Construyendo Simulaciones Sociales Multi-agente: Tutorial de auto-construcción, paso a paso, con NetLogo 6

Primera Parte (1/2)

Francisco J. MIGUEL QUESADA [1]
LSDS-UAB - Dept. Sociologia UAB
(v2.2, Septiembre 2017)

Sobre este tutorial

A lo largo del presente tutorial se guia paso a paso en la construyendo de un pequeño modelo de simulación social (un prototipo de demostración), escrito y ejecutado en un entorno de programación simple llamado NetLogo. El objetivo del tutorial es mostrar, de una manera práctica e inmediata, los elementos básicos de la aproximación a la simulación de sistemas sociales denominada “modelado basado en multiples agentes” (MABM), así como servir de iniciación a la programación usando Netlogo. Obviamente, la simulación se refiere a una sociedad extremadamente estilizada o simplificada, sin embargo resulta útil para introducir los conceptos y elementos generales de cualquier simulación social.

Una vez descargado e instalado el entorno NetLogo se puede seguir el presente tutorial tecleando, o recortando y pegando, las líneas de código informático en el lugar adecuado (la pantalla “Código” del entorno NetLogo), y se puede ejecutar la simulación (desde la pantalla “Ejecutar”) para ver los efectos del código. No se requiere que quien siga el presente tutorial comprenda totalmente el código escrito, ni que aprenda a programar

[1] *Agradecimientos a las revisiones previas: Francisco Linares, José Antonio López, y Oriol Griera.*

sociedades virtuales mediante el mismo. Para profundizar en el aprendizaje de construcción y uso de simulaciones sociales se incluye en la parte final un listado de recursos adicionales.

> ***/!\ Importante:*** *Mientras se van añadiendo líneas de código, siguiendo paso a paso el presente tutorial, pueden aparecer diversos mensajes de error debido a que el interpretador del lenguaje encuentra incoherencias o partes del modelo aún sin definir. Igualmente, hasta que el modelo no está lo suficientemente desarrollado no es posible ejecutarlo.*

> ***Nota:*** *Las imágenes de este tutorial corresponden a la versión 5.3.1. Pueden existir pequeñas diferencias en el interfaz gráfico de versiones posteriores.*

Descargar, instalar y ejecutar Netlogo

Cómo descargar NetLogo

- Accede a la página de descargas oficiales en <http://ccl.northwestern.edu/netlogo/download.shtml> (septiembre 2017)
- En el menú desplegable de **Version:** selecciona la versión 6.0.2, o posterior.[2]
- (Opcional) Si deseas recibir notificaciones introduce tus datos personales.
- Más abajo, haz clic sobre el botón gris [Download].
- En la página de descargas, lee las indicaciones correspondientes al sistema operativo.
- Según el sistema operativo utilizado, haz clic sobre el correspondiente vínculo Download para iniciar la descarga del archivo de instalación.

Cómo instalar NetLogo

- **En M$-Windows:** Una vez descargado el paquete instalable (*.exe, o *.msi) en el propio ordenador local, haz doble clic sobre el mismo para ejecutarlo.

[2] *Personas usuarias de Windows XP: la última versión de Netlogo que funciona en tal sistema es la 5.2.1.*

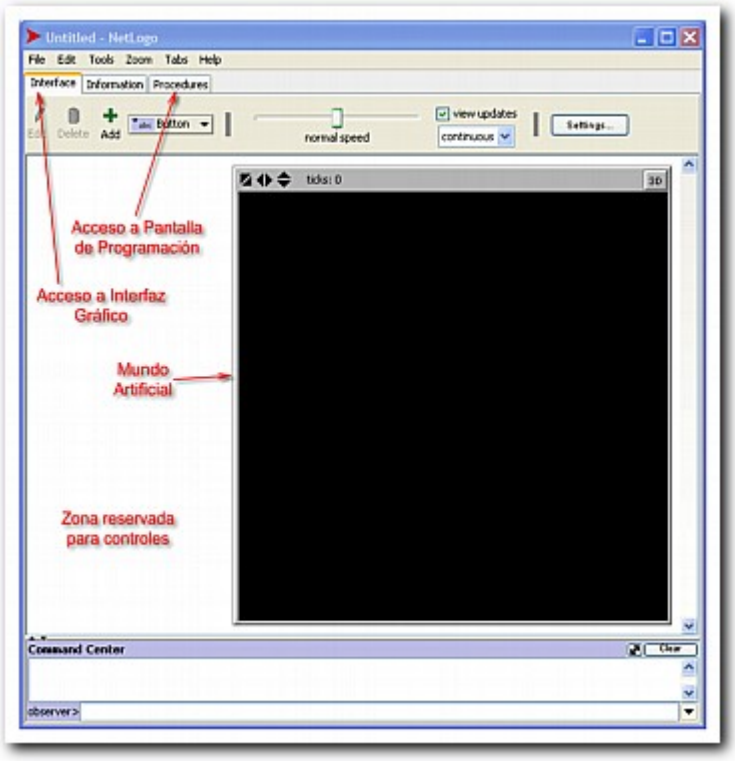
- Sigue las instrucciones de instalación, aceptando todas las opciones por defecto.

  > ***/!\ Atención:*** *NetLogo funciona sobre una plataforma JAVA, que funciona a su vez sobre el sistema operativo de cada ordenador. Esto implica que durante el proceso de instalación 1) se instalará JAVA si es preciso, y 2) a continuación se instalará NetLogo.*

- La instalación de NetLogo incluye la copia en tu ordenador local del manual completo de uso actualizado (en formato HTML) y de numerosos ejemplos excelentemente documentados (en formato *.nlogo).
- **Otros sistemas operativos:** Descomprime el archivo descargado en un directorio ejecutable (/home/...) . En tal directorio encontraras el ejecutable (e.g., *.sh).

Cómo ejecutar NetLogo

- **En M$-Windows**, clic en [Inicia] / [Todos los programas] / NetLogo *(versión)* o doble-clic sobre el icono que se haya creado en el escritorio.
- Se abrirá la pantalla del interfaz gráfico de usuario, esto es, el espacio donde situar los controles de la simulación y visualizar el resultado. El mundo bidimensional artificial se visualiza en forma de ventana cuadrada con fondo negro.



/!\ El idioma del interfaz gráfico será el idioma por defecto del sistema operativo (Inglés, Español, Chino,o Ruso). Para cambiarlo: Menú Principal / Tools / Preferences abrirá una ventana de diálogo con un menú para escoger idioma, a continuación cerrar y volver a abrir Netlogo.

Dónde escribir el código NetLogo para modelizar

- El entorno NetLogo cuenta con 3 pantallas de trabajo: [Ejecutar] donde se controla la simulación y se muestran los resultados, [Información] donde se puede almacenar/consultar información relativa a la simulación (el icono “editar” activa un sencillo editor de texto), y [Código] donde se escribe el código informático ejecutable.
- La pantalla [Código] es un editor de texto especializado para escribir y depurar lenguaje NetLogo: a medida que se van escribiendo instrucciones de programación, el editor las codifica con colores para mejorar la comprensión. Igualmente, al hacer clic sobre el botón [Comprobar] de la barra superior, se comprueban otros aspectos relativos a la coherencia sintáctica y gramatical de la programación Netlogo.

- En cualquier momento se puede almacenar el estado actual de la programación de una simulación. Mediante la opción del Menú Principal *Archivo / Guardar como...* puede grabarse el modelo, incluyendo el interfaz gráfico, la información y el código de programación, todo en un único fichero.

  /!\ El fichero que almacena estos modelos de NetLogo pueden tener cualquier nombre, pero la extensión necesariamente debe ser .nlogo. Es importante comprobar que se incluye .nlogo como extensión del nombre de los modelos al guardarlos.
  Se trata de ficheros ASCII, que pueden ser visualizados y modificados mediante cualquier editor de texto simple.

Cómo ejecutar la simulación de un modelo NetLogo

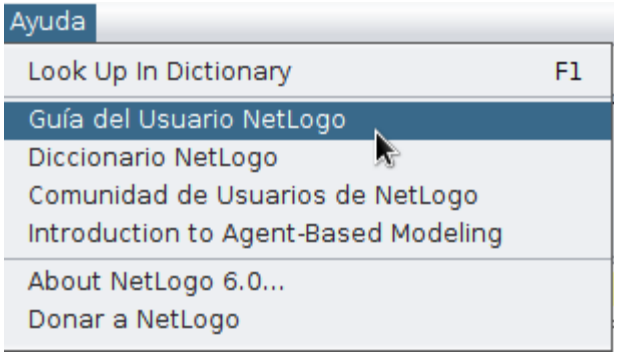
- En primer lugar es necesario "abrir" el modelo dentro de NetLogo, mediante la opción del menú superior *Archivo / Abrir...* (alternativamente, doble clic sobre un fichero *.nlogo en el Explorador de M$-Windows).
- Habitualmente (como parte de las buenas prácticas de programación) cada modelo debe disponer de dos botones de control en la interfaz de usuario: Iniciar (o Setup), y Ejecutar (o Go). El primero establece el estado inicial de la simulación, creando el mundo virtual en el momento t=0, mientras que el segundo inicia el proceso simulado, bien sea paso a paso o bien de forma continua.

  /!\ NetLogo incluye de un amplio catálogo de modelos de simulación disponibles en su página oficial, y una parte de ellos pueden encontrarse en el propio ordenador tras la instalación. Se puede acceder a estos modelos a través de la opción del menú superior Archivo / Biblioteca de Modelos.

- Para poner en funcionamiento la simulación de un modelo: 1) ejecutar el entorno Netlogo, 2) abrir el modelo en NetLogo, 3) en la pantalla [Ejecutar] clic sobre el botón de inicialización, y 4) clic en el botón de ejecución.

Cómo consultar ayuda de NetLogo

- Puedes conseguir información de uso mediante Menú Principal / Ayuda.



- "Guía del Usuario Netlogo" y "Diccionario Netlogo" abre el manual básico de uso de Netlogo en tu navegador favorito (HTML).
- "Comunidad de Usuarios de NetLogo" ofrece acceso al Grupo Yahoo! Que desde 2002 comparte conocimientos sobre este lenguaje de programación.
- "Introduction to Agent-Based Modelling" da acceso al libro "An Introduction to Agent-Based Modeling: Modeling Natural, Social, and Engineered Complex Systems with NetLogo" con el que Uri Wilensky (creador de NetLogo) y William Rand

TUTORIAL NetLogo paso a paso. PRIMERA PARTE (1 de 2)

El problema: "Ir de compras"

Siguiendo este tutorial vas a crear un pequeño mundo social simulado. En esta "sociedad artificial" hay un cierto número de personas simuladas (agentes-compradores) y cada una de ellas tiene unos objetivos (o deseos) a cumplir en forma de su propia "lista de la compra". En el mundo hay también un cierto número de tiendas (agentes-tiendas), cada una de las cuales vende un producto que puede ser adquirido por los compradores. Un agente-comprador tiene capacidad de moverse por el "area comercial" o "mercado", pasar por las tiendas que venden los productos y comprar lo que indica su lista de la compra. Cuando todos los agentes han conseguido su objetivo, esto es comprar todos los productos de sus listas, se consideran satisfechos y la simulación concluye.

[La simulación social, en su modalidad ABM (agent-based modelling), se realiza mediante la construcción y puesta en funcionamiento de sistemas informáticos compuestos por multitud de "agentes". Estos agentes no son más que programas informáticos que interactúan con su propio entorno, que también es virtual; y en el caso de las simulaciones sociales, el entorno esta compuesto principalmente por otros agentes. Los agentes típicamente usados en simulaciones sociales tienen ciertas propiedades específicas: autonomía, habilidad social (lenguaje), reactividad y proactividad ("intencionalidad") y sus acciones se rigen por ciertas reglas que combinan sus deseos, sus creencias y sus oportunidades (teoría DBO).]

Procedimiento general

En primer lugar se construirá el entorno general, esto es, la configuración inicial del "mundo simulado" (un modelo muy simple de un centro comercial con diversas tiendas), y se incorporará a este mundo una primera versión, extremadamente simplificada, de los agentes-compradores. Tras comprobar que funcione esta simulación mínima, se irá reconstruyendo y ampliando recursivamente el modelo con agentes cada vez más complejos cognitivamente. Para poder evaluar tales ampliaciones se establecerá un indicador de **eficiencia agregada del sistema**: el recuento de las unidades de tiempo que tardan todos los agentes en completar todas sus compras en un "mundo" de un tamaño determinado. Posteriormente se comparará la eficiencia de los diversos modelos, sucesivamente mejorados, observando el efecto de la incorporación de mecanismos de "inteligencia" adicional sobre la reducción del tiempo que tardan en completar todas sus compras la totalidad de los agentes.

[Fíjate en cómo el procedimiento descrito es una variante del método experimental clásico, aunque en este caso los "experimentos", con control estricto de variables, no se realizan en un laboratorio físico sino en un laboratorio virtual. Tras establecer un sistema inicial (modelo mínimo, o modelo-0) se introducen sucesivas modificaciones en su diseño, derivadas de la implementación de las diversas hipótesis que se deseen someter a contrastación, para replicar las dinámicas sociales en esta "sociedad artificial" y así poder comparar los resultados con la versión de control anterior (o, eventualmente, con resultados empíricos conocidos).]

Versión 1: "Tontos en el Paraíso"

Los agentes-compradores deambulan aleatoriamente, ciegos, sordos y mudos,

encontrando tiendas y comprando en ellas hasta obtener todos los productos de su lista de la compra. No tienen capacidad de recordar a qué tienda han entrado anteriormente ni qué productos se venden en las tiendas visitadas, por lo que pueden volver a la misma tienda varias veces. Tampoco son capaces de comunicarse con el resto de compradores. Es decir, en esta primera versión, los agentes carecen de capacidad para recordar el entorno conocido ("mapear" el mundo), y sus habilidades sociales ("comunicación" con otros agentes) se restringen a su "interacción" con los agentes-tiendas.

> *[La habilidad social de comunicación implica lógicamente la existencia de una memoria individual, esto es, un registro en el que se almacenen informaciones/creencias, deseos, etc. La "comunicación social" se define como el cambio de estado de esta memoria individual debido a una interacción con otros agentes. En este sentido, obviamente, la interacción con los agentes-tienda (la compra del producto, que implica eliminarlo de la lista de la compra pendiente) es un proceso social. En esta primera versión, los agentes-compradores se "comunican" sólo con los agentes-tienda, pero no con el resto de agentes-compradores. Otras posibilidades de comunicación son posibles y se añadirán en versiones posteriores. Las acciones realizadas por los agentes-compradores modifican sus creencias y deseos, así cuando "entran y compran" en una tienda, eliminan el producto correspondiente se su lista de deseos.]*

Un modelo Netlogo se compone de tres apartados o ventanas: el interfaz gráfico, la ayuda y el código ejecutable. Estos tres contenidos se guardan en un único archivo informático, con extensión **.nlogo**. La parte de código ejecutable debe iniciarse con las definiciones preliminares, y a continuación incluir (en cualquier orden, es indiferente) todos los procedimientos que se usan para ejecutar la simulación. Algunos de estos procedimientos serán activados desde el interfaz gráfico de usuario (GUI), de forma que el modelo se podrá ejecutar y controlar por el usuario mediante el puntero del ratón, o el teclado.

1.1. Definiciones preliminares

En primer lugar es preciso definir los elementos esenciales del modelo, esto es, establecer los componentes del sistema social. En este caso, el modelo supone dos tipos o clases de agentes: los compradores y las tiendas. *(NetLogo permite especificar diferentes clases de objetos como "razas/breeds" diversas)*. La diferencia entre estos dos tipos de agentes está relacionada con cuestiones diversas: 1) la posibilidad de desplazarse o no por

el mundo, 2) la existencia de una memoria interna y la estructura de su contenido, y 3) en su "intención" de conseguir todos los productos de una lista de la compra o bien vender un producto concreto.

Es preciso comenzar el modelo definiendo dos razas:

- una raza de agentes compradores *(compradores)* y
- una raza de agentes tiendas *(tiendas)*.

```
breed [ compradores comprador ]

breed [ tiendas tienda ]
```

> ***/!*** *A lo largo del auto-tutorial aparecerá el código informático en un tipo de letra especial, y dentro de recuadros. Si se tiene un entorno Netlogo en ejecución, se puede usar el portapapeles del sistema operativo para recortar-pegar en el espacio de [Código], aunque se recomienda teclear las instrucciones a mano, para capturar mejor su sentido. Esas instrucciones deben irse escribiendo en el orden que aparecen, para ir construyendo el modelo.*

Cada una de estas "razas" tiene sus propios atributos, esto es, espacios de memoria interna (registros, o variables) donde se almacenaran los datos que cada objeto-agente necesita guardar. Por ejemplo, cada agente-comprador tiene una lista de la compra, que en este caso indica el conjunto de productos que aún faltan por comprar (cuando la lista queda vacía, el agente ha completado su compra). Además, dispone de un espacio genérico de memoria para recordar ciertas cosas *(esta variable no se usa en esta versión tan simple, pero se usará en versiones posteriores)*.

Denominaremos a estos espacios, respectivamente "lista-de-compra" y "memoria".

```
compradores-own [ lista-de-compra memoria ]
```

Por su parte, cada tienda necesita saber cual es el producto que vende.

```
tiendas-own [ producto ]
```

Además, es necesario definir atributos del entorno global del modelo, esto es características generales del "mundo" simulado. Mediante la declaración `globals` definiremos las variables a las que puede acceder cualquier objeto desde cualquier lugar de la simulación. Se aprovechará esta declaración para crear un espacio donde almacenar el catálogo de todos los productos disponibles en esta pequeña sociedad artificial (`productos`).

```
globals [ productos ]
```

En conjunto, la sección de **definiciones preliminares** queda, hasta el momento, así:

```
breed [ compradores comprador ]

breed [ tiendas tienda ]

compradores-own [ lista-de-compra memoria ]

tiendas-own [ producto ]

globals [ productos ]
```

Cuando se ejecute esta primera sección de código informático quedarán definidos dos tipos de agentes, que estarán dotados de sus correspondientes atributos propios, más un atributo global para el mundo. Todos estos espacios de memoria están actualmente vacíos de contenido pero deben ser explícitamente creados de forma previa a la inicialización del mundo artificial en t=0, del mismo modo en que una unidad de almacenamiento magnético (disco duro, USB, etc) debe ser "formateada" antes de introducir ningún dato en ella.

Fíjate en la diferencia entre, por ejemplo, `breed` y `producto`; la primera es una palabra reservada propia del lenguaje NetLogo, mientras que la segunda es el nombre de una variable creada por quien programa el modelo. Estas diferencias quedan destacadas en el editor de código mediante colores diferentes. Obviamente, en el primer caso hay que respetar estrictamente la sintaxis de las palabras-clave, cuya expresión y uso puede encontrase en la opción del menú superior *Ayuda / Diccionario de Netlogo*. Por su parte, las variables inventadas por el usuario deben cumplir con ciertas reglas:

- no pueden contener espacios en blanco *(puede usarse, alternativamente, guiones altos - o bajos _ , p.e. producto_deseado, o producto-deseado)*,

- no pueden contener "caracteres internacionales", esto es, ajenos al inglés como por ejemplo *ñ* o *ç*.

Para más información *(en inglés)* sobre cualquier palabra clave del lenguaje NetLogo, se recomienda escribirla en el editor de código y a continuación pulsar la tecla [F1] en el teclado, lo que abrirá un navegador HTML para poder consultar la información disponible sobre tal palabra clave en el Diccionario de NetLogo incorporado localmente en el momento de la instalación.

1.2. Estado inicial del mundo artificial

Usualmente, en Netlogo se define para cada modelo un *procedimiento* llamado `iniciar` (o *setup*) que establece los valores iniciales de todos los objetos del mundo simulado, esto es, su estado inicial, bien sean atributos de los agentes o del entorno "físico". De este modo se podrá iniciar repetidamente la simulación invocando este procedimiento desde la interfaz gráfica de usuario mediante un botón. Así pues, se trata de:

1. crear en la pantalla de código de NetLogo [Código] el procedimento llamado `iniciar`, compuesto por una secuencia de ordenes, y
2. crear en la interfaz gráfica de usuario de NetLogo [Ejecutar] un botón asociado al mismo.

Para inicializar nuestro modelo necesitamos llevar a cabo diversas operaciones:

- crear algunos agentes, de dos razas distintas, por ejemplo 10 agentes-compradores con capacidad para desplazarse por el mundo, y 12 agentes-tiendas fijas en su ubicación,
- colocarlos aleatoriamente sobre el mundo artificial de NetLogo, y
- dar a cada uno de ellos valores para sus atributos específicos:
  - Para los compradores, una lista de la compra de productos a adquirir.
  - Para las tiendas, el producto concreto que venden.

Comenzamos por declarar que vamos a definir un nuevo procedimiento, y que se llamará "iniciar":

```
to iniciar
```

A continuación se escribirán las instrucciones que pertenecen a este procedimiento. Se suelen escribir con sangría izquierda de un par de espacios en blanco; esto no es estrictamente necesario, pero resulta muy conveniente para mejorar la legibilidad y la inteligibilidad del código. En primer lugar, se creará en este punto una variable "local" para poder identificar más adelante el número de productos disponibles. Esta variable se usa únicamente *dentro de este* procedimiento; como su alcance es local no es necesario definirla globalmente (a diferencia de la anteriormente considerada `productos` que es usada en diversos procedimientos).

```
  let producto-numero 0
```

La inicialización comienza con la puesta a cero del marcador de ciclos temporales y con la eliminación de todos los objetos del mundo virtual, mediante la instrucción pertinente.

```
  clear-all
  reset-ticks
```

El mundo virtual de NetLogo *(la ventana negra de la pantalla Interface)* esta formado por defecto por una cuadrícula ortogonal de 35 x 35 "parcelas" *(patches)*.[3] Estas "parcelas" son agentes sin capacidad para desplazarse, identificados mediante un sistema de coordenadas cartesianas que tiene su origen en el centro de la superficie (0,0).

Pediremos a todas las parcelas que se vuelvan de color amarillo; lo que implica pedirles que establezcan *(set)* su atributo de color *(pcolor)* en un valor concreto "yellow".

```
  ask patches [ set pcolor yellow ]
```

En la parte inferior del espacio de interfaz de usuario [Ejecutar] existe una Terminal de Instrucciones (o Command Center) que permite teclear y ejecutar directamente

[3] Por defecto, esta cuadrícula tiene una topología toroidal, esto es, las parcelas de cada extremo tienen como vecinas inmediatas las del extremo opuesto, de forma que en la práctica se trata de un espacio finito y sin discontinuidades.

instrucciones individuales. Puede comprobarse que si se teclea la orden anterior y la tecla [Intro], todas las parcelas cambian su color de negro a amarillo. Para continuar escribiendo código, regresaremos al espacio de [Código].

A continuación introduciremos un conjunto de datos en nuestra variable global “productos”. En NetLogo al asignar por primera vez cualquier contenido a una variable se está determinando su estructura; en este caso, al introducir un conjunto de datos, la variable tendrá una estructura de conjunto de registros, o vector, o -en terminología Netlogo- de “lista”. De forma similar si introducimos un texto (entrecomillado) la variable tendrá estructura textual o alfanumérica, si introducimos un número entero será numérica simple, si introducimos un número decimal será numérica con precisión doble, y si introducimos los valores `false` o `true` (sin comillas) será de tipo lógico.

```
set productos ["aceite" "bebidas" "cerveza" "detergente"
               "especias" "flores" "gasoil" "huevos"
               "infusiones" "jamon" "kafe" "levadura"]
```

La separación por líneas de esta instrucción es convencional; aquí se ha usado una estructura de indentación, añadiendo espacios en blanco, que facilita la lectura del código. Se puede hacer de otra forma siempre que respete la sintaxis de esta orden, o “primitiva”, que sigue el siguiente esquema: **set *nombre-de-variable* [“*dato 1” “dato 2”* …]**

En el momento de ejecutar esta instrucción, la variable `productos` pasará a contener una lista de 12 items textuales, identificados desde el 0 al 11.

> ***/!\ Atención:*** *Por su funcionamiento interno, en NetLogo los elementos de una lista se identifican por su número de orden* ***comenzando por el 0****, esto es, el item 0 de* `productos` *es “aceite” y el item 3 es “detergente”.*
>
> ***/!\ Atención:*** *Se debe evitar usar en Netlogo caracteres con tilde (letras acentuadas), por tanto* `"jamon"` *y* `"kafe"` *carecen de tilde.*

NetLogo utiliza algunas formas *(shape)* predefinidas para la representación gráfica de los agentes. A continuación se asignará la forma “person” a todos los agentes-compradores, esto es, a todos los agentes de la raza *compradores*:

```
set-default-shape compradores "person"
```

La siguiente orden generará 10 agentes típicos de la raza *compradores*:

```
create-compradores 10
```

Esta instrucción va seguida inmediatamente por una serie de instrucciones adicionales a la definición de tales agentes, entre corchetes []. Los corchetes, de apertura y cierre, son obligatorios, pero pueden colocarse en líneas separadas. La organización aquí propuesta, que implica hacer una segunda sangría izquierda *(cuatro espacios en blanco)*, ayuda a la lectura del código.

Todas las instrucciones entre corchetes se ejecutaran, secuencialmente, para cada uno de los agentes conforme vayan siendo generados.

```
[
```

Mediante diversas variantes de la instrucción ***set*** se asignaran valores (fijos o aleatorios) a los atributos propios de la raza *compradores*, por ejemplo:

Se establece un color *(pink)*, una orientación geográfica *(0 = "mirando hacia arriba")* y un tamaño *(3)*.

```
    set color pink  set heading 0  set size 2
```

Se coloca a cada uno de los compradores en una posición al azar "dentro" del mundo artificial; por lo que se deben restringir sus coordenadas X e Y "entre" los límites máximos de la pantalla o mundo, con la orden:

```
    setxy (random world-width) + min-pxcor

          (random world-height) + min-pycor
```

> ***/!*** *La orden* `setxy 0 0` *ubicaría al agente recién creado exactamente en el centro del mundo. El uso de* `random A` *implica la generación de un número entero aleatorio menor que A según una distribución uniforme. Las variables* `world-width` *y* `min-pxcor` *son propias del sistema (coloreadas al ser reconocidas) y contienen información, respectivamente, sobre la "anchura" del mundo y sobre la coordenada X "más pequeña". Por tanto, el código recién escrito tiene como efecto ubicar al agente en una coordenada X aleatoria*

> *menor que la anchura del mundo pero mayor que el extremo izquierdo, y lo mismo para su coordenada Y. Se asegura así que cada agente esté "dentro" del mundo virtual.*

A continuación, para cada agente, hay que "rellenar" su propia lista de la compra con 10 valores, extraídos de forma aleatoria de entre un conjunto formado por todos los productos disponibles. Cada lista de la compra se construye creando una lista personal de 10 productos (mediante la instrucción `n-values`), seleccionados aleatoriamente (`random`) de entre los que componen la "lista de todos los productos" almacenada en la variable global `productos` *(algunos de ellos pueden estar duplicados en la lista de un comprador)*.

Para conseguir esta asignación de valores concreta se utiliza la combinación de tres "funciones" predefinidas en el lenguaje NetLogo: `item` aplicado a una lista selecciona un elemento en función de su número de orden, `random` aplicado a un valor genera un número entero positivo *(incluye el 0)* menor que el valor de referencia, y `length` aplicado a una lista indica el número de elementos de tal lista.

```
    set lista-de-compra n-values 10
      [ item (random (length productos)) productos]
```

Así, para cada agente recién creado, su atributo lista-de-compra es una lista de 10 productos elegidos al azar de entre los 12 que están en la lista de todos los productos disponibles `productos`.

> ***/!*** *Las ordenes* `set, n-values, item , random` *y* `length` *son "primitivas" propias del sistema. Netlogo dispone de un conjunto de unas 550 funciones elementales predefinidas. La programación de sociedades virtuales implica el conocimiento de estas unidades básicas y su combinación para generar los resultados que interesen a la investigación. Mediante Menú / Ayuda / Diccionario Netlogo se tiene acceso a la descripción, uso y ejemplos de todas ellas.*

A continuación vamos a "rellenar" la memoria con el valor nulo, es decir, se vacía la memoria de cada agente. Este atributo sólo se usará en las versiones posteriores más complejas.

```
    set memoria []
```

Y acaba esta definición específica de la raza de compradores *(create-compradores)*,

mediante el corchete rectangular que cierra el “bloque de instrucciones”.

```
]
```

En cuanto a la modelización de las tiendas: se representaran gráficamente como recuadros, se crearán 12 tiendas, vendiendo cada una de ellas un sólo producto diferente e, igualmente, se distribuirán al azar por toda la pantalla mostrando una pequeña etiqueta roja con el nombre de su producto.

El conjunto de instrucciones para crear e inicializar las tiendas es:

```
set-default-shape tiendas "box"
create-tiendas 12
  [
  set color black set heading 0 set size 2
  setxy (random world-width) + min-pxcor
        (random world-height) + min-pycor
  set producto item producto-numero productos
  set producto-numero producto-numero + 1
  set label producto set label-color red
  ]
```

Fíjate en que, en este conjunto de ordenes hay una estructura lógica CÍCLICA. La orden `create-tiendas 12` genera 12 agentes del tipo tienda, todos ellos ubicados -por defecto- en la parcela central. A continuación se ejecutan las diversas ordenes `set` entre corchetes para cada uno de estos agentes secuencialmente.

Recuerda que previamente, se ha establecido el valor de la variable `producto-numero` en 0. En el caso del primer agente-tienda, cuando se ejecuta la orden `set producto item producto-numero productos` lo que ocurre es que el atributo `producto` de este primer agente se rellena con el valor correspondiente al item situado en la posición `producto-numero` de la variable global `productos`, esto es, en la posición 0 de la lista de todos los productos [`"aceite" "bebidas" "cerveza" "detergente" "especias" "flores" "gasoil" "huevos" "infusiones" "jamon" "kafe" "levadura"`]. El primer agente-tienda ahora sabe su posición en el espacio y, además, sabe que vende “aceite”.

A continuación, la orden `set producto-numero producto-numero + 1` hace que el valor

de la variable `producto-numero` quede establecido en 1, por lo que el siguiente agente "recibirá" como atributo de `producto` el siguiente item de la lista, esto es, "bebidas", y así sucesivamente. Una orden del tipo ***set p p+1*** funciona como un incrementador de valor cada vez que el flujo de ejecución del programa "pasa" por ese punto, es decir, funciona como un contador.

Esta estructura cíclica utiliza además la técnica de programación conocida como "puntero" para avanzar a lo largo de una lista de valores mientras se van asignando secuencialmente los valores del item "apuntado" a diferentes agentes.

Con ello concluye la serie de instrucciones propias del procedimiento "iniciar", y por ello se debe cerrar la definición de procedimiento con la instrucción correspondiente.

```
end
```

Ésta orden está escrita sin sangría izquierda, para dejar claro que cierra la instrucción de definición que se inició algunas líneas más arriba con "to iniciar".

Una vez escrita (o recortada/copiada) esta sección de código es un **buen momento para grabar** el modelo NetLogo. En el menú superior de la pantalla selecciona *Archivo / Guardar como...*, escoge una unidad de disco o USB, una carpeta, dale un nombre al modelo (con extensión **.nlogo**) y guárdalo.

> ***/!*** *A diferencia de otras aplicaciones, Netlogo no guarda automáticamente el trabajo realizado. Es conveniente utilizar con frecuencia Menú / Archivo / Guardar para evitar la perdida eventual de las últimas modificaciones realizadas en el modelo.*

Hasta el momento 1) has definido dos tipos de agentes y algunos de sus atributos, y 2) has creado un procedimiento llamado "iniciar" que colorea de amarillo el mundo simulado, genera un cierto número de cada tipo de agentes, les asigna una forma gráfica según su tipo, los coloca aleatoriamente dentro de los límites del mundo y caracteriza a cada uno asignando ciertos valores a sus atributos. En el caso de los compradores, su lista-de-compra

se asigna de forma aleatoria (muestreo con repetición), mientras que en el caso de las tiendas el producto se asigna exclusivamente, de forma secuencial respecto a una lista.

> *[Como puede observarse cada agente es diferente, aunque dentro de los parámetros establecidos por el modelo para su propio tipo o "raza". Estos parámetros pueden obedecer tanto a requerimientos teóricos, como hipotètico-experimentales, o bien empíricos (si el modelo reproduce determinadas distribuciones de atributos existentes en el mundo real).]*

Para la programación has usando ordenes del lenguaje NetLogo en forma de código informático escrito en el espacio [Código]. Ahora es necesario establecer una interfaz para gestionar el modelo de simulación, de forma que se pueda ejecutar el procedimiento "iniciar" (y los que se añadirán posteriormente). A continuación crearás un "botón" *(button)* en la interfaz de usuario que permitirá ejecutar el procedimiento "inicial" cada vez que se pulse el botón. Para ello abandona la ventana [Código]y trabaja en el entrono gráfico:

- Desplazate a la pantalla o espacio [Ejecutar], haciendo clic sobre la pestaña superior.
- Aparece una nueva barra de herramientas gráficas.
- Haz clic-izquierdo sobre el icono [abc Botón].
- En el desplegable, selecciona el primero de los controles "Botón".
- Situa el cursor (ahora en forma de cruz) en el lugar de la pantalla donde desees ubicar tu botón de acción.
- Hacemos clic-izquierdo. Aparece la pantalla de definición del botón.
- En el apartado central "Instrucciones" teclea el nombre del procedimiento que deseas asociar a este botón, en nuestro caso `iniciar`
- Si quieres que el botón muestre un nombre diferente al del procedimiento introducelo en el apartado "Etiqueta del Botón", por ejemplo: Inicia (con mayúscula).
- Concluye la definición del botón, haciendo clic en el botón inferior [OK].

Es el momento de comprobar el procedimiento de inicialización del mundo artificial: Pulsa el botón que acabas de crear. Al hacerlo se ejecutará el procedimiento `iniciar`, Netlogo "leerá"

el código y lo convertirá en un procedimiento que actualizará el mundo virtual. Deberían aparecer 10 compradores de color rosa, cada uno con su lista de la compra de 10 productos a adquirir, y una tienda para cada uno de los 12 productos disponibles, etiquetadas en rojo según su propio producto; todos ello diseminado aleatoriamente sobre un mundo simulado de color amarillo. Ese es el centro comercial “Paraíso de la compra”.

Una vez establecido el mundo virtual en su estado inicial y creados los agentes constituyentes de su población es posible acceder a la información “interna” de cada agente, y consultar el valor de sus atributos. Para comprobar la lista de la compra de un agente-comprador, en la interfaz gráfica [Ejecutar]:

- Sitúa el cursor (una flecha) sobre el comprador que se desees examinar,
- Haz clic-derecho, aparecerá un menú contextual,
- Selecciona el objeto a examinar (*el cursor puede estar señalando una parcela y, eventualmente diversos agentes*), por ejemplo el “comprador 5” *(los agentes aparecen en la parte inferior del menú contextual)*,
- Selecciona la opción “inspect” para mostrar la ventana de atributos de tal objeto,
- Consulta el contenido de la variable lista-de-compra. Puedes hacer clic dentro de cualquier atributo y desplazar el cursor de texto.

Estas ventanas de propiedades o atributos de los diferentes agentes NetLogo pueden moverse, o cerrarse tecleando [Esc] o haciendo clic en la “x” situada en la parte superior derecha.

- Abre las ventanas de dos agentes de tipo comprador y un agente tienda.
- ¿En qué atributos son similares y en qué atributos son distintos?

> *[Como puedes comprobar TODOS los agentes comparten ciertos atributos (p.e., color, posición XY) pero diversas “razas” pueden tener distintos conjuntos de atributos, y dentro de un mismo tipo, cada agente tiene contenidos distintos en cuanto a los atributos propios de su tipo. La población sintética que constituye la sociedad artificial generada está constituida por agentes heterogéneos.]*
>
> **/!\ *Modelos empíricamente calibrados (ECM):*** *Se denominan modelos*

"calibrados" a aquellos constituidos por agentes cuyos atributos (en términos de distribución poblacional) modelan distribuciones de atributos observados en poblaciones reales. Por ejemplo, si se observa evidencia empírica de que la población de personas compradoras en Areas comerciales está compuesta por un 80% de género femenino y un 20% de género masculino, podríamos añadir el atributo "género" a los atributos de nuestros agentes-compradores, y "iniciar" nuestra población sintética calibrada con 8 agentes mujeres y 2 hombres.

1.3. La ejecución de la simulación

Una vez que todos los "objetos" del mundo artificial *(entorno físico como los agentes-tienda, entorno social como los agentes-compradores, o estados mentales como los deseos y creencias)* han sido inicializados, es preciso un procedimiento que "ponga en marcha" la simulación, así como el correspondiente botón en el interfaz gráfico para activar tal procedimiento.

A continuación crearás el procedimiento `ejecutar` que 1) le pedirá a cada agente que "actúe", esto es, que si aún no ha "tachado" todos los productos de su lista-de-compras busque una tienda para comprar los productos deseados, y 2) a continuación comprueba si la simulación debe finalizar. La condición de detención de la ejecución de este modelo es que todos los compradores hayan adquirido todo lo que indicaban sus respectivas listas.

[Una de las ventajas de la arquitectura de programación del lenguaje NetLogo es su modularidad. En lugar de definir un procedimiento de "actuación" con <u>todas</u> las especificaciones de la simulación resulta conveniente dividir el flujo de ejecución en diferentes procedimientos, definirlos por separado y reconstruir el flujo escribiendo el código informático de forma que unos procedimientos sean llamados o ejecutados <u>desde otros</u>. Esta estrategia permite programar diferentes niveles de ejecución, permite reutilizar y modificar fácilmente el código (al retocarlo en lugares concretos), y facilita la lectura convirtiendo numerosas líneas en simples nombres de procedimientos.]

*[Pero estas ventajas, de carácter meramente técnico, se complementan con otras ventajas de naturaleza metodológica que atañen especialmente a una aproximación "analítica" a la explicación en las CCSS (fuertemente vinculado con la simulación social). La mencionada estrategia obliga a establecer teoricamente, con un elevado nivel de claridad, distinción y detalle, los diversos **procesos** causales que operan en el mundo o fenómeno simulado, así como los **mecanismos** que los desencadenan.]*

Definiremos, de forma modular, un procedimiento general denominado `ejecutar`, que

contendrá una llamada a otro procedimiento principal `actuar`, más el establecimiento de la condición de detención de la simulación y otras ordenes pertinentes. Por otro lado, definiremos el procedimiento denominado `actuar`, que comprobará para un agente concreto si su lista de la compra esta vacía:

- Si la lista no está vacía le enviará a realizar el procedimiento `comprar`, y
- Si la lista ya está vacía, es que el comprador a acabado sus compras.

En el primer caso el agente-comprador continuará ejecutando la simulación, usando el procedimiento `comprar`, definido en otro lugar del código; mientras que en el segundo caso el comprador simplemente permanecerá en su ubicación actual (marcado con un cambio al color azul) y la simulación continuará considerando el siguiente agente-comprador hasta revisarlos a todos.

```
to ejecutar
  ask compradores [ actuar ]
  if count compradores with [ not empty? lista-de-compra ] = 0
       [ stop ]
  tick
end

to actuar
  ifelse not empty? lista-de-compra
   [ comprar ]
   [ set color sky ]
end
```

Como puede apreciarse, ambos procedimientos trabajan en niveles diferentes:

- `ejecutar` es un procedimiento general que adopta la perspectiva *sistémica o macroscópica* del "observador por encima de los agentes". Como tal les pide a todos que lleven a cabo otro procedimiento, evalúa condiciones referidas al conjunto de agentes, o realiza acciones sobre el conjunto de la simulación.
- `actuar` es un procedimiento particular que adopta la perspectiva *individual o microscópica* del "agente que actua en aquel preciso momento" y como tal las

alusiones que incluye se refieren a un agente en concreto.

El funcionamiento de esta simulación, por tanto, puede entenderse como una articulación entre diferentes niveles: el macroscópico que atañe al sistema como conjunto y el microscópico que atañe a cada uno de los agentes en particular [4]. La simulación comienza su ejecución en el nivel macro cuando se pide a todos los agentes que actúen, lo que significa que serán requeridos uno a uno a cumplimentar las ordenes de cada procedimiento particular, y sólo cuando el último agente-comprador haya "actuado" se actualizará el gráfico, se comprobará la condición de finalización y, en caso de no satisfacerse, se aumentará el contador de tiempo simulado (`ticks`) y se volverá a comenzar la nueva ronda pidiendo a todos los compradores que actúen. Si se satisface la condición de que todos los agentes-compradores ya han comprado todos los productos de sus listas *(stop condition)*, entonces la simulación se detiene.

Respecto al procedimiento "ejecutar": la instrucción `tick` tiene como efecto aumentar en uno el valor de la variable global `ticks`, esto es, una variable predefinida del sistema que mantiene un recuento el número de ciclos temporales transcurridos durante la ejecución de la simulación. Así la orden `tick` colocada al final del procedimiento `ejecutar` hace *"avanzar el tiempo"* antes de iniciarse una nueva ronda de acciones para todos los agentes.

> ***/!*** *El tiempo en NetLogo es, como todo lo demás, simulado y por tanto hay que incorporar, explícitamente, un mecanismo de avance de tiempo en algún lugar de la programación. Por otro lado también es importante notar cómo la "acción" de los agentes no es simultánea sino secuencial, lo que puede resultar en una incorrecta simulación de un sistema social real (concurrente) si no se presta la suficiente atención al flujo de acciones, interacciones y al momento concreto del "avance de tiempo". NetLogo hace actuar al conjunto de actores según una secuencia aleatoria, y además contiene otras instrucciones para conseguir programar sistemas más complejos dotados de "concurrencia virtual".*

Una vez escritas o pegadas las líneas de programación correspondiente a los dos

[4] Pueden añadirse otros niveles para modelizar mayores complejidades en el sistema, como ejemplo, por debajo del nivel de "actuación" de un agente podría implementarse con mayor detalle un nivel de "deliberación", para reproducir los procesos internos cognitivos de aprendizaje, creencias, motivaciones, emociones o razonamiento que resultan en una acción concreta.

procedimientos comentados en la pantalla del editor de código, es necesario añadir algunos elementos más en la pantalla del interfaz gráfico.

Añadir un Botón: "Simula"

De forma similar a como has procedido en el caso del botón "Iniciar", tienes que generar un control gráfico para poner en marcha la simulación. Así pues:

- Desplázate a la pantalla [Ejecutar].
- Haz clic sobre el icono [abc Botón].
- Selecciona el primero de los controles "Botón".
- Sitúa el cursor (ahora en forma de cruz) en el lugar de la pantalla donde desees ubicar el nuevo botón.
- Haz clic-derecho. Aparece la pantalla de definición del botón.
- En el apartado central "Instrucciones" teclea el nombre del procedimiento que deseas asociar a este botón, en nuestro caso `ejecutar`
- Si quieres que el botón muestre un nombre diferente lo tecleas en el apartado "Etiqueta del botón", por ejemplo: Simula.

> ***/!*** *Es un buen momento para guardar las últimas modificaciones. Recuerda usar frecuentemente Menú / Archivo / Guardar.*

Así, hasta el momento, haciendo clic sobre el botón "Inicia" se ejecutaran todas las instrucciones contenidas en el procedimiento `iniciar`, mientras que haciedo clic sobre "Simula" se ejecutaran todas las instrucciones de `ejecutar`, esto es, una ronda de ejecución de `actuar` para cada agente-comprador más las actualizaciones pertinentes.

El botón "Simula" es un botón de ejecución *paso a paso* de la simulación, pero lo que interesa es una ejecución *continuada* (que se detendrá en cuanto se cumpla la condición de detención que ya has establecido mediante programación). Para modificar el botón "Simula" y convertirlo en botón de ejecución continuada:

- En la pantalla [Ejecutar], coloca el cursor sobre el botón [Simula].
- Haz clic-derecho sobre el botón a modificar. Aparece un menú contextual.
- Selecciona la opción “Edit...”
- En la ventana emergente de atributos del botón, activa el recuadro “Continuamente” (arriba, a la izquierda).
- Haz clic en el botón “OK” para aceptar la modificación.
- Comprueba si aparece un signo de dobles flechas circulares denotando la recursividad del botón. Ahora el “botón” es un “interruptor”: al pulsarlo se ejecutará repetidamente el procedimiento indicado, al pulsarlo de nuevo la ejecución queda en pausa.

[El código informático que permite generar los componentes de la sociedad artificial y los procedimientos que rigen la interacción entre tales componentes es el modelo de simulación. El modelo, en sí mismo, no lleva a cabo ningún proceso; nada se mueve. Por el contrario, el proceso de “puesta en marcha” de la sociedad artificial y de actualización de atributos de agentes es la simulación del modelo. No hay simulación hasta que no se pulsan los botones que disparan procedimientos.]

Añadir un monitor: “Tiempo”

Dado que el indicador de eficiencia que se utilizará para evaluar los diferentes modelos será el *“tiempo que tardan los compradores en realizar todas sus compras”*,[5] es necesario añadir un control al interfaz gráfico para hacer un seguimiento de tal indicador y obtener su valor final cada vez que se ejecute una simulación del modelo. Se aprovecha la existencia de la variable predefinida global del sistema `ticks`, actualizada tras cada turno de ejecución mediante al orden `tick`, para visualizar el tiempo simulado transcurrido en un monitor.

- Desplázate a la pantalla [Ejecutar].
- Haz clic sobre el icono [abc Botón].
- Selecciona el control “Monitor”.
- Sitúa el cursor (en forma de cruz) en el lugar de la pantalla donde deseas ubicar el

[5] Se trata del tiempo simulado (ticks) transcurrido desde el inicio de la simulación hasta que el último agente-comprador adquiere el último producto de su lista-de-compra.

monitor.

- Haz clic. Aparece la pantalla de definición del monitor.
- En el apartado “Monitor” introduce el nombre de la variable a mostrar: `ticks`
- En el apartado “Etiqueta del Monitor” teclea, por ejemplo: Tiempo
- En el apartado “Núm. Decimales” indica: 0.
- Concluye la definición del monitor, haciendo clic en el botón inferior [OK].

Añadir otros procedimientos específicos

Para los agentes que pueblan esta pequeña y simple sociedad artificial, “actuar” significa comprar, siempre y cuando aún queden productos por ser adquiridos según la lista de la compra propia de cada comprador. Una vez descompuesto analíticamente el proceso de compra, para expresarlo como el resultado de una secuencia de procesos subyacentes, “comprar” consiste, desde el punto de vista de un agente-comprador, en:

1. comprobar si hay algún agente-tienda en la misma parcela (*patch*) en la que se encuentra el agente-comprador actualmente,
2. si es el caso, (por si hay más de una) elegir al azar una de tales tiendas, y “comprar” en ella, y
3. después avanzar por el Centro Comercial “aleatoriamente”.

Tal es el contenido del procedimiento `comprar`, que programarás como sigue:

```
to comprar
  if any? tiendas-here [ compra-si-necesitas ]
  avanzar-al-azar
end
```

En este procedimiento, `tiendas-here` informa al agente-comprador sobre las tiendas que se encuentran en su misma localización. Se trata de un mecanismo de percepción del entorno, que puede activar una acción de respuesta. Si no se trata de un conjunto vacío, es decir si existen tiendas en la parcela en que se encuentra, se llama al procedimiento `compra-si-`

`necesitas`, por el contrario, si no hay ningún agente-tienda en la misma parcela en que se encuentra el agente-comprador actual se pasa a ejecutar el procedimiento `avanzar-al-azar`.

Fíjate en la diferencia entre ambas ordenes: para cada ciclo de tiempo simulado un agente avanzará en todo caso (incondicionalmente), pero sólo intentará comprar si se encuentra sobre una parcela que contenga al menos una tienda (condicionalmente).

> *[Toda la especificación, en términos de código informático, del modelo de simulación está compuesta por un conjunto de reglas de acción, algunas incondicionales, otras condicionales. Es la forma de modelizar los llamados mecanismos sociales, que una parte de la teoría sociológica reclama como elemento fundamental de la explicación científicamente aceptable de fenómenos sociales.]*

Sigue construyendo el modelo, definiendo ahora el procedimiento `compra-si-necesitas`:

```
to compra-si-necesitas
  let tienda-visitada 0
  set tienda-visitada one-of tiendas-here
  if member? [producto] of tienda-visitada lista-de-compra
     [ set lista-de-compra remove [producto] of tienda-visitada lista-de-compra ]
end
```

En primer lugar el agente-comprador considera que puede haber más de una tienda en la misma parcela, de modo que, usando la variable local `tienda-visitada`, el agente implicado selecciona al azar una de ellas. Lo hace tomando el conjunto de tiendas que hay en la parcela en que se encuentra `tiendas-here`, tomando una muestra aleatoria de una de tales tiendas `one-of` y asignando esa tienda a la variable `tienda-visitada`.

A continuación se comprueba si el producto que se vende en esa tienda `[producto] of tienda-visitada` forma parte `member?` de la lista de la compra del agente en cuestión `lista-de-compra` y actúa en consecuencia.

1. Si es el caso, el agente elimina `remove` tal producto de su lista (i.e., ha “comprado” el producto).
2. Si el producto vendido no forma parte de la lista de la compra, bien porque nunca estuvo, bien porque ya fue “comprado” anteriormente, concluye el procedimiento `compra-si-necesitas` sin efecto alguno *(para el comprador actual)*.

Finalmente, el procedimiento `avanzar-al-azar` consiste, analíticamente, en la secuencia:

1. establecer el atributo `heading` del agente-comprador *(i.e., la dirección hacia la que se moverá)* en un ángulo aleatorio menor a 360, y
2. entonces avanzar una unidad (parcela) hacia el que actualmente sea el rumbo de la marcha.

Esto también ha sido definido, mediante la lógica de la programación modular, a través de dos procedimientos diferentes. Escribiremos las siguientes líneas de código:

```
to avanzar-al-azar
  set heading (random 360)
  avanzar
end

to avanzar
  forward 1
end
```

- Tras guardar las últimas modificaciones, haz clic al botón [Inicia] y al botón [Simula].

Añadir un gráfico: "Evolución de compradores satisfechos"

Para expresar los datos resultado de la simulación, en este caso la evolución en el tiempo del indicador seleccionado para evaluar la eficiencia del sistema (el número de compradores satisfechos), puede utilizarse un gráfico actualizado "en tiempo real" en la interfaz de usuario. Es preciso añadir este nuevo elemento, junto a los botones y el monitor ya existentes.

- Desplázate a la pantalla [Ejecutar].
- Haz clic sobre el icono [abc Botón].
- Selecciona el control "Gráfico".
- Sitúa el cursor (en forma de cruz) en el lugar de la pantalla donde desees ubicar el gráfico.
- Haz clic. Aparece la pantalla de definición del gráfico.

- En el apartado “Nombre” introduce el nombre del gráfico: Evolución de la satisfacción
- En el apartado “Etiqueta del eje X” introduce la etiqueta: Tiempo, y en “X máx.” 20000
- En el apartado “Etiqueta del eje Y” introduce la etiqueta: %, y en “Y máx.” 100
- En el apartado “Escala automática?” desactiva tal opción.
- En el apartado “Trazos del gráfico” busca la columna “Instrucciones de Actualización de Trazos”, haz clic y borra todo contenido.
- Para finalizar haz clic en el botón inferior [OK].
- Si se quiere modificar la posición o el tamaño del gráfico: Hacer clic-derecho con el ratón sobre el gráfico, en el menú contextual seleccionar “Select”, utilizar los tiradores negros para modificar el formato del gráfico, o “arrastrarlo” con el ratón a la ubicación deseada.

Para que se actualice el gráfico en cada momento de tiempo (simulado) hay que modificar un procedimiento que se ha escrito anteriormente: ejecutar. Se trata de introducir algunas líneas más de código.

- Desplázate a la pantalla [Código].
- Busca el procedimiento ejecutar. Puedes desplazarte por el código con el cursor, o utilizar el menú desplegable buscador de procedimientos, situado en la barra de herramientas [Procedimientos ∨].
- Añade las líneas de código correspondientes, hasta que el procedimiento tenga el siguiente contenido:

```
to ejecutar
  ask compradores [ actuar ]
  if Grafico? [
     plot count compradores with [ empty? lista-de-compra ] * 10
     ]
  if count compradores with [ not empty? lista-de-compra ] = 0
       [ stop ]
  tick
end
```

Adicionalmente, para controlar desde la interfaz gráfica si se muestra o no el gráfico hay que añadir a la misma otro elemento de control: un “interruptor”.

- Desplázate a la pantalla [Ejecutar].
- Haz clic sobre el icono [abc Botón].
- Selecciona el control “Interruptor”.
- Sitúa el cursor (en forma de cruz) en el lugar de la pantalla donde desees ubicar el interruptor. Un buen lugar es justo en la parte inferior del gráfico que activa/desactiva.
- Haz clic. Aparece la pantalla de definición del interruptor.
- En el apartado “Variable Global” teclea: Grafico? (el mismo nombre que acabas de añadir al procedimiento iniciar).
- Para finalizar haz clic en el botón inferior [OK].

En ocasiones interesa visualizar los resultados “en tiempo real” para analizar su evolución, pero esto puede ralentizar la ejecución de la simulación. En ocasiones interesa conocer únicamente los resultados finales agregados, rápidamente y sin gráficos. Mediante el interruptor recién creado se puede activar o desactivar la actualización del gráfico.

1.4. Evaluación de la eficiencia del modelo básico

El resultado agregado del conjunto de procedimientos es que los agentes-compradores se dedican realizar un “paseo al azar” por toda la cuadrícula que modeliza el Centro Comercial (el “Paraiso de la Compra”). Dada una cantidad suficiente de ciclos de ejecución de la simulación, los compradores pueden llegar a pasar por todas las parcelas en que se ubican las tiendas, y desde luego transcurrirán muchos `ticks` hasta que todos ellos visiten todas las tiendas que necesitan para vaciar sus listas de la compra.

Ahora que el código esta completo, en sus aspectos fundamentales, ya puede inicializarse y ejecutarse tantas veces como se desee. Debido a la naturaleza fundamentalmente azarosa de los movimientos de los agentes, así como a las diferentes posiciones iniciales de compradores y tiendas, y la diversidad en las lista de la compra, el tiempo que tarda cada simulación en completarse variará con cada ejecución concreta.

Comprueba esto, haciendo clic en los botones correspondientes de la interfaz gráfica para inicializar el mundo y ejecutar varias simulaciones del modelo. Consulta, en el monitor, el tiempo final de cada una de ellas.

- Apunta el resultado obtenido ("Tiempo") para cada simulación realizada,
- Inicializar el modelo y replicar simulaciones un cierto número de veces (al menos 10),
- Calcula la media aritmética de Tiempo hasta que el sistema social simulado alcanza el punto de satisfacción.
- Has usado la metodología de Simulación Social como técnica de replicación de experimentos.

En una serie de 100 ejecuciones se obtuvo un valor `tick` final, cuando todos los agentes han comprado todo lo que necesitan, **promedio de 14.310** (con desviación estándar de 4.150). Los agentes de esta primera versión del modelo son altamente ineficientes en cuanto a la localización de las tiendas que necesitan debido a su comportamiento sordo, ciego y mudo, basado en recorrer al azar el centro comercial. En definitiva, el modelo simula las actividades de un conjunto de agentes fundamentalmente "asociales".

Fin de la primera parte

En la segunda parte de este tutorial, podrás incrementar las capacidades cognitivas y sociales de los agentes. Crearás paso a paso versiones del modelo existente el las cuales se añaden mecanismos como la percepción del entorno *(¿qué tienda es aquella?)*, el uso de una memoria o mapa mental del mundo *(¿por qué tiendas he pasado antes, y donde están ubicadas?)*, e incluso la comunicación entre agentes *(Hola comprador, yo conozco la ubicación de estas tiendas ¿tú cuales conoces?).*

Cada modificación del modelo será ejecutada hasta generar resultados agregados comparables con la versión inmediata anterior. La contrastación del indicador "tiempo total de compra" permitirá ejemplificar la potencia explicativa del uso de modelos de simulación social en la evaluación y el desarrollo de hipótesis relativas a cómo se generan procesos sociales.