Logo 2

3 lezione

I cammini chiusi

- Definizioni
  - Diciamo ***cammino chiuso*** un cammino in cui la tartaruga ri-assume lo stato iniziale
  - Diciamo ***rotazione totale*** lungo un cammino la somma (con segno) degli angoli di cui ruota la tartaruga
- Osservazione: **teorema**
  - la rotazione totale lungo un cammino chiuso è multipla di 360.

poligoni

```
to poli :lato :angolo
fd :lato
rt :angolo
poli :lato :angolo
end

72  1   144 60 135  200 125 145 30
```

Colori: uso semplificato

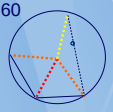
- 0 -> [0 0 0]
- 1 -> [0 0 255]
- 2 -> [0 255 0]
- 3 -> [0 255 255]
- 4 -> [255 0 0]
- 5 -> [255 0 255]
- 6 -> [255 255 0]
- 7 -> [255 255 255]
- 8 -> [155 96 59]
- 9 -> [197 136 18]
- 10 -> [100 162 64]
- 11 -> [120 187 187]
- 12 -> [255 149 119]
- 13 -> [144 113 208]
- 14 -> [255 163 0]
- 15 -> [183 183 183]

Poligoni e colori

```
to poli :lato :angolo :colore
fd :lato
rt :angolo
setpc :colore
poli :lato :angolo :colore + 1
end
```

Teorema di chiusura

- Il cammino di ***poli*** si chiude proprio quando la rotazione totale è multipla di 360



La tarta è su una circo per = triangoli

Fissata una direzione c'è SOLO UNA corda di lunghezza fissata.Quindi se ritrovo la direzione iniziale, rifaccio lo stesso cammino!!

Esempi

- Con angolo = 144 quando si ferma?
  - 144 * 5 = 360 * 2
- Con angolo = 125 quando si ferma?
  - 125 * 72 = 360 * 25

Strutture di controllo

- ifelse test [do if true list] [do if false list]
- repeat number [instruction list]
- Altre istruzioni iterative ma si raccomanda l'uso ricorsivo

Strutture di controllo

- OPERAZIONE [una lista di comandi] [vari dati]
- show map [? * ?] [5 6 7]
- [25 36 49]

Fermiamo poli!!

```
to polinew :lato :angolo
fd :lato
rt :angolo
make "rotazione :rotazione + :angolo
if 0 = modulo  :rotazione 360 [ stop]; ferma la
   sottoprocedura senza restituire valore
polinew :lato  :angolo
end

to polistop :lato :angolo
make "rotazione 0
polinew :lato :angolo
end
```

Fermiamo poli!!

```
to polinew :lato :angolo :rotazione
fd :lato
rt :angolo
if 0 = modulo  :rotazione + :angolo 360 [stop]
polinew :lato  :angolo  :rotazione + :angolo
end

to polistop :lato :angolo
polinew :lato :angolo 0
end
```

Fermiamo poli!!

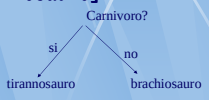
```
to polinew1 :lato :angolo
fd :lato
rt :angolo
if equalp :w  heading [stop]
polinew1 :lato  :angolo
end

to polistop1 :lato :angolo
make "w heading
polinew1 :lato :angolo
end
```

Liste

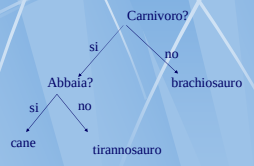
- Rispetto ai vettori sono espandibili
  - first, last, butfirst, butlast, member, item.
  - sentence fput, lput
- Coda con le operazioni **queue**, **dequeue**,
- Stack con le operazioni **push** and **pop**
- La ricorsione è il modo naturale di trattare con le liste.

Una base di conoscenza

- `make "conoscenza [[e' carnivoro?]tirannosauro brachiosauro]`



Accresci conoscenza



[[carnivoro] [[abbaia] cane tirannosauro] brachiosauro]]

Gioco di colori

```
to colori
ht
repeat 40 [setpc (list repcount*20 repcount repcount*255)
    spirale 2 repcount+90 ~
wait 40 home cs ~
setpc (list 255  repcount repcount*255) spirale 2 repcount+90 ~
wait 40 home cs ~
setpc (list repcount*20  repcount*255 repcount) spirale 2
    repcount+90 ~
wait 40 home cs ~
]
end
```

Alfabeto Morse

- A .-
- B -...
- C -.-.
- D -..
- E .
- F ..-.
- G --.
- H ....
- I ..
- J .---
- K -.-
- L .-..
- M --
- N -.
- O ---
- P .--.
- Q --.-
- R .-.
- S ...
- T -
- U ..-
- V ...-
- W .--
- X -..-
- Y -.--
- Z --..

La property list

- [italiano libro inglese book francese livre]
- [A [. _] B [_ . . .] O [_ _ _] S [. . .]]
- pprop "codice "a [. _]
- pprop "codice "b [_ . . .]
- gprop "codice "b

Costruzione del codice

```
make "alfabeto [a b c  s o ]
make "cod [[. _ ] [. . _ ] [ _ . _ ] [. . .] [ _ _ _ ] ]
to compongo :lista1 :lista2
if emptyp :lista1 [stop]
pprop "codice first :lista1 first
  :lista2
compongo bf :lista1 bf :lista2
end
compongo :alfabeto :cod
```

Le procedure “sonore”

```
to .
type ".
sound [440 20] ; frequenza e durata
wait 5 ; si aspetta per separare i suoni
end

to _
type "-
sound [440 300]; durata in clock ticks
wait 5 ; tempo in 1/60ths of a second.
end
```

trasmissione

```
to morse :car
run gprop "codice :car
end
;run esegue la lista che la segue
to trasm_parola :parola
if :parola = " [wait 10 stop]
morse first :parola
trasm_parola bf :parola
end
```

Trasmettiamo in Morse

```
to trasmetti :frase
if :frase = [][print " stop]
trasm_parola first :frase
type "
trasmetti butfirst :frase
end
trasmetti [sos sos]
```

Eventi tastiera

```
to bambini
keyboardon [esegui char keyboardvalue]
setfocus [MSWLogo Screen]
end
to esegui :tasto
if :tasto = "X [label "ok]
if :tasto = "c [ClearScreen]
if :tasto = "i [bk 10]
if :tasto = "s [lt 10]
if :tasto = "d [rt 10]
if :tasto = "a [fd 10]
if :tasto = "p [pd]
if :tasto = "n [pu]
end
```

L’animale tartaruga

```
to randomizza :x1 :x2  ; ipotesi x2 > x1
 op ((random (1+ abs ( :x2 - :x1 ))) + :x1 )
end
```

Domanda:

lt random 50 rt random 50 in cosa differisce da lt randomizza –50 50 ?

Il sensore di tocco

```
to hoilmuroavanti?
local "mypixel
make "mypixel pixel
for [passi 1 5] [ ~
     ht pu fd 1
     if not pixel = :mypixel ~
     [bk :passi st op "true ]
     ]
    bk 5 st op "false
end
```

L’urto con le pareti

```
to ronza
rt randomizza -10 5
wait 5
ifelse hoilmuroavanti? ~
[lt 180 ] [fd 5]
ronza
end
```

Il comportamento “riempie” lo spazio a disposizione

Un insetto

```
to ronza
rt randomizza -10 5
wait 5
ifelse hoilmuroavanti? ~
[raddrizza ] [fd 5]
ronza
end
to raddrizza
lt 1
ifelse hoilmuroavanti? ~
[raddrizza] [fd 5]
end
```

Altro esempio

```
to segna_percorso
pd setpc 4 setpensize [ 20 20 ]
rt 60 fd 60 rt 40 fd 120
rt 100 fd 150 rt 90 fd 120
rt 65 fd 80
end
```

Una "strada" su un burrone!
Costringiamo dentro la tartaruga

Metodologie interessanti

- to cambia_direzione
- bk 1 rt 30
- end

- to controlla
- pu fd 1
- output pixel
- end

- to cammina
- bk 1 pd fd 1
- end

Funziona?

- to segui_traccia
- setpc 0
- setpensize [1 1]
- repeat 10000 [
- ifelse or (4 = controlla) (0 = controlla)
- [cammina] [wait 5 cambia_direzione]
-]
- end

Errore

- No. La procedura controlla può esser chiamata due volte da segui_traccia: quando la tartaruga è sul bordo.

Una correzione

- `to controlla`
- `pu fd 1`
- `output or ( pixel = 4 ) ( pixel = 0)`
- `end`

- `Ancora un loop`

Un po' di fantasia

- `to cambia_direzione`
- `bk 1 rt random 50`
- `end`
- LEZIONE APPRESA
  - Scrivere bene i controlli
  - Un po' di fantasia (non determinismo) non guasta

Multi-agente

- Il concetto di comunicazione con altri agenti porta ai sistemi multi-agente
- I sistemi attualmente più diffusi sono
  - StarLogo
  - NetLogo

Uso finestre

- windowcreate "root "finestra "titolo 0 0 100 100 []
- windowcreate "finestra "finestramia "titolo 25 25 50 25 []
- staticcreate "finestramia "static1 [Heading=0] 25 25 50 25
- repeat 72 [rt 5 staticupdate "static1 se [Heading=] heading wait 60]
- windowdelete "finestra

Esempio

```
to usofinestre
local "wnx make "wnx 100 local "wny make "wny 90
local "marx make "marx 5 local "mary make "mary
  20
local "sizx make "sizx 10 local "sizy make "sizy
  10
windowcreate "main "earth [Saluto] 0 0 :wnx :wny
  []
buttoncreate "earth "bm "* :marx+:sizx*4
  :mary+:sizy*1 :sizx :sizy [galassia 50]
staticcreate "earth "bw [Hello World] 25 50 50 25
staticcreate "earth "bt bf time 25 5 50 25
end
windowdelete "earth
```