Práctica 1 - Lemmings: Guía de implementación

Este documento pretende ayudarte con la implementación de la práctica 1. Si no lo consideras útil, puedes ignorarlo.

Estructura

Se os ha entregado una plantilla con los siguientes paquetes y ficheros:

- `tp1`
  - *`Main`*
- `tp1.view`
  - *`ConsoleColorsAnsiCodes`*
  - *`ConsoleColorsView`*
  - *`ConsoleView`*
  - *`GameView`*
  - `Messages`
- `tp1.control`
  - `Controller`
- `tp1.utils`
  - *`MyStringUtils`*
- `tp1.logic`
  - `GameObjectContainer`
  - `Direction`
  - `Position`
- `tp1.logic.gameobjects`
  - `Lemming`

Como verás, dentro de cada paquete hay varios ficheros. Los ficheros en cursiva se encuentran completamente implementados y no debes cambiar su código. El resto, en cambio, están parcialmente implementados y puedes (y debes) añadir más métodos y atributos a sus clases. Además, también puedes crear más clases y paquetes si lo consideras necesario.

El código entregado sigue algo parecido al patrón *Modelo-Vista-Controlador* (MVC).

- La **Vista** se corresponde con el paquete `view`. Ahí se encuentran las clases encargadas de mostrar el juego. En nuestro caso, se trata de una simple vista por consola. Se encarga de ello `GameView`, que recibe una instancia de `Game` y se encarga de mostrar el estado de juego en consola. Podríamos tener varias vistas e incluso, en un futuro, se podría cambiar añadiendo, por ejemplo, colores, tal y como hicisteis en FP2, pero no pierdas el tiempo con ello.
- El **Controlador** se corresponde con el paquete `control`. Ahí se encuentra la clase encargada de gestionar el "ciclo de vida" del juego: mientras que el juego no se acaba, solicita órdenes al usuario, las traslada al juego y

llama a la vista. Estamos ante un controlador sencillo, que se limita a leer las órdenes que el usuario teclea por consola, en lugar de capturar pulsaciones de teclas u otras alternativas. Aquí es donde se implementa el ciclo de vida del juego, sin entrar en la lógica del juego.

- El **Modelo** se corresponde con el paquete `logic`, donde se encuentra la lógica del juego, es decir, las reglas del juego y el manejo de todos los objetos del juego. La clase principal de este paquete es `Game`. Para poder realizar su tarea, el juego recibe del controlador la orden correspondiente. Entre los comandos del juego hay órdenes para resetearlo (`reset`) o para no hacer nada (`none`). Como puedes observar, en el modelo se implementa todo lo relativo al juego: número de ciclos, número de lemmings, si se ha terminado, si han ganado los lemmings, etc.

Una vez vista la estructura de la práctica, nos damos cuenta de que hay mucho trabajo por delante y nos planteamos…

¿Por dónde empezar?

División en tareas

En proyectos grandes lo mejor es plantearse hitos pequeños o mini-proyectos que vayan funcionando, a los que luego ir añadiendo extensiones. A su vez, es conveniente dividir estos mini-proyectos en pequeños pasos cuya funcionalidad se pueda probar. En este sentido, os planteamos los siguientes mini-proyectos con los siguientes pasos:

[A] Muestra un mapa con varios Lemmings.

1. Si la plantilla no compila, añade el código imprescindible para que compile. Por ejemplo, si un método aún no implementado tiene que devolver un objeto, escribe por el momento `return null`. A esto te ayuda el compilador. ¡Y ejecútala!
2. Muestra un tablero vacío, para ello si es necesario cambia el valor devuelto por algún método para que se dibuje el **tablero vacío**.
3. Ahora añade varios `Lemmings` tanto en casillas distintas, como en la misma y familizarizaté con cómo se representan.
4. Crea las `Wall` y añádelas al mapa.
5. Realiza el ciclo del juego y haz que el juego termine ejecutando **exit** (comando `[e]xit`).
6. Revisa pequeños detalles (número de ciclos, numeros de lemmings, resto de comandos,…) que has podido ignorar durante las etapas anteriores e incrementa el funcionamiento de la práctica añadiendo el resto de comandos (`[h]elp` y `[r]eset`).

¡Que todo funcione correctamente!

[B] Lemming caminante.

1. Haz que el Lemming se mueva sin caerse y cambie de dirección cuando llegue a los laterales. Para ello ignora al mundo y dibuja un único lemming y haz que se mueva. Para ello implementa un *paso* del lemming.
2. Ahora haz que el Lemming se caiga cuando camine y no haya un objeto solido debajo de sus pies.
3. Haz que cuando se salga por caida del mapa se muera y desaparezca.
4. Haz que cuando aterrice tras una caida de 3 o más posiciones se muera y desaparezca y que camine un *paso* si la caída ha sido menor.

[C] Puerta de salida `ExitDoor`

1. Añade la clase `ExitDoor`.
2. Haz que el siguiente movimiento del Lemming cuando esté caminando y se situe en una puerta de salida consista en salir.

[D] Rol `WalkerRole`

1. Crea el paquete `tp1.logic.lemmingRoles`.
2. Añade la clase `WalkerRole` en dicho paquete.
3. Implementala sus tareas sacando parte del código de lemming a esta clase.

[E] Crea varios escenarios

1. Añade varios escenarios. Trata de cubrir todos los casos posibles y comprueba que funciona correctamente.

¡Felicidades! ¡Parece que has terminado!

Realiza las pruebas que te demos.