UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica

Universitat Politècnica de València

Creación de un videojuego a través de un entorno Unity

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Adrián Uria Bruque

Tutor: Javier Lluch Crespo

2017-2018

Resumen

Este proyecto se ha realizado con el entorno *Unity*. Al ser un videojuego, su principal objetivo es entretener al usuario. Se ha desarrollado un videojuego que se basa en la idea clásica de los juegos de recreativas. Toda la implementación del videojuego ha sido mediante *scripts* los cuales contienen el código fuente que permite crear la mayor parte del funcionamiento de un videojuego. El sistema es clásico y se basa en lograr matar el máximo número de enemigos. Las principales mecánicas del videojuego son el desplazamiento lateral, al ser un videojuego en 2D, la mecánica de salto y la mecánica de disparo. Se ha utilizado el entorno de *Unity* porque contiene una gran cantidad de herramientas que hace más sencillo llevar a cabo un proyecto de estas características. Todo el proyecto está realizado a mano a excepción del audio por lo que se decidió por realizar el proyecto con una estética *Pixel Art.*

Palabras clave: Unity, 2D, videojuego, físicas, diseño, C#, Pixel Art.

Resum

Aquest projecte s'ha realitzat amb l'entorn *Unity.* Com que és un videojoc, el seu principal objectiu és entretenir a l'usuari. S'ha desenvolupat un videojoc que es basa en la idea clàssica dels jocs de recreatives. Tota la implementació del videojoc ha estat mitjançant *scripts* els quals contenen el codi font que permet crear la major part del funcionament d'un videojoc. El sistema és clàssic i es basa en aconseguir matar el màxim nombre d'enemics. Les principals mecàniques del videojoc són el desplaçament lateral, en ser un videojoc en 2D, la mecànica de salt i la mecànica de tir. S'ha utilitzat l'entorn de *Unity* perquè conté una gran quantitat d'eines que fa més senzill dur a terme un projecte d'aquestes característiques. Tot el projecte està realitzat a mà a excepció de l'àudio pel que es va decidir per realitzar el projecte amb una estètica *Pixel Art*.

Paraules clau: Unity, 2D, videojoc, físiques, disseny, C#, Pixel Art.

Abstract

This project has been done with the *Unity* environment. Being a videogame, its main objective is to entertain the user. A videogame has been developed based on the classic idea of recreational games. The whole implementation of the video game has been through scripts which contain the source code that allows to create most of the operation of a video game. The system is classic and is based on killing the maximum number of enemies. The main mechanics of the videogame are the lateral displacement, being a 2D video game, jump mechanics and shooting mechanics. The *Unity* environment has been used because it contains a large number of tools that make it easier to carry out a project of these characteristics. The entire project is done by hand, with the exception of audio, so it was decided to carry out the project with an aesthetic Pixel Art.

Keywords : Unity, 2D, video game, physical, design, C#, Pixel Art.

Agradecimientos

A mi familia por el apoyo recibido.

A Lisa por ayudarme a realizar los diseños y el apoyo con el proyecto.

A Javier, mi tutor, por toda la ayuda aportada y por facilitarme tanto las cosas.

Tabla de contenidos

Tabla de imágenes

1. Introducción

Cada vez más, los videojuegos componen una parte importante en la vida cotidiana de mucha gente y últimamente el espectro de personas que consume los mismos se ha visto aumentado al haber aparecido un mercado como el de los teléfonos móviles. Últimamente, ciertos sectores que no habían estado interesados en estos videojuegos los consumen con más frecuencia. Por todo esto, actualmente es la mejor época de los videojuegos, ya que cada vez se está sacando más partido a estos mercados emergentes. Por lo tanto, una idea de negocio basada en videojuegos, como hacen ver empresas tales como *Microsoft* o *Nintendo*, es una buena opción.

En los últimos años se podía ver como el tema de los deportes electrónicos estaba aumentado considerablemente hasta llegar al punto de que actualmente muchísimas empresas están apostando por ellos creando equipos y apoyándolos.

En este documento se puede ver el proceso que se ha seguido para la creación de un videojuego. Este proyecto nace de poder crear un trabajo genuino, obtener una gran cantidad de conocimiento acerca de la creación de un videojuego y poder ver por todas las etapas que pasa, desde unos bocetos, a como finalmente las ideas se convierten en un videojuego.

Este videojuego se ha desarrollado con uno de los entornos más famosos para la creación de un videojuego, que es *Unity*, en especial *Unity 2D*, ya que el proyecto está creado en dos dimensiones como muchos de los proyectos de móvil que se crean hoy en día.

En el próximo apartado vamos a hablar de los objetivos a tener en cuenta a la hora de desarrollar el proyecto y a explicar un poco como se han llevado a cabo esos objetivos.

2. Objetivos

En el proyecto podemos diferenciar ciertos objetivos dentro del videojuego de los cuales se va a hablar después, pero el objetivo principal, el cual es obvio por varias razones es el de realizar una obra única e interesante para el usuario y por eso, para llegar a ese objetivo se han seguido diferentes pasos de los cuales vamos a hablar a continuación.

El primer objetivo que se tenía que llevar a cabo era el de la creación del diseño de todo el nivel ya que al estar todo hecho a mano se necesitaba bocetar previamente que se iba a realizar. Por falta de herramientas con las que realizar otro tipo de diseño diferente se decidió a realizar un juego estilo *Pixel Art,* lo cual quiere decir que todo ha sido hecho píxel a píxel.

Una vez llevado a cabo la creación de los diseños surge otro objetivo que alcanzar, que es el de diseñar el escenario en *Unity* de la forma más coherente posible, que se pueda jugar de una forma correcta y que sea entretenido para que el usuario se sienta satisfecho.

Como tercer objetivo y con ya las ideas más o menos claras se pasa a desarrollar los personajes y todo el código. Por una parte, está la creación del personaje protagonista que maneja el usuario y por otra parte los enemigos, los cuales se mueven por cuenta propia y siguen sus propias reglas, las cuales se definen dependiendo del juego que se quiera llevar a cabo.

Finalmente, el último objetivo que surge es el de probar y ajustar el videojuego, ya que depende de cómo esté hecho puede resultar ser muy difícil o en su defecto ser demasiado fácil y por lo tanto terminar aburriendo al usuario.

3. Estado del arte

Los videojuegos empezaron a ver su aparición a partir de la década de los 50 y desde su creación hasta hoy en día han pasado de ser un pasatiempo para jóvenes estudiantes de ingeniería a convertirse en la industria del ocio más importante. Desde sus inicios ha habido diferentes periféricos que permiten ejecutar dichos videojuegos tales como los ordenadores y otras videoconsolas como las desarrolladas por *Microsoft*, *Sony* o *Nintendo,* aunque previamente habían videoconsolas recreativas que fueron las pioneras y de ahí aparecieron juegos como el *Pong,* creado por la compañía *Atari,* el cual es uno de los primeros videojuegos que se crearon y el que promovió la ascensión de los mismos. Este juego era muy similar a *Tennis for Two*, pero a diferencia de éste, era utilizado en lugares públicos. Es complicado señalar cual fue el primer videojuego, debido a que incluso a día de hoy todavía no se puede determinar una definición exacta de lo que es un videojuego.

Imagen 1: Máquina recreativa con el juego Pong de ATARI

Desde esa época la industria del videojuego ha ido avanzando hasta llegar al punto de que tenemos una cantidad enorme de contenido y de juegos con gráficos de todo tipo desde *Pixel Art* hasta gráficos hiperrealistas. Además, se puede jugar hoy en día en hardware de todos los tamaños, ya que se puede jugar tanto en móviles como en un ordenador o en alguna videoconsola de sobremesa o portátil. Uno de los juegos más

famosos de la actualidad es *League of Legends,* el cual actualmente tiene su propio deporte electrónico y mueve mucho dinero de parte de las empresas, por otra parte tenemos los juegos de móvil los cuales suelen estar dirigidos para un público más amplio, y son juegos más sencillos y cuyas partidas no duran más de 10 minutos, tales como el *Candy Crush.*

Imagen 2: Juego Candy Crush

Imagen 3: Juego League of Legends

Dicho esto, no existe un solo modo de jugar a videojuegos, ni una edad específica ni un único público al que van dirigidos, ya que hay una gran variedad de géneros que permite que todo el mundo tenga acceso a los mismos.

3.1. Motores gráficos

Hoy en día hay una cantidad enorme de motores gráficos desarrollados para crear videojuegos, pero como en todo mercado, hay unos cuantos que son más populares y despuntan más que los demás. En este documento se va a hablar de dos motores en específico, los más usados por tener una gran versatilidad y potencia.

➤ ***Unreal Engine*.** Creado por la compañía *Epic Games*, se mostró inicialmente en un *shooter* en primera persona, *Unreal,* en 1998. Aunque su desarrollo principal fue para ser un *shooter* en primera persona más adelante se ha usado para una gran variedad de géneros, incluyendo videojuegos de sigilo, lucha, videojuegos de rol multijugador masivos en línea (MMORPG) y otros videojuegos de rol. Al tener su código escrito en

c++, el *Unreal Engine* presenta un alto grado de portabilidad y es una herramienta utilizada actualmente por muchos desarrolladores de videojuegos.

Desde su creación ha habido cuatro versiones del mismo, pero aquí vamos a comentar la última ya que es la que se está usando actualmente. Esta es la versión *Unreal Engine 4,* más conocida como *UE4.* Desde el 2 de marzo de 2015 salió disponible esta versión para todo aquel que lo desee de forma gratuita, al igual que todas las actualizaciones posteriores que se lancen del mismo.

Imagen 4: Juego Hellblade Senua's Sacrifice desarrollado con UE4

➤ ***Unity*.** Creado por la compañía *Unity Technologies*, se lanzó la primera versión en la *Conferencia Mundial de Desarrolladores de Apple* en 2005 e inicialmente fue desarrollado en exclusiva para desarrollar proyectos en la plataforma *Mac.* Desde entonces se han desarrollado diversas versiones, pero las más actual y en la que está desarrollado este proyecto es *Unity 5*. Se ha elegido este software para la creación del videojuego porque es de los más sencillos a la hora de aprender si eres nuevo en la creación de un proyecto de estas características. Además de eso, soporta el lenguaje de programación *c#,* que es de los lenguajes más usados hoy en día.

Esta herramienta compite con *UE4* y otras más, como por ejemplo *Cryengine*, por su gran versatilidad y facilidad a la hora de crear proyectos, y por la gran cantidad de información que se encuentra en las redes sobre la de creación de un videojuego de casi cualquier característica en esta plataforma.

Imagen 5: Videojuego Cuphead desarrollado con Unity 5

4. Herramientas utilizadas

Para la realización de este proyecto se han utilizado una serie de herramientas que se van a detallar a continuación.

➤ ***Unity 2D*.** Como ya se ha visto anteriormente esta es la principal herramienta con la que está desarrollada este proyecto. Esta herramienta tiene una gran cantidad de diferentes elementos que se pueden usar para facilitar el trabajo de programación.

Las más importantes y las que se han usado en este proyecto son la de *Animation,* que es la herramienta que nos permite crear las diferentes animaciones tanto de los personajes como del escenario de una forma sencilla e intuitiva permitiendo controlar las animaciones con una herramienta que se llama *Animator.* Esta es la que nos permite realizar transiciones entre las diferentes animaciones y con unas variables hacer muy fácil el manejo entre unas transiciones y otras.

Otra herramienta que tiene *Unity 2D* y que es de gran utilidad es la de *Inspector,* la cual te permite realizar todo tipo de acciones, tanto poder controlar el tamaño de los diferentes elementos del proyecto, así como su posición, o la de crear un *script* u otras opciones más.

Por último, otras herramientas que son importantes de destacar de *Unity 2D* son las herramientas de *Scene* y *Game*, la de *Scene* te muestra la cámara y ahí es donde se monta la escena del juego que podrás ver y probar en la herramienta de *Game,* la cual muestra el juego y es en la que puedes ir probando los diferentes cambios del mismo.

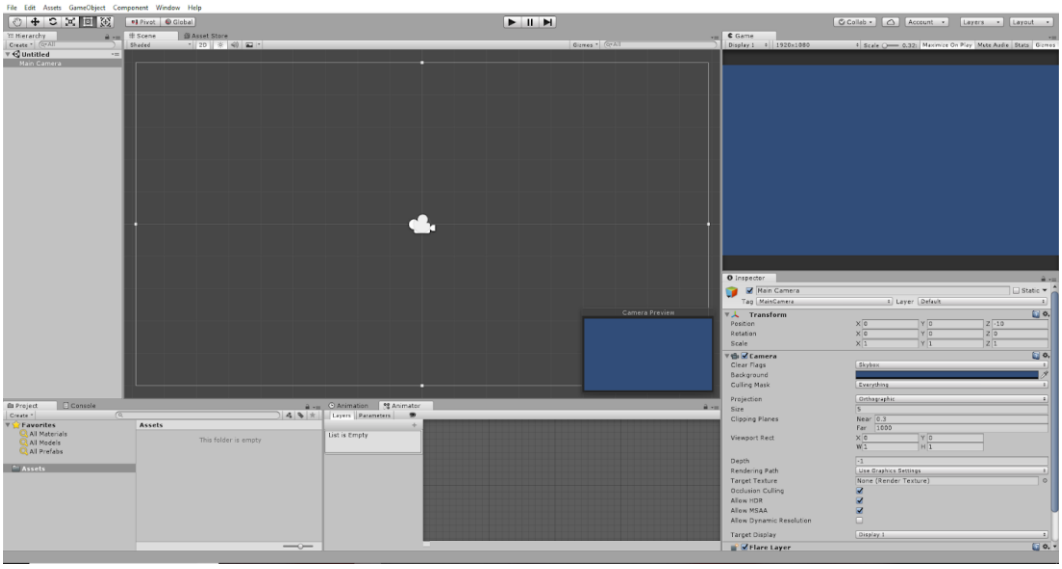

Imagen 6: Proyecto de inicio en Unity 2D

➤ ***Visual Studio.*** Es el entorno de desarrollo integrado que usa *Unity 2D* para la creación de los *scripts,* los cuales son archivos de órdenes que se usan para definir los comportamientos de las diferentes partes del proyecto. También te permite la depuración de código para poder arreglar errores más rápidamente. Como se ha nombrado anteriormente estos archivos de órdenes están escritos en el lenguaje de programación *c#,* el cual es un lenguaje de programación orientado a objetos desarrollado y estandarizado por *Microsoft* como parte de su plataforma *.NET.*

➤ ***Gimp (GNU Image Manipulation Program).*** Se trata de un programa gratuito de manipulación de gráficos en forma de mapa de bits. Forma parte del proyecto *GNU* y tiene una amplia variedad de herramientas. Este programa se ha usado en este proyecto para la realización de los personajes y de los elementos de entorno del videojuego.

5. Desarrollo del videojuego

A continuación, se hablará de cómo se desarrolló la idea del videojuego llamado "***One aleatory guy***".

"*One aleatory guy*" es un videojuego de tipo ***Shoot 'em up*** o también conocido ***matamarcianos***, un juego en el cual el usuario combate a un gran número de enemigos disparándoles mientras esquiva el fuego de estos. El videojuego recibe este nombre ya que en este tipo de juegos es común perder y volver a empezar muchas veces, por lo tanto, el nombre "***One aleatory guy***" hace referencia a que el protagonista es un chico cualquiera sin importancia.

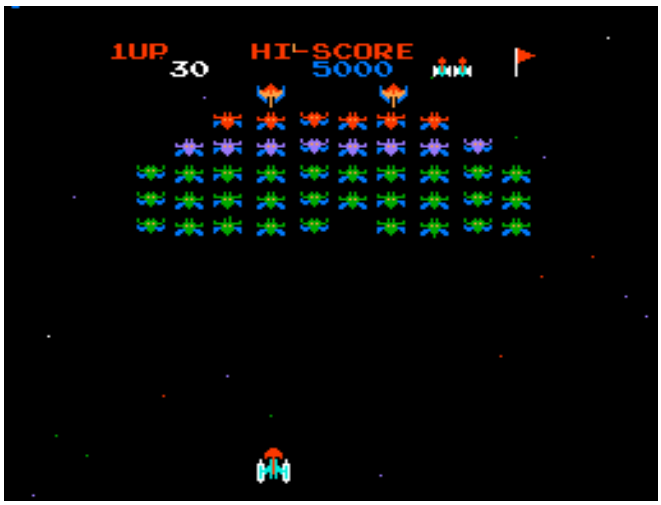

Imagen 7: Galaxian creado en 1979 por la empresa Namco

En el caso de este videojuego, el objetivo es matar al mayor número de moscas posible con un bote de desodorante en spray. El jugador tendrá que manejar al protagonista sobre un barco en movimiento que se encuentra en medio del mar y conseguir matar a todas las moscas que vayan apareciendo mientras esquiva sus disparos y a los enemigos del barco, que serán unas criaturas formadas por los residuos de los disparos de las moscas que hayan caído sobre el barco.

Imagen 8: Captura del juego con todos los enemigos en pantalla

El videojuego comienza con una pantalla principal con dos opciones *Z: Comenzar Juego* y *X: Salir*. Antes de que empiece el videojuego se muestra una animación que explica brevemente en qué situación se encuentra el protagonista y cuál es el objetivo del juego. Una vez termina la animación se abre el videojuego con una ventana de ayuda que explica los controles del juego. Cuando se haya leído la tabla de controles comienza el juego y el jugador tendrá que comenzar a eliminar a las moscas y a esquivar tanto los disparos de los enemigos, como a los monstruos que se irán formando en el barco.

En este videojuego encontramos 3 tipos de enemigos: las moscas normales, las moscas grandes y las criaturas que se forman cuando los residuos de las moscas grandes caen sobre el barco.

El jugador tendrá 3 vidas, es decir, podrá recibir los disparos del enemigo y chocar con ellos 3 veces, estas vidas están indicadas en la parte superior de la pantalla en forma de barra. En caso de que el jugador se caiga del barco perderá sus 3 vidas y tendrá que comenzar desde el principio.

El jugador dispondrá de un disparo cada 3 segundos ya que tiene que recargar el spray cada vez que dispare, para hacer más fácil ver cuando puedes disparar y la vida que tienen se han añadido un indicador de cada uno en la parte superior izquierda de la pantalla.

Los enemigos que elimine se irán sumando a un contador que se mostrará una vez se haya terminado la partida, es decir, cuando el jugador haya perdido sus tres vidas. La decisión de mantener el contador oculto y mostrarlo solo una vez haya terminado la partida se tomó con el objetivo de motivar al jugador a seguir jugando y a intentar batir su récord. Esta es una técnica muy usada en los videojuegos ya que se ha comprobado que, al ocultar este valor, cada vez que se acaba una partida, el comprobar que el resultado tal vez no es el deseado, motiva mucho más al usuario a seguir mejorando su marca personal. Un ejemplo de un videojuego que hace uso de este método es *Cuphead,* el cual, no te muestra la barra de vida del enemigo hasta que pierdes la partida.

Imagen 9: Captura del juego en el momento que acaba una partida

Se decidió crear un videojuego de este tipo ya que es apto para todo tipo de usuarios, es decir, cualquiera puede jugarlo sin que resulte demasiado difícil y tampoco resultará demasiado fácil para los usuarios con más experiencia en el mundo de los videojuegos.

5.1. Evolución del videojuego

Como en cualquier proyecto, la idea principal fue evolucionando con el tiempo. Una de las ideas principales era que el entorno del videojuego fuera un museo de arte, esto cambió ya que este entorno no ofrecía muchas posibilidades de añadir animaciones que añadieran dificultad al juego, es decir, en caso de haber usado dicho entorno el juego habría sido demasiado sencillo y habría resultado aburrido para los usuarios con más experiencia en el mundo de los videojuegos.

Por lo tanto, el entorno pasó de ser un museo a ser un barco situado en medio del océano. Este entorno ofrecía más posibilidades de introducir animaciones y de añadir dificultad al juego, como por ejemplo que el barco esté en movimiento. Además, este entorno resultaba más llamativo y encajaba mejor con la temática del juego.

Una vez estaba decidido el entorno, se realizaron varios modelos de barcos, nubes y cuadrados de agua con diferentes texturas y colores.

Otro cambio que se realizó fue el de los enemigos, en este caso, el cambio también se realizó para añadir dificultad al juego. La idea principal era que los enemigos fueran solo moscas pequeñas y mosquitos, y que ambos realizaran la misma función, disparar. Los mosquitos fueron sustituidos por moscas más grandes, las cuales disparan residuos que al caer sobre el barco se convierten en enemigos. Así, en lugar de tener dos enemigos con la misma función que únicamente se diferencian por su diseño, tenemos 3 enemigos con un diseño único y funciones distintas.

Los enemigos respecto a los primeros bocetos iniciales no recibieron demasiados cambios. Los principales fueron que inicialmente iban a tener seis brazos, pero al realizarlo como un *Pixel Art* quedaba un poco extraño. El tercer enemigo fue diseñado una vez ya el proyecto estaba en marcha ya que pese a ser una idea inicial del proyecto todavía no estaba clara la forma que iba a tener ese enemigo.

Imagen 10: Boceto sobre los enemigos

El título del juego también recibió un cambio, al igual que muchas otras palabras del juego que fueron sustituidas por otras, ya que por el tipo y tamaño de caracteres utilizado era imposible escribir la letra "*M*". Por lo tanto, el titulo paso de ser *"One random guy"* a *"One aleatory guy"* y palabras como "enemigos" fueron sustituidas por otras como en este caso, que fue sustituida por "bichos".

Por último, el personaje principal también experimentó varios cambios. Es este caso los cambios se realizaron para mejorar la estética del mismo. El primer diseño iba a ser un chico con el pelo castaño y una vestimenta común, es decir, un personaje básico, para que cualquier usuario se pueda sentir identificado con él. Esto cambió ya que a pesar de cumplir la función de que los usuarios de identificaran con él, el personaje no resultaba

atractivo al tener un diseño tan simple. Para cambiar esto se realizó un cambio en su vestimenta y en su color de pelo.

También se añadieron varias animaciones una vez el personaje ya tenía su diseño definitivo. Por una parte, se añadió la animación que hace que el personaje parpadee y que le da movimiento a su bufanda cuando está parado, y por otra parte se añadió la animación que hace que el personaje agite el bote de spray para "recargar" el arma.

Respecto a los bocetos iniciales el personaje principal recibió una gran cantidad de cambios ya que las ideas iniciales no acabaron de llamar suficiente la atención y se optó por un diseño mucho más llamativo del mismo.

Imagen 11: Boceto del personaje principal

A continuación, se va a hablar de las diferentes partes que forman el videojuego y de cómo están programadas.

5.1.1. Personaje principal

El personaje principal es un chico con una apariencia bastante llamativa, como se ha nombrado anteriormente, se ha elegido esta apariencia para que el personaje sea atractivo y contraste con el entorno, ya que el entorno está formado por colores neutros.

Los elementos más característicos de este personaje son su pelo y su bufanda. La paleta de colores que se ha elegido para este personaje es muy amplia y prácticamente casi todos los colores contrastan entre ellos además de contrastar con los colores del entorno.

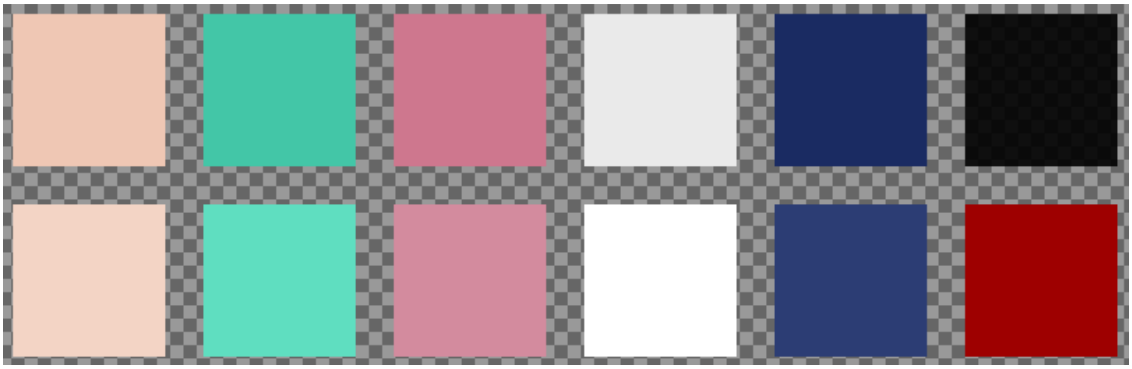

Imagen 12: Paleta de colores usada en el personaje

El personaje principal tiene un total de 6 animaciones diferentes, las cuales están hechas con la herramienta *Animation* de *Unity 2D.* Esta herramienta permite elegir el tiempo que dura la animación y de cuantas imágenes está formada.

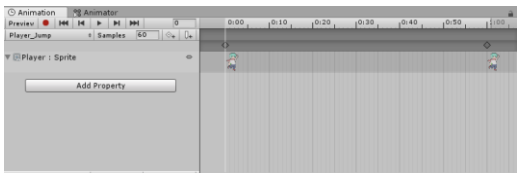

Imagen 13: Animación personaje saltando

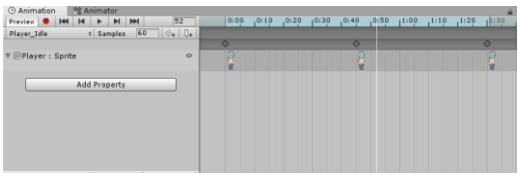

Imagen 14: Animación personaje parado

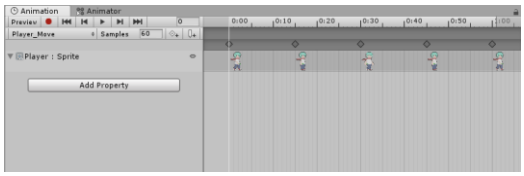

Imagen 15: Animación personaje en movimiento

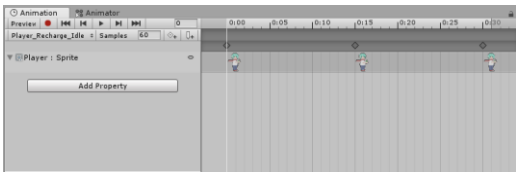

Imagen 16: Animación personaje quieto recargando

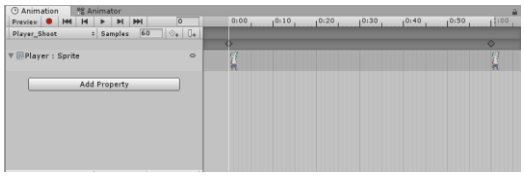

Imagen 17: Animación personaje disparando

Imagen 18: Animación personaje en movimiento recargando

La transición entre animaciones se controla a través de los *scripts* y de la herramienta *Animator,* que es con la cual se pueden enlazar las diferentes transiciones del personaje y permite con unos parámetros cambiar entre las animaciones. En este caso se han creado las variables de *Speed*, *Shoot*, *Grounded*, *Recharge* y *Time Recharge.*

Las variables de *Speed* y *Time Recharge* son del tipo *float* e indican respectivamente cuando cambia el personaje de estado *Player_Idle* a *Player_Move* y viceversa y cuánto tiempo tiene que estar el personaje o bien en la transición *Player_Recharge_Idle* o *Player_Recharge_Move.*

Las demás variables son de tipo *bool* e indican si el personaje pasa a una transición u otra mediante comprobaciones de si algo es cierto o no, por ejemplo, el personaje cambia de *Player_Move* a *Player_Jump* si la variable *Grounded* es falsa y además el personaje tiene una velocidad mayor que cero y en caso contrario cambiaría de *Jump* a *Move* si la variable *Grounded* fuera cierta.

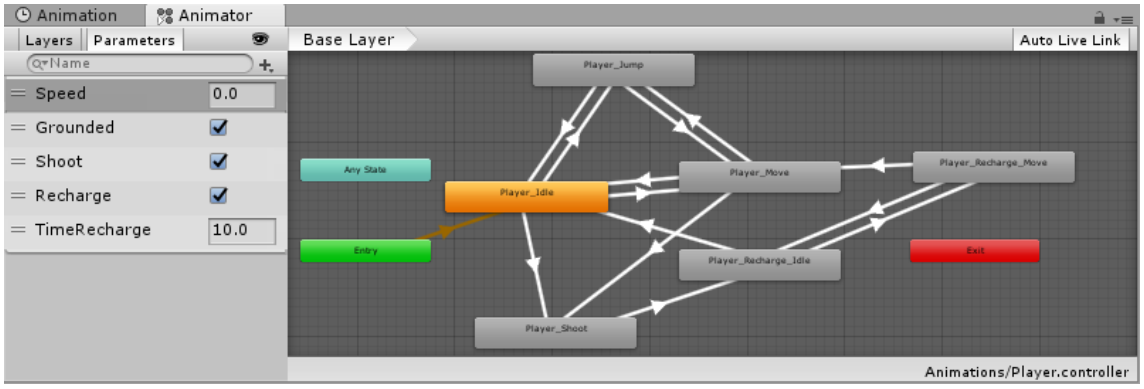

Imagen 19: Estados del personaje principal

Respecto al código y funcionamiento del personaje principal es el que tiene una mayor carga ya que muchos elementos del juego dependen de él para funcionar.

El personaje tiene un *circle collider* y un *box collider,* que son los elementos que se encargan de la colisión con el entorno del personaje. Tiene un total de dos colisionadores ya que en caso de usar únicamente el *box collider* podría ocurrir que se quedara estancado en el escenario ya que al ser el *collider* del escenario también cuadrado tiene esquinas en las cuales se podría quedar estancado, pero a usar un *circle collider* para el movimiento con el suelo se solventa este problema.

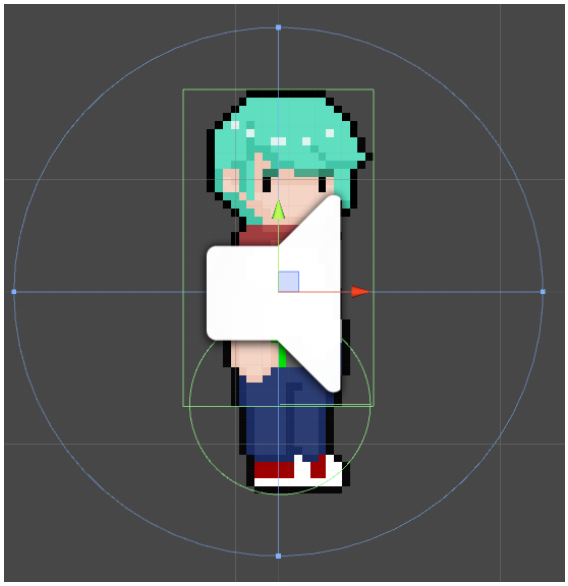

Imagen 20: Personaje con un circle collider y un box collider

Además de estos *collider* el personaje principal cuenta con un *rigidbody* que es el que se encarga de controlar diversos aspectos del personaje tales como la velocidad o la gravedad que actúa sobre el mismo.

Cuenta con un *audio source* el cual se activa cuando se cumplen unas condiciones que estas son que el personaje esté recibiendo daño ya que es un audio que indica que has sido golpeado.

Por último, llegamos al *script* que tiene una gran cantidad de variables ya que con estas podemos calcular y obtener una gran cantidad de funciones. Tiene un total de quince variables públicas las cuales se pueden cambiar desde *Unity* para poder hacer las pruebas más fácilmente. Como todos los elementos del videojuego, los scrips cuentan con un método *Start,* que es el que se ejecuta cuando se inicia la escena en la que se encuentra dicho *script,* y un método *Update*, el cual en algunos casos se ha sustituido por el método *FixedUpdate* ya que el update a veces daba ciertos problemas. Este último método se ejecuta una vez por cada *frame* del videojuego. El funcionamiento de las variables está detallado de la siguiente forma:

➤ ***Max Speed y Speed.*** Estas dos variables son del tipo flotante y son las que se encargan de controlar la velocidad del personaje. Se encuentran en el *FixedUpdate* y se calcula de forma que, si el juego ha comenzado, cosa que se calcula desde un controlador que se encuentra en un código aparte, entonces si el personaje está tocando el suelo se calcula la velocidad. La velocidad se encuentra controlada en el *rigidbody* del personaje y se calcula de forma que la *Max Speed* tiene un valor de 1.5 y la *Speed* de 100. Esto hace que la velocidad de aceleración sea muy elevada, por lo tanto, se consigue limitar la velocidad de forma instantánea, y de esta forma se obtiene una velocidad constante en el personaje que era la idea ya que no tenía ningún sentido añadirle aceleración en este tipo de escenario tan pequeño.

Después de calcular la velocidad el código comprueba si la velocidad que lleva el personaje es positiva y negativa, es decir, si la dirección que lleva es hacia la derecha o a la izquierda y depende de la dirección calcula el *localScale* del personaje, que es el que se encarga del tamaño del personaje o de la dirección que tiene el mismo.

➤ ***Jump Power.*** Con una variable privada de tipo *bool* llamada *Jump* se puede saber si el personaje puede saltar o no ya que se calcula que, si el personaje está tocando el suelo y se presiona la tecla *Z,* en este caso es la tecla de salto, la variable privada *Jump* se activa a *true.* De esta forma se accede al código que calcula el salto del personaje con la variable *Jump Power,* la cual tiene un valor de 10.5 ya que al realizar diferentes pruebas este era el valor que obtenía un mejor salto. Al acabar el cálculo del salto del personaje, el cual es el mismo que el de la velocidad, pero en sentido ascendente, la variable privada se vuelve a poner a *false*, de esta forma no permite que se pueda volver a saltar una vez en el aire.

➤ ***Countdown, Time Recharge, Recharge, Bullet Spawner, Bullet Prefab y Shoot.*** Todas estas variables son las que se encargan del disparo del personaje. La variable *Countdown* tiene un valor de 190 y es la que se encarga de indicar cuánto tiempo tiene que estar el personaje en la animación de recarga al igual que lo hace la variable de *Time Recharge.* La variable R*echarge* tiene la misma función que la variable *Jump* vista anteriormente. Si el personaje puede disparar se pone a *true* y no se puede volver a activar hasta que el *Countdown* llegue a 0. También se recalcula la velocidad cuando se está recargando ya que recargando el personaje tiene una velocidad máxima de 1.25, puesto que recargar le hace ir más lento. Por último, el *Bullet Spawner* es el que cuando se dispara y se instancia el disparo calcula la rotación y la posición de la misma.

La variable *Shoot* es la que le indica al animator que tiene que cambiar a la animación de disparo ya que se activa cuando se dispara y se desactiva cuando ya ha acabado el disparo.

Al disparar se realiza un *Instantiate* del *Bullet Prefab* el cual es un *prefab* que se ha creado de la bala y se instancia en ese instante de forma que se activa y pasa a ejecutarse su código.

➤ ***Grounded.*** Esta variable es la que indica si el personaje está o no, tocando el suelo. No se calcula en el *script* del personaje, sino que se calcula en el script del *circle collider,* ya que el que está en contacto con el suelo es el. En este script se detecta que el personaje está tocando el suelo con las funciones *OnCollisionEnter2D*, *OnCollisionStay2D* y *OnCollisionExit2D,* las cuales detectan cuando está el *collider*

colisionando con otro, en este caso con el del barco, y le dice al script del personaje si está tocando el suelo.

➤ ***Game Controller.*** Esta variable es para poder acceder a las variables públicas del *Game Controller* entre las cuales se encuentra la que indica cuando comienza el juego y de esta forma poder activar el código del personaje.

➤ ***Dead, Lifes y End Controller.*** Estas tres son las que controlan cuando el personaje recibe daño y cuando se acaba el juego. *Dead* es la variable que indica si el personaje está vivo o muerto y en el caso de estar muerto el *script* se encarga de, mediante el *End Controller,* llamar a la función que termina el juego. La variable *Lifes* comienza con un valor de 3 y se disminuye en 1 cada vez que el personaje recibe daño de las moscas, al alcanzar el valor 0 indica que el juego ha terminado.

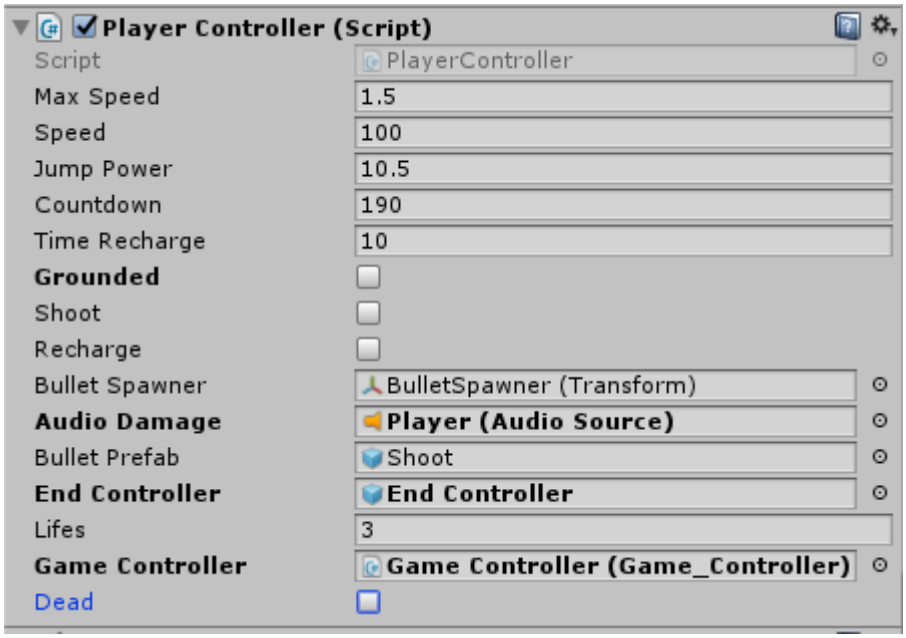

Imagen 21: Script del personaje con las variables públicas

5.1.2. Enemigos

Como hemos nombrado anteriormente, se han creado 3 tipos de enemigos, las moscas pequeñas, las moscas grandes y las criaturas formadas por residuos.

Por una parte, para las moscas se han usado colores oscuros para el cuerpo y colores más llamativos para los ojos. Los tonos elegidos han sido el morado oscuro y el verde, son colores que se suelen utilizar para indicar que algo es venenoso o tóxico.

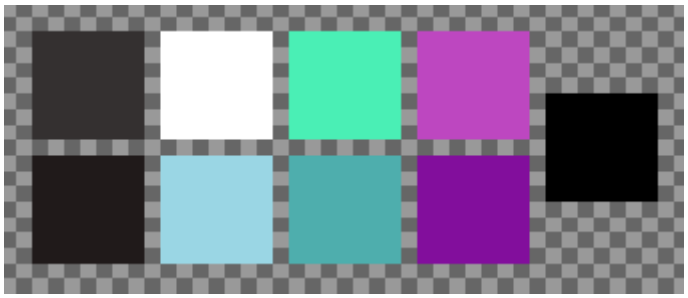

Imagen 22: Paleta de colores usada en las moscas

Los residuos que expulsan las moscas grandes son más grandes que los de las moscas pequeñas, esto no se debe solo al tamaño de estas, si no a que los residuos de las moscas grandes se convierten en otro tipo de enemigo al caer sobre el barco. Para dichos enemigos también se ha usado el color verde, para indicar que se trata de una sustancia tóxica.

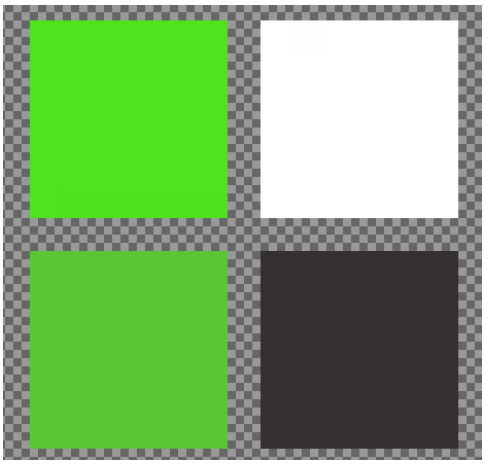

Imagen 23: Paleta de colores de los disparos y el slime

Las moscas cuentan con dos animaciones que son la del movimiento y la animación de muerte y el enemigo que aparece con los residuos de las moscas grandes no cuenta con ninguna ya que es un enemigo muy básico.

Las moscas tienen el *tag* de *Enemy* que sirve para poder controlar las colisiones entre ellas ya que inicialmente las moscas colisionaban entre ellas, pero las colisiones podían dar errores y se eliminaron las colisiones entre ellos. Para que los enemigos no se fueran hasta fuera de la pantalla hay dos *box collider* en cada esquina de la pantalla que al colisionar con el *box collider* de las moscas cambian en sentido ya que calculan que la velocidad ha bajado y cambian de sentido.

El código que calcula el comportamiento de las moscas y el *slime* no es tan complejo como el del personaje principal pero también tiene sus propias peculiaridades.
Para empezar los enemigos están creados como *prefabs* y se generan con unos objetos llamados *Enemy Generator* que generan enemigos cada 15 segundos y cada 2 enemigos normales genera un enemigo grande.

Tenemos tres códigos diferentes, uno por cada enemigo, pero el cálculo de los dos tipos de moscas es muy parecido, con la principal diferencia de que las moscas grandes también generan los enemigos.

Todos los enemigos tienen un *rigidbody* y un *box collider* pero las moscas en su *rigidbody* tienen el valor de gravedad a 0 porque si no caerían y no daría la impresión de que están volando si no que se caerían por la fuerza de la gravedad. El *slime* tiene una gravedad mucho mayor por el hecho de que se quiere que caigan a una velocidad elevada.

Los códigos de las moscas tienen un total de 4 y 5 variables públicas, teniendo una más la mosca grande ya que se usa para la generación del enemigo extra. Este código de las moscas pequeñas funciona de la siguiente manera:

➤ ***Limit.*** Es una variable de tipo flotante. Indica cuanto va a disminuir la altura la mosca antes de moverse de derecha a izquierda. El movimiento es el mismo que el del

personaje principal, con una diferencia primordial, que es que la velocidad que tiene cada mosca es un valor aleatorio entre 0.75 y 1.25 calculado al crearse cada una. Esto hace que sea más difícil prever el comportamiento de la misma ya que un mínimo cambio en la velocidad entre diferentes personajes puede hacerte fallar mucho más de lo debido.

➤ ***Bullet Prefab y Bullet Spawner.*** Son los que se encargan de instanciar el disparo, su funcionamiento es exactamente igual que el del disparo del personaje, la única diferencia es que en este caso también las moscas disparan de forma aleatoria con un valor calculado entre 2 y 4 segundos por el mismo hecho de antes, que el disparo de los enemigos sea lo más imprevisible posible.

➤ ***Die.*** Como en el personaje principal se usa para saber cuándo el enemigo está vivo o muerto y de esta forma cambiar de animación y hacerlo caer en dirección al suelo y en el caso de que colisione con el barco o con el agua destruir el objeto.

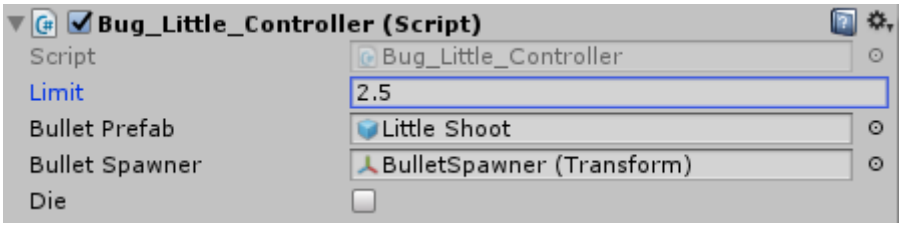

Imagen 24: Script de la mosca pequeña con las variables públicas

A continuación, tenemos las moscas más grandes las cuales tienen el mismo código a la hora de su uso, pero sus disparos tienen un código extra que es que si el disparo colisiona con el barco deja un residuo que a los 2 segundos se convierte en el *slime*, que es el tercer enemigo del juego. Este enemigo tiene su propio código que pese a tener menos carga que los otros sigue siendo parte del proyecto.

El único funcionamiento que tiene el enemigo este es el de moverse a una velocidad aleatoria anteriormente calculada como con los otros enemigos, moverse hacia una posición también aleatoria y en caso de chocar con el personaje destruirse y hacerle

daño. En caso de no chocarse con el personaje se caería del barco y cuando tocara el agua se destruiría de misma manera.

Imagen 25: Enemigos del videojuego

Con todo esto ya se han abordado los personajes que contiene el videojuego y a continuación se verá todo el contenido de las escenas que forman este videojuego.

5.1.3. Las escenas

El videojuego está formado por 3 escenas las cuales definen y contienen todo lo posible para que el proyecto funcione. Se van a ver las escenas en el orden en el que aparecen en el videojuego y con el nombre que tiene cada una.

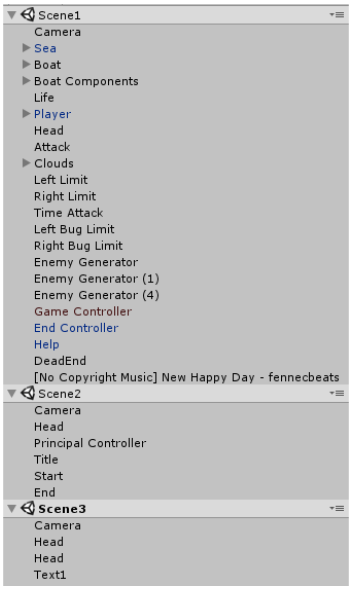

Imagen 26: Escenas del videojuego

La primera escena que tenemos es la que se llama *Scene2* que es en la que aparece la imagen de inicio de juego con una música de fondo. Esta escena tiene la cámara como todas las escenas, la cual está pensada para contener una imagen de una calidad de 1920x1080. Luego nos encontramos con una imagen estática de la cabeza del personaje, un título y un texto de comenzar y salir.

En esta escena encontramos también el controlador de la misma que se llama *Principal Controller,* que es el que se encarga de detectar que si se pulsa la tecla *Z* el juego pasa a la *Scene3* y en caso de apretar la tecla *X* finaliza y cierra el juego.

Imagen 27: Escena de inicio del juego

La siguiente escena es la *Scene3* en la cual se encuentra la misma cámara que en las otras escenas como bien ya se había dicho. También se encuentra la cabeza del personaje principal arriba de color negro para no poder reconocer con quien está hablando el personaje y abajo la cabeza normal ya que es el personaje.

En esta escena podemos ver como el personaje está hablando con alguien acerca de matar unos bichos con un arma y pone un poco en la situación en la que se encuentra el personaje.

Para explicar esta historia esta escena contiene unos textos que van cambiando cada vez que se presiona la tecla *Z.* Esto se hace instanciando cada texto por cada vez que se presiona dicha tecla y de paso destruyendo el texto anterior. Al llegar el último texto el videojuego nos lleva a la última escena, la *Scene1*, la cual nos muestra ya el juego.

Imagen 28: Segunda escena en la que se muestra la historia

En esta última escena es donde se encuentra todo lo que contiene el videojuego. Las partes más importantes de esta escena son el barco, las nubes y el mar. Todos ellos están animados y tienen una función en el videojuego.

➤ Por una parte, el barco tiene la función de plataforma, por la cual se moverá el personaje principal y las criaturas enemigas. Este se moverá de izquierda a derecha durante toda la partida.

➤ Las nubes se encontrarán en la parte superior de la pantalla y además de su función estética, cumplen la función de cubrir la parte por la cual aparecen los enemigos. Estas están animadas de forma que se mueven de izquierda a derecha para simular el viento.

➤ Por otra parte, el océano está formado por diferentes capas, una más oscura y otra más clara en la superficie. Este cumple la función de añadir dificultad al videojuego, ya que es la zona a la cual el personaje no puede acceder, si este cae al océano, perderá sus 3 vidas y tendrá que empezar otra partida. Este también está animado, se mueve en 4 direcciones distintas: de forma ascendente y descendente y de izquierda a derecha. Con esta animación se pretende representar el movimiento de las olas.

Una vez ya vistos los elementos con una mayor importancia en esta escena no nos podemos olvidar que también hay elementos que no parecen tener tanta importancia pero que muestran una información muy importante del videojuego. A continuación, se van a ver esos elementos.

➤ ***Help*.** Es el texto de ayuda que aparece al inicio del juego y que muestra los controles principales del mismo. Las otras mecánicas como la de saltar a los enemigos que salen en el barco, no poder disparar cuando salta el personaje, son mecánicas que se añaden al mismo y se espera que el jugador las vaya descubriendo conforme va jugando hasta entender la totalidad del videojuego. Este cuadro de ayuda se controla de forma que cuando el valor de la variable estado del videojuego del *Game Controller* sea *Idle* se muestre y cuando sea *Playing* se destruya el objeto y deje paso al juego.

➤ ***Left Limit, Right limit, Left Bug Limit y Right Bug Limit*.** De los límites de las moscas ya se había hablado anteriormente pero básicamente son los límites que definen el juego y no permiten ni que los bichos, ni que el personaje principal se salga del juego. Además, se creó un material para las paredes del jugador que tiene la propiedad *Deslizante,* la cual se ha creado de forma que reduciendo la fricción del personaje con el límite a 0 tanto con el barco, y de esta forma no se queda pegado al límite ya que la fuerza de la velocidad que lleva el personaje puede hacer que esto ocurra.

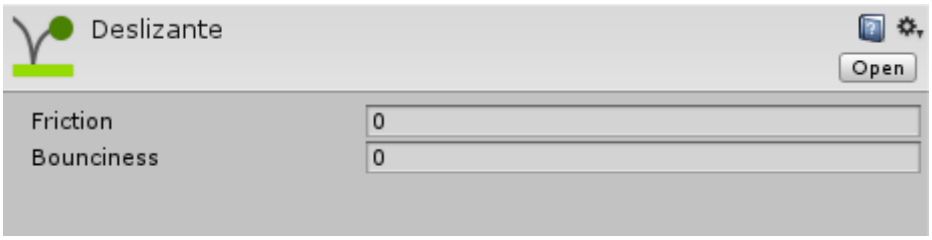

Imagen 29: Material deslizante

➤ ***Attack, Head, Life y Time Attack.*** Estos son los que forman parte del *hud* que muestra el estado del personaje. Tiene por una parte el objeto *Head* que es un icono de la cabeza del personaje, únicamente estético y para hacer notar que la información tiene que ver con él. Luego encontramos el objeto *Life* el cual incluso tiene sus propias animaciones las cuales son el mismo contador de vida, pero con menos vidas y de un color diferente. El *script* de este objeto coge la información de la variable *Life* del jugador, la usa para poder cambiar la animación conforme el personaje va perdiendo vida y de esta forma tener controlada la vida del jugador en todo momento.

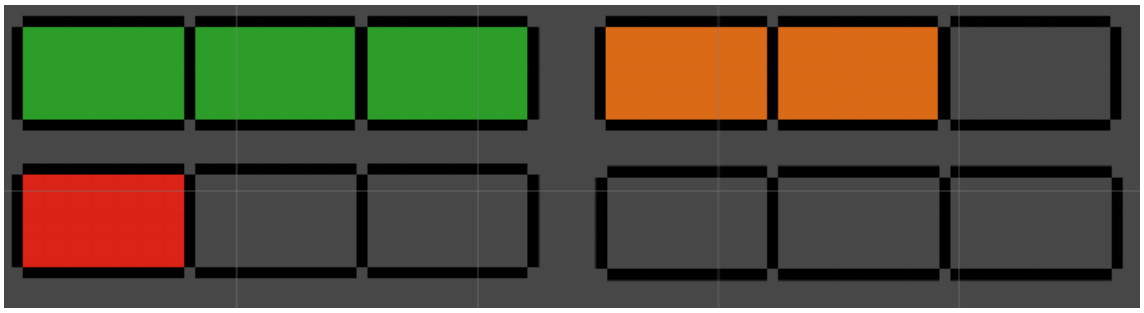

Imagen 30: Vida del personaje principal

Por otra parte, encontramos el objeto *Attack* que únicamente es decorativo y para añadir mayor información al cuadro llamado *Time Attack.* Este es el que muestra el tiempo que se queda el personaje recargando y cuando puede el personaje volver a disparar de nuevo de una forma muy sencilla, que es mediante un círculo que se va llenando con el tiempo.

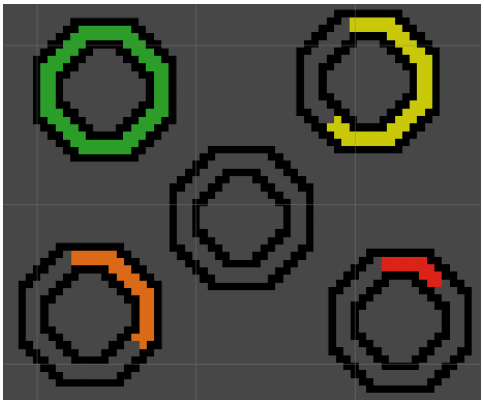

Imagen 31: Recarga del disparo

➤ ***End Controller y Dead End.*** Estos son los que se encargan de cuando el juego ha acabado y de cuando mueren los enemigos. El *Dead End* es un *box collider* que se encuentra en el agua y que detecta cuando un elemento colisiona contra él y se destruyen. Además, detecta siempre que un enemigo choca contra él, los únicos que tienen el *tag* de enemigos son las moscas, y cuenta en el contador de enemigos muertos uno más, cosa que también se hace cuando chocan con el barco.

El *End Controller* es al *script* que llama el jugador cuando se acaba el juego y es también el que lleva la cuenta de los enemigos que se han muerto en la partida. También tiene la función de reiniciar la partida y volver a cargar la escena o de cerrar el juego.

Otra funcionalidad extra y que no está indicada en el juego es que si presionas la tecla *Esc* en mitad de una partida el juego se cierra, por si el jugador no quiere tener que acabar una partida para cerrar o tiene problemas con el juego.

Imagen 32: La escena 1 antes de comenzar

6. Pruebas del videojuego

Una vez el videojuego estaba acabado, comenzó la etapa de test, en la cual se comprueba que todo funciona adecuadamente, que el manejo del personaje principal sea fácil y que el videojuego no es demasiado fácil ni demasiado difícil. Durante esta etapa de test se realizaron varios cambios que se nombran a continuación:

➤ El primer cambio que se realizó estaba relacionado con la cantidad de moscas que iban a aparecer en pantalla, ya que al realizar el test se determinó que había muy pocos enemigos y que por lo tanto el juego era demasiado sencillo.

Inicialmente solo aparecía un enemigo, para hacerlo más complicado se decidió que los enemigos aparecieran en grupos de 5. Este cambio resulto sobrepasar el nivel de dificultad que se le quería dar al videojuego, por lo tanto, se siguieron realizando cambios relacionados con la cantidad de enemigos que iban a aparecer y cada cuanto tiempo iban a aparecer, como, por ejemplo, 3 enemigos a tiempos distintos. Finalmente se decidió que aparecieran 3 enemigos en tiempos distintos en el periodo de 15 segundos.

➤ También hubo dudas con los enemigos formados por los residuos de las moscas, inicialmente iban a formar parte del videojuego, pero se planteó retirarlos ya que hacían que el videojuego fuera mucho más difícil. Finalmente se conservaron reduciendo la cadencia de disparo de las moscas, la cual disminuyó para que no se acumularan tantos enemigos en el barco. Se tomó la decisión de conservarlos ya que en caso de retirarlos la acción de “Salto” del personaje principal carecería de sentido.

➤ Por último, también se realizaron cambios en la distancia de salto del personaje principal, la cual se aumentó ya que al ser demasiado corta no se podían esquivar a los enemigos del barco.

Una vez el videojuego estaba terminado, se inició un periodo de pruebas en el cual 5 personas probaron el videojuego, dos de ellas eran usuarios con experiencia en el mundo de los videojuegos y tres de ellas no. El objetivo de dichas pruebas era averiguar si el videojuego es entretenido para los usuarios, si el nivel de dificultad está bien establecido y si el diseño resulta atractivo.
Una vez terminó el periodo de prueba se realizó una encuesta a los usuarios. De dicha encuesta se sacaron las siguientes conclusiones:

➤ Para los usuarios con experiencia el videojuego tenía una buena dificultad, en cambio, para los otros usuarios resultaba difícil al llegar a cierto punto del videojuego, además, dichos usuarios coincidieron en que les resultaba difícil recordar los controles del videojuego ya que solo se muestran al principio.

➤ Todos los usuarios coincidieron en que el diseño del entorno y de los personajes era atractivo y que la paleta de colores utilizada era una buena opción ya que por el contraste establecido entre todos los elementos es fácil diferenciarlos.

➤ También se les preguntó a los usuarios cuánto tiempo dedicarían al juego por sesión. Más del 50 por ciento respondió que dedicaría más de 10 minutos. Además, todos coincidieron en el que el videojuego era adictivo y que motivaba a seguir aumentando la puntuación.

Imagen 33: Gráfica de tiempo dedicado al videojuego

➤ Dos de los usuarios que lo probaron propusieron que el personaje principal pudiera saltar y disparar al mismo tiempo, ya que esto haría que el videojuego fuera algo más sencillo.

7. Conclusión

En este proyecto se ha utilizado la herramienta de *Unity 2D,* ya que es de las herramientas más sencillas a la hora de desarrollar un proyecto de estas características y se puede ver que la decisión de usar esta herramienta era bastante acertada ya que la cantidad de información al respecto y la facilidad de uso de la plataforma ha hecho posible que se llevará a cabo este proyecto. Al usar también un lenguaje de programación que es *c#,* que actualmente es de los más usados y bastante sencillo de aprender el programador ha podido llevar a cabo la tarea de creación de *scripting* de una forma bastante amena y constante.

El aprendizaje de este lenguaje ha resultado ser bastante sencillo para el programador ya que a lo largo de sus estudios ha utilizado lenguajes de programación muy similares y una gran cantidad de lenguajes de programación orientados a objetos.

Respecto a los objetivos detallados previamente se han podido llevar a cabo todos ellos en mayor medida. El objetivo del que se habló al principio de dicho punto de crear una obra genuina e interesante se puede decir que se ha cumplido ya que se ha creado algo propio al usar unos personajes y entornos propios. También la jugabilidad del mismo ha sido adecuada para los usuarios y entretenido a la par que desafiante. Se puede decir que es un juego con un gran potencial que explota lo bien que puede sus mecánicas y sus diseños.

El primer objetivo como tal que se marcó fue el de crear y diseñar el videojuego y pese a que hubo bastantes dificultades a la hora de crear todo el conjunto y se cambió la idea del mismo varias veces, se llegó a un resultado satisfactorio que permite ver a simple vista toda la información en pantalla y entender con gran facilidad la jugabilidad del mismo.

El objetivo de desarrollar el código y todo el escenario acabó mejor de lo esperado ya que se consiguió realizar todo lo esperado y además se consiguió crear los enemigos que aparecen por el suelo y además de crear una pequeña historia para el mismo, como una pantalla de inicio, las cuales no estaban en la versión inicial del juego y se fueron añadiendo para ampliar la experiencia de juego y mejorar las sensaciones del mismo.

Como último objetivo que se tuvo fue el de añadir pruebas al videojuego tanto por parte del programador como por parte de los usuarios y gracias a estos test fue posible obtener una realimentación que pudo determinar los puntos fuertes y los puntos flojos del videojuego. Gracias a las pruebas del programador se pudo conseguir que se nivelara el videojuego y se pudo controlar la dificultad del mismo.

Para concluir se puede decir que este proyecto ha servido para la creación de algo único y el aprendizaje de un lenguaje nuevo y un programa como es *Unity*. Se podría decir que el balance final del proyecto es muy positivo ya que se han conseguido llegar a cabo todos los objetivos, pero se habría podido ampliar más el proyecto y no se descarta en un futuro añadir diferentes mejoras al mismo.

8. Trabajo futuro

Este proyecto podría seguir desarrollándose en un futuro, aplicando las siguientes mejoras y cambios:

➤ El primer cambio que se realizaría sería añadir más pantallas o niveles. Se crearían 3 niveles distintos, cada uno más difícil que el anterior para aumentar así la atención del usuario. Esto daría la oportunidad de crear nuevos escenarios y de añadir más enemigos distintos, es decir, que cada nivel tendría sus propios enemigos y en el último nivel se podría añadir un "jefe final".

➤ Al desarrollar más el videojuego y hacerlo más largo gracias a los distintos niveles, también se obtendría la posibilidad de desarrollar más la historia del personaje principal y del videojuego en general.

➤ Otra mejora que sería interesante aplicar sería añadir un ranking global y un ranking personal. Es decir, por una parte, el usuario podrá acceder a una tabla con todas sus puntuaciones para poder batir su propio récord, y por otra parte se irían registrando las puntuaciones máximas obtenidas por cada usuario a nivel global.

➤ Por último, la posibilidad de exportar el juego para Android. Gracias a ello se podrían añadir unos controles más descriptivos al videojuego para facilitar el manejo del personaje y solucionar así el problema que tenían los usuarios con menos experiencia para recordar los controles. Además, así el videojuego tendría mayor alcance y se podrían obtener un mayor número de valoraciones que servirán para realizar futuros cambios y mejoras.

9. Referencia bibliográfica

[1] Wikipedia - Videojuego.
https://es.wikipedia.org/wiki/Videojuego

[2] Unreal Engine.
https://www.unrealengine.com/en-US/what-is-unreal-engine-4

[3] Palazuelos, F. (2015). Qué son los motores gráficos y cuáles son los más populares.
https://blogthinkbig.com/motores-graficos

[4] Unity 2D.
https://unity.com/solutions/2d

[5] Wikipedia - Unreal Engine.
https://es.wikipedia.org/wiki/Unreal_Engine

[6] Wikipedia - Unity (motor de juego).
https://es.wikipedia.org/wiki/Unity_(motor_de_juego)

[7] Wikipedia - Script.
https://es.wikipedia.org/wiki/Script

[8] Wikipedia - Microsoft Visual Studio.
https://es.wikipedia.org/wiki/Microsoft_Visual_Studio

[9] Wikipedia - C Sharp.
https://es.wikipedia.org/wiki/C_Sharp

[10] Wikipedia - Gimp.
https://es.wikipedia.org/wiki/GIMP

[11] Wikipedia - Shoot'em up.
https://es.wikipedia.org/wiki/Shoot_%27em_up

[12] Wikipedia - Galaxian.
https://es.wikipedia.org/wiki/Galaxian

[13] Belli, S., López, C. (2008). Breve historia de los videojuegos.
https://dialnet.unirioja.es/descarga/articulo/2736172.pdf

[14] Youtube - Tutorial de Unity 2D Parte 6 (Programando el Disparo).

https://www.youtube.com/watch?v=wjGImRSG29k

[15] Youtube - Tu Primer Videojuego 2D - Curso Unity 5 Completo (Windows, Android y WebGL).

https://www.youtube.com/watch?v=sT5sBkkmuaQ

10. Anexo

Tanto el videojuego como el proyecto se encuentran en Drive y a continuación se proporcionan los enlaces a los mismos.

➤ Videojuego:
https://drive.google.com/drive/folders/10cV1HHa8u2n7IyI8vN9_mN08sJYIf7xQ?usp=sharing

➤ Proyecto:
https://drive.google.com/drive/folders/17NFgcH2RAesXi8MzzRbg5Y64uODA5T-S?usp=sharing