

Trabajo final de trimestre

Programación

Noelia Cantador Bosqued, Fabio Rieker Gonzalez y Mateo García Carrasco

Desarrollo de aplicaciones multiplataforma
Institutos Nebrija

Creación de una aventura conversacional basado en el universo de *Harry Potter*

Índice

1 Introducción	3
2 Guion y Planteamiento del Juego	3
2.1 Flujo de la Narrativa	3
3 Diagramas de Flujo y Modelado	4
3.1 Diagrama de flujo completo de la historia de un personaje	4
3.2 Diagrama de actividades	5
3.3 Diagrama de casos de uso	5
3.3.1 Resumen de Casos de Uso	6
3.3.2 Detalle de Flujos	6
4 Análisis Técnico del Código	7
4.1 Estructuras de Datos Utilizadas	7
4.2 Estructuras de Control y Flujo	7
4.3 Interfaz y Estética (ANSI y UX)	7
4.4 Aleatoriedad y Entrada de Datos	7
5 Tester	8
6 Impacto y gestión del proyecto	8
7 Líneas de Futuro y Mejoras	8
8 Bibliografía	8

1. Introducción

Una aventura conversacional es un género de videojuegos que el jugador debe teclear una serie de acciones que provocan nuevas situaciones hasta llegar a un final y no siempre es el mismo. En este proyecto hemos realizado una aventura conversacional del universo de *Harry Potter* en Java.

2. Guion y Planteamiento del Juego

El juego sigue una estructura de una aventura de texto RPG no lineal con un objetivo final claro, permitiendo al jugador tomar decisiones que afectan sus estadísticas (Vida y Moral) y el desenlace de la historia.

2.1. Flujo de la Narrativa

1. **Acceso al Sistema:** El juego comienza con una pantalla de título estilizada y solicita una contraseña mágica ("lemondrop") para iniciar, simulando la entrada a un lugar secreto.
2. **Selección de Personaje:** El usuario puede elegir entre tres perfiles, cada uno con atributos y habilidades:
 - **Myrtle la Llorona.** Baja vida, pero habilidades especiales espirituales.
 - **Dobby.** Alta moral, comienza con objetos únicos (calcetín).
 - **Estudiante.** Personalizable (nombre) y equilibrado.
3. **El Sombrero Seleccionador.** Si se elige al estudiante, se ejecuta un test de personalidad de 10 preguntas que determina la Casa de Hogwarts (Gryffindor, Slytherin, Ravenclaw o Hufflepuff) y ajusta la moral inicial.
4. **Exploración (Motor del Juego).** El jugador navega por diferentes zonas (Vestíbulo, Gran Comedor, Mazmorras, Biblioteca, Bosque Prohibido, etc.). En cada zona pueden ocurrir eventos:
 - Obtención de objetos (pócimas, varitas, horrocruxes).
 - Aprendizaje de hechizos (*Alohomora, Lumos*).
 - Interacción con NPCs (Hagrid, Snape, Neville y Hermione).
5. **Sistema de Combate.** Encuentros con enemigos (Barón Sanguinario, Acromántula, Mago Oscuro) utilizando un sistema de turnos con cálculo de daño aleatorio y efectos de estado.
6. **Desenlace.** El juego finaliza cuando el jugador completa la historia (5 puntos de progreso), abre la Sala de los Menesteres y derrota al jefe final, o si su vida llega a 0. El final narrativo varía según el nivel de moral acumulado.

3. Diagramas de Flujo y Modelado

En esta sección se presentan los esquemas gráficos que definen la lógica de inicialización, la estructura de navegación del juego y los casos de uso del sistema.

3.1. Diagrama de flujo completo de la historia de un personaje

El siguiente diagrama (Figura 1) ilustra la lógica de ejecución principal. El sistema utiliza una estructura centralizada en el **Vestíbulo**, desde donde el jugador navega hacia cuatro subsistemas:

- **Zona de las Escaleras:** Contiene la lógica para acceder a la Biblioteca, Torre de Astronomía y el combate contra el Jefe Final (Mago Oscuro) si se cumplen las condiciones de historia.
 - **Zona de las Mazmorras:** Incluye verificaciones de inventario (hechizo *Alohomora*) y condiciones de Casa (Slytherin) para acceder a rutas secretas.
 - **Zona del Gran Comedor:** Gestiona la interacción con el Barón Sanguinario y la recuperación de salud.
 - **Zona de los Terrenos:** Maneja la exploración del bosque y el combate contra la Acromántula.

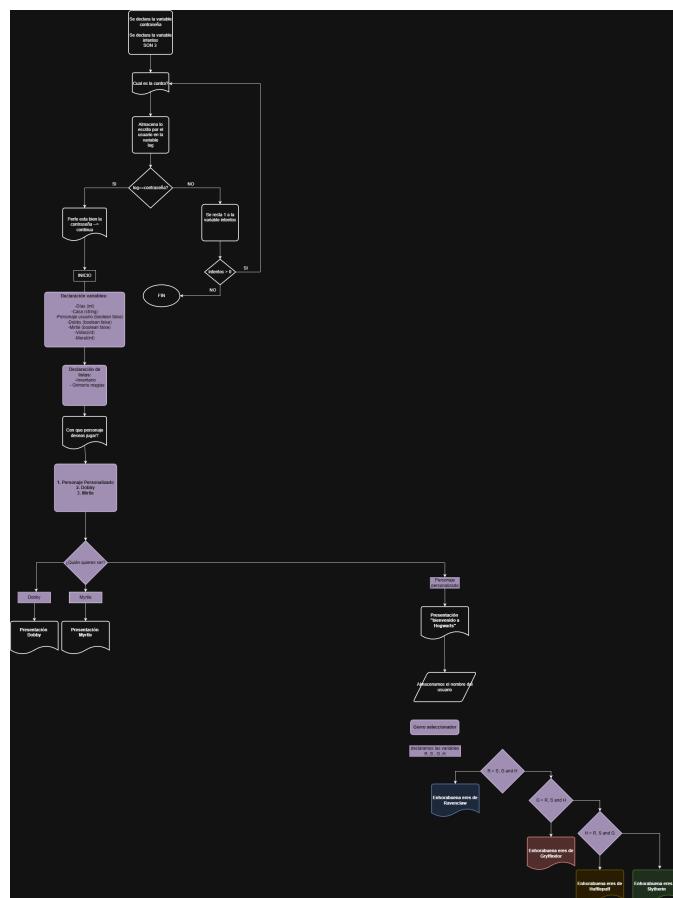


Figura 1: Diagrama de flujo general y navegación por zonas

Nota: Para ver mejor el diagrama de flujo está subido en nuestro repositorio de Github.

3.2. Diagrama de actividades

La Figura 2 detalla la fase de arranque del sistema. Este flujo comprende:

1. **Autenticación:** Bucle de validación de contraseña con límite de 3 intentos.
2. **Instanciación:** Declaración de variables globales (Vidas, Moral) y listas (Inventario).
3. **Selección de Avatar:** Elección entre personajes predefinidos (Dobby, Myrtle) o personalizado.
4. **Lógica del Sombrero Seleccionador:** Algoritmo de decisión que asigna la casa (Gryffindor, Ravenclaw, Hufflepuff o Slytherin) basándose en la comparativa de contadores de atributos (R, S, G, H).

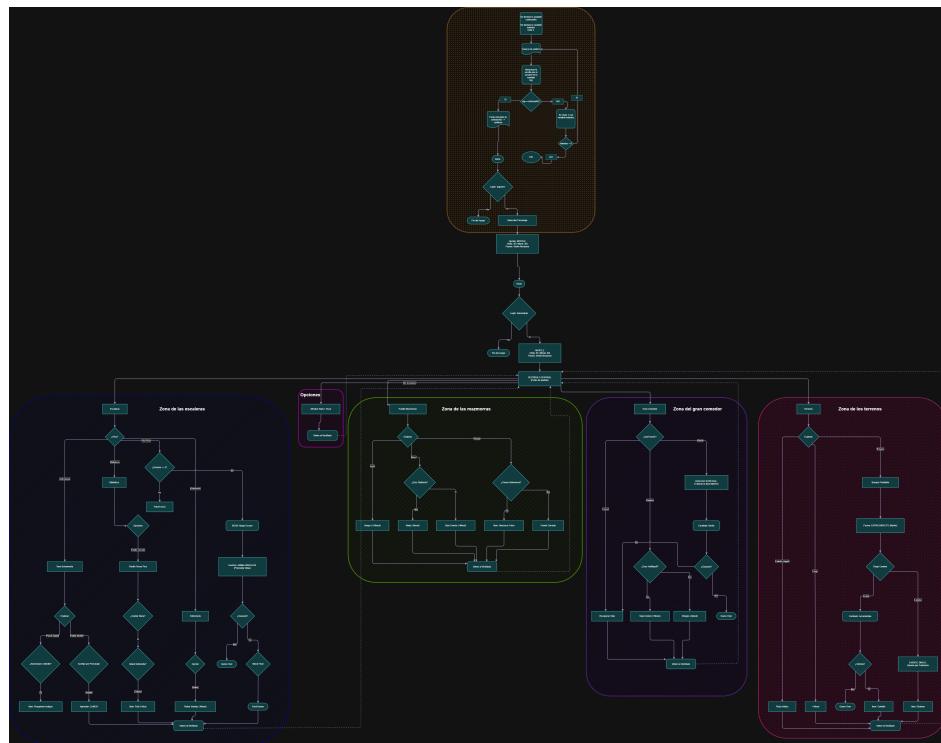


Figura 2: Flujo de inicialización y lógica del Sombrero Seleccionador

3.3. Diagrama de casos de uso

A continuación se describen las interacciones del jugador con el sistema mediante tablas de casos de uso, derivadas de las mecánicas mostradas en los diagramas anteriores.

ID	Actor	Caso de Uso	Descripción Breve
CU01	Jugador	Iniciar Juego	Acceso mediante contraseña ("lemondrop").
CU02	Jugador	Crear Personaje	Selección de arquetipo y asignación de estadísticas.
CU03	Jugador	Selección de Casa	Test del Sombrero Seleccionador.
CU04	Jugador	Navegar	Desplazamiento entre zonas del castillo.
CU05	Jugador	Evento Peeves	Evento aleatorio de daño al moverse.
CU06	Jugador	Combatir	Batalla por turnos contra enemigos.
CU07	Jugador	Gestión Inventario	Uso de objetos y hechizos en batalla.
CU08	Jugador	Finalizar Juego	Enfrentamiento final en la Sala de los Menesteres.

Tabla 1: Tabla resumen de Casos de Uso

3.3.2. Detalle de Flujos

ID	Caso de Uso	Precondición	Flujo Principal
CU01	Autenticación	Iniciar aplicación	1. Sistema pide contraseña. 2. Jugador introduce clave correcta.
CU02	Crear Personaje	Autenticación OK	1. Elegir opción (1-3). 2. Sistema inicializa vida, moral e inventario.
CU03	Test Sombrero	Ser "Estudiante"	1. Responder cuestionario. 2. Asignación de Casa (Algoritmo de contadores).
CU04	Navegación	En exploración	1. Elegir destino del menú numérico. 2. Actualización de variable <code>zonaActual</code> .
CU05	Combate	Encontrar enemigo	1. Bucle de turnos hasta vida 0. 2. Acciones: Atacar o Usar Objeto.
CU06	Aprender Hechizo	Torre o Biblio	1. Interactuar con Neville o Libros. 2. Se añade "Lumos.º Alohomora".
CU07	Interacción Fluffy	Pasillo 3er Piso	1. Elegir: Cantar o Correr. 2. Éxito: Felix Felicis. Fallo: Game Over.
CU08	Batalla Final	Historia ≥ 5	1. Entrar Sala Menesteres. 2. Combate contra Mago Oscuro.

Tabla 2: Especificación detallada de flujos

4. Análisis Técnico del Código

4.1. Estructuras de Datos Utilizadas

Para la gestión de la información, se ha optado por el uso de colecciones dinámicas y variables de estado:

- **ArrayLists.** Se utilizan para gestionar los inventarios de forma dinámica. A diferencia de los arrays estáticos, permiten añadir (`.add()`) y eliminar (`.remove()`) elementos en tiempo de ejecución.
 - Uso en el proyecto: Gestión de objetos, hechizos aprendidos y daños asociados.
- **Variables Primitivas y de Estado:**
 - **Enteros.** Controlan las estadísticas vitales (`vida, moral, casas`) y el progreso de la trama (`contadorHistoria`).
 - **Booleanos.** Actúan como “banderas” (flags) para controlar el flujo narrativo (ej. `puertaSecretaAbierta`) y las características únicas del personaje seleccionado (ej. `esMyrtle`).
 - **Cadenas.** Determinan la ubicación del jugador (`zonaActual`) dentro de la máquina de estados.

4.2. Estructuras de Control y Flujo

El núcleo del juego sigue un patrón de diseño de **Máquina de Estados**:

- **Bucle Principal (while).** Mantiene la ejecución del juego activa mientras la variable `explorando` sea verdadera y el jugador tenga vida.
- **Selección Múltiple (switch-case).** Motor principal de navegación. Evalúa la variable `zonaActual` y ejecuta el bloque de código correspondiente a la habitación donde se encuentra el jugador.
- **Condicionales (if/else).** Gestionan la lógica de combate, la validación de inventario (ej. comprobar si se tiene la “Poción de Vida”) y el sistema de asignación de casa del Sombrero Seleccionador.
- **Manejo de Excepciones (try-catch).** Implementado para controlar la interrupción del hilo durante el efecto de escritura (`Thread.sleep`).

4.3. Interfaz y Estética (ANSI y UX)

Para mejorar la experiencia de usuario en consola, se han implementado códigos de escape ANSI:

- **Colores RGB Personalizados.** Uso de la sintaxis extendida (`\u001B[38;2;R;G;Bm`) para definir colores específicos, como el dorado para los bordes.
- **Reset de Formato.** Uso de `\u001B[0m` para asegurar que el color no se desborde al texto siguiente.
- **Efecto “Máquina de Escribir”.** Uso de la clase `Thread` y el método `sleep(50)` dentro de un bucle `for` para imprimir el texto carácter a carácter, mejorando la inmersión narrativa.

4.4. Aleatoriedad y Entrada de Datos

- **Clase Random.** Genera incertidumbre en el juego, controlando la aparición aleatoria de enemigos (Peeves), la variabilidad del daño en combate y la tasa de acierto de los hechizos.
- **Clase Scanner.** Gestiona la entrada de datos del usuario, tanto para la navegación por menús numéricos como para la introducción de texto libre.

5. Tester

- **Lógica de Game Over** Inicialmente, existía un error lógico donde el bucle `while` principal no evaluaba correctamente si la vida del jugador era menor o igual a 0 tras el turno de un enemigo. Esto provocaba que un jugador muerto pudiera seguir lanzando hechizos. Se solucionó añadiendo una condición de ruptura (`break`) y una validación de estado vital inmediata tras cada ataque recibido.
- **Conflictos de Fusión** Debido al trabajo en paralelo y no saber usar las ramas en GitHub, surgieron errores de duplicidad y eliminación de código al unir las ramas. Esto nos obligó a realizar revisiones manuales línea por línea para asegurar la coherencia del código final.

6. Impacto y gestión del proyecto

Dado que contamos con experiencia como jugadores ocasionales y conocemos el universo de *Harry Potter*, buscamos evitar una estructura lineal secuencial. Por ello, optamos por implementar un concepto de RPG no lineal en esta aventura conversacional.

Este trabajo no habría sido posible sin GitHub, ya que gracias a vincularlo con los IDEs Eclipse y Antigravity hemos podido trabajar de manera asíncrona y remota mediante ramas (branching).

Cabe destacar la excelente coordinación interna del equipo. La comunicación fluida y el compromiso de los tres integrantes permitieron resolver los desafíos técnicos con agilidad y mantener un ritmo de desarrollo constante.

7. Líneas de Futuro y Mejoras

Aunque el proyecto cumple con los objetivos funcionales establecidos, el desarrollo de software es un proceso iterativo. Tras analizar el resultado final y las limitaciones encontradas, hemos identificado varias áreas clave para escalar y profesionalizar la aplicación en futuras versiones:

- **Programación orientada a Objetos (POO).** Al no permitir funciones, clases ni objetos no hemos podido hacer muchas cosas que queríamos como que todo el texto fuera como una máquina de escribir (`Thread.sleep`)
- **Implementación de Interfaz Gráfica (GUI).** Intercambiar la consola por una interfaz creada con JavaFX o Swing. Optamos por una estética de Pixel Art que daría un toque retro.
- **Optimización de la Concurrencia.** Sustituir el uso de `Thread.sleep()`, que bloquea el hilo principal de ejecución, por la clase `javax.swing.Timer`. Esto permitiría realizar animaciones de texto.
- **Sistema de guardado de la partida.** Permitiendo al usuario que pudiera volver a la partida cuando él quiera.
- **Inclusión de Audio Dinámico.** Integrar la librería `javax.sound.sampled` para añadir efectos de sonido al lanzar hechizos y música ambiental que cambie según la zona (tensión en el Bosque Prohibido, calma en la Biblioteca), aumentando la inmersión narrativa.

8. Bibliografía

1. Libros de Harry Potter
2. Películas de Harry Potter.
3. Videojuegos de Harry Potter.
4. <https://www.geeksforgeeks.org/java/thread-sleep-method-in-java-with-examples/>
5. «Realiza un beta tester y dime los errores que ves» ; Consulta a Gemini, versión 3 <https://gemini.google.com/share/e4be1d850afd>
6. Curso Java. Programación genérica. ArrayList; píldoras informáticas <https://www.youtube.com/watch?v=uUWEfmaFOkE>