Programación
1º DAM

Proyecto Final

ONEIRIC

I.E.S. San Vicente
San Vicente del Raspeig (Alicante)
Curso 2018/2019

Alumnos:
Kevin Marín
Jaime Rebollo

Profesor:
Nacho Cabanes

1. Introducción

Nombre del proyecto
Oneiric

Desarrollado por
Kevin Marín
Jaime Rebollo

Descripción breve del proyecto
Es un juego estilo RPG con batallas dinámicas estilo Pokemon (1 vs 1), en el cual podrás manejar al protagonista X debiendo recorrer la senda del programador. Tendrás que avanzar a lo largo del mapa e ir consiguiendo fragementos de Código para conseguir habilidades nuevas y enfrentarte a los grandes Boses. Estará desarrollado con C# usando la biblioteca Tao.SDL.

2. Funcionalidad del proyecto

Cuando inicias el juego te aparece el logo de la empresa que lo ha desarrollado, después llegarás al menú principal, en el cual podrás elegir entre:

- **Continuar**: Donde se cargará la última jugada.
- **Nueva Partida**: Donde se empezará el juego desde cero.
- **Cargar Partida**: Mostrará los datos de la partida, donde el juagdor podrá elegir entre un máximo de 3 partidas guardadas.
- **Opciones**: Donde el jugador podrá cambiar las opciones del juego, entre ellas el idioma (Español-Ingles), también elegir el nivel de dificultad (Fácil-Medio-Dificil-Hacker). Los niveles de dificultad se diferenciarán entre ellos únicamente por un multiplicador que se aplicará a todos los daños que hagan los enemigos. También cambiar a pantalla completa y elegir el nivel de volumen.
- **Ayuda**: Donde el jugador podrá consultar los controles del juego.
- **Salir**: Se podrá salir del juego.

En cualquiera de las 3 primeras opciones comenzará una partida, donde aparecerá el mapa y el personaje. Según donde el jugador se mueva hacia los extremos del mapa, éste se irá moviendo entre fragmentos del mapa. A lo largo del mapa existirán zonas en las cuales cada X pasos (número aleatorio, reseteandose después de cada lucha) aparecerá un combate, en el caso de vencer tú experiencía aumentará y podrás conseguir frágmentos de código de distintos tipos, mientrás que si pierdes resucitarás en el último punto guardado.

Al conseguir x cantidad de frágmentos podrás crear una sentencia de código completa, según que sentencia recibiras una recompensa u otra.

El mapa estará ambientado en un instituto exageradamente grande, en algunas zonas podrás encontrar ordenadores, los cuales te podrán dar x cantidad de fragmentos u otros objetos.

El jugador también tendrá un inventario con objetos que podrá utilizar durante el combate o fuera de él, además de equipables.

Habrán zonas en las que encontraremos boses, para poder enfrentarnos correctamente a ellos necesitaremos poseer un cierto nivel y unas específicas habilidades.

3. Prototipo de la pantalla

La apariencia que se persigue es ésta:

4. Entregas previstas

1. ~~Hacer el esqueleto del juego, incluyendo todas las clases necesarias~~
2. ~~Realizar el menú principal del juego, incluyendo todos los submenús~~
3. ~~Movimiento del personaje~~ ~~y combates aleatorios~~
4. ~~Cargar y dibujar mapa desde archivos y colisiones y scroll~~
5. ~~Guardado de partida~~ ~~e interacción con elementos estáticos (cofres/ordenadores)~~
6. ~~Implementación del menú in-game (Al pulsar I)~~
7. ~~Creación de la interfaz de combate y los enemigos~~
8. ~~Realizar lógica de los combates~~
9. ~~Implementar lógica de los enemigos (normales)~~
10. Creación de Eventos y NPCs (no enemigos)
11. Implementación de habilidades del personaje
12. Agregar Bosses
13. Añadir música y sonidos
14. Realizar zonas de extras
15. Implementar sistema de trucos avanzado (inmortalidad, aumento de estadísticas)
16. Añadir Modo Supervivencia (muchos combates seguidos sin descanso)

5. Trabajo diario realizado

- 2019/04/17 – Se han creado el esqueleto del juego incluyendo todas sus clases, además se ha organizado todo en subcarpetas.
- 2019/05/06 – Creación y organización del menú principal y el submenú de opciones.
- 2019/05/13 – Se ha implementado el movimiento del personaje, cargar y dibujar el mapa desde un archivo, colisiones y scroll.

- 2019/05/15 – Implementación de multilenguaje, creación del menú *in-game* y creación de temporizador para tiempo jugado.
- 2019/05/16 – Se ha implementado el guardado y cargado de partidas además de crear fichero donde se guardarán los errores que ocurran. Para complementar el guardado se ha implementado la interfaz básica del guardado en el menu in-Game.
- 2019/05/17 – Se ha corregido el sistema de guardado y se ha implementado el guardado de las opciones y su persistencia al cerrar el juego. Se ha completado la clase Chest.
- 2019/05/20 – Implementación del inventario del personaje, interacción con los cofres y asignación aleatoria de *items* a los cofres.
- 2019/05/22 – Se han añadido los atributos restantes al jugador, se han creado las clases de los enemigos y se ha actualizado el guardado y carga de ficheros con los datos correspondientes.
- 2019/05/23 – Se ha mejorado el sistema de combate.
- 2019/05/24 – Implementada la lógica de combate y el historial de combate.
- 2019/05/27 – Mejora del historial de combate, implementación del drop de objetos al derrotar enemigos.
- 2019/05/29 – Listado de objetos en menu juego, implementado uso de objetos.
- 2019/05/30 – Mejorado el uso de objetos, posibilidad de usar objetos consumibles en batalla. Finalización del juego al morir el personaje principal enviándolo a GameOver y mostrar estadísticas del jugador en el menú *in-game*.

6. Problemas encontrados durante el desarrollo y sus soluciones

- 2019/04/17 – Tras crear las clases hemos necesitado organizarlas en directorios. Para ello hemos creado las carpetas, hemos movido los fuentes a sus respectivos directorios y hemos modificado las rutas de los mismos en el fichero ".csproj".

7. Estructuras utilizadas

- ~~if~~
- ~~else~~
- ~~Conectores: && y/o || y/o !~~
- ~~switch~~
- ~~?~~
- ~~while~~
- ~~for~~
- ~~foreach~~
- ~~try-catch~~
- ~~(arrays)~~
- ~~struct~~
- ~~(clases + herencia)~~
- ~~(propiedades o getters y setters)~~
- ~~public, protected, (opcional) private~~
- ~~ArrayList o List<>~~
- ~~Hashtable o SortedList o Dictionary~~
- ~~StreamReader o FileStream o BinaryReader~~
- ~~ref o out~~
- ~~(manejo avanzado de cadenas: substring, contains, split, replace o similares)~~

- ~~(consola avanzada o SDL o Windows Forms o Unity)~~

8. Mejoras o restricciones respecto a la idea inicial

(…)

9. Capturas de pantalla del proyecto final

(…)

10. Código fuente del proyecto final

(…)