Zombie Attack

# MEMORIA DEL VIDEOJUEGO

Tecnología de los videojuegos
2020-2021

Grupo 1

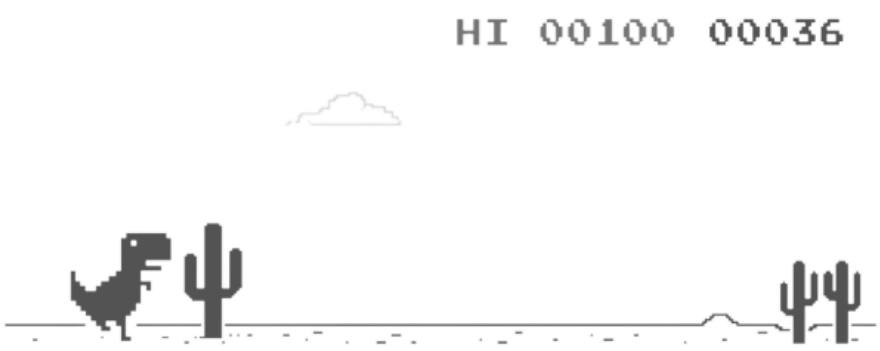
# ÍNDICE

# 1. Introducción

Somos el grupo uno y hemos decidido llevar a cabo la creación de un videojuego de tipo plataformero basándonos en otros videojuegos ya existentes que nos han servido de inspiración. El resultado de este videojuego es fruto del esfuerzo de los 5 miembros del equipo, los cuales nos hemos comprometido y esforzado al máximo en su desarrollo.

## 1.1 Concepto del juego

La propuesta de nuestro videojuego consiste en una mezcla del videojuego Dino T-rex, conocido por aparecer en google chrome cuando no dispones de internet y otro videojuego llamado Risk of Rain 1. El juego consiste en un personaje que trata de salvar su vida mientras se ve atacado por oleadas de zombies.

### 1.2 Características principales

Zombie attack es un juego sencillo que busca entretener a cualquier persona independientemente de su edad. Se trata de un juego con varios niveles los cuales tienen distinto grado de dificultad. Es un juego cuya duración depende exclusivamente de la habilidad del jugador para sobrevivir sin que un zombie lo toque o se caiga al vacío.

## 2. Tecnologías y herramientas de desarrollo

En este apartado vamos a tratar de las herramientas y tecnologías que hemos usado para el desarrollo de nuestro videojuego, estas herramientas son muy básicas pero nos han facilitado el llevar a cabo nuestro trabajo.

### 2.1 Lenguaje de programación

El lenguaje de programación que hemos usado es el que nos han impuesto, que en este caso ha sido python. Aunque no hemos tenido opción de escoger otro lenguaje creemos que es una buena elección pues tiene una gran calidad en su sintaxis, permite la programación orientada a objetos y ofrece un tipado dinámico fuerte.

### 2.2 Plataforma de programación

La plataforma que hemos usado para editar nuestro código ha sido Pycharm, pues nos la recomendó el profesor de la asignatura y creemos que tiene herramientas muy interesantes como la indicación de errores según escribimos el código y alertas de sintaxis incorrecta. Además proporciona arreglos rápidos así como la refactorización de código de forma automática.

Nos ha parecido una plataforma muy útil y sencilla de utilizar, creemos que ha facilitado el desarrollo de nuestro código de forma adecuada y ordenada, por lo que aunque no es obligatorio usar esta plataforma para editar el código si la recomendamos frente a cualquier otra opción.

### 2.3 Librerías

La librería que hemos usado para desarrollar nuestro videojuego es Arcade, pues no había más opciones. Creemos que ha sido una librería útil y sencilla de usar. Es cierto que también hay una librería muy usada para el desarrollo de videojuegos que es Pygame, pero no tiene ciertas características que si tiene arcade. Arcade cubre la mayoría de funcionalidades que no fueron compatibles con Pygame.

Arcade dibuja sprites mucho más rápido que pygame y soporta sprites animados, cosa que hemos agradecido mucho para el desarrollo de nuestro videojuego. Además algo que creemos que es interesante y nos ha ayudado a tener el código más estructurado es que arcade fomenta la separación de la lógica y el código de visualización, cosa que pygame no hace.

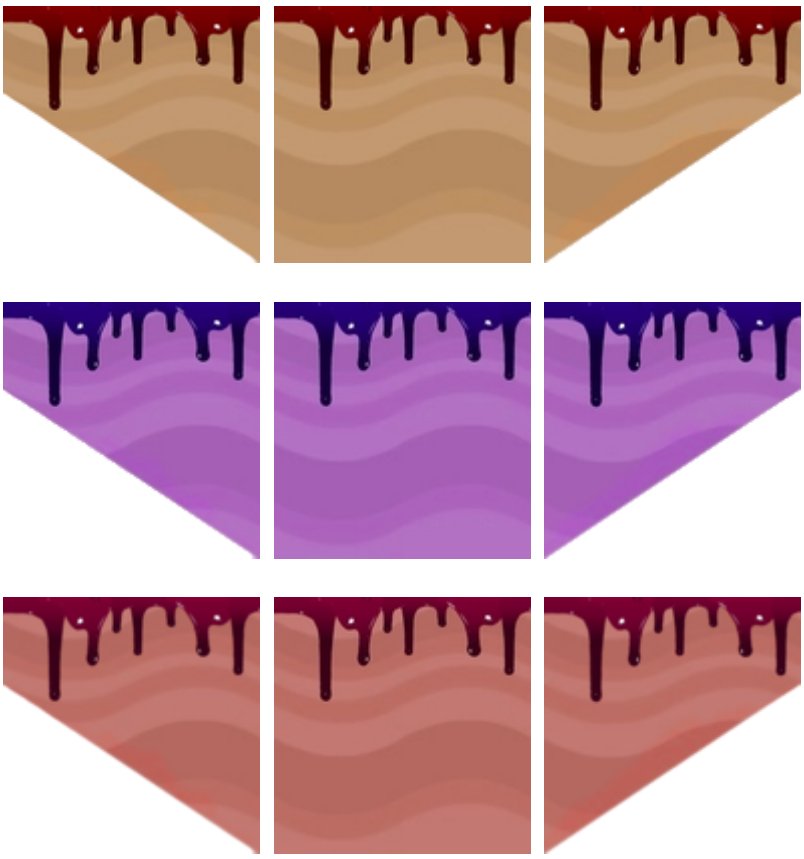
## 2.4 Control de versiones

Para el control de versiones hemos usado Git, ya que es así como lo han establecido los profesores y pensamos que es lo más adecuado pues esta herramienta es un control de versiones distribuido de código abierto muy utilizado en la actualidad en todos los entornos de trabajo a nivel de cooperación, por lo que a parte de ayudarnos a organizarnos nos ha preparado para el futuro.

Hemos usado Github para alojar nuestro código, revisar y descargar las distintas versiones que íbamos lanzando. Github es una plataforma o portal que nos permite gestionar las aplicaciones que utilizan el sistema de Git, por lo que es verdaderamente útil. Todos los miembros del equipo hemos hecho uso de Github, aunque sobre todo ha sido utilizado por el desarrollador y el jefe de proyecto.

# 3. Diseño del videojuego

El diseño del videojuego se ha llevado a cabo previamente mediante unos dibujos a mano alzada en papel para conseguir una idea inicial, después de esto se llevó a cabo el diseño de cada componente de forma digital. A continuación podemos observar los suelos diseñados.



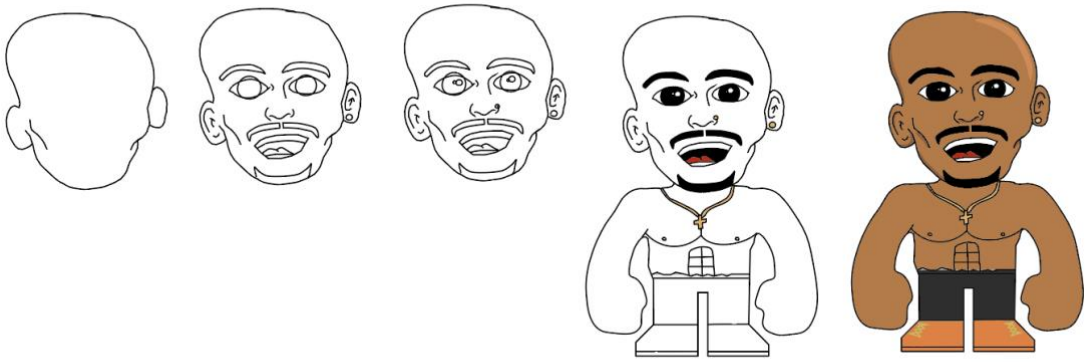
Para el diseño de estos suelos nos hemos basado en algunos existentes que estaban disponibles en arcade, pero los hemos remodelado y añadido algunos detalles para que se adapte al ambiente del ataque de los zombies.

En cuanto a los fondos diseñados también son únicos para los distintos niveles, los podemos ver a continuación.

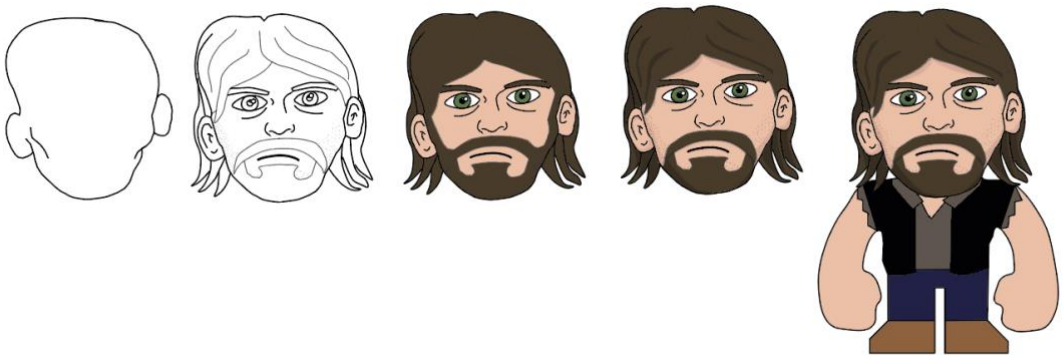
Hemos diseñado tres personajes jugables con sprites animados, de entre los que el jugador debe elegir uno antes de empezar su partida. Únicamente supondrán un cambio visual en la experiencia del jugador.

Proceso de diseño de los tres personajes:

**Personaje 1**



**Personaje 2**



**Personaje 3**

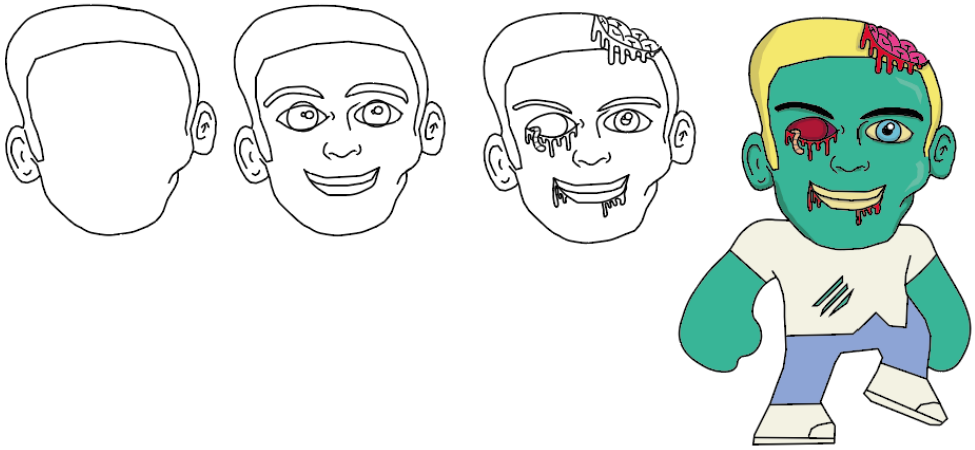
Los personajes también tienen un set de sprites designado para cuando se obtiene el power-up del arma:

Para los enemigos, hemos diseñado tres sprites ,también animados, para introducir algo de variedad visualmente. Los zombies se moverán dentro de las plataformas hasta

encontrarse con un borde u obstáculo, momento en el que se darán la vuelta y se moverán en la dirección opuesta.

**Zombie 1**

**Zombie 2**



**Zombie 3 (Boss)**

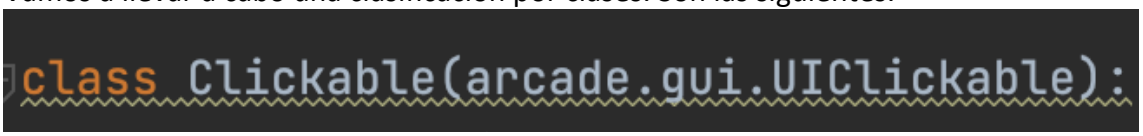
## 4. Implementación

En este apartado vamos a tratar la parte de la implementación, es decir, cómo hemos llevado a cabo el desarrollo del videojuego. Para comenzar lo primero que hemos hecho es declarar aquellas variables globales que vamos a usar en las diferentes partes de nuestro código.

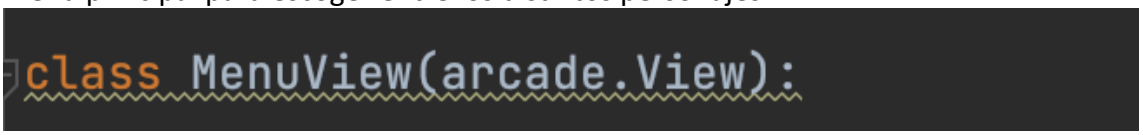
```
SPRITE_SCALING = 0.5
SPRITE_NATIVE_SIZE = 128
SPRITE_SIZE = int(SPRITE_NATIVE_SIZE * SPRITE_SCALING)
TILE_SCALING = 0.5

SCREEN_WIDTH = 800
SCREEN_HEIGHT = 600
SCREEN_TITLE = "Zombie Attack"

# How many pixels to keep as a minimum margin between the character
# and the edge of the screen.
VIEWPORT_MARGIN = 40
RIGHT_MARGIN = 150

# Physics
MOVEMENT_SPEED = 5
UPDATES_PER_FRAME = 7
JUMP_SPEED = 13
GRAVITY = 0.5
BULLET_SPEED = 10

#Texture orientation
RIGHT_FACING = 0
LEFT_FACING = 1

# How many pixels to keep as a minimum margin between the character
# and the edge of the screen.
LEFT_VIEWPORT_MARGIN = 200
RIGHT_VIEWPORT_MARGIN = 400
BOTTOM_VIEWPORT_MARGIN = 150
TOP_VIEWPORT_MARGIN = 100

PLAYER_START_X = 64
PLAYER_START_Y = 225
```

Estas son todas las variables necesarias para que nuestro juego funcione correctamente, a continuación vamos a comentar algunas de las funciones más importantes pero no todas, pues conlleva adjuntar un gran número de imágenes.

Vamos a llevar a cabo una clasificación por clases. Son las siguientes.




La clase Clickable la utilizamos para poder actuar con las flechas que aparecen en el menú principal para escoger entre los distintos personajes.




La clase MenuView es a la que llamamos desde nuestra función main(), porque corresponde al menú de inicio. Desde esta instancia el jugador podrá avanzar al bucle de juego principal.

Dentro de esta clase contamos con varias funciones:

```
def update_texture(self):
```

Usamos esta función para determinar el tamaño de las texturas y de los personajes que vamos a usar en el menú.

```
def on_show(self):
```

En esta función asociamos a las variables cada una de las imágenes que vamos a usar. Por ejemplo en el menú usamos las imágenes de las flechas para escoger los personajes, así como las imágenes de los tres personajes posibles entre los que podemos escoger.

```
def on_draw(self):
```

En esta función dibujamos en pantalla las imágenes que hemos asociado a las variables de la función anterior, como el logo de nuestro videojuego, las flechas y el personaje seleccionado.

```
def on_mouse_press(self, _x, _y, _button, _modifiers):
```

Con esta función nos encargamos de actualizar la vista cada vez que se pulsa alguna de las flechas, para que vaya cambiando el personaje que se muestra en pantalla a medida que vamos haciendo click.

Se llama a la clase GameOverView cuando en el bucle de juego principal se detecta que el jugador ha perdido la partida. Únicamente consiste de una textura que ocupa toda la ventana, la cual reinicia el nivel en el que estaba el jugador cuando se pulse el click.

Consta de las siguientes funciones.

```
def __init__(self, skin, level):
```

Lo primero que hacemos en esta función es cargar la imagen de de game over y setear el nivel al nivel en el que nos encontrábamos antes de perder. También reiniciamos la ventana gráfica.

```
def on_draw(self):
```

Con esta función dibujamos la vista que hemos especificado en la función anterior.

```
def on_mouse_press(self, _x, _y, _button, _modifiers):
```

Por último en esta función simplemente volvemos a iniciar la vista del juego en el nivel en que nos encontrábamos y con el personaje seleccionado.

```
class Level(arcade.View):
```

La clase Level corresponde al bucle principal del juego, donde tiene lugar toda la lógica y renderizado de los elementos de este. Es la función más importante de todo el juego y la más extensa en cuanto a código se refiere. En esta clase se encuentra la función “setup”, que se encarga de construir el nivel en el que el jugador debe entrar.
Las funciones que encontramos en esta clase son las siguientes.

```
def __init__(self, skin_index):
```

En este método se inician las variables necesarias para crear el nivel concreto. Todas estas variables se declaran vacías para posteriormente establecer su valor.

```
def setup(self, nivel):
```

En este método se establecen los valores concretos, como el personaje concreto, las paredes y los suelos, el fondo del juego, los objetos, y los zombies, la música y otras tantas variables.

```
def on_draw(self):
```

Es en este método en el que dibujamos todos los sprites. Todas aquellas variables que ya tienen un valor establecido ahora son mostradas.

```
def on_mouse_press(self, x: float, y: float, button: int,
modifiers: int):
```

Función o método al que llamamos cuando se hace click en cualquier lugar de la vista. Lo único que hacemos en este método es comprobar si el personaje tiene arma o no, si la tiene se llevará a cabo el lanzamiento de la bala, en caso contrario no haremos nada.

```
def on_key_press(self, key, modifiers):
```

Función o método al que se llama cuando se pulsa una tecla. Lo único que hacemos en este método es comprobar si se ha pulsado alguna tecla que nos interese, es decir que la usemos para los controles del juego. Hacemos una comprobación de si esa tecla es la W, la A, o la D. Si es la W llevaremos a cabo el salto, si es la A nos desplazaremos hacia la izquierda, y si es la D hacia la derecha.

```
def on_key_release(self, key, modifiers):
```

Función o método al que llamamos simplemente cuando dejamos de pulsar una tecla, de nuevo hacemos una comprobación de si es la tecla A o D, para detener el movimiento del jugador, no comprobamos si es la W porque el salto tiene un rango limitado de movimiento que finaliza solo por mucho que mantengamos pulsado.

```
def on_update(self, delta_time):
```

Por último este método contiene prácticamente toda la lógica del juego y del movimiento. Hacemos comprobaciones de si el personaje se ha caído del mapa, manejamos el movimiento del personaje también de la vista. Además se actualiza la barra de progreso a medida que nos movemos con el personaje concreto. También comprobamos si la bala que se ha lanzado ha chocado contra un enemigo o no y comprobamos si hemos alcanzado un nuevo nivel para mandar al personaje al mismo.

## 5. Audio

Para el audio del videojuego se han implementado efectos de sonido y música de manera que mejoran la experiencia al jugar.
Los efectos de sonido implementados para las siguientes acciones o eventos:

- Cuando el personaje salta
- Cuando el personaje dispara
- Cuando el personaje sufre daño
- Sonidos de zombies
- Cuando el personaje muere en la pantalla de “game over”

La música que se emplea en los niveles genera un ambiente de acción/supervivencia que encaja con la temática siendo diferente en cada uno de los niveles.

# 6. Problemas encontrados

En este apartado vamos a tratar los problemas que hemos encontrado a la hora de llevar a cabo el desarrollo del videojuego, tanto a nivel de organización como de desarrollo.

## 6.1 Problemas de diseño

En cuanto a los problemas de diseño que hemos encontrado a lo largo del proceso podemos mencionar los más comunes, como es la inexperiencia en herramientas de diseño, como gimp, photoshop illustrator, y otros. Además también hemos experimentado ciertas dificultades a la hora de determinar el tamaño exacto de las imágenes que nos permite python sin que suponga un problema para su visualización.

Nos dimos cuenta de que las imágenes de plataformas y suelos debían tener unas dimensiones de 148x148 píxeles como mucho para ser aceptadas sin problemas. Tuvimos que hacer uso de herramientas que nos permitieran reducir el tamaño de nuestras imágenes, a parte de herramientas que permitieran la eliminación de fondos blancos en las imágenes, para que se respetara el fondo diseñado.

## 6.2 Problemas de desarrollo

En cuanto a problemas de desarrollo podemos mencionar la inexperiencia en programación de la mayoría de los miembros, en concreto del desarrollador, que al estar en primero de carrera no tenía mucha experiencia en ese sentido. Aún así se ha adaptado muy rápido al lenguaje y a sus peculiaridades. Los problemas de desarrollo sobretodo han sido para Miguel, el desarrollador, y Daniel, que aunque era jefe de proyecto se ha encargado de hacer el diseño de todas las plataformas y elementos del videojuego así como de programar la base del videojuego junto con Miguel.

Algunos problemas que hemos encontrado han sido con la herramienta usada para el control de versiones, pues en ocasiones hemos encontrado conflictos al tratar de hacer push y nos hemos cargado parte del código desarrollado por otro compañero, pero ocurrió al inicio únicamente cuando no todos conocíamos esta maravillosa herramienta. Por los demás, los problemas de desarrollo han sido sobretodo solución de errores y bugs que iba encontrando el tester, que finalmente se ha encargado también del diseño de todos los personajes, al no contar con el apoyo de la diseñadora.

## 6.3 Problemas de organización

En este apartado vamos a comentar algunos de los problemas que hemos encontrado en cuanto a organización. Lo primero que debemos comentar es que estos problemas han sido bastante pequeños, pues por norma general los miembros nos hemos comprometido de manera adecuada con nuestras funciones establecidas previamente por el profesor. Decimos por norma general por que si que es cierto que tuvimos una baja
justo antes de comenzar a desarrollar, en concreto de la diseñadora.

La baja de la compañera podemos decir que ha sido el único problema, pues supuso que nos tuviéramos que volver a organizar desde el principio las tareas, y ahora con mayor carga de trabajo, pues debíamos llevar a cabo un trabajo planificado para un número mayor de personas. No obstante no vimos esto como un obstáculo si no como una oportunidad para poder desempeñar más de un papel o rol a la vez.

## 7. Reparto de tareas y cronograma del proyecto

En cuanto al reparto de tareas y cronograma del proyecto lo primero que queremos mostrar es la planificación inicial que fue la que adjuntamos en el ppt de la presentación que hicimos en clase.

| Planificación | |
|---|---|
| Semana 1 (16 abril - 28 abril) | Diseño de los componentes del videojuego |
| Semana 2 (28 abril - 9 mayo) | Programación de las mecánicas básicas y desarrollo |
| Semana 3 (10 - 16 mayo) | Desarrollo de niveles y testing |
| Semana 4 (17 - 23 mayo) | Desarrollo de niveles y testing |
| Semana 5 (24 - 30 mayo) | Presentación videojuego |
| Semana 6 (31 mayo - 6 junio) | Entrega videojuego |

Esta planificación finalmente o se ha llevado a rajatabla por los problemas comentados en el apartado anterior, pero no sirvió de guía para poder presentar una versión muy avanzada el día de la presentación del videojuego y poder tener una primera versión de nuestro juego con varios niveles a parte de un nivel final contra un boss como nos recomendó el profesor.

En cuanto al reparto de tareas, el inicial fue el que podemos observar en la siguiente imagen.

| | |
|---|---|
| Miguel Virtus Rubio | Programador |
| Paula Izquierdo Rodríguez | Diseño |
| Sergio Lorenzo Montiel | Sonido |
| Daniel de Heras Zorita | Web |
| Bryan Sebastián Salguero Pinto | Testing |
| Daniel Martínez Algar | Jefe Proyecto |

No obstante, como ya hemos comentado antes, tuvimos una baja, causada por la diseñadora que dejó la asignatura, por lo que finalmente nos organizamos de la forma que vemos en la siguiente imagen.

| | |
|---|---|
| Miguel Virtus Rubio | Programación |
| Sergio Lorenzo Montiel | Sonido y programación |
| Daniel de Heras Zorita | Web y programación |
| Bryan Sebastián Salguero Pinto | Testing, diseño y programación |
| Daniel Martínez Algar | Jefe Proyecto, diseño y programación |

Cabe destacar que Miguel solo ha programado porque ha sido quien se ha encargado del desarrollo del 90% del código, que es la labor más compleja podríamos decir. En general todos hemos ayudado en aquellas facetas que ha hecho falta y en el momento en el que ha hecho falta, consiguiendo así el resultado de este fantástico videojuego.

## 8. Conclusión y agradecimiento

A modo de conclusión queremos comentar algunas de nuestras opiniones acerca de este trabajo y la asignatura. En primer lugar creemos que ha sido un trabajo muy interesante el que se nos ha pedido llevar a cabo, pues aunque ha supuesto un reto, hemos aprendido mucho. Creemos que nos ha servido de preparación para el día de mañana, cuando en nuestro trabajo tengamos un equipo con el que trabajaremos de la mano y con un objetivo común.

Nos ha permitido darnos cuenta de la complejidad que conlleva también el organizar un equipo y llegar a decisiones comunes sin imponer ciertos gustos o predisposiciones personales. Creemos que el resultado ha estado a la altura de lo pedido, y sin duda que ha sido por una correcta organización y también predisposición de todos los miembros del equipo.

Por último queremos dar unas palabras de agradecimiento a todo el profesorado de la asignatura, que creemos que han desarrollado su papel de manera excepcional y nos han prestado la ayuda necesaria en todo momento, incluso con esta situación complicada en la que no hemos podido tener clases presenciales como nos hubiera gustado. Ha sido un placer trabajar con el profesorado de la asignatura,  con la dinámica de la misma y también con sus contenidos.