Carlos Llobregat \_

Manuel Martín Vázquez

2020-2021

**Técnico superior en Desarrollo de Aplicaciones Multiplataforma**

**Proyecto de Desarrollo de Aplicaciones Multiplataforma**

ÍNDICE

Índice 1

[Introducción 2](#_Toc72705588)

[Tecnologías usadas 3](#_Toc72705589)

[Funcionamiento 5](#_Toc72705590)

[Descripción Gráfica de la Aplicación 6](#_Toc72705591)

[Presupuesto 7](#_Toc72705592)

[Glosario 8](#_Toc72705593)

[Referencias 9](#_Toc72705594)

[Anexo 10](#_Toc72705595)

# Introducción

# Tecnologías usadas

Durante el desarrollo de nuestra aplicación, hemos tenido en cuenta las necesidades técnicas que nuestro proyecto necesitaría. Sin embargo, también teníamos que tener en cuenta el tiempo que tardaríamos en aprender la tecnología o herramienta necesaria para utilizarla. Por lo cual, al final decidimos utilizar las herramientas que durante nuestro curso formativo hemos utilizado. En este caso:

* Unity. (2019.14.18f)
* Microsoft Visual Studio 2019.
* C#
* GitHub (GitHub Desktop)
* Discord
* Google Drive

Para el desarrollo del proyecto, en su gran mayoría, hemos utilizando Unity. Ya que es la base en la que nuestro proyecto se basa. Sin embargo, para utilizar características avanzadas del programa Unity, es necesario un IDE.

En este caso, decidimos utilizar Visual Studio ya que este nos permite depurar durante la ejecución de la aplicación, además de las diversas herramientas de ayuda al desarrollo.

Dentro de Unity, es necesario utilizar el lenguaje de programación C# para crear componentes que adjuntar a objecto y definir el comportamiento de estos.

Uno de los puntos importantes durante el desarrollo de nuestra aplicación ha sido la facilidad del desarrollo en equipo, ya que nos ha facilitado el desarrollo simultaneo de funcionalidades sin la necesidad de que uno de nosotros pare. Una de las plataformas más utilizadas para ello es GitHub y para la sincronización de los archivos, GitHub Desktop que nos permite realizar las tareas sin entrar en la página web.

Para la realización del proyecto, se ha necesitado una gran comunicación entre ambos integrantes del equipo y ya que ambos teníamos experiencia con la plataforma, hemos realizado esta comunicación ya sea por llamadas, chat o video llamadas por Discord.

Durante el desarrollo, hemos intercambiado diferentes Assets y archivos que o no han sido incorporados en la aplicación o que han sido utilizados entre nosotros para el desarrollo de esta. Para ello, hemos utilizado la plataforma de Google Drive.

# Funcionamiento

# Descripción Gráfica de la Aplicación

La aplicación se puede dividir en 3 partes:

* La pantalla de inicio, pausa y gameover.
* El juego en sí.
* La interfaz gráfica.

### Inicio

Dentro de la pantalla de inicio nos encontramos diferentes elementos, estos cambian en particular si se ha realizado un guardado con anterioridad en el dispositivo del progreso. En ese caso, dentro del menú de inicio podemos ver que se ha añadido la opción de ‘Continuar’ que al iniciar la aplicación por primera vez no debería salir.

Esta opción nos permite continuar la ejecución de la aplicación desde el último punto en el que se ha realizado el guardado. Este guardado se puede realizar desde el menú de pausa, el cual se va a desarrollar más adelante.

El siguiente elemento que se puede distinguir es ‘New Game’, este nos permite iniciar el juego. El cual comenzará igual para todos los usuarios.

El siguiente punto es ‘Créditos’ en la que se mostrará información sobre el desarrollo de la aplicación.

El último punto es ‘Exit’ que nos permite salir de la aplicación.

### Pausa

Dentro de este menú, que se puede acceder una vez se ha iniciado el juego, a través de ya sea la tecla ‘ESC’ o a través del botón que se puede percibir en la esquina superior derecha.

El primer elemento nos da la opción de ‘Continuar’ con la ejecución del juego.

El siguiente elemento, nos permite realizar un guardado de la situación actual con ‘Save’.

A continuación, tenemos la opción de mostrar la ventana de ‘Settings’, que una vez se abre nos permite cambiar los ajustes incluso durante la ejecución del juego, ya que es independiente del menú de pausa. Este se va a desarrollar en el apartado de UI.

El último elemento nos permite cancelar la ejecución del juego y volver a la ventana de inicio, aunque este no guarda el progreso de este.

### Game Over

Esta pantalla solo se puede visualizar cuando durante la ejecución del juego, la vida de nuestro personaje llega a 0 o cuando se ha finalizado el juego.

Esta pantalla nos indica de forma positiva si se ha finalizado el juego de la forma correcta o negativa si la vida de nuestro personaje llega a 0.

Esta pantalla solo nos da la opción de volver al menú principal.

### UI

Los elementos que consideramos UI se encuentran formados por:

* Barra superior.
* Botón de reducir.
* Botón de cerrar.
* Contenido.
* Botón de escalado.

La barra superior nos permite mover el elemento a diferentes posiciones de la pantalla. Dentro de esta, se encuentran los botones de reducir y cerrar. Estos como su nombre indica, nos permiten reducir y cerrar la ventana.

El botón de escalado nos permite hacer una ventana más grande o más pequeña hasta cierto grado. Actualmente, la única ventana que nos permite reducir o aumentar el tamaño de estos es la ventana de ‘Inventario’.

Dentro de estos elementos se encuentran las ventanas de:

* Estadísticas.
* Inventario.
* Dialogo.
* Ajustes
* Objeto

La ventana de estadísticas nos muestra el estado actual de nuestro personaje. En este caso, nos permite ver la vida, el nivel y cantidad de experiencia que nuestro personaje tiene.

La ventana de inventario nos muestra la cantidad de ‘Item’ que el personaje tiene y cuales están equipados. Cuando un ‘Item’ es seleccionado dentro de esta ventana, se abre otra ventana llamada Objeto o ‘Item’ que nos permite visualizar las propiedades específicas de este ‘Item’.

La ventana de dialogo nos permite leer lo que un NPC nos quiera decir, además de en ciertos casos realizar algunas opciones.

La ventana de Ajustes o ‘Settings’, nos permite modificar la configuración del juego. En particular, nos permite modificar las teclas asociadas a ciertas acciones tales como atacar o moverse. También nos permite volver a mostrar ciertas ventanas que hemos ocultado, además de cambiar el color de estas.

# Presupuesto

# Glosario

# Referencias

# Anexo

A parte de lo anteriormente comentado, nos gustaría comentar una parte muy importante sobre el desarrollo de la aplicación.

Durante nuestro desarrollo, nos hemos enfocado en la facilidad de añadir contenido y funcionalidades sin la necesidad de profundizar en la base del diseño del código. Esto se puede ver con facilidad en la incorporación de herencia en los controladores de ya sea del jugador como los enemigos.

Ya sean los enemigos como el jugador hereda de la clase ‘EntityBaseController’, que incorpora las funcionalidades básicas que permiten que estos funcionen y puedan interactuar con otras entidades. El comportamiento de las diferentes entidades, se puede customizar al sobre escribir la función ‘Behaviour’, que es lo que actualmente ocurre para las diferentes entidades como ‘Jugador’, ‘Enemigo’, ’Flecha’ y ‘Objecto’.

No solo eso, sino que hemos enfocado el funcionamiento de la aplicación en torno a eventos. Cada vez que se realiza una acción, tal como ‘Atacar’, ‘Recibir daño’, ‘Moverse’ o ‘Obtener Exp’ invocan eventos a los que uno se puede incorporar para realizar diferentes funcionalidades.

Sobre estos eventos, el juego actualmente esta enfocado y la gran parte de las mecánicas del juego se ejecutan de esa manera. Todas las entidades, tales como el personaje o los enemigos, contiene componentes de tipo ‘InteractionController’ e ‘InteractionManager’.

El ‘InteractionController’ nos permite determinar que posibles acciones una entidad puede realizar, que acciones se pueden realizar sobre la entidad y guardar las acciones a realizar cuando los diferentes eventos ocurran.

El ‘InteractionManager’ se encarga de determinar sobre que entidad realizar alguna de las acciones y proporcionar los datos necesarios para realizarlas.

Dentro de ‘EntityBaseController’, se encuentran la mayoría de las posibles acciones que ejecutar cuando un evento se invoca. Uno de los ejemplos es: ‘Attack’ y ‘Attacked’, que determina que hacer cuando la entidad ha atacado o ha sido atacada. Ya que los controladores que utilizan todas las entidades heredan de esta, la funcionalidad básica esta incorporada para cualquier enemigo o entidad que se vaya a implementar en el futuro sin necesidad de modificar gravemente el código de la aplicación. No solo eso, sino que facilita y hace más rápida la incorporación de nuevas entidades con comportamientos personalizados.