5th International Conference on Industrial Engineering and Industrial Management
XV Congreso de Ingeniería de Organización
Cartagena, 7 a 9 de Septiembre de 2011

Creación de programas multi-participante en NetLogo. Aplicaciones en Ingeniería de Organización.

Segismundo S. Izquierdo[1], José A. Pascual[1], Pedro Sanz[1], Cesáreo Hernández[1]

[1] Dpto. de Organización de Empresas y CIM. Escuela de Ingenierías Industriales. Universidad de Valladolid. Paseo del cauce 59, 47011 Valladolid. segis@eis.uva.es, pascual@eis.uva.es, psangulo@eis.uva.es, cesareo@eis.uva.es.

Palabras clave: NetLogo, simulación, programación informática, prácticas docentes.

1. Introducción

NetLogo es un entorno de programación gratuito enfocado a la simulación de modelos basados en agentes (Wilensky 1999). NetLogo ofrece un módulo denominado HubNet (Wilensky and Stroup 1999) que facilita la creación de aplicaciones interactivas en las que los participantes (clientes) disponen de una pantalla con cierta información desde la que pueden enviar sus decisiones a un nodo central (servidor) que puede agregar o combinar esa información según proceda y comunicar los resultados de la interacción de decisiones a los clientes. Los clientes pueden entonces modificar sus decisiones a la vista de los resultados.

HubNet facilita el diseño y la realización de experimentos de interacción social. En particular, en el campo de la ingeniería de organización, es útil para realizar simulaciones de procesos logísticos, tráfico, mecanismos de intercambio (subastas a sobre cerrado, cruces oferta-demanda,…) o decisiones de competencia empresarial bajo distintas estructuras de mercado (simuladores empresariales). Estas aplicaciones pueden usarse tanto para la docencia como para la investigación. Con carácter general, HubNet permite simular sistemas complejos basados en la interacción de las decisiones de distintos individuos.

En este trabajo presentamos la estructura típica de un programa HubNet tomando como base la programación de un modelo sencillo y referimos algunos casos de ejemplo más avanzados (Pascual et al. 2009). Nuestro objetivo es transmitir una idea del proceso de creación de aplicaciones mediante HubNet y servir como guía de aproximación para profesores interesados o bien en desarrollar directamente este tipo de aplicaciones para sus clases y su investigación, o bien en plantear el desarrollo de aplicaciones sobre esta plataforma como proyectos para alumnos de cursos avanzados (principalmente proyectos fin de carrera o cursos de doctorado). Los ejemplos de código y pantallas que vamos a utilizar están basados en el programa de ejemplo denominado “Template” que viene incluido con Netlogo (ver la sección sobre derechos de reproducción al final del documento). El programa de instalación de Netlogo puede descargarse desde la dirección http://ccl.northwestern.edu/netlogo/. Aparte del manual de referencia que puede encontrarse en esas mismas páginas, existen varios otros manuales de introducción a la programación en NetLogo fácilmente accesibles en la red, tanto

en inglés (Izquierdo 2007) como en castellano (Poza 2009). Al programa "Template" se accede a través del menú File → Models Library → Hubnet Computer Activities → Code Examples → Template.

2. El entorno gráfico de trabajo

2.1. La pantalla de participante

La pantalla de participante permite a los jugadores transmitir sus decisiones al ordenador del profesor, llamado también el ordenador servidor, o simplemente "el servidor" (palabra que en los tiempos actuales podría confundirse con el propio profesor) y recibir resultados e información provenientes de este último.

A la zona de diseño de la pantalla de participante se accede mediante la ruta Tools → Hubnet Client Editor. Para el modelo de ejemplo "Template" obtenemos la vista de diseño que se muestra en la Figura 1.

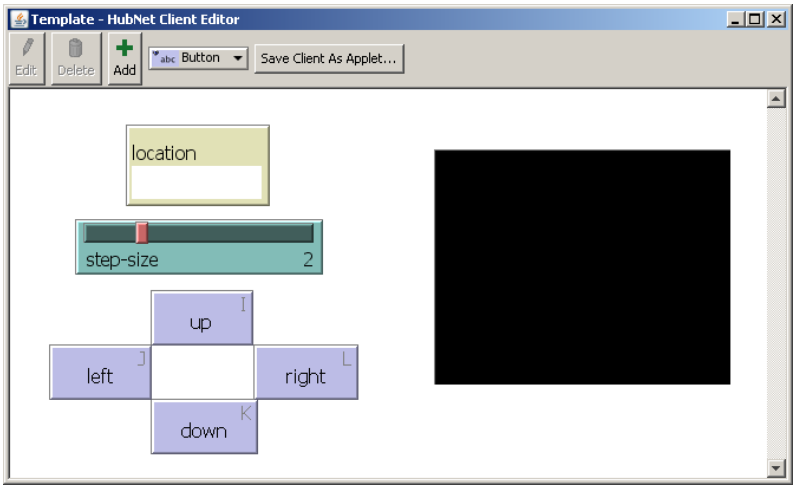

Figura 1. Zona de diseño de la pantalla de participante correspondiente al modelo "Template".

Un menú desplegable en la parte superior de la zona de diseño permite introducir en la pantalla del participante:

– Elementos de recepción de información, como el cuadro llamado "location" en la figura 1. Principalmente se trata de cuadros informativos que mostrarán texto o números que pueden actualizarse durante la ejecución del programa. También puede tratarse de un área denominada "view" (cuadro en negro en la figura 1) que puede mostrar información espacial. Aunque es también posible mostrar gráficos, no se introducen desde esta zona de diseño.

– Elementos para reflejar selección de decisiones, como el selector de valor de variable denominado "step-size" en la figura 1.

– Elementos para tomar decisiones y enviar la información asociada. Principalmente botones para tomar y transmitir determinadas decisiones o acciones, como los que aparecen en la parte inferior izquierda de la figura 1.

La posición de los distintos elementos en la pantalla puede reorganizarse fácilmente seleccionando y arrastrando con el ratón.

2.2. La pantalla de profesor

La pantalla de profesor puede contener el mismo tipo de elementos que la pantalla de usuario y además puede también contener gráficos. Estos gráficos pueden mantenerse exclusivos para el profesor o mostrarse también a los participantes.

Típicamente, la pantalla de profesor contendrá información más o menos agregada sobre los participantes y varios botones asociados al control de la práctica: botones para inicializar algunas variables, botones para parar o continuar el tráfico de información, etc.

3. Control global de una práctica

Para realizar una práctica participativa con Netlogo existen dos opciones:

a) Trabajar en ordenadores conectados en red en los que se ha instalado NetLogo. El profesor debe entonces arrancar NetLogo y los participantes deben arrancar la aplicación que se instala automáticamente junto con NetLogo denominada "HubNet".

b) Instalar NetLogo y un servidor web en el ordenador del profesor. En este caso los participantes pueden acceder sin necesidad de instalar NetLogo: mediante un navegador web y cargando un applet Java que debe instalarse en el ordenador del profesor.

Cuando el profesor abre una práctica, aparece una pantalla como la que se muestra en la figura 2.

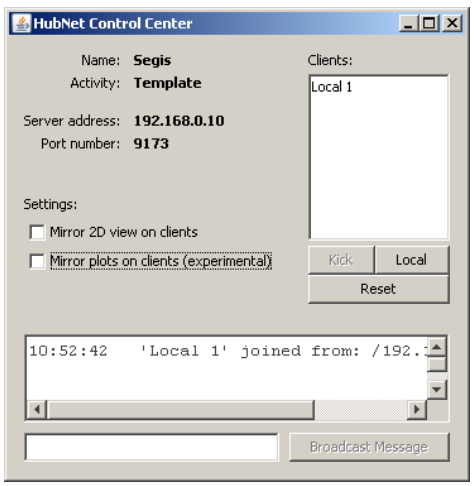

Figura 2. Pantalla de control global de una práctica.

Esta pantalla de control permite:

– Ver los nombres de los participantes que se van conectando a la práctica y expulsar selectivamente.

– Crear pantallas de participante en el ordenador del profesor, las cuales se van creando con los nombres de usuario “Local 1”, “Local 2”,...

– Decidir si se muestra o no a los participantes determinada información: la que aparece en los gráficos y en el área denominada “view”.

– Desconectar a todos los participantes y comenzar la práctica de nuevo.

Cuando un participante abre la aplicación HubNet, le aparece una pantalla como la que se muestra en la figura 3.

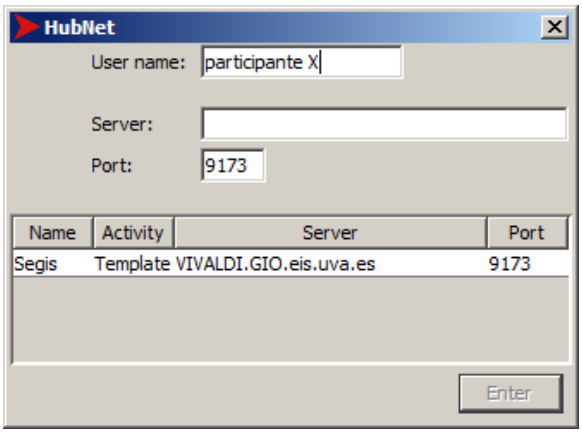

Figura 3. Pantalla de conexión de un participante a una práctica.

En esta pantalla el participante observa la lista de prácticas o actividades que el profesor ha arrancado y a las cuales puede conectarse. El participante debe introducir un nombre que lo identifique y seleccionar una actividad de la lista para conectarse. A continuación accede a la pantalla de participante de la práctica seleccionada.

4. Estructura típica de un programa en HubNet

Comentaremos la estructura típica basándonos en el modelo de ejemplo “Template” incluido con NetLogo.

4.1. La pantalla y variables de usuario

En el modelo “Template” la pantalla de usuario (ver figura 1) viene creada con un cuadro informativo llamado “location”, un selector de valor de variable denominado “step-size” y cuatro botones de comando denominados “up”, “down”, “left” y “right”. También contiene el área denominada “view” (cuadro en negro en la figura 1) que puede mostrar información espacial.

Cada vez que un usuario modifica el valor de un selector de variable como el “step-size”, o pulsa un botón de comando, o pulsa sobre un punto del área “view”, se envía un mensaje al servidor con información asociada.

4.2. Definición de variables globales

Entre ellas el nombre genérico de los agentes que vamos a usar y sus variables asociadas. Normalmente un agente se identifica con un participante. Para el modelo "Template" encontramos el siguiente código tipo:

breed [students student]	*Breed* indica que vamos a usar un tipo de agente cuyas variables asociadas definiremos después y al que damos el nombre genérico de *students*. Un agente particular de este tipo será un *student.*
students-own [	A continuación se define el nombre de las variables asociadas a cada agente *student*, definidas entre los símbolos *"["* y *"]"*.
user-id	Creamos la variable *user-id* para guardar el nombre asociado a cada agente *student*. Haremos que tome el nombre elegido por cada participante al acceder.
step-size]	Creamos la variable *step-size* asociada a cada agente *student*. Cada agente elegirá un valor para esta variable.

4.3. El procedimiento startup

Un procedimiento es una subrutina o conjunto de instrucciones al que podemos referirnos con el nombre del procedimiento. El procedimiento *startup* es un procedimiento especial porque se ejecuta sólo al abrir el programa.

to startup	Indica que a continuación se define el procedimiento denominado *startup*
setup	Llama al procedimiento *setup*, que está definido más adelante y que inicializa las variables.
hubnet-set-client-interface "COMPUTER" []	Indica que los participantes van a usar ordenadores para conectarse.
hubnet-reset	Crea una sesión para que los participantes puedan conectarse y abre el centro de control de la sesión.
end	Indica el fin de la definición de un procedimiento

4.4. Inicialización de variables

Las instrucciones de inicialización de variables se suelen agrupar en un procedimiento al que normalmente se le da el nombre *setup*:

to setup	Indica que a continuación se define un procedimiento denominado *setup*
clear-patches clear-drawing	Limpia el área denominada *view*
clear-output	Limpia los cuadros de texto que pueda haber
ask turtles [	Indica que a continuación se definen una serie de instrucciones que serán ejecutadas secuencialmente por todos y cada uno de los agentes. *Turtles* es el nombre genérico de todos los agentes en NetLogo. La variable *user-id* identificará al agente en particular que está corriendo el comando.
set step-size 1	Hace que la variable *step-size* del agente tome el valor 1.
hubnet-send user-id "step-size" step-size	Envía a la pantalla de usuario, al control llamado "step-size" del agente que está corriendo el comando (el agente *user-id*) el nuevo valor de su variable *step-size*.
]	Indica el final de la secuencia de instrucciones que comenzó con *ask turtles [*
end	Indica el final de la definición del procedimiento

4.5. Procedimientos de control de entrada y salida de participantes y de distribución de acciones asociadas a cada tipo de mensaje

En el modelo "template", el siguiente procedimiento *go* regula cuándo enviar información a los participantes y llama al procedimiento *listen-clients* definido más adelante para procesar los mensajes recibidos de los participantes.

to go	Indica que a continuación se define un procedimiento denominado *go*.
listen-clients	Llama al procedimiento *listen-clients*, definido más adelante, para procesar los mensajes que vayan llegando de los participantes.
every 0.1 [display]	Hace que cada 0.1 segundos se actualice el área gráfica "view" en las pantallas de los participantes.

end	Fin de la definición del procedimiento.

En el modelo "Template" al procedimiento principal de esta parte se le ha llamado *listen-clients*, y decide la acción a tomar según el tipo de mensaje recibido de los participantes. *Listen-clients* elige entre tres procedimientos de detalle: *create-new-student*, que definirá las instrucciones a ejecutar cuando un participante se conecta; *remove-student*, que definirá las instrucciones a ejecutar cuando un participante se desconecta; y *execute-command*, que definirá las instrucciones a ejecutar cuando el mensaje no es de conexión o desconexión.

to listen-clients	Indica que a continuación se define un procedimiento denominado *listen-clients.*
while [hubnet-message-waiting?]	Mientras haya mensajes de los participantes pendientes de procesar.
[	Inicio del bloque de instrucciones *while*.
hubnet-fetch-message	Toma el primer mensaje pendiente de procesar en la cola.
ifelse hubnet-enter-message?	Si el mensaje es de conexión de un nuevo participante ejecuta el primer bloque [] de instrucciones siguientes. Si no, ejecuta el segundo bloque [] de instrucciones.
[create-new-student]	Este bloque entra si el mensaje es de conexión. Llama al procedimiento *create-new-student* que definiremos después.
[	Este bloque entra si el mensaje no es de conexión.
ifelse hubnet-exit-message?	¿Es un mensaje de desconexión de participante? Sí -> primer bloque [] de instrucciones siguientes. No -> segundo bloque [] de instrucciones.
[remove-student]	Este bloque entra si el mensaje es de desconexión. Llama al procedimiento *remove-student* que definiremos después.
[ask students with [user-id = hubnet-message-source] [execute-command hubnet-message-tag]]	Pide al agente que envió el mensaje que ejecute el procedimiento execute-command, definido más adelante. El procedimiento execute-command recibirá el valor de la variable hubnet-message-tag, que indica el nombre del botón que ha pulsado o de la variable que ha modificado el participante, conforme al nombre que se haya dado a los controles al diseñar la pantalla de participante.
]	Fin de bloque de instrucciones para mensajes que no son de conexión.

]	Fin del bloque de instrucciones *while.*
end	Fin de la definición del procedimiento.

to create-new-student	Indica que a continuación se define un procedimiento denominado *create-new-student.*
create-students 1 [	Crea un agente del tipo *students* que se inicializa con el bloque de instrucciones siguiente.
set user-id hubnet-message-source	Hace que la variable *user-id* de este *student* tome el nombre del participante que envió el mensaje de conexión.
set label user-id	Hace que la variable *label* de este *student* tome el nombre del que envió el mensaje.
set step-size 1	Hace que la variable *step-size* de este *student* tome el valor 1.
send-info-to-clients	Llama al procedimiento *send-info-to-clients* que se definirá después para enviar cierta información a los participantes.
]	Fin del bloque de instrucciones.
end	Fin de la definición del procedimiento.

to remove-student	Indica que a continuación se define un procedimiento denominado *remove-student.*
ask students with [user-id = hubnet-message-source] [die]	Pide al agente que envió el mensaje (de desconexión) que abandone la simulación (comando *die*).
end	Fin de la definición del procedimiento.

4.6. Procedimientos de detalle a ejecutar por los agentes

to execute-command [command]	Indica que a continuación se define un procedimiento denominado *execute-command* que al ser llamado (por

	un *student*) recibe el valor de una variable. Este valor recibido se guarda internamente en la variable *command*. A este procedimiento se le llama desde *listen-clients* cuando un *student* pulsa un control o modifica el valor de una variable (el nombre del control o variable está almacenado en *hubnet-message-tag*, y es el valor que se envía en la llamada).
if command = "step-size" [set step-size hubnet-message stop]	Si el usuario modificó el control de selección "step-size", la instrucción *set step-size hubnet-message* hace que la variable *step-size* de este *student* tome el valor que seleccionó el participante, que está guardado en la variable *hubnet-message*. El comando stop hace que se abandone el procedimiento.
if command = "up" [execute-move 0 stop]	Si el usuario pulsó el control llamado "up" se llama al procedimiento *execute-move* enviándole el valor 0.
if command = "down" [execute-move 180 stop]	Si el usuario pulsó el control llamado "down" se llama al procedimiento *execute-move* enviándole el valor 180.
if command = "right" [execute-move 90 stop]	Si el usuario pulsó el control llamado "right" se llama al procedimiento *execute-move* enviándole el valor 90.
if command = "left" [execute-move 270 stop]	Si el usuario pulsó el control llamado "left" se llama al procedimiento *execute-move* enviándole el valor 270.
end	Fin de la definición del procedimiento.
to execute-move [new-heading]	Indica que a continuación se define un procedimiento denominado *execute-move* que al ser llamado por un *student* recibe el valor de una variable. Este valor se guarda internamente en la variable *new-heading*.
set heading new-heading	Hace que la variable *heading* (la variable que indica la orientación espacial de la representación gráfica de un agente en el área "view") del *student* que está corriendo este comando tome el valor recibido (el guardado en *new-heading*). Este valor indica "hacia dónde mira" el agente en el área gráfica "view".
fd step-size	Hace que la representación gráfica del *student* que está corriendo este comando se desplace hacia adelante un número *step-size* de pasos en el área gráfica "view".
send-info-to-clients	Llama al procedimiento *send-info-to-clients,* definido a continuación, que enviará al *student* que corre estas instrucciones el valor de las nuevas coordenadas a las que se ha desplazado su representación gráfica en el área

	"view".
end	Fin de la definición del procedimiento.
to send-info-to-clients	Indica que a continuación se define un procedimiento denominado *send-info-to-clients.*
hubnet-send user-id "location" (word "(" pxcor "," pycor ")")	Esta instrucción envía (*hubnet-send*) al cuadro denominado "*location*" de la pantalla de usuario del *student* que corre esta instrucción (*user-id*) el valor de las coordenadas en que se encuentra situada su representación dentro del área grafica "*view*".
end	Fin de la definición del procedimiento.

5. Derechos de uso y reproducción

El programa y el material de Netlogo pueden usarse y modificarse libremente con fines educativos o de investigación no comerciales siempre que se haga referencia a la nota de copyright de Netlogo que se reproduce a continuación y al nombre del autor original del material, Uri Wilensky:

Referencias

Izquierdo, L.R. (2007). NetLogo 4.0 Quick Guide. http://luis.izqui.org/resources/NetLogo-4-0-QuickGuide.pdf

Pascual, J.A., Galán, J.M., Izquierdo, L.R., Santos, J.I., Izquierdo, S.S., González, J. (2009). Una herramienta didáctica para la enseñanza de la teoría de juegos mediante internet. EDUTEC, Revista Electrónica de Tecnología Educativa, 29.

Poza, D. (2009). Manual de NetLogo en español. http://sites.google.com/site/manualnetlogo/

Wilensky, U. (1999). NetLogo. http://ccl.northwestern.edu/netlogo/. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL.

Wilensky, U.; Stroup, W. (1999). HubNet. http://ccl.northwestern.edu/netlogo/hubnet.html. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL.