Entorno de juegos para el desarrollo de prácticas de Inteligencia Artificial

Juan A. Suárez Romero, Amparo Alonso Betanzos, Marcos Villares Souto
Dpto. de Computación
Facultad de Informática, Universidad de A Coruña
15071 A Coruña
ja@udc.es, ciamparo@udc.es

Resumen

En este artículo se presenta una herramienta que permite implantar prácticas de aplicación de técnicas de Inteligencia Artificial utilizando un entorno de juegos. Se describirán las principales herramientas ya existentes para este objetivo, analizando sus ventajas e inconvenientes, y se describirá la herramienta aquí presentada y sus principales características. Por último se mostrarán algunos ejemplos de su uso para la definición de una práctica para la asignatura de Inteligencia Artificial.

1. Introducción

La asignatura de Inteligencia Artificial, en la Universidad de A Coruña, tiene como objetivo la introducción de los conceptos básicos de la Inteligencia Artificial, la descripción de los algoritmos de búsqueda, la representación del conocimiento y los diferentes modelos de razonamiento más comúnmente empleados. Dicha asignatura es anual, troncal y se imparte en el segundo ciclo de la Ingeniería Informática, y su importancia se puede ver debido al elevado número de créditos que dispone, 10'5, de los cuales 3 son de tipo práctico.

2. Motivación

A la hora de buscar un problema sobre el cual hacer un planteamiento de la práctica, se deben de tener en cuenta varios factores. El primero, y sin duda el más importante, debe hacer que el alumno adquiera los conocimientos buscados a nivel práctico, comprendiendo las ventajas e inconvenientes del aspecto que deseamos que aprenda.

Además de este aspecto, el problema debería ser tal que permita plantear diferentes enunciados que obliguen a poner en práctica distintos conocimientos adquiridos en la parte teórica. Ello es debido a que generalmente, y por razones de tiempo, en la parte teórica se suelen enseñar diferentes técnicas pero no se suele explicar cómo combinarlas. Un ejemplo en el caso de la asignatura de Inteligencia Artificial sería la explicación de los modelos de representación del conocimiento y los modelos de razonamiento, que se pueden explicar por separado. Debería ser objetivo de la práctica, pues, permitir que el alumno pueda aprender a combinar diferentes aspectos, o pueda comparar diversos algoritmos o técnicas para lograr el mismo objetivo.

El que el problema, y por tanto el enunciado de la práctica, resulte atractivo al alumno es otro aspecto que sin duda influye enormemente en el alumno. Una práctica atractiva hace que el alumno invierta más esfuerzo en su realización, muchas veces de forma involuntaria e inconsciente. Si además dicha práctica modela un problema del mundo real, esto incrementa el interés del alumno, puesto que comprende y es capaz de ver las aplicaciones prácticas del conocimiento adquirido.

Por último, el que la práctica sea de tipo competitivo resulta un aliciente para los alumnos, ya que la competición, bien individual o bien por equipos, incrementa el interés de los alumnos por conseguir la mejor solución posible al problema.

Un buen ejemplo de un tipo de problema que permite plantear prácticas con estas características son los juegos El planteamiento de un juego nos enfrenta a un problema concreto, y su solución exige en muchos casos la aplicación de diversas técnicas de Inteligencia Artificial.

Por ello, en este artículo se presenta un entorno de juegos desarrollado en la Facultad de Informática de A Coruña, a través de un Proyecto Fin de Carrera, con el que poder plantear prácticas de Inteligencia Artificial.

Este artículo queda estructurado como sigue. En la sección 3 se describirán las distintas herramientas ya existentes para el mismo objetivo. En la sección 4 se hará una descripción de la herramienta aquí presentada. Posteriormente en la sección 5 se mostrará un ejemplo de uso de la aplicación propuesta para definir una práctica. Y por último, en la sección 6 se expondrán las conclusiones y futuras líneas de trabajo.

3. Sistemas existentes

Actualmente existen diversos entornos de juegos que sirven como plataforma para definir prácticas de Inteligencia Artificial. En general estos juegos consisten en un entorno virtual donde conviven diferentes robots o *bots*. Cada uno de estos *bots* puede moverse por el escenario, y está dotado de una serie de capacidades, tales como un radar para detectar los objetos u otros *bots* que se encuentran en el escenario, y un arma con la cual poder disparar. El objetivo es destruir al resto de los *bots* del escenario. Para ello cada *bot* comienza con una cantidad de energía que va perdiendo cada vez que es alcanzado por un disparo de otro *bot*.

Cada uno de estos *bots* es controlado por un programa, que dicta las órdenes a ser ejecutadas por el *bot*. El objetivo de los alumnos en las prácticas sería implementar estos programas empleando las técnicas y conocimientos de Inteligencia Artificial que queremos que ejerciten.

A continuación expondremos una breve descripción de entornos más conocidos y empleados.

3.1. Real Time Battle

Se trata de un juego de lucha entre varios *bots*, dotados cada uno de un radar y un cañón [4]. El objetivo es eliminar a todos los demás *bots*. Se puede emplear prácticamente cualquier lenguaje de programación para implementar los *bots*, ya que la comunicación se realiza a través de un sencillo protocolo. Para dotar al juego de mayor realismo, éste transcurre en tiempo real y los *bots* se comportan como objetos físicos reales: se pueden mover en cualquier dirección y chocar contra los elementos del escenario. Además, se pueden distribuir por el escenario objetos que o bien incrementen la energía (bonus) o bien la disminuyan (minas). Por último, también permite diseñar diversos escenarios de batalla para poder ver el rendimiento de los *bots* en distintas situaciones.

3.2. Codewars

Se trata de un entorno con una filosofía similar al *Real Time Battle*. Aquí también el objetivo es eliminar al resto de los *bots* presentes en el escenario [5]. Las principales diferencias con respecto al anterior son dos. Por un lado, el sistema funciona en red, esto es, podemos disponer el escenario en una máquina servidora y los *bots* se conectan a ella desde diferentes máquinas de la red. Por otro lado, aquí ya no se simula un mundo físicamente real, sino que más bien el mundo es una matriz en la cual se disponen los *bots*.

3.3. Robot Battle

Al igual que en los anteriores, el objetivo en este juego es programar los *bots* para que eliminen al resto de los contrincantes [6]. Sin embargo, aquí los *bots* se pueden organizar en equipos. De esta forma podemos desarrollar e implementar estrategias distribuidas. La programación de los *bots* se hace a través de un lenguaje propio, de manera que se asegura que la eficiencia de los *bots* dependa exclusivamente de la estrategia implementada y no de la eficiencia del código ejecutable generado por el compilador correspondiente al lenguaje empleado. Por último, hay que mencionar que esta herramienta no permite la ejecución en red de los *bots* ni definir distintos escenarios de juego: sólo jugaremos en un mundo rectangular sin ningún tipo de obstáculo.

3.4. RoboWars

Se trata de otro entorno en el que los *bots* se disponen en equipos cuyo objetivo es eliminar al resto de los equipos [7]. Las características de este entorno son una mezcla de los vistos anteriormente: la acción transcurre en tiempo real, no se permite distribuir los *bots* en una red de ordenadores, no se simula un mundo físico real y tampoco se permite definir los escenarios de lucha.

3.5. Robocode

Se trata de una herramienta muy popular desarrollada por IBM [5]. Sus características son

	Real Time Battle	Codewars	Robot Battle	RoboWars	Robocode	Corewars	GNU Robots
Tiempo Real	Sí	Sí	Sí	Sí	Sí	No	No
En red	No	Sí	No	No	No	No	No
Lenguaje	Cualquiera	Cualquiera	Propio	C / C++	Java	Corewars / Redcode	Guile
Mundo físico	Sí	No	Sí	No	Sí	No	No
Escenario definible	Sí	Sí	No	No	No	No	Sí
Juego en equipo	No	No	Sí	Sí	No	No	No
Gráficos	2D	Consola / Frontends	2D	2D	2D	2D	Consola / Frontends
Objetivos	Luchar	Luchar	Luchar	Luchar	Luchar	Luchar	Recolectar

Tabla 1. Comparativa de las herramientas ya existentes

muy similares al entorno *Robot Battle*, solo que en esta ocasión los *bots* son programados en Java y no se permite la agrupación por equipos de los *bots*.

3.6. GNU Robots

Aunque de filosofía similar a los juegos anteriores, en este caso el objetivo no es eliminar a los contrarios, sino recolectar el mayor número posible de premios distribuidos por el escenario antes de que se agote la energía del *bot* [3]. En este caso el *bot* se programa en el lenguaje Guile y, a diferencia de los otros entornos, aquí el juego transcurre por turnos y no en tiempo real.

3.7. Corewars

Aunque se trata de un entorno bastante limitado, lo mencionamos por tratarse de uno de los primeros juegos de este tipo en publicarse [1]. En este caso el juego consiste en eliminar también a los programas contrarios. Aquí los *bots* se pueden desarrollar en dos lenguajes distintos: Corewars y Redcore. Ambos son similares a un ensamblador, aunque el último es más potente que el primero, pero también más complicado de emplear.

3.8. Comparativa

En la tabla 1 se muestra una comparativa de los siete sistemas presentados. Un pequeño análisis de esta comparativa nos permite extraer las ventajas e inconvenientes de cada uno de ellos para su uso en el planteamiento de prácticas de Inteligencia Artificial. Uno de los aspectos a tener en cuenta es, por ejemplo, el lenguaje de programación a emplear para programar los *bots*. Como podemos observar, algunos sistemas permiten emplear cualquier lenguaje de programación, mientras que otros nos restringen a uno concreto. El poder emplear cualquier lenguaje permite que el alumno se centre exclusivamente en la técnica de Inteligencia Artificial a implementar en la práctica, sin que tenga que aprender a usar un lenguaje concreto, lo cual claramente no es el objeto de las prácticas. Como desventaja, la elección del lenguaje influye mucho en el rendimiento del *bot* implementado, sobre todo cuando se trata de un juego donde la acción transcurre en tiempo real y no por turnos. Así, por ejemplo, un lenguaje de cómo C permitirá obtener un código más eficiente que otro, por ejemplo, interpretado. Además, un lenguaje de más bajo nivel hará que la implementación de la estrategia a seguir por el *bot* sea más compleja que empleando un lenguaje de más alto nivel. La comparación de las estrategias implementadas en los *bots*, por tanto, estará influenciada por el lenguaje empleado y no por la estrategia en sí. Esto se puede minimizar si restringimos al alumno la elección del lenguaje, o si el sistema permite el juego por turnos, donde el tiempo empleado para la selección de la siguiente acción a emplear por el *bot* no influye en el discurrir de la partida.

Otra de las características es que, excepto *Codewars*, los demás entornos exigen su ejecución dentro de una misma máquina. En la Facultad de Informática de la Universidad de A

Coruña los equipos de las aulas de prácticas están conectados en una red local; cada alumno posee una cuenta a la que puede acceder desde cualquier puesto. Sería mucho más cómodo e interesante que el juego se pudiese ejecutar en red, de manera que una máquina actuase de servidor y que cada alumno desde su propio puesto pudiese conectarse a dicho servidor con su *bot*. Esto le permitiría también ir modificándolo para mejorar su eficiencia.

Tampoco la mayoría de los entornos permite trabajar en equipo. Esto limita la posibilidad de plantear prácticas cuyo objetivo sea el uso de técnicas distribuidas de resolución de problemas.

Por último, y quizá la característica más importante de cara a definir prácticas de Inteligencia Artificial, sea el objetivo a conseguir en el juego. Como podemos observar, en la mayoría de ellos el objetivo es simplemente luchar y sobrevivir: el último *bot* o equipo de *bots* que sobreviva gana. Esta limitación de los objetivos plausibles restringe claramente el tipo de planteamiento de los problemas a proponer. Por ejemplo, un problema típico que no se podría plantear con estas herramientas sería la resolución de un laberinto, que se suele usar para como ejemplo de uso de diferentes métodos de búsqueda.

4. Sistema propuesto

Para resolver los inconvenientes de las herramientas existentes hemos desarrollado, a través de un proyecto fin de carrera, un entorno de juegos que toma las características más adecuadas de los sistemas planteados y propone otras nuevas cuyo objetivo es obtener un sistema flexible para la puesta en marcha de prácticas de Inteligencia Artificial [9]. Al igual que en los anteriores, este entorno también se configura como un escenario en el que conviven uno o más *bots* que interactúan entre sí y con el escenario. Este entorno consiste en un escenario en el que uno o varios *bots* tienen que resolver un problema. Las principales características de esta herramienta son:

- Se pueden definir distintos objetivos o combinaciones de objetivos a resolver. Así, un objetivo puede ir desde eliminar al resto de los *bots* hasta alcanzar una zona del escenario, o incluso que haya transcurrido un tiempo determinado.
- Emplea una arquitectura cliente/servidor, donde el escenario o partida se encuentra en el servidor y los alumnos se pueden conectar desde sus puestos a dicho servidor para ejecutar sus *bots*.
- Se pueden añadir diversos tipos de elementos al escenario: paredes, tejados, arcos, banderas, pastillas de energía, minas, armas, etc.
- Los *bots* pueden ser programados en cualquier lenguaje. La comunicación entre los clientes y el servidor se hace a través de un sencillo protocolo de comunicación.
- Las características de los *bots* son configurables. Así se puede indicar el tamaño de los mismos, la energía de la que disponen, si pueden disparar y con qué potencia, si pueden usar el radar para detectar los elementos del escenario y la precisión de los datos obtenidos, la posición de los *bots* al inicio, etc.
- Permite definir si la acción transcurrirá en tiempo real o por turnos. Para este último caso también se puede limitar el tiempo que durará cada turno.
- El entorno visual del juego es en tres dimensiones, para darle mayor atractivo a la herramienta. Además permite diseñar los escenarios con diferentes geometrías y añadirle diversos elementos decorativos.
- Todos los parámetros del juego (escenario, *bots*, objetivos, etc.) se hace a través de un simple fichero de configuración. Además, algunas de estas opciones pueden ser modificadas a medida que transcurre el juego. Por ejemplo, podemos activar o desactivar los radares dinámicamente para ver el efecto sobre los *bots*.
- La herramienta puede ser extendida para añadir nuevos elementos con los que interactuar, o definir nuevos tipos de objetivos a cumplir.

En la figura 1 se muestran los componentes que forman la arquitectura del sistema. Como podemos observar, se sigue un modelo cliente/servidor. El sistema es el propio servidor central, mientras que los *bots* son los programas cliente que se conectan al servidor a través de la red. Así, un *bot* puede ejecutarse en la misma máquina que en el servidor, en otro ordenador de la red local o incluso desde cualquier ordenador a

través de Internet. El alumno incluso podría desarrollar su *bot* desde casa y conectarse al servidor para probarlo, siempre y cuando éste estuviese disponible a través de Internet.

Durante la ejecución cada cliente se comunica con el servidor mediante el intercambio de mensajes de texto siguiendo un sencillo protocolo, de manera que estos mensajes son la única información que tiene recibe el cliente sobre lo que está sucediendo en la partida. El servidor recibe las órdenes emitidas, las evalúa y, si es posible ejecutarlas, las ejecuta y envía el resultado de la acción a los clientes. Además muestra la acción resultante en una interfaz gráfica en la que se muestra el transcurrir de la partida.

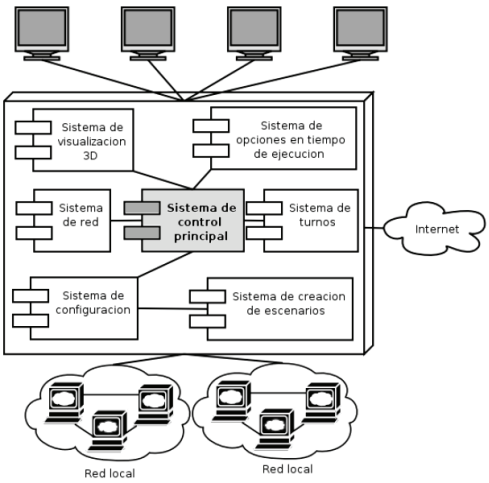

Figura 1. Arquitectura del sistema

El sistema principal está formado por un sistema de control y varios subsistemas de apoyo. En primer lugar tenemos el subsistema de configuración. Este se encarga de leer un fichero de configuración en el cual se definen los parámetros de la partida y los objetivos a cumplir, y una vez obtenidos delega en otro subsistema que se encarga de crear el escenario y colocar los diversos elementos que lo forman. Aquellas opciones que el profesor puede ir cambiando a medida que transcurre el juego están encargadas a otro módulo correspondiente. El encargado de la visualización de la partida es otro subsistema que emplea el estándar *OpenGL* para representar gráficamente el escenario y toda la acción que ocurre, todo de forma tridimensional. Por último, tenemos los subsistemas encargados de manejar los turnos en las partidas y el subsistema que recibe las peticiones de los clientes y devuelve los resultados. En la figura 2 podemos ver un ejemplo de una partida en la que interactúan varios *bots*. Los conos dibujados simbolizan el rango que cubre el radar de cada *bot*.

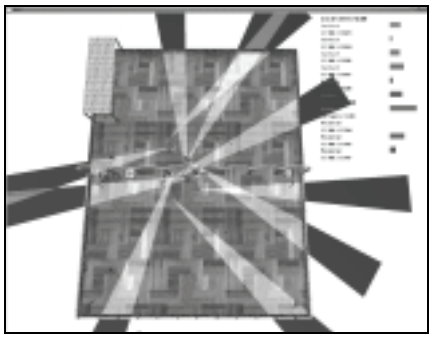

Figura 2. Ejemplo de una partida

5. Uso de la herramienta

Para ilustrar el funcionamiento de la herramienta y su uso para la puesta en marcha de prácticas de Inteligencia Artificial, vamos a describir los pasos generales a seguir.

En primer lugar se debe definir el problema a resolver. Aunque la herramienta se centra en los juegos, podemos definir diferentes problemas: eliminar al resto de los contrincantes, trabajar en equipo, buscar un elemento, etc. El problema deberá ser tal que la búsqueda de su solución nos permita emplear la técnica que queremos que el alumno utilice. Una vez que hemos definido conceptualmente el problema, es necesario plasmarlo en la herramienta. Para ello tendremos que definir, mediante el fichero de configuración correspondiente, el escenario en el cual transcurrirá la acción y el objetivo a conseguir. Además deberemos también definir las características de los *bots* para que se adecúen al problema y si la partida va a transcurrir en tiempo real o por turnos.

Una vez que hemos definido el juego, le tocaría el turno a los alumnos, que deberían implementar los *bots* que, empleando la técnica que se les pide, actúen dentro del escenario planteado para lograr alcanzar el objetivo

especificado. Aunque en principio cualquier lenguaje de programación valdría para implementar los *bots*, podemos restringir la elección a uno o varios lenguajes. Lo único necesario es que éste se conecte al servidor en el cual reside el escenario y le envíe las órdenes de acción para que dicho servidor las ejecute y muestre los resultados.

5.1. Ejemplo práctico

Para ilustrar los pasos descritos anteriormente, vamos a exponer un ejemplo de planteamiento de una práctica.

El objetivo es la puesta en práctica por parte de los alumnos de diversos algoritmos de búsqueda, tales como la búsqueda en anchura, la búsqueda en profundidad o la búsqueda A* [8]. Los algoritmos de búsqueda suelen ser los primeros temas que se explican en una asignatura de Inteligencia Artificial, y sobre este aspecto versan principalmente las prácticas del primer cuatrimestre en la Facultad de Informática de la Universidad de A Coruña.

Como hemos mencionado anteriormente, el primer paso consiste en definir el problema. Dentro del contexto de los juegos, debemos buscar un problema que exija para su resolución el uso de las técnicas de búsqueda que deseamos que se empleen. Aunque se pueden plantear diversos problemas, uno de los más típicos es la resolución de un laberinto. Por tanto, nuestro problema a resolver será que los alumnos implementen un *bot* que pueda llegar de un punto del escenario a otro a través de un laberinto.

Una vez que hemos definido el problema, debemos definir el escenario en un fichero de configuración. Aquí iremos indicando en qué posiciones se encuentran los bloques o paredes, para ir delimitando el laberinto. También podremos añadir si queremos elementos decorativos, tales como por ejemplo arcos, tejados o una bandera que nos muestre el destino. Posteriormente especificaremos el objetivo, que es llegar a un punto determinado del tablero. Si además queremos limitar el tiempo de búsqueda, para evitar que un *bot* entre en un bucle infinito, podemos añadir un segundo objetivo, consistente en que haya transcurrido un determinado tiempo. De esta manera, el juego se acabará cuando o bien el *bot* llegue al punto indicado, o bien transcurra el tiempo especificado. Por último, sólo queda indicar el punto de partida del *bot*.

Un ejemplo de este escenario lo tenemos en la figura 3.

Aparte de definir el escenario en el que transcurrirá el juego y los objetivos a alcanzar, también es necesario describir las características de los *bots*. En este ejemplo nuestro objetivo con la práctica planteada es que los alumnos aprendan y comparen dos técnicas de búsqueda. Por lo tanto vamos a proponer que los alumnos implementen dos *bots*, uno que utilice una búsqueda en profundidad y otro que emplee una búsqueda A*. Como se trata de un problema de búsqueda en el que en cada partida participa un único *bot*, desactivaremos las armas, ya que no tiene sentido que los *bots* disparen. Además, también vamos a desactivar el radar del *bot*. De esta manera la única forma de ir descubriendo el escenario es mediante "choques" contra las paredes. Al final de la partida el *bot* habrá calculado un camino solución, que podrá emplear posteriormente en otras partidas (siempre y cuando no cambiemos el laberinto).

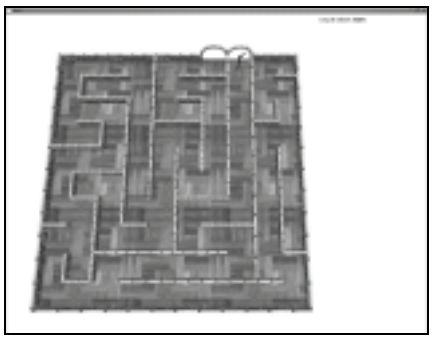

Figura 3. Imagen del laberinto definido

Una vez definido el problema y sus características, sería el turno de los alumnos, que deben implementar los dos *bots* utilizando las dos técnicas, para poder comparar los resultados. En este punto debemos mencionar que el sistema propuesto no ofrece, en su versión actual, ningún esqueleto de un *bot* que los alumnos puedan utilizar: el alumno tiene que implementar desde cero los *bots*. Esto supone un trabajo extra que, en el caso de disponer de un tiempo muy limitado para la realización de la práctica, puede llegar a ser un grave inconveniente. Lo ideal sería que el

alumno pudiese partir de un *bot* que ya implemente todas las acciones, menos la que queremos que el alumno practique. De esta forma el alumno se centraría exclusivamente en dicho aspecto, sin que se distraiga en otros aspectos meramente de programación, que en nuestro caso no sería el objetivo de la práctica.

La ventaja de tener una arquitectura cliente/servidor radica en que, como hemos mencionado anteriormente, podemos disponer el servidor en una máquina y los alumnos pueden conectarse a ella con sus *bots* para ir probando el juego. Así ellos mismos irán comprobando sus progresos y, si lo desean, podrán comparar sus *bots* con los de los otros compañeros. Esto puede llegar a fomentar un cierto nivel de competitividad que resulta ser un estímulo interesante para la realización de la práctica.

Finalmente, una vez acabada la práctica el profesor podría evaluar los *bots* bien desde una misma máquina, o en presencia de los alumnos desde la propia red del aula de prácticas.

6. Conclusiones y trabajo futuro

En este artículo se ha presentado una herramienta de entorno de juegos concebida principalmente para el desarrollo de prácticas de Inteligencia Artificial, y que trata de resolver los problemas de otras herramientas ya existentes. Su principal característica frente a otros sistemas similares es su capacidad de definir diferentes objetivos a cumplir. Además, su arquitectura cliente/servidor, junto con un sencillo protocolo de comunicación, permite implantar este sistema en una red local en la cual se estén impartiendo las prácticas. Por último, la herramienta es fácilmente extensible, pudiendo añadir nuevos tipos de elementos con los que interactuar y nuevos tipos de objetivos que se puedan cumplir.

No obstante, se trata de una primera versión del programa. Aunque todavía no hemos hecho ninguna experimentación a nivel práctico con alumnos utilizando esta herramienta en la asignatura, hemos identificado una serie de aspectos en los que convendría trabajar con el objetivo de potenciar esta herramienta:

- Desarrollo de una herramienta de configuración visual de los escenarios. Actualmente el profesor debe crear un fichero de texto en el que se describa el escenario, los componentes que lo forman, los objetivos a alcanzar y las características de los *bots*. Sería mucho más sencillo disponer de una herramienta gráfica que permita dibujar el escenario e ir situando los elementos de una manera visual e intuitiva.
- Crear un catálogo de esqueletos de *bots*. Tal y como hemos mencionado en el apartado anterior, en estos momentos el alumno tiene que implementar desde cero todo el *bot*. Esto acarrea un gran esfuerzo para el alumno, que puede perder mucho tiempo en detalles que nada tienen que ver con el aspecto de la Inteligencia Artificial que queremos que practiquen. Por ello disponer de una serie de esqueletos de *bots*, en los que el *bot* esté casi completo, excepto aquellas partes relacionadas con la técnica a implementar que sea la que deba completar el alumno, nos permitiría que el alumno se centrase exclusivamente en este aspecto y no en el resto del *bot*.
- Permitir una construcción modular de los *bots*. Actualmente los *bots* están formados por un radar y un cañón, que podemos activar o no según las características del problema. En el futuro pretendemos que los *bots* se construyan de forma modular, pudiendo definir los distintos componentes que lo forman, como por ejemplo un cañón, un radar, o un escudo. Esto permitiría incluso disponer de varias configuraciones distintas para cada componente. Por ejemplo, podríamos tener un cañón que sea capaz de destruir un *bot*, o uno que sea capaz de eliminar cualquier elemento del escenario.
- Permitir que los clientes puedan ver el transcurso de la partida gráficamente. En la versión actual la interfaz gráfica en la que se visualiza la acción se encuentra exclusivamente en el servidor. Esto hace que los alumnos que se conecten con sus *bots* al servidor no vean gráficamente la partida, a menos que se sitúen en el servidor. Ello resulta un inconveniente, sobre todo si los alumnos no pueden acceder físicamente al servidor. Por tanto nuestra intención es desacoplar la interfaz gráfica de acción del servidor, de manera que ésta pueda ser empleada desde los clientes.

Referencias

[1] Codewars. *http://codewars.sourceforge.net*.
[2] Corewars. *http://corewars.sourceforge.net*.
[3] GNU Robots. *http://www.gnu.org/software/robots*.
[4] Real Time Battle. *http://realtimebattle.sourceforge.net*.
[5] Robocode. *http://robocode.sourceforge.net*.
[6] Robot Battle. *http://www.robotbattle.com*.
[7] RoboWars. *http://www.deepnettech.com/-old/robowars.html*.
[8] Russell, S.J.; Norvig, P. *Inteligencia Artificial. Un Enfoque Moderno*. Pearson Prentice Hall, Segunda Edición, 2004.
[9] Villares Souto, M. *Entorno de Juegos para la Aplicación de Técnicas de I.A.* Proyecto Fin de Carrera, Facultad de Informática de la Universidad de A Coruña, 2004.