NetLogo 4.1.2

نرم افزار NetLogo

استاد راهنما:

جناب آقای مهندس یزدانجو

تهیه و تنظیم:

فاطمه علی اکبری

مینا زهرائی

.

فهرست مطالب

NetLogo چیست؟

NetLogo محیط مدل سازی قابل برنامه نویسی برای شبیه سازی پدیده های طبیعی و اجتماعی است.

این نرم افزار توسط Wilensky در سال ۱۹۹۹ نوشته شد و توسعه آن به صورت مداوم در مرکز ارتباط آموزش و مدل سازی کامپیوتری (the Center for Connected Learning and Computer-Based Modeling) ادامه داشته است.

ویژگی ها

NetLogo به خصوص برای مدل سازی سیستم های پیچیده در حال تغییر با زمان مناسب است.

مدل کنندگان می توانند دستورالعمل ها را به صدها یا هزاران "عامل" بدهند که هر کدام به طور مستقل عمل می کنند.

این توانایی امکان کشف ارتباط را بین رفتار هر فرد و الگوهای بوجود آمده در سطح کلان را که از تعامل افراد بسیار زیادی پدیدار شود فراهم می کند.

کاربری

NetLogo اجازه می دهد تا دانشجویان شبیه سازی ها را اجرا کنند و رفتار آنها را تحت شرایط مختلف بررسی کنند.

این نرم افزار همچنین دارای محیطی برنامه نویسی است که دانشجویان ، اساتید و برنامه نویسان را قادر می سازد تا مدل های خود را ایجاد کنند.

NetLogo به اندازه کافی ساده است تا دانشجویان بتوانند به آسانی مدلهای شبیه سازی شده آنرا اجرا کنند و یا حتی برای خود بنویسند. و ، اینکه به عنوان یک ابزار قدرتمند در زمینه های مختلف به اندازه کافی توسعه یافته است تا محققان با زمینه های کاری گوناگون از آن استفاده کنند.

مستندات

NetLogo مستندات و راهنماهای گسترده ای دارد.

همچنین دارای یک کتابخانه ی مدل است، که مجموعه بزرگی از مدلهای شبیه سازی شده ی از پیش نوشته شده را در بردارد.

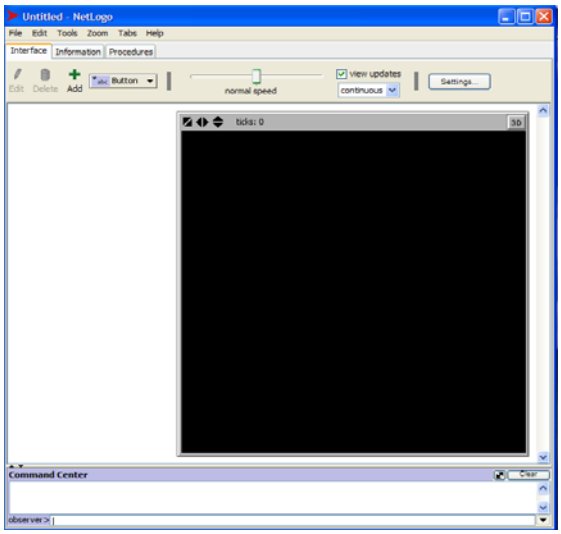
این مدلها می توانند به طور مستقیم مورد استفاده قرار گیرند و یا اصلاح و تغییر داده شوند.
این شبیه سازی ها زمینه های بسیاری از علوم طبیعی و اجتماعی، از جمله زیست شناسی و پزشکی ، فیزیک و شیمی ، ریاضیات و علوم کامپیوتر ، اقتصاد و روانشناسی اجتماعی را در بر می گیرد.

ویژگیهای نرم افزاری

NetLogo نسل بعدی از سری زبان های مدل سازی چند عامله است که با StarLogo آغاز شده است.
NetLogo بر روی ماشین مجازی جاوا اجرا می شود، پس در تمامی سیستم عاملها (مکینتاش ، ویندوز ، لینوکس ، و ...) کار می کند.
می توان آن را به عنوان یک برنامه مستقل اجرا کرد ، و یا از خط فرمان استفاده کرد .
NetLogo به صورت کاملا رایگان برای همه سیستم عاملها در دسترس است.

محیط نرم افزار NetLogo

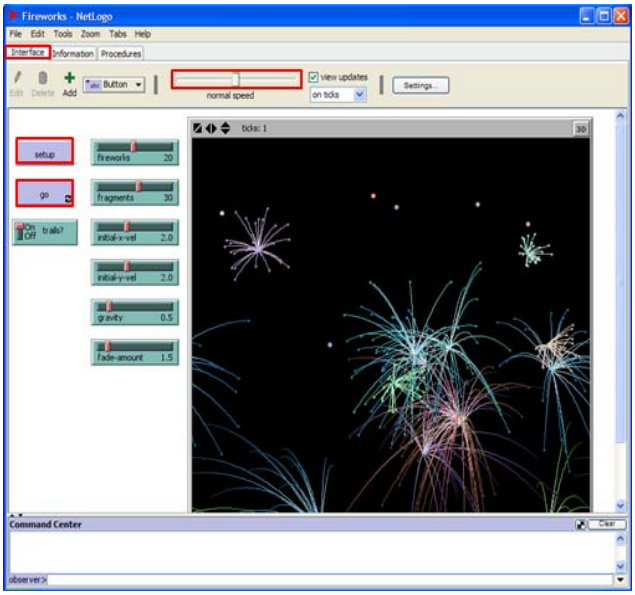
پس از نصب نرم افزار و اجرای آن ابتدا محیطی شبیه به شکل زیر مشاهده می گردد.



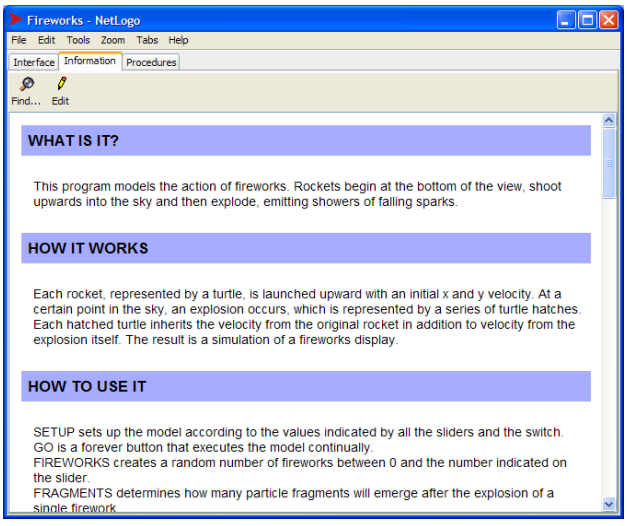
با انتخاب یک مدل ازپیش نوشته شده از منوی فایل، قسمت Models Library، پنجره ای شبیه پنجره سمت چپ پدیدار می شود که به شرح قسمت های مختلف آن می پردازیم:
در قسمت بالای پنجره سه سربرگ وجود دارد:

Interface : در این قسمت شکل مدل قابل مشاهده می باشد که شامل کلید ها و نوارهای لغزان زیر است:

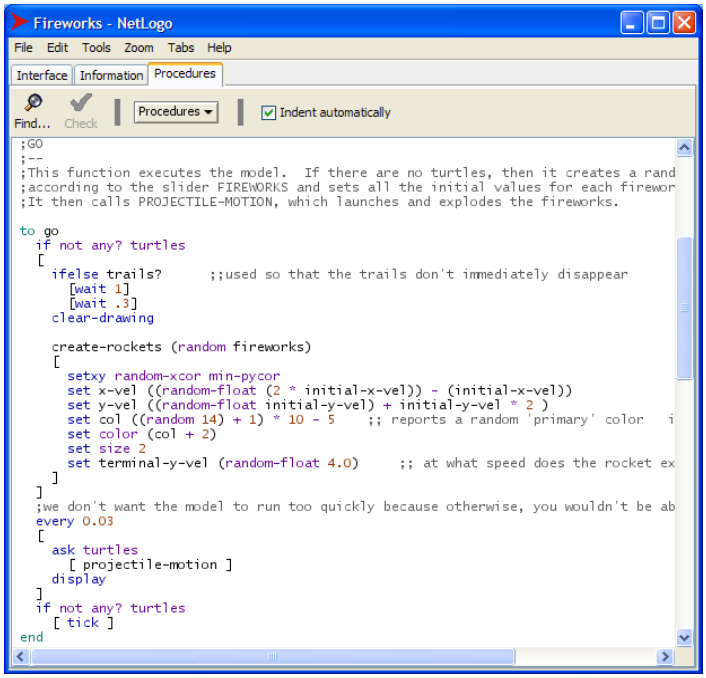
- Setup: با زدن این کلید تنظیمات اولیه مدل انجام می شود.
- Go: با زدن این کلید زمان شروع به حرکت می کند.
- normal speed: بوسیله این نوار لغزان می توان سرعت حرکت مدل را تنظیم کرد.
- به وسیله دیگر نوار های لغزان می توان پارامتر های مدل را تنظیم کرد.
- و ....



Information: این قسمت شامل اطلاعاتی راجع به مدل، اینکه مدل چگونه کار می کند و چگونگی استفاده از مدل، است.

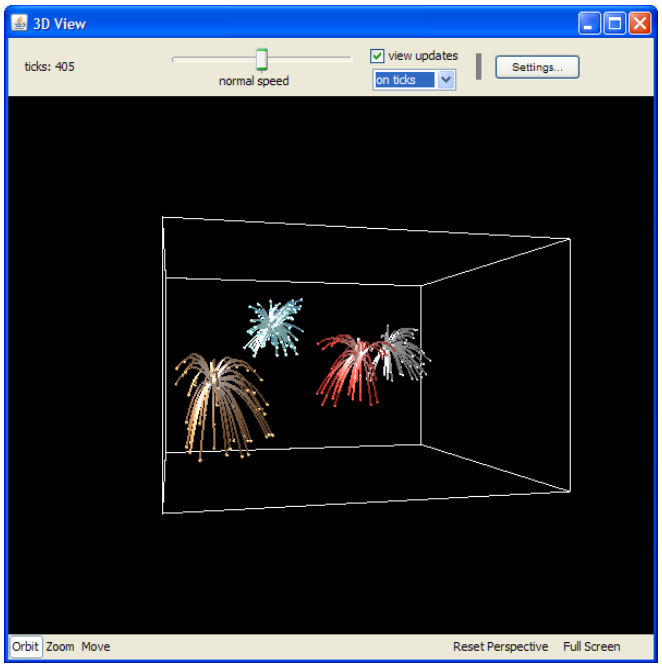


Procedure: در این قسمت کد مدل قابل مشاهده است که می توان این کد را بنا به دلخواه خود تغییر دهیم.



محیط سه بعدی

با نصب نسخه 3D نرم افزار می توان حالت سه بعدی برخی از مدل ها را مشاهده کرد.

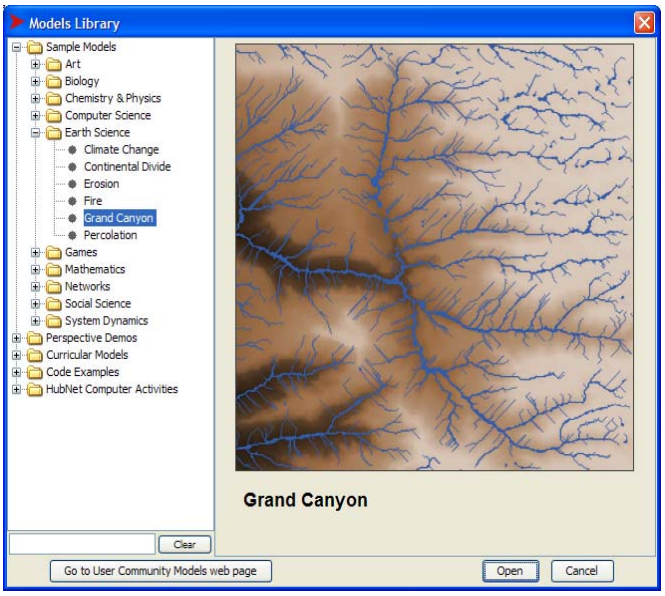


کتابخانه مدلها

همان گونه که قبلا نیز اشاره شد نرم افزار NetLogo دارای مجموعه بزرگی از مدلهای شبیه سازی شده ی از پیش نوشته شده را در بردارد که از منو File قابل دسترسی می باشد.

مدلهای نمونه

همین طور که در شکل زیر نیز قابل مشاهده است شبیه سازی ها، زمینه های بسیاری از جمله هنر، زیست شناسی و پزشکی ، فیزیک و شیمی ، ریاضیات و علوم کامپیوتر ، اقتصاد و روانشناسی اجتماعی را در بر می گیرد.



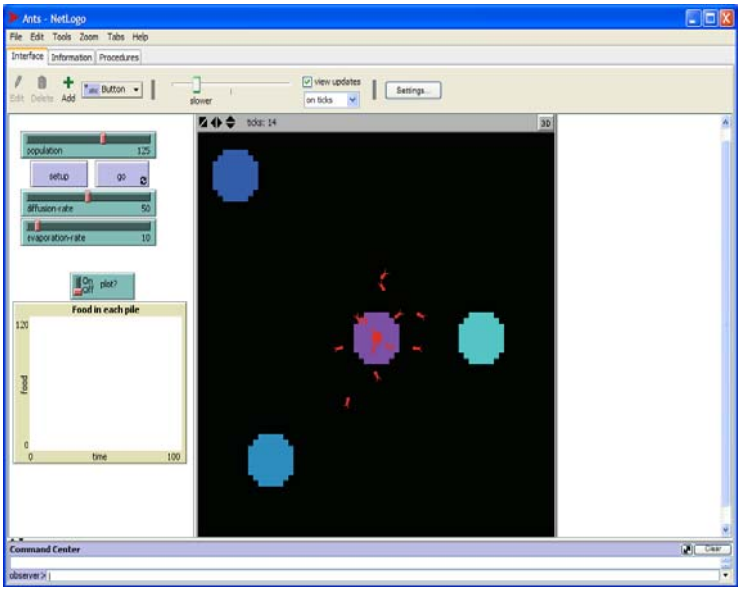
معرفی چند مدل شبیه سازی شده

مدل مورچه ها

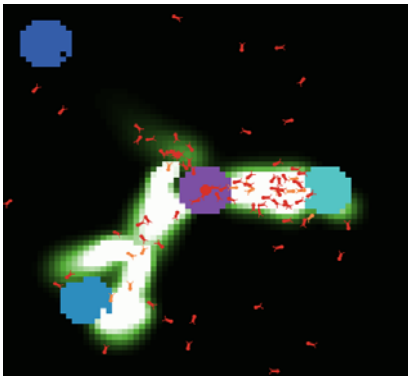
در این پروژه، رفتار یک کلنی از مورچه ها برای جمع آوری مواد غذایی شبیه سازی شده است.

اگرچه هر مورچه مجموعه ای از قوانین ساده را دنبال می کند، اما کلنی به عنوان مجموعه ای از مورچه ها به صورت پیچیده عمل می کند. وقتی که یک مورچه تکه ای از غذا را پیدا می کند، آن ماده غذایی را به لانه می برد، و هنگام حرکت یک ماده شیمیایی بر جای می گذارد.

وقتی که دیگر مورچه ها ماده شیمیایی را بو می کنند، آنها آنرا برای یافتن غذا دنبال می کنند. همانطور که مورچه های بیشتری مواد غذایی را به لانه می برند، مسیر ماده شیمیایی را تقویت می کنند.



با اجرای مدل(زدن کلیدsetup و سپسgo) به راحتی می توان رفتار یک کلونی از مورچه ها را در step های متوالی مشاهده کرد.



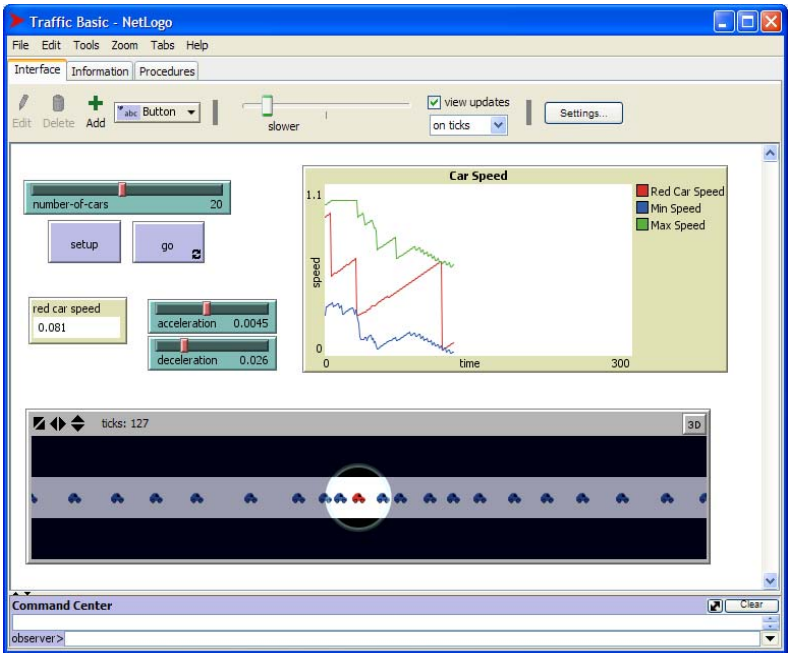
مدل ترافیک

این مدل حرکت اتومبیل ها را در یک بزرگراه مدل می کند.

هر یک از اتومبیل ها مجموعه ای از قوانین ساده ای را دنبال می کند:

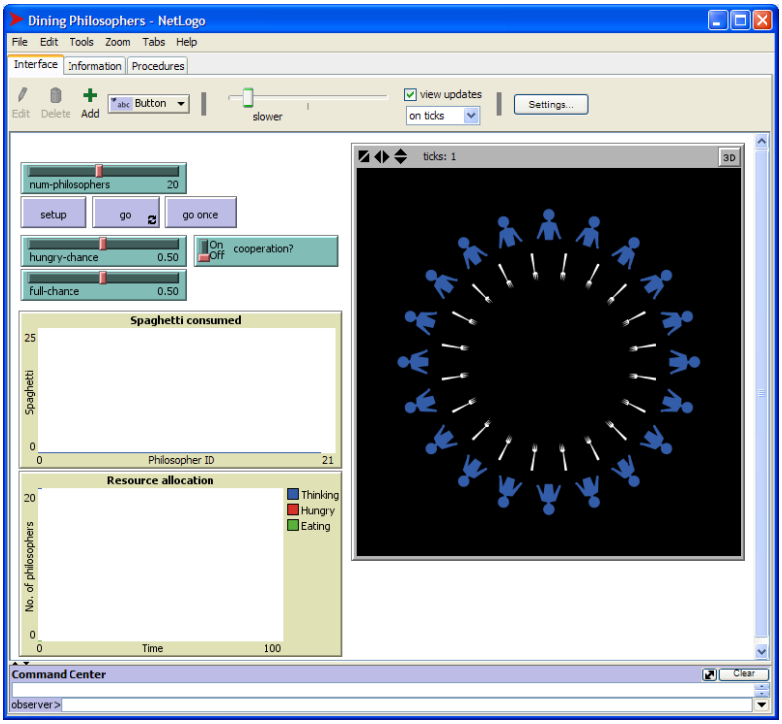
سرعت خود را کاهش می دهد اگر یک اتومبیل مقابل خود در فاصله نزدیک مشاهده کند و سرعت خود را افـزایش می دهد اگر اتومبیلی در مقابل خود نبیند.

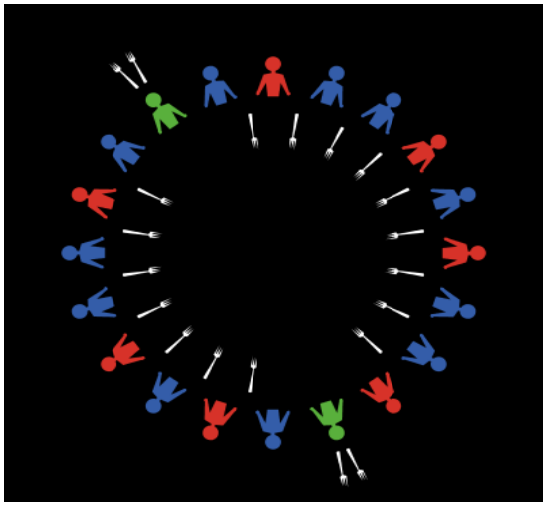
این مدل نشان می دهد که ترافیک می تواند حتی بدون هیچ گونه تصادف، خرابی پل، یا از کار افتادگی کامیون هـم بوجود آید.



مدل غذاخوردن فیلسوفها

در این مدل یکی از مسائل علوم کامپیوتر شبیه سازی شده و روند عملکرد هر فیلسوف بـه راحتـی قابـل مشـاهده مـی باشد.





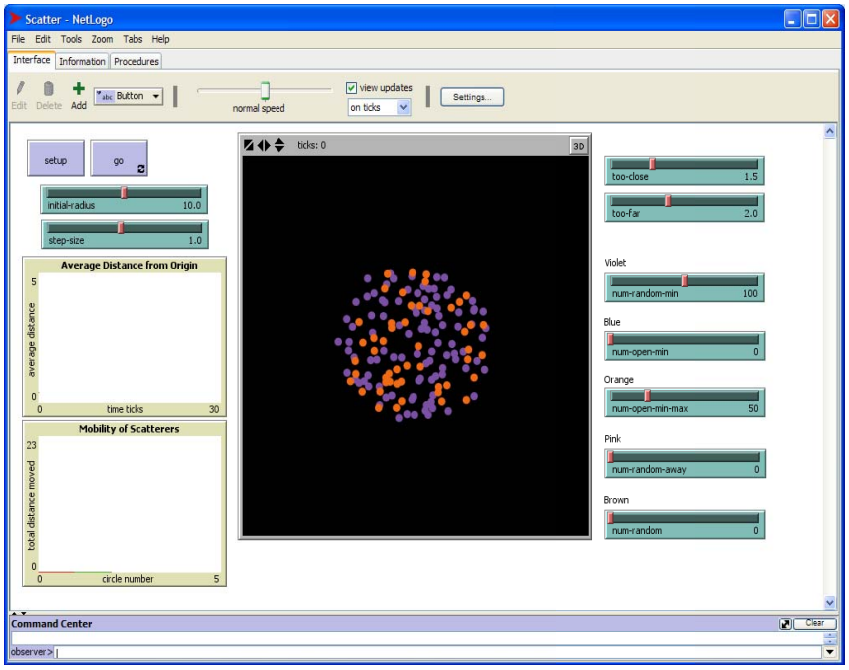
مدل پراکنده کردن

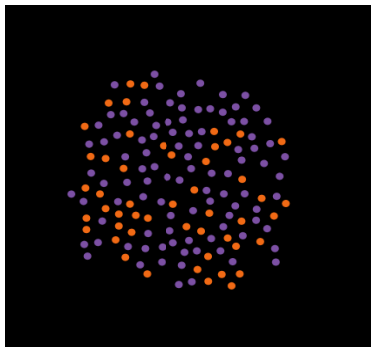
این مدل چندین قانون برای پراکندگی اعمال می کند:

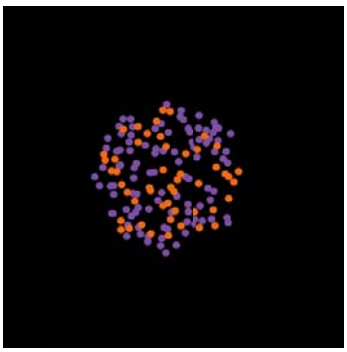
RANDOM-MIN: در یک جهت تصادفی حرکت کن تا به اندازه کافی از همه همسایگان خود دور شوی. (دستورالعمل نقاط بنفش)

OPEN-MIN: به بزرگترین فضای خالی اطراف حرکت کن تا زمانی که به اندازه کافی از همه همسایگان خود دور شوی. (دستورالعمل نقاط آبی)

OPEN-MIN-MAX: به بزرگترین فضای خالی اطراف حرکت کن تا زمانی که به اندازه کافی از همه همسایگان خود دور شوی ، اما نه خیلی دور. اگر خیلی نزدیک بودی، دور شو. اگر خیلی دور بودی، نزدیک شو. (دستورالعمل نقاط نارنجی)







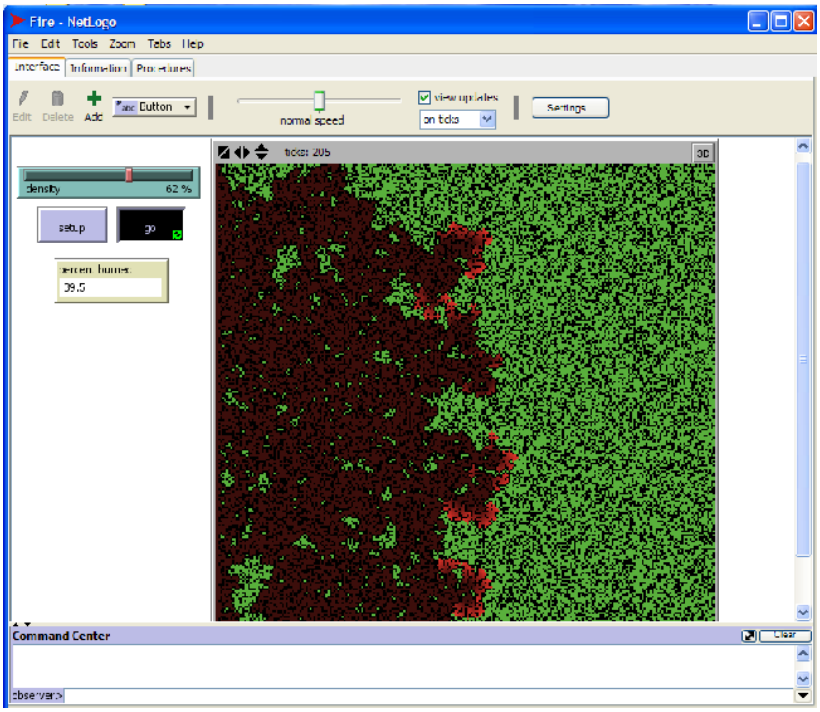
برنامه نویسی در NetLogo

مقدمه

جهان NetLogo از عوامل ساخته شده است. عواملی وجود دارند که می توانند دستورالعمل ها را دنبال کنند.همه عوامل به طور همزمان، قادر به انجام فعالیت خود هستند.

در اینجا به بررسی و تعریف برخی از پارامترها و دستورات مهم NetLogo می پردازیم. با توجه به Sample مدل آتش سوزی

مدل Fire (آتش سوزی)



این پروژه گسترش آتش سوزی از طریق یک جنگل را شبیه سازی می کند. این نشان می دهد که احتمال رسیدن آتش سوزی به لبه سمت راست جنگل ها به تراکم درختان بستگی دارد.

چگونه کار می کند؟

آتش سوزی در لبه سمت چپ جنگل شروع می شود، و به درخت های همسایه گسترش می یابد. آتش در چهار جهت شمال، شرق، جنوب، و غرب گسترش می یابد.

این مدل فرض می کند که باد وجود ندارد. بنابراین، آتش سوزی در جهتی که درختان بیشتری دارد پیشرفت می کند. آتش نمی تواند به یک منطقه غیر جنگل برود (patch)، پس یک patch حرکت آتش در این مسیر را مسدود می کند.

هر turtle که نشانگرِ یک قطعه ای از آتش است تولید می شود و سپس بدون حرکت ازبین می رود. اگر آتش از turtle ها ساخته شده است اما هیچ turtle در حال حرکت نیست پس چرا می گویند آتش حرکت می کند؟ این یک نمونه از سطوح مختلف در یک سیستم است: در سطح turtle های فردی، هیچ حرکتی وجود ندارد، اما در سطح turtle های جمعی در طول زمان، آتش حرکت می کند.

ویژگی NETLOGO

درخت نسوخته توسط patch های سبز و سوزاندن درختان توسط turtle نشان داده می شود. دو نژاد(breed) از turtle ها استفاده می شود ، "fires" و "embers". هنگامی که یک درخت آتش می گیرد، یک turtle آتش جدید ایجاد میشود، در نوبت بعدی یک آتش(fires) به خاکستر گرم(embers) تبدیل می شود. توجه کنید که چگونه برنامه به تدریج رنگ embers را برای رسیدن به اثر بصری سوختن تیره می کند.

"neighbors4"ابتدایی برای گسترش آتش استفاده می شود.

برنامه :

```
globals [
  initial-trees   ;; how many trees (green patches) we started with
  burned-trees    ;; how many have burned so far
]
```

در این برنامه initial-trees نشان دهنده وضعیت درختان (patche های سبز) در آغازاست و burned-trees نشان دهنده وضعیت درختان سوخته شده تا کنون است.

```
breed [fires fire]   ;; bright red turtles -- the leading edge of the fire
breed [embers ember] ;; turtles gradually fading from red to near black
```

turtle ها به رنگ قرمز روشن ــ محدوده لبه آتش ؛ turtle هایی که به تدریج از قرمز به سیاه محو شدن.

```
to setup
  clear-all
  set-default-shape turtles "square"
  ;; make some green trees
```

ایجاد برخی از درختان سبز

```
  ask patches with [(random-float 100) < density]
    [ set pcolor green ]
  ;; make a column of burning trees
```

ایجاد یک ستون از درختان سوزانده شده

```
  ask patches with [pxcor = min-pxcor]
    [ ignite ]
  ;; set tree counts
```

مجموعه ای که درختان را می شمارد

```
  set initial-trees count patches with [pcolor = green]
  set burned-trees 0
end
to go
  if not any? turtles  ;; either fires or embers
    [ stop ]
  ask fires
    [ ask neighbors4 with [pcolor = green]
       [ ignite ]
      set breed embers ]
  fade-embers
  tick
end
;; creates the fire turtles
to ignite  ;; patch procedure
  sprout-fires 1
    [ set color red ]
 set pcolor black
  set burned-trees burned-trees + 1
end
;; achieve fading color effect for the fire as it burns
to fade-embers
  ask embers
    [ set color color - 0.3  ;; make red darker
```

ایجاد تیره قرمز

```
      if color < red - 3.5     ;; are we almost at black?
```

؛ ما تقریبا در سیاه و سفید؟

```
        [ set pcolor color
          die ] ]
end
```

معرفی برخی از دستورات

globals [var1 ...]

این کلمه کلیدی ، مانند کلمات کلیدی turtles-own , patches-own, <*breeds*>-own, breed ، تنها می تواند در آغاز یک برنامه قبل از تعریف هر تابع استفاده شود. این متغیر جدید سراسری تعریف می کند. متغیر های سراسری "global" هستند زیرا آنها توسط تمام عوامل در دسترس هستند و می تواند در هر نقطه از یک مدل استفاده می شود. اغلب ، GLOBALS برای تعریف متغیرها یا ثابت هایی که باید در بسیاری ازقسمت های برنامه استفاده شود، استفاده می شود.

breed [<breeds> <breed>]

این کلمه کلیدی ، مانند کلمات کلیدی GLOBALS، turtles-own, patches-own ، می تواند تنها در آغاز توابع، قبل از تعریف هرتابع، استفاده شود. اولین ورودی نام agentset را در ارتباط با breed تعریف می کند. دومین ورودی نام یک عضو منفرد از breed را تعریف می کند.

هر turtle از breed ارائه شده :

بخشی از agentset نامگذاری شده توسط نام breed

breed در مجموعه ای از متغیر برای agentset ساخته شده است

در اکثر موارد ، این agentset در ارتباط با درخواست برای دادن دستورات به تنها turtleهای یک breed خاص استفاده می شود.

```
breed [mice mouse]
breed [frogs frog]
to setup
  clear-all
  create-mice 50
  ask mice [ set color white ]
  create-frogs 50
  ask frogs [ set color green ]
  show [breed] of one-of mice    ;; prints mice
  show [breed] of one-of frogs   ;; prints frogs
end

show mouse 1 ;; prints (mouse 1)
show frog 51;; prints (frog 51)
show turtle 51;; prints (frog 51)
```

to procedure-name [input1 ...]

برای شروع یک دستور تابع استفاده می شود. یعنی هر تابع بین لغات To و End قرار می گیرد. مانند توابع زیر :

```
to setup
  clear-all
  crt 500
end

to circle [radius]
  crt 100 [ fd radius ]
end
```

end

برای پایان یک تابع استفاده می شود.

clear-all

تمام متغیر های جهانی را به صفر reset می کند ، وclear-turtles, clear-patches, , clear-drawing ، reset-ticks clear-all-plots و clear-output نامیده می شود .

set variable value

متغیر را با مقدار داده شده تنظیم می کنیم.

متغیر می تواند هر یک از موارد زیر باشد:

یک متغیر جهانی را با استفاده از "GLOBALS"اعلام کرد

متغیر جهانی با یک نوار لغزنده ، سوئیچ ، انتخاب ، و یا جعبه ورودی همراه می شود.

متغیر متعلق به این عامل

اگر این عامل است که یک turtle ، یک متغیر متعلق به patch زیر turtle .

متغیر محلی ایجاد شده توسط دستور let.

یک ورودی به روش جاری.

set-default-shape
set-default-shape turtles string
set-default-shape breed string

یک شکل پیش فرض اولیه برای همه turtle ها ، و یا یک نژاد(breed) خاص را مشخص می کند .هنگامی که یک turtle ایجاد می شود ، یا آن breed راتغییر می دهد، این شکل برای شکل داده شده تنظیم شده است. این دستور بر روی turtle های موجود تاثیر نمی گذاردبلکه تنها turtle ها را بعد از آن ایجاد می کند.

breed مشخص یا باید به صورت turtle و یا یک breed تعریف شده توسط کلمه کلیدی breed، و رشته مشخص باید به نام یک شکل در حال حاضر تعریف شود.

در مدل های جدید ، شکل پیش فرض برای همه turtle ها "default " است.

```
create-turtles 1 ;; new turtle's shape is "default"
create-cats 1    ;; new turtle's shape is "default"

set-default-shape turtles "circle"
create-turtles 1 ;; new turtle's shape is "circle"
create-cats 1    ;; new turtle's shape is "circle"

set-default-shape cats "cat"
set-default-shape dogs "dog"
create-cats 1   ;; new turtle's shape is "cat"
ask cats [ set breed dogs ]
  ;; all cats become dogs, and automatically
  ;; change their shape to "dog"
```

setxy x y

turtle،x-coordinate را با X و y-coordinate را باyتنظیم می کند.

معادلی برای set xcor x set ycor y ، به جز آن را در یک زمان به جای دو مرحله اتفاق می افتد.

اگر x با y و یا خارج از جهان ، NetLogoیک خطای زمان اجرا خواهد داشت .

```
setxy 0 0
;; turtle moves to the middle of the center patch
setxy random-xcor random-ycor
;; turtle moves to a random point
setxy random-pxcor random-pycor
;; turtle moves to the center of a random patch
```

set-current-plot plotname

طرح در حال حاضر را به طرح با نام داده شده (یک رشته) تنظیم می کند. طرح دستورات بعدی روی طرح فعلی تاثیر خواهد داشت.

set-plot-x-range set-plot-y-range

مقادیر حداقل و حداکثر محور x یا y طرح فعلی را تنظیم می کند.

این تغییر موقتی است و با مدل ذخیره نمی شود. هنگامی که طرح پاکسازی می شود، رشته برای مقادیر پیش فرض خود به عنوان گفت و گو ویرایش طرح تنظیم می گردد.

set-current-plot-pen penname

قلم طرح در حال حاضر برای penname قلم به نام (رشته) تنظیم می شود .اگر قلمی در طرح حاضر وجود ندارد ، یک خطای زمان اجرا رخ می دهد.

ask agentset [commands]

عامل یا agentset مشخص شده دستورات داده شده را اجرا می کند.

```
ask turtles [ fd 1 ]
  ;; all turtles move forward one step
ask patches [ set pcolor red ]
```

```
  ;; all patches turn red
ask turtle 4 [ rt 90 ]
  ;; only the turtle with id 4 turns right
```

توجه : تنها observer(ناظر) می تواند از همه turtle ها یا همه patche ها سوال کند.

نکته : فقط عواملی که در agentset هستند در زمان درخواست ها اجرای دستورات آغاز می شود.

turtles

agentsetمتشکل از تمام turtleها را شرح می دهد.

```
show count turtles
;; prints the number of turtles
```

patches

Agentset متشکل از تمام patches را شرح می دهد.

random-float number

اگر عدد مثبت است ، عدد ممیز شناور تصادفی بزرگتر یا برابر با ۰ است اما به شدت کوچکتر ازاعداد را شرح می دهد.

اگر عدد منفی است ، عدد ممیز شناور تصادفی کوچکتر یا برابر با ۰ ،اما به شدت بزرگتر ازاعداد را شرح می دهد .

اگر عدد صفر است ، نتیجه همیشه ۰ است.

```
show random-float 3
;; prints a number at least 0 but less than 3,
;; for example 2.589444906014774
show random-float 2.5
;; prints a number at least 0 but less than 2.5,
;; for example 1.0897423196760796
```

color

این یک متغیر turtle یا link ساخته شده است. آن رنگ turtle یا link را نگداری می کند. شما می توانید این متغیر را برای ایجاد تغییر رنگ turtle یا link تنظیم کنید. رنگ می تواند یا یک رنگ NetLogo (یک عدد) ، یا یک رنگ RGB (یک لیست از ۳ عدد) باشد.

pcolor

این یک متغیر patch ساخته شده است. آن رنگ patch را نگه می دارد.شما می توانید این متغیر را برای ایجاد تغییر رنگ patch تنظیم کنید.

همه متغیرهای patch می تواند به طور مستقیم توسط هر turtle مقرر در patch قابل دسترس باشد. رنگ می تواند یا یک رنگ NetLogo (یک عدد) و یا یک رنگ RGB (یک لیست از ۳ شماره) باشد.

xcor

این یک متغیر turtle ساخته شده است. این مختصات X در حال حاضر از turtle را نگهداری می کند. شما می توانید این متغیر را برای تغییر محل turtle تنظیم کنید

این متغیر همیشه بزرگتر یا مساوی با (min-pxcor - 0.5)و به شدت کمتر از .(max-pxcor + 0.5) است .

ycor

این یک متغیر turtle ساخته شده است. این مختصات y در حال حاضر از turtle را نگهداری می کند. شما می توانید این متغیر را برای تغییر محل turtle تنظیم کنید.

این متغیر همیشه بزرگتر یا مساوی با (min-pycor - 0.5)و به شدت کمتر از .(max-pycor + 0.5) است.

pycor pxcor

اینها متغیرهای patch ساخته شده هستند. آنها مختصات X و Y، patch را نگه میدارند.آنها معمولا اعداد صحیح هستند.شما نمی توانید متغیرهایشان را تنظیم کنید زیرا patch ها حرکت نمی کنند.

Pxcorبزرگتر یا مساوی min-pxcor و کوچکتر یا مساوی max-pxcor است ؛ همچنین Pycor بزرگتر یا مساوی min-pycor و کوچکتر یا مساوی max-pycor است .

همه متغیرهای patch می تواند به طور مستقیم توسط هر turtle مقرر در patch قابل دسترس باشد.

min-pycor

این گزارشات حداقل مختصات x و حداقل مختصات Y را ، (به ترتیب) برای patche ، که تعیین اندازه جهان است می دهد.

بر خلاف نسخه های قدیمی تر از NetLogo منشا را برای مرکز جهان ندارد. با این حال ، مختصات حداقل x و y ، باید کمتر یا برابر با صفر باشد.

توجه : شما می توانید اندازه جهان را تنها با ویرایش تنظیم کنید، اینها آنچه نمی تواند تنظبم شود را شرح می دهند .

```
crt 100 [ setxy random-float min-pxcor
                random-float min-pycor ]
;; distributes 100 turtles randomly in the
;; third quadrant
```

count agentset

تعداد عوامل داده شده در.agentset را نشان می دهد .

```
show count turtles
;; prints the total number of turtles
show count patches with [pcolor = red]
;; prints the total number of red patches
```

if condition [commands]

گزارشگر باید ارزش یک مقدار Boolean(درست یا غلط) را گزارش دهد.

اگر شرط درست باشد ، دستورات را اجرا می کند.

خبرنگار ممکن است یک مقدار متفاوت برای عوامل مختلف را گزارش دهد، به طوری که برخی از عوامل ممکن است دستورات را اجرا کنند وبقیه اجرا نشود.

```
if xcor > 0[ set color blue ]
;; turtles in the right half of the world
;; turn blue
```

ifelse reporter [commands1] [commands2]

reporter باید یک ارزش مقدار بولی (درست یا غلط) را گزارش دهد.

اگر reporter درست گزارش بدهد ، commands1اجرا می شود.

اگر reporter غلط گزارش بدهد ، commands2 اجرا می شود.

reporter ممکن است یک مقدارمتفاوت برای عوامل مختلف گزارش دهد، به طوری که برخی از عوامل ممکن است commands1 را اجرا کنند در حالی که بقیه commands2 را اجرا می کنند.

```
ask patches
  [ ifelse pxcor > 0
      [ set pcolor blue ]
      [ set pcolor red ] ]
;; the left half of the world turns red and
;; the right half turns blue
```

stop

این عامل بلافاصله ازتابع خارج میشود.البته فقط روند تابع کنونی متوقف می شود، نه اجرای همه عوامل.

```
if not any? turtles [ stop ]
;; exits if there are no more turtles
```

توجه : stop می تواند برای توقف یک دکمه برای همیشه استفاده شود . اگر دکمه برای همیشه به طور مستقیم یک تابع را فراخوانی کند ، سپس وقتی که تابع متوقف می شود ، دکمه متوقف می شود. (در یک turtle و یا patch برای همیشه ، دکمه متوقف نخواهد شد تا هر turtle و یا patch متوقف شود – یک turtle یا patch تک قدرتی برای جلوگیری از اتمام دکمه ندارد.)

not boolean

اگر *boolean* نادرست است درست گزارش می دهد ، در غیر این صورت اشتباه گزارش می دهد.

```
if not any? turtles [ crt 10 ]
```

neighbors

گزارش یک agentset شامل ۸ ، patche اطراف آن (neighbors) و یا ۴ تا patche اطراف آن ،(neighbors4)

```
show sum [count turtles-here] of neighbors
  ;; prints the total number of turtles on the eight
  ;; patches around this turtle or patch
show count turtles-on neighbors
  ;; a shorter way to say the same thing
```

```
ask neighbors4 [ set pcolor red ]
  ;; turns the four neighboring patches red
```

tick

شمارنده tickتوسط یکی پیشرفت می کند.

die

turtle و یا link از بین می رود.

```
if xcor > 20 [ die ]
;; all turtles with xcor greater than 20 die
ask links with [color = blue] [ die ]
;; all the blue links will die
```

size

این یک متغیر turtle ساخته شده است. این دارای یک شماره را که اندازه ظاهری turtle است نگهداری می کند. اندازه پیش فرض ۱ است ، به این معنی که turtle به همان اندازه یک patch است. شما می توانید این متغیر را برای تغییر اندازه turtle تنظیم کنید.

jump number

turtle توسط واحد اعداد همه در یک بار (کمی از یک گام در یک زمان با فرمان forward) پیش می رود.

اگر turtle به واحد تعداد نمی تواند بپرد به این دلیل است که توپولوژی در حال حاضر مجاز نیست turtle نمیتواند در همه حرکت کند.

forward **number**

turtle توسط تعداد مراحل ، یک گام در یک زمان پیش می رود. (اگر عدد منفی است ، turtle رو به عقب حرکت می کند)

FD 10 معادل با.[jump 1] repeat 10 است .و .[معادل با.jump 0.5 [jump 1] repeat 10 است.

اگر turtle نمی تواند تعداد گام رو به جلو را حرکت کند به این دلیل است که توپولوژی در حال حاضر مجاز نیست تا هنگامیکه گامهای ۱ می تواند کامل شود کامل خواهد شد ، پس از آن متوقف می شود.

right number

لاک پشت (turtle) توسط عدد درجه به راست می چرخد. (اگر عدد منفی است ، آن را به سمت چپ می چرخاند.)

heading

این یک متغیر turtle ساخته شده است. این جهت turtle یی که با آن مواجه است را نشان می دهد. این یک عدد بزرگتر یا مساوی با ۰ و کمتر از ۳۶۰ است. ۰ شمال است ، ۹۰ شرق است ، و غیره. شما می توانید این متغیر را برای ایجاد چرخش turtle تنظیم کنید. به عنوان مثال:

```
set heading 45      ;; turtle is now facing northeast
set heading heading + 10 ;; same effect as "rt 10"
```

who

این یک متغیر turtle ساخته شده است. این "who number" و ID number ی turtle ، یک عدد بزرگتر یا مساوی با ۰ را نگهداری می کند. شما نمی توانید این متغیر را تنظیم کنید؛ یک who number ،turtle هرگز تغییر نمی کند.

who number از ۰ شروع می شود . یک عدد turtle از بین رفته به یک turtleجدید نسبت داده نمی شود زمانیکه شما دستورات clear-turtles یا clear-all را استفاده کنید تا زمانیکه شماره دهی Whoدوباره از ۰ آغاز شود .

به عنوان مثال:

```
show [who] of turtles with [color = red]
;; prints a list of the who numbers of all red turtles
;; in the Command Center, in random order
crt 100
  [ ifelse who < 50
      [ set color red ]
      [ set color blue ] ]
;; turtles 0 through 49 are red, turtles 50
;; through 99 are blue
```

one-of agentset

از یک gentset، یک Agentتصادفی را میدهد . اگر gentset خالیست ، گزارش nobody را می دهد.

از یک لیست ، یک item listتصادفی را می دهد . اگر یک خطا برای لیست هست خالی می شود.

```
ask one-of patches [ set pcolor green ]
;; a random patch turns green
ask patches with [any? turtles-here]
  [ show one-of turtles-here ]
;; for each patch containing turtles, prints one of
;; those turtles

;; suppose mylist is [1 2 3 4 5 6]
show one-of mylist
;; prints a value randomly chosen from the list
```

[reporter] of agentset [reporter] of agent of

برای یک عامل ، ارزش reporter برای عامل (turtle یاpatch) را گزارش می دهد.

```
show [pxcor] of patch 3 5
;; prints 3
show [pxcor] of one-of patches
;; prints the value of a random patch's pxcor variable
show [who * who] of turtle 5
=> 25
show [count turtles in-radius 3] of patch 0 0
;; prints the number of turtles located within a
;; three-patch radius of the origin
```

```
برای agentset ، یک لیست که حاوی ارزشreporter برای هر عامل در agentset (در صورت
تصادفی) گزارش میدهد.
crt 4
show sort [who] of turtles
=> [0 1 2 3]
show sort [who * who] of turtles
=> [0 1 4 9]
```

condition1 and condition2

اگر هر دو شرط *condition1* و condition2 درست است گزارش درست می دهد.

توجه داشته باشید که اگر condition1 نادرست است ، پس از آن condition2، اجرا نمی شود.

```
if (pxcor > 0) and (pycor > 0)
  [ set pcolor blue ]  ;; the upper-right quadrant of
                       ;; patches turn blue
```

boolean1 or boolean2

اگر یا boolean1 یا boolean2 یا هر دو درست است، گزارش درست می دهد.

توجه داشته باشید که اگر condition1 درست است ، پس از آن condition2، اجرا نخواهد شد.

```
if (pxcor > 0) or (pycor > 0) [ set pcolor red ]
;; patches turn red except in lower-left quadrant
```

myself

" self" و "myself " بسیار متفاوت هستند. " self" ساده است ، آن معنی "من ." میدهد. "myself " به معنای "turtle و یا patch است که که از من برای انجام کار من در حال حاضر خواسته است ."

زمانی که یک agent خواسته شده بود تا برخی از کد اجراشود ، استفاده "myself " در آن کد عامل (turtle و یا patch) را گزارش میدهد که درخواست شده است.

"myself " اغلب در ارتباط با of برای خواندن و یا تنظیم متغیرها در درخواست عامل استفاده می شود.

"myself " می تواند

در بلوک کد نه تنها در دستورات درخواست استفاده شود ، و همچنین hatch, sprout, of, with, all?, with-min, with-max, min-one-of, max-one-of, min-n-of, max-n-of.

```
ask turtles
  [ ask patches in-radius 3
      [ set pcolor [color] of myself ] ]
;; each turtle makes a colored "splotch" around itself
```

move-to agent

turtle مختصات Xو Y خود را برای همان عامل داده شده تنظیم می کند.(اگر آن عامل یک patch است ، اثری برای حرکت turtle به مرکز patchاست.)

```
move-to turtle 5
;; turtle moves to same point as turtle 5
move-to one-of patches
;; turtle moves to the center of a random patch
move-to max-one-of turtles [size]
;; turtle moves to same point as biggest turtle
```

توجه داشته باشید که عنوان turtle بدون تغییر است و. شما ممکن است استفاده اولین دستور face برای جهت دهی turtle در جهت حرکت را بخواهید.

to-report procedure-name [input1 ...]

برای شروع یک تابع reporter استفاده می شود.

بدنه تابع باید report را برای گزارش یک مقدار برای تابع استفاده کند.

```
to-report average [a b]
  report (a + b) / 2
end

to-report absolute-value [number]
  ifelse number >= 0
    [ report number ]
    [ report (- number) ]
end

to-report first-turtle?
  report who = 0  ;; reports true or false
end
```

report value

بلافاصله از تابع to-report و ارزش report فعلی به عنوان نتیجه ی این تابع خارج می شود. report و to-report همواره در ارتباط با یکدیگر استفاده می شود

myself

" self" و "myself " بسیار متفاوت هستند. " self" ساده است ، آن معنی "من ." میدهد. "myself " به معنای "turtle و یا patch است که که از من برای انجام کار من در حال حاضر خواسته است ."

زمانی که یک agent خواسته شده بود تا برخی از کد اجراشود ، استفاده "myself " در آن کد عامل (turtle و یا patch) را گزارش میدهد که درخواست شده است.

"myself " اغلب در ارتباط با of برای خواندن و یا تنظیم متغیرها در درخواست عامل استفاده می شود.

"myself " می تواند

در بلوک کد نه تنها در دستورات درخواست استفاده شود ، و همچنین hatch, sprout, of, with, all?, with-min, with-max, min-one-of, max-one-of, min-n-of, max-n-of.

```
 ask turtles
  [ ask patches in-radius 3
      [ set pcolor [color] of myself ] ]
;; each turtle makes a colored "splotch" around itself
```

self

گزارش این turtle و یا patch.

" self" و "myself " بسیار متفاوت هستند. " self" ساده است ، آن معنی "من ." میدهد. "myself " به معنای "turtle و یا patch است که که از من برای انجام کار من در حال حاضر خواسته است ."

plot number

x-value قلم طرح شده توسط پلات قلم فاصله ، آنگاه نمودار یک نقطه به روز شده در x-value و y-value شماره افزایش می یابد. (اولین بار دستور روی یک طرح استفاده می شود ، یک نقطهx-value از ۰ رسم می شود.)

plot-pen-reset

همه چیز که قلم طرح فعلی کشیده است راپاک می کند، آن را به (۰،۰) حرکت می کند ، و آن را پایین قرار می دهد. اگر قلم، قلم دائمی است، رنگ و حالت برای ارزش های متفاوت از طرح گفت و گو ویرایش تنظیم مجدد می شود.

create-turtles
crt
create-<*breeds*>
create-turtles number
create-turtles number [commands]
create-<breeds> number
create-<breeds> number [commands]

تعداد turtle های جدید در مبدا ایجاد می شود. turtle های جدید عنوان عدد صحیح تصادفی و رنگ به طور تصادفی از رنگ های اصلی ۱۴انتخاب شده است.

اگر create-<*breeds*> فرم استفاده می شود ، turtle های جدید ، به عنوان اعضای breedداده شده ایجاد می شود.

اگر دستورات ارائه شده ، turtle های جدید ، بلافاصله آنها را اجرا کنید. این مفید است برای دادن turtle های جدید رنگ های مختلف ، عنوان ، یا هر چیز. (turtle های جدید همه در یک بار ایجاد و سپس در یک زمان اجرا ، به منظور تصادفی است.)

```
crt 100 [ fd 10 ]     ;; makes a randomly spaced circle
breed [canaries canary]
breed [snakes snake]
to setup
  clear-all
  create-canaries 50 [ set color yellow ]
  create-snakes 50 [ set color green ]
end
```

patches-own

patche patche ، خود ، خود را [...var1] این کلمه کلیدی ، مانند GLOBALS ، breed، ، خود ، و turtle ها ، کلمات کلیدی خود ، تنها می تواند در آغاز یک برنامه استفاده شود ، قبل از تعریف تابع هر. این تعریف متغیر است که همه patche می توانید استفاده کنید. همه patche خواهد شد و سپس متغیر داده و قادر به استفاده از آنها. همه متغیرها patch همچنین می توانید به طور مستقیم توسط هر turtle در ایستاده patch قابل دسترسی است.

<breeds>-own

turtle ها ، خود [...var1] ، خود [...var1] کلمه کلیدی خود ، turtle ، مانند GLOBALS ، breed، ، خود ، و patche های خود ، کلمات کلیدی ، می تواند تنها در ابتدای برنامه استفاده شود ، قبل از تعریف تابع هر. این تعریف متغیرهای متعلق به هر یک از turtle . اگر یک breedبه جای "turtle " مشخص می کنید ، turtle ها تنها از آن breedاز متغیرهای ذکر شده است. (بیش از یک breedturtle ممکن است متغیر باشد. لیست(

```
breed [cats cat ]
breed [dogs dog]
breed [hamsters hamster]
turtles-own [eyes legs]   ;; applies to all breeds
cats-own [fur kittens]
hamsters-own [fur cage]
dogs-own [hair puppies]
```

برای دستیابی به اطلاعات بیشتر به Programming Guide در راهنمای نرم افزار NetLogo مراجعه کنید.