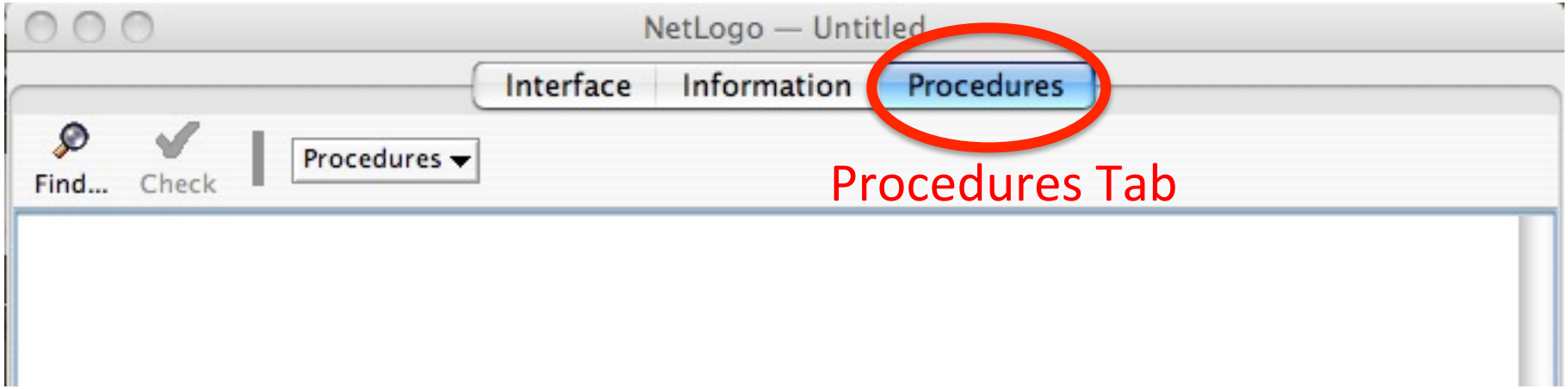
Introduzione al Linguaggio di Programmazione

NetLogo ha un suo proprio linguaggio di programmazione di alto livello (un meta-linguaggio molto simile all'inglese parlato), basato sul Java (un noto linguaggio di programmazione "object oriented") ma con una struttura più semplice fatta da una sequenza di procedure contenenti il codice del programma (case insensitive).

Si può accedere al codice per mezzo dell'apposita sezione "Procedures"...



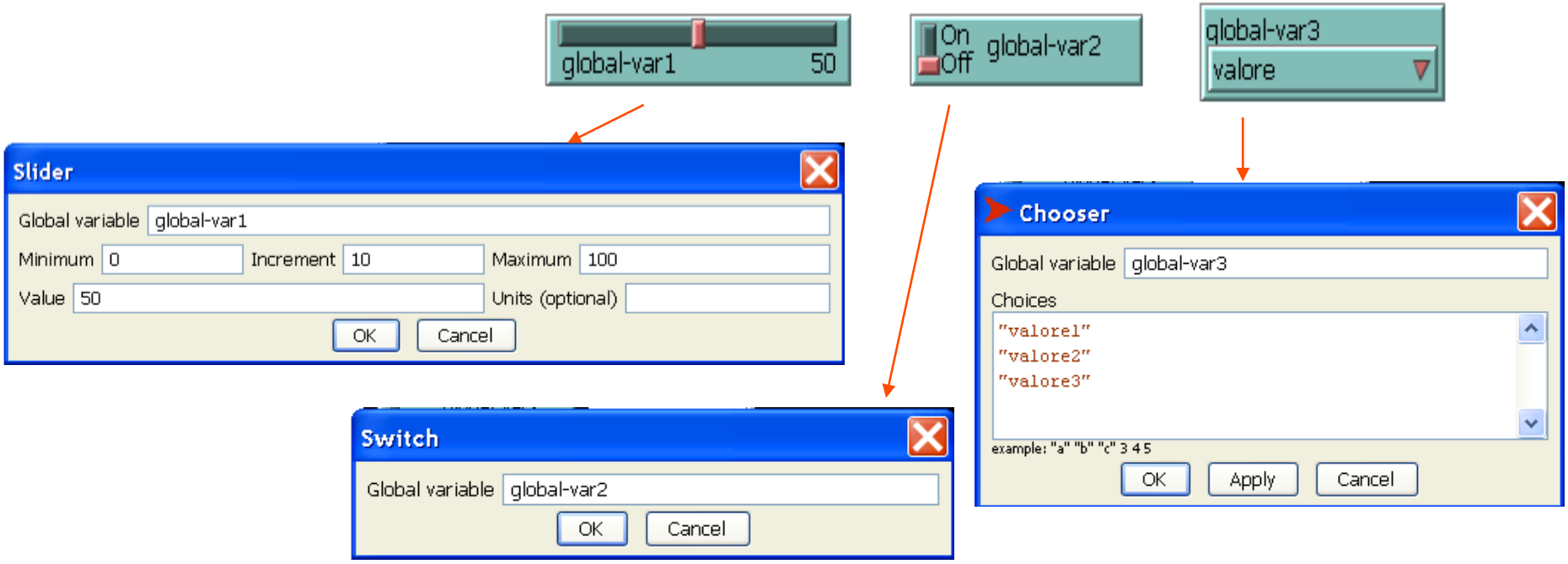
Le Variabili

In NETLOGO si possono definire 3 tipi di variabili:

Global Variables

Le **variabili globali** esistono in esemplari unici e sono visibili da qualunque punto del codice. Vengono dichiarate di solito all'inizio del codice, fuori da ogni altra procedura, oppure direttamente dall'interfaccia di sviluppo attraverso gli Sliders, gli Switch o i Chooser, e vengono aggiornate con la riga di comando:

set nome_var valore



Le Variabili

In NETLOGO si possono definire 3 tipi di variabili:

Global Variables

Le **variabili globali** esistono in esemplari unici e sono visibili da qualunque punto del codice. Vengono dichiarate di solito all'inizio del codice, fuori da ogni altra procedura, oppure direttamente dall'interfaccia di sviluppo attraverso gli Sliders, gli Switch o i Chooser, e vengono aggiornate con la riga di comando:
set nome_var valore

Local Variables

Le **variabili locali** esistono anch'esse in esemplari unici, ma sono visibili solo all'interno della procedura dove esse sono state dichiarate per mezzo della riga di comando: **let** nome_var value

Built-in Variables

Le **variabili proprietarie** esistono invece in molte copie diverse (istanze), una per ogni turtle (turtles-own), per ogni patch (patches-own) o link (links-own), ovviamente con valori diversi. Importante: queste variabili devono essere utilizzate SOLO nel corretto contesto (legato a turtles, patches o links) 

Esempi di variabili proprietarie

turtles-own

who
xcor, ycor
heading
color
shape
size
label
turtles-own [var1 var2 ...]

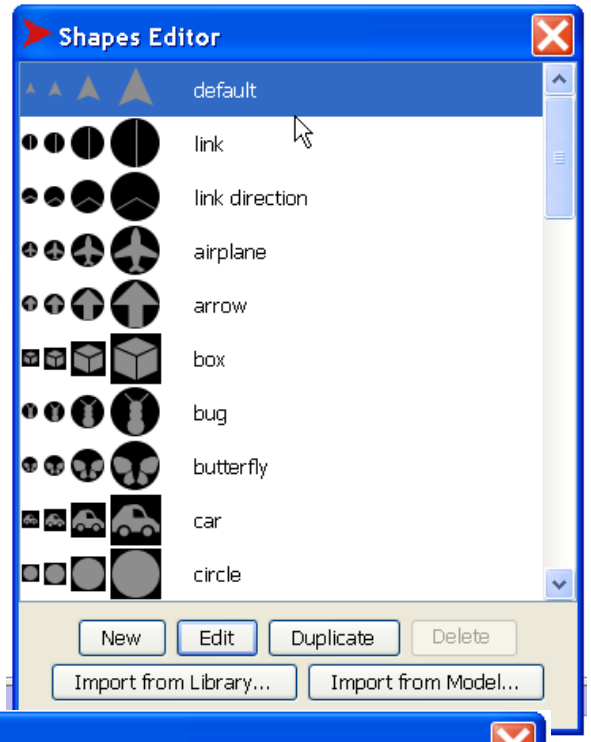
patches-own

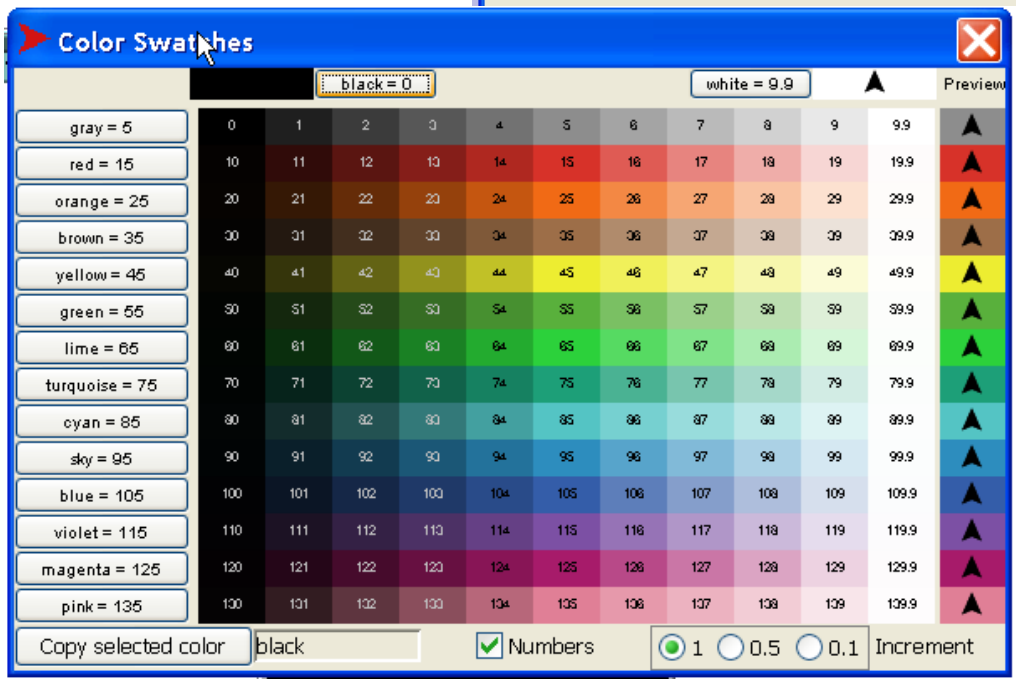
pxcor, pycor
pcolor
plabel
plabel-color
patches-own [pvar1 pvar2 ...]

breeds-own

breed [boxes]
breed [cars]

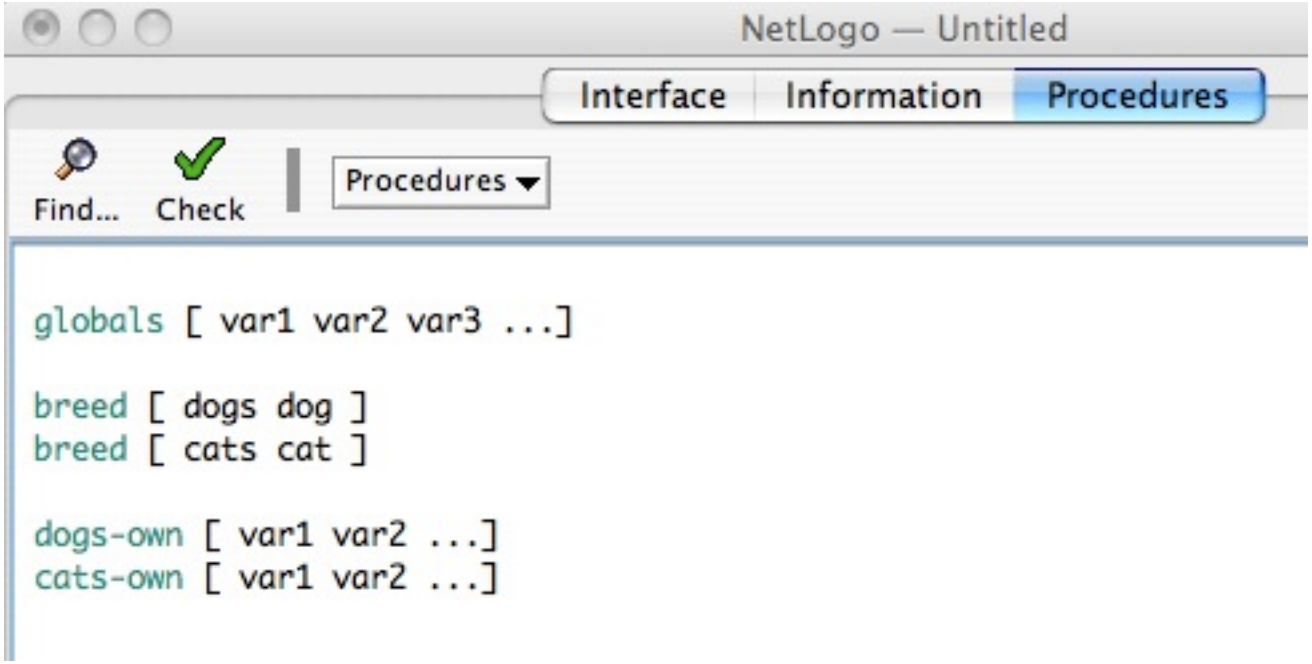
agentset





Definizione delle variabili

La sezione “Procedures” di un tipico modello di NetLogo si apre con alcuni blocchi, ovvero porzioni di codice delimitate da parentesi quadre [...], che permettono di definire le variabili globali (solo quelle che non sono già state definite tramite gli elementi dell’interfaccia grafica), le breeds (specie) di turtles e le loro variabili proprietarie:



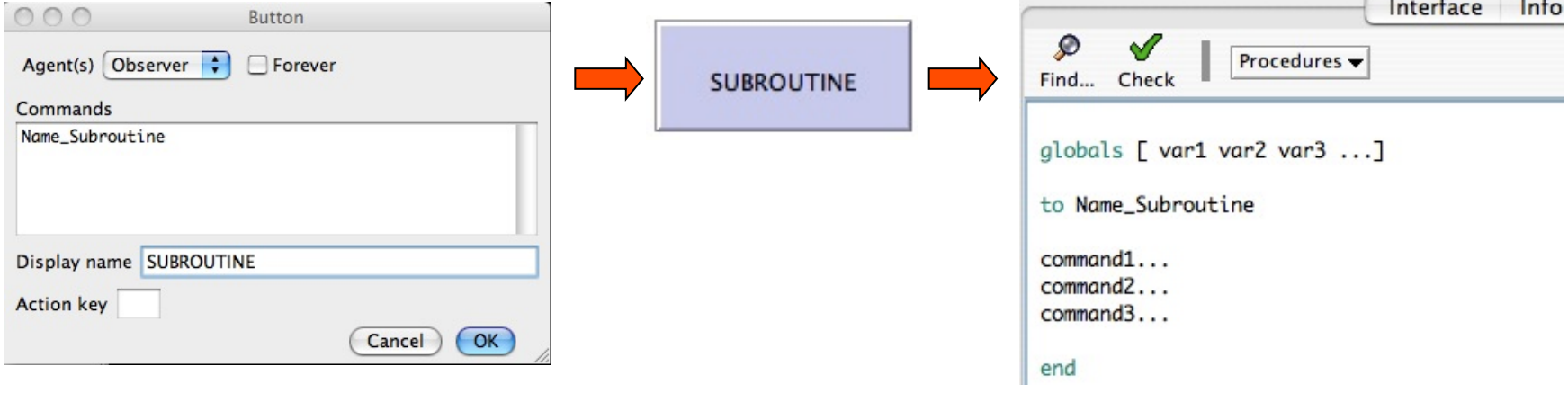
Le Procedure

In NETLOGO il codice è racchiuso in una sequenza di **procedure**, ovvero di blocchi contenenti una lista di comandi che dicono alle turtles come muoversi e/o alle patches come comportarsi. Una volta definita una procedura, la si può richiamare ovunque all'interno del programma (o dall'interfaccia grafica) semplicemente scrivendone il nome.

Esistono due tipi principali di procedure:

Subroutines

Sono procedure richiamate da altre subroutines nel codice o direttamente dall'interfaccia grafica attraverso la pressione dei buttons. Le Subroutines possono accettare valori in input ma non restituiscono alcun output. Tutte le subroutines cominciano con la parola chiave "to" (seguita dal nome della subroutine) e terminano con "end":



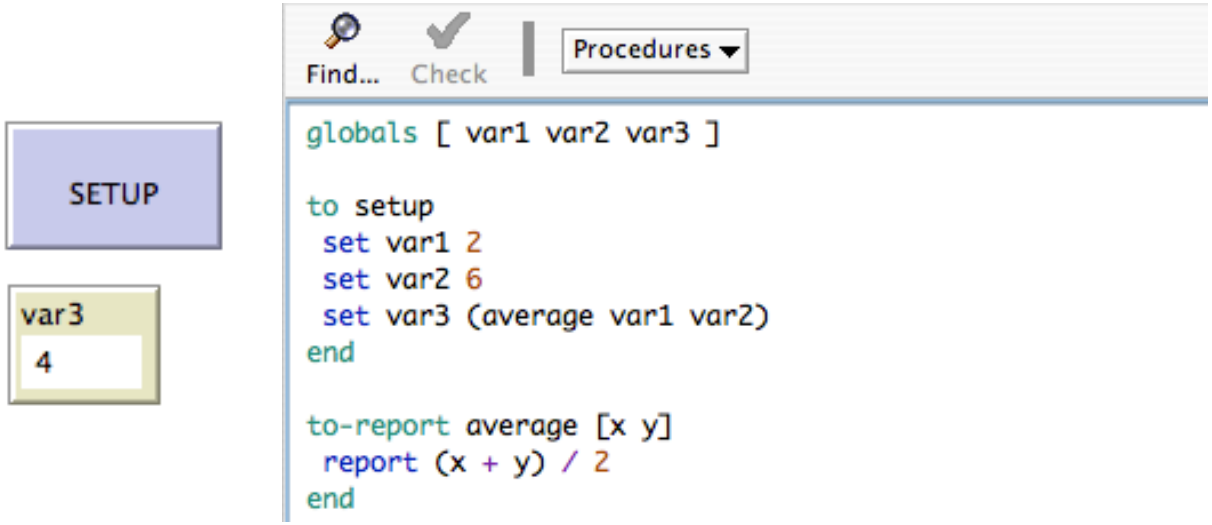
Le Procedure

In NETLOGO il codice è racchiuso in una sequenza di **procedure**, ovvero di blocchi contenenti una lista di comandi che dicono alle turtles come muoversi e/o alle patches come comportarsi. Una volta definita una procedura, la si può richiamare ovunque all'interno del programma (o dall'interfaccia grafica) semplicemente scrivendone il nome.

Esistono due tipi principali di procedure:

Reporters

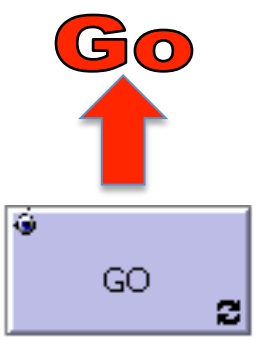
Sono procedure richiamate da altri reporters o subroutines. Esse accettano valori in input e restituiscono sempre un singolo valore in output, per mezzo del comando "report". Tutti i reporters iniziano con la parola chiave "to-report" (seguita dal nome del reporter) e terminano con la parola chiave "end":



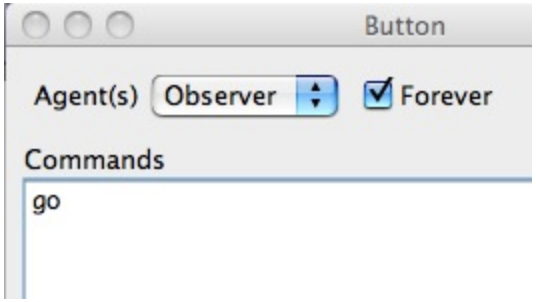
Le Procedure

Abbiamo già visto che le principali subroutines di NETLOGO sono due:

E' richiamata di solito dalla interfaccia grafica per mezzo del corrispondente button e permette di inizializzare le variabili globali, i plots e il World. Ma consente anche di definire le condizioni iniziali della simulazione:
- inizializzando le patches e le loro variabili proprietarie;
- creando le turtles e/o i links e inizializzando le loro variabili proprietarie.



E' anch'essa richiamata dalla interfaccia grafica per mezzo del corrispondente button e di solito contiene le istruzioni necessarie per controllare l'evoluzione temporale delle turtles e delle patches.



Si noti che il button GO è del tipo "forever": ciò significa che la subroutine "go" contiene le istruzioni che specificano l'evoluzione di un singolo passo temporale di tutti gli agenti, ma che verranno ripetute finchè il button GO sarà ripremuto.

Comandi per la generazione di numeri (pseudo)casuali

random

random *number*

If *number* is positive, reports a random integer greater than or equal to 0, but strictly less than *number*.

If *number* is negative, reports a random integer less than or equal to 0, but strictly greater than *number*.

If *number* is zero, the result is always 0 as well.

random-float

random-float *number*

If *number* is positive, reports a random floating point number greater than or equal to 0 but strictly less than *number*.

If *number* is negative, reports a random floating point number less than or equal to 0, but strictly greater than *number*.

If *number* is zero, the result is always 0.

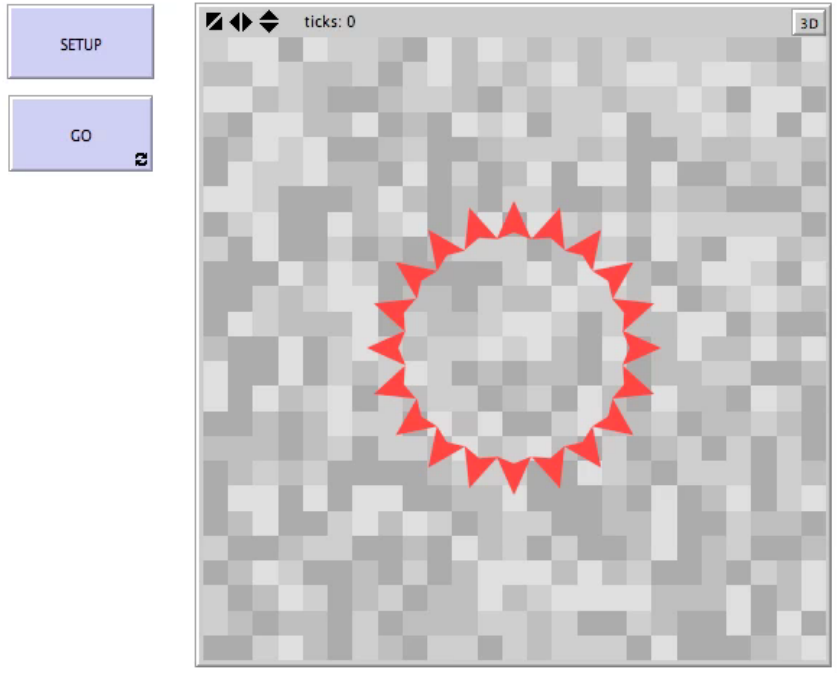
random-exponential
random-gamma
random-normal
random-poisson

random-exponential *mean*
random-gamma *alpha lambda*
random-normal *mean standard-deviation*
random-poisson *mean*

Reports an accordingly distributed random number with the *mean* and, in the case of the normal distribution, the *standard-deviation*.

Procedure e comandi: un semplice esempio...

1-Ask Agentset.nlogo



Il comando "create" (observer-context) crea *n* turtles sulla patch origine (al centro di una topologia"box"). In alternativa, il comando "sprout" (patch-context) crea *n* turtles sulla patch corrente ed esegue i comandi del blocco successivo.

"ask agentset" è uno dei principali comandi di NetLogo ed è usato per chiedere alle turtles, patches o links di eseguire certe istruzioni (in un ordine casuale...)

Il comando "fd" (turtle-context) sposta le turtles di un certo numero di passi in avanti (specificati dal numero successivo...)

NetLogo — Ask Agentset {/Users/Alex/Desktop/NETLOGO-LI

Interface | Information | Procedures

Find... | Check | Procedures

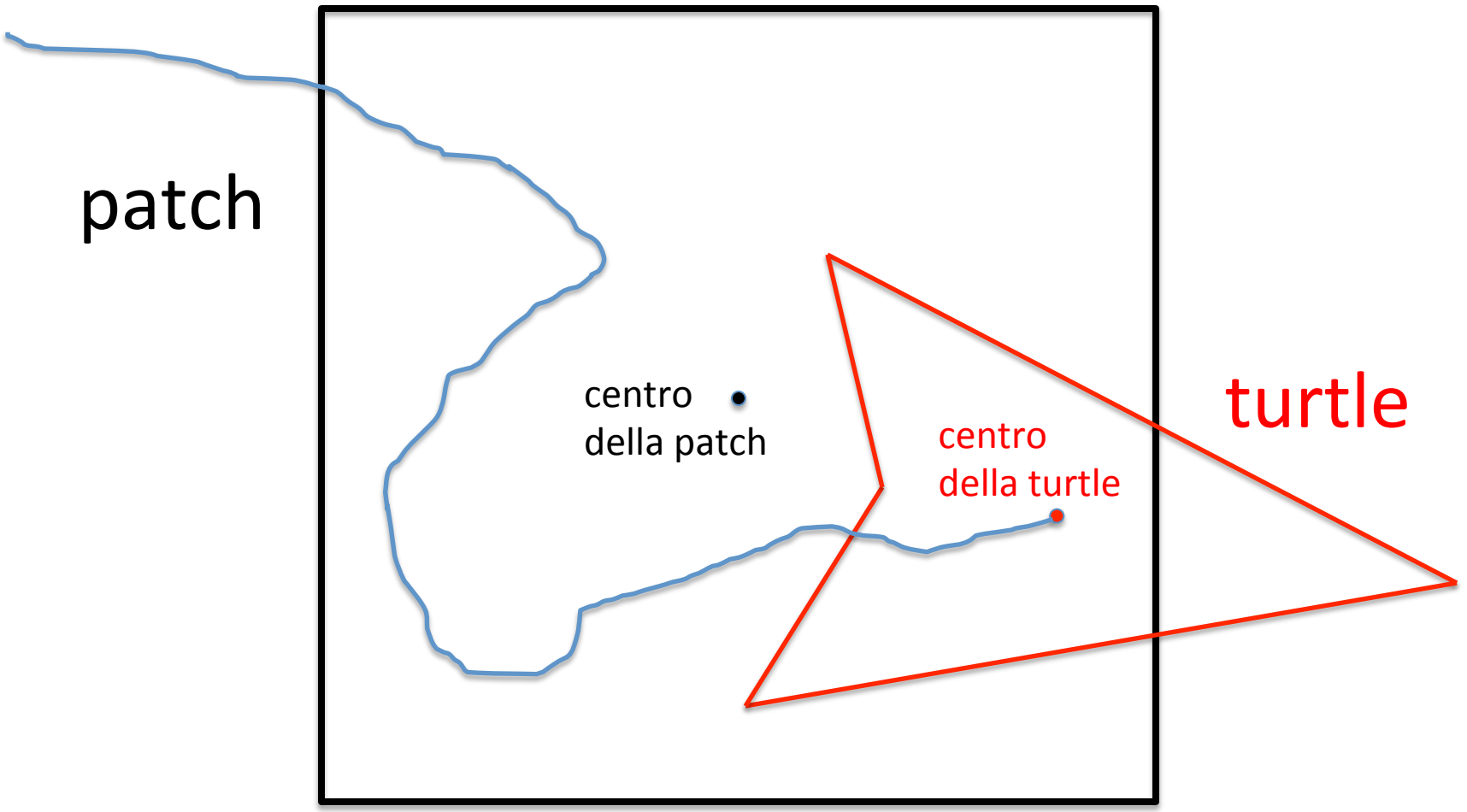
```
;; use double semicolon for comments...

;; "setup" is executed only one time, by pressing the SETUP button
to setup
  ;; clear all the turtles and reset the patches
  ca
  ;; create 20 turtles with clock-wise ordered "who" and "heading"
  create-ordered-turtles 20
  ask turtles
    ;; set "color" and "size" of turtles and move 5 steps forward (fd)
    [
      set color red
      set size 2
      fd 5
    ]
  ask patches
    ;; set the color of the patches in a gray-scale
    [ set pcolor (5 + random 4) ]
end

;; forever button; routine "go" is repeated indefinitely
to go
  ask turtles
  [
  ;; ask turtles to move 1 step and to wait 0.1 seconds
    fd 1 wait 0.1
  ]
end
```

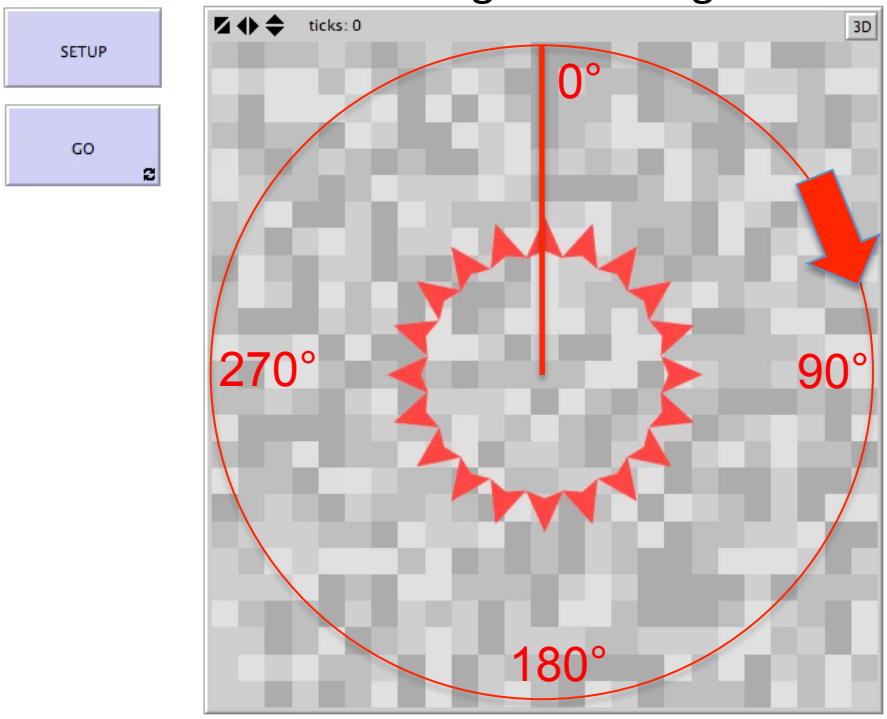
Nota sulle coordinate di Turtles e Patches...

Abbiamo già detto che il centro delle turtles ha coordinate (xcor, ycor) espresse da numeri reali, mentre le coordinate delle patches (pxcor, pycor) sono sempre espresse da numeri interi. Ciò significa che il centro di una turtle non deve necessariamente trovarsi al centro delle patches ma può muoversi in qualunque posizione al loro interno:



Una piccola variante dell'esempio precedente...

2-Ask Agentset.nlogo



heading

E' una variabile proprietaria delle turtles e indica la direzione verso la quale (towards) esse devono puntare, espressa in gradi con un numero maggiore o uguale a 0 e minore di 360.

```
;; forever button; routine "go" is repeated indefinitely
;; variant of command 'forward'
to go
  ask turtles
  [
  ;; forces each turtle to move at the center of the
  ;; next patch ahead and to wait 0.1 seconds
     let target patch-ahead 1
     set heading towards target
     fd distance target
     wait 0.1
   ]
end
```

La subroutine "go" è stata modificata per forzare le turtles a rimanere, ad ogni passo, al centro della patch successiva (patch-ahead), facendole spostare esattamente della distanza (distance) tra il centro di quest'ultima e quello della pacth precedente. Questo tipo di codice è usato molto frequentemente in sociodinamica per simulare il movimento di pedoni in ambienti confinati.