

Laboratorio 4
Simulación basada en agentes con NetLogo

Brenda Enith Velásquez Mayorga - 160003041

Sharik Natalia Amaya Rey - 160003301

Hover Alexander Fuquene Pinzón - 160004015

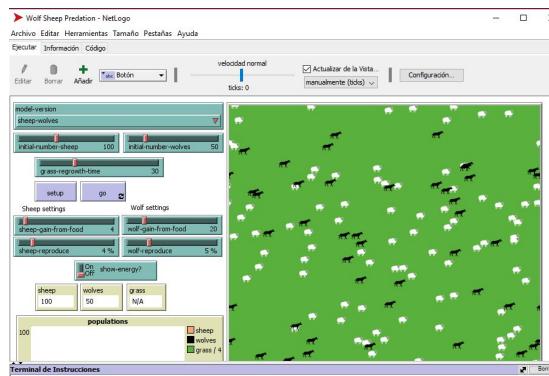
Ing. Angel Alfonso Cruz Roa

Simulación computacional
Ingeniería de sistemas
Universidad de los Llanos

Tutorial # 1. Modelos

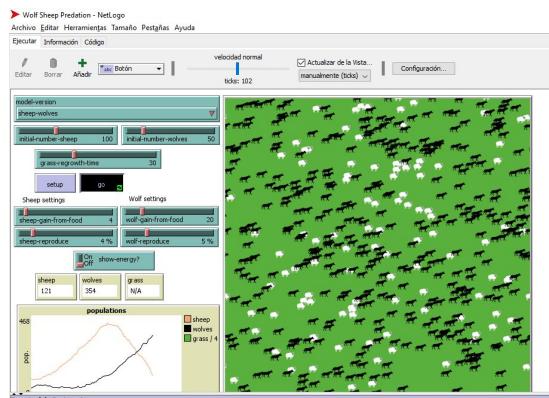
-¿Qué ve aparecer en la vista?

R: Aparece una vista de color verde llena de ovejas y lobos.

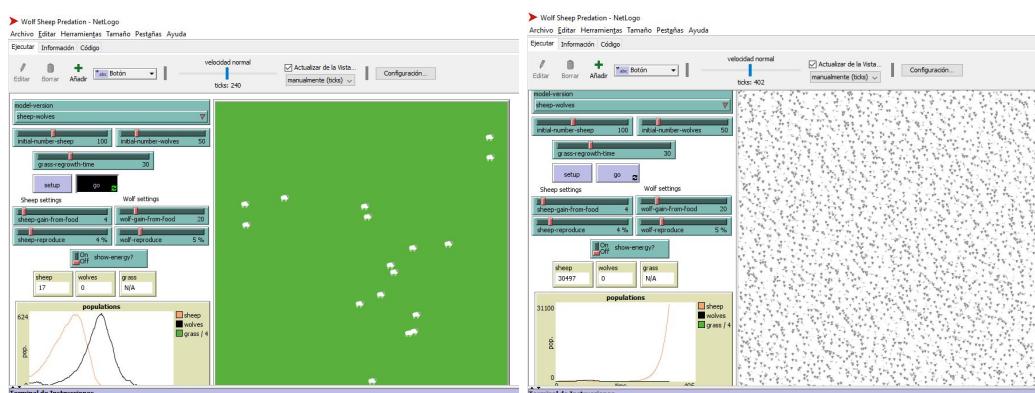


-A medida que el modelo se ejecuta, ¿qué está pasando con las poblaciones de lobos y ovejas?

R: A medida que la simulación se ejecuta, en este caso la población de ovejas se fue incrementando en un lapso de tiempo, mientras que la población de lobos se mantenía baja, después la población de lobos incrementó de manera considerable haciendo que la población de ovejas fuera disminuyendo.



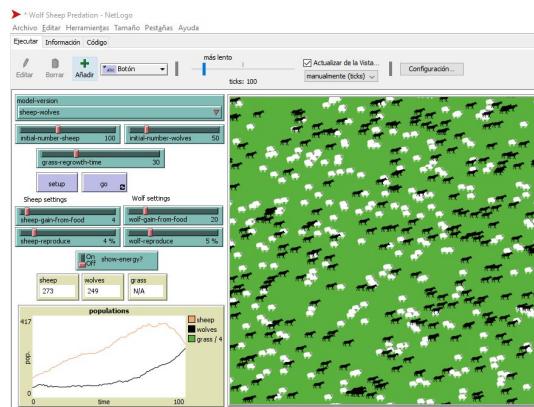
Debido a la escasa población de ovejas, la población de lobos se extinguíó, haciendo que las ovejas se reprodujeran de manera considerable



Ajuste de configuraciones: controles deslizantes e interruptores

-¿Qué pasó con las ovejas con el tiempo?

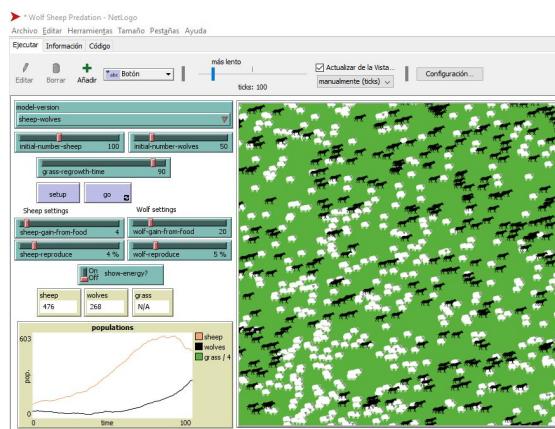
R: La población de ovejas iba aumentando, pero al paso de 100 ticks iba disminuyendo debido al crecimiento de la población de lobos.



-¿Qué provocó el cambio? ¿El resultado fue el mismo que en su ejecución anterior?

R: Al aumentar el tiempo de regeneración de la hierba de 30 a 90, se observa que inicialmente la reproducción de las ovejas se hace más lenta en comparación a la ejecución anterior, y así mismo, afecta la reproducción de los lobos; sin embargo, durante un intervalo del tiempo, las ovejas aumentan rápidamente, haciendo que los lobos también lo hagan, lo cual causa la disminución de ovejas.

Entre ambos resultados, se puede observar que aunque tengan un comportamiento casi igual, en el tick 100 se logran diferenciar, ya que con el grass-regrowth-time= 30, la población de lobos está cerca de superar al de las ovejas, y con el grass-regrowth-time= 90, aún se ve una buena diferencia entre ambas poblaciones.

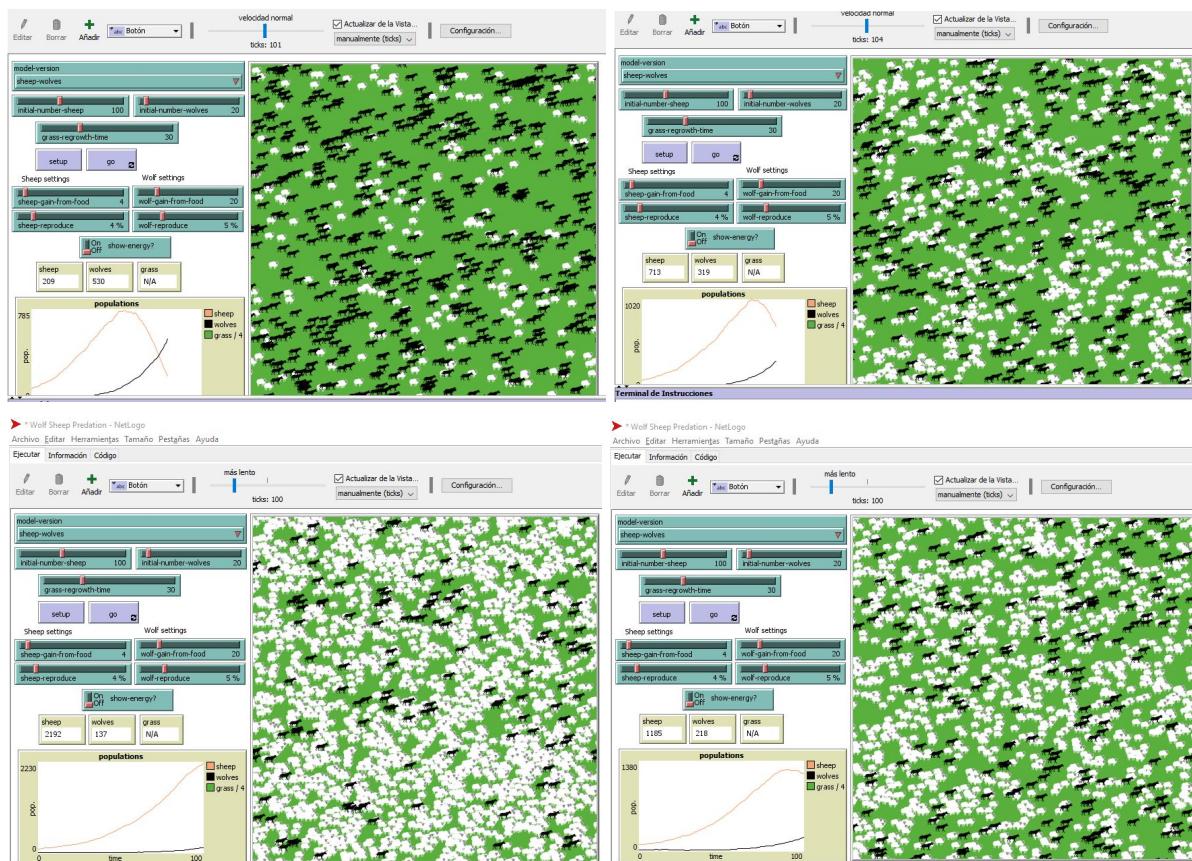


-¿Qué pasaría con la población ovina si hubiera más ovejas y menos lobos inicialmente?

R: Creemos que la población ovina sobreviviría más tiempo en comparación con la población de las ejecuciones anteriores, pero de igual manera se extinguirían debido al constante crecimiento de la población de lobos, quienes también en su momento se van a extinguir por falta de alimento. En pocas palabras, solo se extiende el tiempo para que se extingan.

-¿Qué pasó con la población de ovejas?

R: La población de ovejas tiende a crecer más rápidamente en comparación a la población de lobos, pero cuando llega al pico máximo (población de ovejas), su número desciende y la población de lobos empieza a aumentar considerablemente.



-¿Le sorprendió este resultado? ¿Qué otros controles deslizantes o interruptores se pueden ajustar para ayudar a la población ovina?

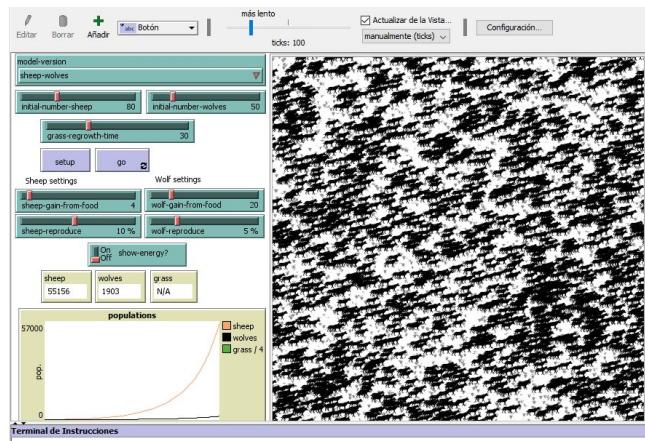
R: No nos sorprendió, ya que teniendo en cuenta las ejecuciones anteriores, esperábamos que el comportamiento fuera similar al observado, puesto que al haber más ovejas, se reproducen más rápido y los lobos tardan un poco más.

Para ayudar a la población ovina también se podrían ajustar los siguientes controles:

- Disminuir el valor de **wolf-gain-from-food** (La cantidad de energía que obtienen los lobos por cada oveja que se comen).
- Aumentar el porcentaje de **sheep-reproduce** (La probabilidad de que una oveja se reproduzca en cada paso de tiempo).
- Disminuir el porcentaje de **wolf-reproduce** (La probabilidad de que un lobo se reproduzca en cada paso de tiempo).

-¿Qué pasó con los lobos en esta corrida del modelo?

R: La población de lobos crece muy despacio, y dejando correr por un rato la simulación, observamos que la población de ovejas nunca va a parar de crecer y por tal razón, los lobos tampoco pararán de reproducirse, pero lo harán muchísimo más despacio que las ovejas.

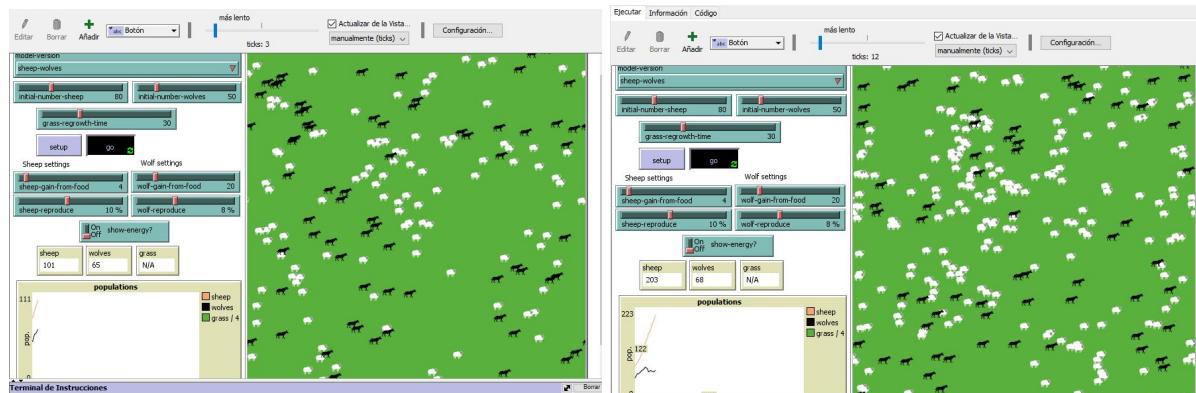


Recopilación de información: gráficos y monitores

Controlando la vista.

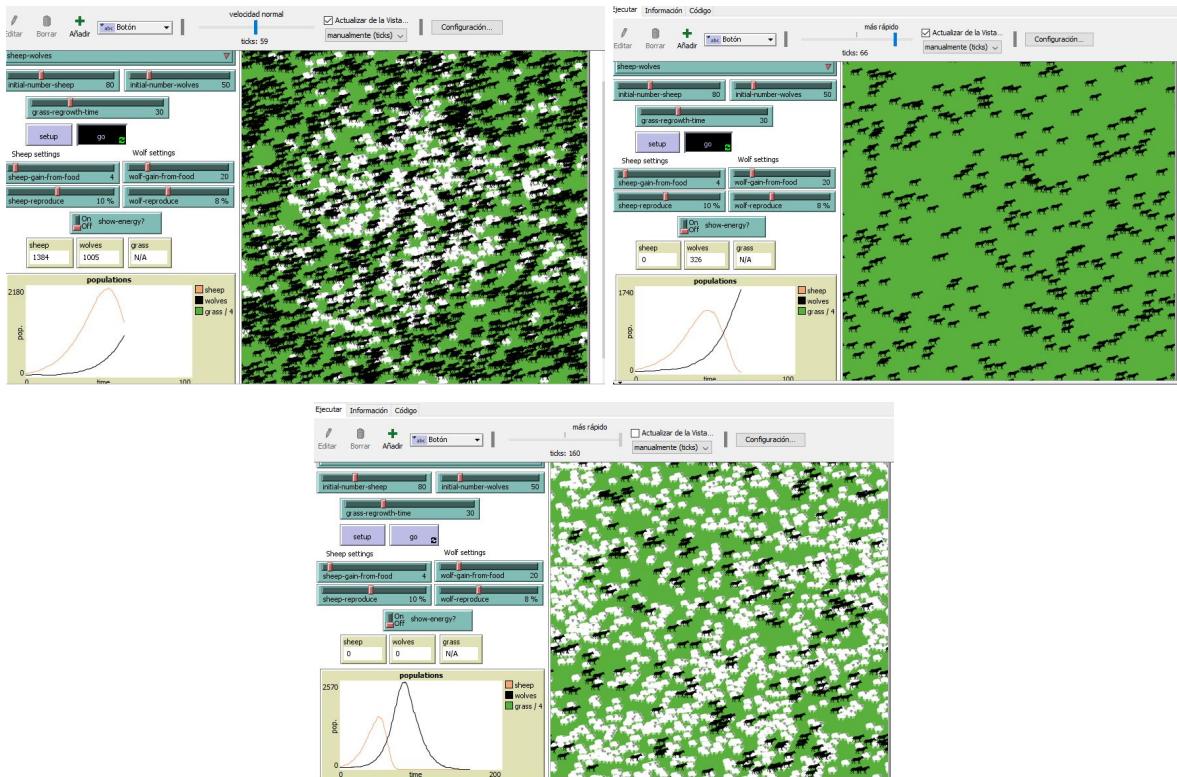
-¿Qué ocurre?

R: La ejecución del modelo disminuye la velocidad, observando también que los números en los monitores avanzan más lentamente.

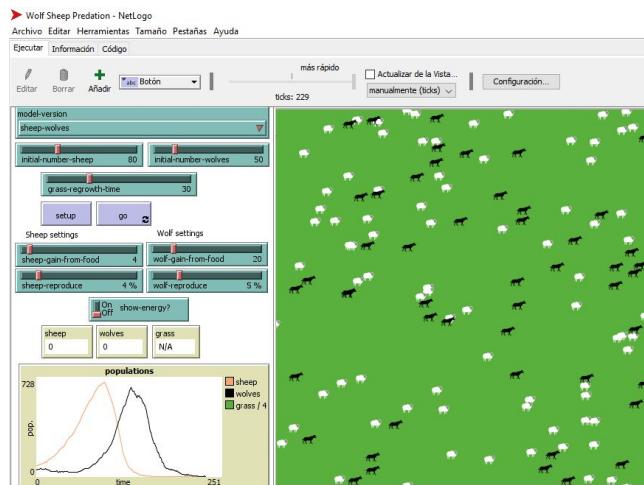


-¿Qué ocurre?

R: Cuando la barra de la velocidad está en el centro entonces la velocidad de la simulación es normal, al desplazarla hacia la derecha la velocidad de ejecución aumenta y al desmarcar la casilla “Actualizar la vista”, la vista se congela, la velocidad sube al máximo, la ejecución termina inmediatamente y muestra la gráfica resultante de todo el proceso.



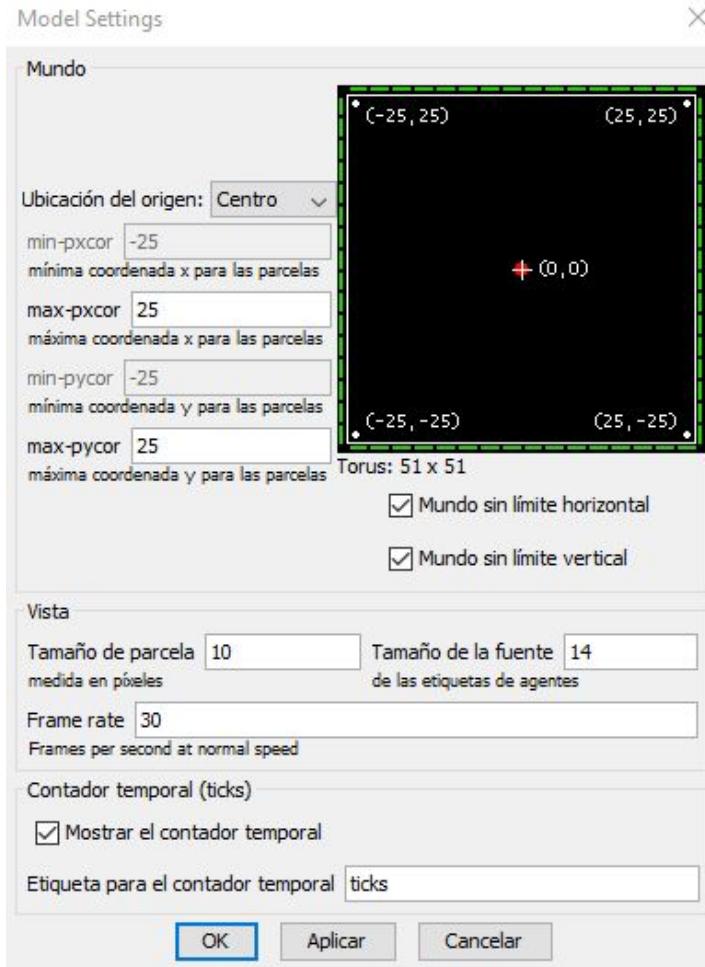
Si ejecutamos el modelo con la casilla desmarcada, únicamente se mostrará la gráfica a la mayor velocidad de ejecución, la vista queda estática, ya que dimos la instrucción de que no se actualizara.



-¿Cuáles son las configuraciones actuales para min-pxcor, max-pxcor, min-pycor, max-pycor y tamaño de parcela?

R: Las configuraciones para cada una de las anteriores características son:

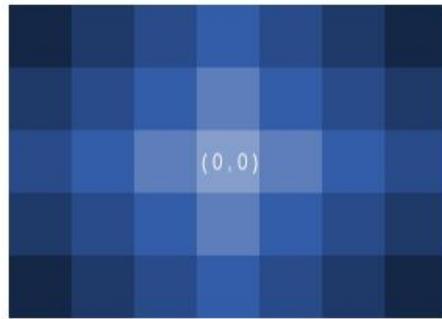
- min-pxcor: -25
- max-pxcor: 25
- min-pycor: -25
- max-pycor: 25
- Tamaño de la Parcela: 10 píxeles



-¿Qué números cambiaron? ¿Qué números no cambiaron?

R: El único número que cambió es el tamaño de la parcela, que anteriormente era de 10 píxeles, ahora es de 8.84314 píxeles. Las demás configuraciones siguieron iguales.





-¿Cuántas baldosas de distancia hay en el mosaico (0,0) del lado derecho de la sala?

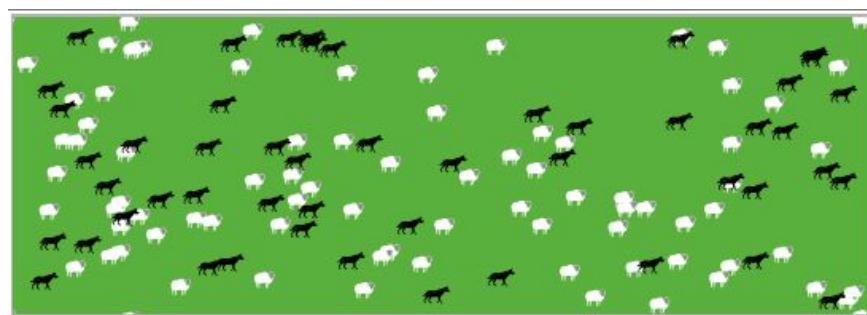
R: Hay 3 baldosas de distancia.

-¿Cuántas baldosas de distancia hay en el mosaico (0,0) del lado izquierdo de la sala?

R: Hay -3 baldosas de distancia.

-¿Qué pasó con la forma de la vista?

R: La forma de la vista pasó de ser cuadrada a ser rectangular debido al cambio de la configuración en max-pxcor y max-pycor, quedando una vista de 30px * 10px.



-¿Qué pasó con el tamaño de la vista? ¿Cambió su forma?

R: El tamaño de la vista se hizo más grande y no cambió su forma.



Model Settings

Mundo

Ubicación del origen: Centro

min-pxcor -30
mínima coordenada x para las parcelas

max-pxcor 30
máxima coordenada x para las parcelas

min-pycor -10
mínima coordenada y para las parcelas

max-pycor 10
máxima coordenada y para las parcelas

Torus: 61 x 21

Mundo sin límite horizontal

Mundo sin límite vertical

Vista

Tamaño de parcela 20 medida en píxeles

Tamaño de la fuente 14 de las etiquetas de agentes

Frame rate 30
Frames per second at normal speed

Contador temporal (ticks)

Mostrar el contador temporal

Etiqueta para el contador temporal ticks

OK Aplicar Cancelar

A detailed view of the 'Model Settings' dialog box. The 'Mundo' tab is active, showing world dimensions from (-30, -10) to (30, 10) with a torus of 61x21. Two checkboxes are checked: 'Mundo sin límite horizontal' (World without horizontal limit) and 'Mundo sin límite vertical' (World without vertical limit). The 'Vista' tab is also shown, featuring settings for parcel size (20 pixels), font size for agent labels (14), frame rate (30 FPS), a temporal counter, and a 'Show temporal counter' checkbox. The 'Tamaño de parcela' input field is highlighted with a red rectangle.

Tutorial # 2. Comandos o Instrucciones

Modelo de Muestra: Traffic Basic.

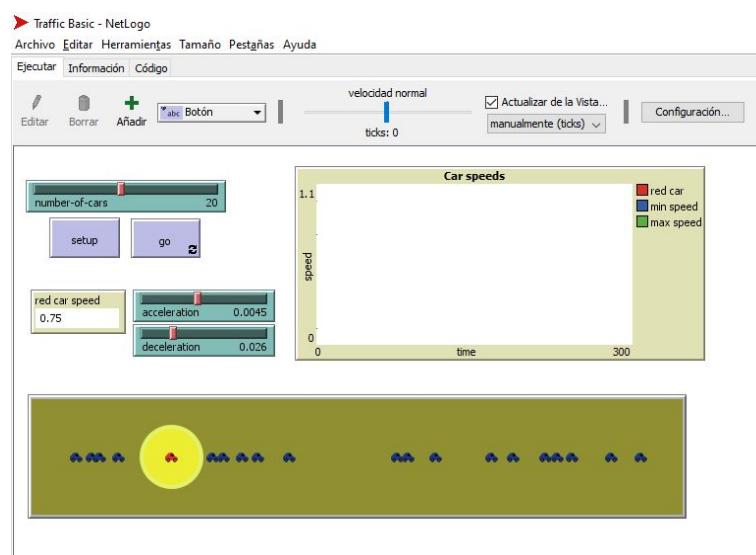
-Dado que está utilizando el modelo Traffic Basic, ¿ha notado alguna adición que le gustaría hacer al modelo?

R: Sí, nos gustaría mejorar los carriles, colocar señalización en el piso, las líneas amarillas, también se podría hacer la simulación un poco más llamativa visualmente, por ejemplo, colocar vehículos de otros colores y que la carretera sea con doble carril, ya que hoy en día son las que más tráfico manejan.

Centro de Comando (Terminal de Instrucciones).

-¿Qué pasó con la vista?

R: Al ejecutar el comando **ask patches [set pcolor yellow]** en la terminal de instrucciones, se observa que todo se vuelve de color amarillo, excepto los autos.



-¿Por qué los autos no se volvieron amarillos también?

R: Los autos no cambian de color debido a que en la instrucción que se ejecutó solo se indica que sean los parches los que cambien de color, y los autos son tortugas.

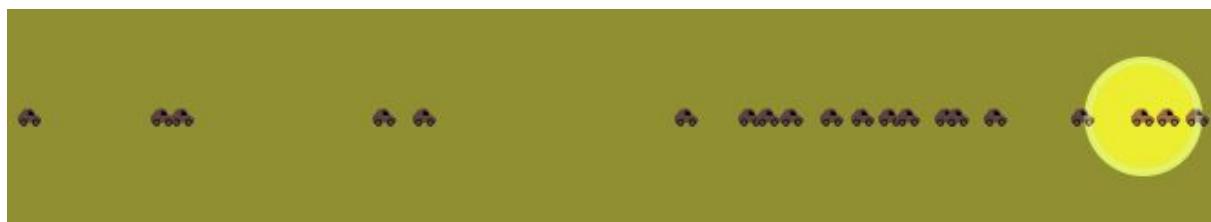
-¿Qué pasó en el Terminal de Instrucciones?

R: Al ejecutar el comando que se escribe en el observador, este aparecerá en la terminal, que muestra las líneas de código o comandos que se ejecutan, en un tipo de historial.

The figure shows a terminal window titled "Terminal de Instrucciones". The text "observador>" is at the bottom, followed by the command "ask patches [set pcolor yellow]".

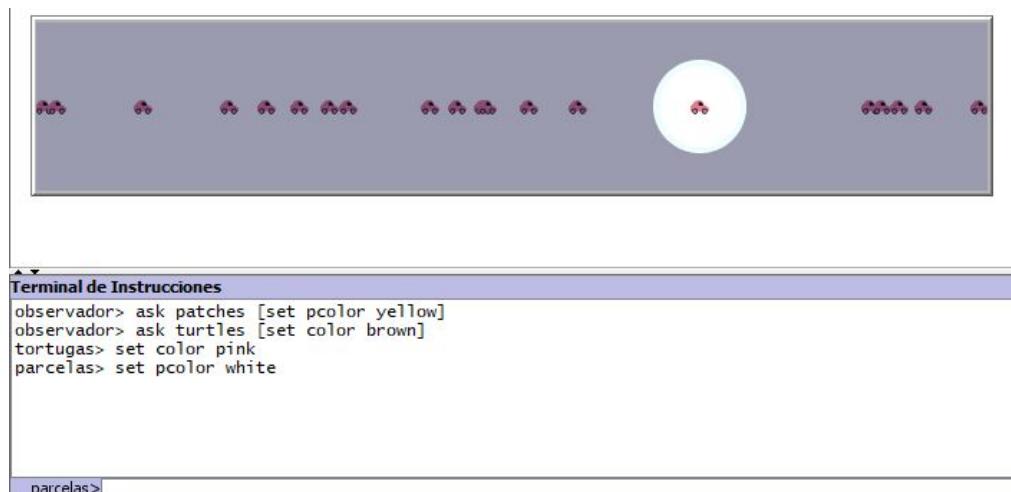
-¿Fue el resultado lo que esperaba?

R: No exactamente, esperábamos que todos los autos a excepción del auto rojo, cambiaran a color café, pero nos dimos cuenta que todos los autos cambian de color, incluso el auto rojo que a simple vista no se identifica, solo por el círculo que sigue enfocándolo.



-¿Cómo se ve la vista ahora?

R: Despues de ejecutar los comandos **set color pink** y **set pcolor white**, los autos cambian a color rosa y el fondo de la vista cambia a color blanco.



-¿Nota alguna diferencia entre estos dos comandos y los comandos del observador de antes?

R: A diferencia de las instrucciones anteriores, los cambios de las nuevas instrucciones se pueden aplicar al agente específico seleccionado en la terminal; con las instrucciones anteriores, al estar en la terminal como observador, era necesario especificar a qué agente se le iba a realizar el cambio.

-¿Qué ocurrió?

R: Los colores de toda la vista volvieron a las asignaciones que tenía por defecto.



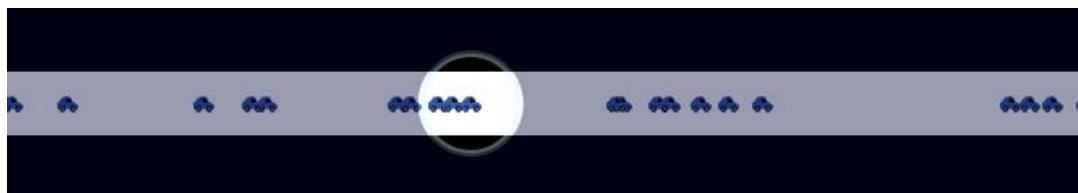
Trabajando con Colores.

-¿Cuál es la diferencia entre color y pcolor?

R: La diferencia es que “color” se usa únicamente para las tortugas y “pcolor” para las parcelas o parches.

-¿Qué pasó con los autos?

R: Los autos que estaban azules siguen iguales, solo el auto rojo se modificó al color azul.



Monitores de agentes y comandantes de agentes.

-¿Cuál es el número de esta tortuga?

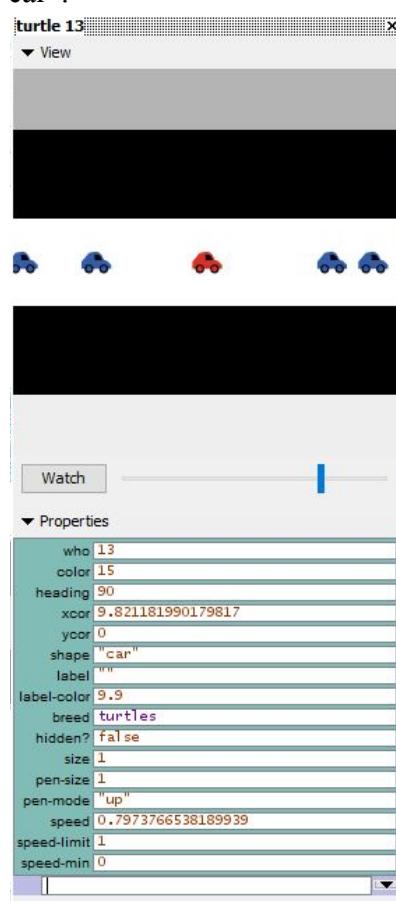
R: El número de la tortuga es 13.

-¿De qué color es esta tortuga?

R: La tortuga es de color 15.

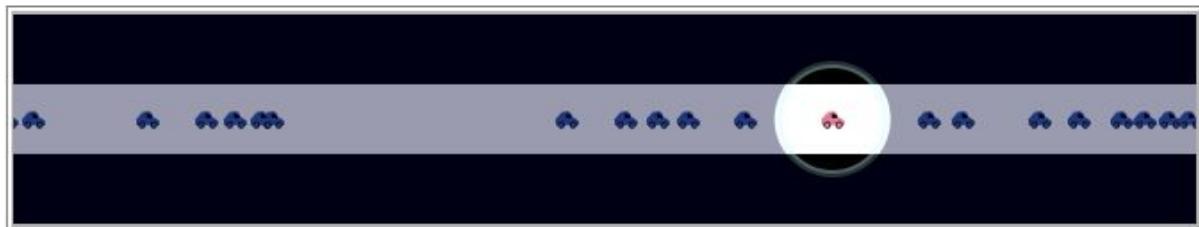
-¿Qué forma tiene esta tortuga?

R: Tiene la forma de un auto, “car”.



-¿Qué sucede en la vista?

R: El auto seleccionado cambia al color rosado.



-¿Cambió algo en el Turtle Monitor?

R: Sí, en el campo “color” se cambió el valor de 15 a 135.

turtle 13

▼ View

Watch

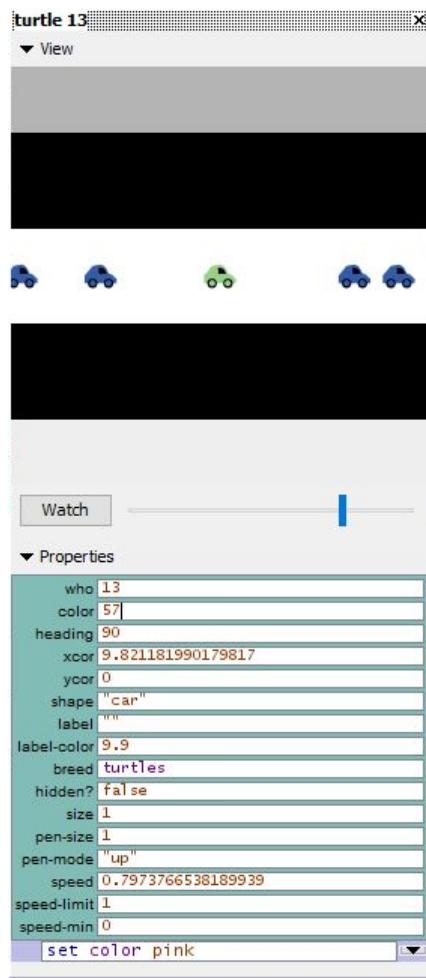
▼ Properties

who	13
color	135
heading	90
xcor	9.821181990179817
ycor	0
shape	"car"
label	" "
label-color	9.9
breed	turtles
hidden?	false
size	1
pen-size	1
pen-mode	"up"
speed	0.7973766538189939
speed-limit	1
speed-min	0

set color pink

-¿Qué pasó?

R: Al escribir “green + 2”, el auto cambió a un color verde claro y el número cambió a 57.



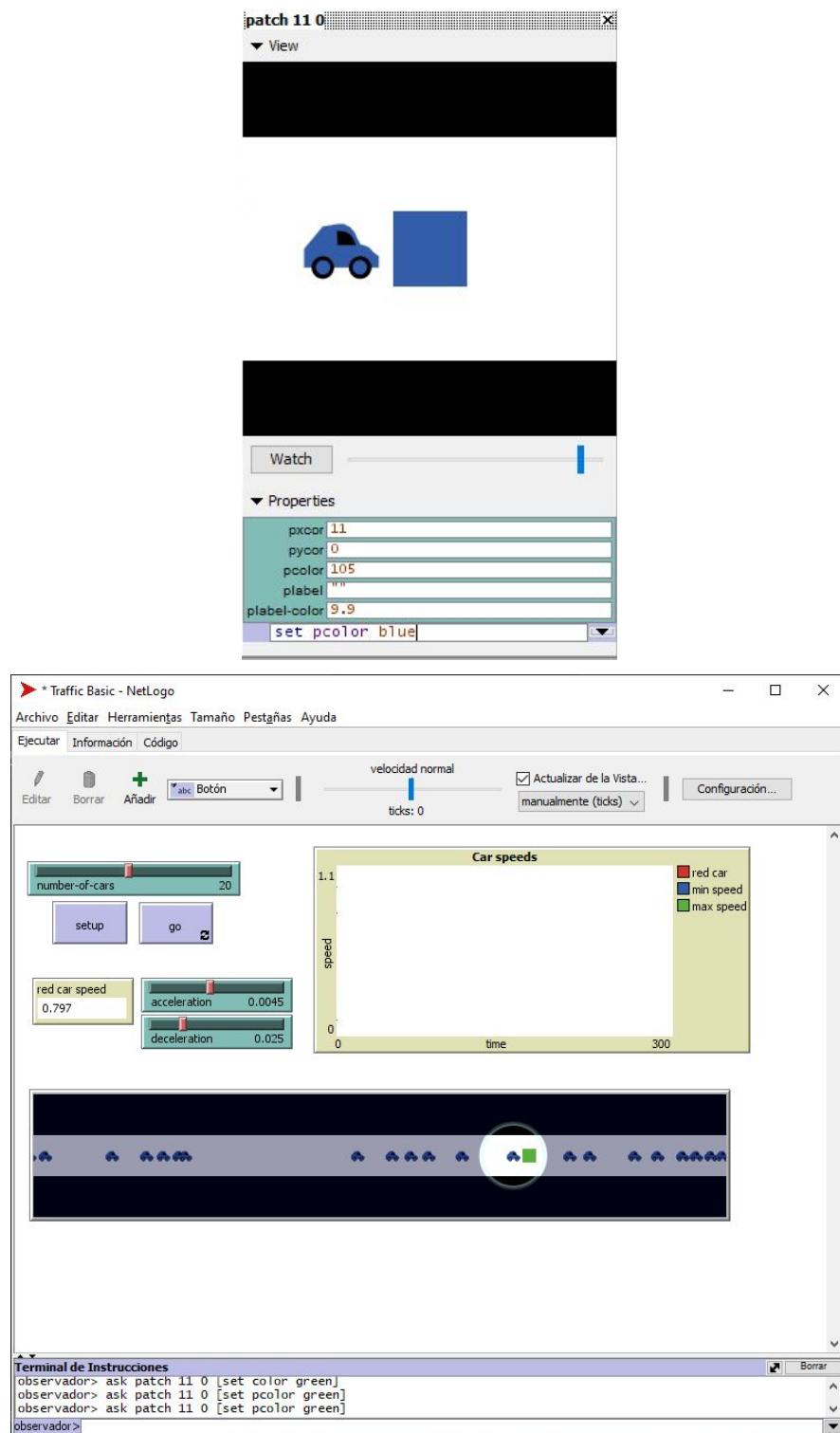
-¿Qué pasa?

R: La tortuga número 13 cambió a color azul.



-¿Puede haber un monitor de parcelas y usarlo para cambiar el color de un solo una parcela?

R: Si es posible, seleccionamos una parcela, de la misma forma como seleccionamos una tortuga, y le damos inspect, el programa va a mostrar el monitor de la parcela seleccionada, al igual que pasó con la tortuga, en el monitor de parcelas, podemos escribir “set pcolor” y el color deseado, la parcela seleccionada cambiara de color; también en la terminal, estando como en el observador, indicamos la parcela que queremos modificar, que a diferencia de una tortuga, la parcela usa coordenadas, por lo tanto para seleccionarla debemos poner las coordenadas en “x” y “y” como se muestra en la segunda imagen.



Tutorial # 3 Procedimientos

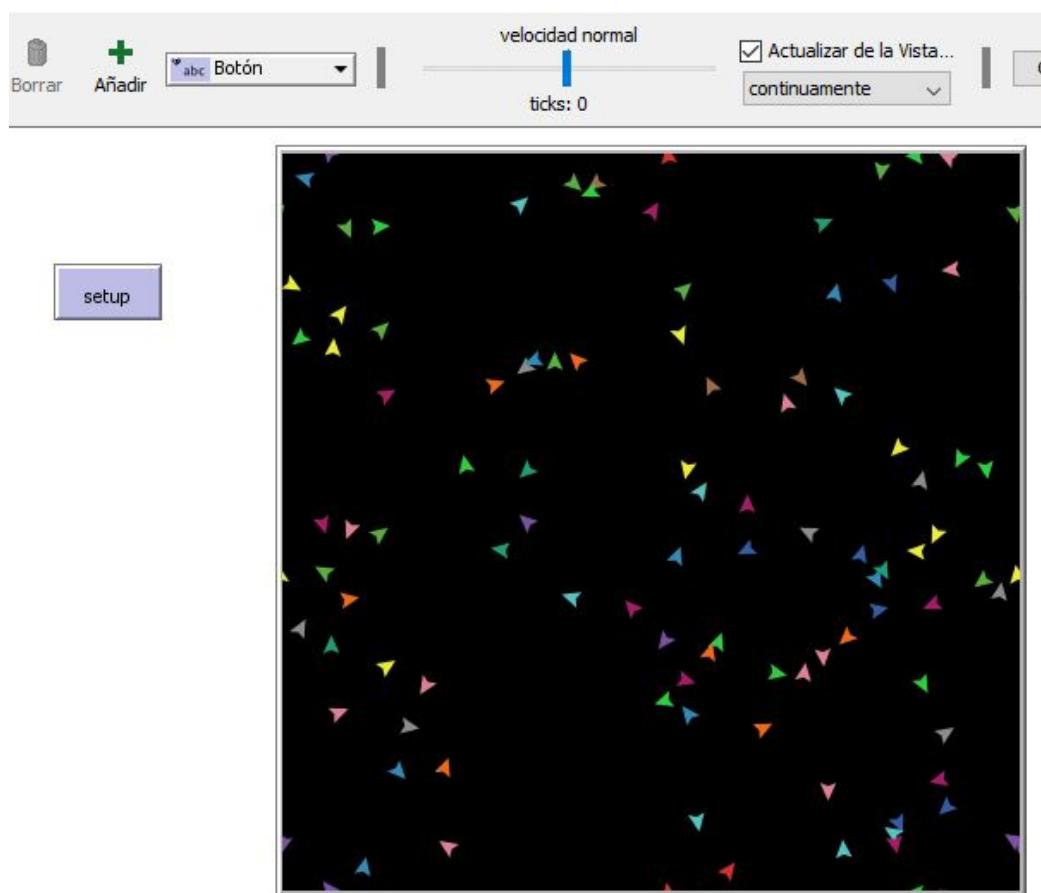
En este tutorial aprendemos a escribir procedimientos que hacen que las tortugas se muevan, coman, se reproduzcan y mueran. También a cómo hacer monitores, controles deslizantes y gráficos.

Construyendo el botón de Configuración (Setup)

Para comenzar, añadimos un botón de configuración al que llamamos “setup” para ejecutar el procedimiento “setup” que se definió de la siguiente manera:

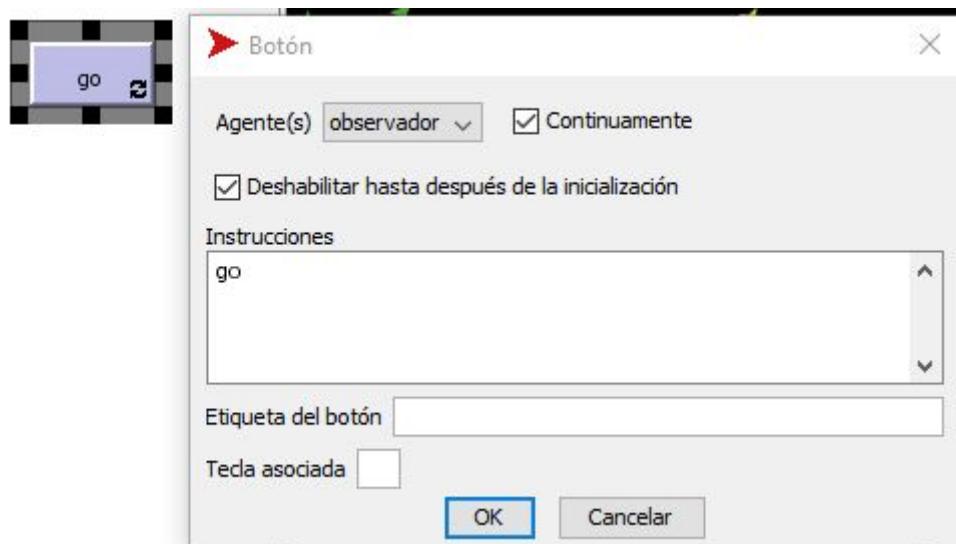
```
to setup
  clear-all
  create-turtles 100 [ setxy random-xcor random-ycor ]
  reset-ticks
end
```

Al presionar el botón “setup”, se ejecutó el procedimiento y vimos a las tortugas dispersas por toda la vista, con sus ubicaciones aleatorias.



Construyendo el botón go

El paso siguiente fue añadir otro botón al que llamamos “go”, configurado como se muestra a continuación:



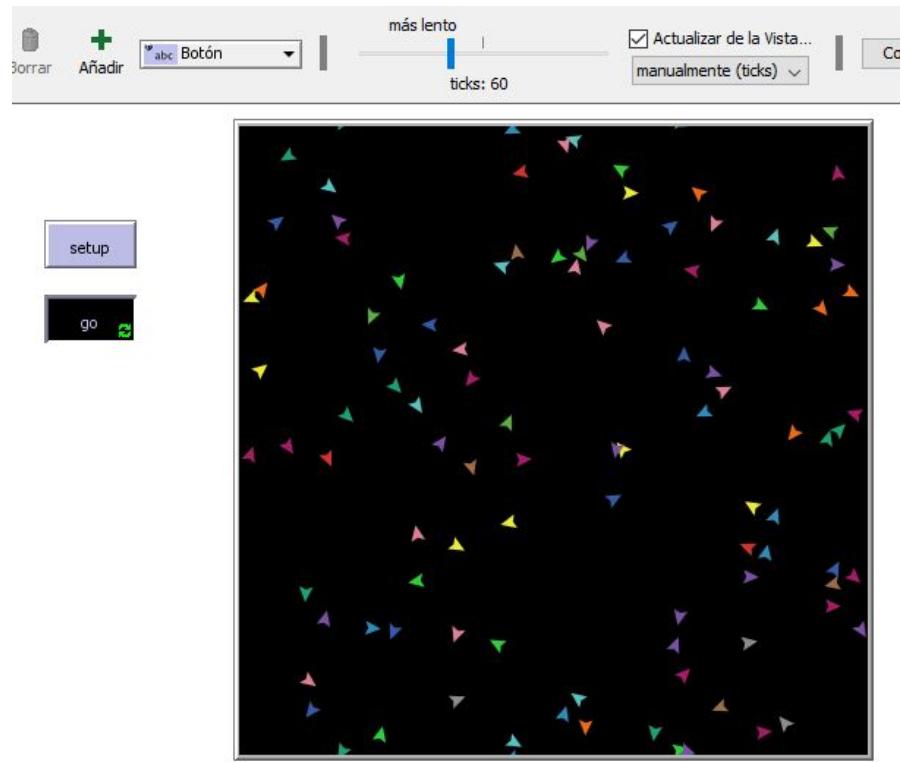
Luego agregamos el procedimiento “go” al código, el cual contiene un procedimiento llamado “move-turtles” que agregamos después de “go”.

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

to go
  move-turtles
  tick
end

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

A continuación, pulsamos el botón “setup” para limpiar el “mundo” y reiniciar la simulación, luego se pulsó el botón go para dar inicio a la simulación. Las tortugas comenzaron a moverse de manera aleatoria en todo el tablero.



Experimentando con Instrucciones

En la terminal de instrucciones escribimos el comando “set color red” para cambiar el color de todas las tortugas a rojo, y también el comando “pen-down” para que se dibujara el rastro que cada tortuga iba dejando. Modificamos el procedimiento “move-turtle” cambiando la línea “right random 360” a “right random 45”.

The screenshot shows the Logo programming interface with the same top controls and script area ('setup', 'go'). The main canvas now displays a dense, chaotic pattern of red lines on a black background, created by many turtles moving in various directions. Below the canvas is a terminal window titled 'Terminal de Instrucciones' (Instruction Terminal) containing the following commands:

```
tortugas> set color red
tortugas> pen-down
```

The cursor is currently at 'tortugas>'.

Parcelas y variables

En este paso le dimos color a los patches, para lo cual se modificó el procedimiento setup, al que se le adiciona un procedimiento llamado “setup-patches” para así darle el color a cada uno de los patches, aparte se agregó otro procedimiento llamado “setup-turtles” en donde se indicó que se crearán 100 tortugas y que aparecerían de manera aleatoria en toda la vista.

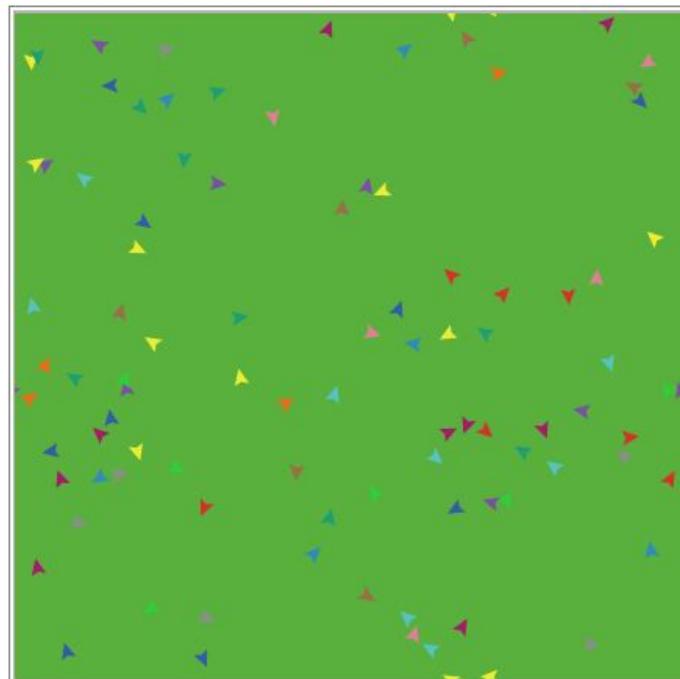
```
to setup
  clear-all
  setup-patches
  setup-turtles
  reset-ticks
end

to go
  move-turtles
  tick
end

to move-turtles
  ask turtles [
    right random 45
    forward 1
  ]
end

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

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



Variables de Tortuga

En esta sección le indicamos a las tortugas que coman pasto, se reproduzcan y posteriormente mueran, además de estas acciones se colocó una nueva variable que mostrará la energía, para esto se agregó una declaración propia de tortugas llamado “turtles-own” el cual permitirá que se muestre la cantidad de energía de manera numérica, esta disminuye o aumenta según el movimiento y la cantidad de pasto que consumen.

Para hacer que las tortugas coman pasto se creó un nuevo procedimiento llamado “eat-grass”, en este se da la instrucción de que los patches que se quedan sin pasto se vuelvan de color negro y que por cada patche que las tortugas coman se les aumentará 10 unidades de energía. Para hacer que las tortugas pierdan energía, se le indica en el procedimiento “move-turtle” que por cada movimiento pierda 1 unidad.

```
turtles-own [energy]

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

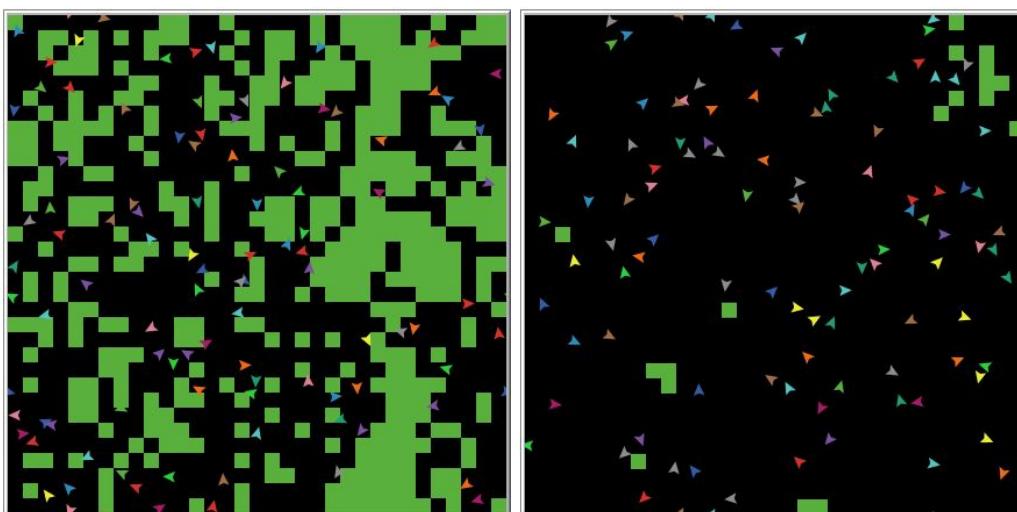
to go
  move-turtles
  eat-grass
  tick
end

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

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

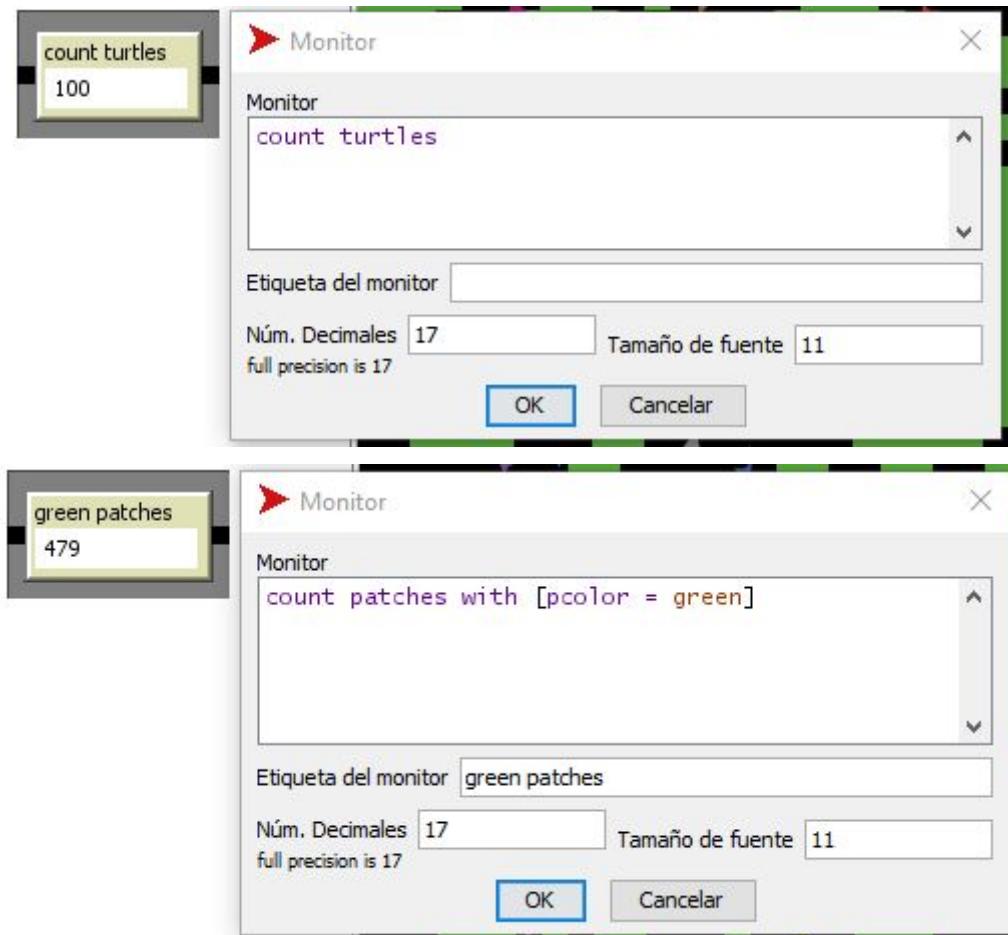
to setup-turtles
  create-turtles 100
  ask turtles [ setxy random-xcor random-ycor ]
end

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

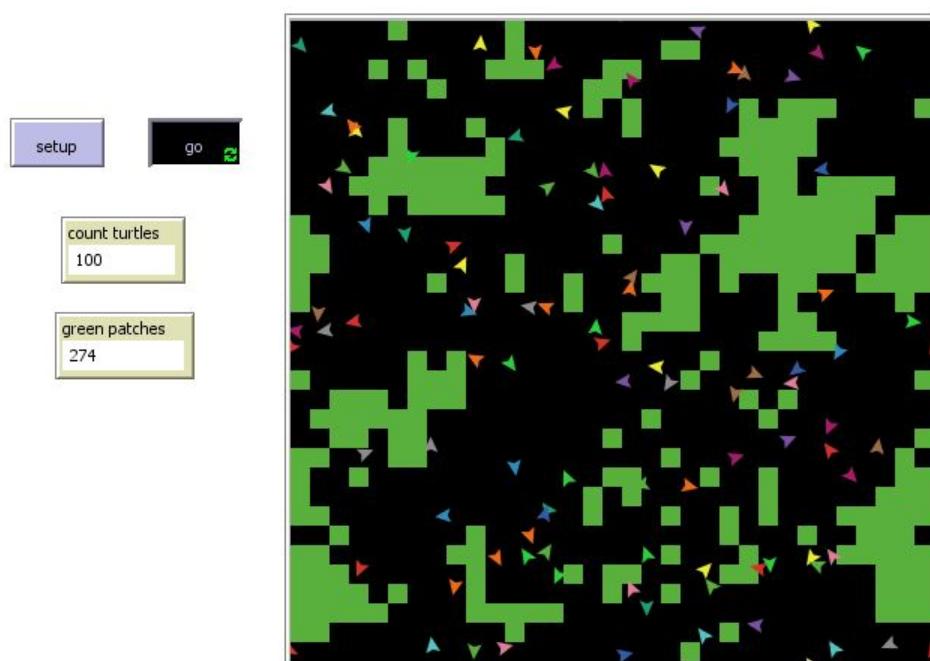


Monitores

En esta sección se agregaron a la interfaz dos monitores con los cuales se puede saber la cantidad de tortugas asignadas en el código y la cantidad de patches de color verde en cada instante de tiempo. Para ello a cada monitor se le dio la instrucción correspondiente.

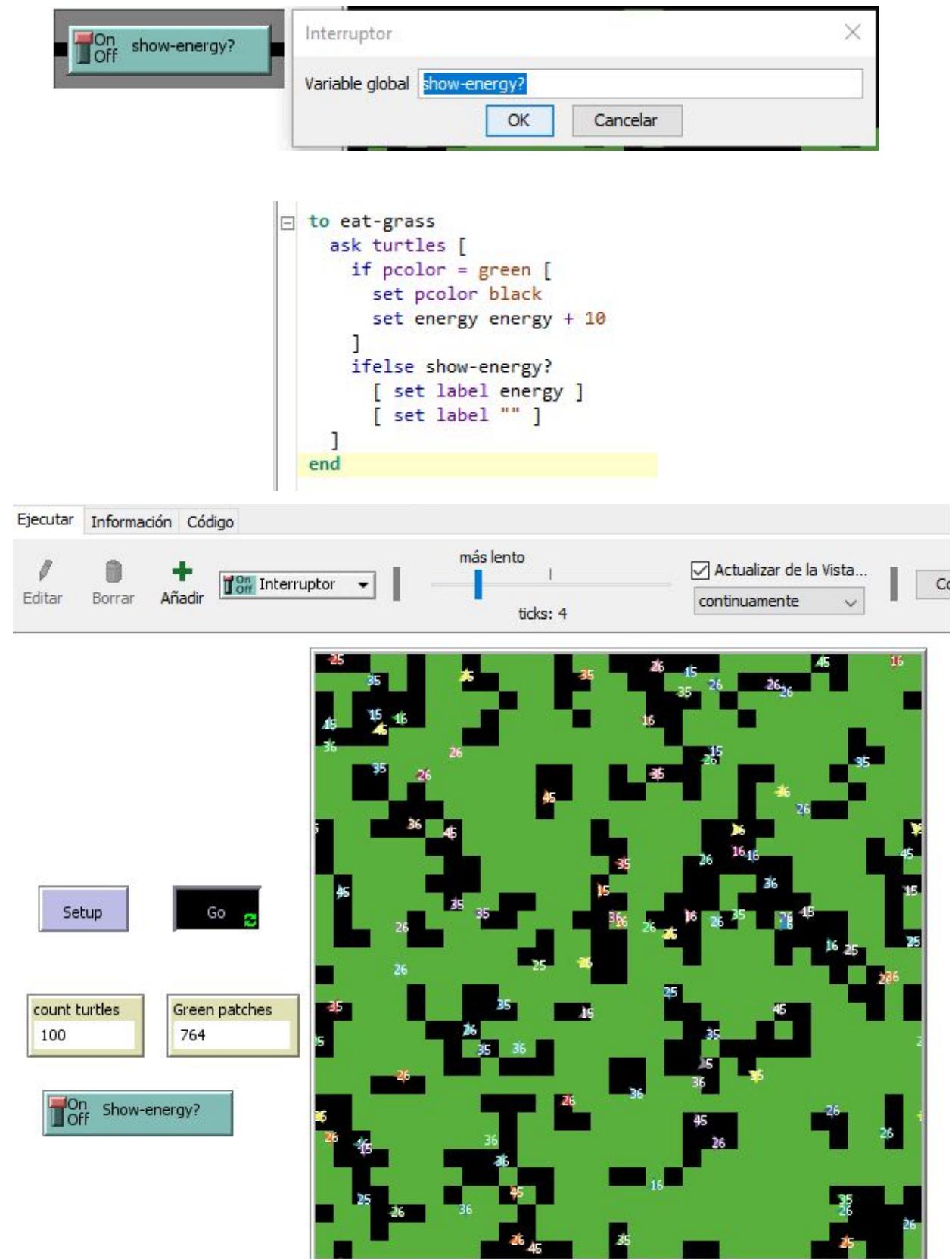


Luego de configurarlos y ejecutar la simulación, se observa de la siguiente manera:



Interruptores y etiquetas

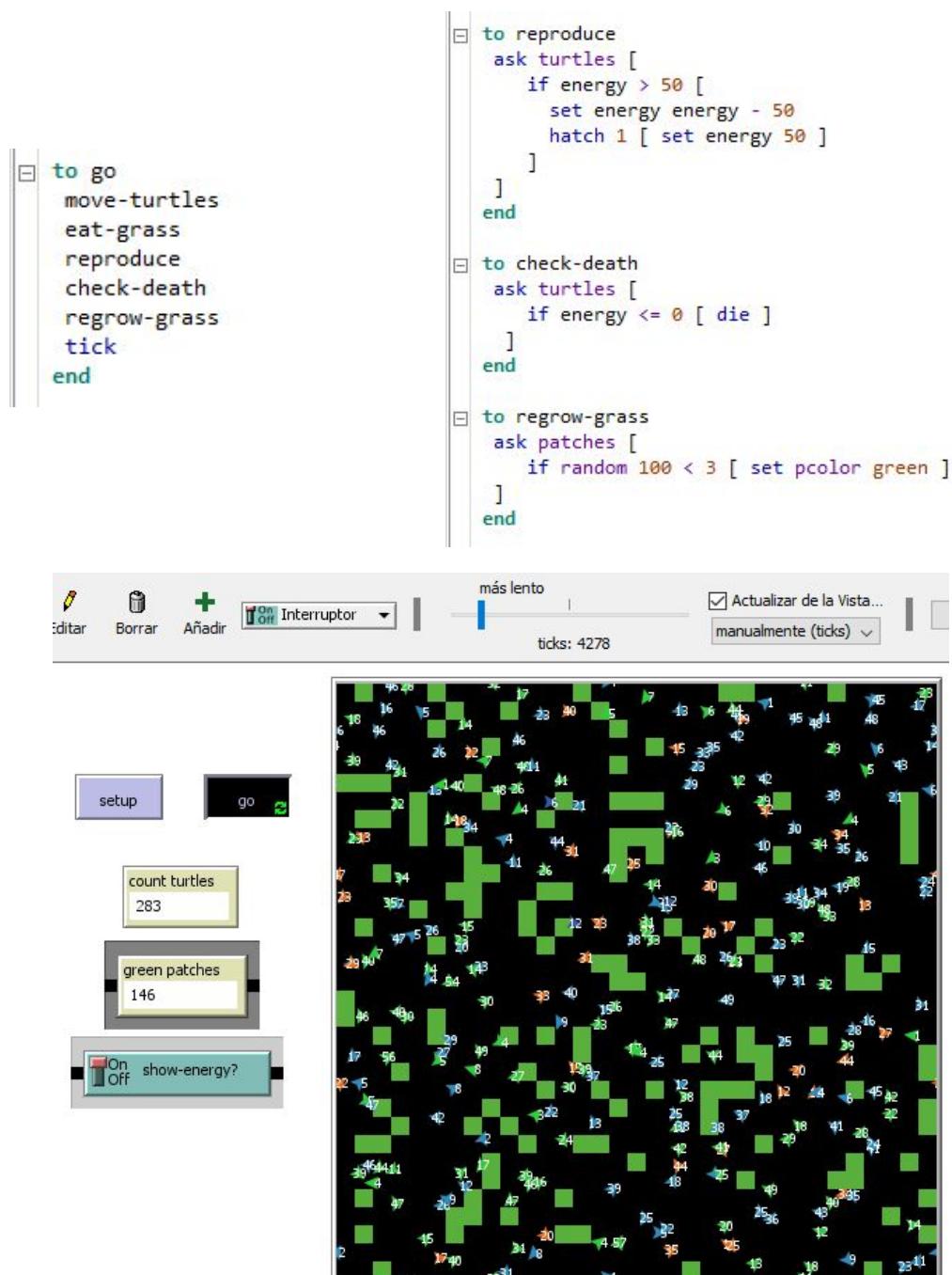
En este paso añadimos un interruptor al cual se le asignó una variable llamada “show-energy” y se modificó el procedimiento eat-grass para mostrar la cantidad de energía de cada tortuga al alimentarse y desplazarse sobre la vista.



Más Procedimientos

Como se mencionó anteriormente, queríamos que las tortugas se reprodujeran, murieran además que el pasto creciera nuevamente, para esto tuvimos que agregar los procedimientos “reproduce”, “check-death” y “regrow-grass”, también modificamos el procedimiento go para que al ejecutarse realice dichos procedimientos.

En “reproduce” se indicó que para la reproducción de las tortugas se hará cada vez que una de ellas tenga la energía mayor a 50, entonces la tortuga se dividirá en dos quedando la nueva tortuga con energía de 50. Para hacer que muera una tortuga, se indicó que cuando la energía de esta sea menor o igual a cero, esta morirá. Por último, para que el pasto crezca se indicó que cuando un número aleatorio entre 0 y 99 sea menor que 3, el patch correspondiente a ese valor tomará el color verde.

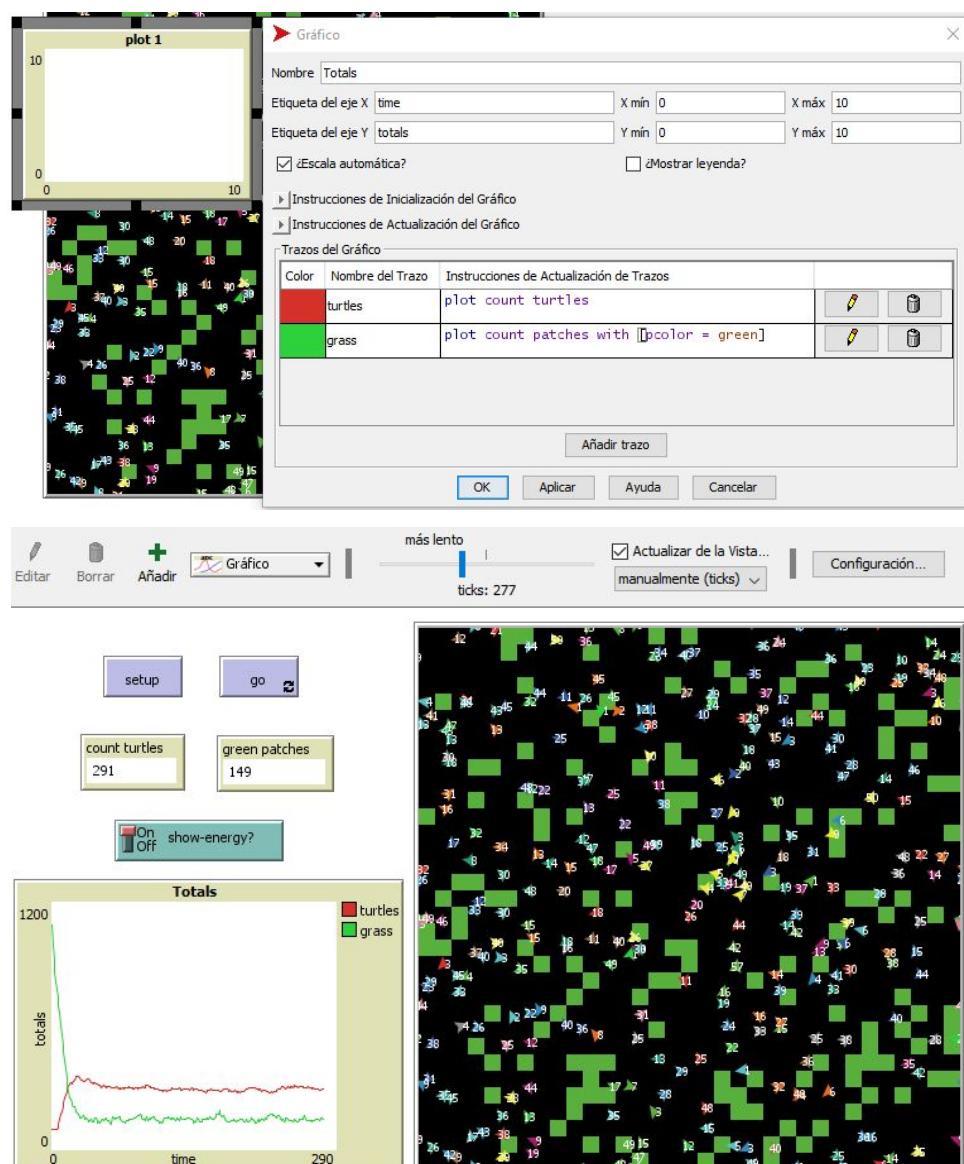


-Si continúa mirando sus monitores en su modelo, verá que count-turtles y green-patches fluctúan. ¿Este patrón de fluctuación es predecible? ¿Hay una relación entre las variables?

R: El patrón no es predecible, los números aumentan y disminuyen en intervalos que no se aprecian a simple vista. Podemos observar que las dos variables tienden a tener un comportamiento inversamente proporcional, es decir, cuando aumentan las tortugas, la hierba disminuye, y cuando disminuyen las tortugas, la hierba aumenta; esto sucede en pequeños valores, por lo tanto debe observarse con una velocidad muy baja.

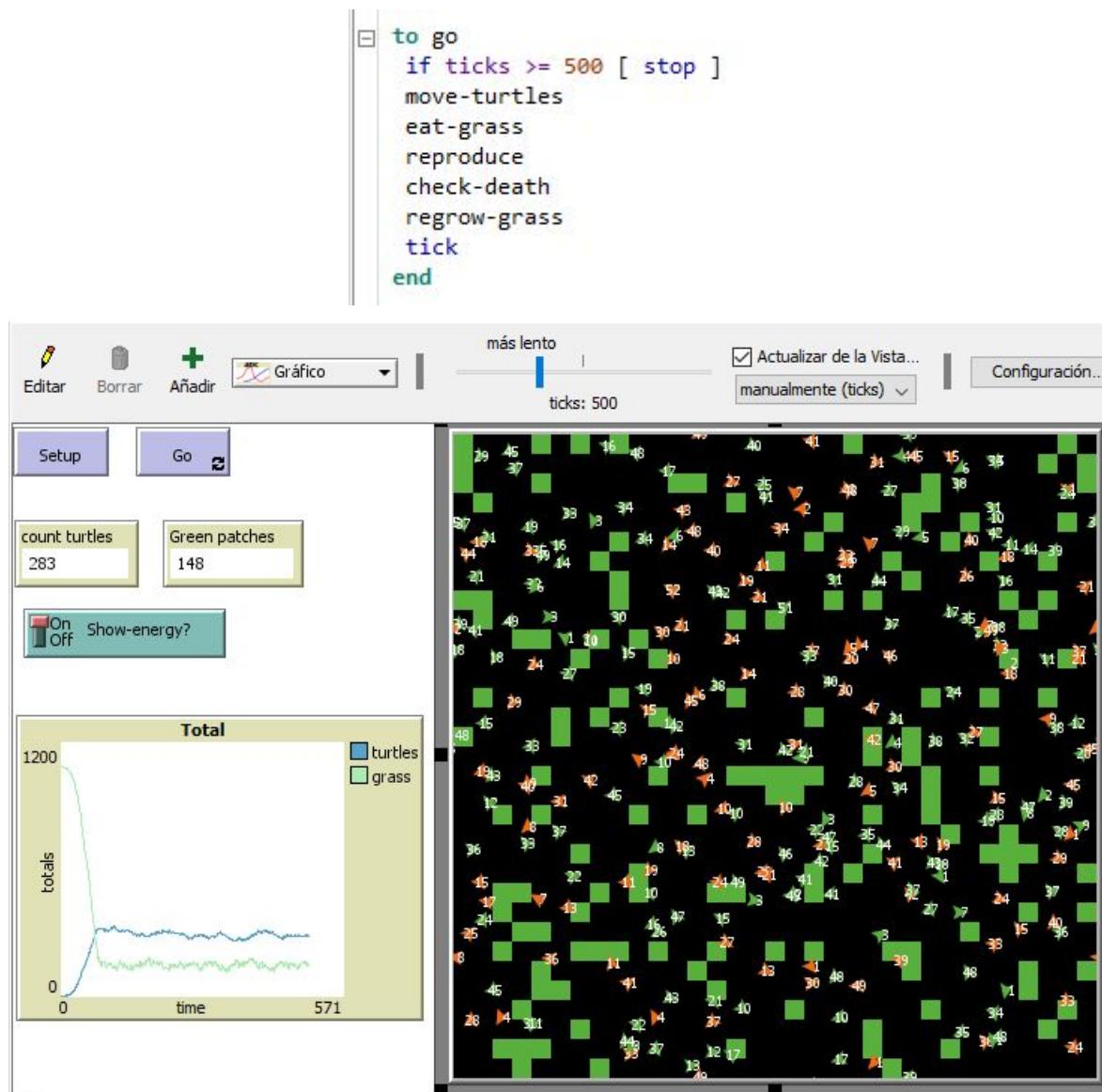
Graficando (Plotting)

En este paso se muestra en una gráfica la cantidad de tortugas vs la cantidad de parcelas en el tiempo, para ello se agregó un gráfico y se le colocó como nombre “totals”, además de que al primer trazo se le asignó el nombre de “turtles” en donde se le colocó la instrucción de contar las tortugas “plot count turtles” y se agregó un nuevo trazo llamado “grass” en donde la instrucción para contar los patches verdes es “plot count patches [pcolor = green]”. Esta gráfica se va dibujando a medida que el modelo se ejecuta.



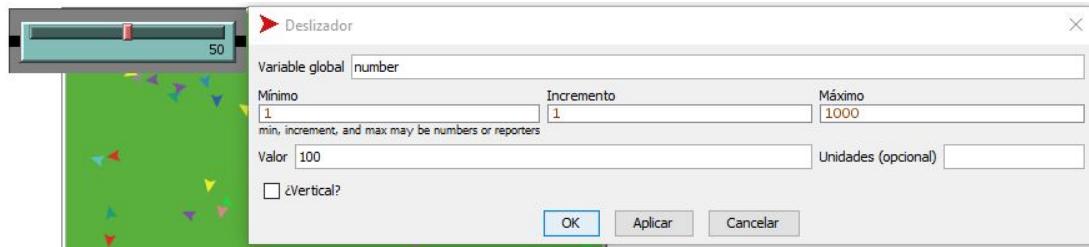
Cuenta Pasos

Este paso nos permite realizar comparaciones del modelo a través de gráficos de diferentes ejecuciones en un mismo lapso de tiempo, para esto se modificó el procedimiento “go” dándole un límite de ticks, el cual al llegar la ejecución a ese límite se detendrá, en este caso se le dio una cantidad de 500 ticks.



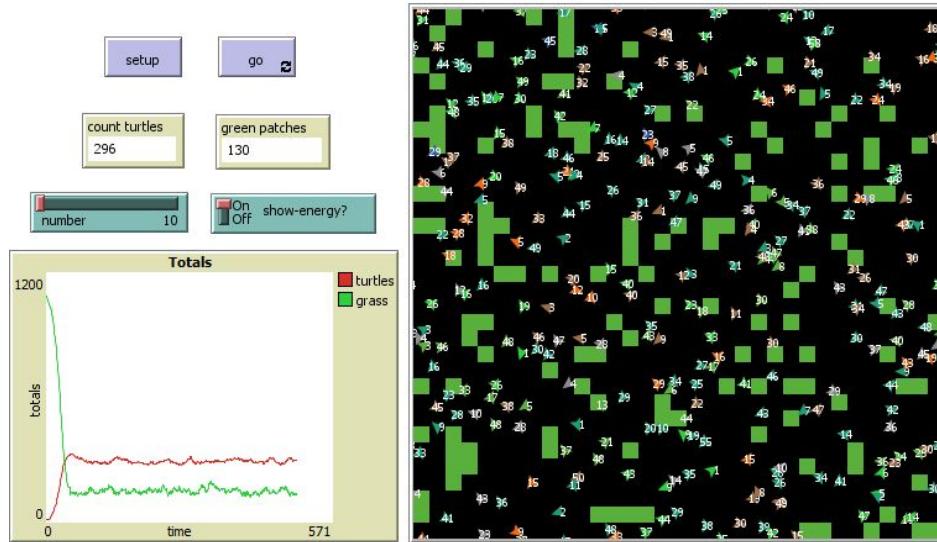
Algunos detalles más

Para ver el comportamiento del modelo en el tiempo, cuando hay más o menos tortugas, se agregó un deslizador que permite desde la interfaz, ajustar la cantidad de tortugas que queremos tener en la ejecución sin tener que ir al código y actualizarlo, pero para lograr que el deslizador funcione se tuvo que modificar el procedimiento setup-turtles, en donde ya no se crearán 100 tortugas si no que ahora será según se establezca en el deslizador.

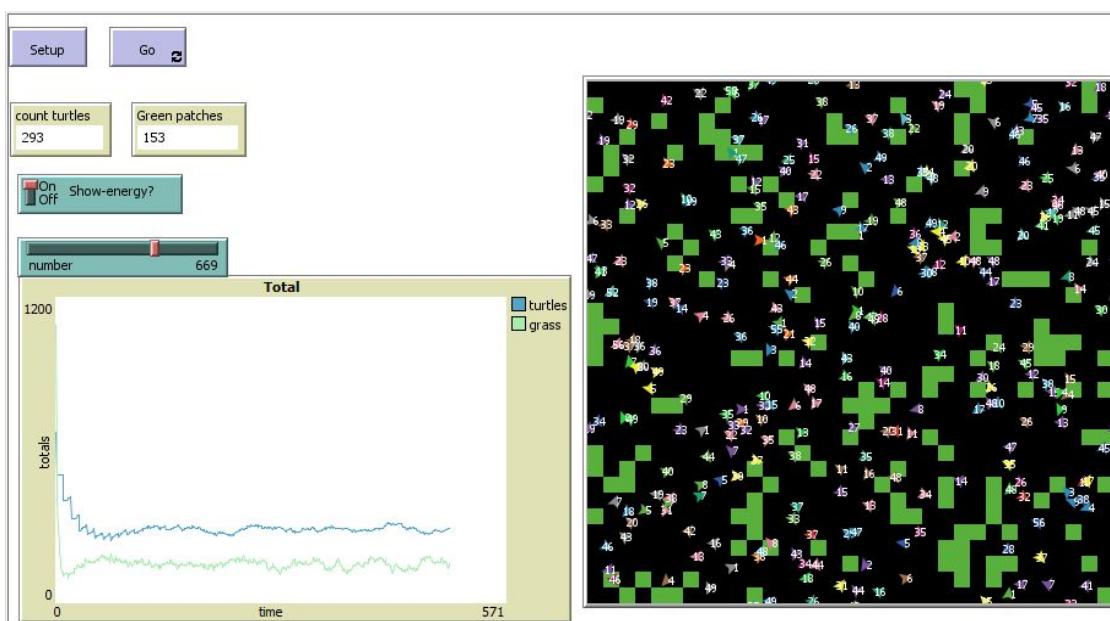


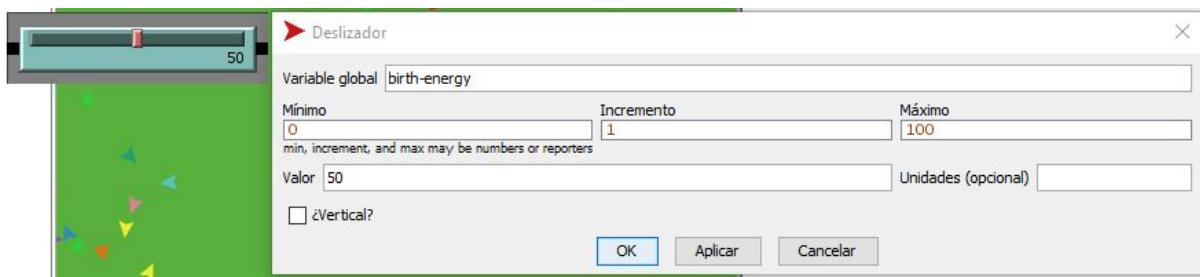
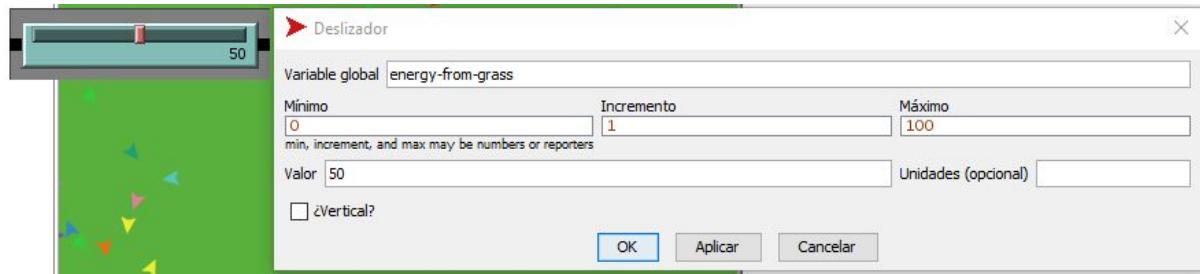
```

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



Así como se pudo ajustar la cantidad de tortugas deseadas mediante un deslizador, también se colocó un deslizador llamado “energy-from-grass” y “ birth-energy” para establecer la cantidad de energía que se quiere que ganen o pierdan las tortugas cuando se alimentan y se reproducen. Para que estos deslizadores funcionaran, se tuvo que actualizar el procedimiento eat-grass para que la energía se pudiese establecer a través del deslizador al igual que en el procedimiento “reproduce”.

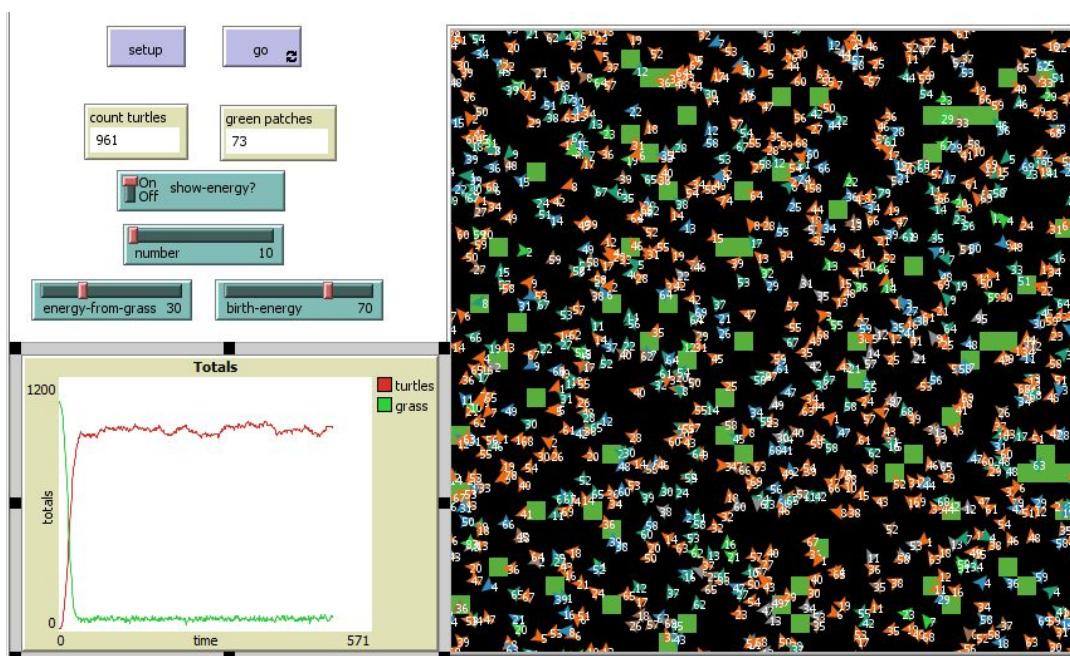


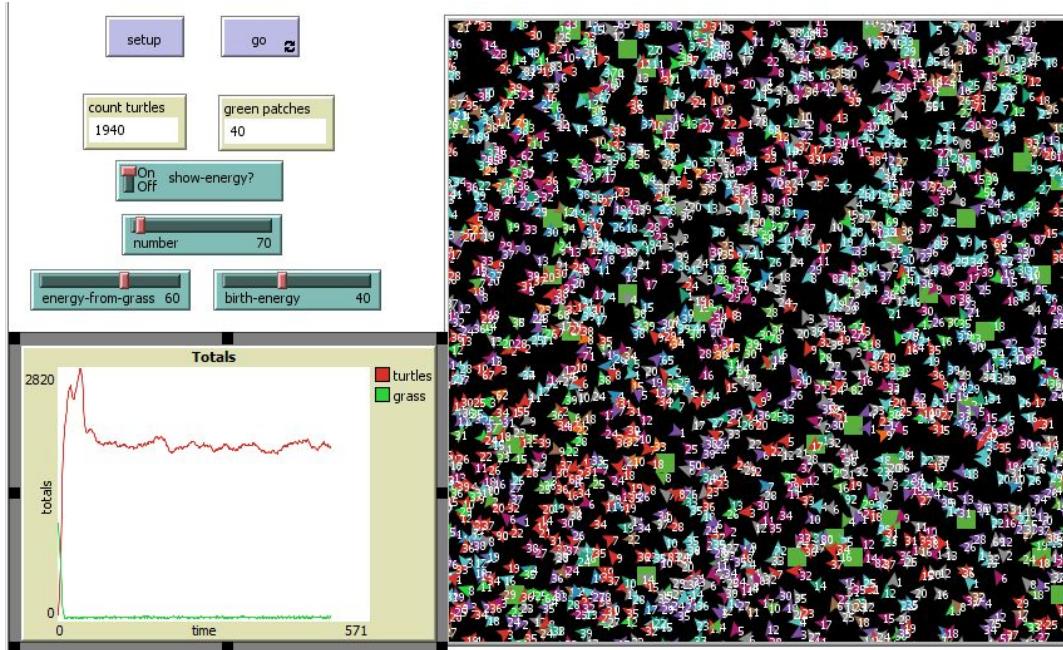


```

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

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





-¿Qué otro deslizador podría agregar para variar la frecuencia con la que la hierba vuelve a crecer?

R: Podríamos agregar un deslizador que permita que la hierba se regenere de acuerdo al tiempo que se indique a través de dicho deslizador.

-¿Hay reglas que pueda agregar al movimiento de las tortugas o a las tortugas recién nacidas que ocurren solo en ciertos momentos?

R: Podría agregarse una regla que indique que las tortugas recién nacidas no se reproduzcan inmediatamente, ya que estas nacen de una vez con el valor de energía límite para reproducirse, de esta manera se controlaría en cierto modo el nivel de reproducción de la población.