

UNIVERSIDAD DE LOS LLANOS

INGENIERÍA DE SISTEMAS
SIMULACIÓN COMPUTACIONAL

NETLOGO

Autor:

MARIELENA AYALA

Docente:

PhD. Angel Cruz Roa

Abril 14, 2016



Abstract

Netlogo es un entorno de programación que permite la simulación de fenómenos naturales y sociales. Fue creado por Uri Wilensky en 1999 y está en continuo desarrollo por el Center for Connected Learning and Computer-Based Modeling.

Netlogo es particularmente útil para modelar sistemas complejos que evolucionan en el tiempo. Los implementadores de modelos pueden dar instrucciones a cientos o miles de agentes para que todos ellos operen de manera independiente, entre sí y con el entorno. Esto hace posible explorar la relación entre el comportamiento a bajo nivel de los individuos y los patrones macroscópicos que surgen a partir de la interacción de muchos individuos entre sí.

Netlogo permite a los usuarios abrir simulaciones y “jugar” con ellas, así como explorar su comportamiento bajo una serie de condiciones. Asimismo, permite al usuario la creación de sus propios modelos. Netlogo es lo suficientemente sencillo como para que los estudiantes y los profesores puedan ejecutar las simulaciones o incluso construir las suyas propias. Además, su grado de desarrollo actual es suficiente como para servir como una herramienta potente para investigadores en muchos ámbitos.

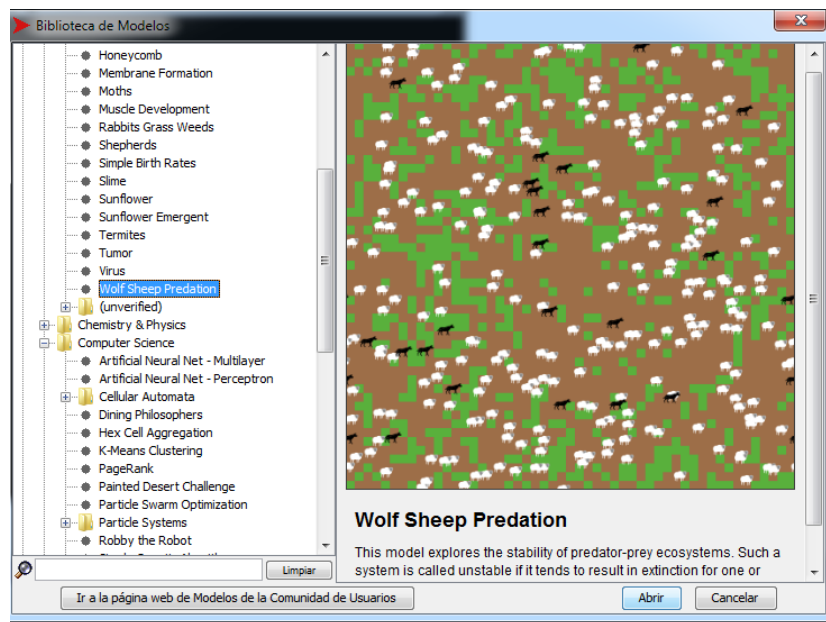
1 Tutorial: Depredación Lobo Oveja

1.1 Resumen

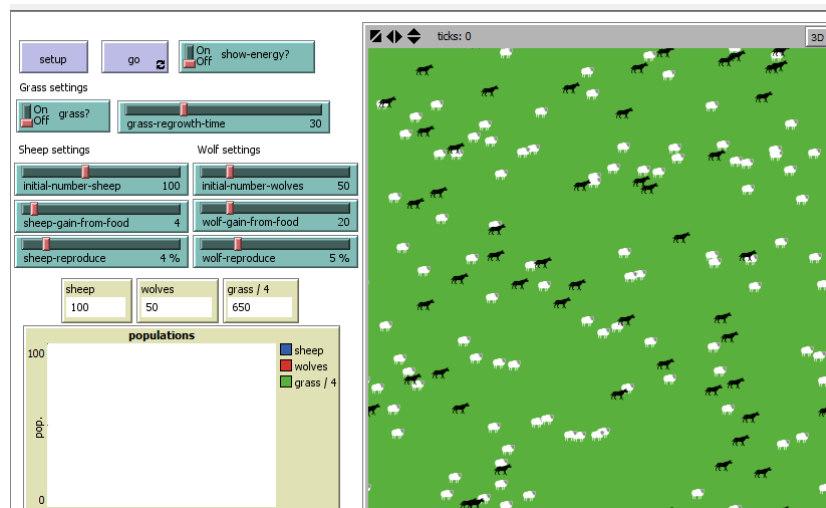
El modelo depredación lobo-oveja muestra como mediante condiciones iniciales se puede ver el futuro de vida ambas especies. El numero inicial de ovejas, lobo y césped juegan un papel fundamental en los resultados. Estos comportamientos no pueden estudiarse individualmente, es decir ovejas y lobos por aparte, ya que, lo importante es ver la población de agentes como un todo.

1.2 Inicio del tutorial

- Para iniciar, se abre la biblioteca de modelos, Se elige la categoría biology, y posteriormente Wolf Sheep Predation:

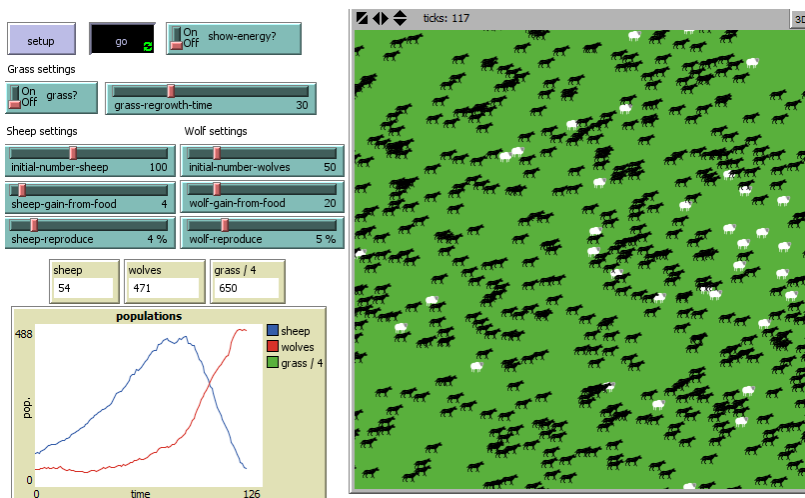


- Al presionar el botón “setup” se muestra en la parte lateral derecha, una multitud de ovejas y lobos en posiciones aleatorias.

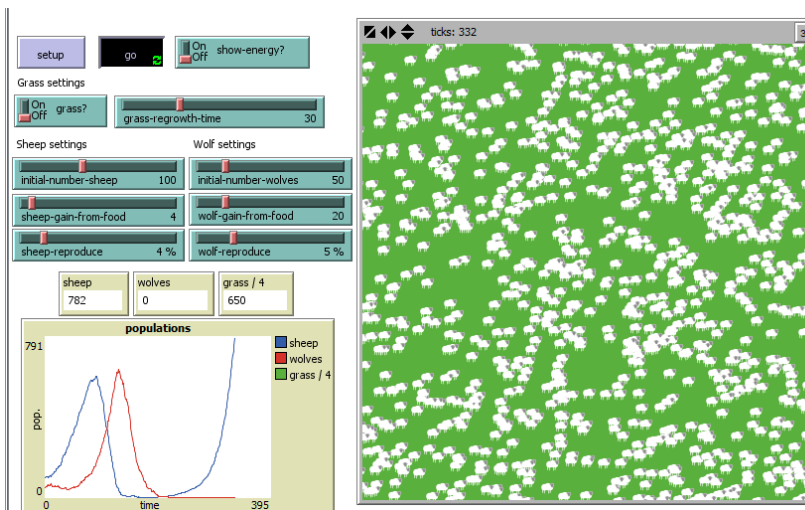


- Posteriormente se presiona el botón “go” que se encuentra en la parte lateral izquierda de la ventana. Esta acción permitirá que se inicie la simulación. Al iniciar la simulación se puede observar que los lobos

devoran a las ovejas limitándolas en cantidad.

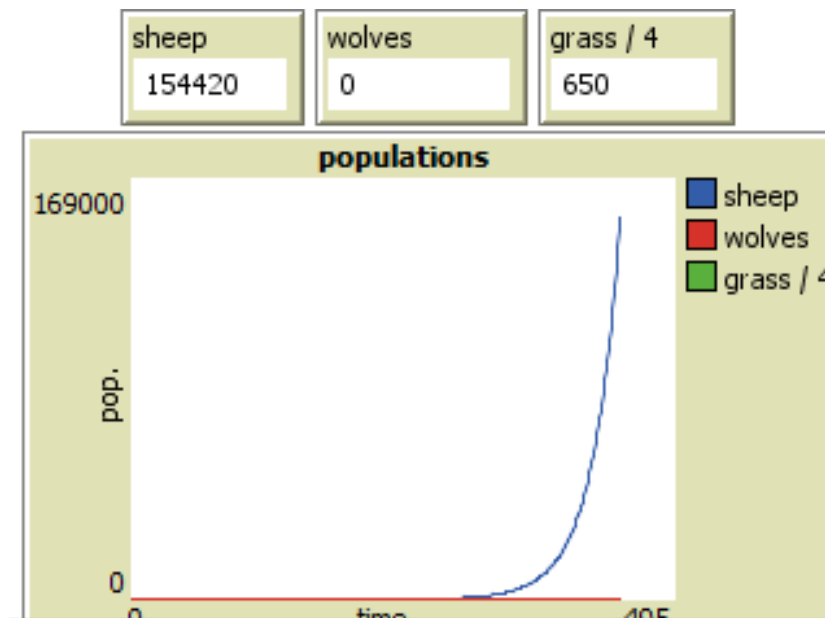


- Transcurrido el tiempo, los lobos se mueren de hambre por la ausencia de alimento (ovejas) y terminan extinguiéndose. En este punto se inicia un proceso contrario, lo cual es la reproducción de las ovejas



- En ese momento se cuentan con 154420 ovejas, las cuales tienden a aumentar al pasar el tiempo

La siguiente grafica muestra la reproducción de las ovejas, estas tienden a infinito.



- ¿Alguna vez obtendrá resultados diferentes si ejecuta el modelo en repetidas ocasiones manteniendo la misma configuración?

- Siempre se obtendrán diferentes modelos, dependiendo del número de ovejas iniciales, el número de lobos y la cantidad de hierba que el usuario ingrese antes de la simulación. También se cuenta, con factores como el tiempo en que la hierba crece, importantísima para la supervivencia de las ovejas.

A continuación se muestran los resultados después de cambiar la configuración de los slides y switches:

- ¿Qué pasó con las ovejas a través del tiempo?
 - La población de ovejas va desapareciendo.
- ¿Qué le hizo este switch al modelo? ¿Fue el mismo resultado de la ejecución previa?
 - El switch en modo OFF, elimina automáticamente el césped, quitándole a las ovejas uno de los factores fundamentales para la supervivencia, por tanto desaparecen rápidamente de la población en general.

- **¿Qué sucedería con la población de ovejas si hay al comienzo de la simulación inician más ovejas y menos lobos?**

- Las ovejas se reproducirán rápidamente.

Para ver diferentes cambios se aplicara la siguiente configuración: Apague "grass?". Establezca el slider del número inicial de ovejas ("initial-number-sheep") a 100. Establezca el slider del número inicial de lobos ("initial-number-wolves") a 20. Presione "setup" y luego "go". Permita que el modelo corra alrededor de 100 ticks de tiempo

- **Qué le ocurrió a la población de ovejas?**

- Inicialmente las ovejas se reproducen rápidamente, pero después tienden a desaparecer llegando a extinguirse completamente.

- **¿Le sorprendió este resultado?, ¿Qué otros sliders o switches se pueden ajustar para ayudarle a la población de ovejas?**

- El porcentaje de reproducción.

La siguiente configuración es: Ajuste el número inicial de ovejas a 80 y el número inicial de lobos a 50. (Esto es cercano a la forma en que estaban cuando usted abrió el modelo por primera vez.) Fije "sheep-reproduce" en 10,0. Presione "setup" y luego "go". Permita que el modelo corra alrededor de 100 ticks de tiempo.

- **¿Qué le pasó a los lobos en esta ejecución?**

- Los lobos de reproducen infinitamente.

Ahora se verá el control de vista: Presione "setup" y luego "go" para iniciar la ejecución del modelo. A medida que corra el modelo, mueva el slider de la velocidad a la izquierda.

- **¿Qué sucede?**

- La simulación disminuye de velocidad.

Mueva el slider de velocidad a la mitad. Pruebe moviendo el slider de la velocidad a la derecha. Ahora intente marcando y desmarcando la casilla de verificación de las actualizaciones de la vista (view updates).

- **¿Qué sucede?**
 - La velocidad de la simulación aumenta, y al desmarcar la casilla la vista de congela, pero el modelo sigue corriendo en segundo plano.
- **¿Cuáles son los ajustes actuales para max-pxcor, pxcor-min, max-pycor, min-pycor, y patch size (tamaño del parche)?**
 - 25, -25, -25, 25 y 9 respectivamente.

Arrastre una de las "asas" cuadradas negras. Las asas se encuentran en los bordes y en las esquinas de la vista. Deseleccione la vista haciendo clic en cualquier lugar del fondo blanco de la Interfaz. Pulse de nuevo el botón "Settings..." y vea los ajustes.

- **¿Qué números cambiaron?**
 - Cambia los números del tamaño de la parcela.
- **¿Qué números no cambiaron?**
 - No cambiaron los datos de max-pxcor, pxcor-min, max-pycor, min-pycor, y patch size.

Utilizando el diálogo de Model Settings que aun sigue abierto, cambie max-pxcor a 30 y el valor de max-pycor a 10. Observe que min-pxcor min-pycor también cambian. Esto se debe a que por defecto el origen (0,0) está en el centro del mundo.

- **¿Qué le ocurrió a la forma de la vista?**
 - Cambio el ancho.
- **¿Qué pasó con el tamaño de la vista?, ¿cambió esto su forma?**
 - Cambio el tamaño en general.

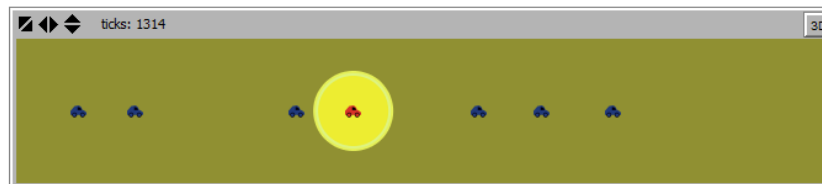
2 Tutorial: Comandos

2.1 Resumen

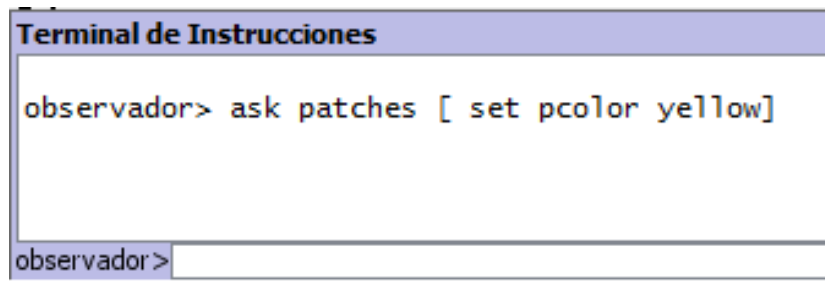
Para este tutorial se escoge el modelo “trafico básico”, que se ubica en la categoría de ciencias sociales. Este modela la manera de cómo pueden formarse atascos en el tráfico sin que exista alguna causa tal como un accidente, un puente roto, o un camión volcado. No es necesaria una “causa central” para que se forme un embotellamiento de tráfico.

2.2 Inicio del tutorial

- En la terminal de instrucciones escriba: `ask patches [set pcolor yellow]`. ¿Qué le pasó a la vista?
 - Cambio su aspecto, y se creó un círculo amarillo para observar al carro rojo, además calle desapareció. El cambio lo describe la siguiente imagen:



- ¿Por qué los coches no se cambiaron también a amarillo?
 - Observando el comando que fue escrito, sólo le pedimos a los parches cambiar su color. En este modelo, los coches están representados por un tipo diferente de agente, llamados “turtles” (tortugas). Por lo tanto, los coches no recibieron estas instrucciones y por ese motivo no cambiaron.
- ¿Qué ocurrió en el Centro de Comando?
 - Se debe notar que el comando que se acabó de escribir, ahora está desplegado en el cuadro blanco en medio de la terminal de instrucciones.



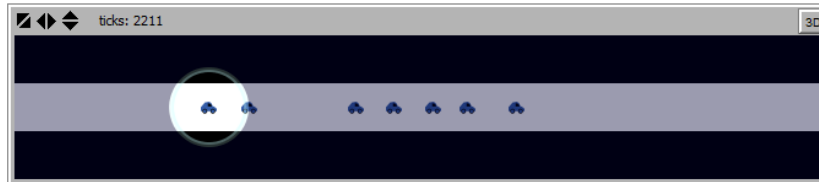
```
Terminal de Instrucciones

observador> ask patches [ set pcolor yellow]

observador> 
```

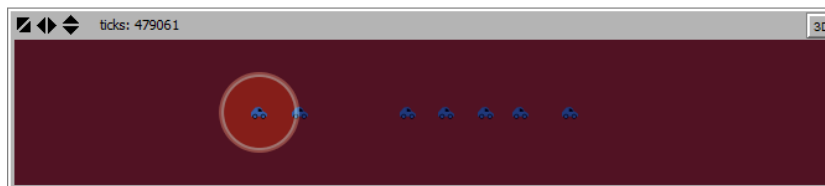
- **Escriba el siguiente comando: ask turtles[set color brown]. Fue el resultado de lo que esperaba?**
 - Si. Los coches cambiaron de color a café.

Elija "turtles" ("tortugas") en el menú emergente. Escriba set color pink y pulse retorno. Pulse la tecla de tabulación hasta que vea "patches;" en la esquina inferior izquierda. Escriba set pcolor white y pulse retorno.
- **¿Cómo luce ahora la vista?**
 - Los coches y el fondo han cambiado.
- **¿Nota alguna diferencia entre estos dos comandos y los comando del observador anterior?**
 - El observador supervisa el mundo y, por tanto, puede dar un comando a los parches o las tortugas utilizando ask. Pero cuando un comando está dado directamente a un grupo de agentes , sólo setiene que escribir el comando sin especificar.
- **Presione setup. ¿Qué pasó?**
 - La vista volvió a la versión antigua. Al pulsar el botón setup el modelo se reconfigura a si mismo y se vuelve a los ajustes predefinidos. La terminal de instrucciones no es a menudo usada para cambiar permanentemente el modelo, es más para usarse como una herramienta y personalizar los modelos actuales

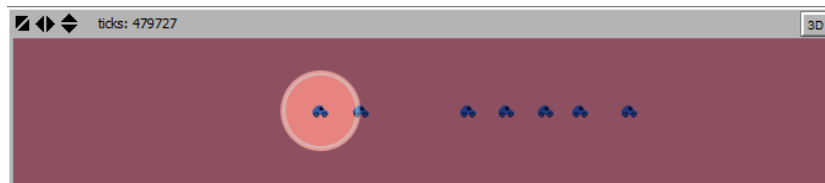


- **¿Cuál es la diferencia entre el color y pcolor?** Algunos comandos y variables son específicos para las tortugas y algunos para los parches. Por ejemplo, la variable color es una variable de tortuga, mientras que pcolor es una variable de parche.

Se elige "patches" en el menú desplegable en el Centro de Comandos y posteriormente escriba `set pcolor rojo - 2`.

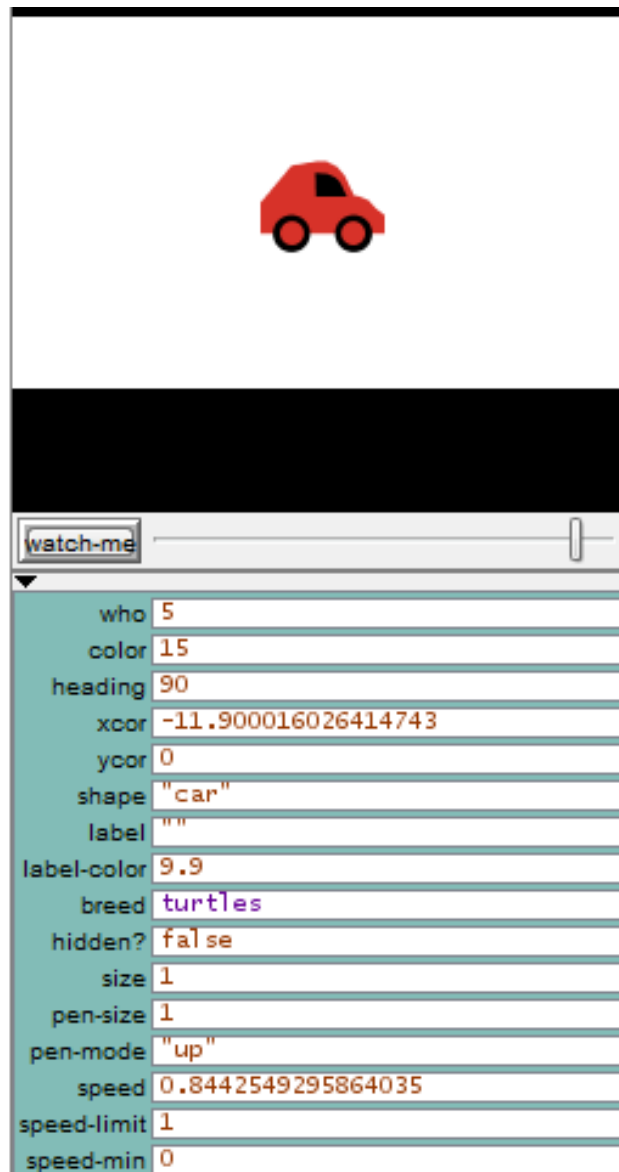


Luego se escribe `set pcolor red + 2`. El rojo se vuelve mas claro.



Inspeccione encima del coche y de clic derecho. Aparecerá inspeccionar turtle.

- **¿Cuál es el who number de la tortuga?**
 - El numero 6.
- **¿De qué color es esta tortuga?**
 - Color rojo.
- **¿De qué forma es esta tortuga?**



– Forma de automovil.

En el Comandante de Agente del monitor de turtle escriba set color pink para la tortuga 0.

- ¿Qué sucede en la vista? ¿Cambi3 algo en el monitor de la tortuga?

- La tortuga 0 cambia a color rosado.

Seleccione el texto a la derecha de "color" en el Monitor de Tortuga. Escriba un nuevo color como `green + 2`.

- **¿Qué pasó?**

- Cambio a color verde claro.

En el Centro de Comando, seleccione "observador" en el menú desplegable (o utilice la tecla de tabulación). Escriba `ask turtle 0 [set color blue]` y pulse retorno.

- **¿Qué sucede?**

- La tortuga 0 cambia de color.

- **¿Puede hacer un monitor de parche y utilizarlo para cambiar el color de un solo parche?**

- Si, indicándole las coordenadas del parche que quiere cambiar de color. Coordenada x y coordenada y. por ejemplo `ask patch -11 -4 [set pcolor green]`.

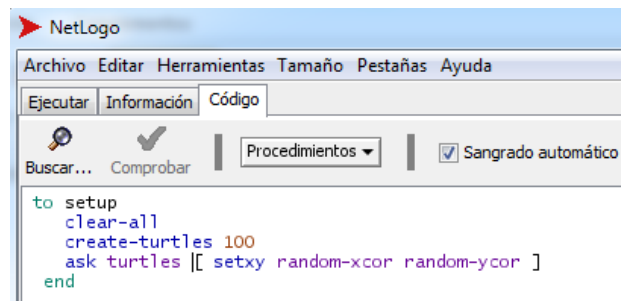
3 Tutorial: Procedimientos

3.1 Resumen

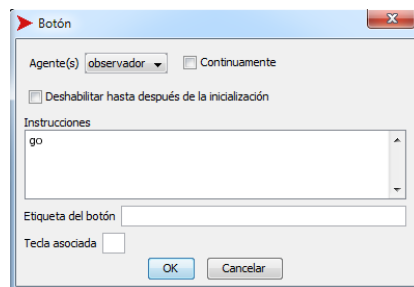
Se puede crear un modelo desde cero, añadiéndole unos determinados procedimientos. Estos manipularan el comportamiento de cada agente y les dirá que hacer, también puede mostrar al usuario datos valiosos para la comprensión del proceso.

3.2 Inicio del tutorial

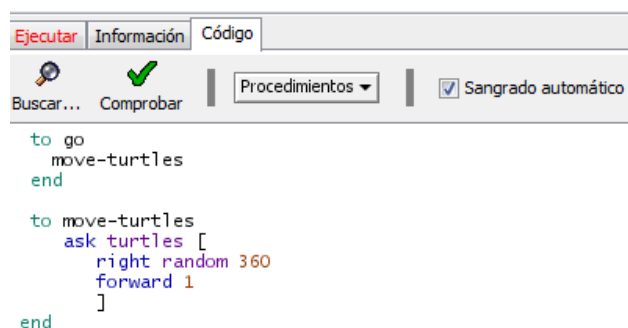
Se crea un botón llamado setup. Posteriormente en la pestaña código se ingresa el siguiente código. Para confirmar se da clic en el botón “comprobar”.



Se ingresa a la pestaña ejecutar y se da clic en “setup”. El botón ya no es de color rojo ya que hay un procedimiento creado para esté. Al dar clic varias veces en el botón setup, se generaran posiciones diferentes de los agentes. Luego se crea un botón llamado “go”.



Se hace clic en “continuamente” para que siempre ejecute este procedimiento, el cual es el siguiente:

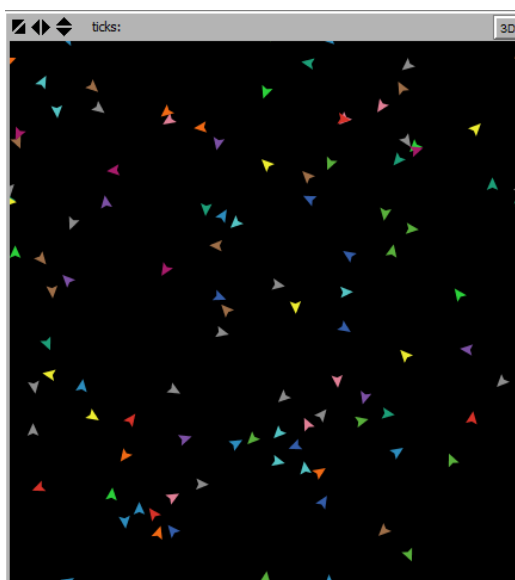


The screenshot shows a code editor window with three tabs: 'Ejecutar' (highlighted in red), 'Información', and 'Código'. Below the tabs is a toolbar with a magnifying glass icon labeled 'Buscar...', a green checkmark icon labeled 'Comprobar', a dropdown menu labeled 'Procedimientos', and a checkbox labeled 'Sangrado automático' which is checked. The code area contains the following text:

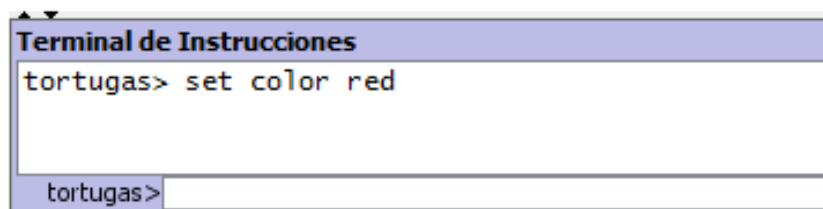
```
to go
  move-turtles
end

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

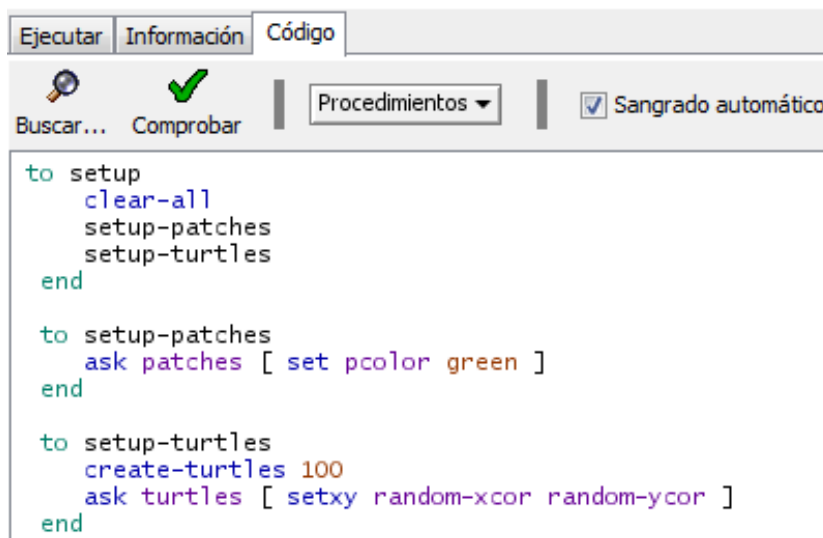
Al presionar el botón go, se ejecutara el procedimiento que se acabó de ingresar, la línea ask turtles dice que cada tortuga debe ejecutar los comandos que se encuentran en los corchetes [], right random 360 hace que a cada una de ellas le pertenezca un numero entre 1 y 359 aleatoriamente y se mueva a la derecha según el número de grados que le halla correspondido. Finalmente forward 1 hace que la tortuga avance un paso hacia adelante.



A continuación se escribe en el centro de comandos lo siguiente:



Entonces, todas las tortugas se adaptaran al color rojo. Posteriormente se modifica el procedimiento del botón “setup”, creado en el inicio del tutorial.



Al ejecutar el procedimiento y dar clic en el botón, se puede ver el siguiente resultado:



Se quiere ahora controlar cuando muere una tortuga y cuando vive, por ese motivo se creara una variable llamada energía que controlara lo dicho anteriormente. Se crea un código al inicio, antes de los procedimientos que ya existen.

Luego se programa la interfaz para que muestre cuando la tortuga ha comido y automáticamente la cambie. Si la tortuga esta parada sobre el césped (verde) entonces que se vuelva negro, para indicar que la tortua se ha comido la hierba en esa posición.

También se desea crear un procedimiento que simule la disminución de la energía. Para ello cada que la tortuga se mueva se le ira disminuyendo la energía 1 unidad.

La siguiente imagen muestra los códigos

```
turtles-own [energy]

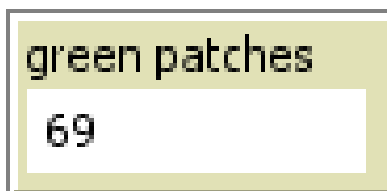
to go
  move-turtles
  eat-grass
end

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

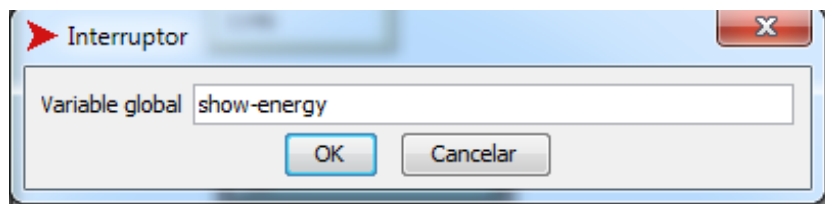
to move-turtles
  ask turtles [
    right random 360
    forward 1
    set energy energy - 1 ]
end
```


A continuación se crea un monitor ubicado en la barra de herramientas. Se le asigna el nombre “count turtles”.

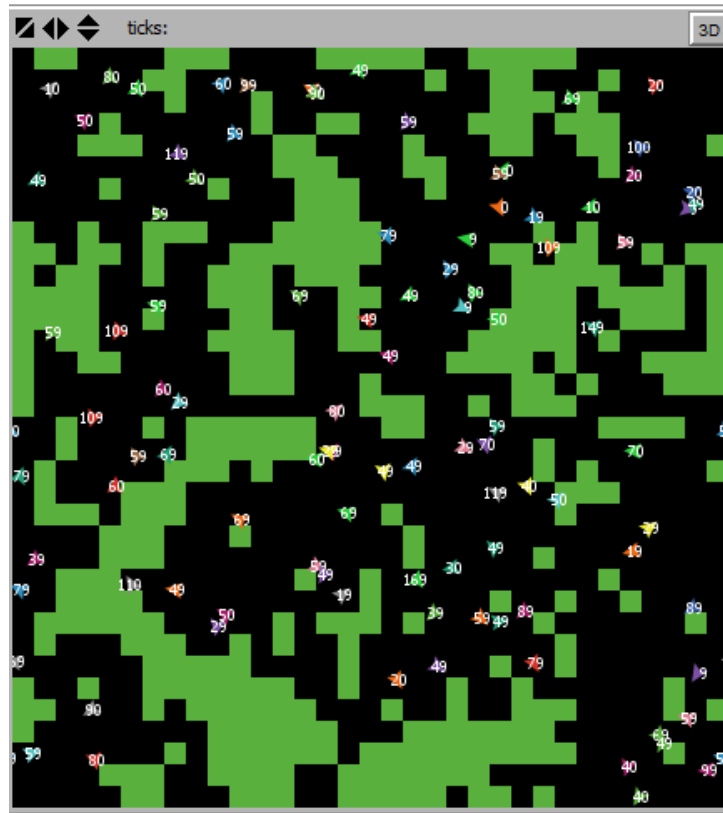
A este monitor se le asigna una función que hara que cuente cada uno de los parches verdes. El código es el siguiente: `count patches with [pcolor = green]`. Y al ejecutarse se mostrara algo como lo siguiente (en ese momento hay 69 parches verdes):



Para crear un interruptor se realiza el mismo procedimiento. Se le asignara el siguiente nombre.



Luego de modificar el procedimiento `eat-grass`. Cuando el interruptor está encendido, verá la energía de cada tortuga incrementarse cada vez que come hierba. También verá su energía disminuir cada vez que se mueve.



En este momento las tortugas están comiendo; se hará entonces que también se reproduzcan y mueran. Además se hará que la hierba rebrote. Se añade tres comportamientos haciendo tres procedimientos separados. Uno para cada comportamiento.

```

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

to check-death ;; verificar muerte
  ask turtles [
    if energy <= 0 [ die ]
  ]
end

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

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

to check-death ;; verificar muerte
  ask turtles [
    if energy <= 0 [ die ]
  ]
end

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

```

Graficación: Cree un gráfico, utilizando el icono plot de la barra de herramientas y haga clic en un lugar abierto en la interfaz. Establezca su nombre como "Totals" (ver imagen inferior) Establezca el eje de las X con la etiqueta "time". Establezca el eje Y con etiqueta "total".

```

to do-plots
  set-current-plot "Totals"
  set-current-plot-pen "turtles"
  plot count turtles
  set-current-plot-pen "grass"
  plot count patches with [pcolor = green]
end

```

Para más información se puede crear un tick que añadirá 1 al contador que se lleve. tick es un reportero que informa el valor actual del contador de tics.