



Introduction to individual-based modeling

Guillaume Beslon
INSA – LIRIS – IXXI



Introduction

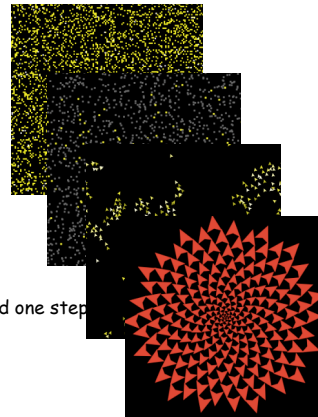
- Aim of the course:
 - Introduce Agent-Based Modeling as a tool
 - Give the main hints to develop your own ABM
 - Present NetLogo as a tool to develop ABMs
 - NetLogo tutorial
 - Build your (first?) NetLogo models
- Who am I?
 - Guillaume BESLON (guillaume.beslon@liris.cnrs.fr)
 - Professor at the INSA-Lyon, LIRIS Lab. (Laboratoire d'Informatique en Image et Systèmes d'Information)
 - Assistant-director of IXXI (Rhône-Alpes Complex Systems Institute)
 - Research topics: Individual-based modeling of complex biological systems (mainly evolution)



- Programmable simulation environment
 - Designed for Agent-Based Modeling
- Used mainly to simulate natural and social systems
 - Available for all systems (java),
 - Can be used in parallel
 - Simple “intuitive” programming language

```
to setup
  ca      ;; clear the screen
  crt 10  ;; make 10 new turtles
end

to go
  ask turtles
  [ fd 1      ;; all turtles move forward one step
    set heading random 360 ;; and turn randomly
  ]
end
```



G. Beslon – CSSS'09 Lecture – July, 23, 2009

3

References

- Bonabeau, E. (2002). Agent-based modeling : Methods and techniques for simulating human systems. *Proceedings of the National Academy of Sciences of the USA (PNAS)*, 99(suppl. 3):7280–7287.
- Grimm, V. (1999). Ten years of individual-based modelling in ecology : what have we learned and what could we learn in the future ? *Ecological Modelling*, 115:129–148.
- Banks, S. (2002). Agent-based modeling : A revolution ? *Proceedings of the National Academy of Sciences of the USA (PNAS)*, 99:7199–7200.
- Grimm, V., Revilla, E., Berger, U., Jeltsch, F., Mooij, W.M., Railsback, S.F., Thulke, H.-H., Weiner, J., Wiegand, T., DeAngelis, D.L. (2005) Pattern-Oriented Modeling of Agent-Based Complex Systems: Lessons from Ecology. *Science*, 310:987-991.
- Macal, C. M. et North, M. J. (2006). Tutorial on agent-based modeling and simulation part 2 : how to model with agents. In WSC06 : *Proceedings of the 38th Winter simulation conference*, Monterey (USA), pages 73–83.

G. Beslon – CSSS'09 Lecture – July, 23, 2009

4

Reminder

- What is a model?

“To an observer B , an object A^* is a model of an object A to the extent that B can use A^* to answer questions that interest him about A .”

[Marvin Minsky]

- Remember that scientific models are instruments for scientific discovery

- Used to explore properties of systems through virtual experiments

- Computational models are those which uses computation to perform the experiments

- The model typically uses an algorithm to compute the state at type t from the state at time $t-1$
- Agent-Based Modeling is a kind of computational models based on an explicit description of the agents.

What is ABM?

- “Bottom-Up” modeling:

- Describe the system at the local level with some formalism
- Simulate it (computational model)
- Observe and analyze the results (at both levels!)

“In agent-based modeling (ABM), a system is modeled as a collection of autonomous decision-making entities called agents. Each agent individually assesses its situation and makes decisions on the basis of a set of rules. Agents may execute various behaviors appropriate for the system they represent -- for example, producing, consuming, or selling. Repetitive competitive interactions between agents are a feature of agent-based modeling, which relies on the power of computers to explore dynamics out of the reach of pure mathematical methods.”

[Bonabeau, 2002]

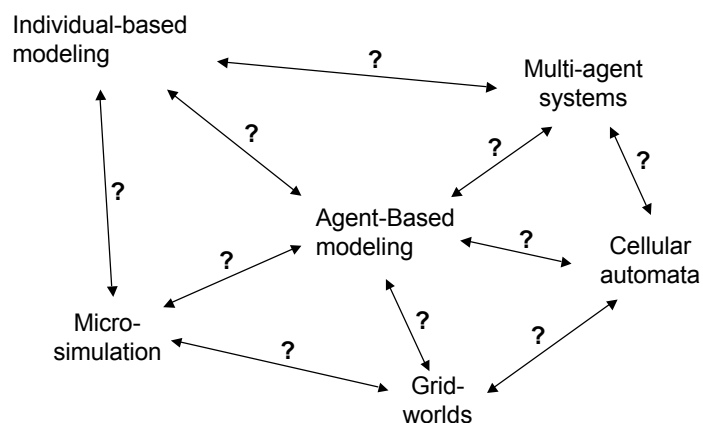
What is ABM?

- Agent-Based Modeling is more a methodology than a precise technique
 - You can choose the formalism you “want” at the agent level (dynamical models, set of rules, discrete/continuous coordinates, punctual particles or not, ...)
 - The only thing you need is a way to compute the interactions and, thus, the resulting behavior
 - But this may not be a trivial question!
 - The used computational tools can be very diverse...

“Agent-Based Model is a mindset more than a technology.”

[Bonabeau, 2002]

What is ABM?

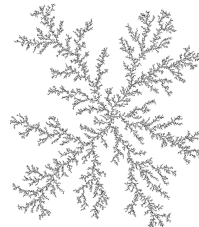


What is ABM?

- Consensus for the principles
- Diversity of the appellations!
 - Micro-simulation (physics)
 - Agent-Based Modeling (computer science, social science)
 - Individual-Based Modeling (biology, ecology)
 - Bottom-Up simulation
- The only real difference is with MAS
 - Multi-Agent Systems are NOT Agent-Based Models
 - MAS are IT technologies trying to use CS approaches to improve the behavior of programs and computers
 - MAS are NOT models
 - MAS can be used to implement ABM but... why?

ABM, Cellular Automata and Grid Worlds

- 2D cellular automata are often presented as ABM
 - In CA rules are associated with the places, not with the agents
 - CA are not ABM, except when dealing with fixed agents (one place-one agent)
- Grid world are 2D worlds (sometimes 3D) where objects move on a grid-based space according to rules
 - The rules are local to the objects, not to the places
 - Probably the simplest ABM
 - E.g., DLA ...

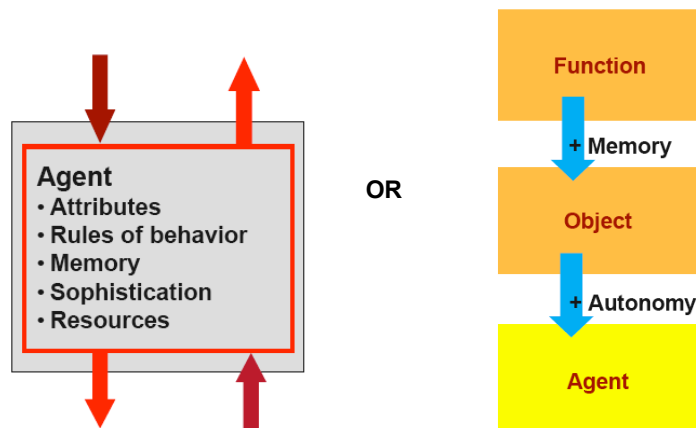


What is an agent?

- What is an agent?
 - A discrete entity/program with its own goals and behaviors
 - Autonomous, with a capability to adapt and modify its behaviors
 - Some key aspect of behaviors can be described.
 - Mechanisms by which agents interact can be described.
- Examples
 - People, groups, organizations, insects, swarms, robots...
- But this definition is strongly rooted in MAS and social systems

What is an agent?

- You will often find figures like:



What is an agent?

- An agent is (only) the unit of description of the micro-level
 - Again, “Agent” is more a methodological concept than a technological concept!
- What is agent (or not) depends on your point of view!
 - What is really important is what is local and what is not!
- It is often very difficult to decide what is an attribute, what is a memory, what is a resource ...
 - E.g. xcor, ycor, speed, energy, ...

What is an agent?

- Care the difference between:
 - “Anthropomorphic” definition: An entity that senses its environment and acts upon it in order to achieve a goal
 - Technical definition: A persistent autonomous software entity dedicated to a specific purpose (e.g. a program, a thread or a robot)
 - Methodological definition: The conceptual unit of interest, defines a boundary between what is modelled and what is observed (but often the observed system is the agent...)

Life-cycle of an ABM

- Developing an ABM seems straightforward!
 - Describe the system at the agent level ; describe the interactions between the agents
 - Create a population of agents
 - Use some simulation method/software to let the agents and the population run
 - Observe the result(s)
 - Draw conclusion
- Actually it is (quite) as simple as this...
 - But some steps may be difficult ;)
- Modeling is an art
 - Agent-Based Modeling is a black-art!

Designing the agent level

- NOT YET: As in every model, define very carefully your system and your aim FIRST!
 - Generally a scientific question but...
 - ABM can also be used to help to define a scientific question!
- Choose the agent level, the agents behavior and the agent interactions
 - Take care: the devil is in the details!
 - You need a good knowledge and skill in order to be able to select the appropriate description at the appropriate level!
 - Care habits, transfer of models from a domain to another one, code reuse, ...
 - Care implicit choices

How to design the agents?

- Actually no real methodology...
 - ABM skill helps,
 - A precise question helps a lot,
 - Domain knowledge helps enormously!
 - The only methodology is trial and errors!
 - Examples of agents
 - Molecules
 - Planets/stars
 - Humans
 - Insects
 - Companies
 - Cars
 - Drops of water
 - Birds...
- Both have similar properties:
inanimate objects following
physical (Newtonian) laws
- ↓
- Can we use the same
agents models

G. Beslon – CSSS'09 Lecture – July, 23, 2009

17

How to choose the “level of complexity” ?

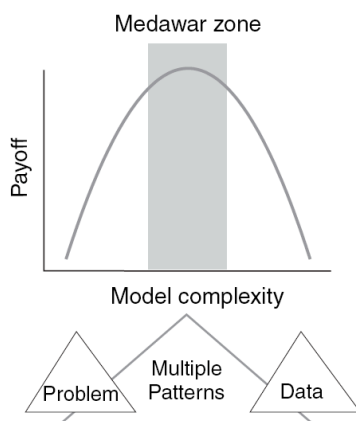


Fig. 1. Payoff of bottom-up models versus their complexity. A model's payoff is determined not only by how useful it is for the problem it was developed for, but also by its structural realism; i.e., its ability to produce independent predictions that match observations. If model design is guided only by the problem to be addressed (which often is the explanation of a single pattern), the model will be too simple. If model design is driven by all the data available, the model will be too complex. But there is a zone of intermediate complexity where the payoff is high. We call this the “Medawar zone” because Medawar described a similar relation between the difficulty of a scientific problem and its payoff (47). If the very process of model development is guided by multiple patterns observed at different scales and hierarchical levels, the model is likely to end up in the Medawar zone.

[Grimm et al., 2005]

G. Beslon – CSSS'09 Lecture – July, 23, 2009

18

The main problem

- The structure and dynamics of complex systems call for Agent-Based Modeling
 - Many elements with non-linear interactions
 - Natural interpretations (Fayerabend) are often false
- But the structure and dynamics of complex systems make modeling difficult and dangerous
 - Complex systems are very sensitive to (generally many) parameters
 - Complex systems are often sensitive to initial conditions
 - Complex systems are often noisy
 - The analysis of the system strongly depends on some subjective judgment
- Playing with models is the only solution!

From agents to multi-agents

- Once you have designed the agents, you still have important choices to make
 - These choices are often forgotten (often implicit!)
- Agent will “live” in a spatio-temporal world
 - Real world is continuous
 - Agents’ world is not!
 - It creates risks and difficulties
- How to model time?
- How to model space?

The time model

- Time is (always?) neglected in MAS approaches
 - Generally considered as a non-problem
- Discrete Time
 - Synchronous, asynchronous, discrete-events
 - What is the correct time step?
 - The higher the time step, the higher the error
 - The lower the time step, the slower the simulation
 - Practitioners are generally NOT able to estimate the correct time step of their systems!
 - The correct time step depends on the movement and on the interaction models
- The time model may strongly influences the global behavior



G. Beslon – CSSS'09 Lecture – July, 23, 2009

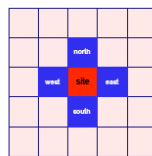
21

The space model

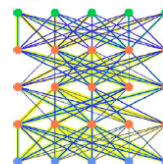
- Space is often at the core of ABM
 - Space is mainly a constraint on agents' neighborhood
 - Very often, you will use ABMs to test the behavior of analytical models in a given spatial framework
- Lots of different space models are possible
 - From “Soup model” to GIS models
 - You often have to mix different space models (e.g. continuous space for agents + diffusion on a grid)



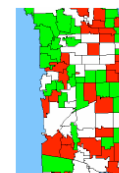
Euclidean
Space: 2D, 3D



Grid: von Neumann
neighborhood



Network



GIS: Geographic
Information
System

[North & Macal, 2006]

2

The space model

- Care: like for time, there are often implicit assumption for the space model
 - Is 2D sufficient?
 - How to model the borders of the space?
 - Absorbing, reflecting, static, periodic...
 - How to model infinite spaces?

“Diffusion is not a perfectly mixing process in low dimension because the diffusing molecule will return to its initial position with probability 1, whereas, for $d > 2$, there is a significant probability that the diffusing molecule will never return to its origin.”

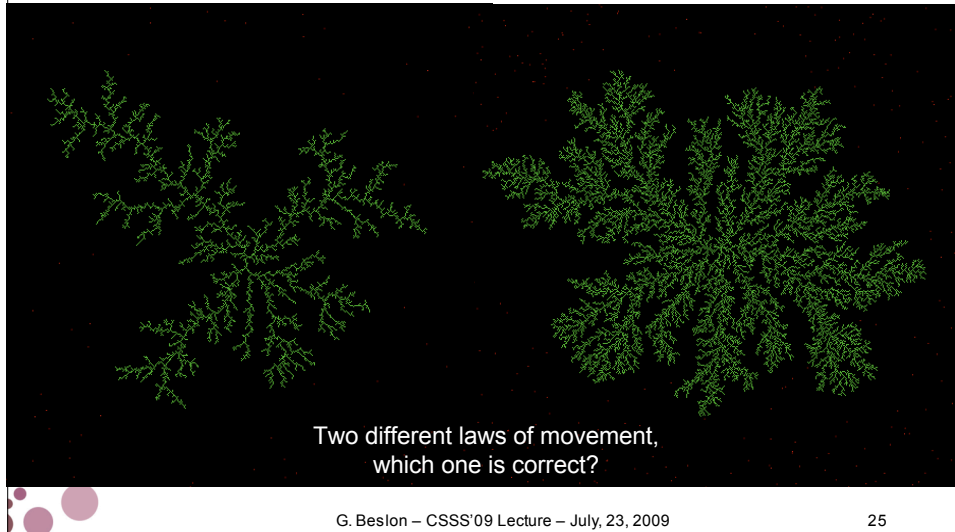
[berry, 2002]

Movement

- Agents will often move in the space
 - The laws of movement are generally supposed simple
 - Very often they are not!
 - Care not to reuse implicitly macroscopic laws of motion into a microscopic world (e.g., planets and molecules)
 - Sometimes the laws of motion explains the “emergent” results by themselves!
- E.G., DLA
 - Agents explore differently their vicinity depending on the laws of motion!

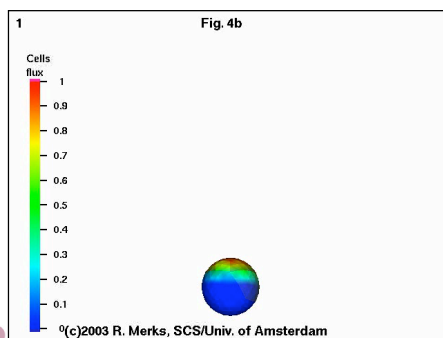


Fractals structures created by DLA

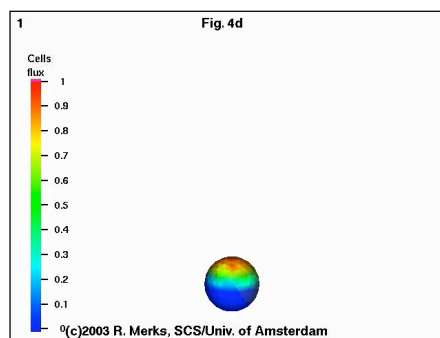


Law of motion matters!

- Coral morphogenesis [Merks et al., 2003]
 - Same agents
 - Different diffusion parameters leads to different shapes



Slow diffusion



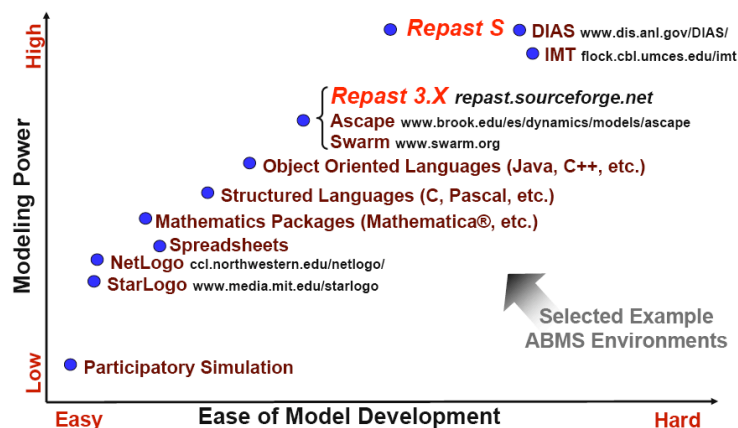
Fast diffusion

Implementation step

- Once you have designed your agents and their relations, how can you implement them and run the simulation?
 - Plate-forms, frameworks,
 - Programming from scratch (which language),
 - Reuse a previous model
- Take care: the implementation phase is NOT the most difficult nor the longer!
- Choose the methods/tools such that they
 - respect the modeling phase
 - will be efficient during the experimental phase
 - enable to follow “strictly” a scientific experimental methodology
- Then, you’ll probably have to program “a little”...

Implementation

- You will often find figures like:



The pro&cons of visualization

- Complex systems are based on subjective judgment
 - We need a visual feedback!
- We often have no mean to decide what is correct and what is not
 - We need a visual feedback!
- We have to care natural interpretations
 - Care visual feedback! ("I like it!")
- We have to repeat the experiments
 - Visual feedback are often slow!
- We have to repeat experiments
 - Visual feedback cannot be aggregated
- Conclusion
 - Care to visualize easily and to emphasize what is important
 - Care not to focus only on visualization: data output are important

Experiments

- Agent-Based Models often have MANY parameters
 - Most of them are often implicit ...
 - E.g., in my own model (Aevol) : 53 parameters!
- Agent-Based Models are generally slow
 - Need lots of computational resources
- It is NOT possible to test all parameters
 - Again, no hint! (except your own knowledge and experiments)
- Don't explore randomly the parameter space
 - Use "good practices" of experimental science
 - Actually ABM is an experimental approach (digital experiments)
 - Having a laboratory notebook is a VERY good practice!
 - Log all your experiments ; finish all your experiments
- Making the model is often less difficult than running the model...
 - Plan resources and time from the beginning of your project

ABM validation

- Verification: The program is doing what you want it to do
 - Very difficult problem! (+/- software engineering)
- Validation: The model produces the “correct” behavior
 - Impossible problem: A model is never “valid”

“Essentially, all models are wrong, but some are useful.”
[G. Box]

- Actually it depends on what you want to do with the model!
 - Predictive models can be tested (but never proved!)
 - Scientific models generally cannot

The meta-life-cycle of ABM

- Actually, ABM are not so difficult to build!
- The difficulty is (again) to produce knowledge with them!
- Meta-life cycle of ABM
 - Identify a good question
 - Build different simple models and play with them to identify what matters or not
 - Build YOUR model and make it stable
 - Make experiments with the model (experimental method helps!)
 - Analyze the results (statistical skill helps!)
 - Hopefully, acquire new knowledge (model the model)
 - Communicate, confront, publish
 - **FORGET YOUR MODEL**

Forget your model?

- Two reasons:
- The model is not the knowledge

"It could be argued that a criterion to determine good models is that they are no longer needed afterwards; The decisive thing with modeling is not the model per se, but what the model and working with the model does to our mind."

[V. Grimm, 1999]
- Remember that a model depends on a question...
 - If you change the question you MUST change the model
 - Of course, you can reuse some pieces of software but be careful on implicit choice
 - The software is not the model
 - Take care not to jump steps in the meta-life-cycle!

Applications

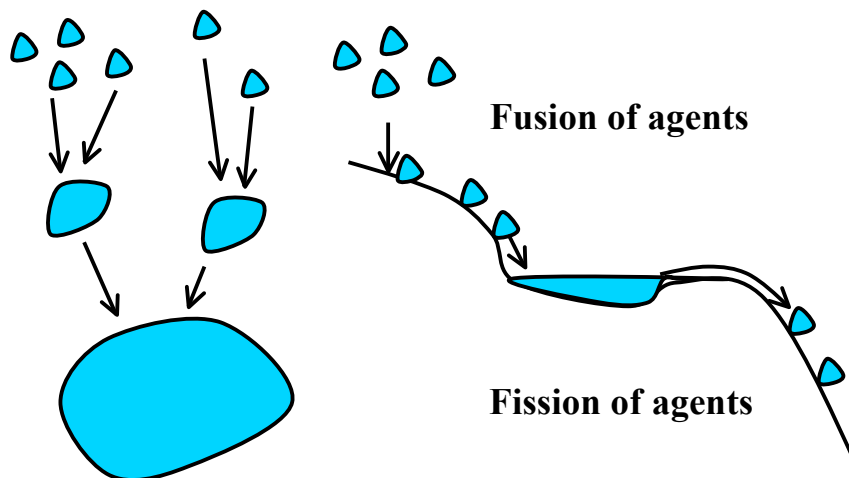
Table 1: Agent-based Modeling Applications

[North & Macal, 2006]	Business and Organizations	Society and Culture	+ evolution
	• Manufacturing	• Ancient civilizations	+ hydrology
	• Consumer markets	• Civil disobedience	+ membrane models
	• Supply chains	Terrorism	+ soil models
	• Insurance	• Social determinants	+ agriculture
Economics	• Artificial financial markets	• Organizational networks	+ diffusion of innovation
• Trade networks			+ ...
Infrastructure	• Electric power markets	Military	
• Hydrogen economy		• Command & control	
• Transportation		• Force-on-force	
Crowds	• Human movement	Biology	
• Evacuation modeling		• Ecology	
		• Animal group behavior	
		• Cell behavior	
		• Sub cellular molecular behavior	

Note that businessmen are not as "narrow-minded" as scientists ;)

No need of "proofs", just need to sell!

Grand challenge of ABM



G. Beslon – CSSS'09 Lecture – July, 23, 2009

35

When/why using ABM?

- [Grimm, 1999]
 - Pragmatic motivation: ABM can model phenomenon impossible to model with other approaches (“another tool in the modelers toolbox”)
 - Paradigmatic motivation: State variables modeling gives a false vision of reality since individuality, discreteness, locality or space matter
- Hum, not clear ... real motivations are more basic
 - Easy to construct, manipulate and extent (easy to change/add/remove parameters, rules,...) ... too easy?
 - Can model unknown phenomenon (if you have knowledge at the lower level)
 - ABM use a domain-based ontology (they are good interfaces between disciplines) easy to describe and to explain ... too easy?
 - “Looks like” (pleasant models) ... too pleasant?

G. Beslon – CSSS'09 Lecture – July, 23, 2009

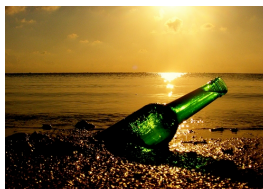
36

Why/when using ABM?

- Very often, it is claimed that ABM must be used when analytical models fails but
 - Analytical models have a long history in ~every scientific domain (are you sure they fail?)
 - Can we (computer scientists) really know when analytical models can or cannot be used
- In practice, always try to use ABM in parallel with analytical models...
 - ABM can be use before analytical model (to propose hypothesis)
 - ABM can be used after analytical model (to validate hypothesis)

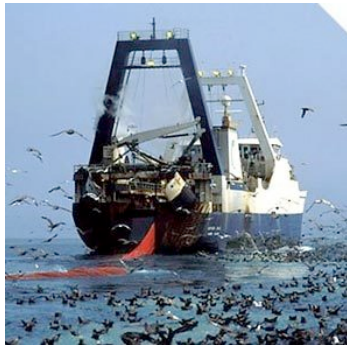
Why/when using ABM?

- ABM is drinking the sea with a teaspoon
 - Each run is easy, each run helps but you'll need a long time!
- Analytical models is drinking bears without a bottle opener
 - Each run is difficult but when you succeed you can drink the whole bottle at once!
- Note that there is much more diversity in the sea than in a bear bottle!



Another metaphor?

ABM vs. Analytical models



G. Beslon – CESS'09 Lecture – July, 23, 2009

39

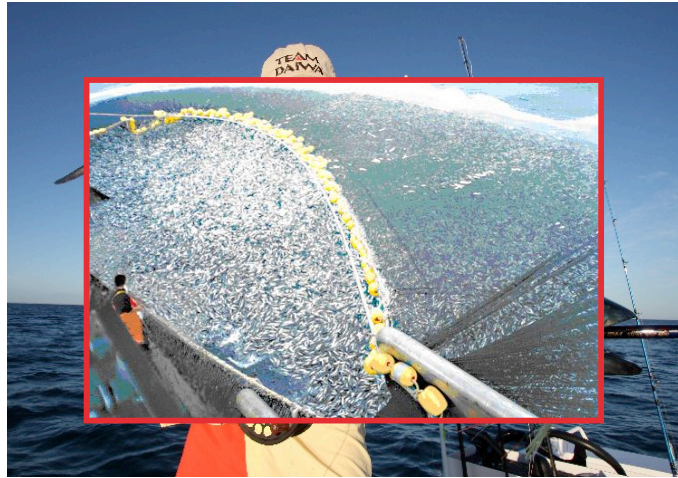
The BIG risk!



G. Beslon – CESS'09 Lecture – July, 23, 2009

40

The other BIG risk!



G. Beslon – CSSS'09 Lecture – July, 23, 2009

41

The future of ABM?



G. Beslon – CSSS'09 Lecture – July, 23, 2009

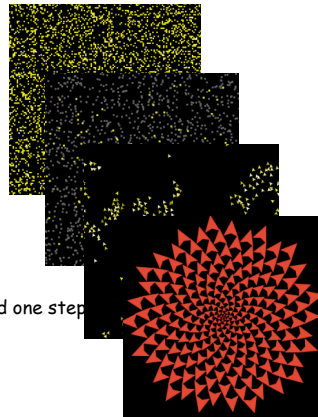
42

NetLogo

- Programmable simulation environment
 - Designed for Agent-Based Modeling
- Used mainly to simulate natural and social systems
 - Available for all systems (java),
 - Can be used in parallel
 - Simple “intuitive” programming language

```
to setup
  ca      ;; clear the screen
  crt 10  ;; make 10 new turtles
end

to go
  ask turtles
  [ fd 1      ;; all turtles move forward one step
    set heading random 360 ;; and turn randomly
  ]
end
```



G. Beslon – CSSS'09 Lecture – July, 23, 2009

43

What is NetLogo?

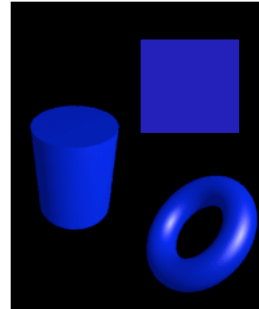
- Programmable environment for modeling phenomena based on collective behavior
 - Well suited for Agent-Based Modeling
 - Well suited for complex systems modeling (not all)
 - Can be used to model almost everything but may not be well suited for numerous applications (e.g., neural systems...)
- Made to be easy (comes from Logo and StarLogo)
 - Lots of high level instructions to manage the agents
 - Time and space are implicitly modeled (!)
- Made to be pedagogical
 - User-friendly interface
 - Large library of “curricular” models
- Made to be efficient? Not exactly but actually it is!

G. Beslon – CSSS'09 Lecture – July, 23, 2009

44

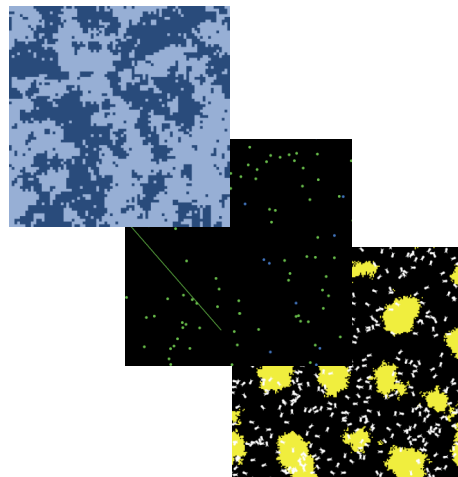
Basic Concepts

- 2D world peopled with agents
 - Square world (finite, cylindric or toric)
- 2+1 kinds of agents
 - **Patches:** Tiling elements that fill the world regularly (cannot move)
 - **Turtles:** Mobile agents moving on the “patches world”
 - **Observer:** Unique agent that perceives everything
- Only the first two ones are agents in (my) methodological meaning
 - The observer is a technological agent...

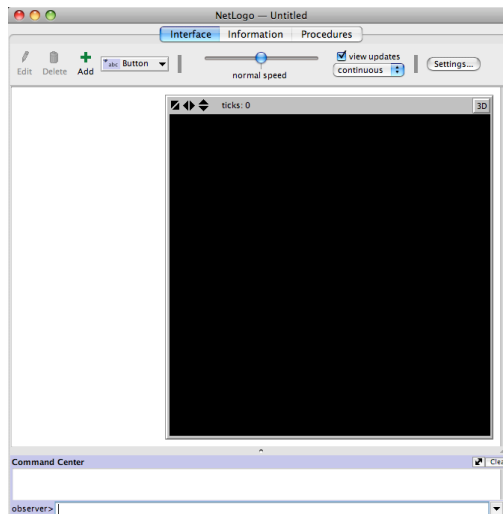


Basic Concepts

- Models in NetLogo may contain patches and/or turtles
 - Patch-models
 - Ising model
Patches: Spins
 - Turtle-models
 - Free-Gaz
Turtles: Molecules
 - Patches and turtle models
 - Termites
Turtles: Termites
Patches: Wood chips



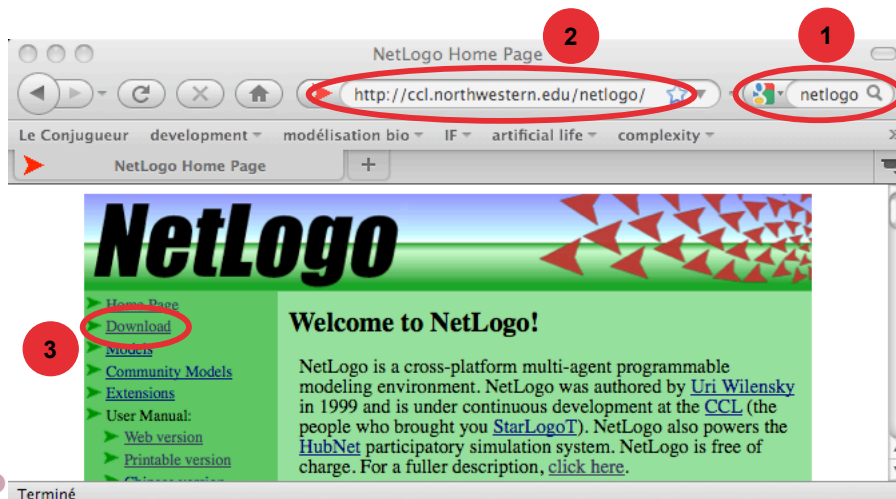
Basic Concepts: interface



G. Beslon – CSSS'09 Lecture – July, 23, 2009

47

Let's install it...



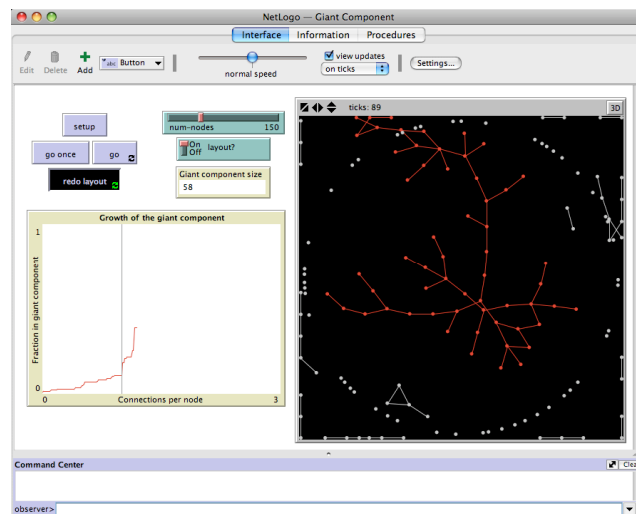
G. Beslon – CSSS'09 Lecture – July, 23, 2009

48

Basic Concepts

- Models

File
+
Model
Library
+
Networks
+
Giant
Component



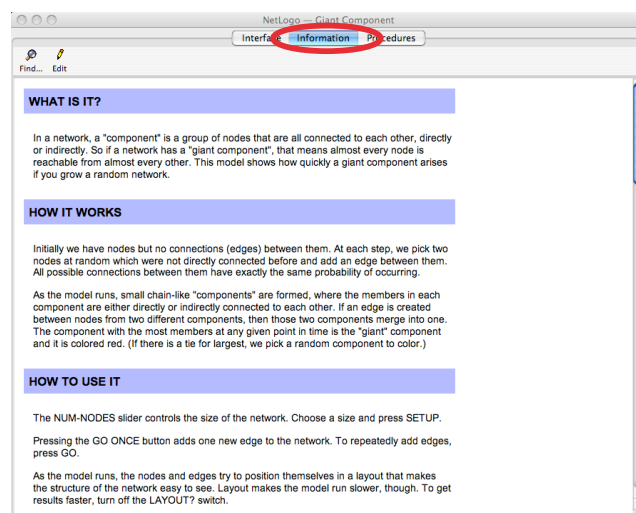
G. Beslon – CSSS'09 Lecture – July, 23, 2009

49

Basic Concepts

- Models

File
+
Model
Library
+
Networks
+
Giant
Component



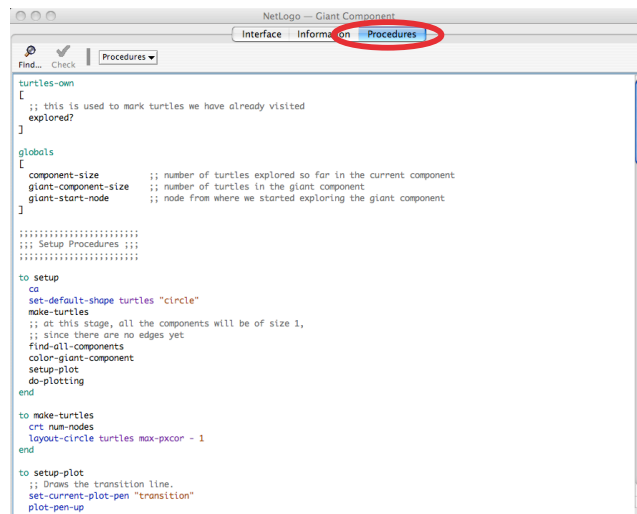
G. Beslon – CSSS'09 Lecture – July, 23, 2009

50

Basic Concepts

- Models

File
+
Model
Library
+
Networks
+
Giant
Component



```
turtles-own
[
  ;; this is used to mark turtles we have already
  explored?
]

globals
[
  component-size      ;; number of turtles explored so far in the current component
  giant-component-size ;; number of turtles in the giant component
  giant-start-node    ;; node from where we started exploring the giant component
]

;; Setup Procedures ;;
to setup
  ca
  set-default-shape turtles "circle"
  make-turtles
  ;; at this stage, all the components will be of size 1,
  ;; since there are no edges yet
  find-all-components
  color-giant-component
  setup-plot
  do-plotting
end

to make-turtles
  crt num-nodes
  layout-circle turtles max-pcor - 1
end

to setup-plot
  ;; Draw the transition line.
  set-current-plot-pen "transition"
  plot-pen-up
```

G. Beslon – CSSS'09 Lecture – July, 23, 2009

51

Command center

- The command center can be used directly to control the agents

- Try e.g. the following command

```
ca
setup
go
setup
ask turtles [ forward 1 ] ;; repeat this command
ask turtles [lt 90] ;; you can also try [rt 90]
ask turtles [ forward 1 ] ;; repeat this command
ask turtles with [color = red][set color green]
ask patches [set pcolor blue]
...
```

comments

- You have the basis!

- Now create your own project (FILES + NEW)

G. Beslon – CSSS'09 Lecture – July, 23, 2009

52

Command center

- Other commands to try:
 - Try the following command

```
crt 100  
ask turtles [forward 1] ; repeat this command
```
 - What is the structure of the universe?
 - You can change it (button “settings”) ... Try it!
 - Now try

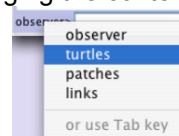
```
ca  
crt 100  
ask turtles [forward 10] ; repeat this command  
ask turtles [set heading 0]  
ask turtles [forward 1] ; repeat this command
```
 - OK? Try the same commands with “heading random 360”

Command center

- Other commands you can try

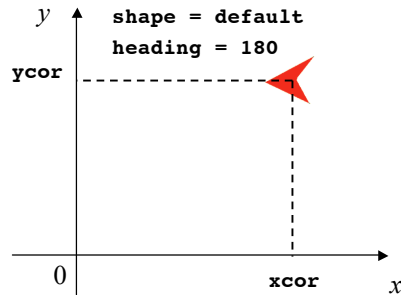
```
ask turtles [setxy random-xcor random-ycor]  
ask turtles [pen-down]  
ask turtles [forward 1 set heading random 360] ; repeat  
ask turtles [pen-up]  
ask turtles with [who < 25] [set color red]  
ask turtles with [who >= 25] [set color green]  
ask turtles with [color = red] [pen-down repeat 1000 [forward  
1 heading random 360]]
```
- Note that you can write multiple commands on a same line or in the same block ([.])
- You can “talk” directly to the turtle by changing the context of the command center

```
pen-down  
forward 1 set heading random 360
```



Agents: Turtles

```
who = 17
color = red
shape = default
heading = 180
```



- `forward #` ;; move forward by # steps
- `back #` ;; move backward by # steps
- `right #` ;; turns right by # degrees
- `left #` ;; turns left by # degrees
- `die`
- ...

“attributes”

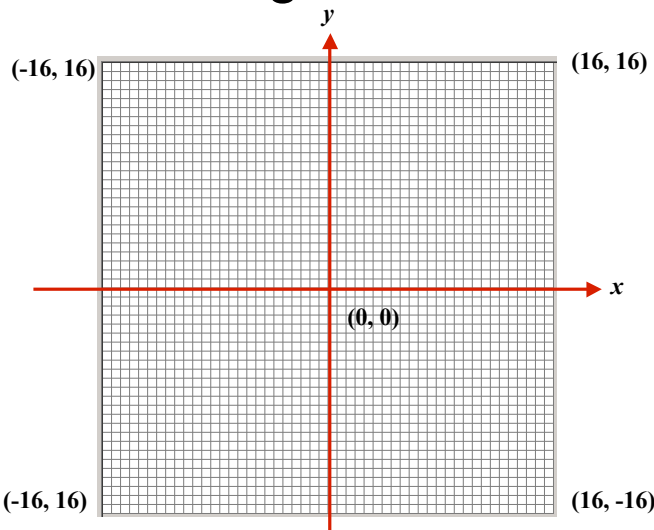
Turtles coordinates
(`xcor`, `ycor`, `heading`)
are real

“perceptions”

G. Beslon – CSSS'09 Lecture – July, 23, 2009

55

Agents: Patches



Default size
 $1089 = 33 \times 33$

Patch attributes

- `pcolor`
- `pxcor`
- `pycor`

Patch attributes
are integers

G. Beslon – CSSS'09 Lecture – July, 23, 2009

56

Agent: Observer

- Disembodied “agent” that sees everything in the world

```
clear-all ;; reset everything
create-turtles # ;; create # turtles
ask turtles [turtle only commands]
ask patches [patch only commands]
```

- Examples:

```
ask turtles [ forward 1 ]
ask patches [ set pcolor red ]
ask turtle 4 [ rt 90 ]
ask patches with [pxcor = 0][set pcolor yellow]
```

- Some commands can be abbreviated (not a good idea)

```
ca ;; clear-all
crt ;; create-turtles
fd ;; forward
...
```

Programming

- Direct use of the command center is very rough
 - To build and use models, you need to program...
- In NetLogo you can program by writing “procedures”
 - A procedure is a list of command with a name
 - The name can be used to execute the whole list
 - You can call a procedure in another one
 - Some procedures can be called from the interface by creating “buttons”
- Lets try!

Programming

- In the “procedure” frame, just write:

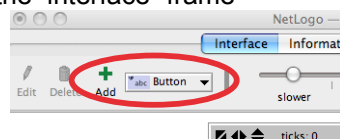

```
to setup          ;; setup is the name of the procedure
  ca              ;; clear the screen
  crt 10          ;; make 10 new turtles
end
```
- Then you can call your procedure from the command center


```
Setup
```
- Write a second procedure and call it


```
to go
  ask turtles
  [ fd 1 ;; all turtles move forward one step
    set heading random 360 ;; and turn randomly
  ]
end
```
- Your first NetLogo program is “finished”...

Calling the program

- But your “program” is really not easy to use!
 - It lacks an interface and a temporal loop
 - Both will be managed from the “interface” frame
- Lets create two buttons
 - Call the first one “setup”
 - Call the second one “go”
 - Care to check the “forever” box for the “go” button (it creates an implicit time loop)
- That's it ...
 - Run your program and test the different possibilities of the control panel (speed, settings, ...)
 - Try to modify your program...



More on procedures

- Procedures can receive “arguments” (inputs)

```
to draw-polygon [sides size]
  pen-down
  repeat sides
    [ forward size right (360 / sides)]
  end
```

- Call it from the command center

```
ask turtles [draw-polygon 5 5]
```

- Procedures can report values

```
to-report max-value [a b]
  if a > b [report a][report b]
end
```

- Test it by yourself...

– Hint: call it by “**print max-value 6 10**”

G. Beslon – CSSS'09 Lecture – July, 23, 2009

61

A first model

- Our program is not really a model...

– Why?

- Adding variables to the program

– Variables are containers for data. Each variable is identified by a name that must be declared before the variable is used

– Variables can be global or local (local variables can only be used in the procedure where they are defined)

```
global [mean-distance] ; global variable
let dist-to-center sqrt((xcor * xcor) + (ycor * ycor))
```

– Variables can be associated to turtles or patches

```
turtle-own [energy speed]
patch-own [local-diffusion-coefficient]
```

- Variables will enable us to measure some local or global information... The program becomes a model!

G. Beslon – CSSS'09 Lecture – July, 23, 2009

62

More on variables

- Agents have built-in variables that can be used directly without previous declaration
 - Turtles: **color**, **heading**, **xcor**, **ycor**, **who**, **shape**, **size**, **label**, **breed**
 - Patches: **pcolor**, **pxcor**, **pycor**, **plabel**
 - **label** and **plabel** are used to print information
 - **breed** is used to create agent classes
- Some examples of built-in variables manipulation

```
ask turtles [ set color red ]
ask patches [ set pcolor red ]
ask turtles [ set pcolor red ]
ask turtle 5 [ set color red ]
ask patch 2 3 [ set color red ]
```
- Lets finish our first model

Measuring information

- Add three global variables to your code (at the beginning of the code)

```
globals
[
  max-distance
  theoretical-mean-distance
  mean-distance
]
```
- Time is measured by the “tick counter”
 - Two commands

```
ticks ;; reports the current value of the counter
tick  ;; add one to the counter
```
- In the “go” procedure, compute the three variables
 - Hint: In a diffusion process, the mean distance is given by the square root of the time...

Printing the information

- Information can be printed in “monitors”
 - Use the monitor icon in the toolbar
 - Choose the reporter you want; Here:
Theoretical-mean-distance - mean-distance
 - Try it; What is the problem? Correct it...
- Monitors only give the immediate value
- You will probably want to draw graphics
 - Use the plot icon in the toolbar and fill the form
 - Here the plot will be called **distance**
 - Create as many pens as you want to draw different curves on the graphic (care to change pen colors)
 - Here, create pens **theoretical**, **simulated**, **max-distance**
 - But to draw the curves, you will need to program a little...

Plotting

- To plot the graph, create a new procedure and call it in the “go” one

```
to do-plotting
  set-current-plot "distance"
  set-current-plot-pen "theoretical"
  plot theoretical-mean-distance
  set-current-plot-pen "simulated"
  plot moy-distance
  set-current-plot-pen "max-distance"
  plot max-distance
end
```

Printing local information

- Labels can be used to show information on each agent
 - Create a turtle variable **distance-to-origin** and update it in the **go** procedure
 - Here, in the **go** procedure, add

```
ifelse show-distance?  
[ set label distance-to-origin ]  
[ set label "" ]
```
 - **show-distance?** Is a global variable that is created by a switch in the interface
 - Use the switch icon in the toolbar
 - In the Global variable section of the switch dialog box type:
show-distance?

Data input

- Using the interface, you can also enter data in your model; Here we will use a slider
 - Use the slider icon in the toolbar
 - Give the slider the name **nb-turtles**
 - Replace everywhere in the procedures the **10** value by **nb-turtles**
 - That's it!
- Your model is ready to be used ... test it
 - How is the error?
 - Add a slider to modify the size of the steps of the random walk
 - Does it change something?
- Now you have an object and a question ... you have a model!

Next steps ...

- In this tutorial, we only used turtles
 - Next step: program a “game of life” in NetLogo to learn to use patches
 - The general principles are exactly similar...
- Next slides will give some more information on NetLogo programming
 - Remember that NetLogo contains lots of dedicated instructions and variables...
 - Here we only saw part of them...
 - Possibilities are (almost?) infinite!

<http://ccl.northwestern.edu/netlogo/docs/>

Have Fun!

Agentsets

- Agentsets enable to access to subsets of agents

```
turtles with [ color = red ] ;; all red turtles
turtles-here with [color = red] ;; all red turtles on the current patch
patches with [ pxcor > 0 ] ;; all patches on the right of the space
turtles in-radius 3 ;; all turtles less than three patches away
patches at-points [ [1 0] [0 1] [-1 0] [0 -1] ] ;; von Neumann neighbors
Neighbors4 ;; von Neumann neighbors
```
- How to use agentsets

```
ask <agentset> [ ... ] ;; ask agents to do [...]
Show any? <agentset> ;; check if the agentset is not empty
Show count <agentset> ;; compute the cardinal of the agentset
```
- Examples

```
ask neighbors4 [ set pcolor red ]
;; turns the four neighboring patches red

ask max-one-of turtles [ sum assets ] [ die ]
;; remove the richest turtle (with the maximum “assets” value)
```

Breeds

- Breeds enable to create “natural classes” of agents

```
breed [wolves wolf]
breed [sheep a-sheep]
create-sheep 50
create-wolves 50
```
- When a breed is created, new functions become automatically available

```
create-<breed>, hatch-<breed>, sprout-<breed>, <breed>-here,
<breed>-at, <breed>-on, is-a-<breed>?
```
- Examples

```
ask turtles 5 [ if breed = sheep ... ] ;; Check the breed
ask turtle 5 [set breed sheep] ;; Change breed to sheep
```
- Note that **breed** is a variable

Programming structures

- Tests
 - One possibility

```
if xcor > 0 [ set color blue ]
```
 - Two alternatives

```
ifelse pxcor > 0
[ set pcolor blue ]
[ set pcolor red ]
```
- Loops
 - Repeat loop

```
repeat 36 [ fd 1 rt 10 ]
```
 - While loop

```
while [any? other turtles-here][ fd 1 ]
```
 - Foreach command

```
foreach [1.1 2.2 2.6][ show (word ? " -> " round ?) ]
```


Turtle Shape

- A turtle's shape is stored in its shape variable and can be set using the set command
- New shapes can be created, or imported from the shapes library or from other models, in the Shapes Editor



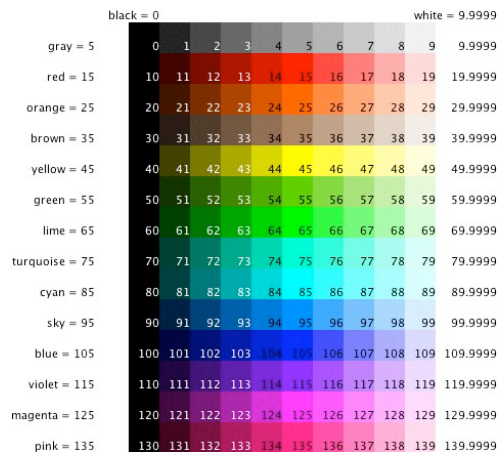
- Shapes have names to identify them : default, airplane, arrow, box, bug, butterfly, car, circle, circle 2, cow, cylinder, dot, face happy, face neutral, face sad, fish, flag, flower, ...

G. Beslon – CESS'09 Lecture – July, 23, 2009

73

Colors

- There are 140 colors
- Wrap-around is used when the color value is out of the range
- Try the following patch command
set pcolor pxcor + pycor



G. Beslon – CESS'09 Lecture – July, 23, 2009

74

Mathematics

- Most mathematical functions are supported
- Two number types
 - 64 bits integers: range from -2147483648 to 2147483647 (-2^{31} to $2^{31}-1$)
 - 64 bits floating numbers: $1.014534154524532 \times 10^{-8}$ (about 16-digit accuracy)

- Different pseudo-random functions are available

```
random 10 ;; returns an integer between 0 and 10
random-float 10.0 ;; returns a float between 0 and 10.0
random-seed 10 ;; initializes the pseudo-random generator
```

- You can also use
`random-normal`, `random-poisson`, `random-exponential`, `random-gamma`

Storing data (I): Outputs

- The simple way: the “output” interface
 - Use the “output” option in the toolbar
 - An “output” is a place where you can write data and store them at once after the end of the simulation (right-click + export)

- Output primitives are very simple

```
output-print ;; basic command sufficient in most situations.
output-show ;; print a value and the name of the agent.
output-type ;; lets you print several things on the same line.
export-output ;; store the content of the output in a file.
clear-output ;; clear ;)
```

- Outputs are quite basic devices; you may want more powerful ones
 - Use file I/O

Storing data (II): File I/O

- File I/O commands enable you to open, close and write files
 - When working with a file, you need to open it before
`file-open <file-name>`
 - When you have finished remember to close it
`file-close <file-name>`
 - I/O commands include:
`file-print`, `file-show`, `file-type`, `file-write`, `file-delete`,
`file-read`, `file-read-line`, `file-read-characters`, `file-at-end?`
 - If several files are opened, you must specify which one you want to use by calling “`file-open`” again

Storing data (III): Movies

- You can capture a QuickTime movie of a NetLogo simulation
 - Four simple primitives; Very easy to use!
`movie-start <file-name.mov> ;; starts a new movie`
`movie-grab-view ;; grab the environment window`
`movie-grab-interface ;; grab the whole interface window`
`movie-close ;; ends the movie`
 - Example
`;; export a 30 frame movie of the view`
`setup`
`movie-start "out.mov"`
`repeat 30`
`[movie-grab-view go]`
`movie-close`
 - Of course, you can call all these primitives from the command center
 - Care the size of the files! NetLogo produced uncompressed videos!

Example

- 300 steps (300x15 frames) : 60.1 Mo!

Managing experiments

- NetLogo includes a “BehaviorSpace” tool which enables to manage experiments and to explore automatically the parameter spaces
 - Menu “Tools + BehaviorSpace”
 - BehaviorSpace collects data from multiple runs of a model
- When creating a new experiment NetLogo opens the experiment window which enables you to choose:
 - Parameter space, varying input variables, reporters...
 - Number of repetitions
 - Stop conditions, to go conditions, setup procedures...
- You can specify lists of values or extreme values and step

```
["max-pxcor" [11 1 14]] ;; go from 11 to 14 one by one (care
the brackets)
["random-seed" 343 564 812 23] ;; to test these 4 values
```
- All input variables can be specified
 - NetLogo also enables to specify some “internal” variables (environment size, seed of the random generator)