Publicado en Septiembre de 2018. Este documento usa la versión 6.0 de NetLogo. Traducido al Castellano por Haroldo Miranda.

Tutorial # 3 Procedimientos

Agentes y Procedimientos

En el Tutorial n.º 2, aprendió a usar el terminal de instrucciones y los monitores de agente para inspeccionar y modificar agentes y hacer que ellos realicen cosas. Ahora está listo para conocer el verdadero corazón de un modelo de NetLogo: la pestaña Código.

Ha visto que los agentes en NetLogo están divididos en parcelas, tortugas, enlaces y el observador. Las parcelas son estacionarias y dispuestas en una cuadrícula. Las tortugas se mueven sobre esa grilla. Los enlaces conectan dos tortugas. El observador supervisa todo lo que está sucediendo y hace lo que las tortugas, parcelas y enlaces no pueden hacer por sí mismos.

Los cuatro tipos de agentes pueden ejecutar instrucciones de NetLogo. Los cuatro también pueden ejecutar "procedimientos". Un procedimiento combina una serie de instrucciones de NetLogo en un único comando nuevo que usted define.

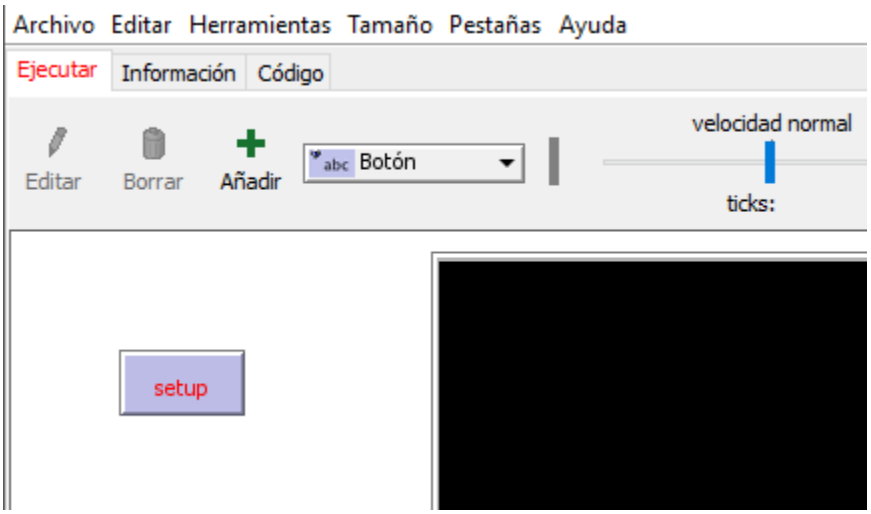
Ahora aprenderá a escribir procedimientos que hacen que las tortugas se muevan, coman, se reproduzcan y mueran. También aprenderá cómo hacer monitores, controles deslizantes y gráficos. El modelo que construiremos es un modelo de ecosistema simple no muy diferente al del modelo Wolf Sheep Predation del Tutorial n. ° 1.

Construyendo el botón de Configuración (Setup)

Para comenzar un nuevo modelo, seleccione "Nuevo" en el menú Archivo. Luego comience creando un botón de configuración:

- Haga clic en el ícono "Añadir" en la barra de herramientas en la parte superior de la pestaña Interfaz.
- En el menú al lado de Añadir, seleccione Botón (si aún no está seleccionado).
- Haga clic donde desee que aparezca el botón en el área blanca vacía de la pestaña Interfaz.
- Se abre un cuadro de diálogo para editar el botón. Escriba `setup` en el cuadro etiquetado "Instrucciones".
- Presione el botón OK cuando haya terminado; el cuadro de diálogo se cierra.

Ahora tiene un botón de configuración (setup). Al presionar el botón se ejecuta un procedimiento llamado "setup". Un procedimiento es una secuencia de instrucciones de NetLogo al que asignamos un nuevo nombre. Definiremos ese procedimiento pronto, porque aún no lo hemos hecho. El botón se refiere a un procedimiento que no existe, por lo que el botón se vuelve rojo:



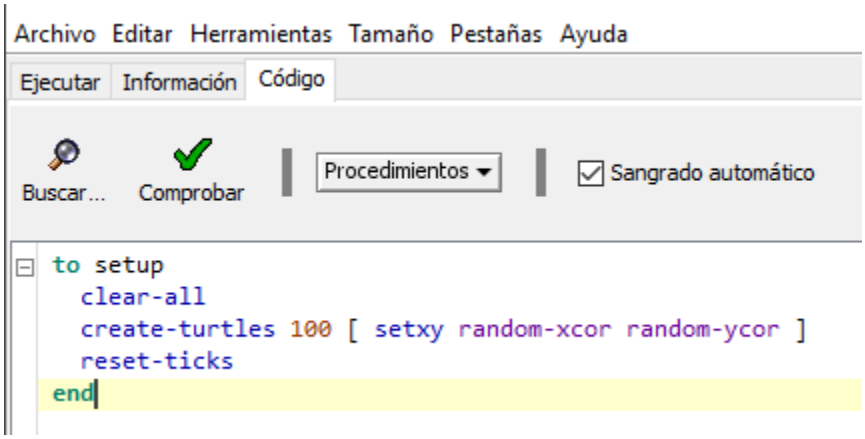
Si desea ver el mensaje de error, haga clic en el botón.

Ahora crearemos el procedimiento de "setup", por lo que el mensaje de error desaparecerá:

- Cambie a la pestaña Código.
- Escriba lo siguiente:

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

Cuando termine, la pestaña Código se ve así:



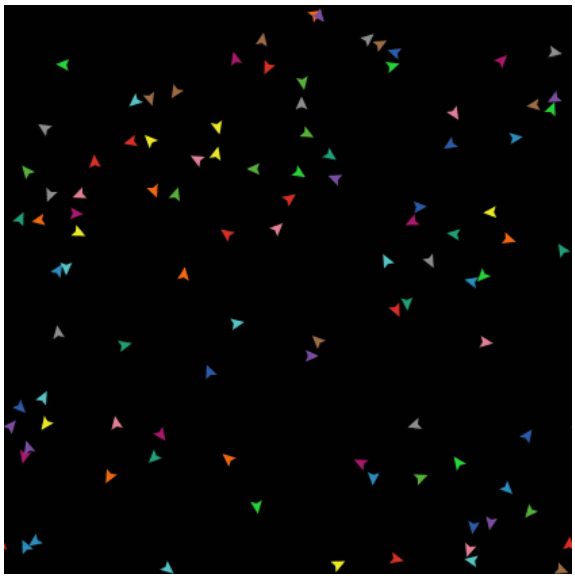
Tenga en cuenta que algunas líneas están con sangría. A la mayoría de las personas les resulta útil sangrar su código. No es obligatorio, pero hace que el código sea más fácil de leer y cambiar.

Su procedimiento comienza con `to` termina con `end`. Cada procedimiento comienza y termina con estas palabras.

Miremos lo que escribió y vea qué hace cada línea de su procedimiento:

- `to setup` comienza definiendo un procedimiento llamado "setup".
- `clear-all` restablece el mundo a un estado inicial, vacío. Todas las parcelas se vuelven negras y las tortugas que se hayan creado desaparecen. Básicamente, borra la pizarra para una nueva ejecución de modelo.
- `create-turtles 100` crea 100 tortugas. Comienzan en el origen, es decir, en el centro de la parcela 0,0.
- Después de `create-turtles` podemos poner instrucciones para que ejecuten las nuevas tortugas, entre corchetes.
- `setxy random-xcor random-ycor` es una instrucción que usa "reporteros". Un reportero, a diferencia y en oposición a una instrucción, informa un resultado. Primero, cada tortuga ejecuta el reporte `random-xcor` que informará un número aleatorio del rango permitido de coordenadas de tortuga a lo largo del eje X. Luego, cada tortuga ejecuta el reporte `random-ycor`, que es lo mismo, pero para el eje Y. Finalmente cada tortuga ejecuta el comando setxy con esos dos números como entradas. Eso hace que la tortuga se mueva al punto con que posee esas coordenadas.
- `reset-ticks` inicia el contador de ticks, ahora que la configuración está completa.
- `end` completa la definición del procedimiento de configuración o "setup".

Cuando haya terminado de escribir, cambie a la pestaña Ejecutar y presione el botón de setup que creó anteriormente. Verá las tortugas dispersas por todo el mundo:



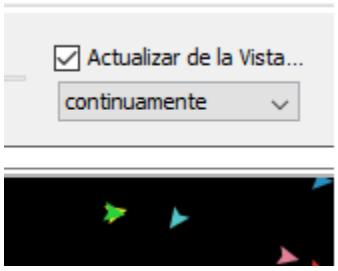
Presione el botón setup un par de veces más y vea cómo la disposición de las tortugas es diferente cada vez. Tenga en cuenta que algunas tortugas pueden estar una encima de la otra.

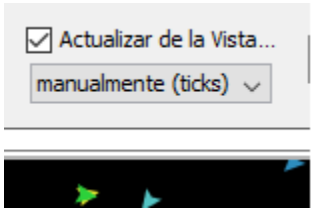
Piense un poco sobre lo que necesitó hacer para que esto sucediera. Necesitó hacer un botón en la interfaz y definir un procedimiento que utilice dicho botón. El botón funcionó sólo una vez que completó estos dos pasos por separado. En el resto de este tutorial, a menudo tendrá que completar dos o más pasos similares para agregar otra característica al modelo. Si parece que algo no funciona después de completar lo que creía que era el último paso para esa nueva función, continúe leyendo para ver si todavía hay más por hacer. Después de leer un par de párrafos más adelante, debe volver sobre las instrucciones para ver si hay algún paso que pueda haber pasado por alto.

Cambiar a actualizaciones de vista basadas en ticks

Ahora que estamos usando el contador de ticks (con `reset-ticks`), debemos decirle a NetLogo que solo necesita actualizar la vista una vez por tick, en lugar de actualizarlo continuamente.

- Encuentre el menú de actualizaciones de vista. Está sobre la vista y por defecto dice "continuamente".
- Elija "manualmente (ticks)" en su lugar.



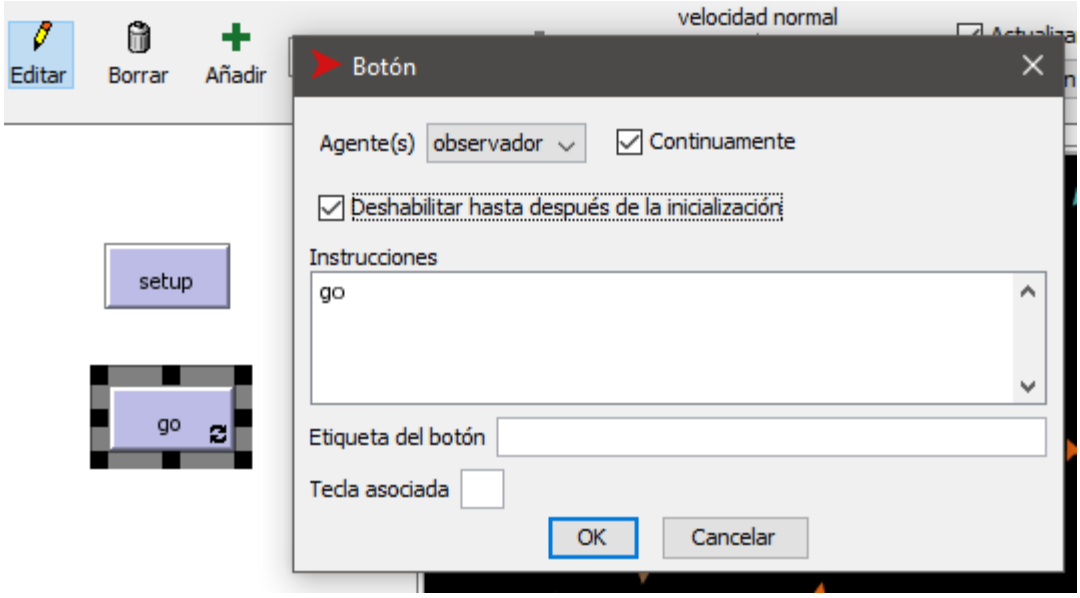


Esto hace que su modelo se ejecute más rápido y garantiza una apariencia consistente (ya que las actualizaciones se realizarán a intervalos constantes). Consulte la Guía de programación para una discusión más completa de las actualizaciones de vista.

Construyendo el botón go

Ahora hará un botón llamado "go". Siga los mismos pasos que usó para hacer el botón de configuración, excepto:

- Para Instrucciones, ingrese `go` en lugar de `setup`.
- Marque la casilla "Continuamente" en el cuadro de diálogo de edición.
- Marque también la casilla de verificación "Deshabilitar hasta después de la inicialización".



La casilla "Continuamente" hace que el botón permanezca presionado una vez, por lo que sus instrucciones se repiten una y otra vez, no solo una vez. La opción "Deshabilitar hasta después de la inicialización" le impide presionar `go` antes de `setup`.

- A continuación, agregue un procedimiento `go` a la pestaña Código:

```
to go
  move-turtles
  tick
end
```

`tick` es una función primitiva que avanza el contador de ticks de un tick por vez.

Pero, ¿qué es `move-turtles`? ¿Es una función primitiva (en otras palabras, incorporada en NetLogo)? No, es otro procedimiento que está a punto de agregar. Hasta ahora, ha introducido dos procedimientos que ha agregado usted mismo: `setup` y `go`.

Agregue el procedimiento `move-turtles` después del procedimiento `go`:

```
to go
  move-turtles
  tick
end

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

Notar que no hay espacios alrededor del guión en `move-turtles`. En el Tutorial n. ° 2 usamos el `red - 2`, con espacios, para restar dos números, pero aquí queremos mover las tortugas, sin espacios. El "-" combina "`move`" y "`turtles`" en un solo nombre.

Esto es lo que hace cada instrucción en el procedimiento `move-turtles`:

`ask tortugas [...]` dice que cada tortuga debe ejecutar las instrucciones entre corchetes.

`right random 360` es otra instrucción que usa un reportero. Primero, cada tortuga elige un número entero al azar entre 0 y 359. (`random` no incluye el número que se le da como posible resultado). Luego, la tortuga gira a la derecha este número de grados.

`forward 1` hace que la tortuga avance un paso.

¿Por qué no podríamos simplemente haber escrito todas estas instrucciones en `go` en lugar de hacerlo en un procedimiento separado? Podríamos haberlo hecho, pero durante el proceso de construcción de su proyecto, es probable que agregue muchas otras partes. Nos gustaría mantener `go` lo más simple posible, para que sea fácil de entender. Eventualmente, incluirá muchas otras cosas que desea que sucedan a medida que se ejecuta el modelo, como calcular algo o graficar los resultados. Cada una de estas cosas para hacer tendrá su propio procedimiento y cada procedimiento tendrá su propio nombre único.

El botón '`go`' que creó en la pestaña Ejecutar es un botón de continuidad, lo que significa que ejecutará continuamente sus instrucciones hasta que lo apague (al hacer clic nuevamente en él). Después de presionar '`setup`' una vez, para crear las tortugas, presione el botón '`go`'. Mire lo que sucede. Apáguelo y verá que todas las tortugas se detienen en seco.

Tenga en cuenta que, si una tortuga se mueve fuera del borde del mundo, "reingresa", es decir, aparece en el otro lado. (Este es el comportamiento predeterminado. Se puede cambiar, consulte la sección Topología de la Guía de programación para obtener más información).

Experimentando con Instrucciones

Sugerimos que comience a experimentar con otras instrucciones de tortuga.

Escriba instrucciones en el Terminal de Instrucciones (como `turtles> set color red`) o agregue instrucciones a `setup`, `go` o `move-turtles`.

Tenga en cuenta que cuando ingresa instrucciones en el Terminal de Instrucciones, debe elegir tortugas>, parcelas>, enlaces> u observador> en el menú emergente de la izquierda, dependiendo de qué agentes ejecutarán las instrucciones. Es como usar `ask turtles` o `ask patches`, pero ahorra escritura. También puede usar la tecla de tabulación para cambiar los tipos de agente, que puede encontrar más conveniente que usar el menú.

Puede tratar de escribir `turtles> pen-down` abajo en el Terminal de instrucciones y luego presionar el botón go.

Además, dentro del procedimiento `move-turtles` puede intentar cambiar el `right random 360` al `right random 45`.

Jugar. Es fácil y los resultados son inmediatos y visibles, uno de los muchos puntos fuertes de NetLogo.

Cuando sienta que ha hecho suficiente experimentación por ahora, está listo para seguir mejorando el modelo que estás construyendo.

Parcelas y variables

Ahora tenemos 100 tortugas moviéndose sin rumbo, completamente ajenas a cualquier otra cosa a su alrededor. Hagamos las cosas un poco más interesantes al darle a estas tortugas un lindo fondo contra el cual movernos.

Regrese al procedimiento de setup. Podemos reescribirlo de la siguiente manera:

```
to setup
  clear-all
  setup-patches
  setup-turtles
  reset-ticks
```

```
end
```

La nueva definición de `setup` se refiere a dos nuevos procedimientos. Para definir `setup-patches`, agregue esto:

```
to setup-patches
  ask patches [ set pcolor green ]
end
```

El procedimiento `setup-patches` establece el color de cada parcela en verde para comenzar. (La variable de color de una tortuga es `color`; la de una parcela es `pcolor`).

La única parte que queda en nuestro nuevo '`setup`' que aún no está definida es `setup-turtles`.

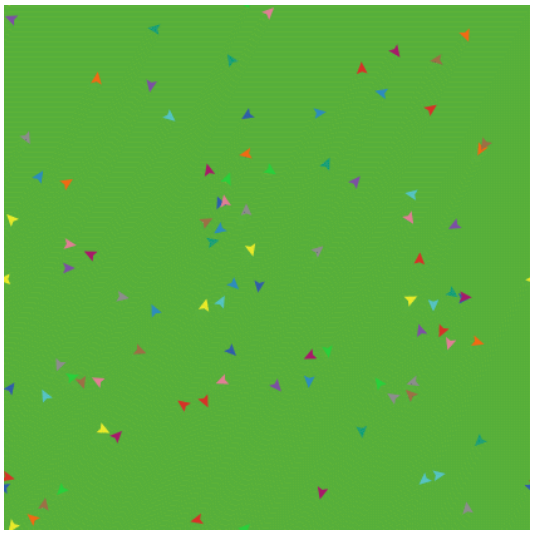
Agregue este procedimiento también:

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

¿Notó que el nuevo procedimiento `setup-turtles` tiene la mayoría de las mismas instrucciones que el antiguo procedimiento `setup`?

- Vuelva a la pestaña Ejecutar.
- Presione el botón de setup.

Voila! Aparece un exuberante paisaje de NetLogo completo con tortugas y manchas verdes:



Después de ver que el nuevo procedimiento `setup` funciona varias veces, puede que le resulte útil volver a leer las definiciones del procedimiento.

Variables de Tortuga

Así que tenemos algunas tortugas corriendo por un paisaje, pero no están haciendo nada con eso. Agreguemos algo de interacción entre las tortugas y las parcelas.

Haremos que las tortugas coman "hierba" (las manchas verdes), se reproduzcan y mueran. La hierba volverá a crecer gradualmente después de comerla.

Necesitaremos una forma de controlar cuándo una tortuga se reproduce y muere. Lo determinaremos haciendo un seguimiento de cuánta "energía" tiene cada tortuga. Para hacer eso, necesitamos agregar una nueva variable de tortuga.

Ya ha visto variables integradas de tortuga, tal como `color`. Para hacer una nueva variable de tortuga, agregamos una declaración propia de tortugas en la parte superior de la pestaña Código, antes de todos los procedimientos. La llamaremos energía:

```
turtles-own [energy]

to go
  move-turtles
  eat-grass
  tick
end
```

Usemos esta variable recién definida (`energy`) para permitir que las tortugas coman.

- Cambiar a la pestaña Código.
- Reescriba el procedimiento `go` de la siguiente manera:

```
to go
  move-turtles
  eat-grass
  tick
end
```

Agregue un nuevo procedimiento `eat-grass`:

```
to eat-grass
  ask turtles [
    if pcolor = green [
      set pcolor black
      set energy energy + 10
    ]
  ]
end
```

Estamos utilizando la instrucción if por primera vez. Mire el código cuidadosamente. Cada tortuga, cuando ejecuta estas instrucciones, evalúa el valor del color de la parcela, esto es, sobre (`pcolor`) con el valor `green`. (Una tortuga tiene acceso directo a las variables de la parcela en la que se encuentra). Si el color de la parcela es verde, la evaluación es verdadera (`true`), y solo entonces la tortuga ejecuta las instrucciones dentro de los paréntesis (de lo contrario, se lo salta). Las instrucciones hacen que la tortuga cambie el color de la parcela a negro y aumente su propia energía en 10. La parcela se vuelve negra para indicar que la hierba en ese punto se ha comido. Y a la tortuga se le provee más energía, después de haber comido.

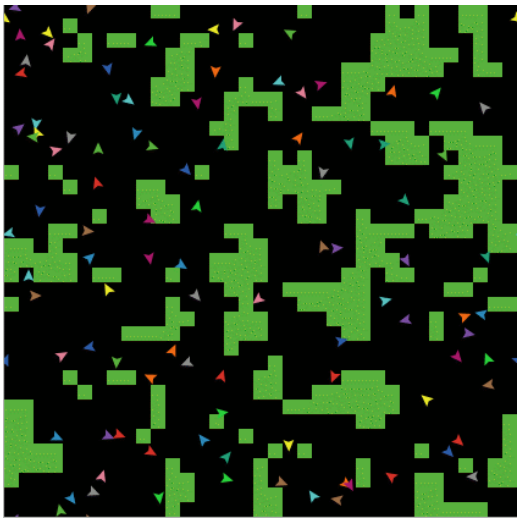
Luego, hagamos que el movimiento de las tortugas consuma parte de la energía de la tortuga.

- Reescriba `move-turtles` de la siguiente manera:

```
to move-turtles
  ask turtles [
    right random 360
    forward 1
    set energy energy - 1
  ]
end
```

A medida que vaga cada tortuga, perderá una unidad de energía en cada paso.

Cambie a la pestaña Ejecutar ahora y presione el botón de setup y el botón go. Verá que las parcelas se vuelven negras cuando las tortugas se desplazan sobre ellas.



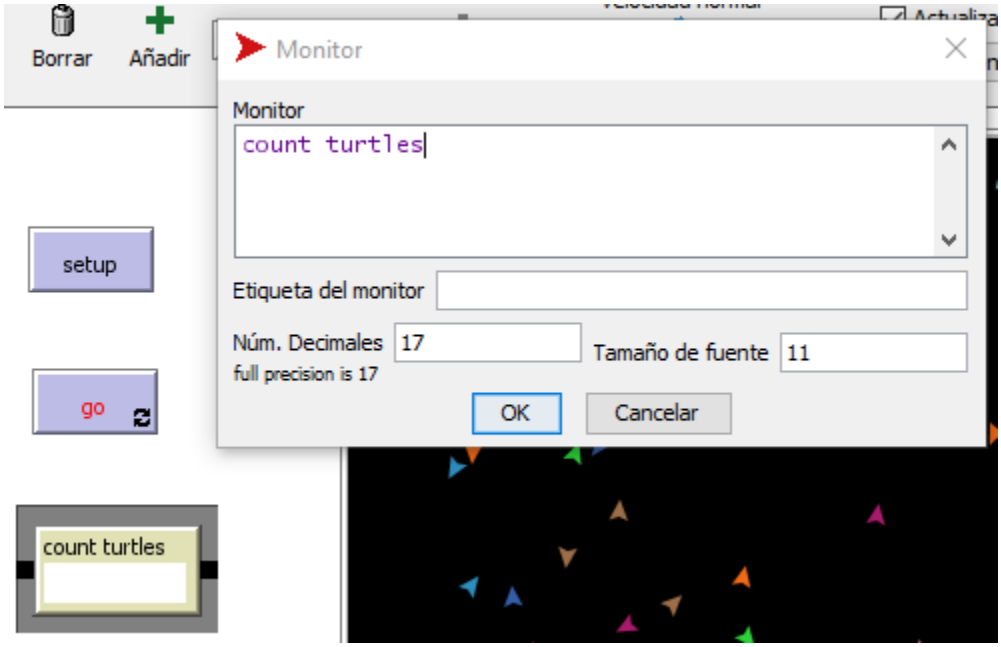
Monitores

A continuación, creará dos monitores en la pestaña Ejecutar con la barra de herramientas. (Los hará como botones y controles deslizantes, usando el ícono Añadir en la barra de herramientas.) Hagamos el primer monitor ahora.

- Cree un monitor haciendo clic en el ícono Añadir en la barra de herramientas, seleccionando Monitor al lado y haciendo clic en un lugar abierto en la interfaz.

Aparecerá un cuadro de diálogo:

- En el tipo de diálogo: `count turtles` (ver imagen a continuación).
- Presione el botón OK para cerrar el diálogo.



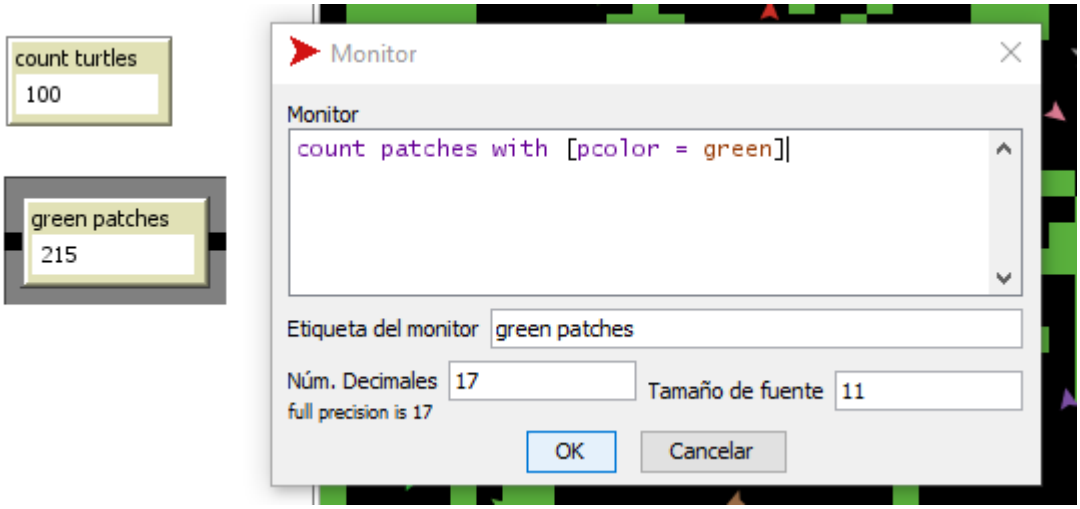
`turtles` es un "conjunto de agentes", el conjunto de todas las tortugas. `count` nos dice cuántos agentes hay en ese conjunto.

Hagamos el segundo monitor ahora:

- Cree un monitor haciendo clic en el ícono Añadir en la barra de herramientas, seleccionando Monitor al lado y haciendo clic en un lugar abierto en la Interfaz.

Un cuadro de diálogo aparecerá.

- En la sección Monitor del cuadro de diálogo, escriba: `count patches with [pcolor = green]` (ver imagen a continuación).
- En la sección Etiqueta del monitor del cuadro de diálogo, escriba: `green patches`
- Presione el botón OK para cerrar el cuadro de diálogo.



Aquí estamos usando conteo nuevamente para ver cuántos agentes hay en un conjunto de agentes. `patches` es el conjunto de todas las parcelas, pero no solo queremos saber cuántas parcelas hay en total, queremos saber cuántas de ellas son verdes. Eso es lo que `with` hace; hace un conjunto de agentes más pequeños de aquellos agentes para quienes la condición entre corchetes es verdadera. La condición es `pcolor = green`, por lo que nos da solo las parcelas verdes.

Ahora tenemos dos monitores que reportarán cuántas tortugas y parcelas verdes tenemos, para ayudarnos a rastrear lo que sucede en nuestro modelo. A medida que el modelo se ejecuta, los números en los monitores cambiarán automáticamente.

Use los botones setup y go y observe cómo cambian los números en los monitores.

Interruptores y etiquetas

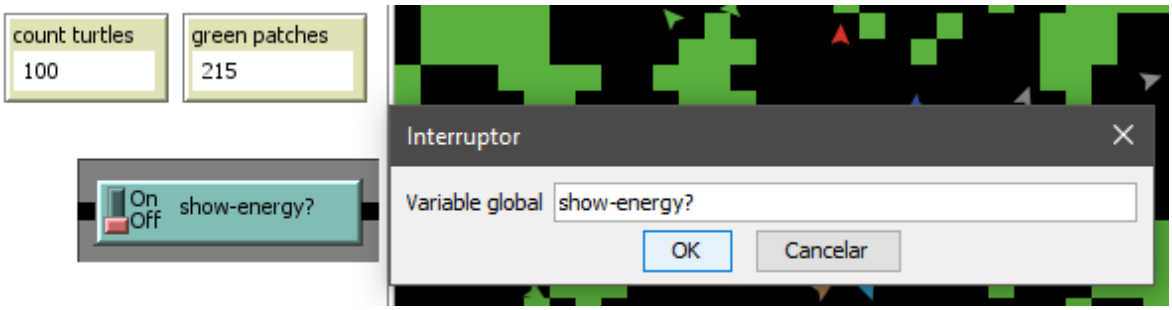
Las tortugas no solo están convirtiendo las parcelas en negras. También están ganando y perdiendo energía. A medida que el modelo se ejecuta, intente utilizar un monitor de tortuga para ver la energía de una tortuga subir y bajar.

Sería mejor si pudiéramos ver la energía de todas las tortugas todo el tiempo. Ahora haremos exactamente eso, y agregaremos un interruptor para poder activar y desactivar la información visual adicional.

- Haga clic en el ícono Añadir en la barra de herramientas (en la pestaña Ejecutar).
- Seleccione Interruptor en el menú al lado de Añadir.
- Haga clic en un lugar abierto en la interfaz.

Un cuadro de diálogo aparecerá:

En el campo de variable global, escriba `show-energy?` No olvide incluir el signo de interrogación en el nombre. (Ver imagen abajo)



Ahora regrese al procedimiento '`go`' usando la pestaña Código con la Barra de herramientas.

Vuelva a escribir el procedimiento de comer hierba como sigue:

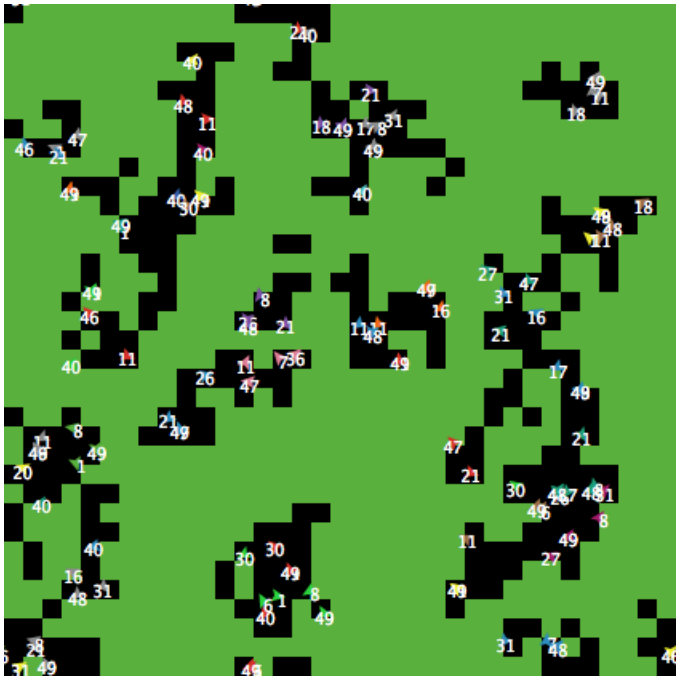
```
to eat-grass
  ask turtles [
    if pcolor = green [
      set pcolor black
      set energy energy + 10
    ]
    ifelse show-energy?
      [ set label energy ]
      [ set label "" ]
  ]
end
```

El procedimiento `eat-grass` introduce la instrucción `ifelse`. Mire el código cuidadosamente. Cada tortuga, cuando ejecuta estas nuevas instrucciones, verifica el valor de `show-energy?` (determinado por el interruptor). Si el interruptor está encendido, la comparación es verdadera y la tortuga ejecutará las instrucciones dentro del primer conjunto de corchetes. En este caso, asigna el valor de la energía a la etiqueta de la tortuga. Si la comparación es falsa (el interruptor está desactivado), la tortuga ejecuta las instrucciones dentro del segundo conjunto de corchetes. En este caso, elimina las etiquetas de texto (al configurar la etiqueta de la tortuga para que no sea nada).

(En NetLogo, una pieza de texto se llama "cadena", abreviatura de cadena de caracteres. Una cadena es una secuencia de letras u otros caracteres, escritos entre comillas dobles. Aquí tenemos dos comillas dobles una al lado de la otra, con nada entre ellos. Esa es una cadena vacía. Si la etiqueta de una tortuga es una cadena vacía, no se adjunta texto a la tortuga).

Pruebe esto en la pestaña Ejecutar, ejecutando el modelo (usando los botones `setup` y `go`) cambiando la energía de la demostración. Haga cambios de ida y vuelta.

Cuando el interruptor está encendido, verá que la energía de cada tortuga aumenta cada vez que come hierba. También verás que su energía disminuye cuando se mueve.



Más Procedimientos

Ahora nuestras tortugas están comiendo. Hagamos que se reproduzcan y mueran también. Y hagamos que crezca la hierba. Agregaremos los tres de estos comportamientos ahora, haciendo tres procedimientos separados, uno para cada comportamiento.

- Vaya a la pestaña Código.
- Vuelva a escribir el procedimiento `go` de la siguiente manera:

```
to go
  move-turtles
  eat-grass
  reproduce
  check-death
  regrow-grass
  tick
end
```

Agregue los procedimientos para reproducir, comprobar la muerte y volver a crecer la hierba como se muestra a continuación:

```
to reproduce
  ask turtles [
    if energy > 50 [
      set energy energy - 50
      hatch 1 [ set energy 50 ]
    ]
  ]
end

to check-death
  ask turtles [
    if energy <= 0 [ die ]
  ]
```

```
end

to regrow-grass
  ask patches [
    if random 100 < 3 [ set pcolor green ]
  ]
end
```

Cada uno de estos procedimientos usa la instrucción `if`. Cada tortuga, cuando se ejecuta `check-death`, verificará si su energía es menor o igual a 0. Si esto es cierto, se le dice a la tortuga `to die` (die es una primitiva NetLogo).

Cuando cada tortuga ejecuta `reproduce`, verifica el valor de la variable `energy` de la tortuga. Si es mayor que 50, entonces la tortuga ejecuta las instrucciones dentro del primer conjunto de corchetes. En este caso, disminuye la energía de la tortuga en 50, luego 'eclosiona' una nueva tortuga con una energía de 50. La instrucción `hatch` es una primitiva NetLogo que se ve así: `hatch` *número* [*instrucciones*]. Esta tortuga crea un número de tortugas nuevas, cada una idéntica a su padre, y les pide a las nuevas tortugas incubadas que ejecuten las instrucciones. Puede usar las instrucciones para dar a las nuevas tortugas diferentes colores, encabezados o lo que sea. En nuestro caso, ejecutamos una instrucción. Ponemos la energía en un valor de 50 para las tortugas recién eclosionadas.

Cuando cada parcela ejecuta `regrow-grass`, comprobará si un número entero aleatorio de 0 a 99 es menor que 3. De ser así, el color de la parcela se establece en verde. Esto ocurrirá el 3% del tiempo (en promedio) para cada parche, ya que hay tres números (0, 1 y 2) de cada 100 posibles que son menores que 3.

Cambie a la pestaña Ejecutar ahora y presione los botones setup y go.

Debería ver un comportamiento interesante en su modelo ahora. Algunas tortugas mueren, se crean nuevas tortugas (eclosionan) y algo de hierba vuelve a crecer. Esto es exactamente lo que nos propusimos hacer.

Si continúa mirando sus monitores en su modelo, verá que **count-turtles** y **green-patches** fluctúan. ¿Este patrón de fluctuación es predecible? ¿Hay una relación entre las variables?

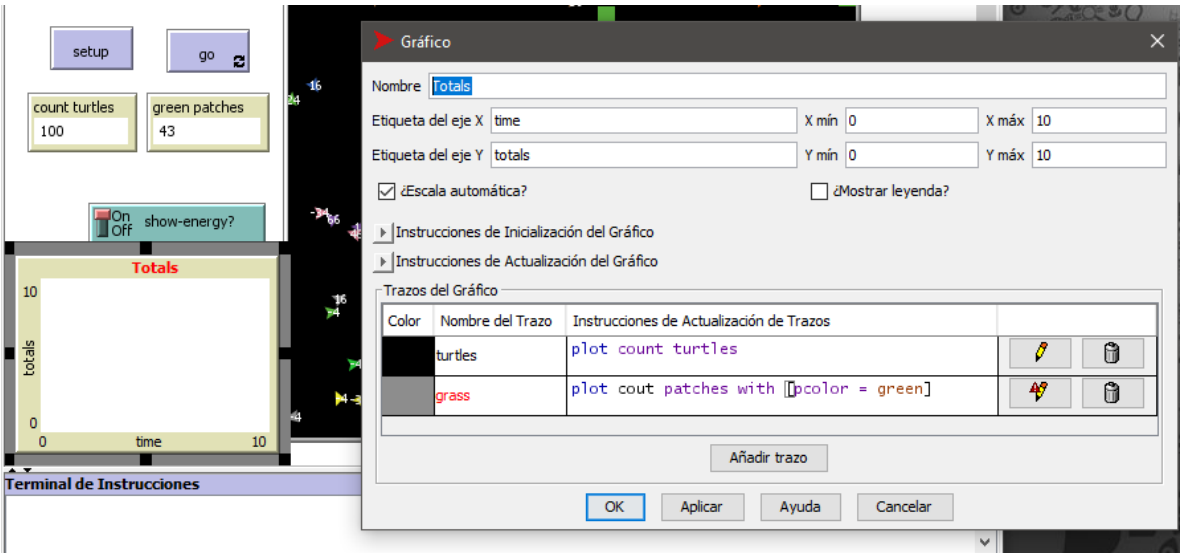
Sería bueno si tuviéramos una forma más fácil de seguir los cambios en el comportamiento del modelo a lo largo del tiempo. NetLogo nos permite graficar los datos a medida que avanzamos. Ese será nuestro siguiente paso.

Graficando (Plotting)

Para que el gráfico funcione, tendremos que crear un gráfico en la pestaña Ejecutar y colocar algunas instrucciones dentro de él. Las instrucciones que ponemos en las gráficas se ejecutarán automáticamente cuando nuestro procedimiento de setup llame a `reset-ticks` y cuando nuestro procedimiento `go` funcione.

- Cree un gráfico haciendo clic en el ícono Añadir en la barra de herramientas, seleccionando Gráfico al lado y haciendo clic en un lugar abierto en la Interfaz.
- Establezca su nombre en "Totals" (ver imagen a continuación)
- Establezca la etiqueta del eje X en "time"
- Establezca la etiqueta del eje Y en "totals"
- Cambie el nombre del rotulador por defecto "pen" por "turtles".
- Ingrese el conteo de las tortugas y de conteo de parcelas debajo de los Pen update commands.
- Presione el botón "Add pen".
- Cambie el nombre del nuevo pen a "grass".
- Ingrese los patches de recuento de parcelas con `[pcolor = green]` debajo de los Pen Update Commands.

Cuando termine, el diálogo debería verse así:



- Presione OK en el cuadro de diálogo Gráfico para finalizar la edición.

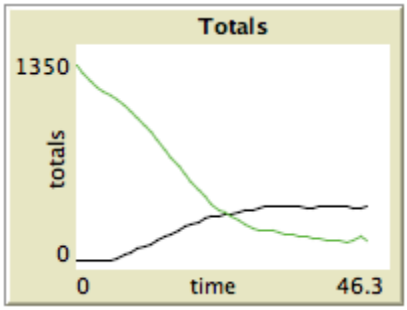
Tenga en cuenta que cuando crea el trazado también puede establecer los valores mínimo y máximo en los ejes X e Y. Deberá dejar marcada la casilla de verificación "Escala automática", de modo que, si alguna traza excede los valores mínimo y máximo para los ejes, los ejes crecerán automáticamente para que pueda ver todos los datos.

Considere que utilizamos la instrucción `plot` para agregar el siguiente punto a un gráfico. Esta instrucción mueve el lápiz de gráfico actual al punto que tiene una coordenada X igual a 1 mayor que la coordenada X previamente graficada y una coordenada Y igual al valor dado en la instrucción de grafico (en el primer caso, el número de tortugas, y en el segundo caso, el número de parcelas verdes). A medida que los bolígrafos se mueven, cada uno traza una línea.

- Configure y ejecute el modelo nuevamente.

Ahora puede ver la gráfica que se dibuja mientras el modelo se está ejecutando. Su forma debe tener la forma general como la siguiente, aunque su gráfica puede no ser exactamente igual.

Recuerde que dejamos "**¿Escala automática?**" encendido. Esto permite que la gráfica se ajuste en tamaño cuando se agote el espacio.



Si olvida cuál lápiz es cuál, puede editar el gráfico y marcar la casilla de verificación "¿Mostrar leyenda?".

Puede intentar ejecutar el modelo varias veces para ver qué aspectos de la gráfica son los mismos y cuáles son diferentes de ejecución en ejecución.

Cuenta Pasos

Para hacer comparaciones entre gráficos de un modelo ejecutado y otro, a menudo es útil hacer la comparación para la misma duración de ejecución del modelo. Aprender cómo detener o iniciar una acción en un momento específico puede ayudar a que esto suceda al detener el modelo en el mismo punto en que se ejecuta cada modelo. Hacer un seguimiento

de cuántas veces se ejecuta el procedimiento `go` es una forma útil de indicar estas acciones. Eso es lo que hace el contador de ticks.

Ya está usando el contador de ticks en su modelo, con las instrucciones `reset ticks` y `tick`, que también desencadenan el gráfico.

También puede usar el contador de ticks para otras cosas, como establecer un límite en la duración total de una ejecución.

- Cambie el procedimiento de go:

```
to go
  if ticks >= 500 [ stop ]
  move-turtles
  eat-grass
  check-death
  reproduce
  regrow-grass
  tick
end
```

- Ahora configure (setup) y ejecute (go) el modelo.

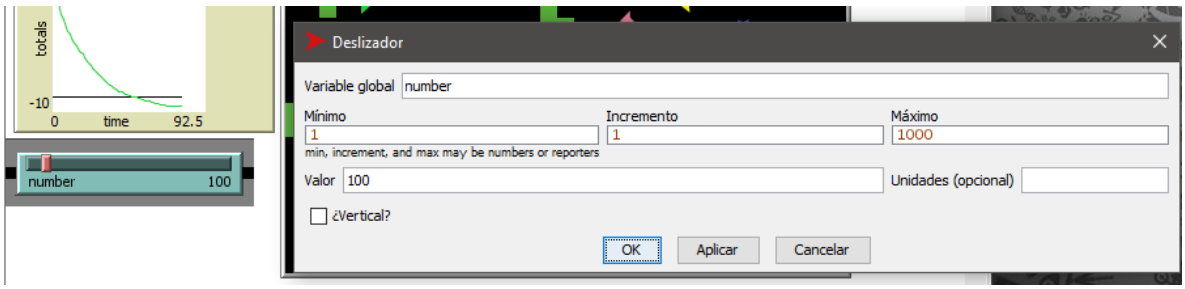
El gráfico y el modelo no seguirán funcionando continuamente. Deben detenerse automáticamente cuando el contador de ticks en la barra de herramientas de la pestaña Ejecutar alcance el valor de 500.

La instrucción `tick` avanza el contador de ticks en 1. `ticks` es un reportero que informa el valor actual del contador de ticks. `reset-ticks`, en su procedimiento de configuración (`setup`), se encarga de reiniciar el contador de ticks en 0 cuando se configura una nueva ejecución y está lista para comenzar.

Algunos detalles más

Primero, en lugar de usar siempre 100 tortugas, puede tener un número variable de tortugas.

Cree un control deslizante llamado "`number`": haga clic en el icono Añadir en la barra de herramientas, seleccione Deslizador al lado y haga clic en un lugar abierto en la interfaz. Intente cambiar los valores mínimo y máximo en el Deslizador.



Luego, dentro de `setup-turtles`, en vez de `create-turtles 100` puede escribir:

```
to setup-turtles
  create-turtles number [ setxy random-xcor random-ycor ]
end
```

Pruebe este cambio y compare cómo el hecho de tener más o menos tortugas afecta inicialmente las parcelas a lo largo del tiempo.

En segundo lugar, ¿no sería agradable ajustar la energía que las tortugas ganan y pierden cuando comen hierba y se reproducen?

- Haga un Deslizador llamado energy-from-grass.

- Haga otro Deslizador llamado birth-energy.
- Luego, dentro de `eat-grass`, haga este cambio:

```
to eat-grass
  ask turtles [
    if pcolor = green [
      set pcolor black
      set energy (energy + energy-from-grass)
    ]
    ifelse show-energy?
      [ set label energy ]
      [ set label "" ]
  ]
end
```

Y, dentro de `reproduce`, haga este cambio:

```
to reproduce
  ask turtles [
    if energy > birth-energy [
      set energy energy - birth-energy
      hatch 1 [ set energy birth-energy ]
    ]
  ]
end
```

Finalmente, ¿qué otro Deslizador podría agregar para variar la frecuencia con la que la hierba vuelve a crecer? ¿Hay reglas que pueda agregar al movimiento de las tortugas o a las tortugas recién nacidas que ocurren solo en ciertos momentos? Intente escribirlos.

¿Qué sigue?

Entonces ahora que se tiene un modelo simple de un ecosistema. En las parcelas crecen hierba. Las tortugas deambulan, comen la hierba, se reproducen y mueren.

Ha creado una interfaz que contiene botones, deslizadores, interruptores, monitores y un gráfico. Incluso ha escrito una serie de procedimientos para darles algo que hacer a las tortugas.

Aquí es donde este tutorial finaliza.

Si desea consultar más documentación sobre NetLogo, la sección Interface Guide del manual lo guía por cada elemento de la interfaz de NetLogo en orden y explica su función. Para obtener una descripción detallada y detalles sobre los procedimientos de escritura, consulte la Guía de Programación. Todas las primitivas se enumeran y describen en el NetLogo Dictionary.

Además, puede seguir experimentando y expandiendo este modelo si lo desea, experimentando con diferentes variables y comportamientos para los agentes.

Alternativamente, es posible que desee volver a visitar el primer modelo en el tutorial, Wolf Sheep Predation. Este es el modelo que usó en el Tutorial n. ° 1. En el modelo Wolf Sheep Predation, observó ovejas que se movían, consumían recursos que se reponen ocasionalmente (hierba), se reproducían bajo ciertas condiciones y morían si se les acababan los recursos. Pero ese modelo tenía otro tipo de criatura en movimiento: lobos. La adición de lobos requiere algunos procedimientos adicionales y algunas primitivas nuevas. Lobos y ovejas son dos "razas" diferentes de tortuga. Para ver cómo usar las razas, estudie el modelo Wolf Sheep Predation.

Alternativamente, puede mirar otros modelos (incluidos los muchos modelos en la sección Code Examples de la Biblioteca de modelos) o incluso seguir adelante y crear su propio modelo. Ni siquiera tiene que modelar nada. Puede ser interesante simplemente ver parcelas y tortugas formando patrones, para tratar de crear un juego para jugar, o lo que sea.

Esperamos que haya aprendido algunas cosas, tanto en términos del lenguaje NetLogo como sobre cómo construir un modelo. El conjunto completo de procedimientos que se creó arriba se muestra a continuación.

Apéndice: Código Completo

El modelo completo también está disponible en la Biblioteca de Modelos de NetLogo, en la sección de Ejemplos de Código. Se llama "Tutorial 3".

Tenga en cuenta que esta lista está llena de "comentarios", que comienzan con punto y coma. Los comentarios le permiten mezclar una explicación del código directamente con el código en sí. Puede usar comentarios para ayudar a otros a entender su modelo, o puede usarlos como notas para usted mismo.

En la pestaña Código, los comentarios son grises, por lo que sus ojos pueden seleccionarlos fácilmente.

```
turtles-own [energy] ;; for keeping track of when the turtle is ready
                     ;; to reproduce and when it will die

to setup
  clear-all
  setup-patches
  setup-turtles
  reset-ticks
end

to setup-patches
  ask patches [ set pcolor green ]
end

to setup-turtles
  create-turtles number    ;; uses the value of the number slider to create turtles
  ask turtles [ setxy random-xcor random-ycor ]
end

to go
  if ticks >= 500 [ stop ]  ;; stop after 500 ticks
  move-turtles
  eat-grass
  check-death
  reproduce
  regrow-grass
  tick                    ;; increase the tick counter by 1 each time through
end

to move-turtles
  ask turtles [
    right random 360
    forward 1
    set energy energy - 1  ;; when the turtle moves it looses one unit of energy
  ]
end

to eat-grass
  ask turtles [
    if pcolor = green [
      set pcolor black
           ;; the value of energy-from-grass slider is added to energy
      set energy energy + energy-from-grass
    ]
  ifelse show-energy?
    [ set label energy ] ;; the label is set to be the value of the energy
    [ set label "" ]     ;; the label is set to an empty text value
  ]
```

```
end

to reproduce
  ask turtles [
    if energy > birth-energy [
      set energy energy - birth-energy  ;; take away birth-energy to give birth
      hatch 1 [ set energy birth-energy ] ;; give this birth-energy to the offspring
    ]
  ]
end

to check-death
  ask turtles [
    if energy <= 0 [ die ] ;; removes the turtle if it has no energy left
  ]
end

to regrow-grass
  ask patches [ ;; 3 out of 100 times, the patch color is set to green
    if random 100 < 3 [ set pcolor green ]
  ]
end
```