Práctica 0. Introducción.

21 de noviembre de 2017

1. Descripción general

En el año 2100 ya no quedan suficientes recursos en la Tierra y la humanidad se lanza a la colonización de otros planetas. Ya desde las primeras misiones descubrimos que no estamos solos en el universo y debemos luchar contra otras civilizaciones alienígenas para conseguir dichos recursos. Los *ucos* son una raza de extraterrestres que pueblan los planetas donde se encuentran los tan ansiados recursos y harán todo lo posible por evitar su extracción.

Hemos sido encomendados con la tarea de diseñar la estrategia defensiva de las bases que estableceremos en los planetas asaltados. Las bases serán atacadas por alienígenas y nuestro objetivo es mantenerlas a salvo el mayor tiempo posible para obtener así la mayor cantidad de recursos del planeta.

2. Reglas de juego

Cuando se descubre un planeta viable se le asocia un código numérico único y se procede al envío desde la tierra de un conjunto de edificios defensivos para desplegar en dicho planeta. Estas defensas se utilizarán para evitar que los *ucos* destruyan el centro de extracción de minerales, enviado también junto con ellas. Puesto que el enemigo nos supera ampliamente en número, nuestro objetivo consiste en extraer la mayor cantidad posible de minerales antes de que la base sea destruida. Se considerará que la base ha sido destruida cuando el centro de extracción de minerales sea destruido por los *ucos*.

Las batallas contra los *ucos* se desarrollan en terrenos de juego cuadrados de dimensiones $n \times n$. Dentro del campo de batalla podemos encontrar tres tipos de entidades u objetos: obstáculos, defensas y unidades de UCOs. Cada uno de estos tipos de entidades tiene sus propias características, si bien comparten algunas de ellas.

- Todos los objetos desplegados en el campo de batalla tienen asociado un código único.
- Todos los objetos tienen asociados un valor que indica el espacio ocupado por dicho objeto. Por simplicidad, supondremos una forma más o menos cilíndrica de los objetos y bastará, por tanto, con indicar el valor de su radio. En general y salvo que se indique lo contrario, ninguno de estos objetos podrá solaparse en el campo de batalla con ningún otro objeto.

  Todos los objetos tienen asociada una posición en el campo de batalla, que vendrá expresada por una coordenada tridimensional expresada como $< x, y, z >$. La esquina superior izquierda del terreno de juego se define como la posición $< 0, 0, 0 >$. La esquina inferior derecha coincide con la coordenada $< n, m, 0 >$. Por simplicidad supondremos un terreno de juego plano, por lo que la tercera coordenada (z) siempre tendrá un valor nulo [1].

2.1. Obstáculos

Los obstáculos se distribuyen de forma aleatoria a lo largo del campo de batalla y su ubicación no cambia durante el desarrollo de la misma. Su ubicación será siempre la misma para un determinado planeta. Los obstáculos pueden solaparse entre sí, formando conjuntos de obstáculos con formas irregulares. Ningún otro tipo de entidad puede solaparse con un obstáculo.

[1] En futuras versiones del juego podrán incluirse terrenos con desniveles.

2.2. Defensas

Para el asalto a cada planeta disponemos de un conjunto de defensas. Al inicio de la batalla estos elementos se colocarán en determinadas localizaciones en el planeta, y esta ubicación no podrá variar a lo largo de la batalla. La ubicación de estas defensas se considerará válida siempre y cuando se encuentren por completo dentro de los límites del terreno y no se solapen con ninguna otra defensa u obstáculo. Una defensa colocada en una ubicación inválida supondrá la completa y automática aniquilación de la base. El daño que cada tipo de defensa puede causar a las unidades de *ucos* viene determinado por los siguientes parámetros:

- Rango: es la distancia a la que una defensa puede atacar a un enemigo.
- Salud: representa la cantidad de daño que la defensa puede soportar. Cuando la salud de una defensa llega a cero esta queda destruida, liberando el espacio ocupado por ella.
- Daño: es cantidad de salud que cada ataque resta a las unidades de UCOs afectados.
- Dispersión: cada ataque puede infligir daño a varias unidades de UCOs. Este parámetro indica la distancia a la que el impacto de un proyectil tiene efecto. A medida que la unidad se aleja del punto de impacto recibe menor cantidad de daño.
- Ataques por segundo: es la cantidad de veces que una defensa puede atacar por segundo.

Cada defensa enviada al planeta supone un coste de recursos, por lo que el número de defensas que pueden enviarse al planeta puede depender de los fondos disponibles en ese momento. La escasez de cobre en el futuro ha hecho que este mineral se convierta en el más valioso, dando lugar al *As*, moneda de cambio hecha principalmente de este metal[2].

La primera defensa desplegada en el planeta actuará siempre como centro de extracción del mineral y mantendrá sus propiedades defensivas.

2.3. Unidades de *ucos*

Además de las características de las defensas, las unidades de *ucos* poseen atributos que indican la forma en que se mueven a lo largo del terreno de batalla.

- Velocidad: indica el número de unidades de espacio que la unidad puede recorrer por segundo.

Por simplicidad supondremos que las unidades de *ucos* pueden solaparse entre sí. En cambio, cualquier *uco* que en cualquier momento de la batalla ocupe el mismo espacio que una defensa u obstáculo será inmediatamente destruido.

Cada cierto tiempo un nuevo *uco* entrará al campo de batalla. Lo hará siempre desde alguno de los bordes del terreno.

3. Herramientas para el desarrollo de las prácticas

Para el desarrollo de las prácticas de la asignatura es necesario el uso de dos programas creados al efecto: el simulador de batallas y el visor de batallas. Ambos programas se distribuyen ya compilados y no es posible acceder a su código fuente. Se describe a continuación la función de cada uno de ellos.

Para la realización de las prácticas es necesario, además, contar una serie de herramientas instaladas en el equipo: compilador de c++ que incluya la utilidad *make* y un entorno de compilación de documentos en LaTeX. En el campus virtual tiene disponible un completo manual de la utilidad *make*.

3.1. Simulador de batallas

Se trata de una aplicación desarrollada en c++ que permite simular el ataque de los *ucos* a las bases construidas en los planetas asaltados. No dispone de interfaz gráfica por lo que debe ser ejecutada desde una consola de comandos. Tampoco se trata de una aplicación interactiva. Una vez lanzada efectuará la simulación de la batalla y mostrará el resultado de la misma, sin solicitar la intervención del usuario en ningún caso. En cualquier caso, el usuario puede

[2]El nombre proviene de una de las primeras monedas utilizadas en el imperio romano, compuesta principalmente por cobre

modificar parcialmente su comportamiento mediante el uso de diferentes parámetros que pueden ser añadidos al ser lanzada.

El objetivo de las prácticas del curso consiste en su mayor parte en definir una estrategia óptima para la colocación de las defensas de las bases. Puesto que los alumnos no pueden acceder al código fuente de esta aplicación, se ha optado por definir este comportamiento en una biblioteca dinámica (*.so* en Linux ó *.dll* en Windows), externa a la aplicación. En lugar de generar un programa ejecutable, usted deberá generar esa biblioteca dinámica, que el simulador consultará para colocar las defensas. El simulador buscará esta biblioteca en una dirección concreta, por lo que es importante que siga atentamente las instrucciones que se dan a continuación.

Descargue el fichero *practicas-da.zip* del campus virtual y descomprímalo en alguna ubicación de su disco duro. Sustituya la carpeta *APELLIDO1_APELLIDO2_NOMBRE* por su nombre y apellidos, en mayúscula. Utilice únicamente caracteres del alfabeto inglés (no incluya tildes ni la letra eñe). En adelante nos referiremos a esta carpeta que acaba de renombrar como directorio *BASE* de las prácticas.

Abra el fichero *autor.tex* con algún editor de texto básico y añada su información personal.

Sitúese en la carpeta BASE/simulador y ejecute las siguientes dos órdenes:

```
chmod u+x simulador
./simulador -level 2153
```

Tras un periodo de tiempo la aplicación responderá indicando que la base ha sido asaltada. El atributo *level* indica el código del planeta que va a ser asaltado. A partir de este valor puede obtenerse la siguiente información acerca del planeta que va a ser asaltado:

- tamaño del mapa $= ((level\ div\ 1000)) + 1) \cdot 100$
- dificultad $= level\,\%\ 1000$
- número de obstáculos y defensas $= dificultad \cdot f$

donde f es un factor positivo definido internamente en el simulador. Dicho de otra forma, los tres dígitos menos significativos indican la densidad de elementos dentro del terreno de batalla (obstáculos y defensas). Despreciando esos tres últimos dígitos, el valor del atributo level representa el tamaño del planeta. El valor 2153, indicado en el ejemplo anterior, representa un planeta de dimensiones 300×300 unidades de longitud[3] y con una densidad de obstáculos y defensas relativamente baja (153 sobre 1000).

Puede utilizar la opción *verbose* (sin argumentos) para conocer como se ha desarrollado la batalla con más detalle. Pruebe a ejecutar la siguiente orden.

```
./simulador -level 2153 -verbose
```

Como seguramente haya podido comprobar, al incluir la opción *verbose* también se genera en el directorio una imagen que representa el terreno de batalla. En color negro se muestran las celdas ocupadas por un obstáculo. En color rojo se indica la ubicación de las defensas. La imagen se genera en formato PPM, un formato de imagen muy simple y sin compresión, que permite generar y escribir muy rápidamente las imágenes en el disco duro. Sin embargo, este formato no es siempre soportado de forma nativa por todos los sistemas operativos. En caso de que su sistema operativo no lo soporte o desee realizar un escalado de las imágenes, por ejemplo, puede utilizar la herramienta ImageMagick[4]. Una vez instalada ejecute la siguiente orden para generar una copia escalada de la imagen en formato PNG.

```
convert game.ppm -scale 400x400 game.png
```

El fichero *Makefile* disponible en el directorio en el que se encuentra incluye un objetivo y un *script* para convertir y escalar automáticamente todos los archivos en formato PPM del directorio actual. Para tal fin basta con ejecutar la orden siguiente.

```
make convertimages
```

Aunque con anterioridad se ha indicado que el número de obstáculos y de defensas depende del planeta que va a ser asaltado, con objeto de facilitar la depuración de las aplicaciones a desarrollar el simulador permite establecer ciertos parámetros en su lanzamiento:

[3] $((2 + 1) \cdot 100)$

[4] http://www.imagemagick.org/

- obstacles: permite establecer el número de obstáculos que aparecerán en el terreno de batalla.
- defenses: permite establecer el número de defensas disponibles para colocar en el planeta.
- defensetypes: mediante este parámetro es posible indicar el número de tipos diferentes de defensas que se enviarán al planeta.
- defenserandomness: se trata de un parámetro que establece el porcentaje de variación entre los diferentes tipos de defensas, expresado en el intervalo $[0, 1)$.

Pruebe a ejecutar las siguientes órdenes y compruebe el resultado.

```
./simulador -level 2153 -verbose -obstacles 1 -defenses 20
./simulador -level 2153 -verbose -defenses 20 -defensetypes 2
./simulador -level 2153 -verbose -defenses 20 -defenserandomness 0.9
```

En el desarrollo de las prácticas supondremos que el número de *ucos* es ilimitado, de acuerdo a lo descrito en la sección 2. En cualquier caso, por razones de depuración es posible modificar este comportamiento mediante el parámetro *units*. Ejecute la siguiente orden y compruebe el resultado.

```
./simulador -level 2153 -verbose -units 1
```

De igual forma es posible establecer el número de *ucos* que se unen a la batalla por segundo. Ejecute la siguiente orden para crear un nuevo *uco* cada dos segundos.

```
./simulador -level 2153 -verbose -ups 0.5
```

Por razones de depuración, en ocasiones es necesario establecer una duración máxima de la batalla. Para este fin puede utilizarse el parámetro time. La siguiente orden establece una duración máxima de cinco segundos.

```
./simulador -level 2153 -verbose -time 5
```

Otro de los parámetros que puede resultar útil mientras se desarrollan las aplicaciones solicitadas en las prácticas posteriores es el parámetro *cell*. Mediante este parámetro es posible indicar el número de celdas en las que se ha de dividir el terreno de batalla. A diferencia de los parámetros anteriores, el parámetro *cell* recibe dos valores diferentes separados por una coma. Ejecute la siguiente orden y compruebe el resultado en la imagen *game.ppm*.

```
./simulador -level 2153 -verbose -cells 50,50
```

3.2. Visor de batallas

Para tener una mejor visión del desarrollo de la batalla se incluye en el fichero comprimido una aplicación para visualizar en tres dimensiones las repeticiones de las batallas generadas en el simulador. En primer lugar será necesario, por tanto, simular una batalla en el simulador. Para poder visualizar esta batalla en el visor será necesario guardar el desarrollo de la batalla en un archivo. Ejecute la siguiente orden en el directorio BASE/simulador.

```
./simulador -level 2153 -replay replay.txt
```

Si está usando la versión de 32 bits del material de prácticas, ejecute a continuación las siguientes dos órdenes.

```
chmod u+x ../visor/bin/lin/asedio_player.x86
../visor/bin/lin/asedio_player.x86 -replay replay.txt
```

Si está usando la versión de 64 bits, ejecute las siguientes dos órdenes en lugar de las anteriores.

```
chmod u+x ../visor/bin/lin/asedio_player.x86_64
../visor/bin/lin/asedio_player.x86_64 -replay replay.txt
```

Acepte los parámetros sugeridos y espere a que se cargue la repetición de la batalla. Puede utilizar los siguientes controles para manejar la aplicación:

- Tecla *p*: desbloquear o pausar la repetición.

- Tecla *r*: reiniciar la repetición.
- Tecla *t*: acelerar la repetición.
- Tecla *y*: lee de nuevo el fichero de la repetición y la reinicia.
- Tecla *Esc*: salir de la aplicación.
- Arrastre del ratón manteniendo pulsado el botón derecho: desplazarse a lo largo del planeta.
- Rueda del ratón: acercar o alejar la cámara al terreno.

Revise el fichero *Makefile* y encontrará varios objetivos que facilitan la tarea de creación y visualización de las repeticiones.

Genere las repeticiones de las batallas presentadas en el apartado anterior y cárguelas en el visor.