Tecnología de Videojuegos
Práctica 7: Desarrollo de la IA de un videojuego

UAH, Departamento de Automática, ATC-SOL
http://atc1.aut.uah.es

Objetivos:

- Implementar la lógica de un bot en un videojuego completo
- Desarrollo del pensamiento algorítmico
- Utilización de heurísticas sencillas
- Familiarización con la lectura de código ajeno
- Manejar documentación en formato Javadoc

Comentario inicial

El objetivo de la presente práctica es el desarrollo de la inteligencia de un bot dentro de un videojuego completo. Para ello se utilizará *Robocode*[1], un videojuego pensado como plataforma de competición de IA que ofrece un modelo de programación extremadamente sencillo. Esto nos va a permitir evadir dificultades de la programación para poder dedicar todo el esfuerzo al desarrollo de la inteligencia del bot.

Robocode es un juego de combate entre tanques (o robots, si usamos los mismos términos que Robocode), en el varios tanques combaten entre sí. Cada partida está dividida en una serie de rondas independientes en el que combaten un número variable de tanques. El ganador es quien acumule más puntos al final de dichas rondas. La figura 1 representa una captura de pantalla del juego.

La práctica consiste en desarrollar el controlador del tanque, de manera que el tanque pueda combatir eficazmente de manera autónoma sin ningún tipo de intervención humana.

Descripción del juego

Las partidas de Robocode transcurren dentro de un escenario rectangular sin obstáculos delimitado por paredes en el que se sitúan todos los tanques. Cada tanque dispone de una cantidad de energía que disminuye cuando recibe un disparo, choca contra otro tanque o con la pared. La acción de disparar también consume energía, dependiente de la potencia del mismo. La única manera de recuperar energía es acertando a un tanque adversario; se ganará más energía cuanto más potente haya sido el disparo. Cuando la energía es consumida, el tanque queda inutilizado, sin poderse mover ni disparar. Si la energía desciende de cero, el tanque es destruido. Al disparar el cañón se recalienta, y alcanzada una cierta temperatura no puede seguir disparando; la temperatura baja si se deja enfriar el cañón no disparando.

[1] http://robocode.sourceforge.net/

Figura 1: Captura de pantalla de una partida de Robocode.

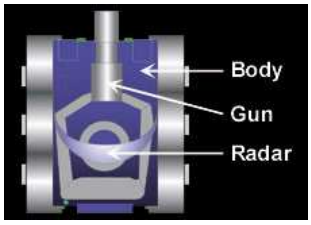

Figura 2: Componentes de un tanque en Robocode.

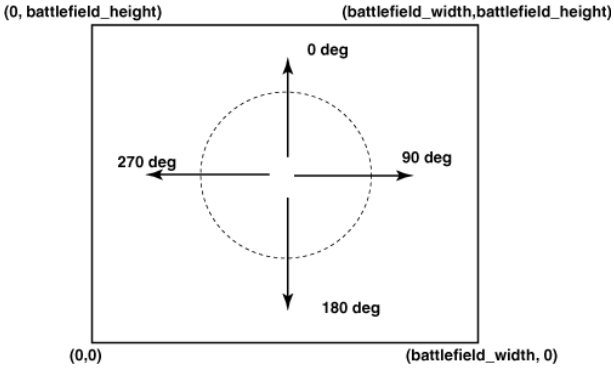

Figura 3: Sistema de coordenadas utilizadas en el juego.

Los tanques están divididos en tres componentes que se pueden mover independientemente: Cuerpo, torreta y rádar. El cuerpo puede girar a derecha e izquierda, y moverse adelante o atrás. La torreta contiene el cañón, puede girar y girar a izquierda o derecha. El rádar permite localizar a tanques adversarios, y al igual que la torreta, puede girar a izquierda y derecha. Los compontentes del tanque están ilustrados en la figura 2.

El sistema de coordenadas, tal y como se puede ver en la figura 3, es cartesiano y tiene su origen en la esquina inferior izquierda. La unidad de medida utilizada es el pixel, si bien se guardan dentro de variables de tipo `double`, por lo que en la práctica se pueden especificar unidades fraccionarias de pixel. Para especificar giros y posiciones de elementos giratorios (cuerpo, torreta o rádar) se puede usar grados, utilizando como referencia el norte (0 grados). Algunos métodos admiten como argumento el número de grados relativos al elemento móvil, indicando, por ejemplo, que gire 30^{o} a la izquierda, en vez de indicarle que gire a la posición de 30^{o} (respecto a la parte superior de la pantalla).

Modelo de programación

La programación de un tanque es muy simple. Para programar un robot de Robocode es necesario crear una nueva clase que herede de la clase `Robot`[2]. El robot así creado tiene, sin necesidad de codificar, una gran cantidad de métodos disponibles, y no hace falta preocuparse por cuestiones como la visualización del robot o la gestión de eventos.

El método principal que hay que sobrecargar es `run()`[3], que es llamado cuando el robot es instaciado y es donde hay que introducir la lógica del tanque. El motor del juego se encarga de paralizar la ejecución del código del robot cuando sea necesario y realizar otras tareas, por lo que el programador del robot simplemente puede asumir que es un método constantemente en ejecución, con alguna limitación de tiempo.

Los métodos principales que tiene un robot son los siguientes:

- `public void fire(double power)`

[2]Existe un modelo de programación avanzado, centrada en la clase `AdvancedRobot`, que es más compleja de programar, pero también ofrece más posibilidades

[3]Siendo más técnicos, cada tanque es en realidad un hilo; el método `run()` es la forma en Java de crear hilos. No obstante, la programación de hilos no está contemplada en la asignatura y no es necesario su conocimiento para realizar esta práctica

- `void turnGunLeft(double degrees)`
- `void turnGunRight(double degrees)`
- `void turnLeft(double degrees)`
- `void turnRadarLeft(double degrees)`
- `void turnRadarRight(double degrees)`
- `void turnRight(double degrees)`
- `void ahead(double distance)`
- `void back(double distance)`

Un elemento importante a tener en cuenta es la utilización de eventos. Cuando sucede algo relevante para el robot que no sea predecible, como el impacto de una bala, o la colisión con un muro, se genera un evento. En términos de programación, ese evento se traduce en que se invoca la ejecución de un método, independientemente del código que el robot estuviera ejecutando en ese momento, todo de forma transparente al programador.

Los métodos asociados a eventos más importantes son los siguientes:

- `public void onScannedRobot(ScannedRobotEvent e)`
- `void onHitByBullet(HitByBulletEvent event)`
- `void onBulletMissed(BulletMissedEvent event)`
- `void onHitRobot(HitRobotEvent event)`
- `void onHitWall(HitWallEvent event)`

Se puede cambiar el comportamiento del tanque ante estos eventos simplemente sobrecargando el método en cuestión. Hay que destacar que el evento se aporta como parámetro al método, y contiene información sobre el mismo. El evento más importante es `onScannedRobot()`, que se activa cada vez que un robot enemigo es detectado por el rádar.

Robocode tiene integrado un editor de código fuente que permite editar y compilar los robots, por lo que el desarrollo se simplifica en gran medida. Si se desea, también se puede trabajar desde un IDE externo como NetBeans o Eclipse.

Aspectos a considerar

Hay una serie de reglas en el juego que, si se tienen en cuenta, introduce una complejidad mayor, pero también permite crear robots mucho mejores. Algunas de estas reglas están descritas a continuación:

- El choque con los muros u otros robots reduce la energía.
- La velocidad de las balas es finita.

- La velocidad de las balas depende de su potencia.
- La energía consumida en cada disparo depende de la potencia del mismo.
- Se puede conocer la dirección de movimiento de un tanque cuando se le detecta.
- La dinámica de los tanques está sujeta a una aceleración.
- Existe una tasa de giro máxima para el tanque, la torreta y el rádar.

Ejercicio 1: Manejo de Robocode

Cuando se comienza a programar en un nuevo entorno conviene, en primer lugar, realizar una serie de tareas sencillas que permitan familiarizarse con él. Para empezar, realice las siguientes tareas:

1. Descargar, instalar y ejecutar Robocode. Consultar la guía de instalación si es preciso.
2. Hay dos robots interactivos (`sample.Interactive` y `sample.Interactive_v2`), muy útiles para tareas de depuración. Jugar varias partidas con dichos robots y otros contenidos de ejemplo para comprobar sus diferencias y comprender las reglas del juego.
3. Comprender la lógica que siguen los distintos robots de ejemplo que vienen con Robocode.

Ejercicio 2: Manejo de la documentación de Robocode

El ejercicio profesional de la programación exige el aprendizaje constante de nuevas herramientas y bibliotecas. Además, cualquier aplicación suele ser tan extensa que siempre implica utilizar nuevas bibliotecas, por lo que resulta imprescindible aprender a aprender el uso de las mismas. Para ello es imprescindible aprender a manejar documentación, leer código fuente y manejar documentación en formato JavaDoc es imprescindible.

1. Localizar la documentación aportada por Robocode. ¿Qué documento da información elemental sobre cómo se programa un robot?
2. Buscar información relativa a la física del juego. ¿A qué velocidad se mueven los robots? ¿Y las balas? ¿Cuánto daño inflinge el impacto de una bala?
3. JavaDoc contiene todos los métodos disponibles en Robocode. Localice la página en la que se encuentra este documento.
4. En base al punto anterior, responda a las siguientes preguntas: ¿Qué método permite conocer la energía restante en el robot? ¿Qué métodos permiten conocer la posición del robot? ¿Cómo se puede conseguir que el movimiento del cuerpo del robot y la torreta sean independientes? ¿Cómo se puede conocer cuánta energía le queda a un robot adversario que ha sido detectado?

Ejercicio 3: Programación de Robocode

Una regla básica esencial en programación (e ingeniería en general) es partir de sistemas sencillos e ir añadiendo complejidad poco a poco, abordando un único problema cada vez.

1. Abrir el editor integrado en Robocode, crear un nuevo robot con él y ejecutar el robot por defecto que se crea.
2. Modificar el robot por defecto personalizando los colores del cuerpo, cañón y rádar.
3. Crear un robot que, permaneciendo quieto, vaya girando la torreta y dispare cuando se detecte un robot adversario.
4. Modificar el robot anterior para que constantemente se desplace en una única dirección hasta chocar con una pareded o robot, y entonces se desplace en el sentido inverso.
5. Localizar la ubicación de las clases Java con la implementación de robots de ejemplo. Abrir varios de ellos e intentar comprender el código. Compararlo con el código desarrollado anteriormente.

Ejercicio 4

Implementar un robot optimizado para realizar batallas contra entre dos y tres robots adversarios. No se puede utilizar ninguno de los algoritmos de ejemplo ni robots publicados en Internet.