GRUPO DE TRABAJO: NOENTIENDO

Memoria del proyecto: The Freemium Adventure

Tecnología de videojuegos

Adrián de Juan Lora
Ángel Martín Segura
Alberto Cabrera Plata
Álvaro Bustos Lara
David Fernández
Sergio Salvador Salgado

2016

Índice:

Contenido

1 - Introducción

El propósito de este documento es hacer un análisis detallado del desarrollo del proyecto *The Freemium Adventure*, un videojuego creado y desarrollado por el grupo de trabajo *Noentiendo*.

Para ello, se dará una breve explicación del concepto creativo del juego, también se detallará el diseño de alto nivel del programa y, posteriormente, se hará un análisis de las clases más importantes del proyecto de NetBeans.

Cabe destacar que *Noentiendo* ha usado una estructura de desarrollo en espiral, es decir, ha ido revisando los requisitos del videojuego, comprobando si se cumplían, y añadiendo o quitando requisitos si se decidía que era mejor que el proyecto evolucionara de manera distinta a la prevista. Todo esto ha estado coordinado por Ángel Martin Segura, jefe de proyecto.

2 - Concepto creativo

The Freemium Adventure (TFA), pretende ser un videojuego cómico, de acción sencilla, de corta duración y con controles simples e intuitivos.

El juego pretende hacer burla a la maniobra empresarial de desarrollo de expansiones para un mismo videojuego, siendo los juegos denominados como freemium el máximo exponente de ellos. En un juego freemium, aunque el juego se pueda jugar de manera gratuita sólo podrás avanzar en el juego si compras los diferentes paquetes de expansión o mejora. Con la premisa de burlarse de ello, TFA lanza al jugador a una pequeña aventura lineal en la que cada ocasión es aprovechada para recordar al jugador que se gaste dinero en paquetes de mejora inexistentes y con precios ridículamente altos.

Los diferentes escenarios del videojuego ayudan a este concepto, usando tentadoras puertas con supuesto botín detrás, o cortándole el paso al protagonista si no está dispuesto a comprar el DLC de ese momento.

Para cumplir con esto decidimos desarrollar un juego rogue-like, en el que:

1) El protagonista solo cuenta con un arco y una daga para enfrentarse a varios enemigos

2) Estos han de comportarse de manera distinta, atacando de diferentes maneras

3) El protagonista debe ser capaz de interactuar con ciertos objetos de su entorno, para mostrar texto, para cambiar de mapa, o destara ciertos eventos.

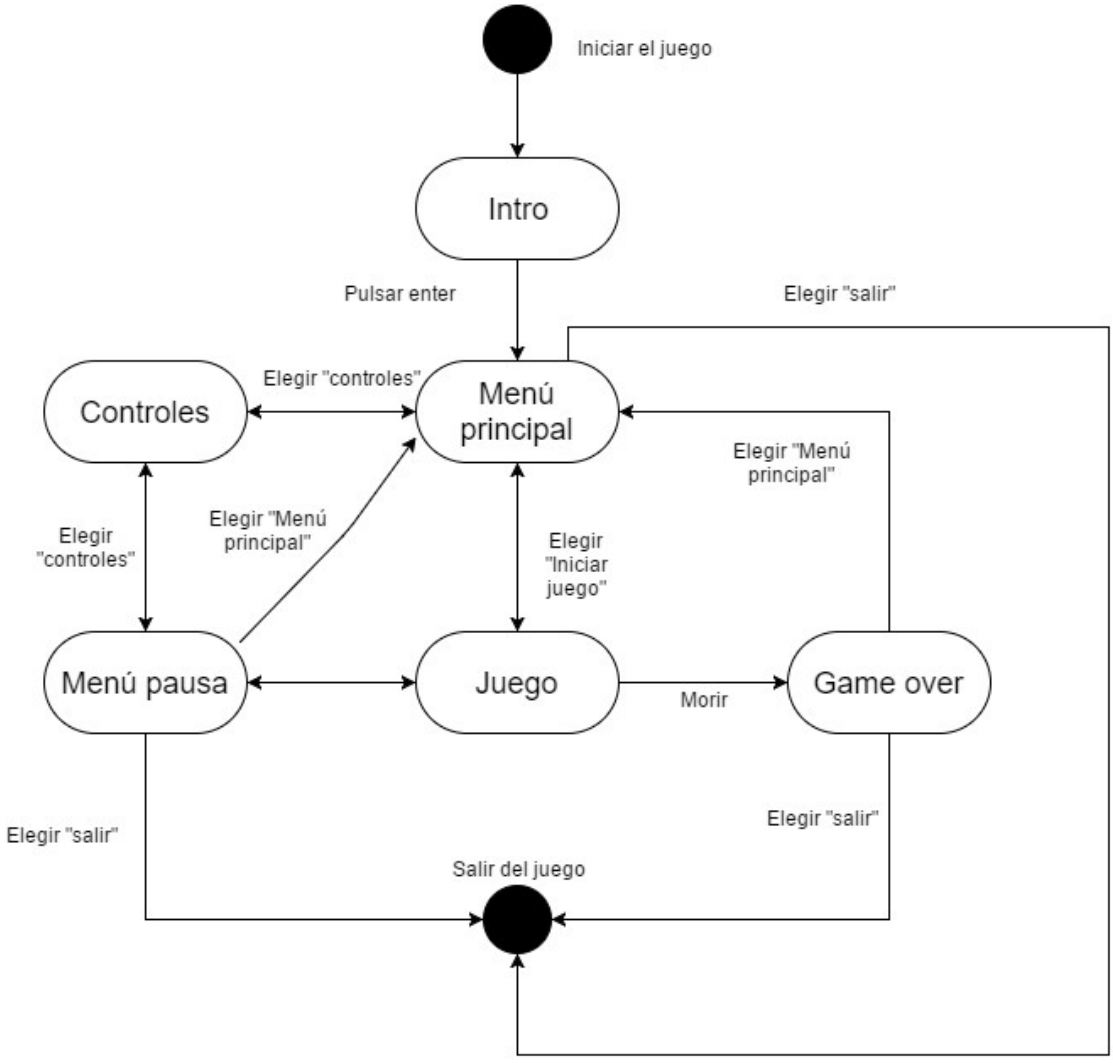
4) El jugador debe de ser capaz de pausar el juego en cualquier momento

5) Si el jugador muere, se le presentan dos opciones, o vuelve al menú principal o sale del juego.

El concepto creativo es explicado de forma más detallada en el documento de diseño adjunto.

3 - Diseño de alto nivel

El programa se basa en una "StateMachine", o Máquina de estados, siendo el ciclo normal de ejecución el siguiente:



Explicación de los diferentes estados:

-Intro: Muestra una pequeña imagen de introducción del grupo *Noentiendo*.

-Menú principal: Le da al usuario la posibilidad de salir del juego, empezar a juagar o consultar los controles.

-Juego: El juego en sí.

-Menú pausa: El juego se pausa completamente si se accede a este menú, dentro de este menú el usuario puede volver al menú principal,

-Controles: Este estado muestra los controles al jugador, cabe destacar que el diagrama es algo confuso, al salir de este estado te devuelve al estado anterior, es decir al menú principal si estabas en él o al menú de pausa en caso contrario.

-Game over: Se accede a este estado al morir, te permite volver al menú principal para seguir jugando o salir del juego directamente.

4 - Estructura del proyecto.

Para facilitar el desarrollo del proyecto y minimizar la aparición de errores (así como el tiempo ocupado en detectarlos), se decidió dividir el proyecto en múltiples paquetes, cada uno encargado de una tarea diferente. En este apartado se explicará de qué se encarga cada paquete de clases, así como las clases significativas de este.

Estados

El paquete estados se compone de las diferentes clases que conforman los estados de la StateMachine, todos los estados explicados en el apartado anterior están separados en diferentes clases, que heredan de una clase padre abstracta llamada "Estado", la mayoría de estados se dedican sólo a poner una imagen de fondo, y dar las opciones a seleccionar para cambiar de estado, así como comprobar en update() que teclas se han pulsado para saber que se está seleccionando. Sin embargo, la clase Juego se encarga de llevar el juego en sí, es decir, leer la entrada del jugador y actuar acorde a ella (moverse, atacar, interactuar...)

Juego

Este paquete está compuesto por una sola clase "App" que es la clase ejecutable, se encarga de inicializar todos los elementos necesarios para que el jugador pueda empezar a jugar.

Lógica

Este paquete se forma a partir de las clases que se encargan de la "lógica" del programa, es decir, cosas que han de pasar sin interacción del usuario.

La clase cámara se encarga de implementar una cámara móvil, que siempre siga al jugador.

La clase GestorColisiones se encarga de manejar las colisiones con objetos.

Mapa

Este paquete está formado por las diferentes clases que se ocupan de guardar los diferentes mapas del juego, haciendo más fácil la transición entre diferentes mapas.

Todos los mapas heredan de una clase padre "Mapa" que implementa métodos para seleccionar el mapa y aplicar colisiones a la capa de patrones colisiones.

La capa de patrones colisiones, es la más baja de todas para que no se vea, y se encarga de poner patrones con los que siempre colisionará el jugador y hacer más sencillo las colisiones con las paredes, decorado, etc.

Personajes

Este paquete guarda dentro las diferentes clases que implementan a los distintos personajes del juego, todas ellas heredan de una clase padre "Personaje".

Todas las clases menos la de "Heroe" implementan enemigos, con un sistema de IA similar, los enemigos pueden estar patrullando, yendo hacia el protagonista si este está en rango visual, atacándole si esta en rango de ataque, atacándole más rápido si el combate se endurece o huyendo porque tiene poca vida.

La clase "Heroe" implementa las diferentes acciones que es capaz de realizar el jugador, como atacar, moverse, interactuar...