Enunciado del proyecto final
MINIDUNGEON

Curso 1º – Programación
Grado en Ingeniería Informática
2016

El proyecto final de programación consiste en la implementación en JAVA de un juego llamado MINIDUNGEON en el que un jugador deberá ir explorando una mazmorra en la que hay distintos pisos. El objetivo del jugador es descender hasta el último piso de la mazmorra con la mayor cantidad de oro posible.

1. Reglas básicas del juego

La mazmorra se representa mediante una cuadrícula tridimensional. El jugador en cada momento está posicionado en una casilla de dicha cuadrícula. El interfaz del juego muestra únicamente el plano que corresponde al piso actual en el que se encuentra el jugador.

Cada piso es un laberinto compuesto por **corredores** y **habitaciones**. Los corredores son pasillos de una casilla de ancho. Las habitaciones son rectángulos de $N \times M$ casillas. Entre cada habitación y los corredores que llegan a ella hay una **puerta**. Todos los corredores y habitaciones están conectados entre sí, es decir, desde cualquier lugar se puede llegar a cualquier otro lugar de alguna manera. En una de las habitaciones existe una casilla especial que permite al jugador descender y ascender de piso.

El jugador, que inicialmente está situado en el piso más alto, se mueve realizando movimientos horizontales y verticales. La distribución de la mazmorra en cada piso está parcialmente oculta para él. A medida que el jugador avanza ésta se va descubriendo. Si se encuentra una puerta es necesario *golpearla* para abrirla. Adicionalmente el jugador se puede encontrar con **enemigos** que podrá eliminar si los *ataca*.

En todo momento el jugador tiene asociadas las siguientes características:

- **Vida actual**: representa el nivel de vida o salud actual del jugador. Cuando la vida actual es cero la partida se termina.
- **Vida máxima**: representa el valor máximo de vida que puede obtener el jugador.
- **Fuerza**: representa el daño que reciben los enemigos y puertas con cada golpe. La fuerza mínima es uno.
- **Percepción**: representa el radio de visión que tiene el jugador sobre el mapa. La percepción mínima es uno.
- **Comida**: representa el nivel de alimentación del jugador. Cuando este nivel es mayor que cero, desciende con las acciones que el jugador lleva a cabo:
  - La acción **Andar** consume una unidad de comida.
  - La acción **Golpear puerta** consume dos unidades de comida.
  - La acción **Atacar** consume tres unidades de comida.

  Si el nivel de comida llega a cero, tanto la fuerza como la percepción se dividen automáticamente entre dos, aunque su valor mínimo nunca podrá ser menor que uno.

Otros elementos que se pueden encontrar en la mazmorra son:

- **Espadas**. Hay una espada en cada piso. Si el jugador se situa en una casilla con espada su fuerza aumenta en una unidad.
- **Corazones**. Hay un corazón en cada piso. Si el jugador se situa en una casilla con un corazón su vida máxima aumenta en X unidades.
- **Ojos**: Hay un ojo cada cinco pisos. Situarse en una casilla con ojo aumenta la percepción del jugador en una unidad.
- **Pociones**. Situarse en una casilla con poción aumenta la vida actual del jugador en Y unidades y como mucho hasta la vida máxima.
- **Oro**. Incrementan la puntuación del jugador.
- **Manzanas**. Incrementan el nivel de alimentación (comida) del jugador.

Cuando un enemigo ataca al jugador, éste pierde vida. Existen enemigos de distintos tipos, que se diferencian en dos aspectos: la fuerza que se requiere para eliminarlos y la cantidad de vida que restan sus ataques al jugador. En cada piso hay un enemigo jefe de piso que cuando se elimina deja caer un mapa que revela el piso completo.

2. Juego ejemplo

Se proporciona al alumno un juego MINIDUNGEON ya programado para que pueda jugar y entender el funcionamiento del mismo.

En la interfaz del juego programado, además de mostrarse el estado actual del mismo, existe un historial y una pequeña consola en la que se pueden escribir comandos. Actualmente están programados los siguientes comandos:

- `explore`: revela el piso entero.
- `floor <número>`: teletransporta al piso indicado si estaba previamente generado.
- `new <semilla>`: crea una nueva mazmorra con la semilla aleatoria indicada.
- `god`: activa o desactiva el modo invulnerable.
- `noclip`: activa o desactiva la colisión con las paredes.
- `save`: imprime una línea en la consola del juego con un comando para que pueda continuarse el juego más tarde.
- `load <parametros>`: Carga una partida con los parámetros descritos. No se conservan ni la exploración de los mapas ni los enemigos eliminados. Los objetos de los pisos anteriores al indicado no aparecen.

3. Descripción de la tarea a realizar

Los alumnos deberán programar el juego completamente, excepto la interfaz del mismo, que ya está programada y es proporcionada por los profesores. El juego programado por los alumnos **no tiene por qué comportarse exactamente** igual que el juego ejemplo que se proporciona, aunque si deberá seguir las reglas descritas en la Sección 1.

Para la programación del juego se deberá realizar un diseño de clases adecuado. **El diseño de clases es muy importante**. Será responsabilidad del alumno asegurarse con su profesor de prácticas correspondiente de que su diseño de clases es correcto.

Una vez esbozado el diseño de clases, la programación del juego será incremental, de forma que se irán incluyendo funcionalidades poco a poco. Para la programación, se recomienta realizar los siguientes pasos:

- **Parte 1: mazmorra, jugador básico, comida y cambio de piso**

  1. Generar la estructura de la mazmorra con sus corredores y habitaciones en cada piso. Inicialmente no considerar las puertas.
  2. Situar al jugador en el primer piso y conseguir que se mueva en él.
  3. Dotar al jugador de unas características iniciales: vida actual, vida máxima, fuerza, percepción, oro y comida.
  4. Disminuir la comida en una unidad cuando el jugador se mueve de una casilla a otra.
  5. Incluir en una habitación de cada piso una casilla especial que permita cambiar entre pisos y dotarla de su funcionalidad.

- **Parte 2: oro, percepción, objetos útiles**

  1. Generar oro distribuido por la mazmorra e implementar la funcionalidad de aumentar el oro del jugador cuando pasa por esas casillas.
  2. Programar la percepción que el jugador tiene de la mazmorra: dado un nivel de percepción se debe ocultar todo aquello que no corresponda a ese nivel.
  3. Incluir en la mazmorra los demás elementos que el jugador se puede encontrar (espadas, corazones, ojos, pociones y manzanas) y dotar al juego de la funcionalidad descrita cuando el jugador recoja estos elementos.

- **Parte opcional: puertas y enemigos**

  1. Incluir las puertas de habitaciones en la mazmorra y dotarlas de funcionalidad.
  2. Incluir en el juego enemigos. Incluir al menos dos tipos distintos de enemigos y dotar al juego de la funcionalidad correspondiente.
  3. Incluir el jefe de piso.
  4. Incluir más enemigos si el alumno lo desea.

4. Comunicación con la interfaz

La interfaz del juego está totalmente programada y proporciona un API (Application Programming Interface) con una serie de métodos a utilizar para interactuar con ella. Estos métodos, así como sus entradas y salidas están descritos en el *Javadoc* que se proporciona al alumno junto con la librería que permite utilizarlos.

Mediante esta API se pueden realizar acciones básicas sobre la interfaz como cambiar el color de una casilla, cambiar un texto y añadir o quitar imágenes.

Además el interfaz captura la última acción que el usuario realiza sobre ella. Esta acción se representa mediante un `String` y puede ser:

- ``` ``left'' ```: se presionó la flecha hacia la izquierda.
- ``` ``right'' ```: se presionó la flecha hacia la derecha.
- ``` ``up'' ```: se presionó la flecha hacia la arriba.
- ``` ``down'' ```: se presionó la flecha hacia abajo.
- ``` ``new game'' ```: se presionó el botón N de nuevo juego.
- ``` ``command <thecommand>'' ```: se introdujo un comando en la consola.

El método que proporciona la última acción realizada se llama `gui.md_getLastAction()`.

5. Documentación a entregar sobre la práctica

El alumno deberá entregar una memoria de como máximo 15 páginas en formato **PDF** que contendrá al menos los siguientes contenidos:

- Portada, índice y breve descripción explicando los contenidos del documento.
- Descripción del diseño de clases realizado.
- Descripción general de los algoritmos más relevantes utilizados en esta práctica.
- Descripción de las partes de la práctica realizadas y de la funcionalidad que el alumno haya decidido incluir en ellas.
- Conclusiones:
  - Conclusiones finales sobre la práctica realizada.
  - Descripción de los problemas encontrados a la hora de realizar esta práctica.
  - Comentarios personales. Opinión personal acerca de la práctica, críticas, etc.

6. Evaluación

La práctica se valorará de 0 a 10 puntos. Para aprobar la práctica es necesario realizar al menos la parte básica, que corresponde a la parte 1 descrita en la Sección 3. Para obtener la máxima nota es necesario tener al menos las partes 1 y 2 implementadas. La parte opcional permitirá subir la nota 1 punto sobre la calificación obtenida en las partes anteriores, hasta un máximo de 10.

Para evaluar la práctica se considerarán los siguientes aspectos:

- Diseño de clases coherente y adecuado.
- Claridad y limpieza del código.
- Ejecución correcta tal como se especifica en este enunciado de la práctica.
- Comentarios en el código fuente.
- Uso exclusivo del temario visto en la asignatura.
- Descripción general del código implementado en la memoria como se describe en la sección 5.
- El respeto a las normas de entrega explicadas en la sección 7.

7. Normas de entrega

El incumplimiento de alguna de las siguientes normas de entrega podría ocasionar el suspenso de la práctica.

La práctica se deberá realizar en grupos de dos personas y deberá ser entregada a través del entregador que se publicará en Aula Global **hasta las 23:55 horas del día 18 de Diciembre de 2015**. Todo el contenido deberá comprimirse en un fichero .zip, cuyo nombre contenga los 6 últimos dígitos del NIA de los alumnos (ej. minidungeon-387633-209339.zip). El fichero deberá incluir:

- Una memoria **en formato PDF** con los contenidos descritos en la sección 5.
- El código fuente de las clases estructurado en carpetas (Carpeta **src/** del proyecto de Eclipse).