ED04_1 Sistema para un Videojuego de Aventura

Introducción

Estás desarrollando un videojuego de aventuras en 2D con mecánicas clásicas (estilo Super Mario Bros, Sonic, etc). El objetivo es modelar los elementos principales del juego y sus interacciones en un diagrama de clases detallado. A continuación, se describen los requisitos del sistema:

Mundo y Progresión

El juego se organiza en mundos, que contienen varias fases. Cada fase tiene enemigos, ítems y objetivos específicos. Al completar una fase, el personaje del jugador avanza a la siguiente, acumulando puntos y desbloqueando nuevas áreas. Los mundos y las fases se identifican mediante un número. Así, por ejemplo, hablaríamos del mundo 1 o de la fase 2 del mundo 1. A los mundos, además, les asignamos un nombre ("Montañas rocosas", "Catacumbas húmedas", ...)

Personaje

El jugador controla un personaje, que se representa como un "muñequito" en pantalla. Este personaje tiene un nombre, una posición en la pantalla, un contador de anillos y un número de vidas. A lo largo de la partida, ese personaje va acumulando puntos.

- Cada vez que recolecta 100 anillos, gana una vida adicional, y el contador de anillos vuelve a cero.
- El personaje puede moverse por la pantalla, recolectar ítems y enfrentar enemigos.
- El personaje agregará puntos por recolección de anillos y monedas, así como cada vez que finalice una fase.

Ítems

En el juego existen tres tipos principales de ítems:

- **Monedas:** Cada moneda tiene un valor en puntos que el personaje puede sumar a su puntuación. En cada fase hay un número máximo de 10 monedas. Dependiendo de cuantas monedas recoja el personaje en la fase, conseguirá más o menos puntos.
- **Anillos:** Además de otorgar puntos, los anillos contribuyen al contador que permite ganar vidas.

- **Power-Ups:** Estos ítems otorgan al personaje un poder especial (por ejemplo, invulnerabilidad temporal o mayor velocidad).

Cada ítem tiene una posición específica en la pantalla, y el personaje debe interactuar con ellos para recolectarlos.

Enemigos

Los enemigos en el juego presentan desafíos para el personaje. Hay dos tipos principales: enemigos comunes y jefes.

- Los enemigos comunes tienen un nivel de dificultad y un rango de movimiento en la pantalla.
- Los jefes son más poderosos, pueden tener habilidades especiales y, en algunos casos, subordinados (enemigos comunes) que actúan como sus protectores. Los jefes, además, tienen nombre propio.
- Los enemigos atacan al personaje reduciendo sus vidas, y al ser derrotados otorgan puntos al jugador.

Pantalla

La pantalla del juego muestra todos los elementos visibles, como el personaje, los enemigos y los ítems.

- Cada elemento visible en la pantalla tiene una posición representada por coordenadas X e Y.
- El sistema debe calcular la distancia entre los elementos para verificar interacciones, como recoger un ítem o ser atacado por un enemigo.
- Todos los elementos visibles comparten propiedades comunes, como su posición en la pantalla, y deben ser modelados adecuadamente.

Gestión de Puntos y Récords

El juego debe llevar un registro de los puntos acumulados por el jugador en cada mundo.

- Se debe mantener un sistema de récords, que almacene el nombre del jugador y su mejor puntuación en ese mundo.
- No es necesario modelar una clase para el jugador, simplemente se debe almacenar en la clase SistemaPuntos su nombre (string) junto a su record de puntuación.

Funcionalidades Adicionales

- El personaje puede interactuar con múltiples enemigos y recolectar varios ítems durante una fase.
- Los cálculos de distancia entre elementos, así como la validación de interacciones, deben implementarse mediante un método reutilizable.

Este modelo debe ser lo suficientemente flexible para permitir la adición de nuevos tipos de ítems o enemigos en el futuro, así como la implementación de mecánicas avanzadas como habilidades adicionales para el personaje o nuevos modos de juego.

Instrucciones de la práctica

Realiza el diagrama de clases cumpliendo con las funcionalidades y requisitos definidos previamente. Realiza un análisis exhaustivo de los requisitos, siguiendo los pasos sugeridos en los apuntes. Añade las aclaraciones que consideres necesarias mediante el uso de notas dentro del propio diagrama.

Una vez analizado el problema, realiza el diagrama en **Visual Paradigm**. Revisa cómo generar un diagrama de clases en Visual Paradigm siguiendo este pequeño tutorial.

Una vez realizado el diagrama deberás entregar el proyecto de Visual Paradigm (.vpp), así como una imagen del diagrama generado, exportándolo como .jpg (Proyec>Export>Active Diagram as Image...). Si deseas incluir algún tipo de comentario sobre decisiones tomadas para realizar el diagrama, inclúyelo como notas dentro del propio diagrama.

Deberás comprimir ambos archivos en uno único y con el nombre siguiendo el formato **apellidos_nombre_ED04_2.zip**

1. **Identificación y uso correcto de relaciones (3 puntos):**
    - Composición (1 punto).
    - Asociación con multiplicidad (1 punto).
    - Herencia y uso de clases abstractas (1 punto).
2. **Uso de interfaces y elementos estáticos (2 puntos):**
    - Definición e implementación correcta de la interfaz (1 punto).
    - Uso de atributos/métodos estáticos en el contexto adecuado (1 punto).
3. **Implementación de métodos y funcionalidades (3 puntos):**
    - Métodos adecuados en cada clase (1.5 puntos).
    - Correcta representación de las funcionalidades del sistema (1.5 puntos).
4. **Claridad y completitud del modelo (1 punto):**
    - Representación clara de atributos y métodos.
    - Inclusión de multiplicidades y anotaciones relevantes.