آشنایی با نرم افزار نت لوگو

علی اکبر براتی

NetLogo: Version 4.1.1

(August 2010)

http://ccl.northwestern.edu/netlogo/

بخش اول : آشنایی با نرم افزار

Netlogo یک نرم افزار مدل سازی برای شبیه سازی پدیده های مختلف است. این نرم افزار توسط Uri Wilensky در سال ۱۹۹۹ معرفی شد و در مرکز آموزش ارتباطات و مدلسازی کامپیوتری توسعه یافت.

این نرم افزار برای شبیه سازی سیستم های پیچیده طرحی شده است. کاربردهای آن در علوم ریاضی، فیزیک، شیمی، کامپیوتر، پزشکی، اجتماعی و اقتصادی و .. است.

یکی از مزیت های این نرم افزار این است که با زبان جاوا نوشته شده است. این ویژگی باعث شده که این نرم افزار بتواند بر روی هر پلت فورمی اجرا شود (Mac, Windows, Linux).

ویژگی های این نرم افزار:

۱- سیستم عامل : قابل اجرا بر روی پلت فرم های مختلف

۲- زبان : کاملا قابل برنامه ریزی، ساختار زبانی ساده، قابلیت مشاهده عوامل ایجاد شده در بستر ایجاد عوامل، امکان مرتبط نمودن اشیاء ایجاد شده و شبکه سازی آنها، دارا بودن مجموعه قابل توجهی از اشکال از پیش طراحی شده و ...

۳- محیط : قابلیت مشاهده مدل بصورت دو و سه بعدی، اشکال قابل چرخش و اندازه پذیر، خط فرمان هوشمند، دارا بودن اهرم سرعت که اجازه کنترل سرعت مدل و مشاهده خروجی را با دقت بیشتر می دهد، سیستم رسم نمودار انعطاف پذیر و قدرتمند و ...

و ...

بخش اول : آشنایی با نرم افزار

کنترل کننده های مدل :کلید ها

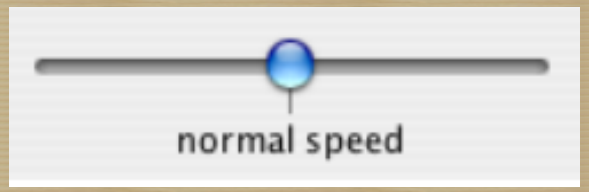
کلیدها دو دسته اند :

۱-کلید های دائمی : که تا وقتی مجدداً زده نشوند عمل می نمایند.

۲- کلید های یک عمله : با هر بار زدن تنها یکبار (در هر واحد از زمان) عمل می نمایند

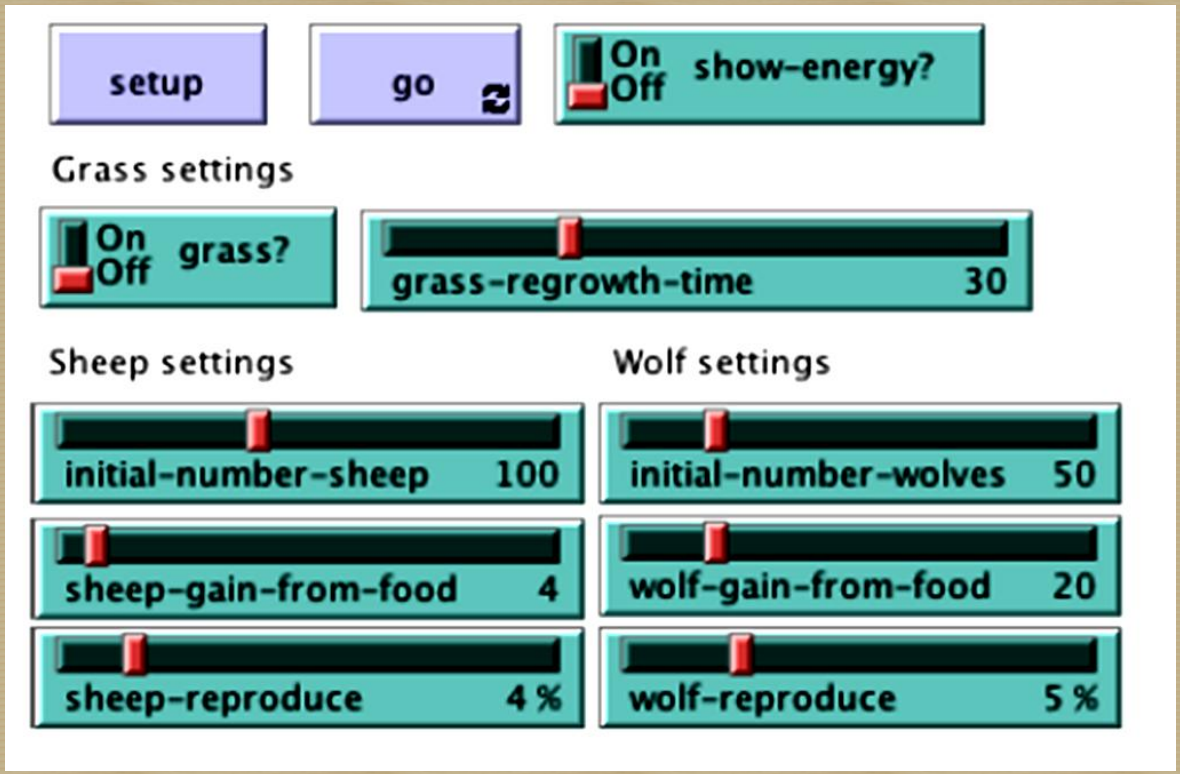
کنترل کننده سرعت : لغزنده سرعت

که وظیفه اش تنظیم سرعت حرکت اجسام طراحی شده، تغییر رنگ زمینه ها و از این قبیل است.



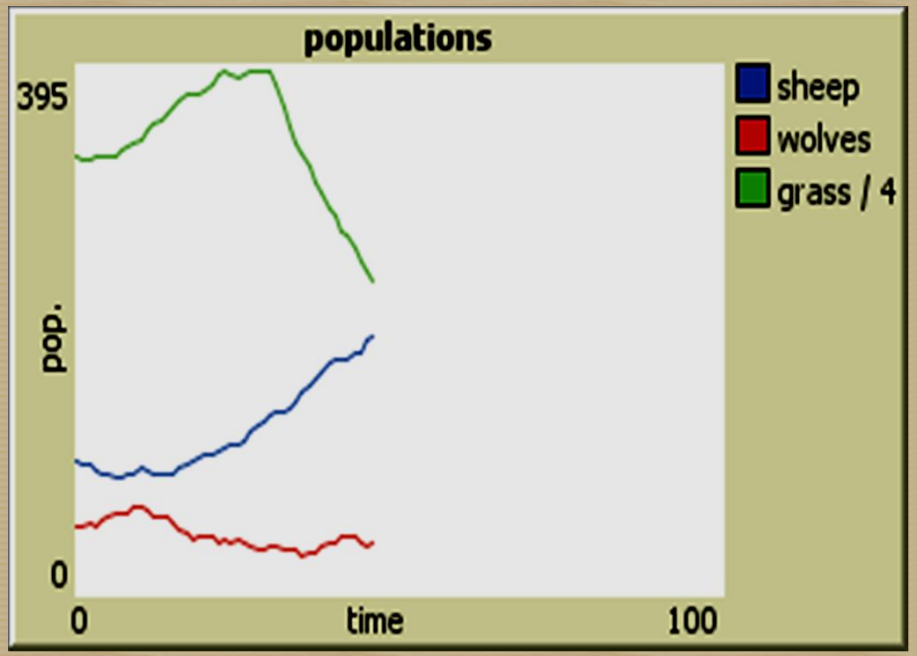
انجام تنظیمات : لغزنده ها و سوئیچ ها

تنظیمات درون مدل به شما این امکلان را می دهد که سناریوها و پیش فرض های مختلف را آزمون نمایید. به این شکل درک عمیق تری نسبت به آن پدیده پیدا می نمایید. لغزنده ها و سوئیچ ها به شما این امکان را در مدل می دهند.



جمع آوری اطلاعات : نمودارها و نمایشگرها

یکی از اهداف مدل سازی جمع آوری اطلاعات در مورد یک موضوع یا موضوعاتی است که انجام آنها در شرایط آزمایشگاهی بسیار دشوار است. NetLogo دو راه اصلی برای نمایش داده ها به کاربران دارد : یکی نمودارها و دیگری نمایشگرها.

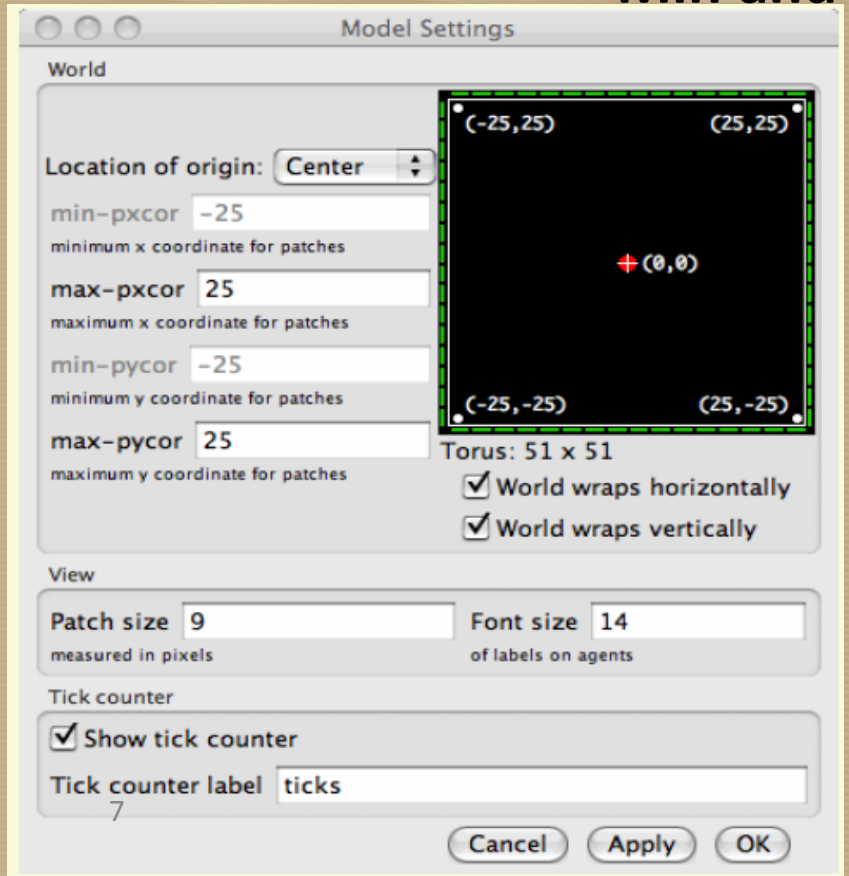


کنترل مشاهده :

با استفاده از لغزنده و گزینه های موجود در نوار ابزار نرم افزار شما می توانید سرعت اجرای مدل و نحوه مشاهده جزئیات تغییرات را تنظیم نمایید.

اندازه حوزه مشاهده نیز توسط ۵ پارامتر تعیین می شود:

Min and Max X, Min and Max Y and, Patch Size



(حداقل و حداکثر ایکس، حداقل و حداکثر وای و اندازه عرصه)

برای انجام این تنظیمات دکمه Settings را در نوار ابزار فشار داده، در کادر ظاهر شده تنظیمات دلخواه را انجام می دهیم.

مرکز فرماندهی (COMMAND CENTER)

مرکز فرماندهی در زبانه Interface واقع شده است و به شما اجازه می دهد تا فرمان ها را وارد کرده یا مدل را هدایت نمایید. فرمان ها در واقع دستورالعمل هایی هستند که شما به عوامل NetLogo (اجسام یا موضوعات (turtles)، عرصه ها (patches)، روابط یا پیوندها (links) و ناظرین (observer)) می دهید.

وقتی از گزینه observer در سمت چپ خط فرمان استفاده می نماییم با استفاده از کلمه ask می توانیم به سایر عوامل فرمان بدهیم. مانند موارد ز یر:

```
observer> ask patches [ set pcolor yellow ]
observer> ask turtles [ set color brown ]
```

اما هنگامی که از سایر گزینه ها استفاده می نماییم در واقع به آن عوامل مستقیما فرمان می دهیم. مانند فرمان های زیر:

```
turtles> set color pink
patches> set pcolor green
```

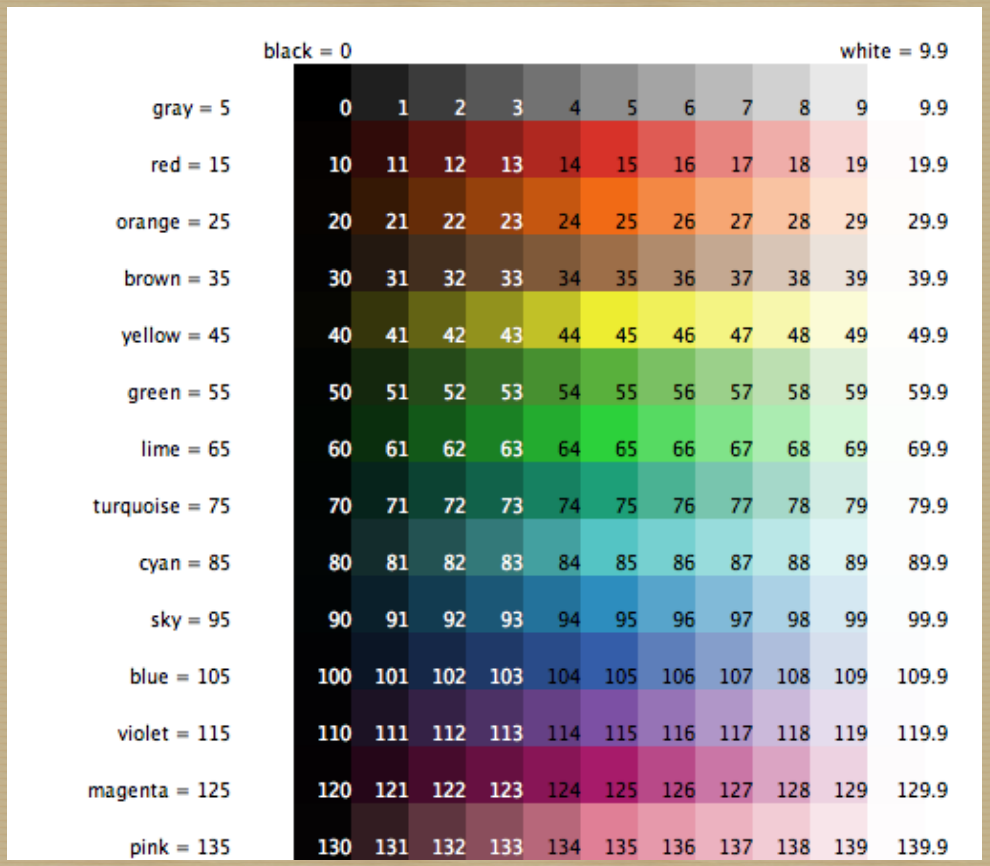
بخش اول : آشنایی با نرم افزار

تفاوت بین متغیر، فرمان و دستورالعمل

متغیر: عباراتی مانند color و pcolor را گویند. برخی متغیرها خاص turtles و برخی خاص patches هستند.

فرمان : عباراتی مانند set، ask

دستورالعمل : مجموعه ای از فرمان ها و متغیرها



کار با فرمان تنظیم رنگ:

نرم افزار Netlogo قابلیت تشخیص شانزده رنگ اصلی را دارد و به سه صورت می توان رنگ ها را در آن تعریف نمود:

ورود نام رنگ:

```
turtles> set color pink
```

ورود کد رنگ :

```
turtles> set color 135
```

ترکیب نام رنگ و عدد :

```
turtles> set pcolor red – 2
```

بخش اول : آشنایی با نرم افزار

نحوه کار با patches, turtles, links, and the observer (فضاها یا عرصه ها، اجسام، پیوندها و مرورگر یا نمایشگرها)

Patches ثابت یا ایستا بوده و شامل شبکه ای از سلول ها اند که کنارهم چیده شده اند.

Turtles در سطح شبکه حرکت میکنند.

Links دو Turtles را بهم مرتبط می نمایند

Observer یا ناظر بر هر آنچه که در حال اتفاق افتادن است نظارت می کند. کاری که patches, turtles, links نمی توانند انجام دهند.

تمام این فرمان ها می تواند در نرم افزار NetLogo اجرا گردد.

یک فرآیند یا دستورالعمل مجموعه ای از فرمان ها را در قالب یک دستور اجرا می نماید.

بخش دوم : اجرای یک مدل عملی

می خواهیم با استفاده از این نرم افزار یک طرح تعادل دام و مرتع را شبیه سازی نماییم. بعبارت دیگر می خواهیم ببینیم با افزایش یا کاهش تعداد دام ها و تغییر کیفیت مراتع در آینده چه اتفاقی خواهد افتاد. بمنظور ارائه تصویری روشن از آینده این مراتع برنامه ریزان تصمیم به مدلسازی فرآیند تخریب مراتع می نمایند. این کار قبل از هر چیز مستلزم یک تیم شامل کارشناسان مرتعداری، تغذیه دام، و مدلسازی است.

کارشناسان مرتع پس از بررسی مرتع مورد نظر کیفیت مراتع را به شکل زیر تعیین نموده اند :

۱– مراتع غنی
۲– مراتع سطح دو
۳– مراتع سطح سه
۴– مراتع سطح چهار
۵– مراتع تخریب شده

هر بخش از مرتع پس از چرا ۳ درصد احتمال دارد رشد نموده و دوباره کیفیت خود را ارتقاء دهد.

بخش دوم : اجرای یک مدل عملی

کارشناسان تغذیه دام نیز ضمن بررسی ارزش غذایی مراتع فوق ارزش غذایی هر یک را به شرح زیر تعیین نموده اند :

۱- مراتع غنی با ارزش غذایی ۱۰ واحد در هر چرا برای هر واحد دامی

۲- مراتع سطح دو با ارزش غذایی ۶ واحد در هر چرا برای هر واحد دامی

۳- مراتع سطح سه با ارزش غذایی ۴ واحد در هر چرا برای هر واحد دامی

۴- مراتع سطح چهار با ارزش غذایی ۲ واحد در هر چرا برای هر واحد دامی

۵- مراتع تخریب شده و فاقد ارزش غذایی در هر چرا برای هر واحد دامی

هر واحد دامی پس از رسیدن به سطح رشد مناسب (۱۰۰ واحد انرژی) آماده زاد و ولد شده و به ازاء هر مورد بره متولد شده ۵۰ واحد انرژی از دست خواهد داد.

هر واحد دامی به ازای هر بار جابجایی برای چرا ۲ واحد انرژی از دست می دهد.

هر واحد دامی با کاهش انرژیش به کمتر از ۱۰ واحد حذف خواهد شد.

کارشناسان برنامه ریزی و مدلسازی ضمن بررسی نظرات کارشناسان فوق و ارائه طرح مفهومی اولیه کار طراحی مدل عملیاتی را آغاز می نمایند. در ادامه بصورت گام به گام این فرآیند را دنبال می نماییم.

بخش دوم : اجرای یک مدل عملی

می خواهیم با استفاده از نرم افزار نت لوگو این طرح را شبیه سازی نماییم.

برای شروع یک مدل جدید، گزینه New را از منوی File انتخاب کنید. کار را با ایجاد یک دکمه Setup شروع کنید :

۱) گزینه Button را از زبانه Interface انتخاب نمایید.

۲) در یک بخش خالی از صفحه کلیک نمایید.

۳) در کادر ظاهر شده در بخش Command کلمه setup را تایپ نموده Okرا بزنید.

۴) اکنون شما یک کلید setup دارید.

از آنجایی که در حال حاضر هیچ دستوری برای این کلید ننوشته ایم رنگ آن قرمز است. اگر می خواهید پیغام خطای واقعی را ببینید، بر روی آن کلیک نمایید.

بخش دوم : اجرای یک مدل عملی

حالا می خواهیم فرایند Setup را بنحوی تعریف نماییم، که پیغام خطا از بین برود. به زبانه Procedures بروید و عبارت زیر را تایپ نمایید:

```
to setup
clear-all
create-turtles 100
ask turtles [ setxy random-xcor random-ycor ]
end
```

همانگونه که ملاحظه می نمایید یک دستورالعمل با to آغاز می شود و با end پایان می یابد.

اجازه دهید ببینیم شما چه چیزی تایپ کرده و در ازای هر سطر چه چیزی خواهید دید.

بخش دوم : اجرای یک مدل عملی

to setup به معنای اجرای فرمان برای کلید **setup** است

clear-all همه چیز را به حالت اول بر میگرداند. تمام اجسام و فضاهایی را که ساخته شده پاک می نماید و فرمان را برای یک اجرای تازه آماده می نماید.

create-turtles 100 ۱۰۰ عدد جسم را می سازد. ساختن اجسام از مرکز عرصه آغاز میشود.

ask turtles [...] دستور می دهد که هر یک از **turtle** ها باید دستور داخل کروشه را اجرا نمایند.

setxy random-xcor random-ycor فرمانی است برای استفاده در **"reporter"** ها. **reporter** عکس دستور بوده، نتیجه ای را گزارش می نماید.

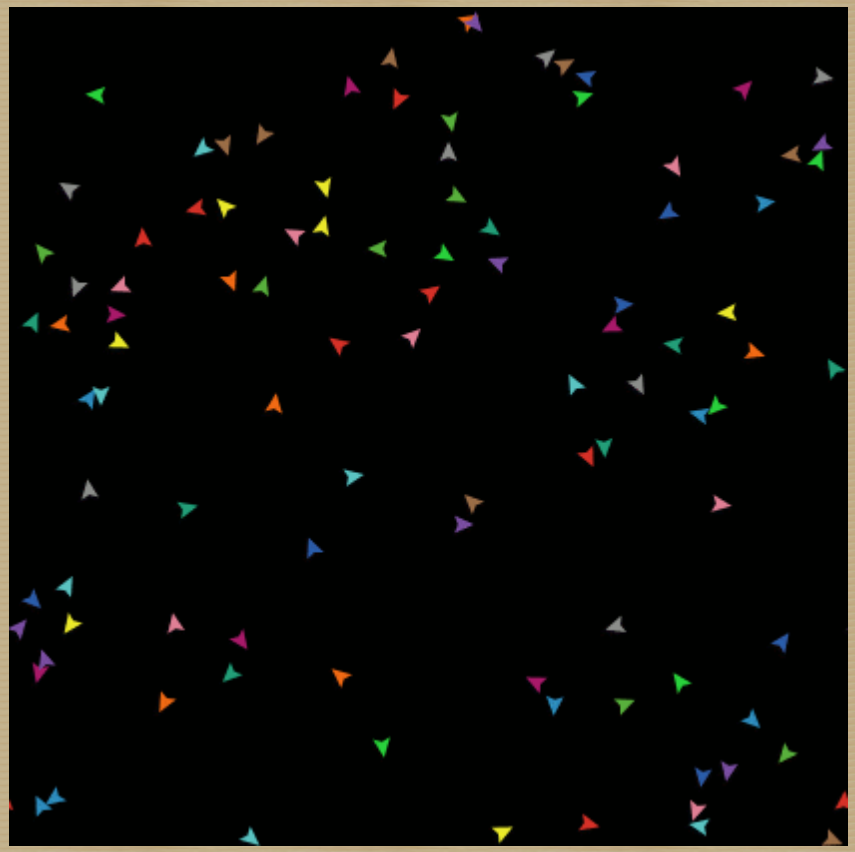
در ابتدا هر **turtle** فرمان **random-xcor** را اجرا می کند. که یک عدد تصادفی از محدوده مجاز را برای مختصات **turtle** در امتداد محور ایکس ها گزارش خواهد کرد.

سپس هر **turtle** یک **random-ycor** را برای مختصات محور **Y** اجرا می نماید . در نهایت هر **turtle** دستور **setxy** را با دو عدد ورودی اجرا می کند، که باعث می شود جسم بین نقاط مختصات حرکت نماید.

end دستور **setup** را کامل می نماید.

بخش دوم : اجرای یک مدل عملی

وقتی که کار تایپ شما انجام شد، به تب interface برگشته و دکمه setup را که قبلا ساخته اید فشار دهید. ما اجسام پراکنده در سراسر صفحه را مشاهده می کنید.



اگر کمی فکر کنید خواهید فهمید که شما به چه چیزی احتیاج داشته اید. شما احتیاج به ساختن یک کلید و یک فرمان برای استفاده از آن کلید داشتید. یک کلید وقتی کامل میشود که شما این دو کار را انجام داده باشید. در ادامه روش افزودن دیگر ویژگی ها به مدل را خواهید آموخت.

بخش دوم : اجرای یک مدل عملی

اما این تصویر شاید تداعی گر واقعی یک طرح تعادل دام و مرتع نباشد لذا می خواهیم این دستور را بشکلی ویرایش نماییم که بهتر جلوه نماید.

```
breed [sheep a-sheep]

to setup
clear-all
set-default-shape sheep "sheep"
create-sheep 100
ask sheep [ setxy random-xcor random-ycor set color white set size 1 set energy 30]
ask patches [ set pcolor 65 ]
end
```

روش ساختن کلید Go

تمام مراحل ساخت کلید Setup را انجام دهید منتها بجای کلمه setup کلمه go را بنویسید و چک باکس forever را نیز فعال نمایید. این گزینه این امکان را می دهد که دستور طراحی شده برای این کلید با هر بار فشار متوقف شده و دوباره اجرا گردد. بدون اینکه محدودیتی در تعداد دفعات اجرا داشته باشیم.

عبارت زیر را در تب Procedures اضافه نمایید:

```
to go
move-sheep
end
```

اما move-sheep چیست؟ آیا یک دستور نخستین است (به عبارت دیگر ، در NetLogo ساخته شده است)، آیا مثل فرمان clear-all است؟ خیر، این دستور باید تعریف گردد.

فرآیند move-sheep را بعد از فرآیند go در تب Procedures بصورت زیر اضافه نمایید :

```
to move-sheep
ask sheep [ right random 360 forward 1]
end
```

فرمان pen-down برای رد یابی می تواند به انتهای خط فرمان افزوده شود.

بخش دوم : اجرای یک مدل عملی

نکته:

توجه داشته باشید که بین دو کلمه move و sheep هیچ فاصله ای وجود نداشته باشد.

هر یک از فرمان های move-sheep چه کاری را انجام می دهند.

فرمان ask sheep [...] : دستور می دهد که هر یک از sheep ها باید دستور داخل کروشه را اجرا نمایند.

فرمان right random 360 : دستور دیگری است که از یک reporter استفاده می کند. در ابتدا هر sheep بشکل تصادفی عددی بین ۰ تا ۳۵۹ را اختیار می کند. سپس به اندازه این عدد میچرخد.

فرمان forward 1 : باعث می شود sheep یک گام به جلو برود.

تجربه کار با دستورات

پیشنهاد می کنیم فرمان های دیگری را نیز امتحان نمایید.

دستوراتی شبیه turtles> set color red را در Command Center وارد نمایید. یا دستوراتی را به setup, go یا move-sheep اضافه نمایید.

توجه داشته باشید که وقتی شما فرمانی را به Command Center وارد می نمایید. باید یک از حالات turtles> ، patches> یا observer> را از popup menu در سمت چپ، بسته به اینکه کدام یک از عوامل قرار است آن فرمان را اجرا نمایند، انتخاب کنید. این موضوع دقیقا شبیه استفاده از ask turtles یا ask patches ها است. از کلید tab جهت حرکت در بین گزینه ها می توانید استفاده نمایید.

می تونید در فرآیند move-sheep عدد ۳۶۰ را تغییر داده و مثلا ۴۵ بگذارید و نتیجه را مشاهده کنید.

متغیرهای turtles

تا کنون ما turtle هایی را طراحی نمودیم که بدون هیچ تعاملی حرکت می کردند. اجازه دهید کار را کمی پیچیده تر کرده و turtles و patches طراحی نماییم که بر هم اثر دارند.

می خواهیم turtle هایی طراحی نماییم که فضاهای سبز را خورده رشد کرده و نابود شوند. مناطق سبز نیز باید بعد از خرده شدن به تدریج رشد نمایند.

ابتدا باید تعیین نماییم هر گوسفند چقدر انرژی دارد. برای این منظور ما باید یک متغیر sheep اضافه نماییم.

ابتدا یک عبارت تحت عنوان sheep-own به بالای زبانه procedure و قبل از تمام دستورات اضافه می نماییم، و آنرا energy می نامیم. یعنی می نویسیم:

```
sheep-own [energy]
```

فرمان energy به گوسفندها اجازه می دهد که تغذیه کنند.

بخش دوم : اجرای یک مدل عملی

به زبانه procedure برگشته فرمان go را بشکل زیر اصلاح نمایید:

```
to go
move-sheep
eat-grass
end
```

بصورت زیر یک فرآیند جدید تحت عنوان eat-grass اضافه نمایید :

```
to eat-grass
ask sheep [
if pcolor = 65 [ set pcolor 66 right random 360 forward 1 set energy (energy + 10) ]
if pcolor = 66 [ set pcolor 67  left random 360 forward 2 set energy (energy + 6) ]
if pcolor = 67 [ set pcolor 68 forward 3 set energy (energy + 4) ]
if pcolor = 68 [ set pcolor black left random 360 set energy (energy + 2) ]
if pcolor = black [ left random 360 forward 2 set energy (energy - 2) ]
]
end
```

بخش دوم : اجرای یک مدل عملی

برای اولین بار است که از دستور if استفاده می نماییم، به کد دقت کنید. هر گوسفند وقتی که این دستورات اجرا می شود، رنگ patch که در (pcolor) است را با رنگ سبز مقایسه می کند. اگر رنگ patch سبز باشد، گزارش مقایسه درست خواهد بود، و تنها در این صورت است که دستورات درون کروشه اجرا خواهد شد (در غیر این صورت اجرا نمی شود).

این دستور turtle ها را وادار می نمایند رنگ patch را متناسب دستور تغییر داده و انرژی خود را بسته به رنگ مرتع افزایش دهند. رنگ patch تغییر کرده و این به معنی خورده شدن گیاهان مرتعی در آن نقطه است به این صورت انرژی هر دام تنها از طریق تغذیه از مراتع بیشتر می شود.

اجازه دهید با اجرای فرمانی تعریف نماییم که دام ها در نتیجه حرکت نمودن به اندازه نظر کارشناسان انرژی از دست بدهند.

فرمان move-sheep را بصورت زیر بازنویسی نمایید:

```
to move-sheep
ask sheep [ right random 360 forward 1 set energy (energy - 2) ]
end
```

با اجرای این دستور در واقع هر واحد دامی در هر بار چرا دو واحد از انرژیش را از دست می دهد.

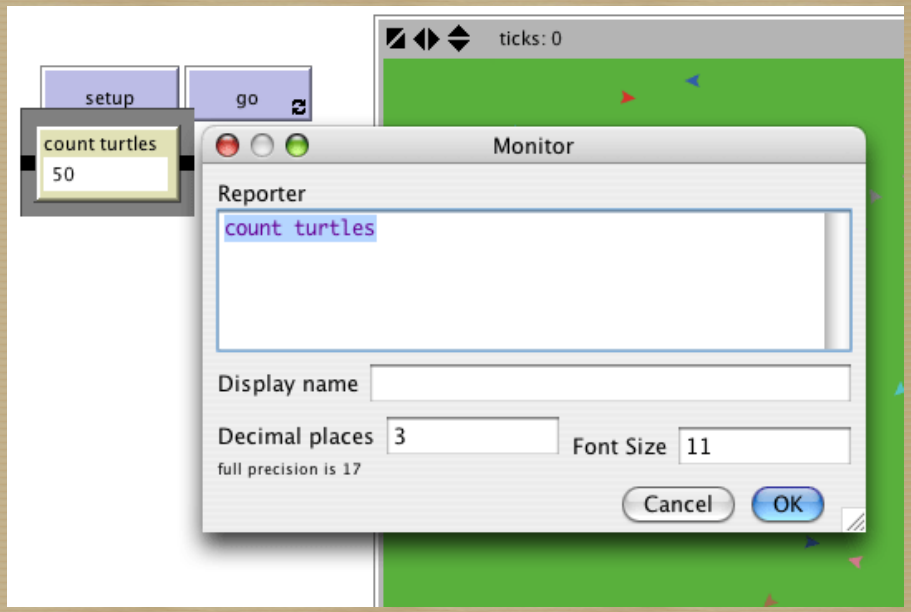
با بازگشت به زبانه Interface و فشردن کلید setup و سپس go نتیجه را ملاحظه فرمایید.

بخش دوم : اجرای یک مدل عملی

نمایشگر:

در مرحله بعد می خواهیم چند نمایشگر در زبانه interface ایجاد نماییم. (می توانید آنها را دقیقا مانند کلیدهای و لغزنده ها، با استفاده از آیکون مانیتور موجود در نوار ابزار ایجاد نمایید) اجازه دهید که اولین مانیتور را بسازیم.

یک نمایشگر با استفاده از آیکون monitor در نوار ابزار ایجاد نمایید. یک کادر محاوره ای ظاهر خواهد شد. در کادر محاوره ای بنویسید: count turtles (به تصویر زیر نگاه کنید). دکمه ok را فشار دهید.



اکنون اجازه دهید که نمایشگر دوم را بسازیم.

در کادر محاوره ای ظاهر شده این بار بنویسید:

count patches with [pcolor = 65]

در محل Display name بنویسید high grass

با زدن ok کادر محاوره ای را ببندید.

count برای نمایش میزان مراتع مرغوب باقی مانده و فرمان with [pcolor = 65] برای محدود نمودن نمایش به مراتع سبز یا تخریب نشده است.

این کار را برای نمایش سایر مراتع تکرار نموده برای هر یک تنها کد رنگ را تغییر دهید و نام مناسبی برای آن انتخاب کنید.

بخش دوم : اجرای یک مدل عملی

سوئیچ ها و برچسب ها

turtle ها نه تنها رنگ patche ها را عوض می کنند؛ بلکه انرژی به دست آورده و از دست می دهند. می خواهیم با استفاده از یک نمایشگر ببینیم که چگونه انرژی یک turtle بالا و پایین می رود.

اگر ما بتوانیم انرژی هر turtle را دائما ببینیم، بهتر است. در این مرحله می خواهیم دقیقا همین کار را انجام دهیم، و یک سوئیچ را به گونه ای اضافه نماییم که بتوانیم آنرا روشن و خاموش کنیم.

برای ایجاد یک سوئیچ ، بر روی آیکون switch در نوار ابزار کلیک نمایید (در زبانه interface) و سپس در یک نقطه خالی از صفحه کلیک کنید. در کادر ظاهر شده در قسمت Global variable بنویسید show-energy? علامت سئوال را فراموش نکنید.

اکنون به زبانه Procedures برگشته دستورالعمل eat-grass را بشکل زیر اصلاح نمایید:

```
to eat-grass
ask sheep [
if pcolor = 65 [ set pcolor 66 right random 360 forward 1 set energy (energy + 10) ]
if pcolor = 66 [ set pcolor 67  left random 360 forward 2 set energy (energy + 6) ]
if pcolor = 67 [ set pcolor 68 forward 3 set energy (energy + 4) ]
if pcolor = 68 [ set pcolor black left random 360 set energy (energy + 2) ]
if pcolor = black [ left random 360 forward 2 set energy (energy - 2) ]
ifelse show-energy? [ set label energy ] [ set label "" ]
]
end
```

بخش دوم : اجرای یک مدل عملی

دستورالعمل eat-grass شامل فرمان ifelse است. به کد این فرمان دقت کنید. هر turtle وقتی که این فرمان های جدید اجرا می شود مقدار show-energy? را چک می کند (توسط سوئیچ تعیین می شود). اگر سوئیچ روشن باشد، مقایسه درست است و turtle دستورات داخل اولین کروشه را اجرا خواهد کرد. در این حالت، مقدار انرژی را بصورت برچسب به turtle می دهد. اگر مقایسه نادرست است (سوئیچ در حالت خاموش است) و turtle دستورات درون کروشه دوم را اجرا می کند. در این مورد، متن برچسب را حذف می نماید.

در NetLogo به قطعه ای از متن string می گویند (رشته). یک رشته پیوستاری از حروف و نمادهای دیگر است، که بین دو کوتیشن " " نوشته شده است. در اینجا ما دو کوتیشن در کنار هم داریم، بدون هیچ چیزی در بین آنها. این یک رشتهٔ خالی است. اگر برچسب یک turtle یک رشتهٔ خالی است، هیچ متنی به turtle اضافه نمی شود). با اجرای برنامه ملاحظه می نمایید که هر turtle با خوردن چمن ها مقدار انرژیش افزایش و با حرکت کردن کاهش می یابد.

دیگر دستوالعمل ها

در حال حاضر دام ها مشغول خوردن هستند، اجازه دهید برنامه را طوری طراحی نماییم که آنها قادر به تکثیر و مرگ باشند، همچنین مراتع بتوانند رشد کنند.

می خواهیم هر سه این رفتارها را از طریق سه دستورالعمل جداگانه، اضافه کنیم.

ابتدا به زبانه Procedure بروید و دستورالعمل go را بشکل زیر بازنویسی کنید

```
to go
move-sheep
eat-grass
reproduce
check-death
end
```

بخش دوم : اجرای یک مدل عملی

دستورالعمل ها را برای reproduce، check-death و regrow-grass بشکل زیر اضافه نمایید:

```
to reproduce
ask sheep [
if energy > 100 [ set energy (energy - 50) hatch 1 [ set energy 30] ]
]
end

to check-death
ask sheep [
if energy <= 10 [ die ]
]
end

to regrow-grass
ask patches [
if random 100 < 3 and pcolor = black [ set pcolor 68 ]
if random 100 < 3 and pcolor = 68 [ set pcolor 67 ]
if random 100 < 3 and pcolor = 67 [ set pcolor 66 ]
if random 100 < 3 and pcolor = 66 [ set pcolor 65 ]
]
end
```

بخش دوم : اجرای یک مدل عملی

هر یک از این دستورالعمل ها از فرمان if استفاده می کنند. هر turtle زمانیکه check-death اجرا می گردد بررسی می نماید که آیا انرژی آن برابر یا کمتر از ۱۰ است؟ آنگاه به turtleدستور داده می شود که بمیرد. (مرگ یکی از فرمان های اولیه NetLogo است).

وقتی هر turtle دستورالعمل reproduce را اجرا می کند. میزان متغیر energy را کنترل می کند. اگر بیشتر از ۱۰۰ بود. دستورات داخل اولین کروشه را اجرا می کند. در این حالت انرژی turtle را تا ۵۰ کاهش داده و یک turtle جدید با انرژی ۳۰ متولد می گردد.

فرمان hatch نیز یکی از فرمان های اولیه NetLogo است که به این صورت نوشته می شود:

hatch عدد [فرمان]

عددی که در این فرمان قرار می گیرد نشانگر تعداد turtle های متولد شونده است. که همانند والدین خود خواهند بود. می توان فرمان را بنحوی نوشت که رنگ، عنوان یا ... turtle های جدید متفاوت باشد.

بخش دوم : اجرای یک مدل عملی

هنگامی که هر patch دستورالعمل regrow-grass را اجرا می کند ، کنترل می نماید که آیا عدد تصادفی انتخاب شده از ۱۰۰ کمتر از ۳ است یا خیر؟. اگر چنین باشد رنگ patch بصورت زیر تغییر می کند. احتمال اتفاق افتادن این موضوع برای هر patch ۳ درصد است. زیرا در اینجا سه عدد کمتر از ۳ وجود دارد (۰، ۱ و ۲).

با رفتن به زبانه interface و فشار دادن کلید setup و سپس go نتیجه را ملاحظه می نمایید NetLogo به ما اجازه رسم داده ها را نیز می دهد. که گام بعدی ما است.

```
if random 100 < 3 and pcolor = black [ set pcolor 68 ]
if random 100 < 3 and pcolor = 68 [ set pcolor 67 ]
if random 100 < 3 and pcolor = 67 [ set pcolor 66 ]
if random 100 < 3 and pcolor = 66 [ set pcolor 65 ]
```

رسم داده ها

برای اینکه داده ها را ترسیم نماییم احتیاج به ایجاد یک نمودار در زبانه interface و تعیین یکسری تنظیمات برای آن داریم. همچنین ما باید چند دستورالعمل به زبانه Procedures اضافه نماییم که نمودار را برایمان بروز رسانی نماید.

ابتدا به زبانه Procedures رفته دستورالعمل setup را با افزودن do-plots بشکل زیر اصلاح نمایید:

```
to setup
clear-all
set-default-shape sheep "sheep"
create-sheep 100
ask sheep [ setxy random-xcor random-ycor set color white set size 1 set energy
30]
ask patches [ set pcolor 65 ]
do-plots
end
```

do-plots را به بخش go نیز اضافه نمایید.

بخش دوم : اجرای یک مدل عملی

اکنون نوبت اضافه نمودن دستورالعمل های جدید است. می خواهیم نمودارها تعداد دام ها و مراتع را در هر لحظه از زمان برایمان نشان دهد. برا این منظور دستور زیر را برای do-plots می نویسیم:

```
to do-plots
set-current-plot "Totals"
set-current-plot-pen "sheep"
plot count sheep
set-current-plot-pen "high grass"
plot count patches with [pcolor = 65]
set-current-plot-pen "good grass"
plot count patches with [pcolor = 66]
set-current-plot-pen "medium grass"
plot count patches with [pcolor = 67]
set-current-plot-pen "low grass"
plot count patches with [pcolor = 68]
set-current-plot-pen "no grass"
plot count patches with [pcolor = black]
end
```

از فرمان plot برای افزودن نقطه بعدی در نمودار استفاده می نماییم. قبل از آن ما باید دو چیز را به NetLogo بگوییم. نخست، باید نمودارمان را مشخص نماییم (زیرا ممکن است بعدا بیش از یک نمودار نیاز باشد) و دوم اینکه باید نوع قلمی را که برای نمودار لازم داریم مشخص نماییم.

بخش دوم : اجرای یک مدل عملی

هر عدد در محور X از فرمان plot یک واحد زمان را نشان می دهد و اعداد محور Y تعداد دام ها و سطح هر یک از مراتع را در آن زمان نشان می دهند. با حرکت مداد در طول محور زمان، نمودار رسم می گردد.

برای اجرای دستور set-current-plot "Totals" باید یک نمودار به زبانه interface اضافه نمایید و نام آنرا دقیقا مشابه همان نام در زبانه procedure قرار دهید. حتی یک فاصلهٔ اضافه نیز نباید بگذارید چرا که آنرا خاموش خواهد نمود.

در زبانه interface و از نوار toolbox بر روی plot کلیک کرده و یک نمودار در قسمت خالی صفحه نمایش ایجاد نمایید. نام آنرا Totals بگذارید. محور X را time و محور Y را total بنامید.

بر روی گزینه Create کلیک نمایید. نام این مداد را sheep بگذارید. بعد از ok نمودن یک قلم جدید ایجاد کرده آنرا grass بگذارید. بعد ز ok نمودن رنگ این مداد را سبز انتخاب نمایید. دقت داشته باشید که در هنگام ایجاد یک نمودار شما میتوانید مقادیر حداقل و حداکثر برای محورهای x و y تان نیز انتخاب کنید.

با زدن کلیدهای setup و سپسgo خروجی کارتان را در زبانه interface ملاحظه می نمایید

یکسری جزئیات بیشتر

اولاً ممکن است شما بخواهید بجای اینکه با تعداد صد واحد دامی شروع نمایید بتوانید تعداد آنها را خودتان تعیین نمایید.

برای این منظور یک متغیر لغزنده با نام number با استفاده از از ایکون monitor در نوار ابزارتان بسازید. سعی نمایید مقدار حداقل و حداکثر را در لغزنده تغییر دهید. سپس بجای ۱۰۰ در قسمت create-sheep از دستور to setup بنویسید number (بصورت زیر):

```
to setup
clear-all
set-default-shape sheep "sheep"
create-sheep number
ask sheep [ setxy random-xcor random-ycor set color white set size 1 set energy 30]
ask patches [ set pcolor 65 ]
do-plots
end
```

یکسری جزئیات بیشتر

ثانیاً بد نیست اگر انرژی ای که دام ها در نتیجه خوردن چمن ها بدست آورده یا درنتیجه زاد و ولد از دست می دهند را نیز بتوانید تغییر دهید. برای این منظور بشکل زیر عمل می نماییم:

یک لغزنده با نام **energy-from-grass** و یک لغزنده دیگر نیز با نام **birth-energy** بسازید.

سپس دستور **eat-grass** را بشکل زیر اصلاح نمایید :

```
to eat-grass
ask sheep [
if pcolor = 65 [ set pcolor 66 right random 360 forward 1 set energy (energy +
energy-from-grass) ]
if pcolor = 66 [ set pcolor 67  left random 360 forward 2 set energy (energy + 6) ]
if pcolor = 67 [ set pcolor 68 forward 3 set energy (energy + 4) ]
if pcolor = 68 [ set pcolor black left random 360 set energy (energy + 2) ]
if pcolor = black [ left random 360 forward 2 set energy (energy - 2) ]
ifelse show-energy? [ set label energy ] [ set label "" ]
]
end
```

بخش دوم : اجرای یک مدل عملی

و دستور reproduce را بشکل زیر اصلاح نمایید :

```
to reproduce
ask sheep [
if energy > 100 [ set energy (energy - birth-energy) hatch 1 [ set energy 30] ]
]
end
```

برای متوقف نمودن مدل در یک زمان خاص می توان دستور زیر را به مدل اضافه نمود:

```
to go
if ticks >= 30 [ stop ]
move-sheep
eat-grass
reproduce
check-death
regrow-grass
tick
do-plots
end
```

پایان

هرچه موانع جدی‌تر و سخت‌تر باشد، لذت تلاش و پیروزی بیشتر است.

اریک باتروورت