Fight Knights

Memoria: Desarrollo de un videojuego en Java

17 de Mayo de 2018

Alejandro Fernández Maceira

Oscar Fernández Sánchez

Adrián Palma Lima

Daniel López Moreno

Pedro Elías Cabada

Índice.

La pantalla principal.

1. Introducción

Fight Knights es un videojuego de cartas en 2D por turnos con perspectiva top-down.

En la época medieval, dos caballeros luchan por ver quién es el elegido para salvar el reino de la amenaza que se avecina. Para ello, lucharán en combate singular valiéndose de sus cartas y de su estrategia.

Solo uno será el vencedor. ¿Quién se alzará con la victoria?

1.2. Metodología/Herramientas

Para la edición de las cartas, el tablero y demás elementos gráficos se han usado programas como Photoshop, Paint, TiledMap etc. Para la creación del propio juego, la librería Slick2D del lenguaje de programación Java ha sido empleada.

2. Diseño del videojuego

El tipo de videojuego elegido es cartas basado en fantasía medieval con perspectiva top-down. Es un videojuego con un concepto simple, pensado para el gran público, cuyas normas son claras y no requiere una gran curva de aprendizaje.

2.1. Requisitos mínimos

Los requisitos mínimos de hardware para jugar al juego no son muy elevados, cualquier ordenador actual puede ejecutarlo sin problemas de estabilidad ni cuelgues en el equipo, ya que, al no tener inteligencia artificial, el uso de CPU es escaso.

En cuanto a los requisitos software, un entorno de desarrollo compatible con Java es necesario para la ejecución, como por ejemplo NetBeans. No se necesita otro software específico, pues la librería Slick2D ya viene incluida en el propio proyecto.

2.2. Objetivo del juego

El objetivo del videojuego es ser el vencedor en el combate entre caballeros. Para ello, las cartas que se invocan en la mesa atacarán al jugador enemigo y le restarán puntos

de vida en función de su ataque. Gana el jugador que consiga llevar a 0 los puntos de vida del contrario.

2.3. Lógica del juego

Los elementos del juego son:

a) Jugadores. Representados por su salud en un orbe azul, es el avatar del usuario. Solo dispone de la vida actual, y para jugar deberá usar las cartas de su mazo.
b) Unidades. Representadas por cartas, son los objetos con los que los jugadores combatirán. Cada jugador solo puede tener 5 cartas en su mano y 5 cartas en el tablero como máximo en todo momento. Cada unidad tiene características específicas, como curar vida al jugador o eliminar cartas del mazo del oponente. Sin embargo, para usar una carta, el jugador necesita maná.
c) Maná. Elemento necesario para usar cartas. Los jugadores empiezan su turno con 1 de maná, y cada turno se incrementará en uno. En cada carta está indicado el coste en maná, al usar la carta, el maná se restará del contador.
d) Tablero. Donde se colocarán las unidades de cada carta. Está representado con un estilo medieval, y en él se muestra la información de los jugadores y las cartas seleccionadas, así como botones para rendirse, salir, reiniciar o pasar turno.

El juego comienza en un turno aleatorio, puede tocarle al jugador 1 o al 2. Se mostrarán 3 cartas en el mazo, y el jugador decide lo que hacer.

La partida termina cuando un jugador quiera rendirse, salir o uno de los dos llegue a 0 puntos de vida.

2.4. Controles

El método de control es con el ratón. Se seleccionarán los cuadros de texto haciendo clic con el ratón, al igual que las cartas y el ataque. Para pausar la partida, basta con pulsar la tecla de escape y el juego se detendrá.

2.5. Interfaz

Al ejecutar el programa, aparece una pantalla de carga. Después aparecerá un menú con la opción de pulsar start. Se clica en el cuadro de texto y empieza la partida. En el tablero hay varias opciones:

Captura de un momento del juego.

a. Reiniciar. Al pulsarse, reiniciará la partida y generará nuevas cartas para los jugadores.
b. Rendirse. Declara la victoria al jugador contrario.
c. Salir. Sale del juego.
d. Pasar turno. Termina el turno del jugador actual y empieza el del siguiente. Aumenta en 1 el maná del nuevo jugador.

Para dar una ambientación medieval característica al juego, se reproduce un hilo musical con inspiración medieval. Si se quiere desactivar el sonido, en el menú inicial hay una opción para ello.

El juego se reproduce a 1280x720 de resolución, en modo ventana. En todo momento se muestran los FPS, para comprobar que todo funciona correctamente.

2.6. Patrones de diseño

Para la realización del videojuego se han empleado los siguientes patrones de diseño de videojuegos:

-Singleton. Para la creación de una sola instancia de partida. Este patrón de diseño permite que solo haya activa una instancia de la clase Match. Con esto se evita posibles pérdidas de información o escritura en instancias equivocadas. Para implementar el patrón, se ha creado un constructor privado, al que se accede mediante un método público y que no permite ser llamado más de una vez, garantizando la unidad de la instancia.

-Factory. Para la creación de las cartas. Este patrón de diseño permite crear instancias de distintos tipos de cartas fácilmente. Se ha implementado creando un switch en el constructor de la clase Unit. Este constructor recibe un String como parámetro, el nombre de la carta, y busca en el switch el nombre que coincide. Una vez encontrado, inicializa los atributos de la clase con el valor establecido dentro del código case del switch.

3. Implementación

El juego está basado en estados, cada ventana es un estado diferente, y se ejecuta desde la clase MainEstados. Los métodos básicos render, update e init se han implementado. En init se inicializan las variables necesarias en cada clase, en update se actualiza la posición del ratón y si se ha hecho clic o no, y en render se generan las imágenes de las cartas, del tablero y demás elementos visuales.

La clase MainEstados contiene la definición de cada pantalla/estado y el método main desde el que se ejecutará la aplicación. También se define el tamaño de pantalla y la resolución.

La clase PantallaCarga representa la primera pantalla que verá el jugador. Contiene el logo del equipo y un indicador para comprobar el estado de la carga del juego. No hay ninguna interacción con el jugador en esta pantalla. De esta clase se pasa a la siguiente, Menú.

La clase Menú es el menú principal del juego, donde el jugador elegirá empezar la partida. Contiene el logo del juego. Para capturar el ratón se emplea el método isMousePressed() de la clase Input de Slick2D. Si el ratón está pulsado y se encuentra dentro de las coordenadas del cuadro de texto dibujado, entonces empieza la partida. No empezará hasta que se pulse.

La clase Unit representa una unidad. El constructor es un switch en el que se inicializan las cartas según su número. También contiene métodos que emplean las cartas como initSkill, la habilidad que tienen al ponerse en la mesa; attackSkill, la habilidad al atacar; o deathSkill, la habilidad al morir. Según la carta, estas habilidades realizan una acción u otra.

La clase Match emplea el patrón de diseño singleton, creando una única instancia de la partida, con todos los valores necesarios para su funcionamiento: los mazos de cada jugador, representados con un arrayList, el turno, la vida de los jugadores, su maná, las unidades en el tablero, etc. También se incluyen métodos get y set de cada atributo establecido.

La clase GameMethods contiene los métodos necesarios para el correcto funcionamiento de las cartas y sus acciones. Un método es cambiar turno, en el que se

aumenta el maná del jugador siguiente, se suma una carta a su mano y cambia el turno al siguiente. Otro método es executeAttack, donde se comprueban los valores necesarios para que una carta ataque a otra, teniendo en cuenta si existe un efecto de provocación activo en ese momento. Para ello, se resta la vida de la carta objetivo en tantos puntos como ataque tenga la carta atacante.

También está en esta clase el método para atacar al jugador, en el que se pulsa la carta atacante y se selecciona el orbe azul del contrario. Después se resta tanta vida como ataque tenga la carta. El último método es invokeCard, donde se comprueba que el jugador tiene energía, y después se elimina la carta del mazo y se coloca en el tablero.

La clase PantallaJuego es donde se desarrolla la acción del videojuego. Aquí se crean y se usan todos los métodos y atributos de las clases anteriores, para el correcto desarrollo de la partida. Lo más importante es la captura de coordenadas del ratón. Según la posición en la que se encuentre, se realiza una acción u otra. También se realiza la lógica de los botones y su acción, como reiniciar, pausar o salir. Para usar una carta, primero se comprueba si el ratón está sobre esa carta, luego se obtienen datos de la carta y se arrastra al tablero, automáticamente se colocará en una posición disponible, o en ninguna si no hay espacio. Si se desplaza el cursor sobre las cartas del mazo, se ampliará la imagen para ver sus características. Dentro del tablero, cada unidad tiene a sus lados unos orbes azul y rojo que representan la vida y el ataque, respectivamente. Luego se captura la zona de actuación del ratón, y se actúa según la entrada del jugador. Si pulsa en la carta y luego en el orbe de vida del contrario, se atacará. Una carta solo puede atacar una vez por turno, y si hay alguna carta con efecto de provocación deberá ser atacada. Cuando la partida acaba, bien porque el jugador eligió rendirse, o porque la vida de alguno llegó a 0, se pasa a la siguiente clase.

La clase VictoryScreen se muestra cuando se cumple una de las circunstancias anteriores. Contiene la información sobre quién es el ganador de la partida y dos botones, uno para salir, y otro para volver a jugar.

La pantalla de victoria.

La clase PantallaPausa se muestra cuando, dentro del juego principal, un jugador pulsa la tecla escape del teclado. La partida se detendrá y se reanudará una vez que se haga clic en cualquier parte de la pantalla.

4. Conclusión

Fight Knights es un juego de cartas medieval desarrollado en Java con el motor Slick2D y que permite a dos jugadores en el mismo ordenador jugar por turnos. Tiene una interfaz amigable y una gran carga lógica y matemática.

5. Bibliografía y fuentes de apoyo

Para la parte de código se ha consultado el manual de Java de Oracle y los foros de Stack Overflow.

En la parte de diseño artístico, las cartas han sido creadas por nosotros, cortesía de Adrián Palma Lima. El tablero también lo hemos creado nosotros, gracias a Pedro Elías Cabada. La música la hemos cogido del artista Kevin Mc Leod, sin copyright. El estilo de letra es Hellgrazer de la página www.dafont.com.