Práctica 4: Controlando a WALL·E desde Swing

Fecha de entrega: 21 de Abril de 2013. 14:00h

Material proporcionado:

Fichero	Explicación
`TestsPr4.zip`	Proyecto de tests para la práctica.
`validador.zip`	Programa validador de la entrega.
`documentacion.zip`	Documentación de ficheros y clases necesarias para la implementación.
`headingIcons.zip`	Imágenes para representar los iconos de hacia dónde mira WALL·E.

Objetivo: Interfaces gráficas de usuario en Swing. Ventanas y componentes de Swing. Uso avanzado de los argumentos de ejecución

1. Interfaz de consola e interfaz de ventana de Swing

Hasta ahora hemos podido ver a WALL·E funcionando en consola. En esta práctica vamos a desarrollar una interfaz gráfica en Swing que nos permita dar órdenes a WALL·E desde un entorno más amigable. Sin embargo no nos vamos a deshacer de la interfaz de consola sino que vamos a poder lanzar nuestra aplicación usando una u otra interfaz. Así, si ejecutamos:

```
$> java tp.pr4.Main -i console -m map.txt
```

Estaremos ejecutando nuestra aplicación tal y como lo hemos hecho hasta ahora. Sin embargo, si ejecutamos

```
$> java tp.pr4.Main -i swing -m map.txt
```

Entonces ejecutaremos nuestra aplicación usando la interfaz de Swing que describiremos en la sección 2.

Por último, también se puede especificar `-h` para solicitar que la aplicación muestre un mensaje.

La sintaxis concreta para ejecutar nuestra práctica será la siguiente:

```
tp.pr4.Main [-h] [-i <type>] [-m <mapfile>]
 -h,--help                Shows this help message
 -i,--interface <type>    The type of interface: console or swing
 -m,--map <mapfile>       File with the description of the city
```

Para realizar esto se recomienda el uso de bibliotecas auxiliares que facilitan el procesamiento de los argumentos de ejecución de un programa, como Commons CLI de Apache[1].

El funcionamiento de la aplicación será similar al de la práctica anterior en cuanto a valores devueltos: un 0 si todo fue bien, un 1 si hay error en los parámetros y un 2 si el mapa no se puede cargar.

2. Interfaz gráfica

El aspecto de la interfaz gráfica será algo parecido a la Figura 1:

En esta interfaz distinguimos los siguiente elementos:

- Un menú. Actualmente sólo podemos realizar la acción de QUIT desde el menú. Antes de salirse, se pide confirmación al usuario.
- Un panel de acciones donde están los botones que nos permitirán ejecutar las acciones de MOVE, OPERATE, QUIT, PICK y DROP. Adicionalmente, este panel contiene un combo box para seleccionar la dirección en la que queremos girar (acción TURN) y un campo de texto para escribir el nombre del objeto que queremos coger del lugar actual (acción PICK). Las acciones OPERATE y DROP usarán el objeto que esté seleccionado dentro del inventario del robot, que se mostrará en el siguiente panel. La opción QUIT termina la aplicación tal y como se explicó antes.
- Un panel de información del robot que contiene información sobre el nivel combustible, la cantidad de material reciclado que porta y un inventario. El inventario es una tabla en la que cada fila contiene el nombre y la descripción de un objeto cogido por el robot. Estos objetos pueden ser seleccionados para poder ser usados (OPERATE) o soltados (DROP).
- Un panel que representa la ciudad y que tiene un área de texto en el que se presentan las descripciones de los lugares visitados. La ciudad es una rejilla de 11x11 celdas. Cada celda representa un lugar de la ciudad. El lugar inicial se representa en la celda (5,5)[2]. Cada celda muestra el nombre de un lugar, si el lugar ya ha sido visitado. En caso contrario la celda aparece vacía. El lugar en el que actualmente se encuentra el robot se muestra con fondo verde mientras que el resto de lugares visitados se muestran con fondo gris. Cada vez que el robot se mueva (acción MOVE) se presentará el nombre del nuevo lugar en la dirección en la que se ha movido y cambiaremos de lugar. Si pulsamos sobre alguna de las celdas que representan un lugar ya visitado

[1] Más información acerca de esta biblioteca en `http://commons.apache.org/cli/`

[2] Supondremos que la ciudad que ejecutemos cabe completo en esta rejilla

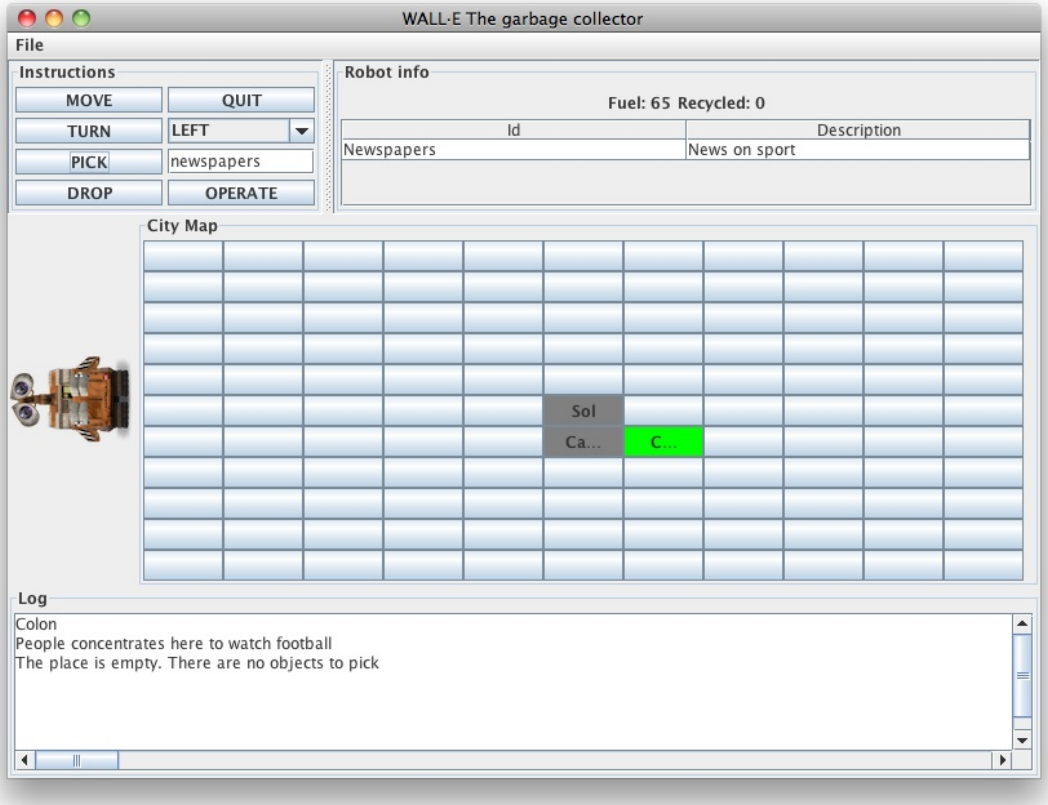

Figura 1: Apariencia de la interfaz gráfica

veremos la descripción de dicho lugar (similar a lo que hacíamos con el comando RADAR).

- Un icono que representa la dirección en la que WALL·E está mirando. Aparece a la izquierda del panel anterior e irá cambiando de orientación según usemos la instrucción TURN (en la figura de ejemplo, WALL·E está mirando hacia el oeste).

En caso de que alguna de las acciones que solicitemos no se puedan realizar presentaremos un mensaje en un cuadro de diálogo emergente (`JOptionPane`). Como se puede ver, algunas de las instrucciones (como SCAN o HELP) han dejado de tener sentido en esta interfaz.

3. Clases para la práctica

Todas las clases de esta práctica estarán en el paquete `tp.pr4`. Vamos a crear un paquete nuevo donde meteremos todas las clases relacionadas con la interfaz de Swing (`tp.pr4.gui`). La documentación proporcionada de estas clases aparece vacía; para la implementación se deberá por tanto elegir los métodos a incluir en cada una de ellas. A modo de guía, se presenta aquí una pequeña explicación de lo que debe hacer cada clase.

- `PlaceCell`. Es un botón que representa una casilla en la ciudad. Tendrá una referencia al lugar visitado que representa.

- `NavigationPanel`. Representa la sección de la ventana que incluye el mapa junto con WALL·E, así como el área de texto con la descripción de los lugares. Contiene por tanto una matriz de `PlaceCells`, una etiqueta (`JLabel`) con el icono de WALL·E, y un área de texto (`JTextArea`). Es el responsable de que cuando el usuario pulse los `PlaceCell` se actualice la descripción del lugar. También debe ser avisado cuando el robot se mueve para actualizar convenientemente la interfaz.

- `RobotPanel` Es la parte de la ventana que contiene el estado del robot, con el combustible y material reciclado actual, así como el inventario. Contiene por tanto `JLabel` y un `JTable` para el inventario.

- `MainWindow` Es la clase que representa la ventana principal del juego. Contiene todos los elementos anteriores y es responsable de crearlos y colocarlos convenientemente. Puede contener los oyentes de las acciones que se pueden realizar desde el panel de acciones. Además, guarda una referencia al `RobotEngine` para poder comunicarle las acciones que el usuario ejecute desde la interfaz.

Para que las clases anteriores puedan presentar esa información, las clases responsables de la lógica del juego necesitarán llamar a algunos de sus métodos, en los sitios donde en la práctica 3 se aparecían los `System.out`. Por ejemplo:

- `NavigationModule`. Contendrá una referencia al `NavigationPanel` que lo representa. Necesita que algunos de sus métodos avisen a la interfaz de Swing sobre los cambios de orientación del robot así como de los cambios de lugar.

- `RobotEngine`. Contiene una referencia al `RobotPanel` que presentará la información del robot. Es necesario añadir llamadas a los métodos de `RobotPanel` cada vez que modifiquemos el nivel de combustible o de material reciclado del robot o cada vez que añadamos o eliminamos objetos del inventario. Así mismo guardará una referencia a la `MainWindow` para avisar sobre los eventos de finalización de la simulación así como para presentar los posibles errores que pueden ocurrir al ejecutar las instrucciones.

Hay que tener en cuenta que estas referencias a los objetos de la interfaz pueden no estar definidos cuando se utilice la interfaz textual, por lo que habrá que tener cuidado al llamar a sus métodos.

Por último, dentro del paquete `tp.pr4.gui` colocaremos `tp.pr4.gui.images`. Este paquete *no* contendrá código Java sino únicamente los ficheros con las imágenes WALL·E. En el proceso de compilación, éstas serán copiadas automáticamente por el IDE al directorio `./bin`, desde donde podrán ser leídas por la aplicación.

4. Opcional: Instrucción UNDO

De manera opcional se pide implementar una instrucción UNDO. Esta instrucción deshace la última acción que se realizó durante la simulación. Esto implica que tenemos que guardar todas las acciones que se realicen durante la simulación. Las acciones deshechas tienen que devolver la simulación al estado anterior y hay que tener en cuenta que hay acciones que no hacen nada al ser deshechas (como por ejemplo, HELP, SCAN o RADAR).

5. Entrega de la práctica

La práctica debe entregarse utilizando el mecanismo de entregas del campus virtual, no más tarde de la fecha indicada en la cabecera de la práctica.

Es indispensable que el código fuente supere los tests de unidad proporcionados y que el fichero enviado pase el programa validador publicado.

Sólo uno de los dos miembros del grupo debe hacerlo, subiendo al campus un fichero llamado `GrupoNN.zip`, donde `NN` representa el número de grupo con dos dígitos.

El fichero debe tener al menos el siguiente contenido[3]:

- Directorio `src` con el código de todas las clases de la práctica.
- Fichero `alumnos.txt` donde se indicará el nombre y apellidos de los componentes del grupo.

[3]Puedes incluir también opcionalmente los ficheros de información del proyecto de Eclipse