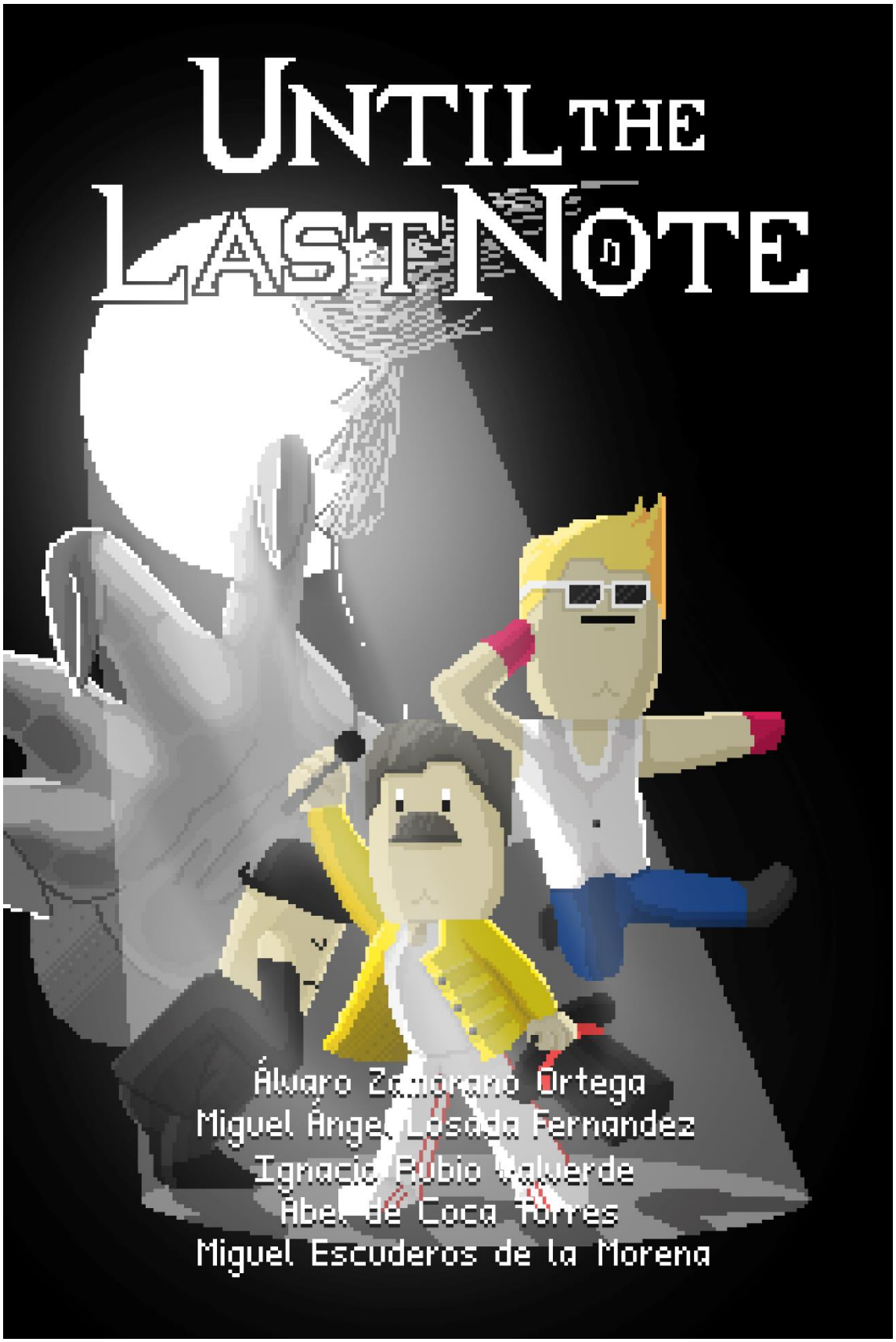


TECNOLOGÍA DE VIDEOJUEGOS-TEAM5

ÍNDICE

INTRODUCCIÓN

El videojuego que hemos desarrollado para esta asignatura se denomina *Until The Last Note.* Se trata de un PixelArt de perspectiva 2D, en el cual vamos investigando a través de varias salas (camerinos y pasillos) hasta encontrarnos con los "jefes", en el que se desarrollará un combate por turnos, en el que el personaje principal atacará y el enemigo responderá.

El juego trata música en sí, la cual la tendremos que salvar de las terribles garras en la que se encuentra con los géneros actuales. Para ello, elegiremos al principio entre tres personajes distintos, el cual cada uno de ellos se caracterizará por representar un género musical (clásico) distinto. A partir de aquí, un narrador nos contará una pequeña introducción y nos guiará en esta intensa aventura, en la que tendremos que derrotar a las géneros que están destruyendo el arte musical. A lo largo de estos duelos, tendremos que analizar a cada personaje y usar la estrategia como arma principal para derrotar hasta el último jefe.

Por otro lado, en el apartado relacionado con la programación,hemos desarrollado el videojuego en lenguaje Java, utilizando el motor Slick2D con las librerías de Java y usando Netbeans como IDE. Así, el equipo artístico se ha ayudado de Photoshop y de ---------- basándose en obras de 8bit, para dar más realismo y dinamismo al juego.

Por lo tanto, Until The Last Note se trata de un videojuego con un intenso apartado gráfico y sonoro, en el que se narra la aventura musical de un personaje que deberá vencer a varios enemigos para devolver a la música a su anterior estado de esplendor.

FASES DEL VIDEOJUEGO Y CONTROLES

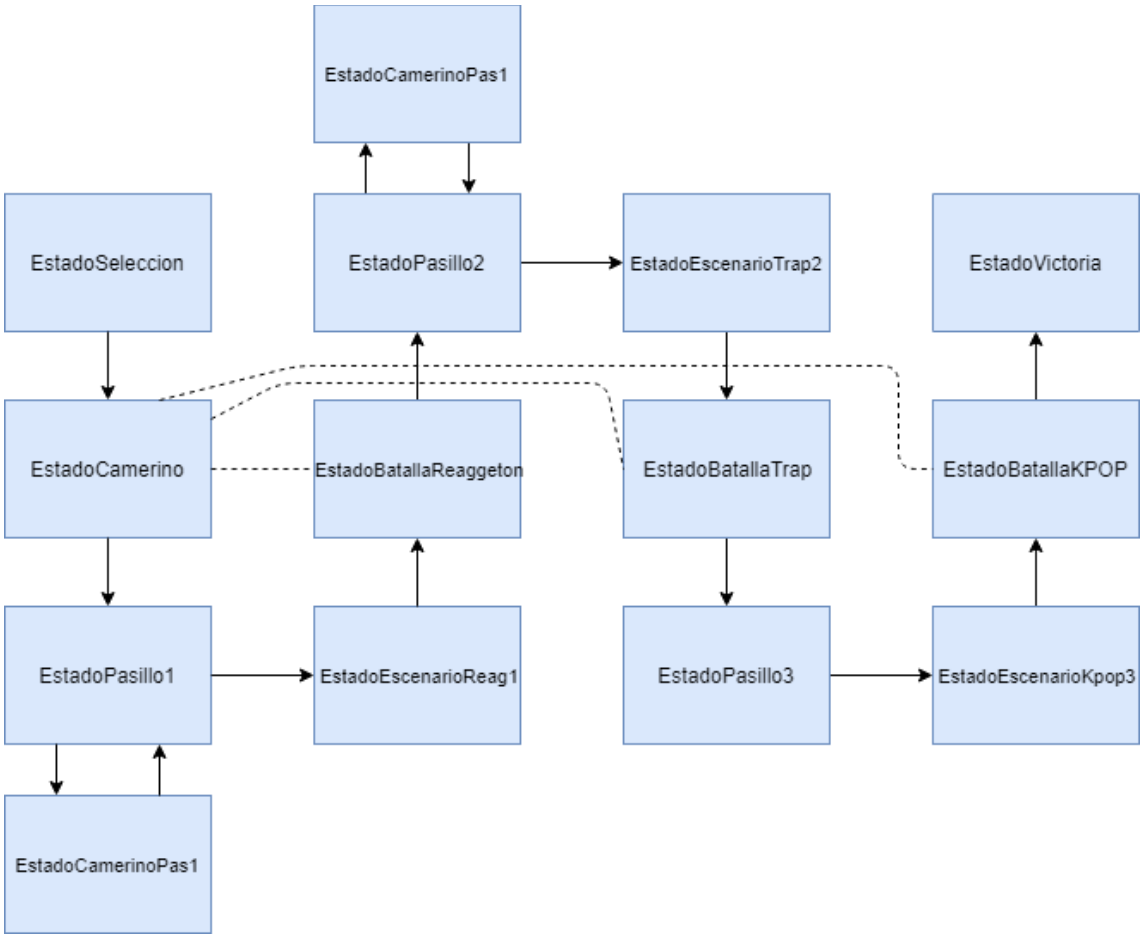
Las clases principales del videjuego extienden de la clase BasicGameState, por lo que Until The Last Note es un videjuego en el cual cada sala será un estado diferente a los demás. Por lo tanto, cada sala (estado) se identificará con un identificador y será administrado de forma independiente de los demás para mayor comodidad al mantenerlos de forma más sencilla.

En primer lugar, en el menú principal hay tres opciones: JUGAR, PERSONAJES Y SALIR. En la opción JUGAR iremos directamente al juego en sí, que explicaremos posteriormente. En la segunda opción, nos saldrá la información de cada personaje del videojuego, con sus características principales y que diferencian a cada uno de ellos a la hora del combate contra los jefes. Y finalmente, en la opción SALIR, saldremos directamente del videjuego, sin nigún tipo de guardado.

Si la opción es la de JUGAR, inicialmente tendremos que elegir al personaje principal de esta aventura. Para ello, los controles serán ENTER para pasar los diálogos, las flechas del teclado para elegir el personaje y ENTER para seleccionar el personaje. Posteriormente, el jugador saldrá en su camerino, en el cual nos hablará un narrador que nos acompañará a lo largo del videojuego. Para movernos usaremos las flechas del

teclado o las teclas WASD para desplazar al personaje hasta la puerta. De este modo saldremos en el primer primer pasillo, y aquí tendremos que averiguar la entrada a otro camerino para encontrar una poción de vida que se usará mas adelante en los combates. Si seguimos por la puerta de la derecha del pasillo, entraremos al primer escenario, en el cual nos encontraremos al primer jefe. Las mecanicas de la batalla son bastante simples. Nos saldrá un HUD referido a los movimientos del personaje principal, y a través de las flechas del teclado elegiremos ATACAR o HUIR. Si la opción es ATACAR (seleccionando por medio de la tecla ENTER), elegiremos el ataque que más nos convenga en ese momento, atendiendo a los usos de cada ataque, su daño y la vida del enemigo. A su vez, podremos recuperar vida (si se posee pociones de regeneración) o aumentar el daño de cada ataque por medio de las teclas V o F respectivamente. Si no se posee usos del ataque seleccionado, se perderá el turno. Al realizar el ataque, el enemigo reponderá con otro o utilizará pociones, dependiendo de su vida. Finalmente, se pasará al siguiente estado si se gana la batalla, o se perderá todo lo realizado y se comenzará de nuevo en el camerino inicial.

Por tanto, esta mecánica de juego se repetirá con los dos jefes siguientes, en los que entremedias se podrá encontrar diferentes pociones de salud y de daño para poder vencer a los siguientes enemigos. Un dato importante es que, tras la batalla contra el segundo jefe, se regenerará la salud y los usos de los ataques del personaje principal.



Transicción de Estados

CÓDIGO

El código principal del juego se encuentra en el paquete *ideavidejuego.* Sus clases principales son las siguientes:

Personaje.java

Esta clase guarda todo lo referente a los personajes del videjuego, tanto al protagonista como a los enemigos. Para ello, almacena todos los atributos relacionados con los personajes:

- Nombre del personaje.
- Vida actual y vida máxima del personaje.
- Ataques del personaje
- Animaciones del movimiento del personaje, tanto a la derecha como a la izquiera (atributo a null en enemigos)
- Música de pasillos y de combate del personaje (atributo a null en enemigos)
- Instancias de la clase Image (imagen) para el HUD de combate de cada personaje (atributo a null en enemigos) y su cuadro de diálogo.
- Número de pociones de vida y de daño que posee en cada momento cada personaje.

Atendiendo a sus métodos, aparte de los correspondientes *get* y *set* para obtener y establecer los valores a cada atributo, posee los siguientes métodos:

- public boolean useHealthPotion() : este método es usado en el combate musical y sumará 75 puntos de vida a la vida actual. Para ello, retornará un valor booleano true si lo ha realizado con éxito (restando en uno las pociones de vida que tenga) o false si le quedan pociones de vida.

- public boolean useDmgPotion() : al igual que el anterior, pero en vez de recuperar salud, aumentará el daño de cada ataque.

- public String atacar(Personaje penemigo, int seleccion) : se trata del método principal del ataque del personaje principal al enemigo. Para ello, se deberá meter como valores de entrada al personaje enemigo y un número que identificará el ataque que haya seleccionado el jugador. Si al ataque elegido le quedan usos, se le restará a uno el número de usos que tenga y, si ha sido realizado con éxito (explicación en la clase Ataque), el ataque podrá ser normal (daño normal) o daño crítico (explicación en la clase Ataque), restando el daño hecho al enemigo. De este método, sonará el sonido característico del ataque.

Finalmente, este método devolverá un texto (String), en el que aparecerá el ataque realizado, si ha sido realizado con éxito, si ha sido crítico y el daño que ha hecho al enemigo.

- public String ataqueEnemigo(Personaje personajeBueno) : se trata del método a partir del cual ataca el enemigo al personaje protagonista (valor de entrada).
  Este método se diferencia del anterior en que el personaje enemigo elige un ataque de forma aleatoria, elegiendo siempre un ataque que le queden usos a través de una estructura *while().* En lo referente a si ha sido acertado o no y si es crítico o no, es exactamente igual al método anterior.

- public boolean Probabilidad(int x) : se trata de un método que devolverá true o false en función de si 10 números generados aleatoriamente (1 ó 0) y su suma es mayor (true) o menor (false) que el valor de entrada x. Este método se usa cuando en la batalla, el personaje enemigo tiene una probabilidad de tomar un poción de salud o de daño.

Ataque.java

Esta clase guarda todo lo referente a los ataques de los personajes. Sus atributos principales son:

- Nombre del ataque
- Daño: número que indica el daño que resta vida al personaje.
- Usos y usos máximos: número de intentos actualesque se puede realizar el ataque y número de intentos máximos.
- Sonido que suena al realizar el ataque.´
- Probabilidad de fallo del ataque.

Atendiendo a sus métodos, aparte de los correspondientes *get* y *set* para obtener y establecer los valores a cada atributo, posee los siguientes métodos:

- public boolean isAcertado() : este método devuelve true o false en función de si la suma de diez números (multiplicada posteriormente por 10) aleatorios entre el 0 y el 1 es mayor que la probabilidad de fallo (true si es mayor).

- public boolean isCritico(int x) : este método devuelve devuelve si el ataque es critico o no (el daño del ataque se multiplica por 2). Al igual que en el anterior método, si la suma de diez números (multiplicada posteriormente por 10) aleatorios entre el 0 y el 1 es mayor que el valor de entrada x (probabilidad de fallo), se devolverá true.

ClaseEstatica.java

Esta clase guarda al personaje principal elegido y al enemigo actual. A su vez, nos sirve de ayuda para guardar la música que suena en el menú inicial del videojuego, el sonido de los pasos del personaje al moverse, y también en el combate, guardando si el último ataque ha sido acertado o no, y en caso de haber sido acertado, se guardara el ataque realizado. Finalmente, también nos sirve de ayuda para guardar el último estado en el que se encontró el personaje protagonista para hacer más dinámico la transicción entre estados.

ClasePunto.java

Esta clase guarda las coordenadas de los puntos en los que situamos las imágenes en los diferentes estados.

ClaseSprite.java

Esta clase hereda de la clase Image, y representa a los diferentes Sprites que colocamos en los diferentes estados. Tiene como atributo principal una instancia de la clase Punto, en el cual se guardan las coordenadas en las que se sitúa la imagen.

ESTADOS:

Principal.java

Se trata de la clase principal del juego, a partir de la cual se ejecuta el programa. Esta clase extiende de StateBasedGame, de modo que en esta clase añadimos todos los estados que conforman el videojuego.

En esta clase aplicamos el tamaño de la ventana en la que se ejecutará el videojuego. De este modo, fijamos el tamaño en 1080x720, y para una mayor comodidad y visualización, establecemos que no se visualicen los FPS.