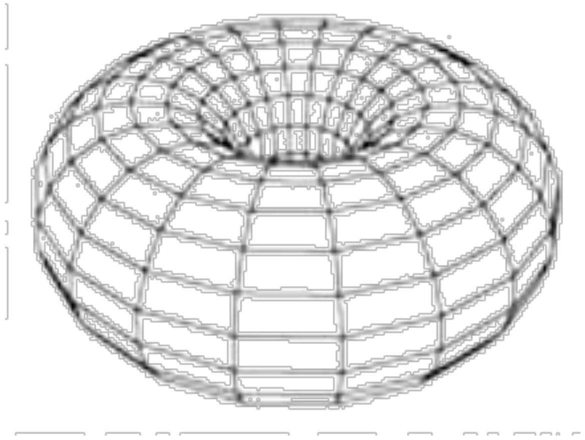
Costruire una nuova simulazione con NetLogo

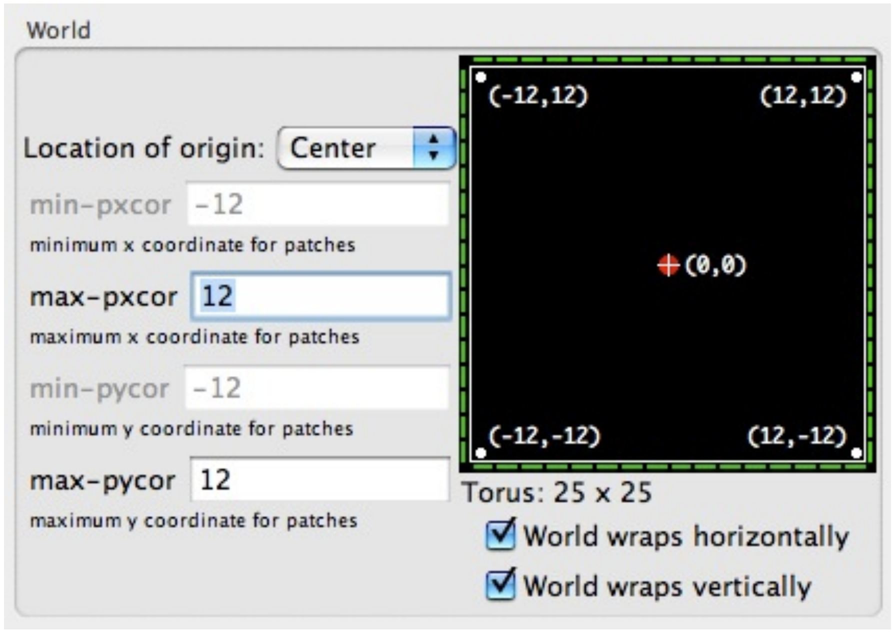
Adesso costruiamo assieme un nuovo modello dove due specie (breeds) di turtles, cani e gatti, si muoveranno all'interno del World con due modalità di comportamento diverse:

I cani si spostano provando a catturare uno dei gatti attorno a loro;

i gatti si muovono a caso;

open boundary conditions




World

Location of origin: Center

min-pxcor -12
minimum x coordinate for patches

max-pxcor 12
maximum x coordinate for patches

min-pycor -12
minimum y coordinate for patches

max-pycor 12
maximum y coordinate for patches

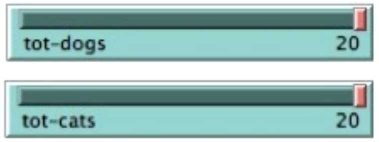
Torus: 25 x 25

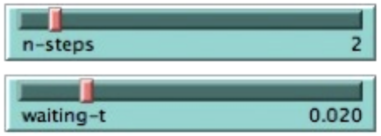
☑ World wraps horizontally

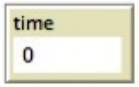
☑ World wraps vertically

3-Dogs and cats.nlogo

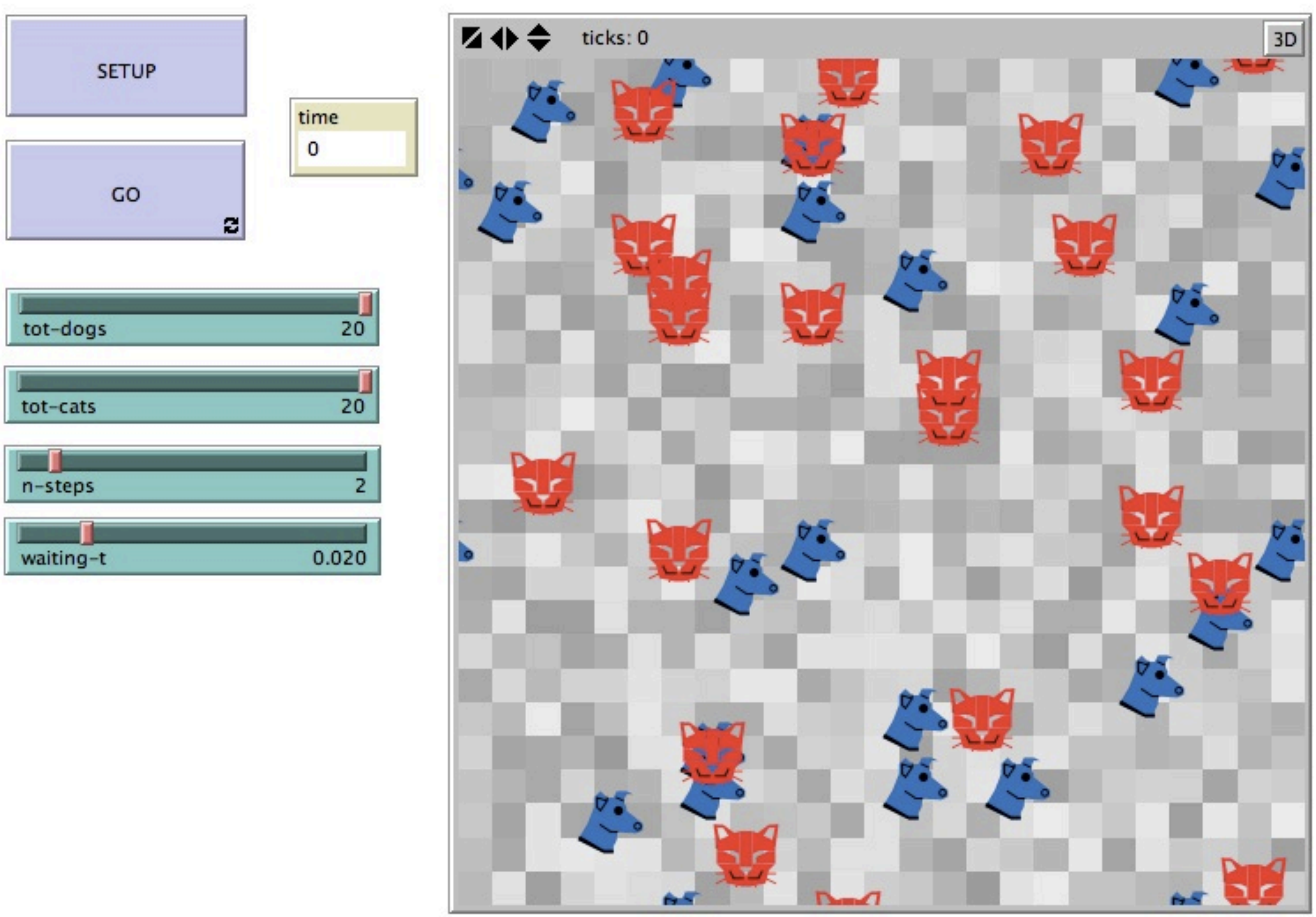
```
globals[ time ] ;; defines the global variable "time"

breed [ dogs ]  ;; defines the breed "dogs"
breed [ cats ]  ;; defines the breed "cats"

to setup
  ca   ;; clear all
  set time 0
  set-default-shape dogs "dog"  ;; set the shape of the breed 'dogs'
  set-default-shape cats "cat"  ;; set the shape of the breed 'cats'

  ask n-of tot-dogs patches  ;; selects a number 'tot-dogs' of patches at random
    [
      sprout-dogs 1 [ set color blue  set size 2 ] ;; creates a dog over each selected patch
    ]
  ask n-of tot-cats patches  ;; selects a number 'tot-cats' of patches at random
    [
      sprout-cats 1 [ set color red  set size 2 ] ;; creates a cat over each selected patch
    ]
  ask patches  ;; set the color of the patches in gray-scale
    [ set pcolor (5 + random-float 4) ]
end

to go ;; forever button
  ask dogs
  [ ;; each dog is asked to choose a cat at random within a radius of 10 patches, to move ahead
    ;; n steps towards the cat and to wait some time
    set heading towards target-dogs 10 fd n-steps wait waiting-t
  ]
  ask cats
  [ ;; each cat is asked to rotate on the right of a random quantity of degrees, to move ahead
    ;; n steps and to wait some time
    rt random 360 fd n-steps wait waiting-t
  ]
  set time (time + 1)
end

to-report target-dogs [radius]        ;; returns one of the cats situated within
 report one-of cats in-radius radius   ;; a circle of a given radius
end
```


tot-dogs 20
tot-cats 20


n-steps 2
waiting-t 0.020


time
0

3-Dogs and cats.nlogo



time
0

tot-dogs 20

tot-cats 20

n-steps 2

waiting-t 0.020

A questo punto potremmo sentire l'esigenza di complicare il comportamento di cani e gatti, per esempio:

chiedendo ai cani di "mangiare" i tutti i gatti che, ad un certo passo temporale, si trovino sulla loro stessa patch;

chiedendo ai gatti di cercare di evitare di essere mangiati dai cani;

Ma per realizzare queste nuove modalità comportamentali è necessario introdurre nuovi importanti elementi di programmazione del linguaggio di NetLogo, ovvero le cosiddette strutture per il controllo di flusso, che permettono alle turtles di compiere delle scelte o ripetere iterativamente certe azioni...

NetLogo

Strutture per il controllo di flusso

if *condition* [*commands*]

...dove *condition* è una espressione booleana

ifelse *condition* [*commands1*] [*commands2*]

ifelse-value *condition* [*value1*] [*value2*]

loop [*commands*]

...bisogna usare il comando “stop” per uscire dal “loop”

repeat *number* [*commands*]

while [*conditions*] [*commands*]

foreach *list* [*commands*] →

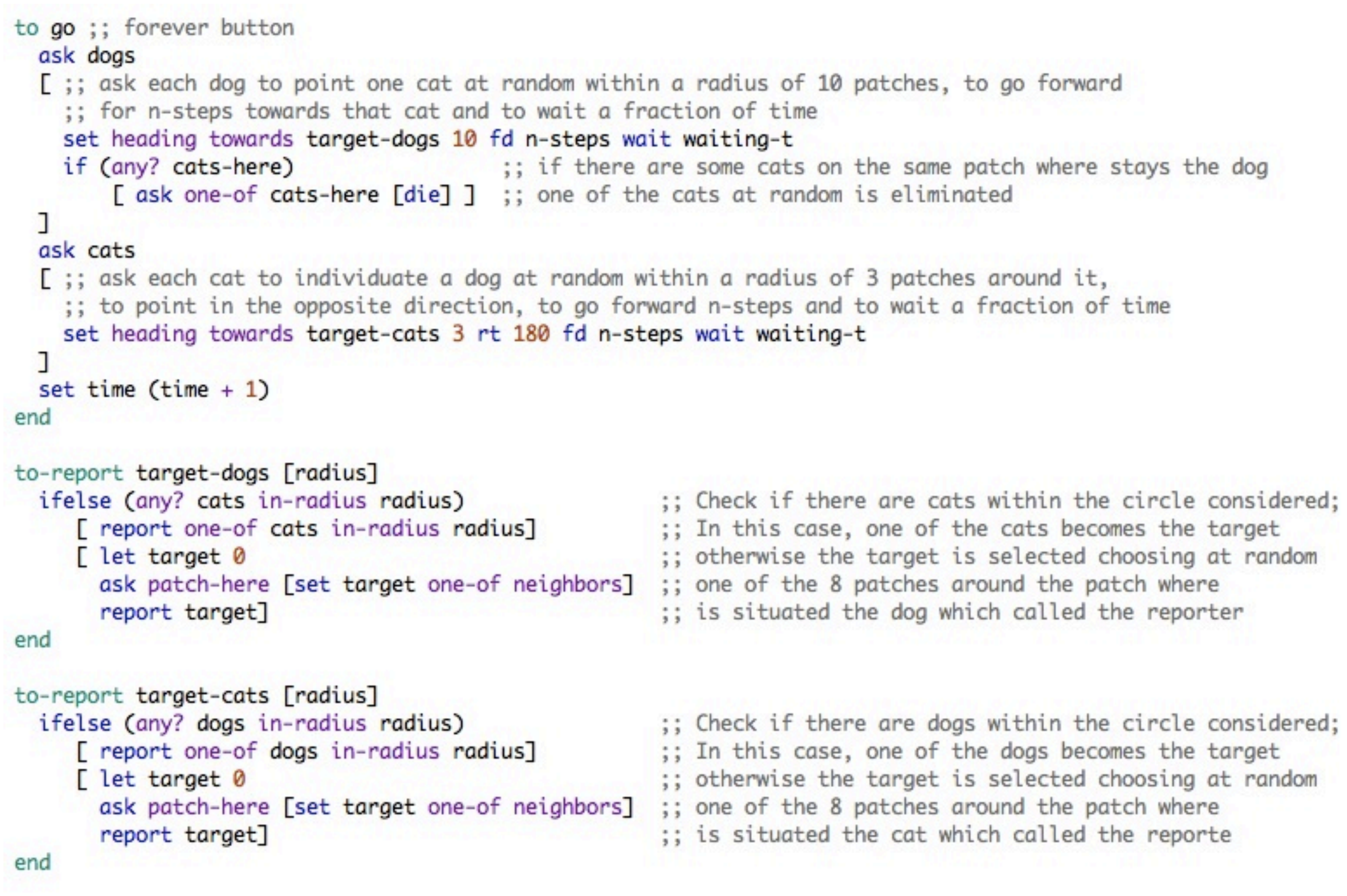
Il comando “foreach” permette all’observer o a un certo agente di scorrere gli elementi di una “lista”, cioè di un array di variabili (numeri, stringhe, agenti, agentsets o anche altre liste), definito con una istruzione del tipo:

set mylist [2 10 3.14 "Bob"]

Con il comando: **set** mylist **lput** 47 mylist
la lista “mylist” diventa: [2 10 3.14 "Bob” 47]

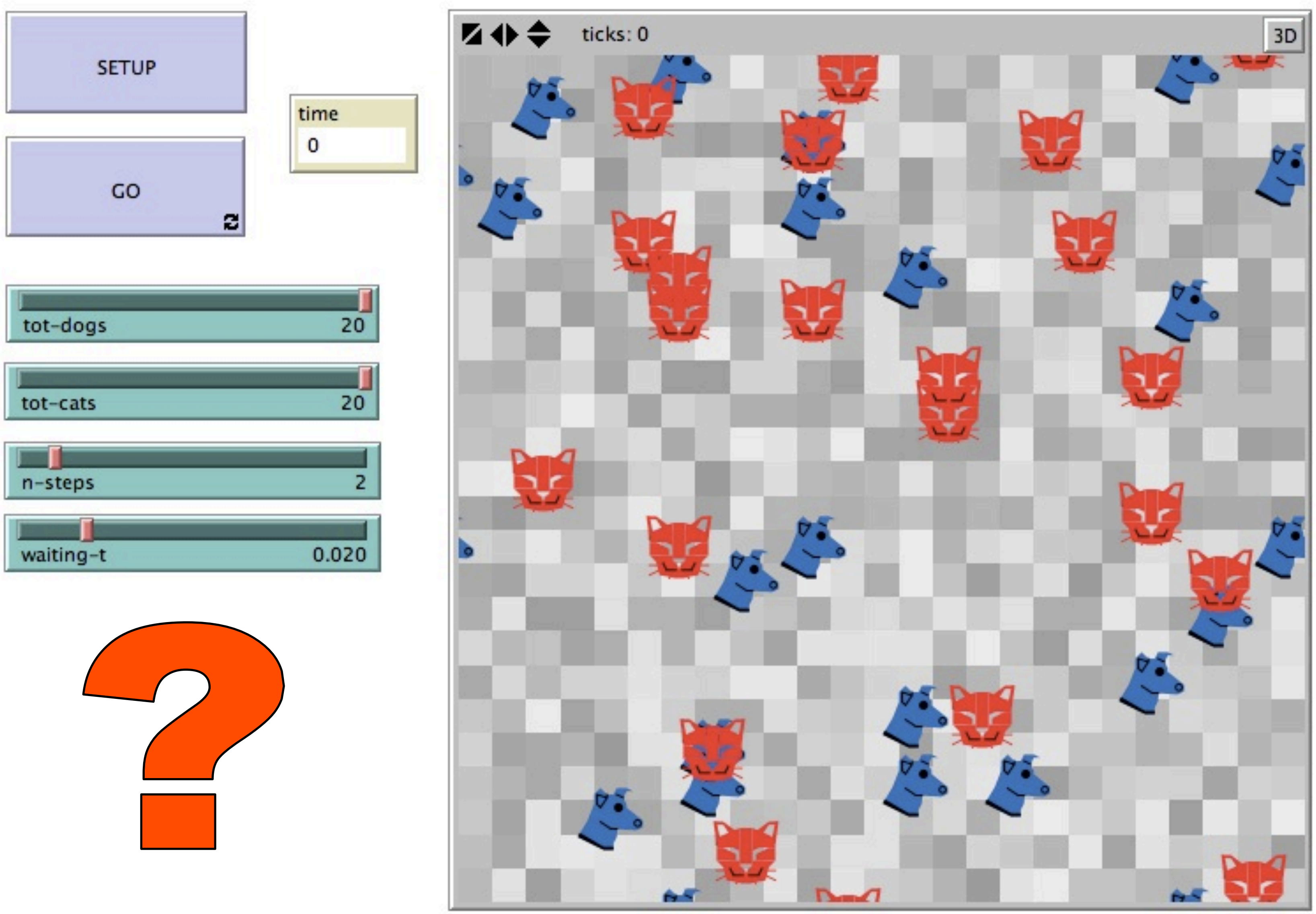
Con il comando: **set** mylist **fput** 47 mylist
la lista “mylist” diventa: [47 2 10 3.14 "Bob”]

4-Dogs and cats.nlogo



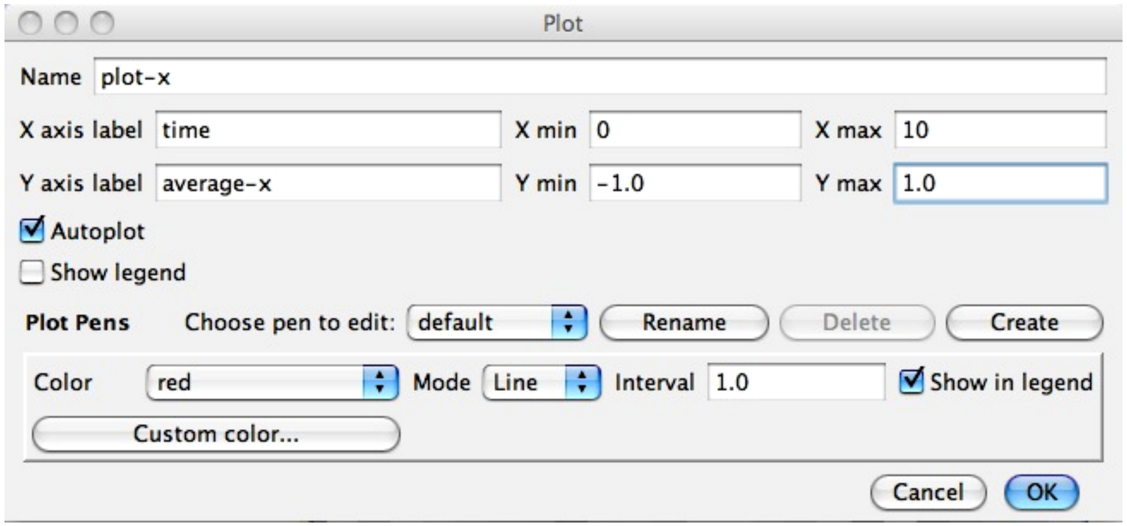
```
to go ;; forever button
  ask dogs
  [ ;; ask each dog to point one cat at random within a radius of 10 patches, to go forward
    ;; for n-steps towards that cat and to wait a fraction of time
    set heading towards target-dogs 10 fd n-steps wait waiting-t
    if (any? cats-here)                 ;; if there are some cats on the same patch where stays the dog
        [ ask one-of cats-here [die] ]  ;; one of the cats at random is eliminated
  ]
  ask cats
  [ ;; ask each cat to individuate a dog at random within a radius of 3 patches around it,
    ;; to point in the opposite direction, to go forward n-steps and to wait a fraction of time
    set heading towards target-cats 3 rt 180 fd n-steps wait waiting-t
  ]
  set time (time + 1)
end

to-report target-dogs [radius]
  ifelse (any? cats in-radius radius)            ;; Check if there are cats within the circle considered;
     [ report one-of cats in-radius radius]      ;; In this case, one of the cats becomes the target
     [ let target 0                              ;; otherwise the target is selected choosing at random
       ask patch-here [set target one-of neighbors] ;; one of the 8 patches around the patch where
       report target]                            ;; is situated the dog which called the reporter
end

to-report target-cats [radius]
  ifelse (any? dogs in-radius radius)            ;; Check if there are dogs within the circle considered;
     [ report one-of dogs in-radius radius]      ;; In this case, one of the dogs becomes the target
     [ let target 0                              ;; otherwise the target is selected choosing at random
       ask patch-here [set target one-of neighbors] ;; one of the 8 patches around the patch where
       report target]                            ;; is situated the cat which called the reporte
end
```

4-Dogs and cats.nlogo



tot-dogs 20
tot-cats 20
n-steps 2
waiting-t 0.020

Grafici e Istogrammi



Plot

Name	plot-x				
X axis label	time	X min	0	X max	10
Y axis label	average-x	Y min	-1.0	Y max	1.0

✓ Autoplot

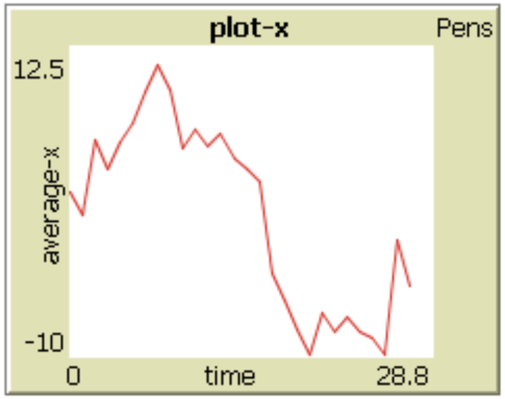
Show legend

Plot Pens — Choose pen to edit: default

Color: red — Mode: Line — Interval: 1.0 — ✓ Show in legend

plot *number*

Incrementa il valore x della "current plot pen" della quantità specificata nella variabile "*Interval*", quindi traccia una linea (una barra o un punto, a seconda del "Mode") in corrispondenza del nuovo valore della x e al valore y specificato da *number*.



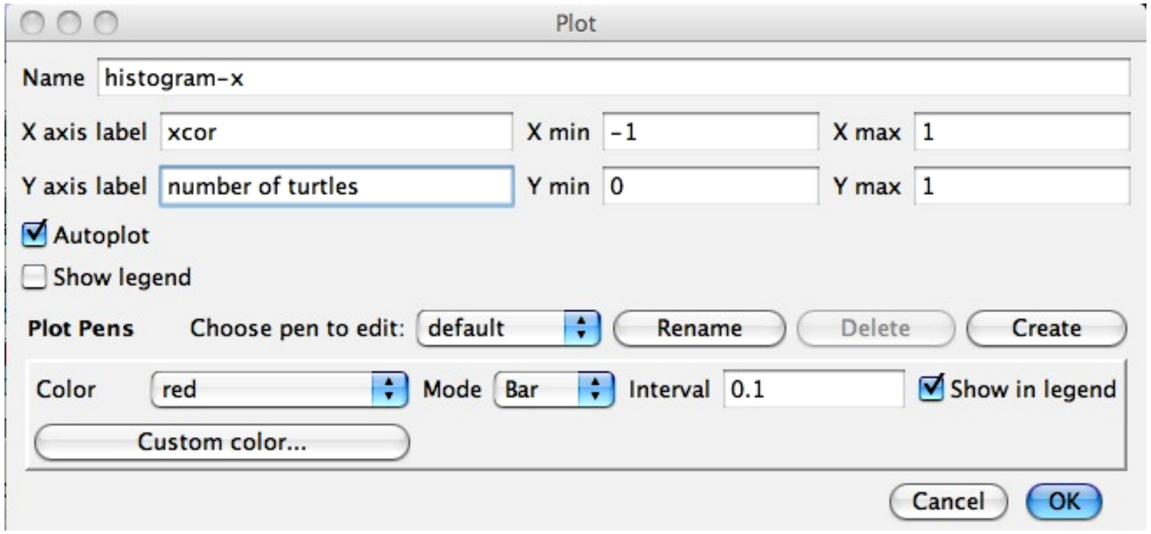
Inizializzazione del plot nel codice:

```
set-current-plot "plot-x"
set-plot-x-range 0 10
set-plot-y-range -1.0 1.0
```

plotxy *number1 number2*

Sposta la "current plot pen" al punto di coordinate (*number1, number2)*. Se la "pen" è "down", verrà disegnata una linea, una barra o un punto (a seconda del "Mode").

Grafici e Istogrammi

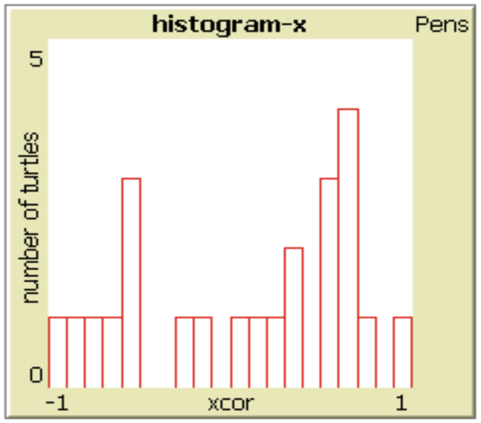


Plot

Name: histogram-x
X axis label: xcor — X min: -1 — X max: 1
Y axis label: number of turtles — Y min: 0 — Y max: 1
☑ Autoplot
☐ Show legend
Plot Pens — Choose pen to edit: default — Rename — Delete — Create
Color: red — Mode: Bar — Interval: 0.1 — ☑ Show in legend
Custom color...
Cancel — OK

histogram *list*
or
histogram [*variable*] of *agentset*

Disegna un istogramma che mostra la distribuzione di frequenza dei valori in una lista o nelle variabili proprietarie di un certo agentset. L'altezza di una certa barra dell'istogramma rappresenta il numero di occorrenze di un certo valore

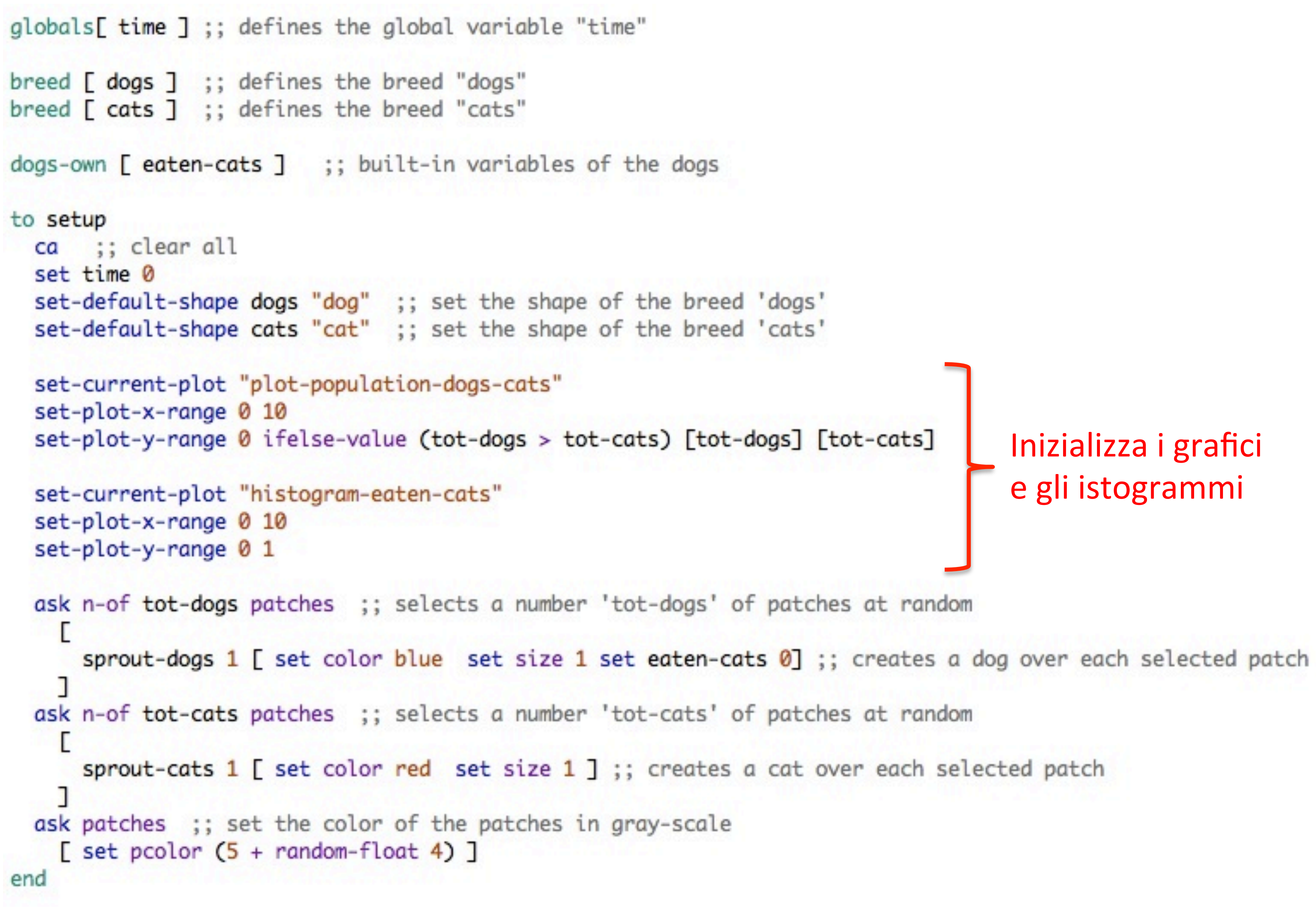


Inizializzazione dell' istogramma nel codice

```
set-current-plot "histogram-x"
set-plot-x-range -1.0 1.0
set-plot-y-range 0 1
```

Usare set-plot-x-range per controllare il range di valori che deve essere graficato e settare il "pen interval" (direttamente col comando "set-plot-pen-interval", o indirettamente con "set-histogram-num-bars") per controllare da quante barre deve essere costituito l'istogramma.

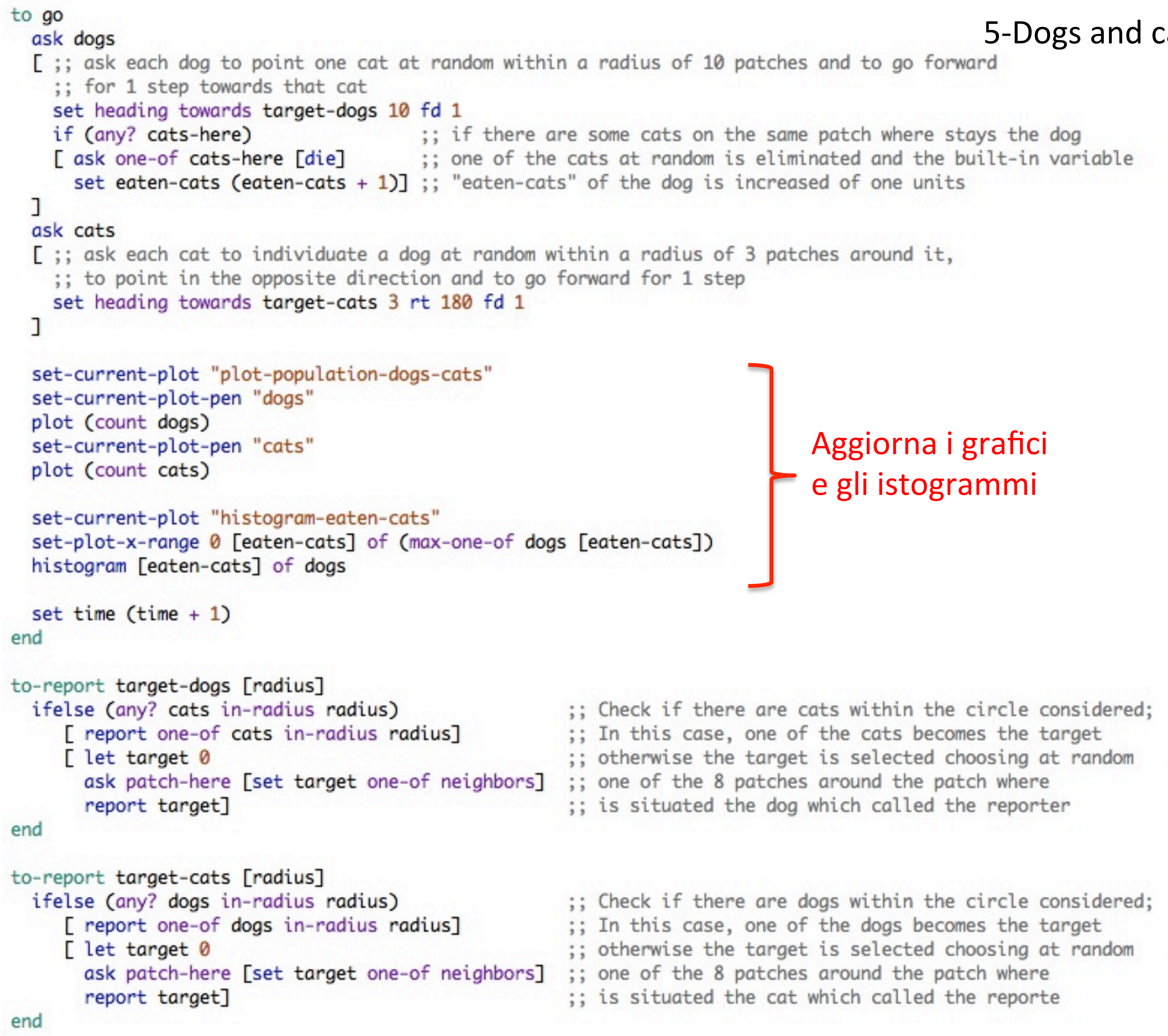
5-Dogs and cats.nlogo



```
globals[ time ] ;; defines the global variable "time"

breed [ dogs ]  ;; defines the breed "dogs"
breed [ cats ]  ;; defines the breed "cats"

dogs-own [ eaten-cats ]   ;; built-in variables of the dogs

to setup
  ca   ;; clear all
  set time 0
  set-default-shape dogs "dog"  ;; set the shape of the breed 'dogs'
  set-default-shape cats "cat"  ;; set the shape of the breed 'cats'

  set-current-plot "plot-population-dogs-cats"
  set-plot-x-range 0 10
  set-plot-y-range 0 ifelse-value (tot-dogs > tot-cats) [tot-dogs] [tot-cats]

  set-current-plot "histogram-eaten-cats"
  set-plot-x-range 0 10
  set-plot-y-range 0 1

  ask n-of tot-dogs patches  ;; selects a number 'tot-dogs' of patches at random
    [
      sprout-dogs 1 [ set color blue  set size 1 set eaten-cats 0] ;; creates a dog over each selected patch
    ]
  ask n-of tot-cats patches  ;; selects a number 'tot-cats' of patches at random
    [
      sprout-cats 1 [ set color red  set size 1 ] ;; creates a cat over each selected patch
    ]
  ask patches  ;; set the color of the patches in gray-scale
    [ set pcolor (5 + random-float 4) ]
end
```

Inizializza i grafici
e gli istogrammi

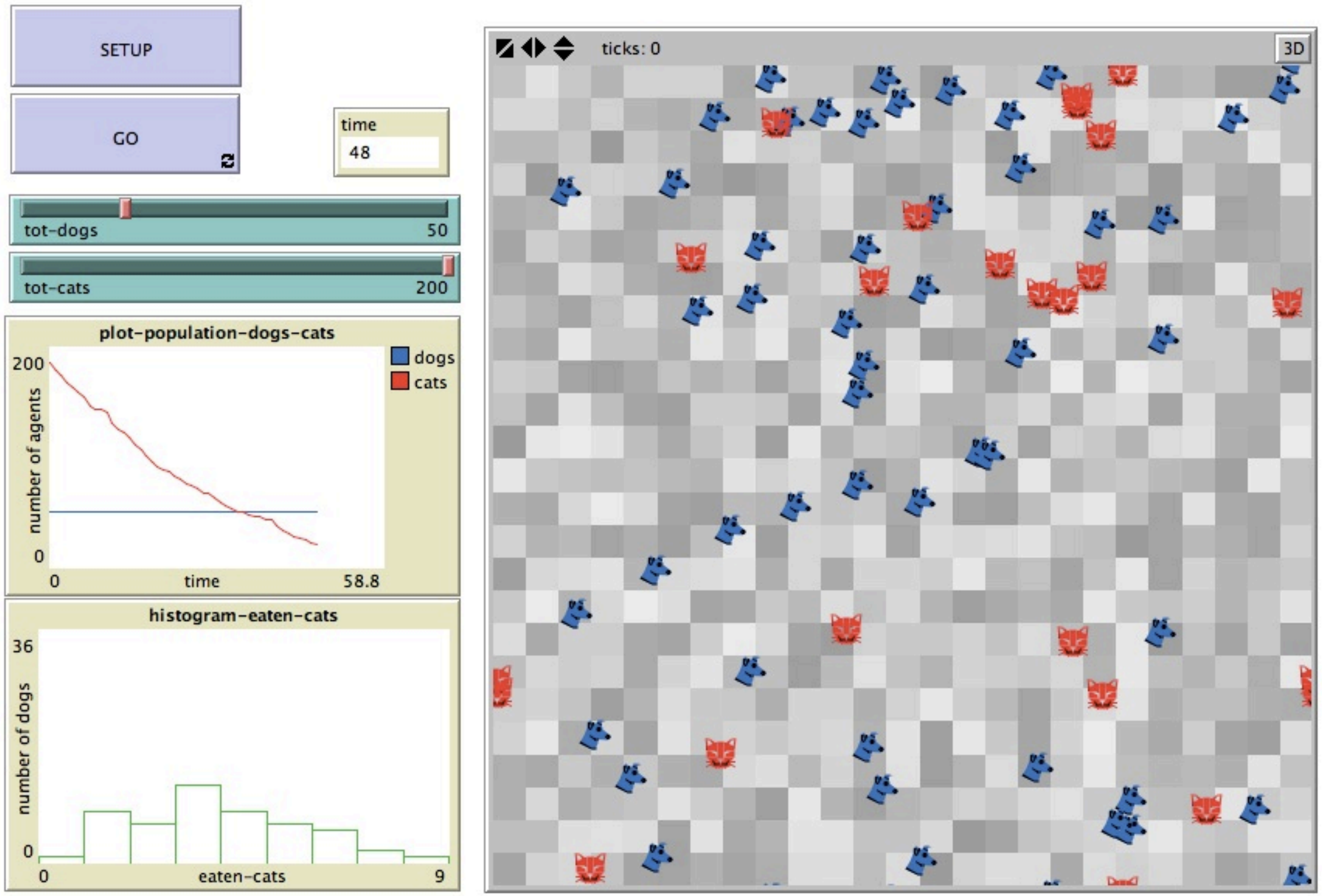
5-Dogs and cats.nlogo

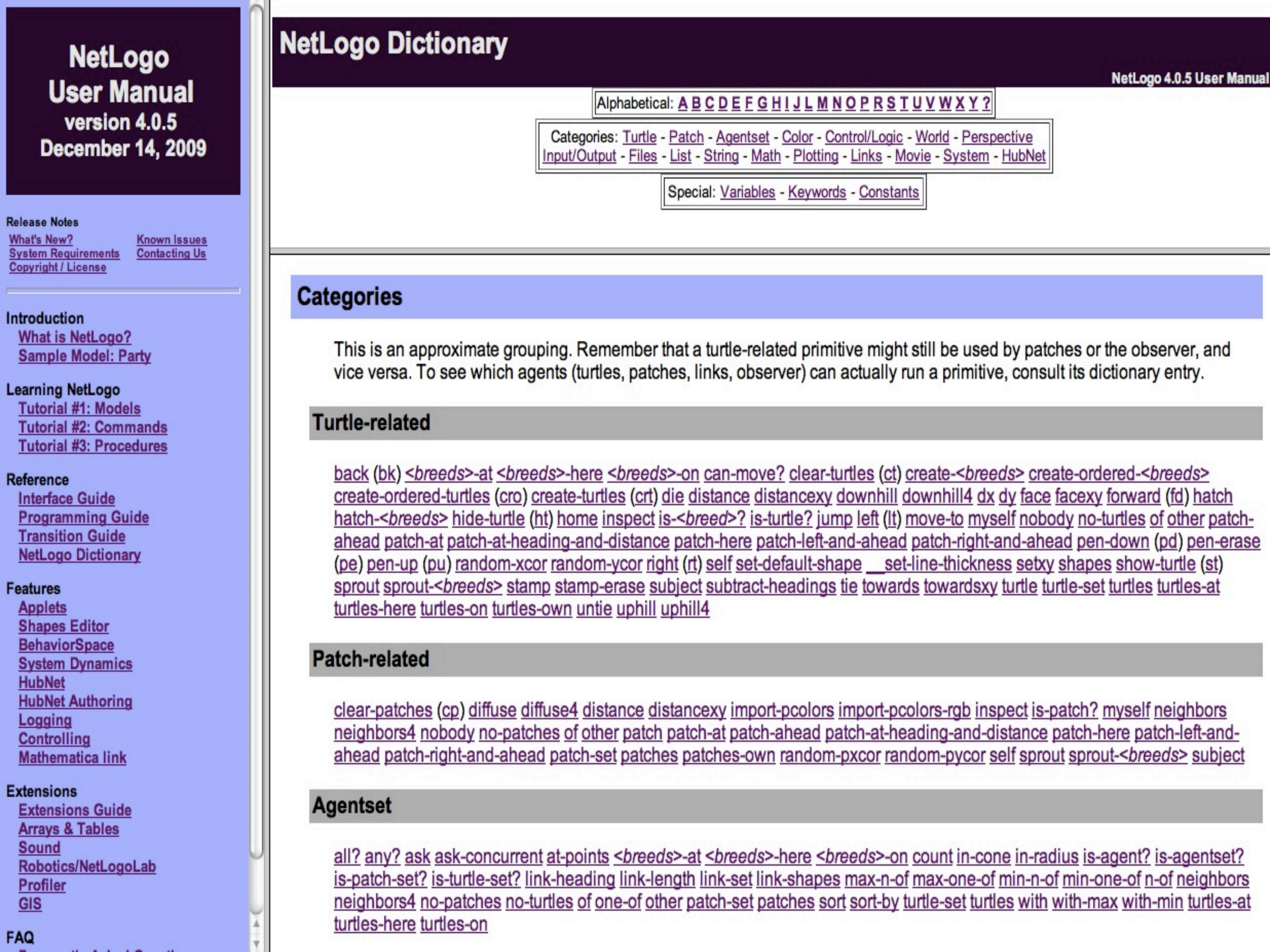


```
to go
  ask dogs
  [ ;; ask each dog to point one cat at random within a radius of 10 patches and to go forward
    ;; for 1 step towards that cat
    set heading towards target-dogs 10 fd 1
    if (any? cats-here)               ;; if there are some cats on the same patch where stays the dog
    [ ask one-of cats-here [die]      ;; one of the cats at random is eliminated and the built-in variable
      set eaten-cats (eaten-cats + 1)] ;; "eaten-cats" of the dog is increased of one units
  ]
  ask cats
  [ ;; ask each cat to individuate a dog at random within a radius of 3 patches around it,
    ;; to point in the opposite direction and to go forward for 1 step
    set heading towards target-cats 3 rt 180 fd 1
  ]

  set-current-plot "plot-population-dogs-cats"
  set-current-plot-pen "dogs"
  plot (count dogs)
  set-current-plot-pen "cats"
  plot (count cats)

  set-current-plot "histogram-eaten-cats"
  set-plot-x-range 0 [eaten-cats] of (max-one-of dogs [eaten-cats])
  histogram [eaten-cats] of dogs

  set time (time + 1)
end

to-report target-dogs [radius]
  ifelse (any? cats in-radius radius)              ;; Check if there are cats within the circle considered;
     [ report one-of cats in-radius radius]        ;; In this case, one of the cats becomes the target
     [ let target 0                                ;; otherwise the target is selected choosing at random
       ask patch-here [set target one-of neighbors] ;; one of the 8 patches around the patch where
       report target]                              ;; is situated the dog which called the reporter
end

to-report target-cats [radius]
  ifelse (any? dogs in-radius radius)              ;; Check if there are dogs within the circle considered;
     [ report one-of dogs in-radius radius]        ;; In this case, one of the dogs becomes the target
     [ let target 0                                ;; otherwise the target is selected choosing at random
       ask patch-here [set target one-of neighbors] ;; one of the 8 patches around the patch where
       report target]                              ;; is situated the cat which called the reporte
end
```

Aggiorna i grafici
e gli istogrammi

5-Dogs and cats.nlogo





NetLogo User Manual
version 4.0.5
December 14, 2009

Release Notes
What's New? Known Issues
System Requirements Contacting Us
Copyright / License

Introduction
What is NetLogo?
Sample Model: Party

Learning NetLogo
Tutorial #1: Models
Tutorial #2: Commands
Tutorial #3: Procedures

Reference
Interface Guide
Programming Guide
Transition Guide
NetLogo Dictionary

Features
Applets
Shapes Editor
BehaviorSpace
System Dynamics
HubNet
HubNet Authoring
Logging
Controlling
Mathematica link

Extensions
Extensions Guide
Arrays & Tables
Sound
Robotics/NetLogoLab
Profiler
GIS

FAQ

NetLogo Dictionary

NetLogo 4.0.5 User Manual

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

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

Special: Variables - Keywords - Constants

Categories

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

Turtle-related

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

Patch-related

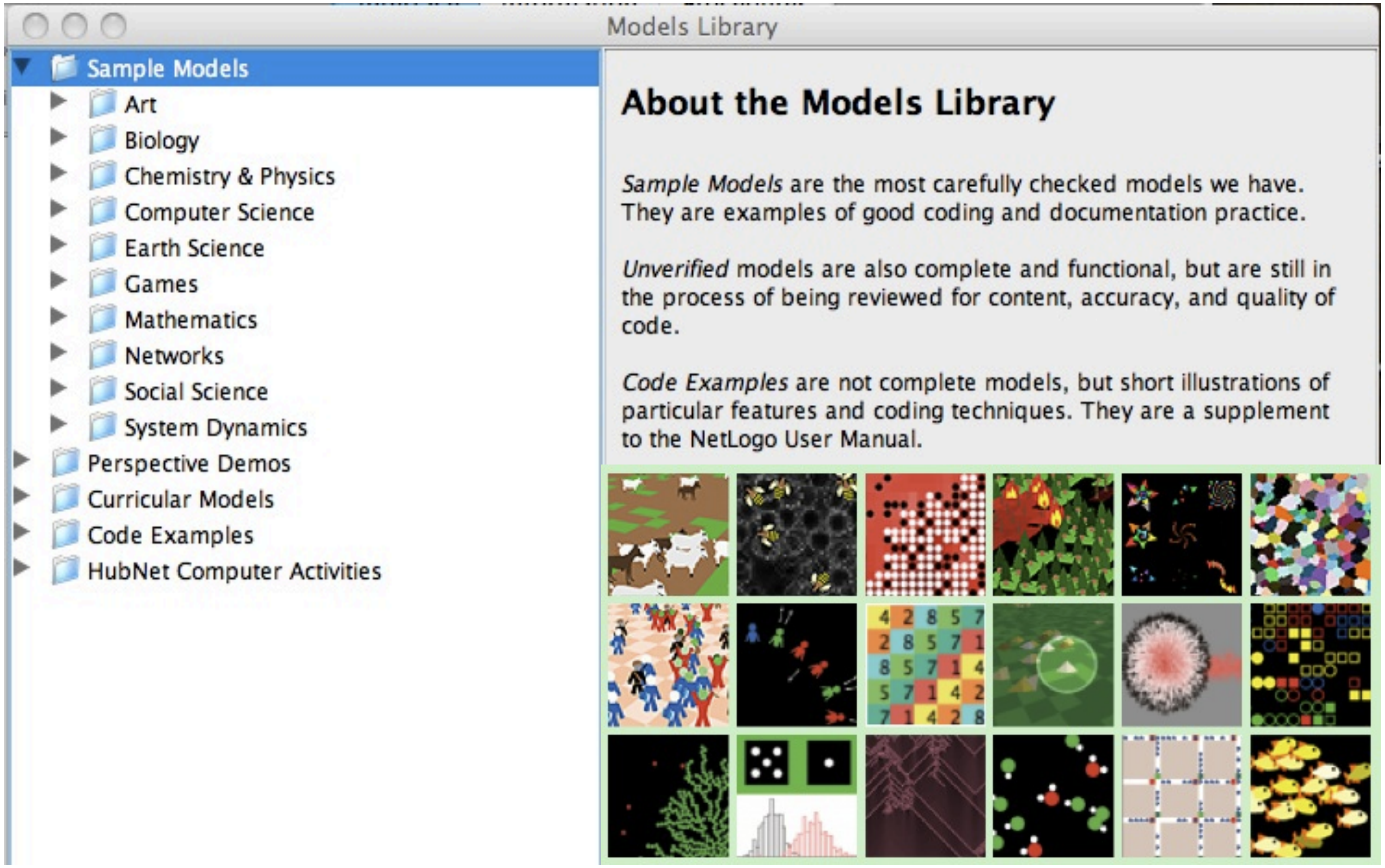
clear-patches (cp) diffuse diffuse4 distance distancexy import-pcolors import-pcolors-rgb inspect is-patch? myself neighbors neighbors4 nobody no-patches of other patch patch-at patch-ahead patch-at-heading-and-distance patch-here patch-left-and-ahead patch-right-and-ahead patch-set patches patches-own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

Agentset

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

NeLogo and Agent-Based Simulations

NetLogo has extensive documentation and tutorials. It also comes with a Models Library, which is a large collection of pre-written simulations that can be used and modified. These simulations address many content areas in the natural and social sciences, including biology and medicine, physics and chemistry, mathematics and computer science, and economics and social psychology. Several model-based inquiry curricula using NetLogo are currently under development.



Models Library

- Sample Models
  - Art
  - Biology
  - Chemistry & Physics
  - Computer Science
  - Earth Science
  - Games
  - Mathematics
  - Networks
  - Social Science
  - System Dynamics
- Perspective Demos
- Curricular Models
- Code Examples
- HubNet Computer Activities

About the Models Library

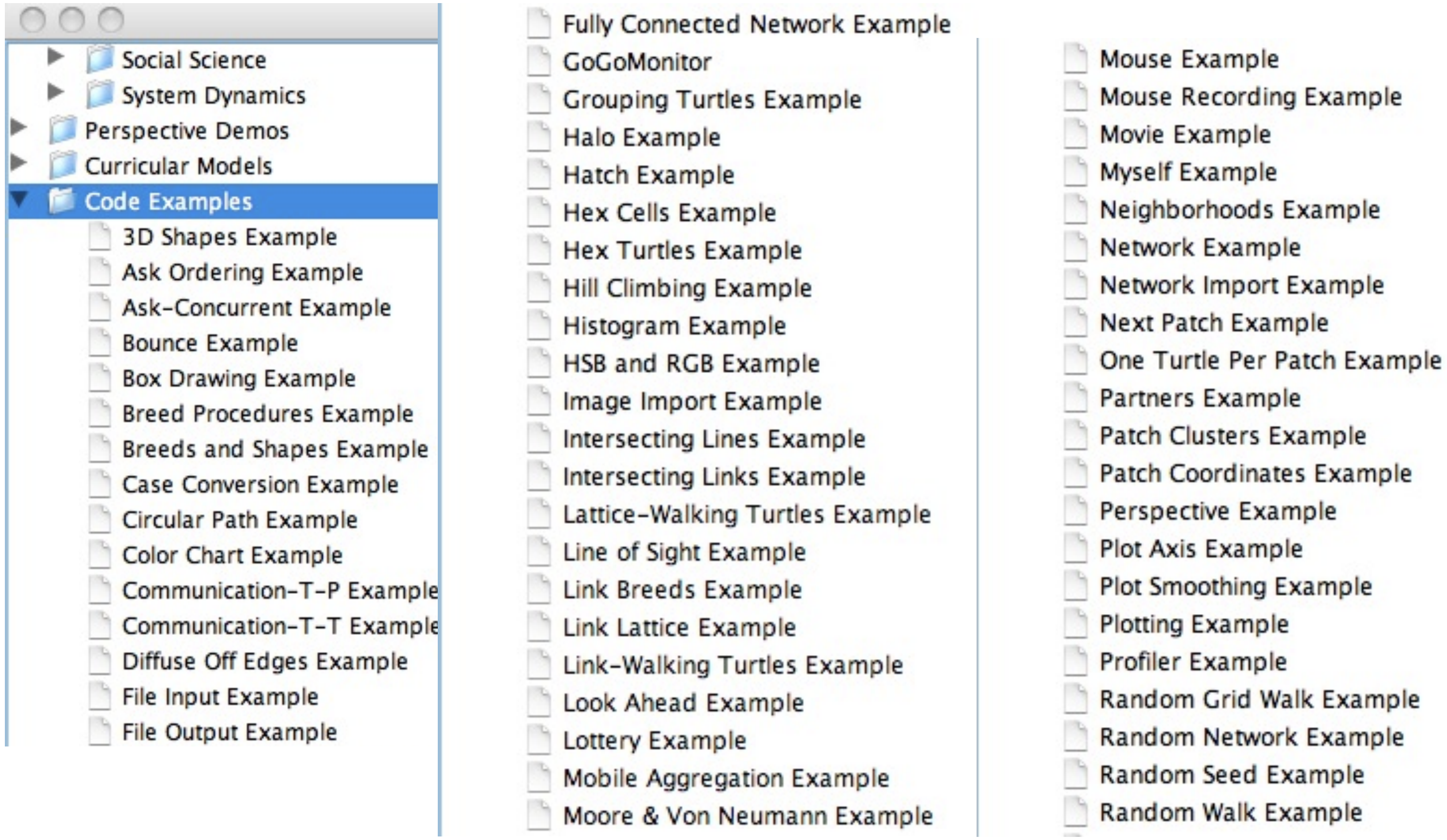
Sample Models are the most carefully checked models we have. They are examples of good coding and documentation practice.

Unverified models are also complete and functional, but are still in the process of being reviewed for content, accuracy, and quality of code.

Code Examples are not complete models, but short illustrations of particular features and coding techniques. They are a supplement to the NetLogo User Manual.

NeLogo and Agent-Based Simulations

NetLogo has extensive documentation and tutorials. It also comes with a Models Library, which is a large collection of pre-written simulations that can be used and modified. These simulations address many content areas in the natural and social sciences, including biology and medicine, physics and chemistry, mathematics and computer science, and economics and social psychology. Several model-based inquiry curricula using NetLogo are currently under development.



▶ Social Science
▶ System Dynamics
▶ Perspective Demos
▶ Curricular Models
▼ Code Examples
3D Shapes Example
Ask Ordering Example
Ask-Concurrent Example
Bounce Example
Box Drawing Example
Breed Procedures Example
Breeds and Shapes Example
Case Conversion Example
Circular Path Example
Color Chart Example
Communication-T-P Example
Communication-T-T Example
Diffuse Off Edges Example
File Input Example
File Output Example
Fully Connected Network Example
GoGoMonitor
Grouping Turtles Example
Halo Example
Hatch Example
Hex Cells Example
Hex Turtles Example
Hill Climbing Example
Histogram Example
HSB and RGB Example
Image Import Example
Intersecting Lines Example
Intersecting Links Example
Lattice-Walking Turtles Example
Line of Sight Example
Link Breeds Example
Link Lattice Example
Link-Walking Turtles Example
Look Ahead Example
Lottery Example
Mobile Aggregation Example
Moore & Von Neumann Example
Mouse Example
Mouse Recording Example
Movie Example
Myself Example
Neighborhoods Example
Network Example
Network Import Example
Next Patch Example
One Turtle Per Patch Example
Partners Example
Patch Clusters Example
Patch Coordinates Example
Perspective Example
Plot Axis Example
Plot Smoothing Example
Plotting Example
Profiler Example
Random Grid Walk Example
Random Network Example
Random Seed Example
Random Walk Example