Práctica 3: Programación Dinámica

1 Elección de torretas

En la batalla por preservar el conocimiento en la Universidad de los Sabios, los estrategas defensores se enfrentan a la crucial tarea de seleccionar las torretas más efectivas para defender los pasillos académicos de los invasores goblins. Con un presupuesto limitado, cada elección estratégica debe ser sabiamente considerada.

El desafío radica en optimizar la asignación de recursos financieros para adquirir torretas con habilidades y atributos específicos. Cada torreta tiene un coste asociado y aporta capacidades únicas, ya sea mediante un alcance excepcional, un poder de ataque formidable, u otras habilidades especiales, que pueden inclinar la balanza a favor de los defensores.

Los estrategas deben ponderar cuidadosamente sus opciones, evaluando el equilibrio entre la cantidad y calidad de las torretas. ¿Priorizarán la cantidad para cubrir una área más amplia o invertirán en torretas especializadas capaces de frenar a los goblins más resistentes?

La destreza táctica y la toma de decisiones calculadas serán esenciales para la defensa exitosa de la Universidad de los Sabios. ¡Que la sabiduría guíe la elección de torretas y asegure la preservación del conocimiento en esta batalla estratégica!

2 Desarrollo de la práctica

El juego sigue las mismas reglas que se describieron en el guión de la práctica anterior. Sin embargo, en esta práctica sí que se llevarán a cabo varias partidas completas. Como jugadores, nuestro objetivo es determinar qué torretas, de entre un conjunto de torretas disponibles, seleccionaremos para colocar en el terreno. Las deberemos seleccionar en función de su coste y de un presupuesto máximo disponible. Una vez seleccionadas, el juego se encargará de su colocación. Lo hará tratando de colocarlas cerca de las zonas de paso de los enemigos. En esta práctica **no podemos controlar la colocación de las torretas**.

Cada partida está formada por varias rondas. Cada ronda comienza con una lista de torretas y un presupuesto máximo. Deberán seleccionarse las defensas que se crean más oportunas para conseguir que escapen el menor número posible de enemigos. Evidentemente, el coste total de la torretas seleccionadas no podrá superar el presupuesto

total. Una vez seleccionadas y colocadas la torretas, comenzarán a aparecer enemigos, que tratarán de alcanzar las casillas de salida. La ronda finalizará cuando no queden enemigos en el terreno, bien porque han alcanzado alguna casilla de salida o porque han sido eliminados por las torretas. La puntuación de la ronda será igual al número de enemigos que han alcanzado alguna de las celdas de salida.

Tras finalizar una ronda la siguiente comenzará automáticamente. Cuando se acabe la última ronda la partida finalizará. La puntuación final de la partida será la suma de las puntuaciones obtenidas en cada ronda. El objetivo es, evidentemente, conseguir que escapen la menor cantidad posible de enemigos.

Para definir la estrategia de compra, debe modificarse el método `buyTowers` la clase `TowerBuyer`, situada dentro del paquete `net.agsh.towerdefense.strats`. El método recibe una lista de objetos de tipo `Tower` y el presupuesto disponible. El método debe devolver una lista con los índices de la torretas en la lista que se desean comprar. La lista `<2, 5>`, indicaría por ejemplo que se han seleccionado la segunda y la quinta torretas de la lista que se pasa por argumento al método para su compra.

Para compilar la práctica basta con ejecutar el archivo `compile.bat` ejecutar la siguiente orden en la consola (simpre desde el directorio raiz de la práctica):

```
javac -cp "towerdefense.jar" net\agsh\towerdefense\strats\*.java
```

Y para ejecutarlo basta con usar el comando `run.bat` o ejecutar la siguiente orden en la consola de comandos (siempre desde el directorio raíz de la práctica):

```
java -cp ".;towerdefense.jar" Main
```

Se imprimirán a continuación una serie de mensajes con información sobre la partida. Opcionalmente, es posible visualizar la partida de forma gráfica. Basta con añadir el parámetro `GUI=1` al final de la instrucción anterior o hacer uso del archivo `run-gui.bat`. En ambos casos se mostrará una ventana similar a la mostrada en la Fig. 1.

3 Ejercicios

Para completar la práctica se proponen una serie de tareas, que deberán llevar a cabo cada alumno individualmente. El resultado de estas tareas dará lugar al desarrollo de una aplicación en Java y de una pequeña memoria a incluir al comienzo del código fuente en forma de comentario. El resultado de la práctica deberá entregarse a través del campus virtual antes de la fecha límite, a través de la tarea creada al efecto y en el formato solicitado. No se corregirán productos enviados por otros medios o que se envíen en un formato distinto al solicitado.

El objetivo de la práctica consiste en desarrollar un algoritmo, **basado en el enfoque de Programación Dinámica**, que sirva para determinar qué torretas, de entre un conjunto de torretas disponibles, seleccionaremos para colocar en el terreno. El coste de las torretas seleccionadas no podrá ser superior al presupuesto máximo de compra.

En el material de prácticas podrá encontrar un código de ejemplo que selecciona por orden las torretas de la lista hasta quedarse sin presupuesto. Este código, aunque funcional, es muy poco eficaz. Deberá sustituir ese código de ejemplo por uno mejor.

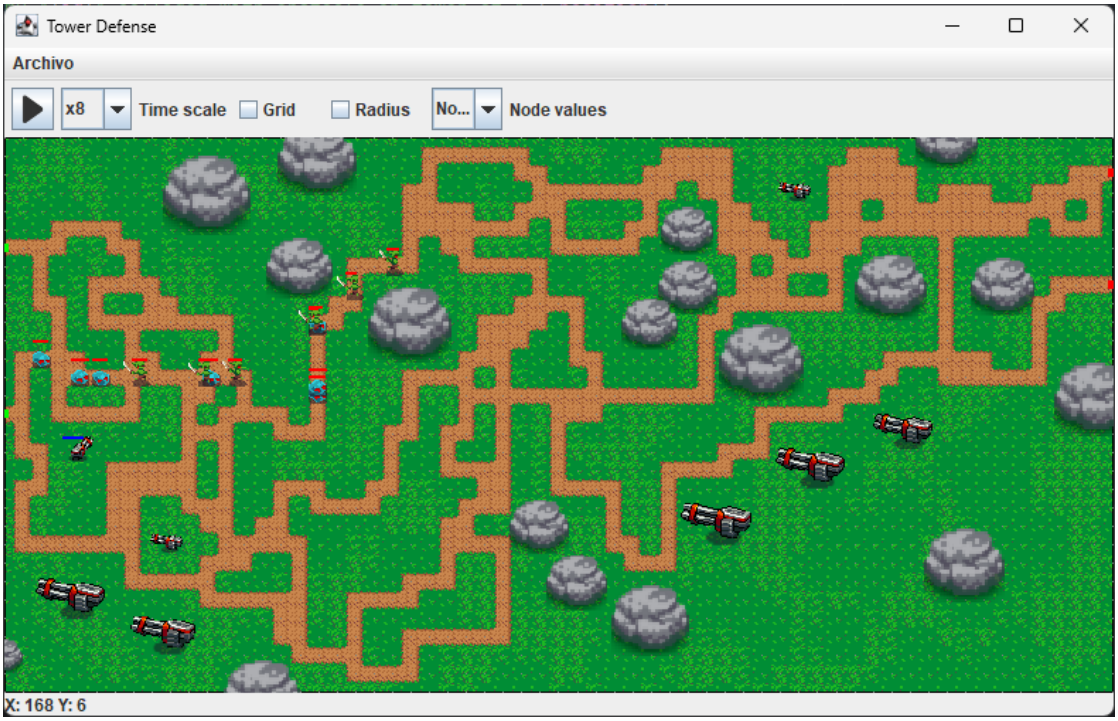

Figure 1: Interfaz gráfica del juego

Para ello, lo ideal es tener en cuenta las características de las torretas. A igualdad de coste, siempre será mejor seleccionar una torreta con un radio de alcance superior, por ejemplo. Concretamente, las tareas a llevar a cabo son:

1. Descargar el archivo comprimido del campus con los códigos de ayuda y estudiar su contenido.
2. Ejecutar el archivo "compile.bat" desde una consola de comandos para compilar el código de ejemplo.
3. Ejecutar el archivo "run.bat" para ejecutar el código de ejemplo.
4. Ejecutar el archivo "run-gui.bat" para mostrar la interfaz gráfica.
5. Crear un método de nombre `getTowerValue` que reciba por argumento una Torreta y devuelva un valor numérico con el valor estimado dicha torreta. Ese valor dependerá de las características de la torreta.
6. Añadir un comentario al comienzo del fichero `TowerBuyer.java` con la formula o procedimiento empleado para determinar el valor de una torreta.
7. Modificar el método `buyTowers` de la clase `TowerBuyer`, para que devuelva la lista de índices con las torretas seleccionadas. Para ello se empleará el método `getTowerValue`, creado previamente.

4 Entrega

Todo el código desarrollado en esta práctica deberá estar incluido en el fichero "TowerBuyer.java". Será éste el único archivo que se envíe a través de la tarea habilitada al efecto en el campus virtual. El fichero debe enviarse sin comprimir y listo para ser compilado. Si el fichero no compila no se evaluará la práctica.

5 Evaluación

Todos los programas enviados por los alumnos serán ejecutados en uno de los ordenadores del laboratorio y se anotará la puntuación obtenida por cada alumno. La calificación de la práctica se otorgará en función de la puntuación obtenida.