?
- **Patterns:** What aspects of the world are we modeling?
  - How will you judge whether the model is realistic enough for the purpose?
  - What can you neglect?

2. Entities, State Variables, Scale

- What kinds of **entities** are in the model?

  Agents, collectives, spatial units, global environment, …
  - Turtles, breeds, patches, links, etc.
- What attributes (state-variables) characterize the entities?

  Age, sex, wealth, mood, opinion, soil type, land costs, rainfall, market price, …
  - `turtles-own`, `patches-own`, etc.
  - You only need state-variables if they vary
    - values are different for different entities, change over time, etc.
  - Use the fewest attributes you need
- What are the temporal and spatial resolutions and extents of the model?
  - **Resolution:** how detailed?
  - **Extents:** how big?
  - “Model settings” button in NetLogo

3. Process Overview and Scheduling

- How do states change?
- What entities do what, and in what order?
  - Schedule:
    1. Which entities take actions?
    2. What actions do they take?
       - Details of actions often go in **submodels**
         - e.g., `to-go` used submodels `to-eat` & `to-move` in our sugarscape model
    3. In what order do they take them?
    4. Design Concepts

4. Design Concepts

There are **11 design concepts** (see Table 3.1).

Textbook has one chapter for each.

Outline of Design Concepts

1. **Basic Principles:** Basis of model in general concepts and theories
2. **Emergence:** What emerges as the model runs?
   (phenomena not imposed or directly programmed)
3. **Adaptation** How do agents respond to changes in their environment?
   What decisions do they make, and how do they decide?
   Do they seek objectives directly (*deliberately*) or indirectly (*mimic natural behavior*)?

   Note: *Adaptation* means responding to current conditions. It's different from *persistent change* (the way the word is used in *evolution*)

   Seek shelter when it starts to rain.
4. **Objectives (Fitness):** Goals of agents? What determines survival?
   Do objectives change as agent changes?
5. **Learning:** How do individuals change behavior as they gain experience?

   Note: This is where *persistent change* happens. It's often confused with *adaptation*

   After getting rained on several times, start carrying an umbrella when it's cloudy.

Outline of Design Concepts (cont.)

6. **Prediction:** How do agents predict consequences of their decisions? (learning, memory, environmental cues, programmed assumptions)
7. **Sensing:** What do agents know or perceive when making decisions? (Is sensing process itself explicitly modeled, or do they *just know*?)
8. **Interaction:** What forms of interaction among agents are there?
9. **Stochasticity:** Is there randomness in model? *Randomness must be justified!*
10. **Collectives:** Grouping of individuals (herds, families, teams, social networks, …)
11. **Observation:** How do scientists collect data from the model for analysis?

Details

5. Initialization

- What is the initial state of the model world?
  - Time $t = 0$ of a simulation run
- In detail:
  - How many entities, of what type, are there initially?
  - What are the exact values of their state variables?
    (Or how were they set at random?)
  - Is initialization always the same,
    or does it vary from one simulation run to the next?
  - Are initial values chosen arbitrarily, or based on data?
  - References to those data should be provided.

6. Input data

- Does the model use input from external sources to represent processes that change over time?
  - data files, other models, human interaction
- If so, what data?
  - Where did they come from?
  - Provide references, citations.

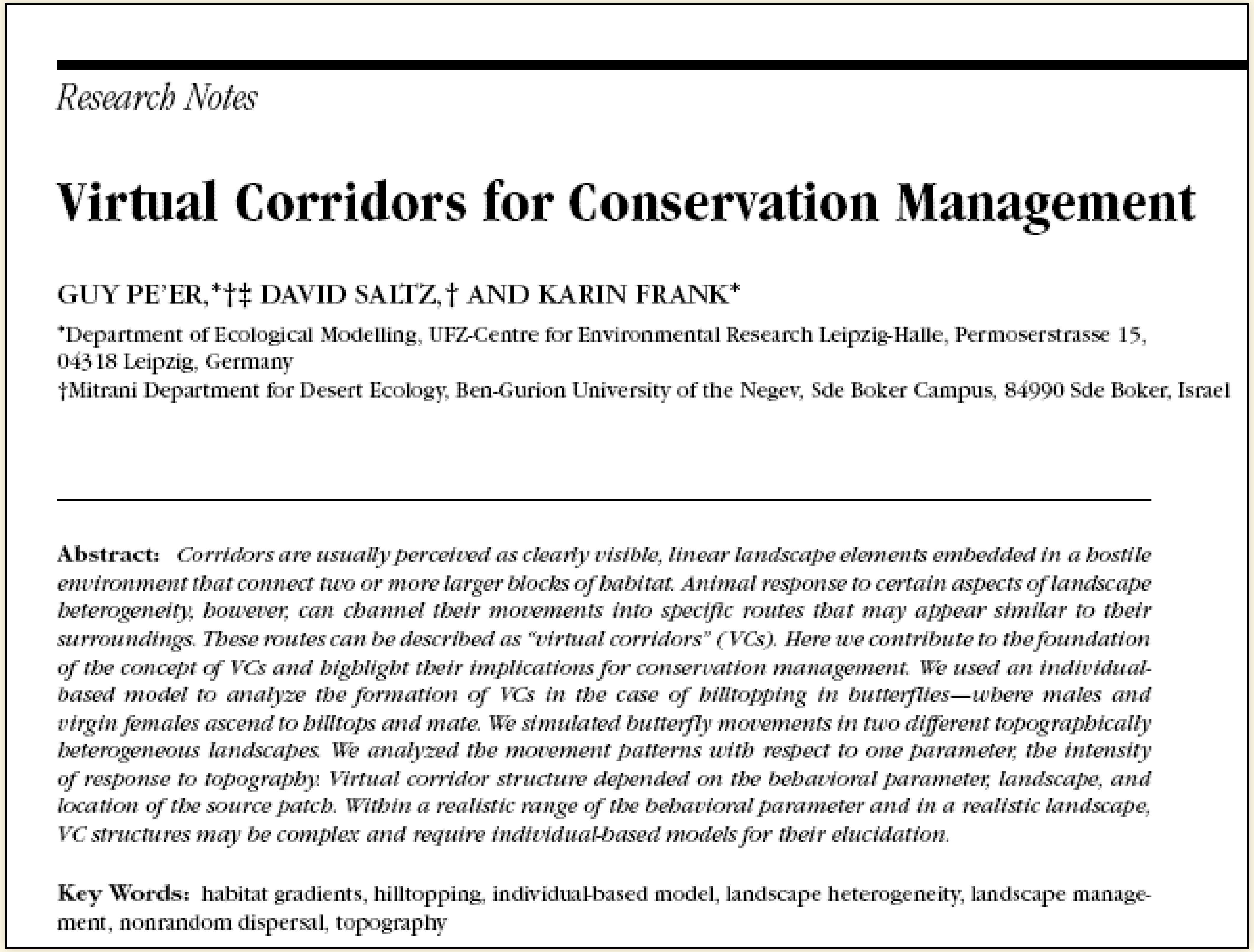
7. Submodels

If the **process scheduling** step contains a list of processes or actions, explain, in detail what **submodels** represent those processes or actions.

- What are the model parameters?
- How were the submodels designed or chosen?
- How were they tested?

Example:
Virtual Corridors for Conservation Management

Example: Virtual Corridors for Conservation Management



Research Notes

Virtual Corridors for Conservation Management

GUY PE'ER,*†‡ DAVID SALTZ,† AND KARIN FRANK*

*Department of Ecological Modelling, UFZ-Centre for Environmental Research Leipzig-Halle, Permoserstrasse 15, 04318 Leipzig, Germany
†Mitrani Department for Desert Ecology, Ben-Gurion University of the Negev, Sde Boker Campus, 84990 Sde Boker, Israel

Abstract: *Corridors are usually perceived as clearly visible, linear landscape elements embedded in a hostile environment that connect two or more larger blocks of habitat. Animal response to certain aspects of landscape heterogeneity, however, can channel their movements into specific routes that may appear similar to their surroundings. These routes can be described as "virtual corridors" (VCs). Here we contribute to the foundation of the concept of VCs and highlight their implications for conservation management. We used an individual-based model to analyze the formation of VCs in the case of hilltopping in butterflies—where males and virgin females ascend to hilltops and mate. We simulated butterfly movements in two different topographically heterogeneous landscapes. We analyzed the movement patterns with respect to one parameter, the intensity of response to topography. Virtual corridor structure depended on the behavioral parameter, landscape, and location of the source patch. Within a realistic range of the behavioral parameter and in a realistic landscape, VC structures may be complex and require individual-based models for their elucidation.*

Key Words: habitat gradients, hilltopping, individual-based model, landscape heterogeneity, landscape management, nonrandom dispersal, topography

Pe'er, G., D. Saltz, & K. Frank, "Virtual corridors for conservation management," *Conservation Biology* **19**, 1997 (2005).

Butterfly Model in NetLogo

Open NetLogo and load “butterfly_odd.nlogo”

- Code section is blank, but ODD is filled in on “Info” tab.
  - As you read Chapter 4, for Wednesday, the book will guide you to fill in the code based on ODD, as you read.
    - You don’t need to turn this in, but you will use it as the basis for homework due Sept. 10.
- Click on “Edit” (pencil icon) to see what Info tab looks like when you edit it.
  - For details on editing “Info” tab, open NetLogo User Guide from the NetLogo Help menu and go to “Info Tab Guide” in the “Reference Section”

Purpose

- Ecologists observe that as butterflies move uphill, they concentrate into narrow and well-definied *virtual corridors* rather than following any old path to the top of the hill.
- Explore the concept of *virtual corridors:*

  Can concentrations of migrating animals emerge spontaneously from movement behavior and topography, instead of being a special habitat?
- Specifically, How does the concentration of hill-topping butterflies emerge from:
  - The way butterflies move uphill
  - The topography of the landscape

Entities, State Variables, and Scales

- **Landscape:**,
  - Square grid cells (patches) with one *state variable*: **elevation**.
- **Butterflies:**
  - Have one *state variable*: **location**
    (discrete: which patch they’re in)

Entities, State Variables, and Scales

- **Spatial Scale:**
  - 150 × 150 cells (patches)
  - Two modes:
    - Generic mode: No specific size for a patch
    - Mapping mode: Each patch corresponds to 25 × 25 meters in the real landscape
- **Time Scale:**
  - Simulations last 1000 ticks
  - Tick length is unspecified (time for a butterfly to move one patch).

Process Overview and Scheduling

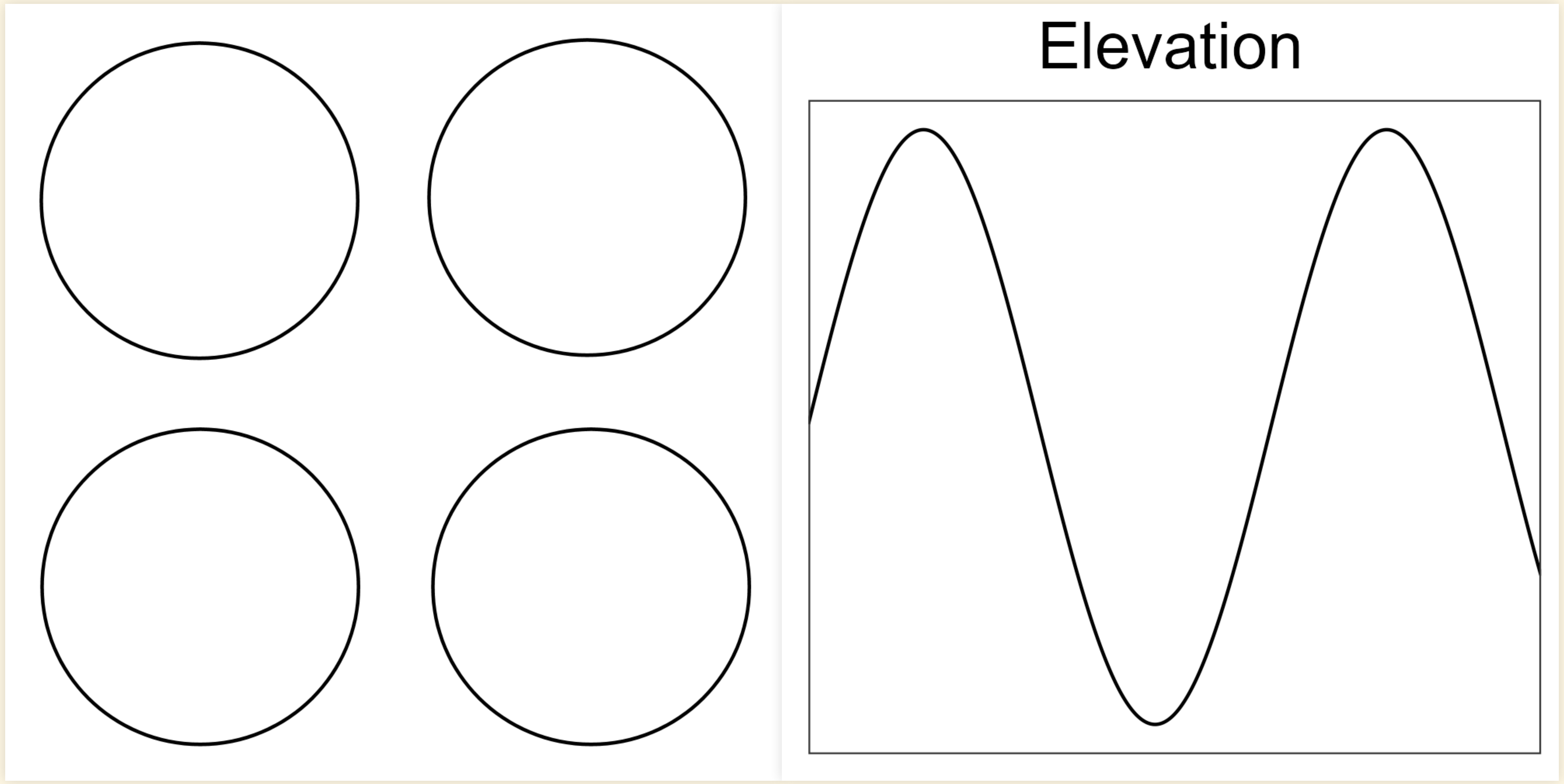
- **Only one process:** butterfly movement
  - On each tick, each butterfly moves once
  - The order in which butterflies move is not important because they don’t interact

Design concepts (important ones)

- **Basic Principles:** The concept of *virtual corridors*
- **Emergence:** results (concentration of butterflies in corridors) emerge from movement rule and topography
- **Sensing:** Butterflies can sense elevation in current and 8 surrounding patches
- **Interaction:** None
- **Stochasticity:** Used to represent reasons why butterflies do not move straight uphill
- **Observation:** We need a way to measure of butterfly concentration

Initialization

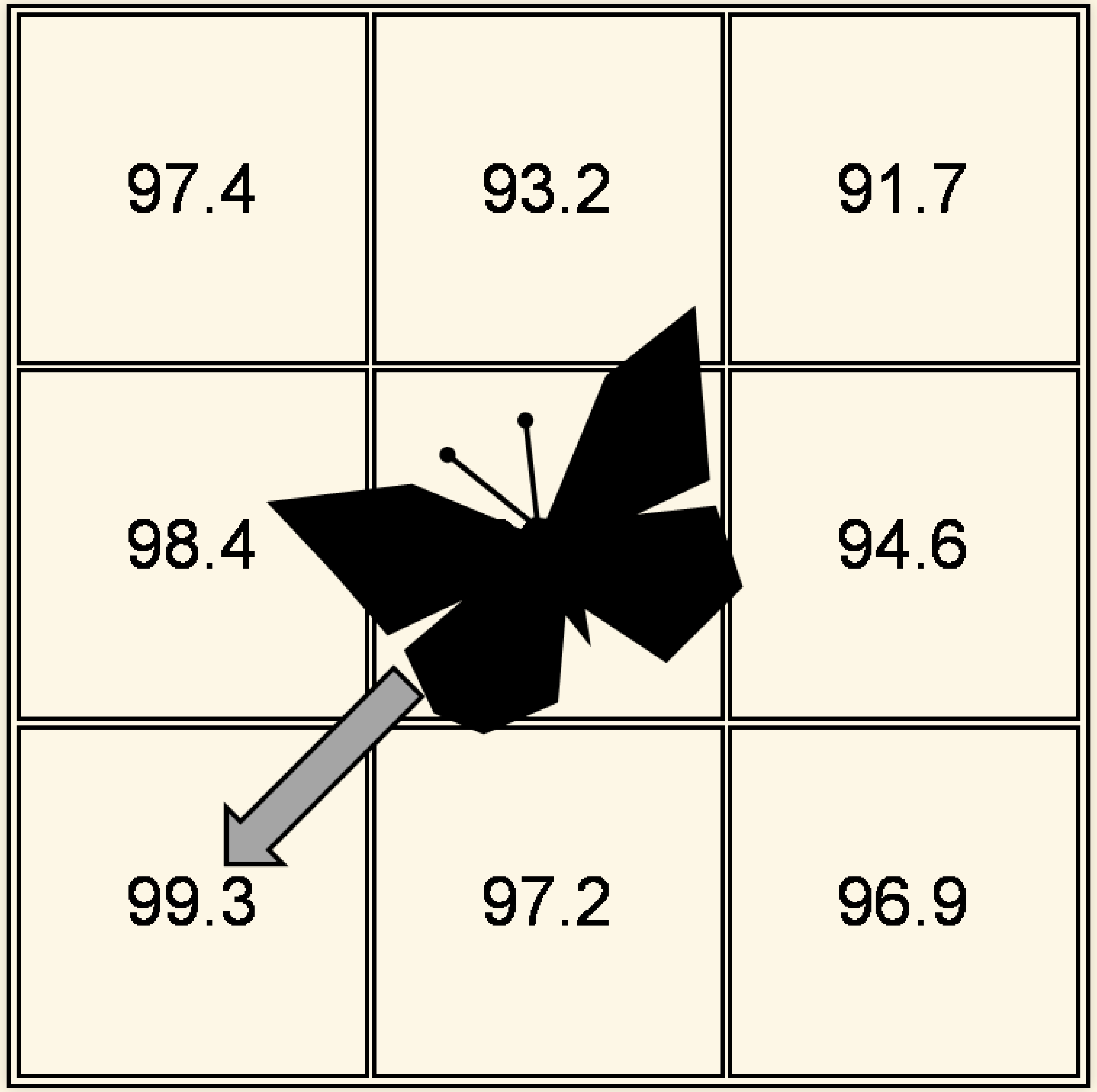
- **Landscape:** cell elevations set to a flat landscape with four hills



- **Butterflies:** 500 are created and placed in one patch

Submodel: Butterfly movement

- Global parameter *q* is probability that butterfly moves straight uphill, vs. moving to random neighbor cell.



Next Steps

Next Steps

For Wednesday, you will follow the book and write code to implement the butterfly model.

Catching up

Catching up from Wednesday

- Download the example model: Go to Downloads page, “Sugarscape model from class #3”, and
  - download “Netlogo model from class”,
  - or download directly from https://ees4760.jgilligan.org/models/class_03/class_03_example.nlogo

Working with agentsets (`with`)

- `turtles` is an agentset of all turtles.
- "`with`" is one of many primitives that create a subset of an agentset:

  `ask turtles with [color = blue] [move]`

- Similar keywords for subsetting:
  - `with-min`, `with-max` (one or more turtles)
  - `n-of`, `max-n-of`, `min-n-of` (exactly *n* turtles)
  - `one-of`, `max-one-of`, `min-one-of` (exactly one turtle)
- Use the dictionary to look up correct syntax.
- Click "setup" and then try this:

```
ask patches [ set pcolor black ]
ask patches with-max [sugar] [ set pcolor cyan ]
ask max-n-of 5 patches [sugar] [ set pcolor pink ]
```

Working with agentsets (`of`)

- "`of`" provides a *list* of the values of an `-own` variable

  ```
  set happiness min [happiness] of turtles-on neighbors
  ```

  - In a patch context, `neighbors` makes an agentset of the neighboring patches.
  - `turtles-on` makes an agentset of all the turtles on `neighbors`
  - `[ happiness ]` gets the `turtles-own` variable `happiness` in the context of each turtle in `turtles-on neighbors`
- More generally, "`of`" is a primitive for getting a value from *another* agent or agentset: if `friend` is another turtle,

  ```
  set happiness [happiness] of friend
  ```

- Use the dictionary to look up correct syntax.

Movement in our Model

```
to move
  if sugar <= 0 or any? other turtles-here
  [
    move-to max-one-of neighbors [ sugar ]
  ]
end
```

If there isn’t any sugar, or if there’s another turtle on the patch, it moves to the neighboring patch with the most sugar.

- We run `move` in the context of a turtle:
  - For each neighbor of the patch the turtle is on, look up `sugar` in the context of that neighbor
    - The brackets `[ … ]` signal the change of context.
  - Find the neighbors with the maximum `sugar`, and pick one of them at random
  - In the context of the original turtle, move to that neighboring patch

Working with variables (`=`, `set`)

- Two fundamental kinds of operations:
  - Changing the value of a variable:
    - Assignment operations (`set`)
  - Checking to see whether a value satisfies some condition:
    - Conditional operations (`=`, also `>`, `<`, `>=`, `<=`, `!=`)

Equals or no equals?

Assignment statements

- Wrong:

```
happiness = ([happiness] of a-neighbor-turtle)
```

- Right:

```
set happiness ([happiness] of a-neighbor-turtle)
```

Conditional statements (Logical: yes or no)

```
if happiness = 3
  [stop]

if happiness <= 3
  [stop]

if happiness != 5 or ticks > 17
  [stop]
```

Working with variables: `set` vs. `let`

- Global variables (known to all procedures)
- Local variables (known only to one procedure)
- Use `let` to *create* and *set the value* of a new local variable:

```
let mean-neighbor-size mean [size] of turtles-on neighbors
```

- Use `set` to change the value of an existing variable (global, local, patches-own, turtles-own, etc.)

```
set wealth wealth * 1.1

set hypotenuse sqrt(a ^ 2 + b ^ 2)
```

Variables in Our Model

Examine the `to eat` and `to move` procedures on the code tab, and edit `to-move`

```
to eat
  ifelse hunger > sugar
  [
    ; Use set to change an existing variable "hunger"
    set hunger hunger - sugar
    set sugar 0
  ]
  [
    set sugar sugar - hunger
    set hunger 0
  ]
end

to move
  if sugar <= 0 or any? other turtles-here
  [
    ; Use let to create a new variable "dest"
    let dest max-one-of neighbors [ sugar ]
    move-to dest
  ]
end
```

Working with variables: Giving a value to another agent

- How does one patch (or turtle) give the value of one of its variables to other patches?
  - There are two ways to do this.

```
let my-color pcolor
ask neighbors [set pcolor my-color]
```

  or

```
ask neighbors [set pcolor [pcolor] of myself]
```

- Turtles implicitly access `patches-own` variables (e.g., `pcolor`, `sugar`) of the patch they're on as though they were `turtles-own`
- Converse is not true: Patches don't automatically see `turtles-own`
- Why?
  - A turtle can only be on one patch at a time,
  - But a patch may have multiple turtles.

Code branching (conditional statements)

```
if (logical condition)
  [
    ; Do this if condition is true
  ]
```

or

```
ifelse (logical condition)
  [
    ; Do this if condition is true ...
  ]
  [  ;else
    ; Do this if condition is false
  ]
```

Working with stochasticity (randomness)

- Uniform distribution of random integers less than `x`

```
random x
```

  - `random 5` would return 0, 1, 2, 3, or 4

- Uniform distribution of floating-point numbers less than `x`

```
random-float x
```

  - `random-float 5` would return a number between 0 and 4.9999…. The number could be very close to 5.

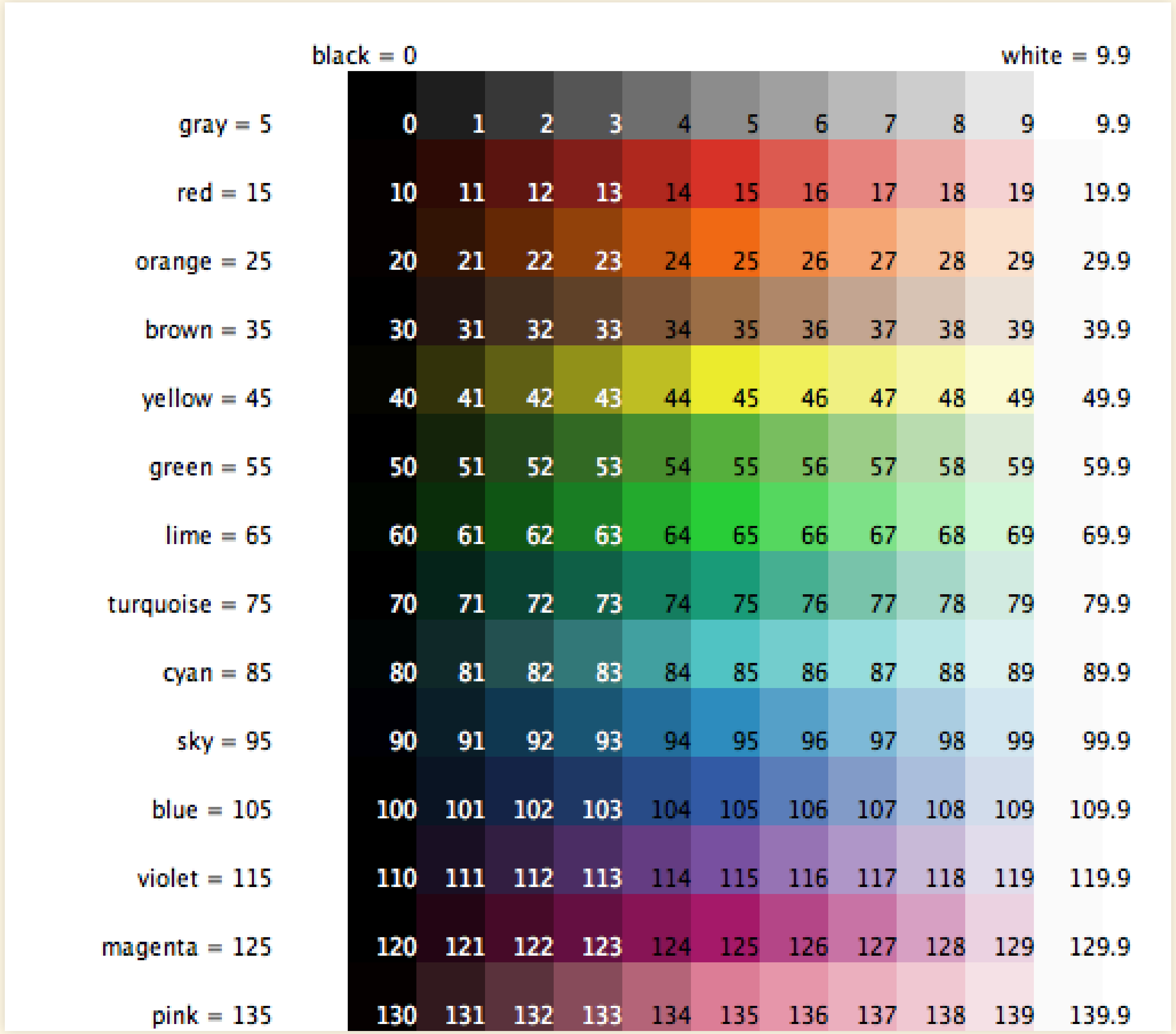
- Select one patch at random and turn it green

```
ask one-of patches [set pcolor green]
```

- Select one agent at random and turn it right 5 degrees:

```
ask one-of turtles [right 5]
```

Working with graphics



	black = 0										white = 9.9
gray = 5	0	1	2	3	4	5	6	7	8	9	9.9
red = 15	10	11	12	13	14	15	16	17	18	19	19.9
orange = 25	20	21	22	23	24	25	26	27	28	29	29.9
brown = 35	30	31	32	33	34	35	36	37	38	39	39.9
yellow = 45	40	41	42	43	44	45	46	47	48	49	49.9
green = 55	50	51	52	53	54	55	56	57	58	59	59.9
lime = 65	60	61	62	63	64	65	66	67	68	69	69.9
turquoise = 75	70	71	72	73	74	75	76	77	78	79	79.9
cyan = 85	80	81	82	83	84	85	86	87	88	89	89.9
sky = 95	90	91	92	93	94	95	96	97	98	99	99.9
blue = 105	100	101	102	103	104	105	106	107	108	109	109.9
violet = 115	110	111	112	113	114	115	116	117	118	119	119.9
magenta = 125	120	121	122	123	124	125	126	127	128	129	129.9
pink = 135	130	131	132	133	134	135	136	137	138	139	139.9

- 0, 10, 20, ..., 130 are all black
- 0.9, 9.9, 19.9, 29.9, ..., 139.9 are all white

Coloring Patches and Turtles

Examine the procedures `update-pcolor` and `update-color`

```
to update-pcolor
  ; black for 0, bright yellow for max-sugar
  set pcolor scale-color yellow sugar 0 (2 * max-sugar)
end

to update-color
  ifelse hunger > 5
  [
    ; white for 5 to bright red for 10
    set color scale-color red hunger 15 5
  ]
  [
    ; bright green for 0, white for 5
    set color scale-color green hunger -5 5
  ]
end
```

- `scale-color color value range1 range2` sets the lightness of the color.
  - Higher values = lighter, lower = darker.
- If `range1 > range2`, light and dark are reversed (`scale-color red hunger 15 5`).

Running Our Model

- Press “Check” and make sure there are no syntax errors
- Go to “Interface” tab
- Click on “setup”
- Click on “go”
- You can download a copy of the model from https://ees4760.jgilligan.org/models/class_03/class_03_example.nlogo

Interacting with a Model

- On Wednesday, we set a value for the global variable `sugar-growth`:

```
globals
[
  max-sugar
  sugar-growth
]
```

  and

```
to setup
  clear-all

  set sugar-growth 0.050
  set max-sugar 100
  ...
end
```

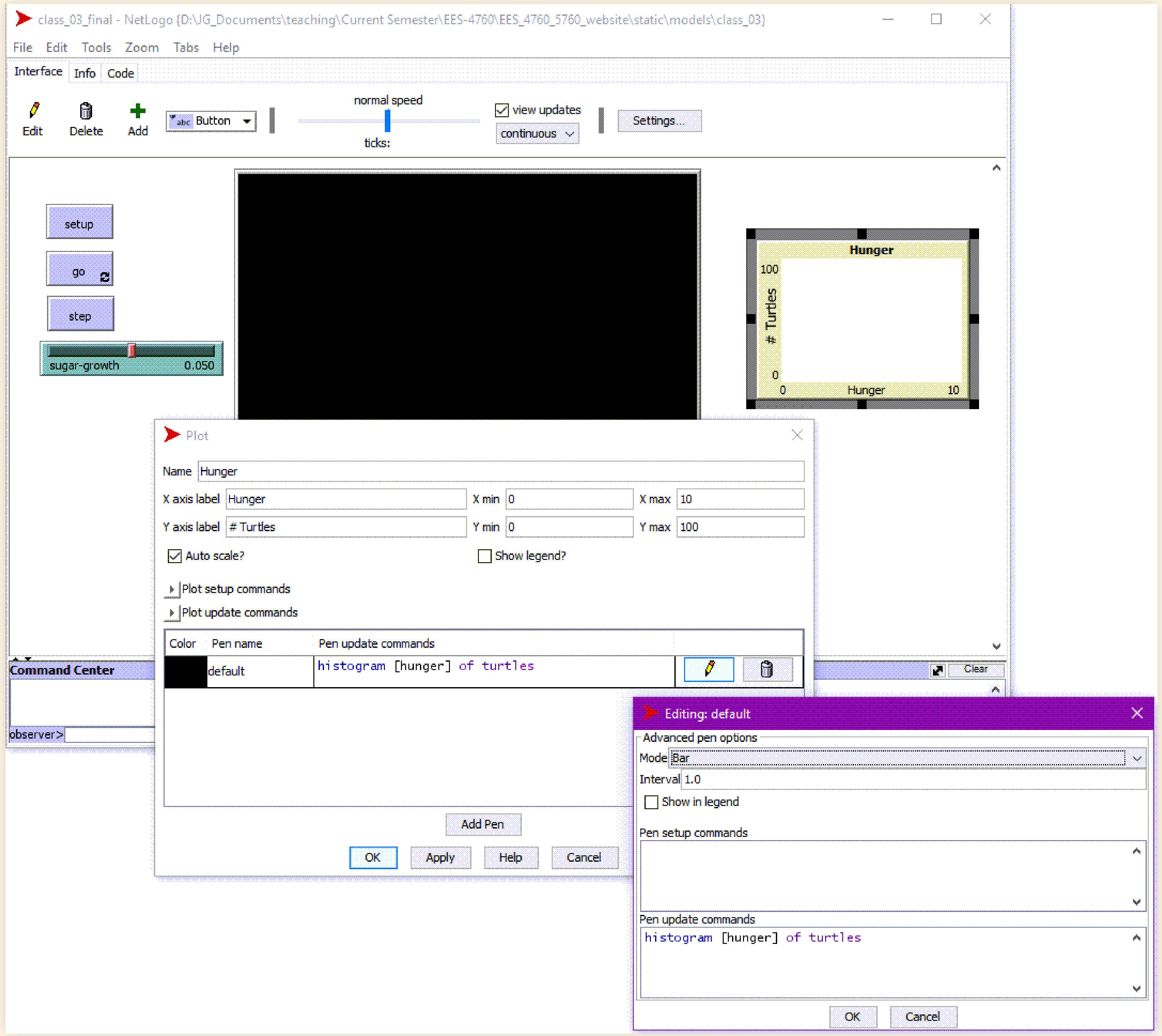
- In the model you downloaded, I replaced this with a slider on the interface:
  - Added a Slider
    - Type “sugar-growth” into “Global Variable”
    - Set minimum to 0, increment to 0.005, maximum to 0.1, and value to 0.050
  - In the code tab, commented out or deleted the definition and initialization of `sugar-growth`

```
globals
[
  max-sugar
  ; sugar-growth
]
...
to setup
  clear-all
  ; set sugar-growth 0.050
  ...
end
```

Monitoring a Model

On the “interface” tab:

- Added a Plot
  - Name the plot “Hunger”
  - Set X max to 10 and Y max to 100
  - Type “Hunger” for “X axis label” and “# Turtles” for “Y axis label”
  - Click on the pencil icon next to “default” pen
    - Choose “Bar” for “Mode”
    - In “Pen update commands” type `histogram [hunger] of turtles`
  - Press “OK”



Name: Hunger
X axis label: Hunger — X min: 0 — X max: 10
Y axis label: # Turtles — Y min: 0 — Y max: 100
☑ Auto scale? ☐ Show legend?

Color	Pen name	Pen update commands
■	default	`histogram [hunger] of turtles`

Mode: Bar
Interval: 1.0
☐ Show in legend
Pen update commands:

```
histogram [hunger] of turtles
```

Play with the model

- Do interesting things happen for different values of `sugar-growth`?
- It might be fun to comment out the line in `to go` that stops the model after 2000 ticks

```
  ; if ticks > 2000 [ stop ]
```

More sophisticated Sugarscape model

- Download from https://ees4760.jgilligan.org/models/class_03/fancy-sugarscape.nlogo
    - You will also need to download https://ees4760.jgilligan.org/models/class_03/sugar-map.txt
    - This is similar to the research model used by Axtell and Epstein to study theoretical economics.

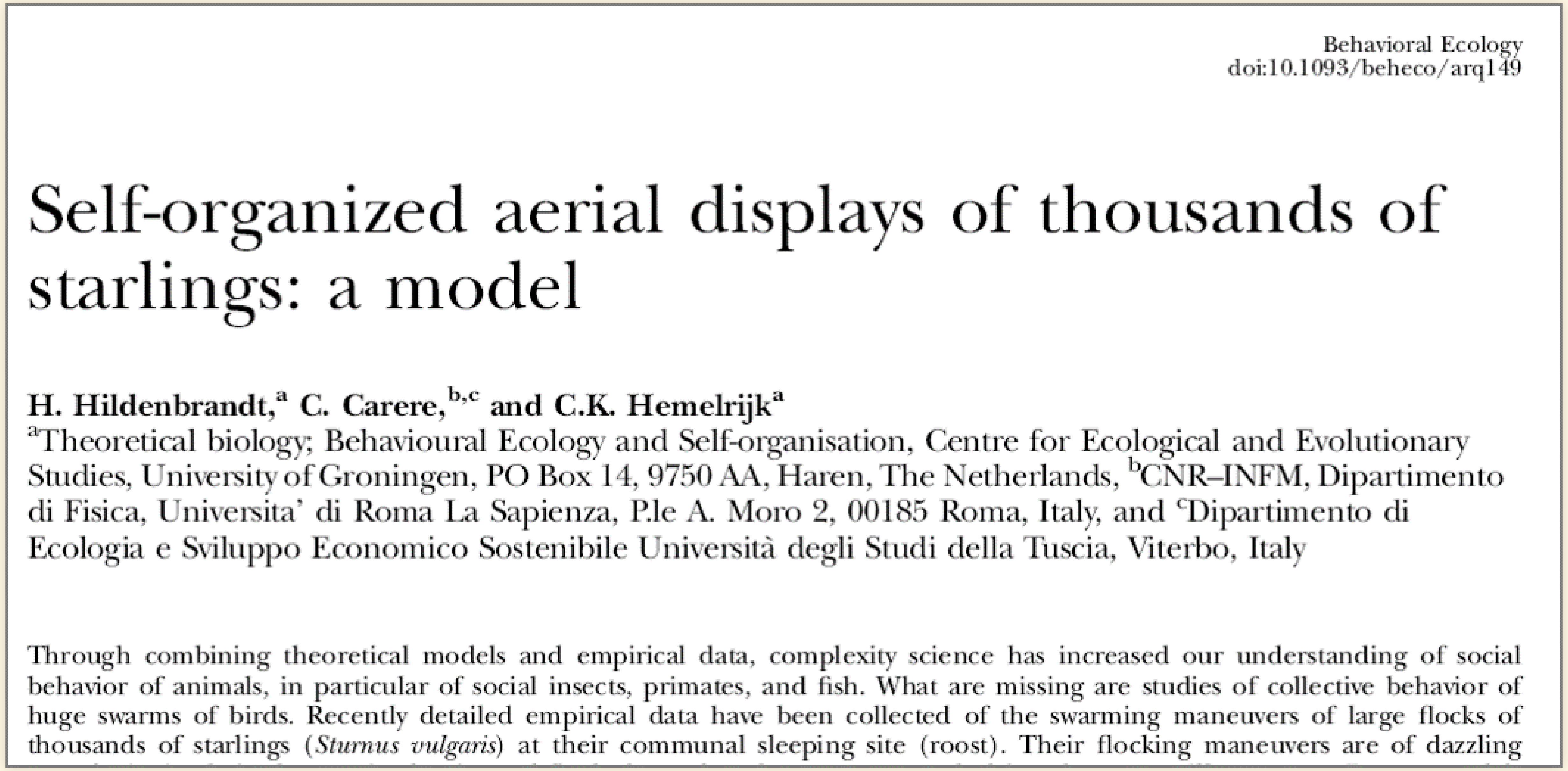
More about Agent-Based Models

Agent-based models

- Agents/Individuals are discrete, unique, and autonomous entities.
    - Discrete entities: Important at low densities
    - Unique: Individuals, even of same age and species, can be **different**
    - Individuals have a **life history**
- Interactions among individuals are usually **local**, not global
- Individuals make decisions, which can be **adaptive**
- Ecology or society **emerges** from individual behavior (bottom-up)

Example: flocks of starlings

- Thousands of individuals
  - unique and different
  - interact locally
  - show adaptive behavior



Behavioral Ecology
doi:10.1093/beheco/arq149

Self-organized aerial displays of thousands of starlings: a model

H. Hildenbrandt,[a] C. Carere,[b,c] and C.K. Hemelrijk[a]
[a]Theoretical biology; Behavioural Ecology and Self-organisation, Centre for Ecological and Evolutionary Studies, University of Groningen, PO Box 14, 9750 AA, Haren, The Netherlands, [b]CNR–INFM, Dipartimento di Fisica, Universita' di Roma La Sapienza, P.le A. Moro 2, 00185 Roma, Italy, and [c]Dipartimento di Ecologia e Sviluppo Economico Sostenibile Università degli Studi della Tuscia, Viterbo, Italy

Through combining theoretical models and empirical data, complexity science has increased our understanding of social behavior of animals, in particular of social insects, primates, and fish. What are missing are studies of collective behavior of huge swarms of birds. Recently detailed empirical data have been collected of the swarming maneuvers of large flocks of thousands of starlings (*Sturnus vulgaris*) at their communal sleeping site (roost). Their flocking maneuvers are of dazzling

Starling murmuration

By Liberty Smith & Sophie Windsor Clive, Islands and Rivers, https://vimeo.com/31158841

Flock of thousands of starlings

Simulated flock of thousands of starlings

Simulated flock of thousands of starlings