Práctica 5: Implementación del MVC

Fecha de entrega: 26 de mayo de 2012. 16:00

Material proporcionado:

Fichero	Explicación
`TestsPr5.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.

Objetivo: Uso del patrón arquitectónico Modelo-Vista-Controlador (MVC).

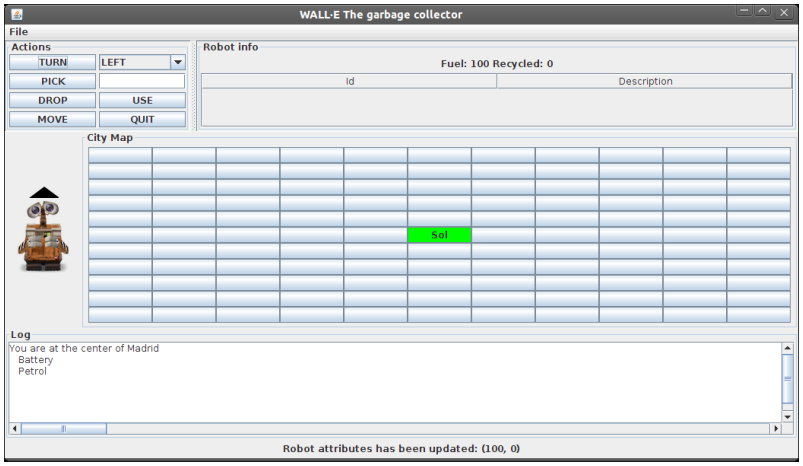
1. Introducción

La práctica anterior sufría de un alto acoplamiento entre el juego en sí y la manera de presentarlo al usuario. Además, no permitía la coexistencia de ambas interfaces (swing y consola) o de otras interfaces adicionales. En esta nueva versión vamos a cambiar principalmente la arquitectura de nuestra práctica, implementando un patrón Modelo Vista Controlador (MVC).

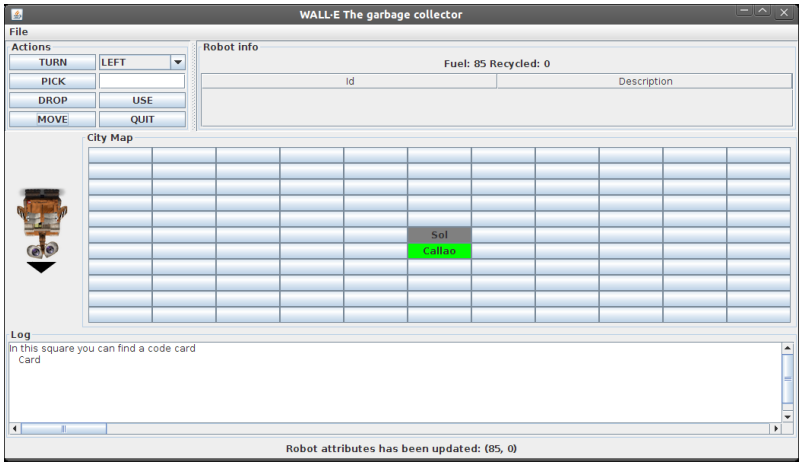
Para ver los beneficios de esta nueva arquitectura nuestra aplicación va a tener una funcionalidad adicional:

- La vista de Swing añade en la parte inferior una etiqueta en la que irán apareciendo mensajes relacionados con la ejecución del juego, como se puede ver en la Figura 1.

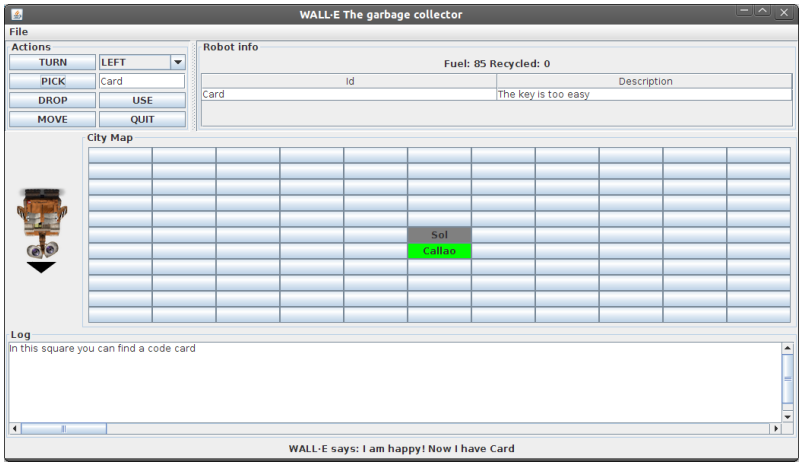
- El argumento `-i` tendrá un posible valor adicional (además de `console` y de `swing`). Si ejecutamos la aplicación con `-i both` tendremos una ejecución de la ventana de swing pero, además, los resultados de la ejecución de los comandos aparecerán en la consola (pero no se podrán ejecutar comandos desde la consola).



(a) Comienzo de la partida



(b) Tras girar 180 grados y avanzar



(c) Tras ejecutar PICK Card

Figura 1: Algunos de los mensajes presentados en la interfaz a lo largo de la ejecución de una partida.

Antes de continuar os recomendamos que penséis la repercusión que tendría implementar estos cambios usando la estructura actual.

La implantación del MVC más las directrices de diseño que se han explicado en clase hacen que tengamos que añadir algunos cambios adicionales a nuestra aventura conversacional, que se detallarán en las próximas secciones.

2. Modelo-Vista-Controlador

Para mejorar nuestro diseño vamos a implementar el patrón MVC. En este patrón de arquitectura se distinguen tres partes

- Modelo. Está compuesto por la clase `RobotEngine`, que será el punto de entrada para ejecutar acciones en el modelo, y el resto de clases usadas por él (`City`, `Place`, `Street`, ...). En el modelo tendremos que eliminar todas la referencias a los componentes de Swing, así como todos los *System.out* que presentaban la información por consola. En su lugar, tendremos que avisar a los observadores sobre los cambios producidos durante la ejecución. El modelo proporciona tres observadores distintos que aglutinan los tres tipos de eventos que pueden ocurrir:

  - `RobotEngineObserver`. Tiene los métodos que se han de implementar si queremos ser avisados de los eventos "de alto nivel" del juego: que el robot ha dicho una frase o se ha cambiado su fuel o material reciclado, que se ha producido un error al intentar ejecutar un comando, se ha solicitado que se presente la ayuda o salir, y que el juego ha concluido.
  - `NavigationObserver`. Eventos relacionados con el movimiento y localización del Robot. Por ejemplo, que cambia su orientación o su posición o que el lugar es modificado porque se ha añadido un item en él.
  - `InventoryObserver`. Eventos relacionados con modificaciones del inventario, como inserción o eliminación de elementos, o notificaciones cuando el jugador mira (con `SCAN`) un Item del inventario.

- Controlador. Tendremos un controlador por cada una de las vistas. Opcionalmente, podemos tener una clase abstracta `Controller` que define la funcionalidad básica común a los dos controladores. Los dos controladores concretos que hay que implementar son:

  - `ConsoleController`. Se usa cuando la aplicación se ejecuta en modo consola. Tiene el bucle de juego, responsable de leer la entrada de usuario, parsearla y solicitar al `RobotEngine` que ejecute el comando solicitado por el usuario.
  - `GUIController`. Se usa cuando la aplicación se ejecuta en modo Swing. Se encarga de mandar al `RobotEngine` las peticiones de ejecución de comandos que se realizan usando las vistas de Swing, que veremos a continuación.

- Vista. Para cada interfaz habrá una (o varias, en el caso de swing) clases responsables de presentar el estado del juego al usuario. Dependiendo de la clase concreta, ésta deberá ser observadora de los tres tipos de eventos proporcionados por el modelo o únicamente de alguno/s de ello/s.

3. Modificaciones adicionales

Aparece un nuevo interfaz, `PlaceInfo` que es implementado por `Place` y que contiene los métodos que permiten acceder al estado del lugar pero no cambiarlo. Por lo tanto, puede verse `PlaceInfo` como un `Place` "constante", pues ninguno de sus métodos permite modificarlo. Este interfaz es utilizado por el modelo en los observadores cuando necesita pasarle a las vistas los lugares por los que se va moviendo el personaje. Es la forma de garantizar que las vistas no podrán cambiar el modelo a pesar de tener referencias a objetos internos suyos.

4. Parte Opcional: Buscar la salida

Implementa una aplicación adicional, `FindSpaceship` que *juegue* una aventura pasada como parámetro y busque el camino más corto para llegar a la nave. La aplicación recibirá como parámetro el fichero del mapa y el número máximo de acciones admisibles para limitar la búsqueda:

```
tp.pr5.FindExit (-d|-max-depth) n (-m|-map) <mapFilename>
```

Ten en cuenta que las acciones pueden consistir en movimientos (giros y avances) pero también cualquier otra operación sobre el entorno, como coger items, usarlos, etc.

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[1]:

- 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.

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