

NetLogo



Andrea Sterbini – sterbini@di.uniroma1.it

NetLogo and NetLogoWeb

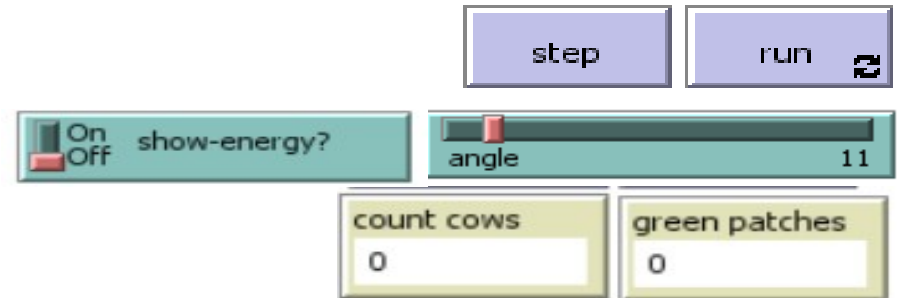
turtles + patches = movable agent simulations

Full Logo:

- procedures + reporters (functions)
- lists and filters
- anonymous functions (parametric code blocks)
- new agent types with added properties (OOP without inheritance?)

Easy GUI construction:

- Buttons to call functions/procedures
- Sliders to change global variables
- Labelled boxes to show values
- Plot graphs of values during simulation
- 2 NetLogo versions: 2D and 3D canvas showing turtles, patches and edges



3 type of Agents (+ custom agents)

Turtles: movable entities (with respect to the 2D or 3D canvas)

Patches: the canvas is covered by a grid of unmovable squares (cubes)
- e.g. the grass of a field or a pixellated volume (2 or 3 dim. MATRIX concept)

Edges: directed/undirected links between two Turtles in 3D space

Other “custom animal groups” can be easily defined:
- breed [singular plural]

Separate breeds can have separate sets of properties:
- cows-own [energy]

The Turtles' set contains all other breeds (like “object” in Java)

An agent can change its breed type! (set breed 'breedname')

Programming style

Single-threaded (BUT the order of set elements is random)

Procedural (“to” for procedures)

Functional (“to-report” for functions)

Data types:

- lists (immutable, untyped)
- arrays (mutable, untyped)
- list-based operations (map / filter / collect / ask / ...)
- anonymous functions (code blocks passed as arguments)

A LOT of built-in commands are functions / filters

THUS the language is very very readable

NetLogo and other Logos

Small syntactic differences

most Logos

```
to square :x
  output :x * :x
end
```

NetLogo

```
to-report square [x]
  report x * x
end
```

to-report
report
[args]

instead of
instead of
instead of

to
output
:arg

some precedence differences

Demo 1: Random walk

- start with N randomly placed turtles
- move each turtle
by 1 step
by changing slightly its heading

NO main loop, just use a repeating button
with a single simulation “step” procedure

Globals: (interactive slider) max turn angle,
of turtles

to **step**

```
ask turtles [  
  set heading (heading + (random (2 * angle)) - angle)  
  forward 1  
]  
tick          ;; update tick count  
end
```

This is an anonymous block executed IN EACH TURTLE's CONTEXT!



Demo 2: a flock of birds

Here each turtle should:

- turn towards her nearest neighbour
- and move

Globals:

- # of turtles,
- attraction force towards the nearest one

```
to-report closest-turtle
```

```
  report min-one-of (other turtles) [  
    distance myself ]
```

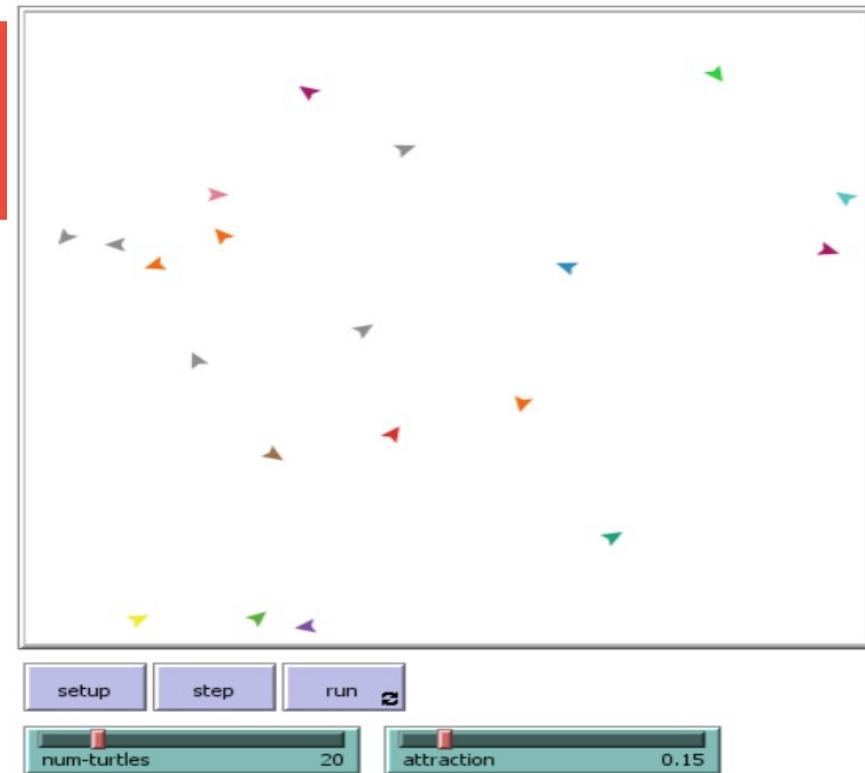
```
end
```

```
to turn-towards [somebody]
```

```
  let difference subtract-headings (towards somebody) heading
```

```
  set heading (heading + (attraction * difference))
```

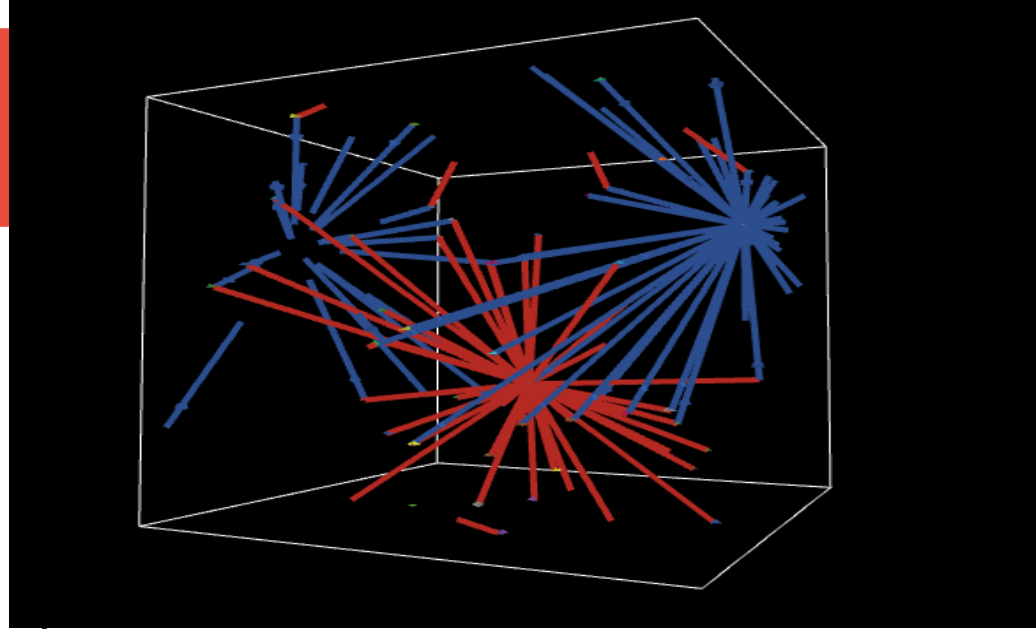
```
end
```



Demo 3: 3D links

- N turtles in random 3D position
- 2 random turtles are connected to all other turtles with **directed** and **undirected** edges
- NOTICE: the world is a TORUS!

```
undirected-link-breed [ ulinks ulink ]
directed-link-breed   [ dlinks dlink ]
to setup
  clear-all
  create-turtles N [ setxyz random-xcor random-ycor random-zcor ]
  ask turtle random N
    [ create-ulinks-with other turtles [ set color red ] ]
  ask turtle random N
    [ create-dlinks-to   other turtles [ set color blue ] ]
end
```



Demo 4: cows on grass

Cows:

- loose 1 energy per tick
- move at random
- eat grass gaining 10 energy
- if energy > 50 spawn

Grass:

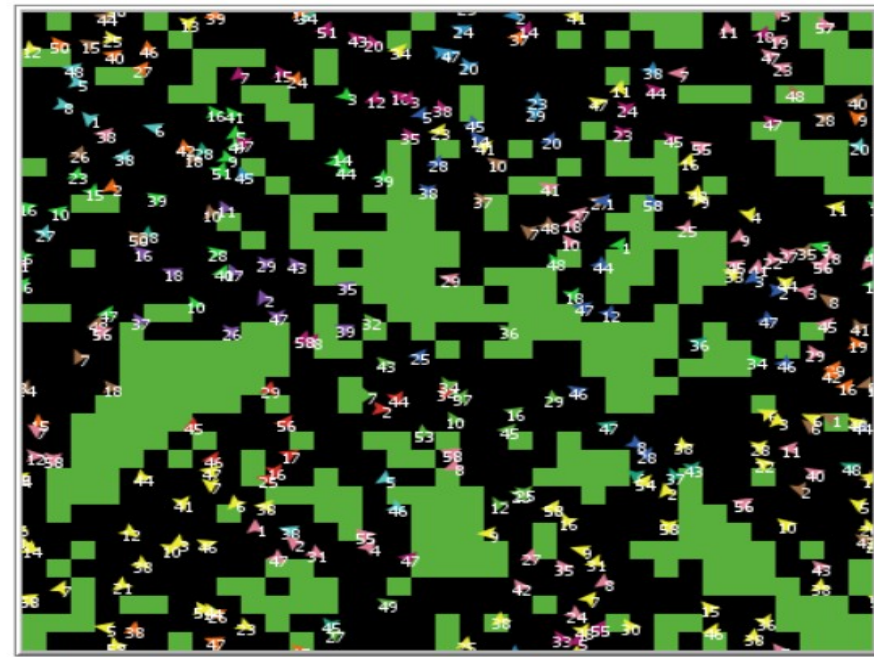
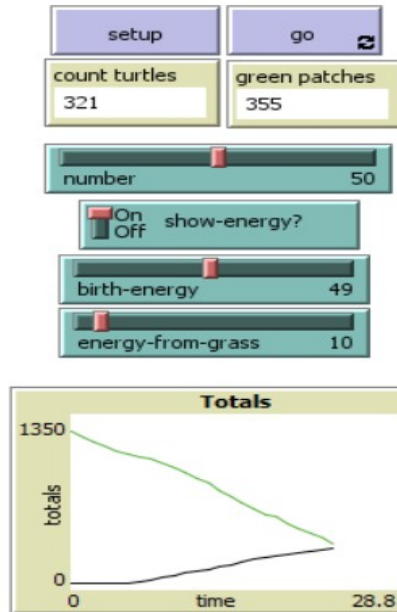
- new grass grows with 3% probability

Globals:

- show cow energy?, energy to give birth, energy from grass

Display:

- # of cows, # of grass patches



Demo 4: implementation ...

```
breed [ cows cow ]      ; define cows
cows-own [energy]        ; add attribute
... (setup removed)

to go                    ; single step
  if ticks >= 500 [ stop ]
  move-cows
  eat-grass
  check-death
  reproduce
  regrow-grass
  tick
end
```

```
to eat-grass            ; eating increases energy
  ask cows [
    if pcolor = green [
      set pcolor black
      set energy (energy +
                    energy-from-grass)
    ]
    ifelse show-energy?
      [ set label energy ]
      [ set label "" ]
  ]
end
```

... continue

```
to move-cows ; move all cows
  ask cows [
    right random 360 ; change
    direction
    forward 1 ; move
    set energy energy - 1 ; lose energy
  ]
end
```

```
to reproduce ; healthy cows reproduce
  ask cows [
    if energy > birth-energy [
      set energy energy - birth-energy
      hatch 1 [ set energy birth-energy ]
    ]
  ]
end
```

```
to check-death ; remove dead cows
  ask cows [
    if energy <= 0 [ die ]
  ]
end
```

```
to regrow-grass ; 3% of grass regrows
  ask patches [
    if random 100 < 3 [
      set pcolor green
    ]
  ]
end
```

Extensions!!!

Arduino

CSV

Continuous f. optimiz.

Modular models

Clustering

Cognitive Agents

GIS

Python

Webcam

GoGo boards

Database

Function roots

Linear programming

Freq. Distributions

Q-learning

Epidemiology

R

Isometric visualization

Profiler

Matrix math

Time series

Statistics

Fuzzy logic

Physics

Scala

Web

* Bug Hunters Camouflage HubNet - NetLogo

File Edit Tools Zoom Tabs Help

Interface Info Code

Edit Delete Add | normal speed | ☒ view updates | Settings...
ticks: 0

carrying-capacity 20 | setup | go

environment poppyfield.jpg | change background | clear background

bug-size 1.5 | flash

max-mutation-step 25 | make a generation

☐ On show-genotype? | offspring-distance 5

Top hunter | Top hunter's catches 0

Bugs Caught by All Hunters vs. Time

bugs 50 0 | days 0 100



Command Center

observer>

Client GUI

Simplified GUI edited in the main app. The main code must handle the standard messages.

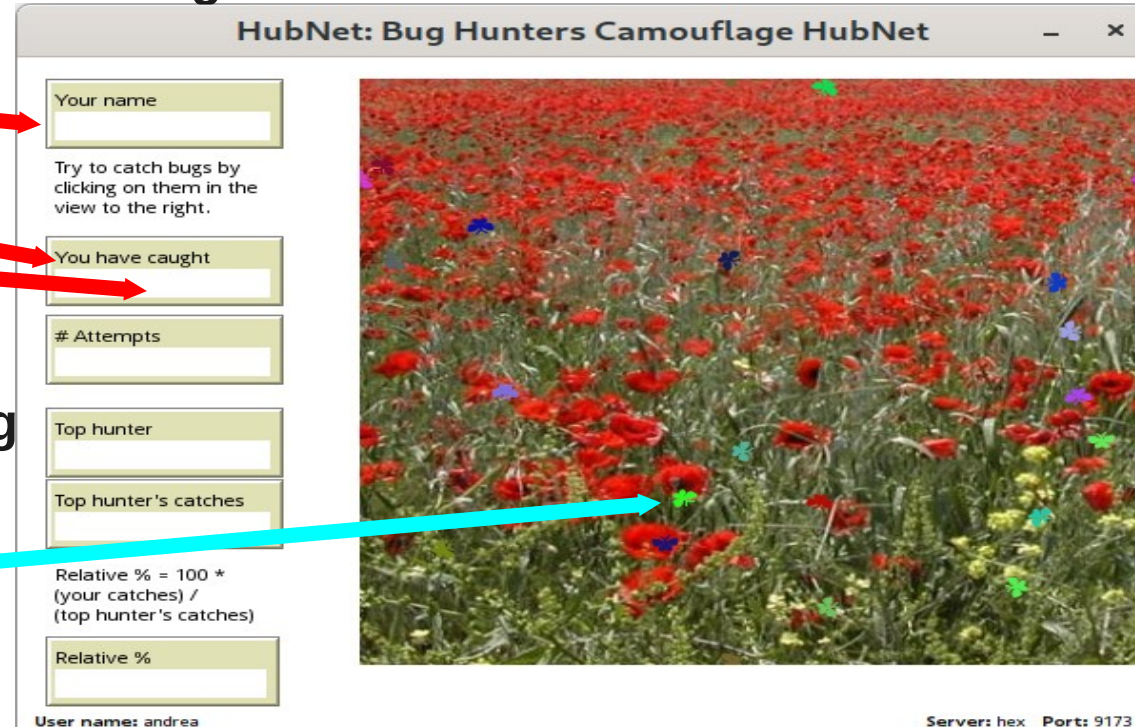
All widgets send/receive standardized messages

- client ID
- widget tag
- widget content

The View shows the canvas+turtles

Clicks of clients on views -> 2 messages

- mouse down / up
- mouse position



NetTango: block-based N

Web-based NetLogo
editor / simulator

Define blocks
containing
snippets of
code
(macro operations)

Kid program by using
your defined blocks



The screenshot displays the NetTango web-based NetLogo editor interface. On the left, a sidebar contains a 'powered by' logo and a 'food' variable definition. The main workspace shows a 'Go' block with a red border, containing an 'Each ant' loop. Inside the loop, there are two conditional blocks: 'If I am not carrying food' and 'If I am carrying food'. The 'If I am not carrying food' block contains 'Turn towards pheromone smell', 'If there is food here' (which includes 'Pick food', 'Turn around' with a value of 180, and 'Turn towards nest smell'), and 'If I am at the nest' (which includes 'Drop food' and 'Turn around' with a value of 180). The 'If I am carrying food' block contains 'If I am not at the nest' (which includes 'Turn towards nest smell' and 'Drop pheromone') and 'If I am at the nest' (which includes 'Drop food' and 'Turn around' with a value of 180). Below the conditional blocks are 'wobble' and 'Move forward' blocks. On the right, a 'Recompile' button is visible, and a list of blocks is shown, including 'Go', 'Each ant', and various conditional and action blocks. At the bottom, 'Update Block' and 'Discard Changes' buttons are present.

Display name: wiggle

NetLogo code for wiggle: left ({})

Note - extra info

Can blocks follow: Blocks can be followed by:

Tags (0) C

Parameters

Parameter: Display name: amount, Min

Properties (0)

Control Clauses

Block Styles

Go

Each ant

If I am not carrying food

Turn towards pheromone smell

If there is food here

Pick food

Turn around 180

Turn towards nest smell

If I am carrying food

If I am not at the nest

Turn towards nest smell

Drop pheromone

If I am at the nest

Drop food

Turn around 180

wobble

Move forward

Update Block Discard Changes

Other ideas

- Modeling Commons:** cooperatively shared repository of models
- Behavior Space:** hyper-parameters optimization over many model runs
- System Dynamics:** high-level modeling (instead than agent-based)
with evolution of global measures (e.g. #of sheeps vs # of
wolves)
- Mathematica Link:** call Mathematica from Netlogo

Demo

DEMO