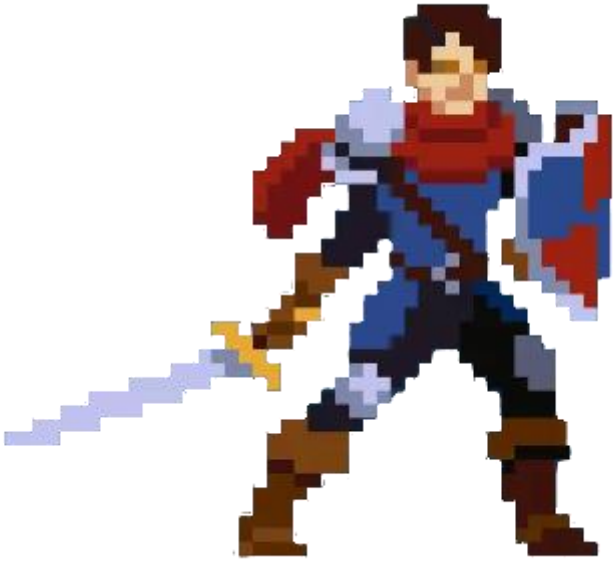


Práctica – Creación de un videojuego

GRUPO N

IVÁN GUTIÉRREZ AGUILERA
CARLOS JIMÉNEZ LÓPEZ
ADRIÁN RUBIO GARRIDO

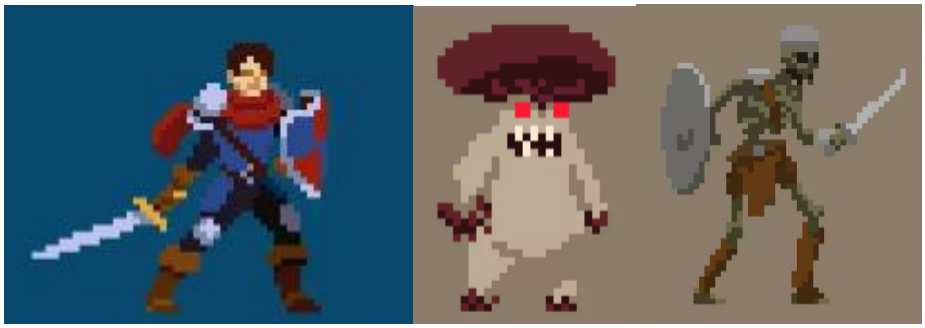
INGENIERÍA DE VIDEOJUEGOS 2022-2023

Universidad
Rey Juan Carlos

ÍNDICE

1. Introducción y explicación del juego.

El juego consiste en un plataformas en la que controlamos a un caballero el cual enfrentará a diferentes enemigos y recogerá monedas por el camino.



El personaje se mueve utilizando las teclas WASD o las flechas de dirección, salta con la barra espaciadora y ataca con la Z.
Tras acabar con el Jefe final, finalizará el juego pasando a una pantalla de enhorabuena donde, pulsando cualquier tecla, se puede volver al menú principal.

2. Reparto de trabajo

Cada uno de los integrantes del grupo hemos participado de manera equitativa en todos los ámbitos de desarrollo del videojuego. Tanto el diseño visual del juego como la implementación y diseño de patrones han sido realizados conjuntamente y tomando decisiones de manera consensuada. Sin embargo, puestos a hacer una división, quedaría de la siguiente manera:

- Iván Gutiérrez Aguilera - diseño de niveles y programación
- Carlos Jiménez López - implementación de patrones y diseño de menús
- Adrián Rubio Garrido - animaciones y diseño de memoria

3. Patrones implementados

3.1 Dirty Flag

En nuestro juego se ha utilizado el patrón Dirty Flag para que ciertas monedas que denominamos especiales, como gemas, por ejemplo, no vuelvan a aparecer tras cada ejecución, si no que una vez recolectadas, desaparecen para siempre.

3.2 FlyWeight

El patrón FlyWeight se emplea para reducir el uso de memoria compartiendo los datos comunes entre los distintos tipos de enemigos. Cabe destacar que se ha implementado por medio de ScriptableObjects.

3.3 Observer

El patrón Observer es utilizado para llevar un recuento de las monedas recogidas por nuestro héroe en el HUD. De esta manera, el personaje será nuestro sujeto, que avisa al Game Manager al coger una moneda, y este, como Observer, actualiza el HUD añadiendo al texto de la interfaz la cantidad que suma cada moneda recogida.

3.4 State

Por último, el patrón State se utiliza como una máquina de estados en la que el jefe final, cambia su velocidad de movimiento en función de su vida restante. El personaje permanece estático mientras su porcentaje de vida sea del 90% o mayor. Una vez tiene menos del 90% de vida comienza a moverse, y cuando su porcentaje de vida sea menor del 30%, aumenta su velocidad de movimiento.

4. Aspectos a destacar

Además de los patrones implementados, cabe destacar el uso de prefabs, que si bien no es la requiere implementación, utiliza el patrón Prototype. También se puede reseñar que el uso de estos patrones ya implementados podría expandirse en una versión más desarrollada del juego, por ejemplo añadiendo más tipos de enemigos, más niveles o más eventos sucedidos en tiempo real**.** Por último reseñar que, el patrón State se encuentra dentro de la carpeta en la que se almacena la implementación del patrón FlyWeght por razones de comodidad ya que ambos comparten el uso de la clase Enemi y por tanto, su interfaz IEnemi**.**