



Introduction to NetLogo

Lecture 2016

Stéphane GALLAND



During this lecture, I will present:

- 1 the principles of NetLogo.
- 2 the basics of the NetLogo development environment.
- 3 the programming concepts of NetLogo.
- 4 a tutorial for simulating termites.



- 1 What is NetLogo?
- 2 Programming Concepts of NetLogo
- 3 Graphical Interface of NetLogo
- 4 Basics of the Model Design and Execution
- 5 Tutorial: termites
- 6 Conclusion



- 1 What is NetLogo?
 - Modeling Complex Systems
 - Brief History
 - Base Concepts of NetLogo
- 2 Programming Concepts of NetLogo
- 3 Graphical Interface of NetLogo
- 4 Basics of the Model Design and Execution
- 5 Tutorial: termites
- 6 Conclusion



- 1 What is NetLogo?
 - Modeling Complex Systems
 - Brief History
 - Base Concepts of NetLogo
- 2 Programming Concepts of NetLogo
- 3 Graphical Interface of NetLogo
- 4 Basics of the Model Design and Execution
- 5 Tutorial: termites
- 6 Conclusion



Programmable modeling environment for simulating natural and social phenomena

- Well suited for modeling complex systems evolving over time.
- Hundreds or thousands of independent agents operating concurrently.
- Exploring the connection between the micro-level behavior of individuals and the macro-level patterns that emerge from the interaction of many individuals.



Easy-to-use application development environment

- Opening simulations and playing with them.
- Creating custom models: quickly testing hypotheses about self-organized systems.
- Models library: large collection of pre-written simulations in natural and social sciences that can be used and modified
- Simple scripting language.
- User-friendly graphical interface



- 1 What is NetLogo?
 - Modeling Complex Systems
 - **Brief History**
 - Base Concepts of NetLogo
- 2 Programming Concepts of NetLogo
- 3 Graphical Interface of NetLogo
- 4 Basics of the Model Design and Execution
- 5 Tutorial: termites
- 6 Conclusion



LOGO [Papert, 1993]



- Theory of education based on Piaget's constructionism ("hands-on" creation and test of concepts).
- Simple language derived from LISP.
- Turtle graphics and exploration of "microworlds".

StarLogo [Resnick, 1991], OpenStarLogo, MacStarLogo, StarLogoT, StarLogo TNG



- Agent-based simulation language.
- Exploring the behavior of decentralized systems through concurrent programming of 100s of turtles

NetLogo [Wilensky, 2015, Tisue, 2004]



- Further extending StarLogo (continuous turtle coordinates, cross-platform, networking, etc.).
- Most popular today (growing cooperative library of models).



- 1 What is NetLogo?
 - Modeling Complex Systems
 - Brief History
 - Base Concepts of NetLogo
- 2 Programming Concepts of NetLogo
- 3 Graphical Interface of NetLogo
- 4 Basics of the Model Design and Execution
- 5 Tutorial: termites
- 6 Conclusion



Each patch is a part of the background or “landscape”



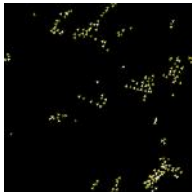
Belousov-Zhabotinsky Reaction



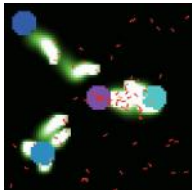
Fur, or how the animal skin is drawn



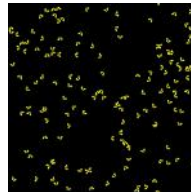
Turtles are entity that move around on top of the patches



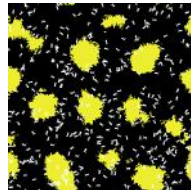
Flocking of birds



Ants



Fireflies synchronization



Termites



1 What is NetLogo?

2 Programming Concepts of NetLogo

- Agents
- Variables
- Procedures
- Asks
- Agent Sets
- Breeds
- Parallel Execution

3 Graphical Interface of NetLogo

4 Basics of the Model Design and Execution

5 Tutorial: termites



- 1 What is NetLogo?
- 2 Programming Concepts of NetLogo
 - Agents
 - Variables
 - Procedures
 - Asks
 - Agent Sets
 - Breeds
 - Parallel Execution
- 3 Graphical Interface of NetLogo
- 4 Basics of the Model Design and Execution
- 5 Tutorial: termites



Agent [Wooldridge, 2001]

An agent is an entity with (at least) the following attributes / characteristics:

- Autonomy
- Reactivity
- Pro-activity
- Social Skills - Sociability



Agent [Ferber, 1999]

An agent is a physical or virtual entity that:

- 1 is able to act inside an environment;
- 2 can directly interact with other agents;
- 3 is driven by a set of tendencies (individual goals, satisfaction or living functions to optimize);
- 4 owns its own resources;
- 5 is able to perceive its environment with a limited extent;
- 6 has a partial knowledge on the environment (possibly none);
- 7 owns skills and offers services;
- 8 can reproduce itself; and
- 9 has a behavior aiming to satisfy the agent goals, by taking into account its resources, skills, perception, and communications.



Turtles

- Move on top of the patches, not necessarily in their center.
- Have **decimal** coordinates (**xcor**, **ycor**) and orientation (**heading**).

Patches

- Don't move, form a 2D wrap-around grid.
- Have **integer** coordinates (**pxcor**, **pycor**).

Links

- Edges (oriented or not) between two turtles.
- **Not presented in this lecture.**

Observer – The User

- Can create new turtles.
- Can have read/write access to all the agents and variables.



- 1 What is NetLogo?
- 2 Programming Concepts of NetLogo
 - Agents
 - **Variables**
 - Procedures
 - Asks
 - Agent Sets
 - Breeds
 - Parallel Execution
- 3 Graphical Interface of NetLogo
- 4 Basics of the Model Design and Execution
- 5 Tutorial: termites



Variable

Place to store values, such as numbers or text.

Types of Variables

Three types of variables:

- 1 Global variables
- 2 Agent variables
- 3 Local variables



Definition

- A global variable can be accessed by all the agents.
- It must be declared before all the procedures.

Example

```
globals [  
    max_energy ;; maximum energy  
    speed ;; traversed cells per simulation step  
]
```



Definition

- A global variable can be accessed by all the agents.
- It must be declared before all the procedures.

Example

```
globals [  
    max_energy ;; maximum energy  
    speed ;; traversed cells per simulation step  
]
```

The keyword for declaring global variables.



Definition

- A global variable can be accessed by all the agents.
- It must be declared before all the procedures.

Example

```
globals [  
    max_energy ;; maximum energy  
    speed ;; traversed cells per simulation step  
]
```

The brackets are the
delimiter of the declaration
block.



Definition

- A global variable can be accessed by all the agents.
- It must be declared before all the procedures.

Example

```
globals [  
  max_energy ;; maximum energy  
  speed ;; traversed cells per simulation step  
]
```

Name of the variable that is declared at the global scope.



Definition

- A global variable can be accessed by all the agents.
- It must be declared before all the procedures.

Example

```
globals [  
    max_energy ;; maximum energy  
    speed ;; traversed cells per simulation step  
]
```

A comment starts with the ";;" characters; and ends at the end of the line.



Definition

- Each turtle and patch has its own set of variables, named agent variables.
- The value of an agent variable may differ from agent to agent.

Examples

```
turtles-own [  
  life  
]
```

```
patches-own [  
  grass_quantity  
]
```



Definition

- A variable is defined and accessible only inside a procedure.
- Scope: the narrowest square brackets, or the procedure itself.

Example

```
to permuter [val1 val2]
  let tmp val1
  set val1 val2
  set val2 tmp
end
```



Definition

- A variable is defined and accessible only inside a procedure.
- Scope: the narrowest square brackets, or the procedure itself.

Example

```
to permuter [val1 val2]  
  let tmp val1  
  set val1 val2  
  set val2 tmp  
end
```

Declare the variable "tmp",
and **copy** the value of the
variable "val1" inside "tmp".



Definition

- A variable is defined and accessible only inside a procedure.
- Scope: the narrowest square brackets, or the procedure itself.

Example

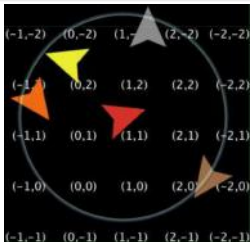
```
to permuter [val1 val2]
  let tmp val1
  set val1 val2
  set val2 tmp
end
```

Copy the value of the variable "val2" inside "val1".



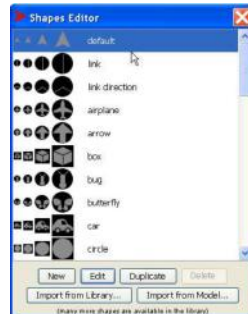
For turtles

- **color**
- **heading** — orientation in degrees
- **label** — name
- **shape**
- **size**
- **xcor** — coordinate along x axis
- **ycor** — coordinate along y axis
- **who** — identifier



For patches

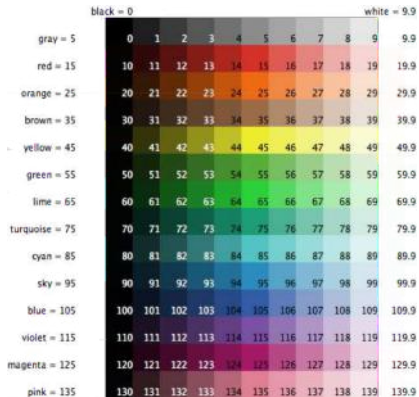
- **pcolor** — color
- **pxcor** — coordinate along x axis
- **pycor** — coordinate along y axis
- **plabel** — label





Colors are defined by:

- a floating-point number in [0; 140]: `ask patches [set pcolor 55]`
- the RGB code: `ask patches [set pcolor [0 255 0]]`
- a predefined name: `ask patches [set pcolor green]`





1 What is NetLogo?

2 Programming Concepts of NetLogo

- Agents
- Variables
- **Procedures**
- Asks
- Agent Sets
- Breeds
- Parallel Execution

3 Graphical Interface of NetLogo

4 Basics of the Model Design and Execution

5 Tutorial: termites



Definition

- Actions for the agents to carry out.
- The command does not reply a value (“void” functions).

- Declaration:

```
to <command_name> [<parameter1> <parameter2> ...]  
  <commands>  
end
```

- Call: <command_name> <argument1> <argument2>

Example

```
to INITIALISE  
  clear-all ;; destroy all agents  
  create-turtles 100 ;; create 100 turtles  
end
```




Definition

- Set of procedures for computing a value.
- The reporter replies a value (a true functions).
- The value is reported with the **report** keyword (equivalent to “return” in other languages).
- Declaration:

```
to -report <reporter_name> [<parameter1> <parameter2> ...]  
  <commands>  
  report <value>  
end
```

- Call: <reporter_name> <argument1> <argument2>

Example

```
to -report abs [nb] ;; Replies the absolute value of nb  
  ifelse nb >= 0  
    [ report nb ]  
    [ report (-nb) ]  
end
```



- Built-in commands or reporters (language keywords).
- Some have an abbreviated form:
create-turtles \Leftrightarrow crt
clear-all \Leftrightarrow ca



1 What is NetLogo?

2 Programming Concepts of NetLogo

- Agents
- Variables
- Procedures
- **Asks**
- Agent Sets
- Breeds
- Parallel Execution

3 Graphical Interface of NetLogo

4 Basics of the Model Design and Execution

5 Tutorial: termites



Definition

Specify commands to be run by turtles or patches.

Asking all turtles

- **Syntax:** `ask turtles [<commands>]`
- **Example:** Asking all turtles to move 50 units forward.
`ask turtles [fd 50]`

Asking all patches

- **Syntax:** `ask patches [<commands>]`

Asking one turtle

- **Syntax:** `ask turtle <id> [<commands>]`

Asking one patch

- **Syntax:** `ask patch <x> <y> [<commands>]`

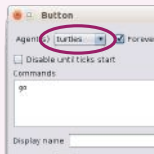


Constraint 1

Cannot be factored out in button specs.

Example:

```
to go [  
  ask turtles [ ... ]  
]
```



Constraint 2

Observer code **cannot** be inside any ask block.

Example: creating 100 turtles.

```
crt 100
```



1 What is NetLogo?

2 Programming Concepts of NetLogo

- Agents
- Variables
- Procedures
- Asks
- **Agent Sets**
- Breeds
- Parallel Execution

3 Graphical Interface of NetLogo

4 Basics of the Model Design and Execution

5 Tutorial: termites



Definition

Definition of a subset of agents.

Grouping with a condition

- Create the subset with agents that are validating a given condition.
- **Syntax:** `<agentset> with [<condition>]`
- **Examples:** All red turtles.

turtles with [color = red]

All red turtles on the patch of the current caller (turtle or patch).

turtles-here with [color = red]



Grouping according to distance

- Create the subset with agents that are at a distance lower than or equal to a given one.
- **Syntax:** `<agentset> in-radius<distance>`
- **Examples:** All turtles less than 3 patches away from caller (turtle or patch).

turtles in-radius 3

Grouping with a relative position

- Create the subset with agents that are located at the given positions, relatively to the caller.
- **Syntax:** `<agentset> at-points[[x1 y1] [x2 y2] ...]`
- **Examples:** The four patches that are the neighbors of the caller.

patches at-points[[1 0] [0 1] [-1 0] [0 -1]]

The previous code has a shorthand: `neighbors4`



Executing a command

- Ask such agents to execute a command.
- **Syntax:** `ask <agentset> [...]`
 - **Examples:** All red turtles should go forward of 1 cell.
`ask turtles with [color = red] [fd 1]`

Is the set empty?

- Check if there are agents in the set.
- **Syntax:** `any? <agentset>`
- **Examples:** Show if there is any red turtle?
`show any? turtles [color = red]`

Size of the set

- Count the agents inside a set.
- **Syntax:** `count <agentset>`
- **Examples:** Show the number of red turtles.
`show count turtles [color = red]`



1 What is NetLogo?

2 Programming Concepts of NetLogo

- Agents
- Variables
- Procedures
- Asks
- Agent Sets
- **Breeds**
- Parallel Execution

3 Graphical Interface of NetLogo

4 Basics of the Model Design and Execution

5 Tutorial: termites



Definition

A “natural” kind of agent set (other species than turtles).

Declaration of breeds

- For declaring new breeds, you should use the “**breed**” keyword.
- **Syntax:** **breed** [<breeds> <**breed**>]
 - First value is the plural for the breed.
 - Second value is the singular for the breed.
- **Example:** Declaring breed for mice.
breed [mice mouse]



When a new breed is defined, derived primitives are automatically created:

- `create-<breeds> <number>`
Create the given number of agents of the breed.
- `create-<breeds> <number> [<commands>]`
Create the given number of agents of the breed, and run immediately the given commands.
- `<breed>-own`
Declare variables for all the members of the given breed.
- `<breed>-here`
An agentset containing all the turtles of the given breed on the caller's patch.
- `<breed>-at <dx> <dy>`
An agentset containing all the turtles of the given breed on patch (dx, dy) from the caller.
- etc. (See NetLogo documentation, <http://ccl.northwestern.edu/netlogo/docs>).



The breed is a variable of the turtle.

Examples

- Ask turtle no. 5 to do something if it is a sheep.

```
ask turtle 5 [ if breed = sheep ... ]
```

- Ask turtle no. 5 to change its breed to “wolf.”

```
ask turtle 5 [ set breed = wolf ]
```



1 What is NetLogo?

2 Programming Concepts of NetLogo

- Agents
- Variables
- Procedures
- Asks
- Agent Sets
- Breeds
- **Parallel Execution**

3 Graphical Interface of NetLogo

4 Basics of the Model Design and Execution

5 Tutorial: termites

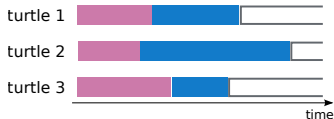


Agents run in parallel (each agent is an independent thread).

Example

Execute commands in parallel by all the turtles.

```
ask turtles [  
  fd random 10  
  do-calculation  
  ... ]
```





Agent threads wait and join at the end of an execution block.

Example

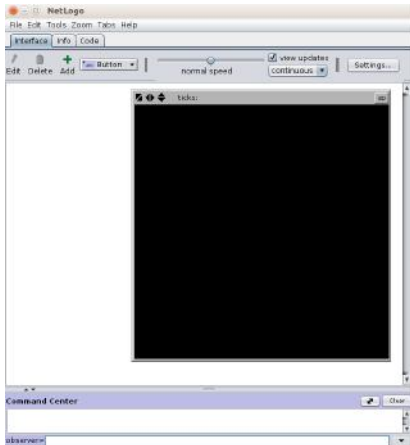
All turtles wait for the first commands to be run by all of them before running the second commands.

```
ask turtles [ fd random 10 ]  
ask turtles [ do-calculation ]  
...
```





- 1 What is NetLogo?
- 2 Programming Concepts of NetLogo
- 3 Graphical Interface of NetLogo**
- 4 Basics of the Model Design and Execution
- 5 Tutorial: termites
- 6 Conclusion



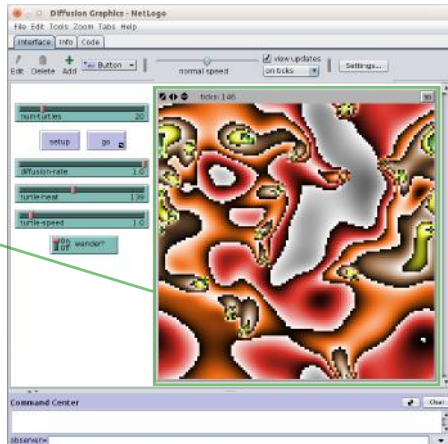
Three tabs:

- **Interface:** viewer and UI tools for the simulation.
- **Info:** Text information on the simulated model.
- **Code:** NetLogo source code of the simulated model.



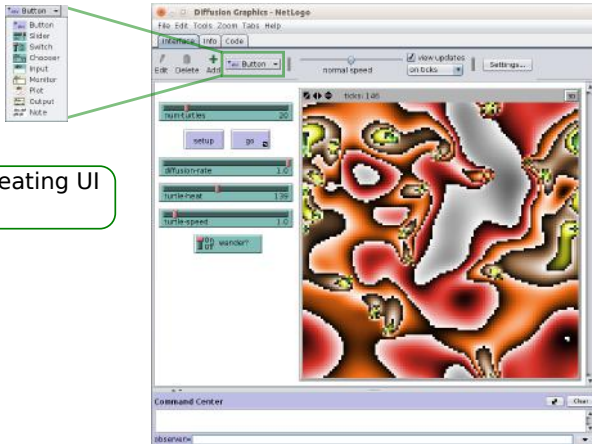


A component that displays the patches.



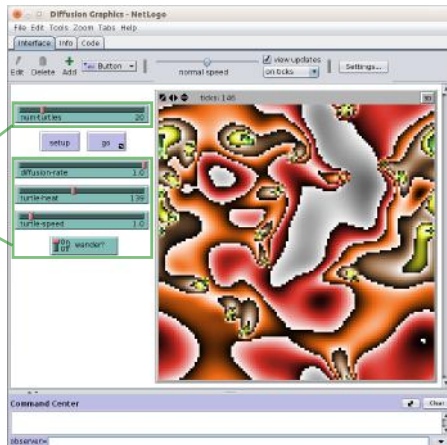


Button for creating UI components



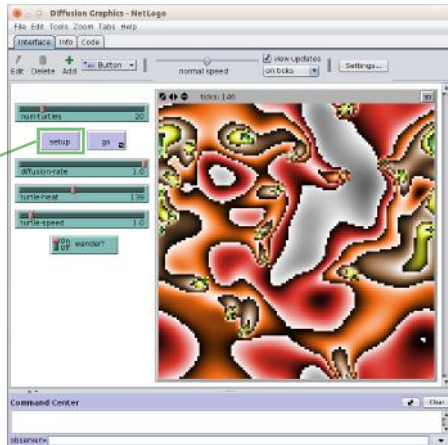


UI components that allow the user to change variables' values.





A button for launching a command.





How to edit the button settings

Select the runner of the command: observer, turtles, patches, links.





How to edit the button settings

Type the command(s) to run when the button is clicked.





How to edit the button settings

Change the name of the button.





setup

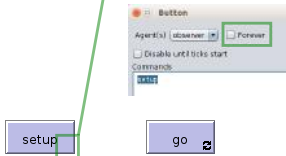
go





No symbol at the bottom right corner of the button.

The commands are run once at each click.





Loop symbol at the bottom right corner of the button.

The commands are run in an loop until the button is pressed again.





UI component for typing commands.

Type a command here is equivalent to ask to an agent to run the command.





The commands below are equivalent:

- first, ask the observer.
 - second, ask turtles.
- (click on the label for changing the caller)

Command Center

```
observer> ask turtles [ fd 1 ]
```

Command Center

```
turtles> fd 1
```





WHAT IS IT?

Diffusion Graphics is unlike most other NetLogo models, in that it really doesn't 'model' anything. It simply explores the power behind an interesting patch primitive: 'diffuse'.

It's not intended to closely model real heat, just a number that behaves something like heat – that slowly spreads itself evenly across a plane.

HOW IT WORKS

In this model, the turtles are "hot spots" – they set a certain value (a patch variable called 'heat') to the maximum level every time step. Each patch (through the 'diffuse' primitive) then shares its value of 'heat' with its surrounding patches.

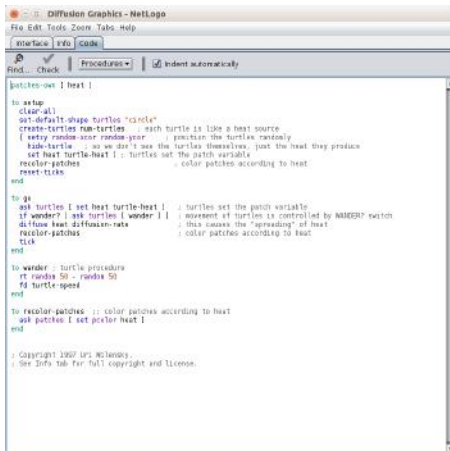
Here you can watch what happens as hot-spots interact with each other, as they move around, as their values become negative, or as the 'heat' slowly decays down to nothing. The whole point of the project is to give you an idea how patches interact via the 'diffuse' primitive. (Or maybe just to give you something nice to stare at if you're bored.)

HOW TO USE IT

Two buttons, SETUP and GO, control execution of the model. As in most NetLogo models, the SETUP button will initialize the 'hot-spots' and other variables, preparing the model to be run. The GO button, a forever button, will then run the model.

Four sliders and two switches determine the various properties of the model. Each of them can be set prior to initialization; most can be used mid-run to affect what will happen.

NUM-TURTLES determines how many turtles there are. TURTLE-SPEED determines



```
Diffusion Graphics - NetLogo
File Edit Tools Zoom Tabs Help

interface info code
And... Check Procedures Indent automatically

[patches-own | heat |

to setup
  clear-all
  set-default-shape turtles "circle"
  create-turtles run-turtles ; each turtle is like a heat source
  ( setxy random-accor random-accor ; position the turtles randomly
    hide-turtle ; so we don't see the turtles themselves, just the heat they produce
    set heat turtle-heat | ; turtles set the patch variable
    recolor-patches ; color patches according to heat
    reset-ticks
  )
end

to go
  ask turtles [ set heat turtle-heat | ; turtles set the patch variable
    if wander? [ ask turtles [ wander | | movement of turtles is controlled by wander? switch
    diffuse heat diffusion-rate ; this causes the "spreading" of heat
    recolor-patches ; color patches according to heat
    tick
  ]
end

to wander ; turtle procedure
  rt random 50 - random 50
  fd turtle-speed
end

to recolor-patches ; color patches according to heat
  ask patches [ set pcolor heat |
end

; Copyright 1997 Uri Wilensky.
; See Info tab for full copyright and license.
```



Area where the NetLogo code is displayed, and can be edited.

```
Diffusion Graphics - NetLogo
File Edit Tools Zoom Tabs Help
interface info code
Find Check Procedures Indent automatically

patches-on 1 heat

to setup
  clear-all
  set-default-shape turtles "circle"
  create-turtles run-turtles ; each turtle is like a heat source
  [ setxy random-accor random-year ; position the turtles randomly
    hide-turtle ; so we don't see the turtles themselves, just the heat they produce
    set heat turtle-heat ; turtles set the patch variable
    recolor-patches ; color patches according to heat
  ]
end

to go
  ask turtles [ set heat turtle-heat ; turtles set the patch variable
    if wander? [ ask turtles [ wander ] ; movement of turtles is controlled by NUMBER? switch
    diffuse heat diffusion-rate ; this causes the "spreading" of heat
    recolor-patches ; color patches according to heat
    tick
  ]
end

to wander ; turtle procedure
  rt random 50 - random 50
  fd turtle-speed
end

to recolor-patches ; color patches according to heat
  ask patches [ set pcolor heat ]
end

; Copyright 1997 Uri Wilensky.
; See Info tab for full copyright and license.
```



Drop-down list that contains all the defined commands. It enables you to go to the selected command in the code area.

```
Diffusion Graphics - NetLogo
File Edit Tools Zoom Tabs Help
interface info code
Find... Check Procedures Indent automatically

patches-on heat |
to setup
  clear-all
  set-default-shape turtles "circle"
  create-turtles run-turtles ; each turtle is like a heat source
  ( setxy random-axon random-year ; position the turtles randomly
    hide-turtle ; so we don't see the turtles themselves, just the heat they produce
    set heat turtle-heat | ; turtles set the patch variable
    recolor-patches ; color patches according to heat
    reset-ticks
  )
end

to go
  ask turtles [ set heat turtle-heat | ; turtles set the patch variable
    if wander? [ ask turtles [ wander ] | ; movement of turtles is controlled by NUMBER? switch
    diffuse heat diffusion-rate ; this causes the "spreading" of heat
    recolor-patches ; color patches according to heat
    tick
  ]
end

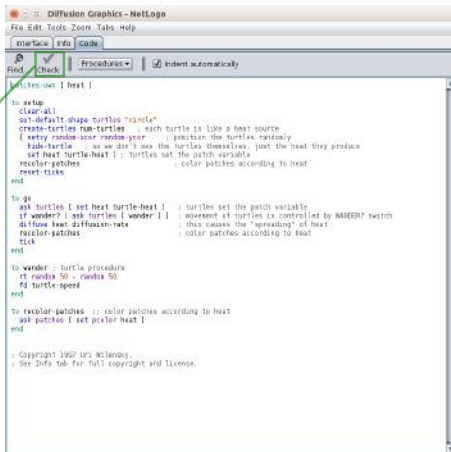
to wander ; turtle procedure
  rt random 50 - random 50
  fd turtle-speed
end

to recolor-patches ; color patches according to heat
  ask patches [ set pcolor heat ]
end

; Copyright 1997 Uri Wilensky.
; See Info tab for full copyright and license.
```



Button for starting the process of verification of the NetLogo syntax in the code area.



```
Diffusion Graphics - NetLogo
File Edit Tools Zoom Tabs Help
interface info code
[Check] [Procedures] [Indent automatically]

[atches-on? | heat |]

to setup
  clear-all
  set-default-shape turtles "circle"
  create-turtles run-turtles ; each turtle is like a heat source
  [ setxy random-accor random-accor ; position the turtles randomly
    hide-turtle ; so we don't see the turtles themselves, just the heat they produce
    set heat turtle-heat | ; turtles set the patch variable
    recolor-patches ; color patches according to heat
  ]
end

to go
  ask turtles [ set heat turtle-heat | ; turtles set the patch variable
    if wander? [ ask turtles [ wander | | ; movement of turtles is controlled by wander? switch
    diffuse heat diffusion-rate ; this causes the "spreading" of heat
    recolor-patches ; color patches according to heat
    tick
  ]
end

to wander ; turtle procedure
  rt random 50 - random 50
  fd turtle-speed
end

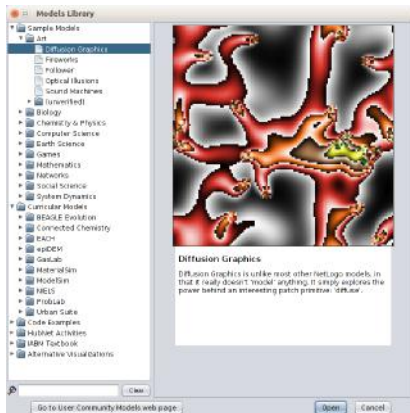
to recolor-patches ; color patches according to heat
  ask patches [ set pcolor heat |
end

; Copyright 1997 Uri Wilensky.
; See Info tab for full copyright and license.
```



NetLogo provides plenty of ready-to-use models in its **model library**

- Select the menu item:
> File > Models Library
- Double click on the predefined model to load.





- 1 What is NetLogo?
- 2 Programming Concepts of NetLogo
- 3 Graphical Interface of NetLogo
- 4 Basics of the Model Design and Execution**
- 5 Tutorial: termites
- 6 Conclusion



1 Definition of:

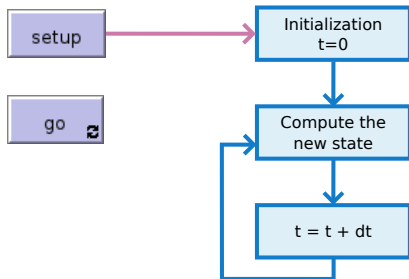
- the different breeds of turtles.
- the global variables.
- the agent variables.

2 Definition of the init procedure:

- The command is usually named “setup”.
- The command is usually invoked by the “setup” button.
- The command generates the agents, and initializes the variables.

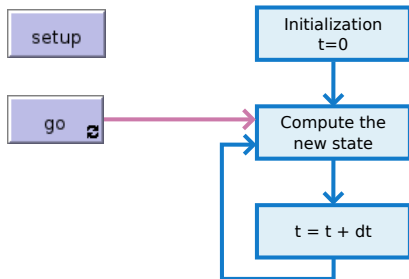
3 Definition of the dynamics of the system:

- The command is usually named “go”.
- The command is usually invoked by the “go” button.
- The sub-procedures are called into the go command.



```
to setup
clear-all
...
reset-ticks
end
```

- Initialize the simulation.
- **clear-all**
Call the clearing functions: clear-globals, clear-ticks, clear-turtles, clear-patches, clear-drawing, clear-all-plots, and clear-output.
- **reset-ticks**
Set the current time t to zero.



```
to go
...
display
tick
end
```

- Run one step of the simulation (eventually in a loop).
- **display**
Update the display of the NetLogo interface.
- **tick**
Increment the current time t by one.



Time t

The time t is an information internally declared by NetLogo. The model is in charge of making the time evolving.

Getting the time t

ticks

Advance the time

- **tick**
Advance by one the time t .
- **tick-advance<dt>**
Advance by <dt> the time t .

Reset the time

- **clear-ticks**
Unset the time t (not equal to zero).
- **reset-ticks**
Set the time t to zero.



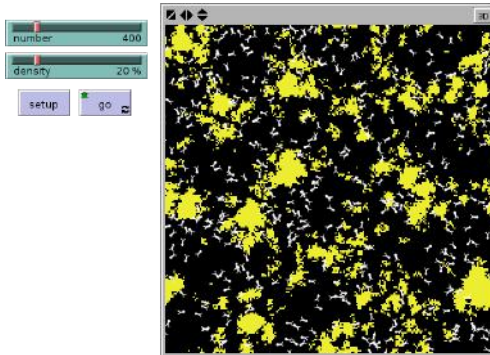
- 1 What is NetLogo?
- 2 Programming Concepts of NetLogo
- 3 Graphical Interface of NetLogo
- 4 Basics of the Model Design and Execution
- 5 Tutorial: termites**
 - Step 1: Build the Interface
 - Step 2: Setup
 - Step 3: Go
 - Step 4: Adding outputs
- 6 Conclusion



- 1 What is NetLogo?
- 2 Programming Concepts of NetLogo
- 3 Graphical Interface of NetLogo
- 4 Basics of the Model Design and Execution
- 5 Tutorial: termites**
 - Step 1: Build the Interface
 - Step 2: Setup
 - Step 3: Go
 - Step 4: Adding outputs
- 6 Conclusion



Build the interface by creating:



- 1 two buttons:
 - **setup**
observer, once.
 - **go**
turtles, forever.
- 2 two sliders:
 - **number**
1 \rightarrow 300(1)
 - **density**
0 \rightarrow 100%(1)



- 1 What is NetLogo?
- 2 Programming Concepts of NetLogo
- 3 Graphical Interface of NetLogo
- 4 Basics of the Model Design and Execution
- 5 Tutorial: termites**
 - Step 1: Build the Interface
 - **Step 2: Setup**
 - Step 3: Go
 - Step 4: Adding outputs
- 6 Conclusion



- Randomly strew yellow wood chips with a given density.
- The chips are patches.
- `density` is the variable that is automatically created from the slider.

```
to setup-chips
  ask patches [
    if random-float 100 < density
      [ set pcolor yellow ]
  ]
end
```



- Randomly position a given number of white termites.
- The termites are turtles.
- `number` is the variable that is automatically created from the slider.

```
to setup-termites
  set-default-shape turtles "bug"
  create-turtles number
  ask turtles [
    set color white
    setxy random-float screen-size-x
      random-float screen-size-y
    ;; or
    ;; setxy random-xcor random-ycor
  ]
end
```




- Call the setup commands for setting the system up.

```
to setup
  clear – all
  setup–chips      ;; Create the environment
  setup–termites   ;; Create the population
  reset – ticks
end
```



- 1 What is NetLogo?
- 2 Programming Concepts of NetLogo
- 3 Graphical Interface of NetLogo
- 4 Basics of the Model Design and Execution
- 5 Tutorial: termites**
 - Step 1: Build the Interface
 - Step 2: Setup
 - **Step 3: Go**
 - Step 4: Adding outputs
- 6 Conclusion



Termites follow 3 rules:

- 1 Look around for a wood chip and pick it up.
- 2 Look around for a pile of wood chips.
- 3 Look around for an empty spot in the pile and drop off the chip.

```
to go      ;; turtle code
  search-for-chip
  find-new-pile
  drop-off-chip
  tick
end
```



Termites explore the environment through random walk:

```
to explore
  fd 1
  ;; Compute an angle between -50 and 50
  let angle random-float 50
           - random-float 50
  rt angle
end
```



Find a wood chip, pick it up and turn orange:

```
;; Recursive version
to search-for-chip
  ifelse pcolor = yellow
    [ set pcolor black
      set color orange
      fd 20 ]
    [ explore
      search-for-chip ]
end
```

```
;; Iterative version
to search-for-chip
  while [ pcolor !=
        yellow ]
    [ explore ]
  set pcolor black
  set color orange
  fd 20
end
```



Find a new pile of chips:

```
;; Recursive version
to find-new-pile
  if pcolor != yellow
    [ explore
      find-new-pile ]
end
```

```
;; Iterative version
to find-new-pile
  while [ pcolor !=
        yellow ]
    [ explore ]
end
```



Find an empty spot, drop off chip and get away:

```
:: Recursive version  
to drop-off-chip  
  ifelse pcolor = black  
    [ set pcolor yellow  
      set color white  
      fd 20 ]  
    [ explore  
      drop-off-chip ]  
end
```

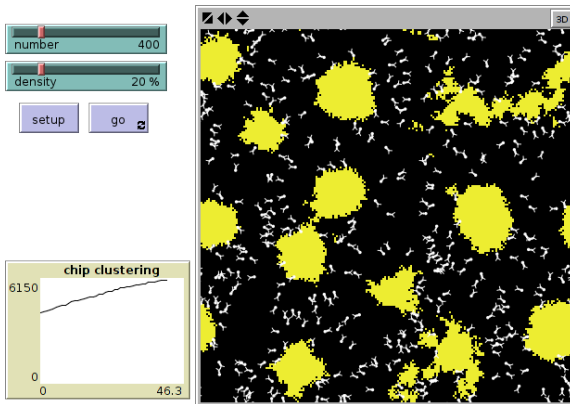
```
:: Iterative version  
to drop-off-chip  
  while [ pcolor !=  
          black ]  
    [ explore ]  
  set pcolor yellow  
  set color white  
  fd 20  
end
```



- 1 What is NetLogo?
- 2 Programming Concepts of NetLogo
- 3 Graphical Interface of NetLogo
- 4 Basics of the Model Design and Execution
- 5 Tutorial: termites**
 - Step 1: Build the Interface
 - Step 2: Setup
 - Step 3: Go
 - **Step 4: Adding outputs**
- 6 Conclusion



Update the interface with a “plot” with the name “chip clustering”:





For updating the plot, you must define the commands to execute for computing the new values to display.

```
to draw-plot
  set-current-plot "chip clustering"
  plot count patches with
    [ count neighbors4 with
      [ pcolor = yellow ]
      = 4 ]
end
```



- The plot drawing command must be call by the observer agent.
- The caller of the “go” command must be change from “turtles” to “observer.”
- The code og the “go” command must be updated to:

```
to go
  ask turtles [
    search-for-chip
    find-new-pile
    drop-off-chip ]
  tick
  draw-plot
end
```



- 1 What is NetLogo?
- 2 Programming Concepts of NetLogo
- 3 Graphical Interface of NetLogo
- 4 Basics of the Model Design and Execution
- 5 Tutorial: termites
- 6 Conclusion**



Name	Domain	Hierar. ^a	Simu. ^b	C.Phys. ^c	Lang.	Beginners ^d	Free
GAMA	Spatial simulations		✓		GAML, Java	**[*]	✓
Jade	General		✓	✓	Java	*	✓
Jason	General		✓	✓	Agent-Speaks	*	✓
Madkit	General		✓		Java	**	✓
NetLogo	Social/natural sciences		✓		Logo	***	✓
Repast	Social/natural sciences		✓		Java, Python, .Net	**	
SARL	General	✓	✓	✓	SARL, Java, Xtend, Python	**[*]	✓

^a Native support of hierarchies of agents.

^b Could be used for agent-based simulation.

^c Could be used for cyber-physical systems, or ambient systems.

^d *: experienced developers; **: for Computer Science Students; ***: for others beginners.

This table was done according to experiments with my students.



Introduction to NetLogo

Lecture 2016

Stéphane GALLAND



Appendix



- 1 Books
 - NetLogo
 - Multiagent Systems
 - Simulation Theory

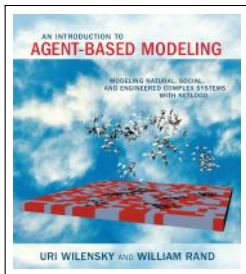
2 Acknowledgements

3 About the Author

4 Bibliography



- 1 Books
 - NetLogo
 - Multiagent Systems
 - Simulation Theory
- 2 Acknowledgements
- 3 About the Author
- 4 Bibliography



Introduction to Agent-Based Modeling: Modeling Natural, Social and Engineered Complex Systems with NetLogo

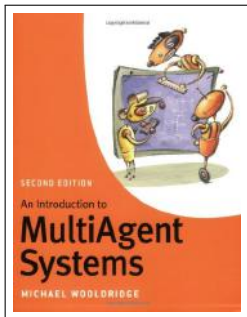
Uri WILENSKY and Bill RAND

The MIT Press, 2015

ISBN 978-0262731898



- 1 Books
 - NetLogo
 - Multiagent Systems
 - Simulation Theory
- 2 Acknowledgements
- 3 About the Author
- 4 Bibliography



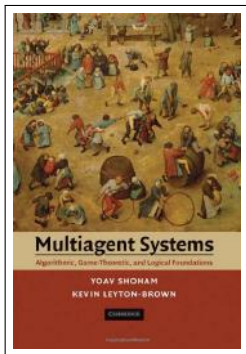
An Introduction to Multiagent Systems

2nd edition

Michael WOOLDRIDGE

Wiley, 2009

ISBN 0-47-0519460

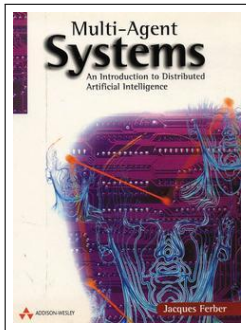


Multiagent Systems: algorithmic, game-theoretic, and logical foundations

Yoav SHOHAM and
Kevin LEYTON-BROWN

Cambridge University Press, 2008

ISBN 0-52-1899435

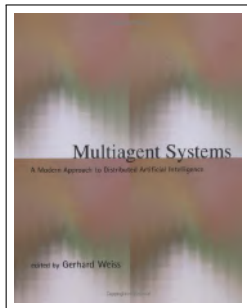


Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence

Jacques FERBER

Addison Wesley, 1999

ISBN 0-20-1360489



Multiagent Systems: a modern approach to distributed Artificial Intelligence

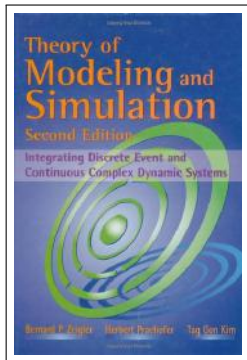
Gerhard WEISS

MIT Press, 2000

ISBN 0-26-2731312



- 1 Books
 - NetLogo
 - Multiagent Systems
 - Simulation Theory
- 2 Acknowledgements
- 3 About the Author
- 4 Bibliography



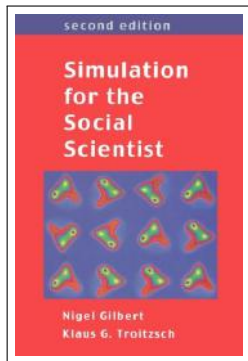
Theory of Modeling and Simulation

2nd edition

Bernard ZEIGLER, Herbert Praehofer, and
Tag Gon Kim

Academic Press, 2000

ISBN 0-12-778455-1



Simulation for the Social Scientist

2nd edition

Nigel GILBERT and Klaus TROITZSCH

Open University Press, 2005

ISBN 0-33-5216005



- 1 Books
- 2 Acknowledgements
- 3 About the Author
- 4 Bibliography



Other lectures

The slides of this lecture was partially based on the lecture slides of [Doursat, 2005] and [Chauvet, 2014].

NetLogo Documentation

Several pictures in this lecture are from the documentation of NetLogo.





- 1 Books
- 2 Acknowledgements
- 3 About the Author**
- 4 Bibliography



Associate Professor

Université de Bourgogne Franche-Comté

Université de Technologie de Belfort-Montbéliard, France

Topics: Multiagent systems, Agent-based simulation, Agent-oriented software engineering, Mobility and traffic modeling



Web page: http://www.multiagent.fr/People:Galland_stephane

Email: stephane.galland@utbm.fr

Open-source contributions:

- <http://www.sarl.io>
- <http://www.janusproject.io>
- <http://www.aspecs.org>
- <http://www.arakhne.org>
- <https://github.com/gallandarakhneorg/>





Chauvet, P. (2014).

NetLogo: Introduction & premiers exemples.

AT11. Université Catholique de l'Ouest, Anger, France.

In French.

Doursat, R. (2005).

Introduction to NetLogo.

CS 790R Seminar. University of Nevada, Reno, USA.

Ferber, J. (1999).

Multiagent Systems: An Introduction to Distributed Artificial Intelligence.

Addison-Wesley Professional.

Papert, S. A. (1993).

Mindstorms: Children, Computers, And Powerful Ideas.

Basic Books, 2nd edition.

Resnick, M. (1991).

Animal simulations with starlogo: Massive parallelism for the masses.

From Animals to Animats.

Tisue, S. and Wilensky, U. (2004).

Netlogo: Design and implementation of a multi-agent modeling environment.

In *Proceedings of the Agent 2004 Conference on Social Dynamics: Interaction, Reflexivity and Emergence*,

Chicago, IL, USA.

Wilensky, U. and Rand, B. (2015).

Introduction to Agent-Based Modeling: Modeling Natural, Social and Engineered Complex Systems with NetLogo.

The MIT Press.



Wooldridge, M. and Ciancarini, P. (2001).

Agent-oriented software engineering: The state of the art.

In *Agent-Oriented Software Engineering: First International Workshop (AOSE 2000)*, volume 1957 of *Lecture Notes in Computer Science*, pages 1–28. Springer-Verlag.