Battle Royale por turnos en Java

Índice

Introducción

Juego basado en principios de un battle royale por turnos, con generación de mapa procedural, inteligencia artificial, objetos, trampas y otras acciones.

El jugador tiene control total acerca de las acciones llevadas a cabo durante la partida por la interfaz gráfica. El jugador al iniciar la partida puede escoger el tipo de personaje que utilizar, así como su equipamiento.

Una vez dentro de la partida, el jugador podrá moverse a cualquier casilla. Si el jugador se mueve a una casilla especial, podrá lootear o alguna acción especial (como una trampa) podrá ocurrir en el transcurso del turno.

A su vez, los bandidos (jugadores controlados por la máquina) dispondrán de las mismas acciones durante el transcurso de la partida asegurando así una experiencia desafiante para el jugador.

Además, podrás sacar en un fichero todos los turnos realizados en tu partida así como configurar la generación del mapa en un archivo XML.

Resumen

El proyecto desarrollado es un **juego de Battle Royale por turnos en Java**, que combina elementos estratégicos y tácticos en un entorno dinámico y visual para el jugador.

El objetivo principal del juego es proporcionar una experiencia interactiva donde el jugador controla un personaje y compite contra enemigos controlados por una inteligencia artificial (IA) en un mapa generado de forma procedural. El juego integra múltiples componentes y funcionalidades clave:

1. **Clases de Personajes**:
   - **Guerrero**: Orientado al combate cuerpo a cuerpo con habilidades de contraataque.
   - **Arquero**: Especializado ser preciso con sus ataques y reducción de posibilidades de escape del enemigo.
   - **Mago**: Capaz de realizar ataques críticos y con acceso a pociones desde el inicio.

Cada personaje cuenta con atributos únicos y puede equiparse con diversas armas y herramientas que potencian sus habilidades.

2. **Inteligencia Artificial**:
   - La IA implementada permite que los enemigos tomen decisiones autónomas basadas en el estado del juego.
   - Los enemigos pueden atacar, defenderse, moverse o retirarse, lo que añade un nivel de desafío y realismo al juego.

3. **Interfaz Gráfica de Usuario** (GUI):
   - Desarrollada con **Swing**, la GUI permite al jugador interactuar fácilmente con el juego.
   - Incluye paneles de inventario, estadísticas, acciones y un mapa visual que representa el estado actual del juego.

4. **Mecánicas de Juego**:

- **Turnos**: El juego se basa en turnos estructurados donde el jugador y los enemigos realizan sus acciones.
- **Acciones Disponibles**: Movimiento, ataque, defensa, uso de objetos y habilidades especiales.
- **Condiciones de Victoria y Derrota**: El juego finaliza cuando el jugador derrota a todos los enemigos o es derrotado.

Este proyecto representa una implementación completa de un juego de estrategia por turnos, combinando programación avanzada en Java, diseño de interfaces gráficas y mecánicas de juego inteligentes. La integración de enemigos autónomos, junto con la generación dinámica de mapas y un sistema de registros detallado, ofrece una experiencia de juego bastante completa. Además, el enfoque en buenas prácticas de desarrollo y documentación asegura que el proyecto sea sostenible y escalable.

Palabras clave: Java, Battle Royale, Juego por Turnos, Inteligencia Artificial, Generación Procedural, Interfaz Gráfica, Programación Orientada a Objetos.

Jorge Matesanz

.- Ideas sobre personajes y relaciones:

En nuestro proyecto de Battle Royale por turnos, los personajes son el eje central de la experiencia de juego. Hemos estructurado su diseño utilizando herencia e interfaces para garantizar flexibilidad, organización y capacidad de expansión.

Herencia

La clase base **Personaje** define los atributos y comportamientos comunes, como vida, ataque, defensa e inventario. A partir de ella, hemos desarrollado subclases especializadas:

- **Guerrero**: Experto en combate cuerpo a cuerpo, con alta defensa y ataque. Posee la habilidad especial "Ira Espartana", que potencia su contraataque.
- **Arquero**: Especialista en ataques a distancia, con un atributo de puntería que mejora la precisión de sus disparos.
- **Mago**: Maestro de la magia, capaz de realizar ataques críticos que pueden cambiar el curso de una batalla.

Interfaces

Hemos utilizado interfaces como **HabilidadEspecial** para definir comportamientos específicos que cada tipo de personaje puede implementar según su rol. Esto asegura que las habilidades únicas de cada personaje sigan un estándar común, sin afectar la estructura base de las clases.

Relación Herencia-Interfaces

- **Herencia**: Facilita la organización y reutilización del código al establecer una jerarquía lógica de personajes.
- **Interfaces**: Añaden flexibilidad al permitir comportamientos adicionales sin modificar la estructura principal.

Este diseño combina robustez y versatilidad, permitiendo que cada personaje sea único y contribuya de manera dinámica a la jugabilidad.

.- Items y Herramientas:

En nuestro proyecto Battle Royale por Turnos, las herramientas complementan las habilidades de los personajes, permitiéndoles personalizar su estilo de combate y adaptarse a diferentes situaciones. Estas herramientas se dividen en armas, escudos e ítems.

Las armas son esenciales para los ataques y están diseñadas específicamente para cada clase de personaje. Los guerreros cuentan con armas como la Espada Bastarda, la Lanza Puntiaguda y el Hacha Doble Filo. Los arqueros disponen de opciones como el Arco de Guerrilla, el Arco de Precisión y la Ballesta. Por su parte, los magos utilizan herramientas mágicas como la Varita de Cristal, el Bastón de Sabiduría y el Orbe Ancestral. Todas las armas heredan de la clase abstracta "Arma", compartiendo atributos como daño, precisión y probabilidad de crítico.

Los escudos otorgan defensa adicional, reduciendo el daño recibido y afectando las probabilidades de retirada en combate. Existen variantes como el Escudo Ligero, el Escudo Normal y el Escudo Pesado, todos derivados de la clase base "Escudo" para mantener coherencia y facilitar expansiones futuras.

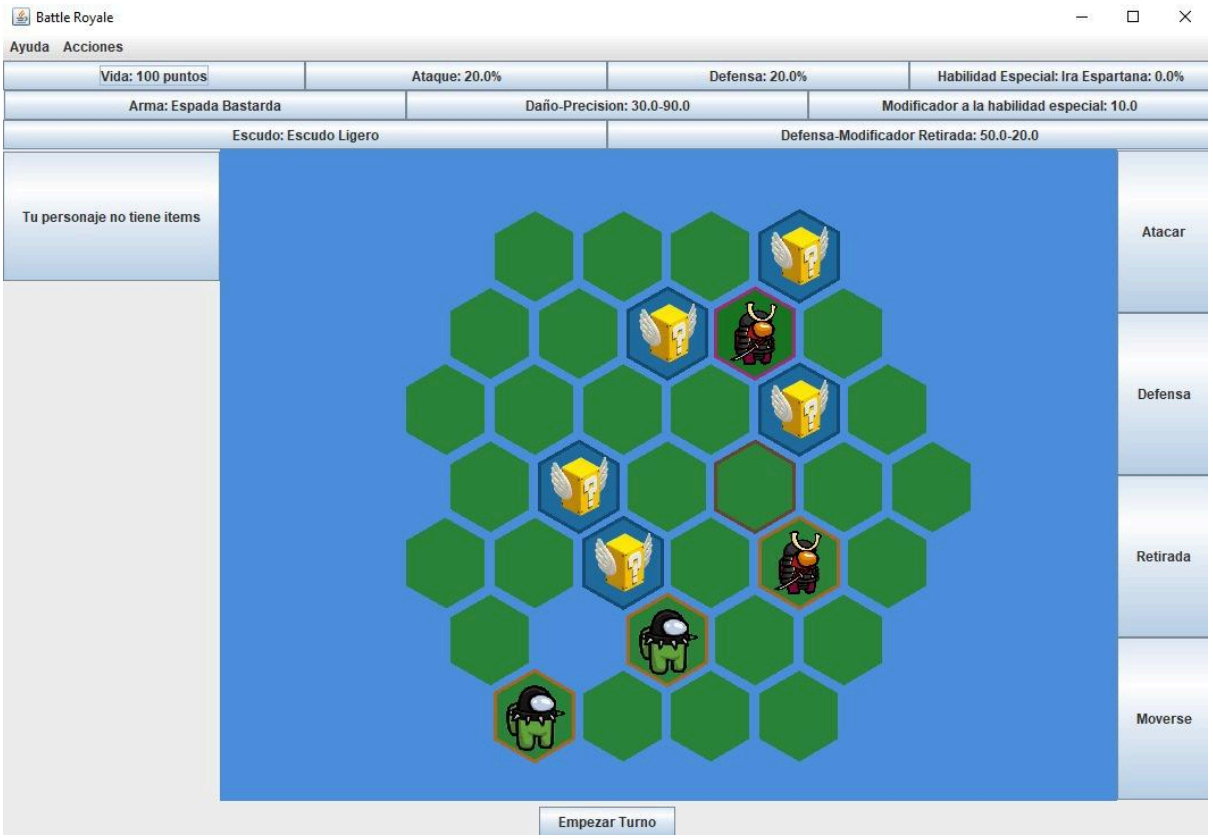
Además, los ítems proporcionan efectos adicionales, ya sea aumentando atributos o restaurando salud. Ejemplos de estos ítems son la Poción de Ataque y la Poción de Defensa, que se gestionan a través de la clase abstracta "Item", con métodos para aplicar y revertir sus efectos.

Cada personaje puede equipar un arma, un escudo y una cantidad limitada de ítems. Estos elementos se integran en la clase "Personaje" mediante atributos específicos, lo que permite combinaciones únicas y define estrategias personalizadas en las batallas.

La clase **CharacterSetup** gestiona la creación y asignación de armas y escudos a los personajes durante la configuración inicial del juego. Por ejemplo, el método **selectWeapon(String characterType)** en **GraphicUserSetup** permite al jugador seleccionar el arma según el tipo de personaje elegido.

.- Parte Gráfica

La interfaz gráfica de usuario (GUI) de **Battle Royale por Turnos** está desarrollada con **Swing** y presenta una disposición intuitiva que permite a los jugadores interactuar fluidamente con el juego. A continuación, se detallan sus componentes principales:



1. **Mapa Hexagonal:**
    - El área central del juego está representada por un **mapa hexagonal** que facilita el movimiento estratégico de los personajes.
    - Cada casilla hexagonal puede contener:

        - **Personajes**: El jugador y los enemigos controlados por la IA se representan con íconos distintivos.
        - **Objetos**: Casillas con signos de interrogación (?) representan **pociones u objetos sorpresa** que los personajes pueden recoger durante la partida.
2. **Panel de Información del Jugador:**

    - En la parte superior de la interfaz, se muestran las estadísticas clave del personaje controlado por el jugador, incluyendo:
        - **Vida**: Representada como puntos de vida (por ejemplo, Vida: 100 puntos).
        - **Ataque y Defensa**: Atributos porcentuales que indican las capacidades ofensivas y defensivas.
        - **Habilidad Especial**: Probabilidad de ataque crítico y su modificador correspondiente.
        - **Arma y Escudo Equipados**: Información del equipo activo, como la **Varita de Cristal** y el **Escudo Ligero**.
3. **Panel de Acciones:**

    - En los extremos derecho e izquierdo se encuentran botones de acción que permiten al jugador elegir entre las siguientes opciones durante su turno:
        - **Atacar**: Realiza un ataque al enemigo.
        - **Defensa**: Aumenta temporalmente la resistencia a los ataques enemigos.
        - **Retirada**: Permite retroceder o huir estratégicamente.
        - **Moverse**: Desplaza al personaje a otra casilla hexagonal disponible.
4. **Indicadores Visuales:**

    - Los personajes tienen **bordes de colores** para diferenciarlos:
        - El **jugador** suele tener un borde distinto al de los enemigos.
        - Los enemigos controlados por la IA también tienen un borde claro para identificarlos.

5. **Botón "Empezar Turno":**

    - Ubicado en la parte inferior central, este botón permite al jugador **iniciar su turno** después de planificar sus acciones.

La interfaz está diseñada con un enfoque en la **claridad y accesibilidad**. La disposición de los elementos guía al jugador hacia las acciones más importantes sin sobrecargar visualmente el espacio. El uso de un mapa hexagonal fomenta la toma de decisiones tácticas, como el movimiento y posicionamiento estratégico.

Sergio Júlbez

.- Tile.java

Para poder realizar la generación procedural del mapa, se deben de instanciar los *tiles* (o casillas) las cuales guardarán la información relacionada con los objetos del juego. Ya sean imágenes, utilizables, personajes entre otros.

Cada tile es responsable de darse color así mismo, en función del tipo asignado, así como ser responsable de añadir distintos rasgos en función de eventos del usuario (*hover* encima del tile, realizar click sobre él para hacer una acción….)

Para el dibujado del Tile en sí, se utiliza el método **createHexagon()** el cual realiza la generación de los vértices utilizando la clase **Polygon** de **java.awt**.

.- MapGenerator.java

Clase encargada de la generación de mapa, empleando un algoritmo para diseñar un patrón romboidal de los Tiles, para una mayor estética.

Esta clase se encarga de la creación procedural del mapa, creando primero todos los Tiles, y después asignando de manera aleatoria los distintos tipos de tiles especiales, así como la instanciación de sus objetos para el inicio de la partida.

.- Entrada / Salida de los ficheros:

La E/S consta del empleo de dos funciones primordiales, siendo **CreateLogs** y **IniciarPartidaFichero.**

La clase de **CreateLogs** se encarga de guardar todos los logs ocasionados durante el transcurso del juego. El objetivo es guardar en cualquier momento de la ejecución del programa los logs y poder hacer un dump de estos cuando el usuario lo desee.

Al principio del programa si es requerido, los valores empleados por **MapGenerator** serán obtenidos a partir del archivo **iniciar.xml** el cual guarda los valores necesarios y mínimos para la generación del mapa. Esta clase **IniciarPartidaFichero** se encarga de leer ese fichero y guardar en variables privadas los valores obtenidos.

Github

Para la creación y organización de este proyecto, hemos empleado un repositorio de GitHub para facilitarnos la vida.