Agent-Based Modelling and Simulation with NetLogo

Day 1: Session 2

Diving into NetLogo

Session 2 Outline

- NetLogo **features** and the **interface**.
- Interact with an existing model.
- NetLogo components: **observer**, **turtles**, **patches** and **links**.
- NetLogo **programming** environment.
- **Documentation** and how to use it.

NetLogo: features and interface

- **Get and install NetLogo:**
  http://ccl.northwestern.edu/netlogo/

- Get the ***"walk 1.nlogo"*** model file.

- Open the file with NetLogo.

NetLogo Interface

switch between:
model interface
model description
model code

model interface
building tools

parameter slider

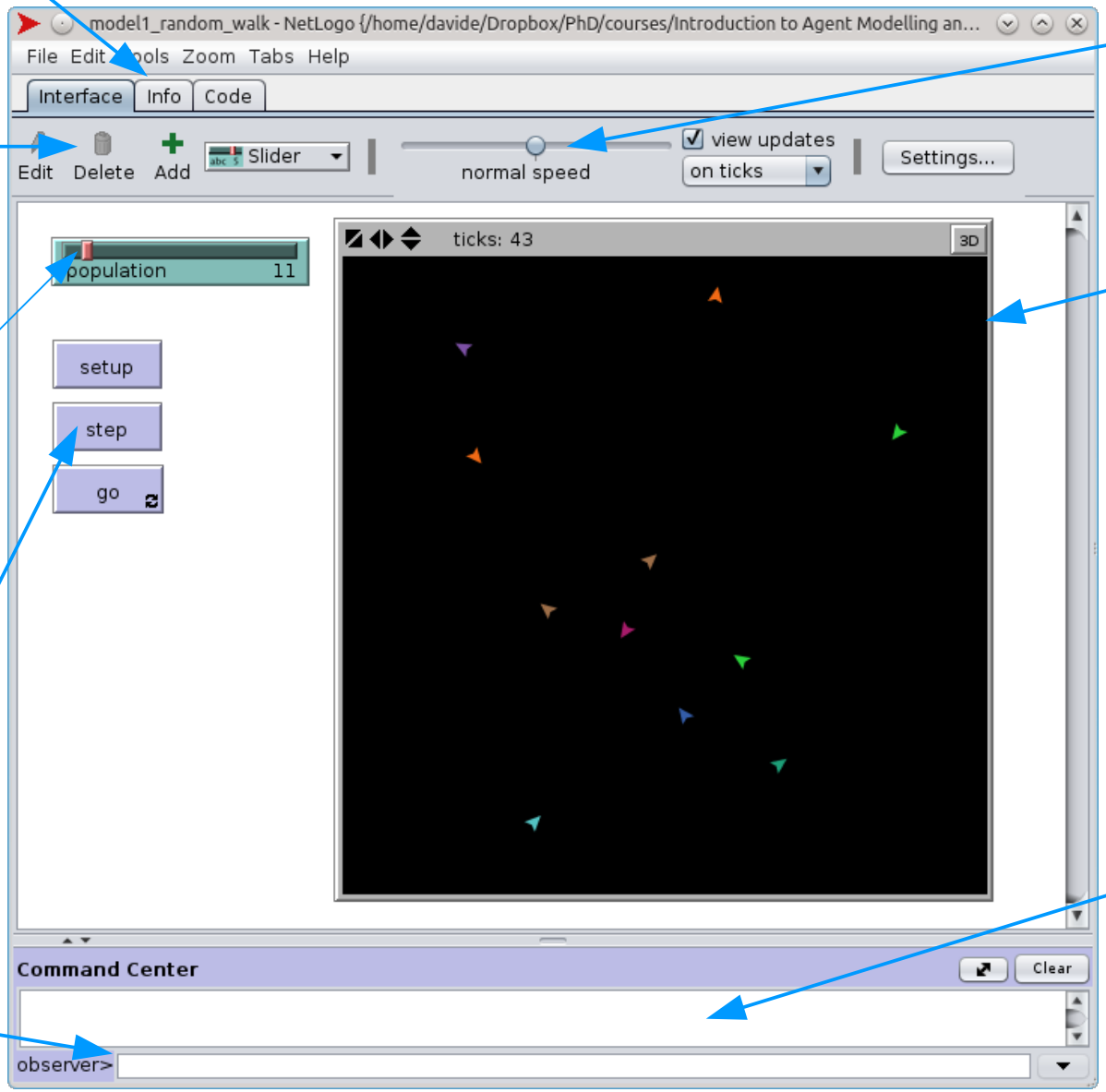
command buttons

command console

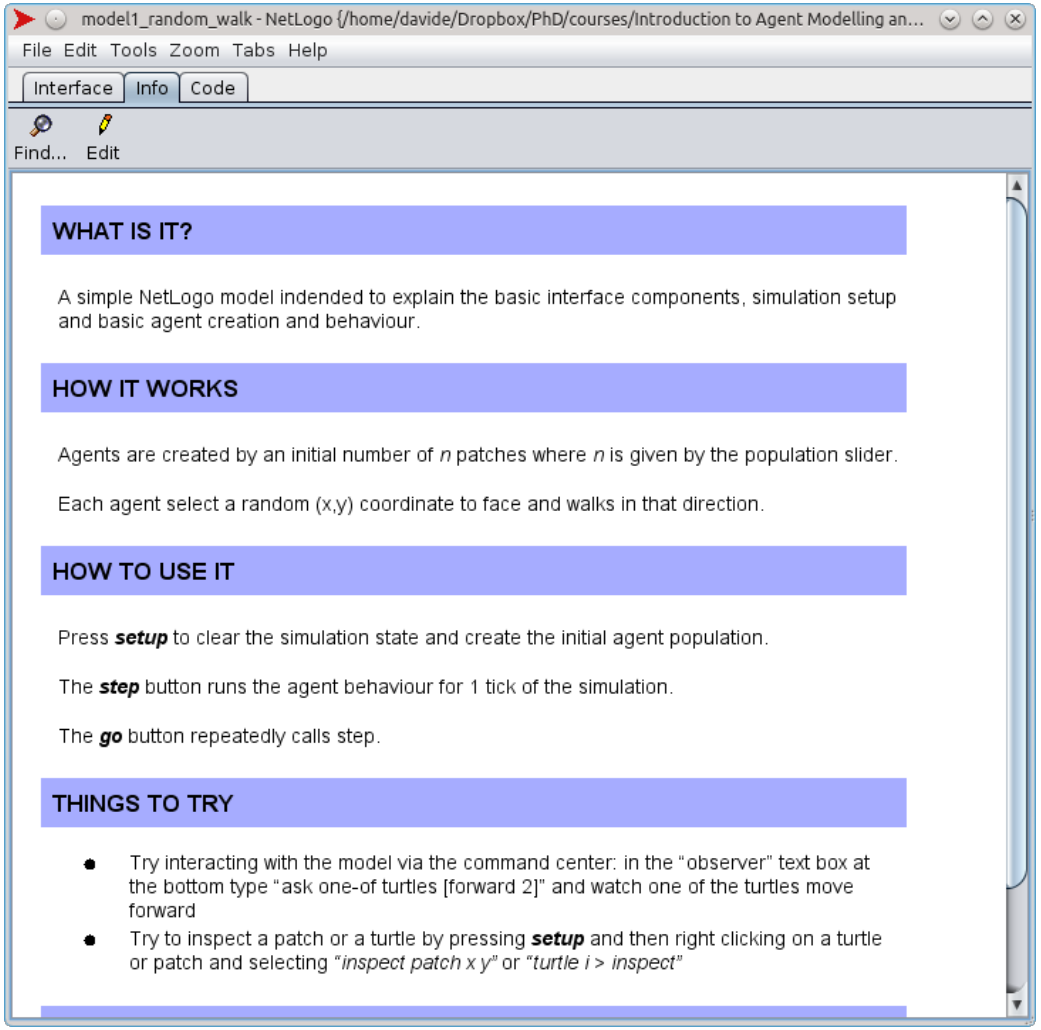
simulation speed

world 2D view

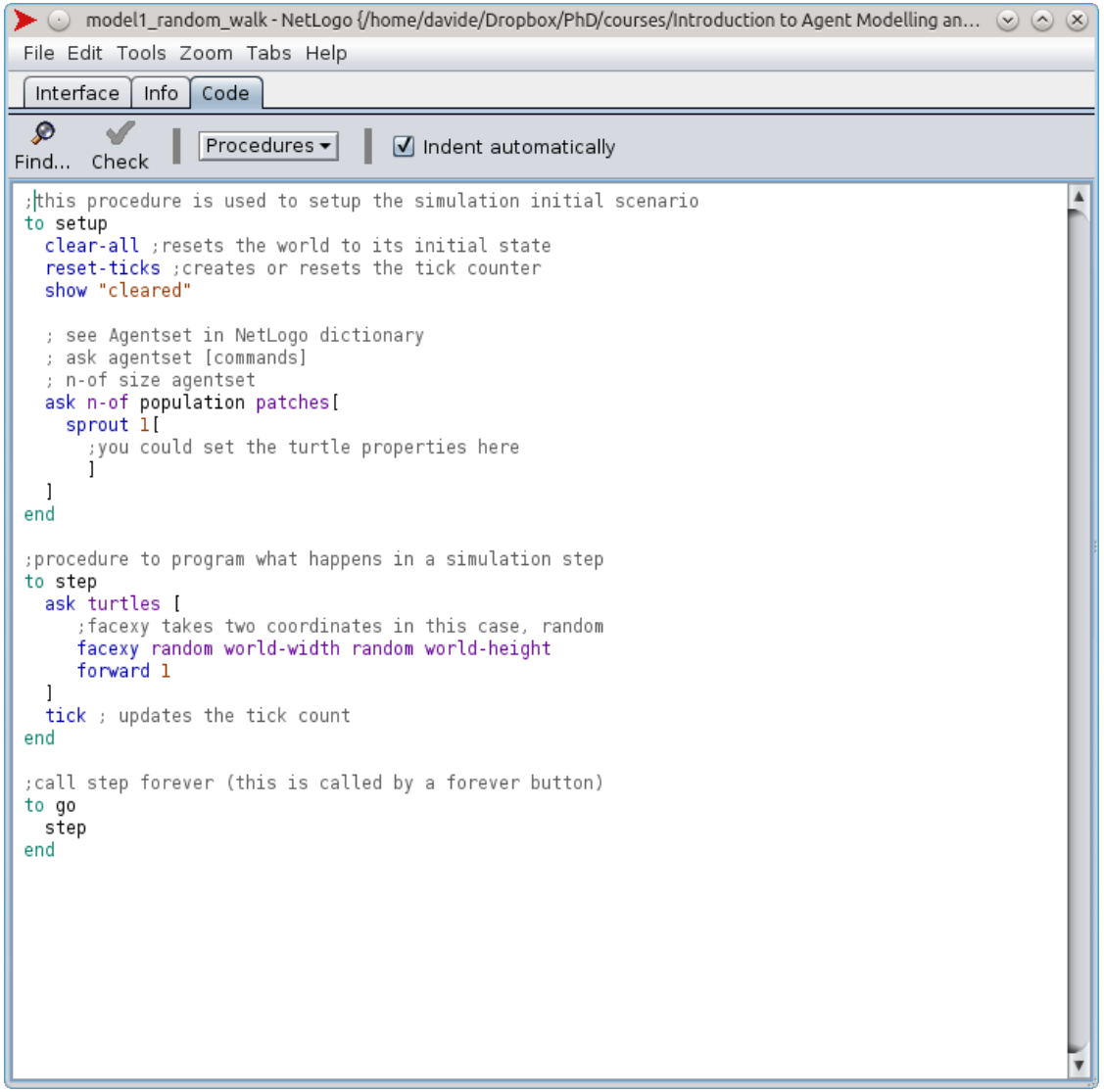
text output



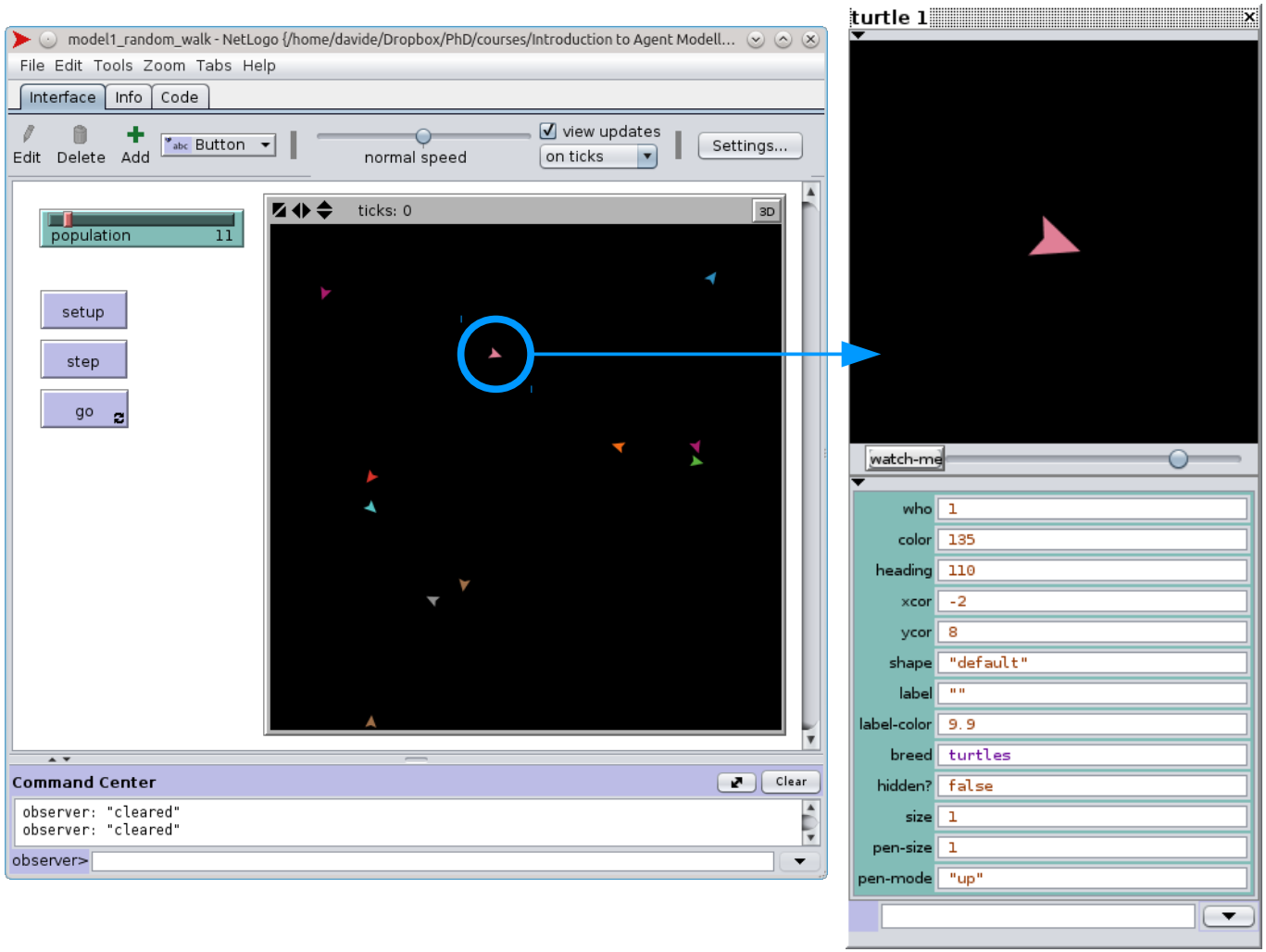
NetLogo Interface: model description



NetLogo Interface: model code



NetLogo Interface: inspect



NetLogo Components

- **Observer:** an agent that "observes" the simulation and is located outside the scope of the other elements.

- **Patches:** The NetLogo world is a two dimensional grid of "*patches*". Patches are the individual squares in the grid.

- **Turtles:** Mobile agents (*turtles*) move over a grid of stationary agents (*patches*).

- **Links:** *link* agents connect turtles to make networks and graphs.

NetLogo Components

- **Important concepts**:
  - All the components can be seen as agents.
  - They can have their own properties, can be given commands, can detect and interact with other agents in their environment.
  - The simulation model is controlled by the **observer**.
  - The definition of agent behaviours is defined by the observer by using the **ask** command.

NetLogo Model Skeleton: setup

- Tipicaly you start by definning a button for setting up the initial model state. (model1 ***setup*** button for instance)
- When creating a button you have to assign a command to it. (in this case ***setup***).
- The command associated with the button has to exist in the ***code pannel*** and will be executed when the button is pressed.
- Each command is defined by a structure:

```
to command-name
    set of instructions
end
```

NetLogo Model Skeleton: setup

```
;this procedure is used to setup the simulation initial state
to setup
  clear-all ;resets the world to its initial state
  reset-ticks ;creates or resets the tick counter
  show "cleared"

  ; see Agentset in NetLogo dictionary
  ; ask agentset [commands]
  ; n-of size agentset
  ask n-of population patches[
    sprout 1[
      ;you could set the turtle properties here
      ]
  ]
end
```

NetLogo Model Skeleton: step, go

- You can then create two more buttons:
  - one ***step*** button that executes one step of the simulation.
  - one ***go*** forever button (a button that executes a given command continuously) that executes the step command forever or untill a given condition is met.
- Associate the respective commands to each button.
- **Note:** you can just call step on the go button or add multiple commands to it (see model "walk 2 cluster.nlogo" and press edit on the go button).

NetLogo Model Skeleton: step, go

- In your setup command you want to:
  - clear the model components.
  - reset all the variables to their default initial values.
  - create the initial state for your model:
    - In our example (walk 1.nlogo), we ask a set of random patches to create (***sprout*** command) a set of turtles.
    - The sprout command creates a turtle on the same patch that calls it.
- In your step command you will:
  - ask a group of agents to do something:

```
ask turtles [
   <set of instructions, either existing netlogo commands or newly created commands>
]
```

NetLogo Model Skeleton: step, go

```
;procedure to determine what happens in a simulation step
to step
  ask turtles [
     ;facexy takes two coordinates in this case, random
     facexy random world-width random world-height
     forward 1
  ]
  tick ; updates the tick count
end

;call step forever (this is called by a forever button)
to go
  step
end
```

Model Skelleton Final Remarks

- Notice that population is used as a value to define how many turtles will be created. This comes from the population slider**.**
- Open the "walk 2 cluster.nlogo" **–** things to notice:
  - observe that turtle behaviour can be encapsulated by user created commands.
  - the **show** command is executed either by a **turtle** or by the **observer**, depending on where it is called in the code.
  - Try adding a "show color" to a command called by a turtle.
  - Having all the turtles outputing text **slows down the simulation**, keep it to a minimum and debug visualy or by using commands.

NetLogo Programming Environment

- We can program NetLogo models using:
  - NetLogo built-in commands.
  - User-defined procedures.
  - NetLogo or user-defined reporters (model "walk 3 reporters plots.nlogo")

```
;report the ratio of turtles that walked in the last step
to-report average-walking-turtles
  let sum-walked 0
  ask turtles [
    if turtle-walked? [
      set sum-walked sum-walked + 1
    ]
  ]
  report sum-walked / population
end
```

Good practices and tips

- Indent your code.

```
;report the ratio of turtles that walked in the last step
to-report average-walking-turtles
  let sum-walked 0
  ask turtles [
    if turtle-walked? [
      set sum-walked sum-walked + 1
    ]
  ]
  report sum-walked / population
end
```

- Tip: Double clicking just outside a square bracked highligts the corresponding code section.

```
;procedure to determine what happens in a simulation step
to step
  ask turtles [
      ;facexy takes two coordinates in this case, random
      facexy random world-width random world-height
      forward 1
  ]
  tick ; updates the tick count
end
```

- Add comments.

NetLogo Documentation

- Keep the documentation at hand: http://ccl.northwestern.edu/netlogo/docs/dictionary.html

NetLogo Dictionary

NetLogo 5.0.4 User Manual

Alphabetical: A B C D E F G H I J L M N O P R S T U V W X Y ?

Categories: Turtle - Patch - Agentset - Color - Task - Control/Logic - World - Perspective
Input/Output - File - List - String - Math - Plotting - Links - Movie - System - HubNet

Special: Variables - Keywords - Constants

Categories

This is an approximate grouping. Remember that a turtle-related primitive might still be used by patches or the observer, and vice versa. To see which agents (turtles, patches, links, observer) can actually run a primitive, consult its dictionary entry.

Turtle-related

back (bk) *<breeds>*-at *<breeds>*-here *<breeds>*-on can-move? clear-turtles (ct) create-*<breeds>* create-ordered-*<breeds>* create-ordered-turtles (cro) create-turtles (crt) die distance distancexy downhill downhill4 dx dy face facexy forward (fd) hatch hatch-*<breeds>* hide-turtle (ht) home inspect is-*<breed>*? is-turtle? jump layout-circle left (lt) move-to myself nobody no-turtles of other patch-ahead patch-at patch-at-heading-and-distance patch-here patch-left-and-ahead patch-right-and-ahead pen-down (pd) pen-erase (pe) pen-up (pu) random-xcor random-ycor right (rt) self set-default-shape __set-line-thickness setxy shapes show-turtle (st) sprout sprout-*<breeds>* stamp stamp-erase subject subtract-headings tie towards towardsxy turtle turtle-set turtles turtles-at turtles-here turtles-on turtles-own untie uphill uphill4

Patch-related

clear-patches (cp) diffuse diffuse4 distance distancexy import-pcolors import-pcolors-rgb inspect is-patch? myself neighbors neighbors4 nobody no-patches of other patch patch-at patch-ahead patch-at-heading-and-distance patch-here patch-left-and-ahead patch-right-and-ahead patch-set patches patches-own random-pxcor random-pycor self sprout sprout-*<breeds>* subject turtles-here

Agentset

all? any? ask ask-concurrent at-points *<breeds>*-at *<breeds>*-here *<breeds>*-on count in-cone in-radius is-agent? is-agentset? is-patch-set? is-turtle-set? link-heading link-length link-set link-shapes max-n-of max-one-of member? min-n-of min-one-of n-of neighbors neighbors4 no-patches no-turtles of one-of other patch-set patches sort sort-by sort-on turtle-set turtles with with-max with-min turtles-at turtles-here turtles-on

Next session: NetLogo programming language

- Variables, procedures and reporters
- Basic operators.
- Variable scopes and code contexts.
- Control flow.
- NetLogo dictionary: testing built-in commands.