NetLogo

Andrea Sterbini – sterbini@di.uniroma1.it

NetLogo:
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?)

Easy GUI construction with widgets that:
- call functions/procedures
- change global variables
- show values
- plot graphs of values during simulation

2 versions: 2D and 3D

3 type of Agents (+ custom agents)

Turtles: movable entities

Patches: the canvas is covered by a grid of unmovable squares
- e.g. the grass of a field

Edges: links between two Turtles

Other “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

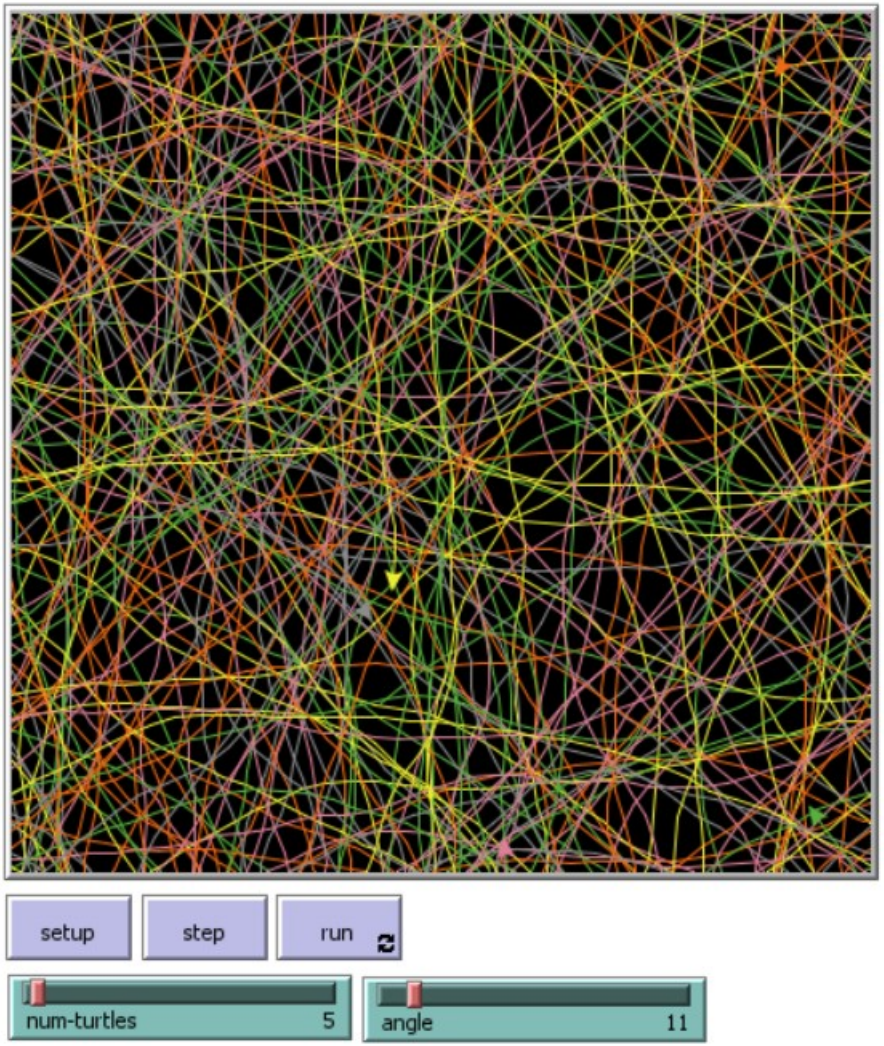
An agent can change its breed type (set breed ‘breedname’)

Demo 1: Brownian motion

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

Globals:

- max turn angle, # of turtles

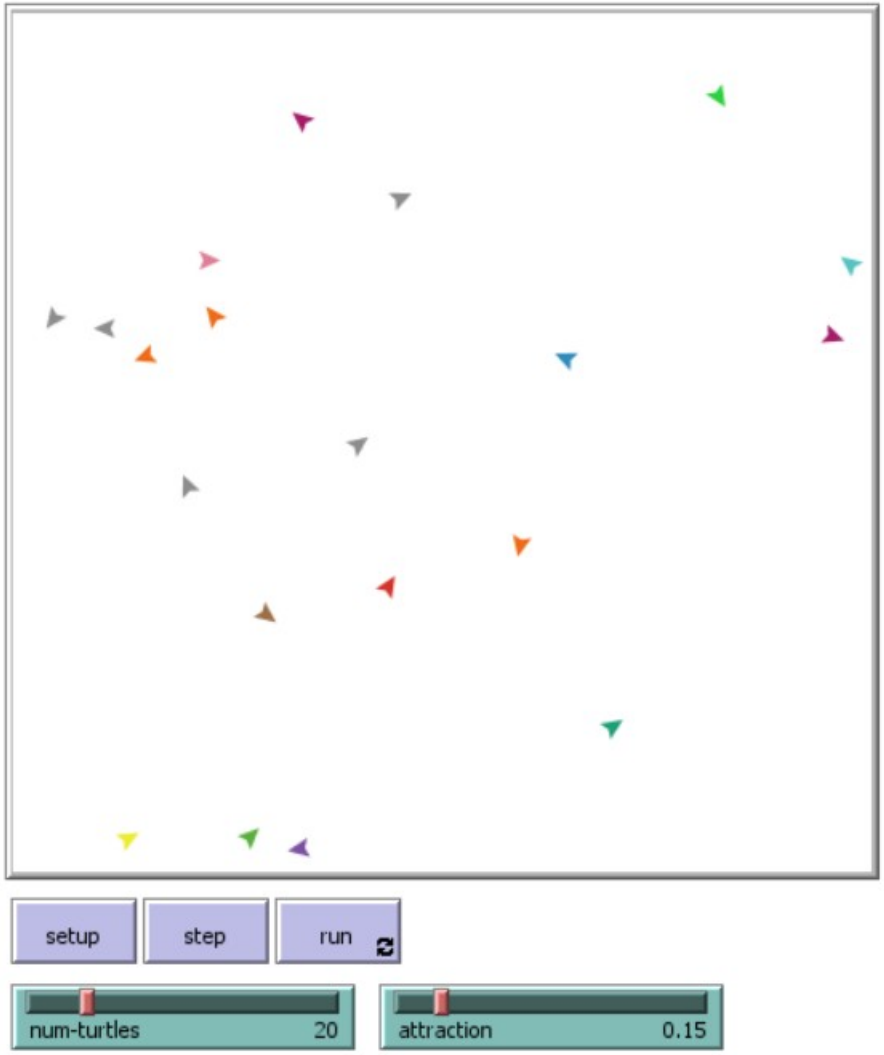



```
to step
  ask turtles [
    set heading (heading + (random (2 * angle)) - angle)
    forward 1
  ]
  tick
end
```

Demo 2: a flock of birds

Here each turtle should:
- turn towards her nearest neighbour
- and move

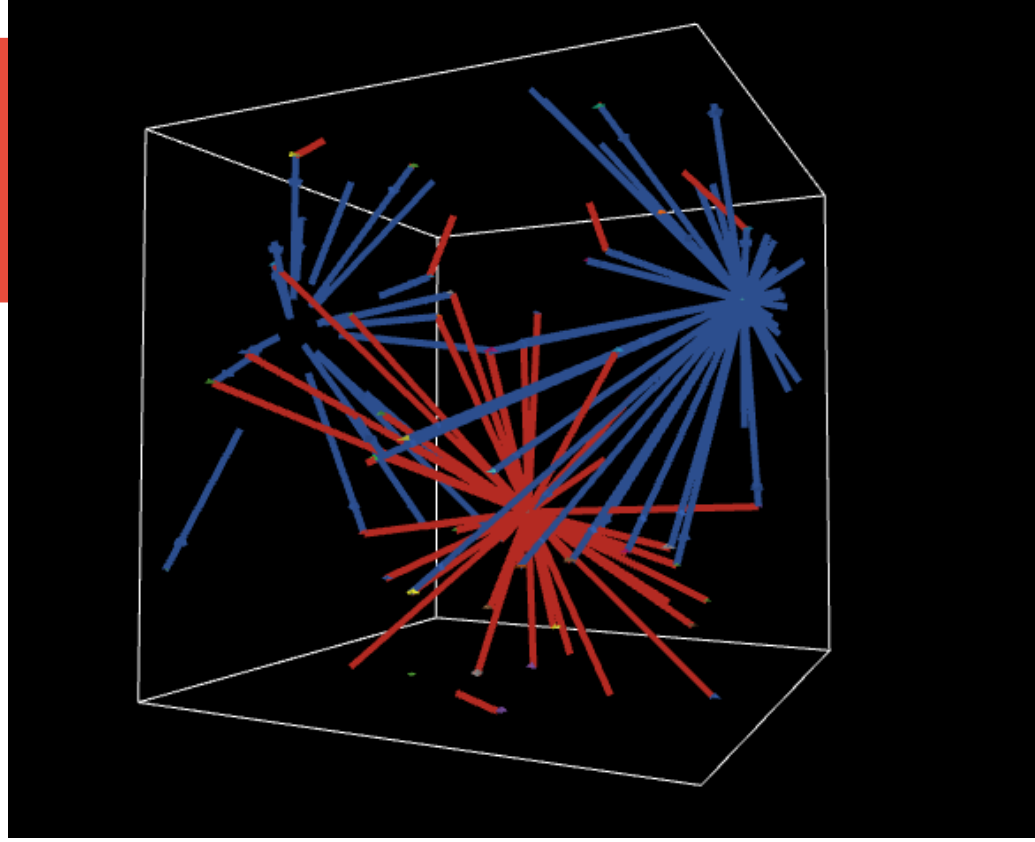
Globals:
- # of turtles, attraction towards nearest




```
to-report closest-turtle
  report min-one-of (other turtles) [
      distance myself ]
end

to turn-towards [somebody]
    let difference subtract-headings heading (towards somebody)
    set heading (heading + (attraction * difference)
end
```

Demo 3: 3D links



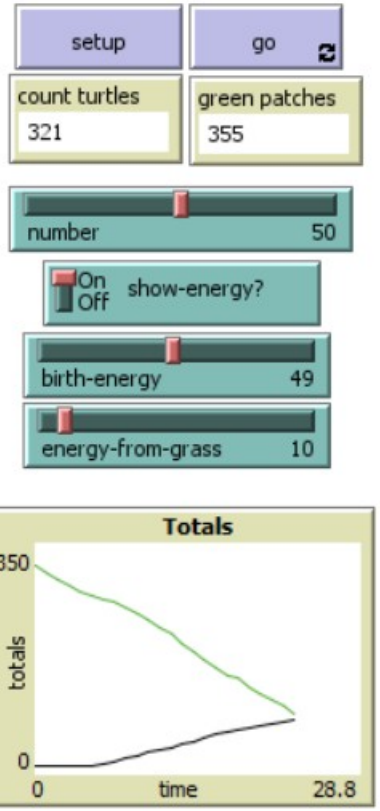
- N turtles in random 3D position

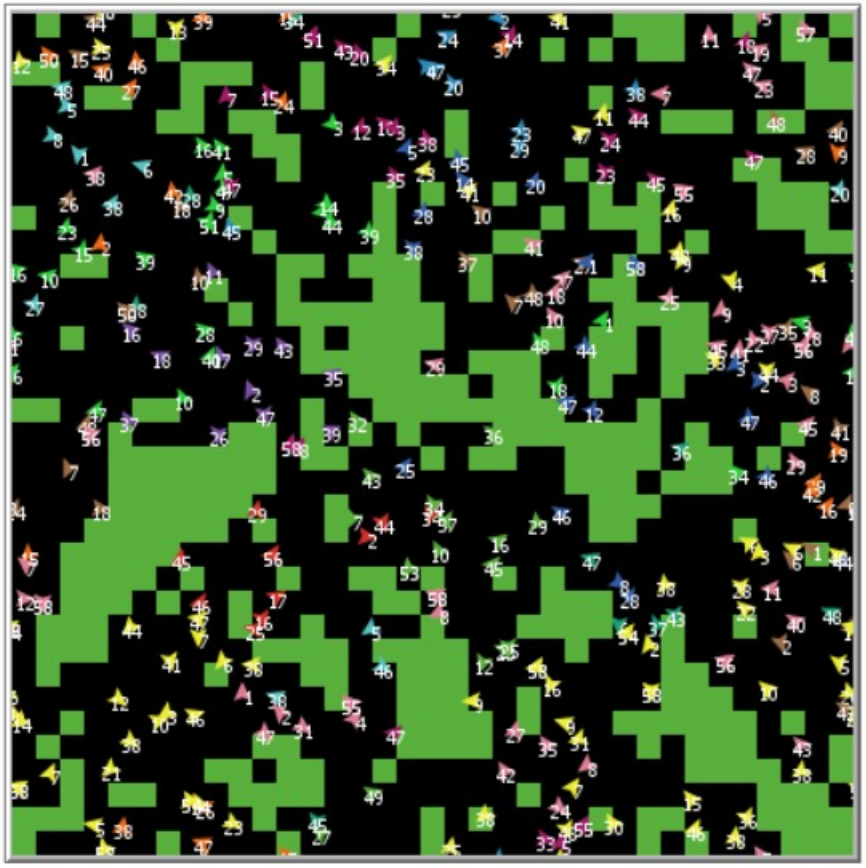
- 2 random turtles are connected to all other turtles with directed and undirected edges

- 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 e.
- 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 3: implementation ...

```
breed [ cows cow ]

cows-own [energy]

... (setup removed)

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

```
to eat-grass
  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
  ask cows [
    right random 360
    forward 1
    set energy energy - 1
  ]
end

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

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

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

Programming style

Single-threaded (but the order of set el. is random)

Procedural ("to" procedures)

Functional !!! ("to-report" functions)

Data types:

- lists (immutable)
- arrays (mutable)
- list-based operations (map/filter/...)
- anonymous functions (code blocks)

A LOT of built-in commands are functions/filters

THUS the language is very readable

Demo

DEMO