Cañonero - Videojuego

Ignacio Ordorica Cagigas
Grado en Ingeniería Informática
TFG - Videojuegos

Nombre Profesor/a responsable de la asignatura:
Joan Arnedo Moreno y Joel Servitja Feu

07/01/2018

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Cañonero (Arcade Game)*
Nombre del autor:	*Ignacio Ordorica*
Nombre del consultor/a:	*Joan Arnedo Moreno*
Nombre del PRA:	*Joel Servitja Feu*
Fecha de entrega (mm/aaaa):	01/2018
Titulación::	*Grado en Ingeniería Informática*
Área del Trabajo Final:	*TFG - Videojuegos*
Idioma del trabajo:	***Español***
Palabras clave	*Arcade, Turnos, Retro*

Resumen del Trabajo (máximo 250 palabras): *Con la finalidad, contexto de aplicación, metodología, resultados i conclusiones del trabajo.*

El objetivo de este trabajo de fin de grado es crear un videojuego desde cero hasta conseguir un producto completamente jugable. Como el proyecto debe durar apenas 4 meses, el juego es un desarrollo en 2D con aspecto retro, con el fin de poder abarcar todas las facetas creativas. Por tanto, el juego contara con variedad de sprites, varias mecánicas jugables, diferentes sonidos y alta rejugabilidad.

Para este desarrollo hemos seguido un modelo en cascada, estableciendo las fechas de inicio y fin de cada parte, ya que gran parte de ellas debían estar listas antes de comenzar la siguiente, como se puede observar en el diagrama de Gantt creado.

El resultado final ha sido satisfactorio, ya que hemos podido implementar mas del 95% de los hitos marcados. La mayoría de ellos se han realizado en el tiempo estipulado. Las mecánicas generales pensadas al inicio del proyecto han funcionado bien, a pesar de tener que rehacer alguna ya que una vez implementada no funcionaba a nivel jugable.

Terminar el proyecto en el tiempo establecido ha sido un desarrollo duro y difícil, ya que utilizar una tecnología nueva y una fase de preparación pequeña ha resultado en tiempos mas largos a lo planificado o re diseños de partes completas, aunque nos ha servido para dar más importancia a determinadas fases en desarrollos futuros.

Abstract (in English, 250 words or less):

The main goal of this project is to create a videogame from scratch until a completely playable product is obtained. As the project should last only around 4 months, the game is a 2D development with a retro look, in order to cover all the creative facets. Therefore, the game will feature a variety of sprites, several playable mechanics, different sounds and high replay value.

For this development we have followed a cascade model, establishing the start and end dates of each part, since a large part of them had to be ready before starting the next, as can be seen in the created Gantt chart.

The final result has been satisfactory, since we have been able to implement more than 95% of the marked milestones. Most of them have been made in the stipulated time. The general mechanics thought at the beginning of the project have worked well, despite having to redo some since once implemented it did not work at a playable level.

Finishing the project in the established time has been hard and difficult, since using a new technology and a small preparation phase has resulted in longer times than planned or redesigns of complete parts, although it has served to give more importance to certain phases in future developments.

Índice

1. Introducción

1.1 Contexto y justificación del Trabajo

Los videojuegos son cada vez mas relevantes, tanto a nivel de usuarios y ventas como a nivel de impacto social, como demuestra el incremento de las películas y libros basados en videojuegos además de programas de televisión especializados o e-sports.

Además, el proceso de creación de un videojuego incluye todas las fases por las que cualquier aplicación informática debe pasar (Planificación, Desarrollo, Testing, Publicación) además de tener un componente creativo y técnico importante, debido a la necesidad de utilizar material audiovisual y tener muy en cuenta la usabilidad.

De todo esto surgió la idea de crear un videojuego como TFG en el que utilizar todo lo aprendido durante la carrera además de aportar una parte personal al desarrollo.

1.2 Objetivos del Trabajo

- Realizar una planificación completa para la creación de un videojuego desde su inicio hasta su publicación.
- Adquirir el conocimiento necesario en una herramienta de desarrollo real como Unity para poder afrontar proyectos de videojuegos completo.
- Crear un videojuego con una alta rejugabilidad gracias a la generación aleatoria de escenarios y la inclusión de múltiples temas (enemigos, objetos, etc).
- Publicar en Google Play y así conocer de primera mano los requisitos necesarios.
- Adquirir conocimiento y soltura en la modificación de imágenes 2D y Audios sencillos..
- Diseñar un juego que pueda ser jugado en cualquier momento y por personas de todas las edades, debido a la implementación de «juego por turnos» en el que puedes pensar cada jugada con tranquilidad.

1.3 Enfoque y método seguido

Mi ilusión para este proyecto era crear un producto que bebiera directamente de varios de mis videojuegos favoritos. Quería una jugabilidad sencilla como el clásico Space Invaders, que se pudiera jugar sin prisa como el Puzzle Bobble y que tuviera un componente aleatorio para que cada partida fuese diferente como en el Binding of Isaac.

De estas premisas salio Cañonero, un juego por turnos en 2D, con controles sencillos y una progresión de personaje y niveles aleatoria. Además, quería que se pudiera jugar en cualquier lugar y momento, de ahí la importancia a que fuera por turnos, no solo por el componente estratégico, sino para que el jugador pudiera realizar mientras otras actividades diarias sin que el juego lo condicionara negativamente.

1.4 Planificación del Trabajo

El primer paso fue realizar el documento de diseño del juego, en el que se recogían todos los aspectos relevantes, como la jugabilidad, la interacción del jugador con el juego, el tipo de gráficos a utilizar y la progresión del usuario.
Algunos otros elementos, como la totalidad de enemigos,tipos de escenarios, pantallas de transición adicionales o sonidos se dejaron para más adelante.
Por tanto, lo primero fue crear los modelos básicos del juego, como el cañón, un objeto destruible y un enemigo, a partir de las hojas de estilo elegidas. También se creo el escenario principal, suelo, muros y el espacio para la interfaz del usuario.
Después se fueron añadiendo las clases básicas para controlar el cañón, generar y mover los enemigos, controlar los rebotes del disparo, etc.
Una vez creada la base del juego se fueron añadiendo las características especiales del juego que le aportaban identidad, como la progresión aleatoria del usuario por medio de mejoras y compras, la selección de habilidades o la generación procedimental de niveles.
Por último se fueron implementando los sonidos, así como el resto de enemigos y escenarios, para finalmente preparar el juego para ser publicado en Google Play.

1.5 Breve sumario de productos obtenidos

- Un ejecutable para PC de la versión final del producto.
- Un APK instalable en Android 4.0 o superior, publicado en Google Play, con la versión final del producto.
- Memoria final del proyecto.
- 3 Videos relativos a las 3 entregas parciales explicando los progresos.
- 1 Video resumen explicando todo el proceso de creación del videojuego.
- 1 Diagrama de Gantt con la planificación del proyecto.
- 1 Documento explicando las posibles estados y transiciones de pantalla del videojuego.

1.6 Breve descripción de los otros capítulos de la memoria

El resto de capítulos de la memoria detallaran las diferentes partes del videojuego y como se ha creado cada una de ellas, desde el diseño de los elementos visuales, la lógica de la aplicación por medio de clases y los pasos necesarios para su publicación en Google Play.

2. Resto de capítulos

2.1 Elementos principales del juego.

Al ser un juego en 2D, todos los elementos han sido creados a partir de hojas de sprites o archivos de imágenes individuales.

Se han importado los archivos al proyecto y han sido editados con el editor de sprites de Unity, que permite subdividir las regiones de cada hoja en múltiples sprites. Una vez obtenidos los sprites, se han creado «prefabs» de Unity con cada uno de ellos, que actúan como contenedores. Ademas del sprite, en cada prefab se han incluido los elementos necesarios para su funcionamiento en el juego.

A continuación paso a detallar los objetos mas importantes del juego:

- **Cañón:**
  Representa al jugador. Puede moverse lateralmente o rotarlo hasta los 60º izquierda/derecha. Puede realizar un disparo por turno, ya sea básico o especial.

- **Muros superior, inferior y laterales:**
  Delimitan el espacio jugable y en ellos rebotaran los disparos. No se pueden destruir y los enemigos no pueden atravesarlos. Si los enemigos llegan al muro inferior explotan, dañando al jugador.

- **Enemigos:**
  Aparecen en las 3 filas superiores de la pantalla, en posiciones aleatorias. Cada vez que el jugador acaba su turno, se mueven hacia el muro inferior. Existen 4 tipos de enemigos, rápidos, normales, lentos y especiales. Cada uno tiene una cantidad de vida, ataque y velocidad diferentes, con el objetivo de que el jugador tenga que priorizar sus disparos y tomar decisiones. Los tipos de enemigos que aparecen también son aleatorios, aunque tienen mas probabilidad de aparecer enemigos rápidos y normales que lentos y especiales.

Estos son todos los tipos de enemigos creados:

- **Objetos destruibles:**
  Aparecen repartidos de forma aleatoria al inicio del nivel. Su función es entorpecer los disparos del jugador, aunque éste puede ayudarse de ellos para rebotar sus disparos y llegar a zonas imposibles. Además, proveen al jugador de oro si los destruye, con lo que son un objetivo importante además de eliminar enemigos.

  Estos son todos los tipos de objeto destruibles creados:

- **Disparos especiales:**
  Los disparos especiales pueden ser comprados al finalizar un nivel en la tienda, en función del oro disponible. Varían su precio en función de su poder y aportan un componente táctico importante. Su uso es imprescindible para avanzar en el juego ya que sin ellos el jugador se verá sobrepasado por los enemigos fácilmente. Tienen un tiempo de reutilización base de 9 turnos, aunque se puede mejorar ese tiempo comprando los siguientes niveles en la tienda.
  Estos son todos los tipos de disparo creados:

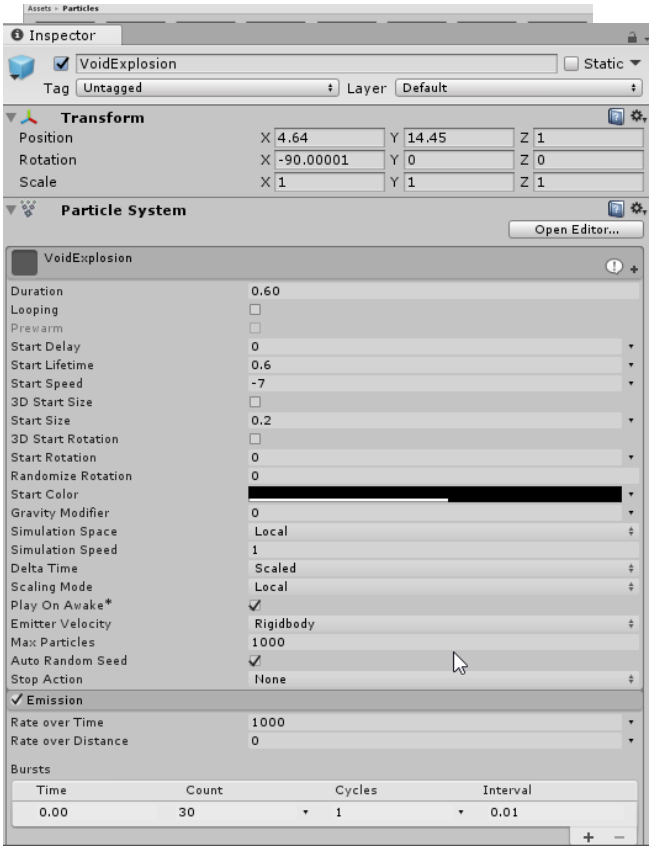
2.2 Resto de elementos del juego.

Además de los objetos que interactúan con el usuario descritos en el apartado anterior, se han creado otros para ayudar al jugador a entender el juego y aportar inmersión al mismo:

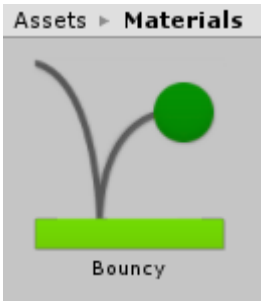
- **Textos flotantes**:
  Se han añadido textos flotantes a todas las interacciones que ocurren en el escenario. Como pueden ser; el daño que realiza el disparo sobre enemigos/objetos, el dinero/experiencia que reportan estos al ser destruidos o el daño que realiza un enemigo al usuario cuando llega al muro inferior. El texto flotante se compone de un controlador de animación, un script para controlar los datos que muestra (número, icono) y 3 animaciones diferentes para el color (daño, experiencia u oro).Al ser un juego en 2D, todos los elementos han sido creados a partir de hojas de sprites o archivos de imágenes individuales.

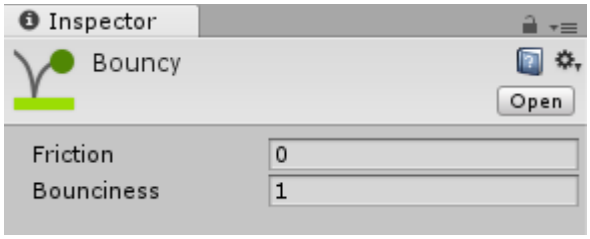
- **Barras de vida a enemigos/objetos destruibles:**
  Se ha creado un objeto genérico que sirve para representar la vida de los enemigos y objetos del escenario. La vida de los enemigos tiene un color verde, mientras que para los objetos destruible tiene un color marrón. Este objeto es individual para cada enemigo/objeto por lo que podría ajustar su tamaño para cada caso.

- **Partículas**:
  Se han utilizado diferentes partículas para crear efectos en el juego, como por ejemplo: una explosión roja cuando un enemigo es eliminado, marrón cuando un objeto es destruido, amarilla/naranja para el disparo de “bola de fuego” o una implosión para el disparo de “vacío”.
  Aquí podemos ver las partículas base que se han usado y las propiedades del objeto especifico para crear la explosión de “vacío”:



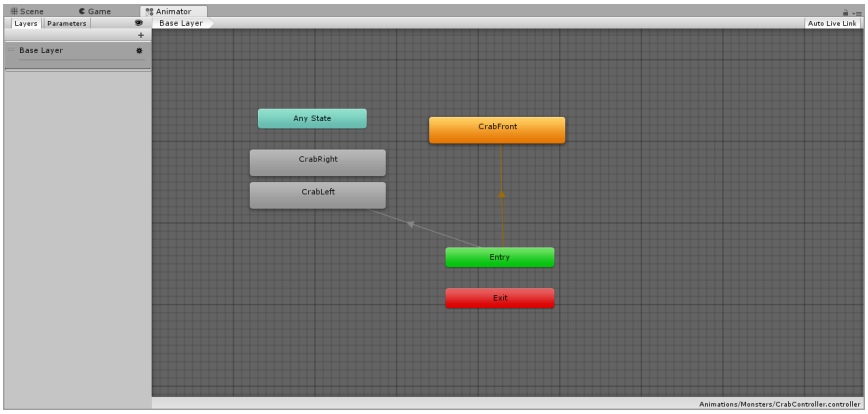
- **Materiales:**
  Se ha creado un material especifico para el disparo con el objetivo de que se comportara físicamente como una pelota que rebota sin fricción y sin perder fuerza en el rebote.
  Aquí podemos ver como esta este material definido:



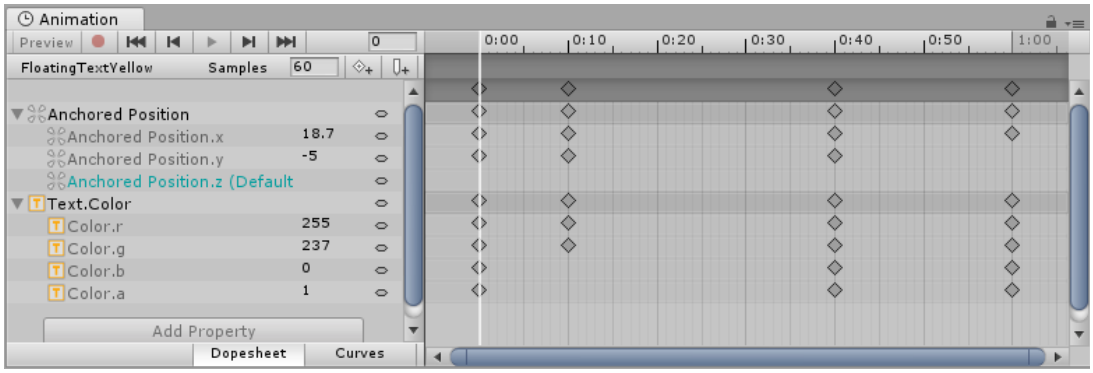


- **Animaciones:**
  Se han creado animaciones para el movimiento de los enemigos, disparos o textos flotantes. Cada animación completa se compone de un controlador y una animación. El controlador tiene la función de establecer el punto de entrada de la animación y su flujo. La animación contiene la definición de como debe comportarse, como por ejemplo velocidad de movimiento, sentido de la transición, colores..

  En la siguiente captura se puede apreciar el controlador de animación del tipo de enemigo “Crab”.



  También podemos ver los detalles de la animación de un texto flotante en la siguiente captura:



  Aquí podemos apreciar como esta animación controla el color mediante los parámetros RGB y la posición del texto mediante la variación de su valor de posición “Y”.

- **Temáticas:**
  El juego consta de 4 temáticas diferenciadas, Mazamorra, Desierto, Hielo y Jungla.
  Al generar cada nuevo nivel se elige una de estas temáticas de forma aleatoria, afectando a los siguientes elementos Enemigos, Objetos Destruibles, Suelo.

- **Tienda:**
  Se ha creado una pantalla de tienda donde el jugador puede gasta el oro que posee al terminar un nivel. La tienda siempre ofrecerá tres disparos especiales para comprar de forma aleatoria entre las 10 disparos especiales que he incluido en el juego.
  Los botones se desactivan automáticamente cuando el usuario compra una habilidad o no dispone de dinero suficiente para comprarla.

- **Pantalla de mejoras del jugador:**
  Cuando el jugador sube de nivel al eliminar enemigos, se le ofrecen tres mejoras aleatorias entre un total de seis. Esta mejora es gratuita y condiciona el tipo de partida del jugador.

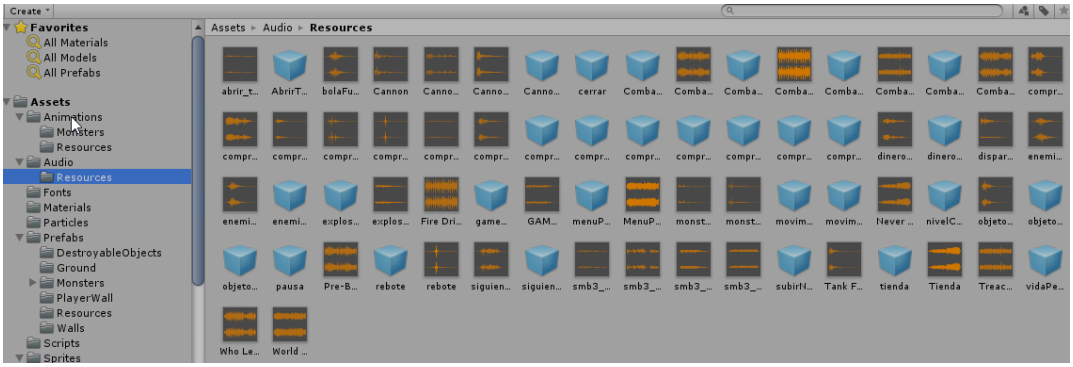
- **Pantallas de transición:**
  Además de la pantalla principal del juego donde transcurre gran parte de la acción, se han creado otras adicionales que sirven como hilo conductor del juego.

- **Selector de disparo:**
  El selector de disparo sirve, como su propio nombre indica, para elegir que tipo de disparo quiere usar el jugador.

  Cada disparo especial tiene un tiempo de recarga diferente que aparece sobre su dibujo una vez realizado el tiro. Con este selector el usuario puede cambiar de habilidad con facilidad y ver aquellas que puede utilizar. El disparo “Base” siempre estará disponible.

- **Sonidos:**
  Se han añadido sonidos de uso libre por Internet que pudieran ajustarse a las necesidades del juego. También se han añadido diferentes músicas de fondo para el menú principal, los 4 escenarios diferentes, la tienda y la pantalla de fin de partida Además de añadir los sonidos se han tenido que ajustar manualmente (volumen y reverberación) para que encajaran correctamente.

Estos son todos los sonidos utilizados en el juego:



2.3 Lógica y programación del juego.

A continuación voy a describir el propósito de cada una de las clases creadas así como sus funcionalidades mas importantes y detalles de las elecciones realizadas a la hora de decidir sus implementaciones.

GameManager:

Clase principal del juego que arranca al iniciar al estar incluida en el objeto “GameManager” de la escena principal.

Su cometido principal controlar el estado del juego, controlar las transiciones y cargar todos los objetos que van a ser compartidos por diferentes clases.

Su mayor complejidad reside en la función FixedUpdate(), la cual se ejecuta en cada frame del juego. En esa parte es donde el la lógica comprueba si se deben mover/aparecer los enemigos o hacer las comprobaciones para ver si el jugador ha eliminado a todos los enemigos de la pantalla o si por el contrario ha perecido.

También cuenta con varias funciones auxiliares que utilizan los botones del juego.

Enemy:

Como su nombre indica, esta clase sirve para modelar los enemigos del juego. En esta clase se definen sus propiedades como la cantidad de vida, ataque o numero de movimientos por turno.

La parte mas importante radica en las funciones de movimiento, ya que estos tienen que realizarse todos al mismo tiempo para cada objeto, a la vez que ir esquivando obstáculos.

Al principio se implementó una lógica para esquivar objetos basada en físicas, en la que se calculaba mediante un “Raycast” si el enemigo chocaría con algo antes de realizar el movimiento y así decidir si hacerlo o no. Como los objetos y los enemigos no tienen todos el mismo tamaño pero el objetivo del juego es que las posiciones de movimiento si fuesen fijas (por casillas), no funcionaba correctamente.

Finalmente se cambió el sistema de detección de colisiones a uno basado en una tabla de valores. Se guarda en memoria una tabla con todas las casillas del tablero y aquellas ocupadas se marcan con un uno, así cuando un enemigo va a moverse, solo se debe comprobar si la casilla esta vacía o no. Esto implicó añadir alguna lógica extra en los

objetos destruibles, ya que el juego tiene que ir controlando aquellos que se destruyen para liberar las casillas correspondientes en la tabla (si no, los enemigos chocarían contra objetos ya destruidos).

DestroyableObject:
Esta clase sirve para modelar los objetos destruibles del juego. Es una clase sencilla que sirve para almacenar las propiedades básicas de cada objeto y solo contiene un método que se lanza al dañar un objeto y desencadenar el resto de acciones necesarias, como mostrar el texto flotante o destruirlo.

CannonManager:
Con esta clase se controla el movimiento y disparo del cañón. Su parte mas interesante reside en el movimiento de cañón. en función de donde se esta tocando/haciendo click. La función que detecta el movimiento se llama múltiples veces por frame para que el movimiento sea preciso. El mayor reto fue controlar correctamente el giro, ya que hay que aplicar rotaciones parciales en función del giro previo que tuviera el cañón.
También desde esta clase se controla la acción del disparo aunque sus efectos se delegan a otra clase que se describe a continuación.

ShootManager:
Esta clase controla todas las interacciones del disparo con el entorno. Además se encarga de instanciar todos los tipos de disparo para así tenerlos listos cuando el jugador los compre en la tienda, controlando también toda la lógica especifica de cada disparo.
Las colisiones de las balas se controlan de dos maneras diferentes, con “triggers” y con “collision enter”. Los objetos contra los que la pelota reacciona de forma física (rebota) son detectados mediante colisión, mientras que los objetos contra los que la pelota no cambia su estado se controlan mediante triggers (como pueden ser los enemigos).
Otra función interesante de la clase es ocuparse de limpiar el escenario de los objetos que algunos tipos especiales de disparos dejan durante dos turnos, como fuego, hielo o veneno.

Skill:
Clase básica que modela los objetos de las habilidades con las propiedades necesarias.

SkillCarousel:
Esta clase controla el selector de disparos especiales del jugador. Se encarga realizar todas las acciones asociadas, como el movimiento o la adición de disparos cuando el jugador compra alguno en la tienda.
Como última modificación, también controla un mensaje flotante que aparece como ayuda al usuario cada vez que se elige una disparo especial diferente con el objeto de recordar sus propiedades.

Player:
Clase que se utiliza para modelar la instancia del jugador en cada partida. Es una clase sencilla que sirve también para aplicar las mejoras del jugador cuando este sube de nivel.

PlayerUpgrades:
Esta clase sirve para definir las posibles mejoras del jugador. Su parte mas interesante es la que se utiliza para proponer las mejoras aleatorias a mostrar cada vez que el jugador sube de nivel.
Como los botones de la pantalla son fijos, se tuvieron que utilizar Listeners para cambiar de forma dinámica la función a ejecutar cada vez que se generaban las mejoras propuestas al jugador.

BoardManager:
Esta clase se encarga de crear el tablero del juego en cada nivel, además de realizar muchas de las acciones que transcurren a lo largo del juego.
Se encarga de seleccionar un tema aleatoriamente para cargar sus baldosas, enemigos y objetos destruibles correspondientes, para posteriormente mostrarlos.
También lleva el control de los enemigos que restan por aparecer así como de mantener la tabla de posiciones ocupadas mencionada anteriormente.

TutorialManager:
Esta clase se encarga exclusivamente de gestionar el tutorial, instanciando sus objetos necesarios y estableciendo los métodos para ir al siguiente/anterior paso.

CanvasController:
Clase estática que se encarga de controlar la aparición de las pantallas del juego.
En ella se definen todas las pantallas posibles ademas de tener una función genérica que sirve para hacer las transiciones. También tiene efecto sobre la clase principal Game Manager ya que controla las pausas del juego.

FloatingText:
Clase simple que sirve para establecer dinamicamente el valor del texto flotante a instanciar.

FloatingTextController:
Esta clase sirve para manejar el tipo de texto flotante que aparecerá en pantalla (daño, oro, experiencia o vida).
También sirve para posicionar el texto en la posición requerida.

Shop
Esta clase es la responsable de realizar todas las tareas relacionadas con la tienda del juego.
En primera instancia obtiene todos los componentes necesarios de la pantalla de la tienda y registra todos los tipos de disparos disponibles.
Al igual que las mejoras del jugador al subir de nivel, los disparos ofrecidos en la tienda son aleatorios, por lo que la tienda se encarga de instanciarlos cada vez que la pantalla correspondiente es mostrada.

Como el usuario puede comprar varias habilidades a la vez pero depende del oro que disponga, la clase posee una pequeña lógica que cambia el estado de los botones de compra en función de oro actual del jugador.

2.4 Interfaz de usuario.

La interfaz del usuario esta diseñada con el objetivo de que sea sencilla y ágil, además de vistosa, buscando un aspecto “retro”.

La parte inferior de la pantalla contiene los elementos mas importantes:

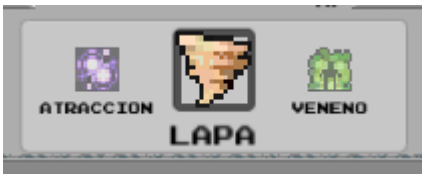
- **Vida del jugador:** Muestra la cantidad de vida restante del jugador, si se vacía, el jugador pierde la partida.

- **Ataque:** Muestra el ataque actual del cañón. del jugador.
- **Rebotes:** Muestra la cantidad de rebotes que soportara la bola disparada antes de destruirse.

- **Experiencia:** Muestra la cantidad de experiencia actual del jugador en relación a la requerida para subir de nivel reflejada en una barra de progreso de color verde.

- **Oro:** Muestra el oro actual del jugador.

- **Selector de Disparos:** Como se menciono anteriormente, el selector de disparos sirve para saber que disparo se tiene seleccionado así como para también poder cambiar entre los diferentes disparos que tiene disponibles el jugador.



Además de la interfaz inferior, existen dos elementos situados en la parte superior del tablero de juego:

- **Enemigos restantes:** Marcado con una calavera, muestra el numero de enemigos que faltan de eliminar antes de pasar al siguiente nivel.

- **Puntos:** Marcador de puntos del jugador, que aumentan al derrotar enemigos o destruir objetos del escenario.

- **Botón de pausa:** Este botón cumple la función de pausar el juego y hace aparecer un menú en el que puedes quitar el sonido, salir o continuar el juego.

2.5 Mecánica del juego

La premisa principal del juego es evitar que los enemigos lleguen a tocar el muro inferior. El juego consta de las siguientes fases principales:

- **Fase del Jugador:** El jugador debe valerse del cañón. para posicionarlo y rotarlo a su antojo antes de disparar. Una vez que el jugador dispara, el proyectil atravesará a los enemigos, dañándolos en un valor igual al ataque del jugador, y rebotando contra cualquier obstáculo un numero de veces determinado por el atributo de rebotes del jugador antes de romperse. Una vez que todos los proyectiles han desaparecido , comienza la fase del enemigo.

- **Fase del Enemigo:** Aparecerán un numero aleatorio de enemigos, entre 2 y 5 (siempre que queden al menos ese numero de enemigos por aparecer en la pantalla actual). Estos enemigos aparecerán en la parte superior de la pantalla, siempre entre en la primera y tercera columna en posiciones aleatorias. Una vez que los enemigos han aparecido, todos los enemigos del tablero se moverán hacia el muro defendido por el jugador. Cada enemigo tiene una cantidad de movimientos diferentes (rápido, normal y lento). Una vez que todos los enemigos han realizado su movimiento, dará lugar otra vez a la fase del jugador.

  Además de las fases principales, el juego consta de varias fases/pantallas adicionales:

- **Menú Principal:** Es la pantalla inicial del juego, mediante la cual se puede acceder a jugar una nueva partida, recorrer el tutorial o activar/desactivar el sonido.

- **Subida de nivel:** Cuando la experiencia representada por la barra verde se llena, el jugador sube de nivel, apareciendo una pantalla que le

permitirá elegir una mejora. Las mejoras ofrecidas son aleatorias y siempre se le ofrecerán tres entre las 6 que hay disponibles:

- Vida
- Ataque
- Rebotes
- Espacio
- Bonus Experiencia
- Bonus Oro

Esta mejora es gratuita y el jugador deberá elegir en función del estilo de juego que quiere para esta partida. Por ejemplo, puede intentar centrarse en tener un cañón poderoso con ataque y rebotes mejorados, o por contra puede centrarse en ampliar el bono de experiencia y oro, haciendo que a la larga tenga muchas mas mejoras disponibles y mas dinero para comprar habilidades en la tienda.

- **Nivel Completado:** Cuando el jugador elimina el último enemigo de la pantalla, el jugador habrá completado el nivel. En esta nueva pantalla, el jugador podrá ver el numero de enemigos que ha derrotado y los objetos que ha destruido. Ademas, tendrá dos opciones, visitar la tienda de habilidades o continuar al siguiente nivel.

- **Tienda del juego:** El jugador podrá acceder a la tienda cada vez que termine un nivel. La tienda siempre ofrecerá al jugador la compra de tres habilidades, escogidas de forma aleatoria entre las 10 de las que consta el juego. El jugador podrá comprar todas aquellas habilidades que se pueda permitir en función del oro que tiene disponible. Una vez que el jugador acabe de utilizar la tienda, pulsara sobre el botón «Listo» para continuar al siguiente nivel.

- **Siguiente Nivel:** El jugador no podrá realizar ninguna acción en esta pantalla. Sirve como transición entre niveles e indica el número del nivel al que el jugador se va a enfrentar.

- **Game Over:** Si la vida del jugador, representada por la barra roja se vacía, el jugador pierda la partida y aparecerá esta pantalla. En ella podrá elegir si quiere ir al menú principal o jugar otra partida.

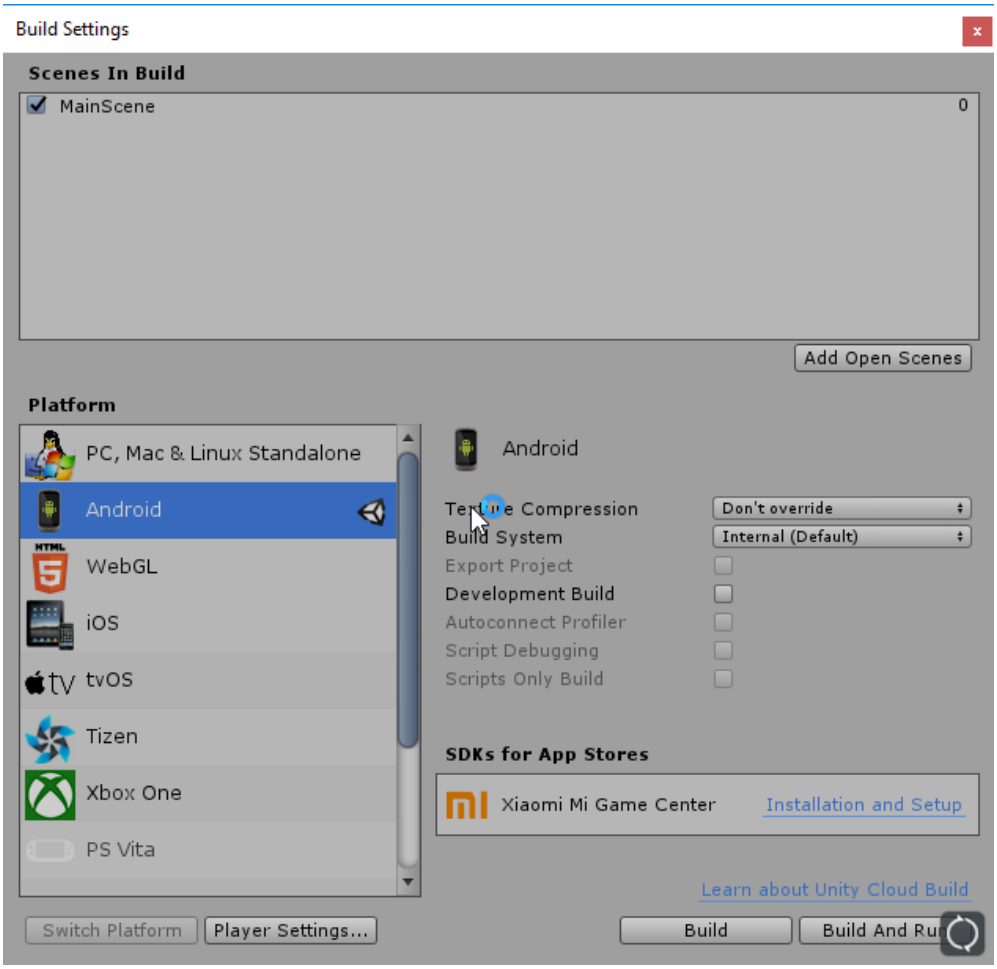
- **Pausa:** En cualquier momento, el jugador podrá pulsar el botón de pausa para parar el juego, apareciendo una ventana con las opciones de apagar/encender el sonido, continuar la partida o salir al menú principal.

- **Progresión e incremento de la dificultad:** El número de enemigos y la cantidad de vida de estos y de los objetos destruibles se incrementa en cada nivel. El reto del jugador es conseguir mejorar sus estadísticas y disparos a un nivel igual o superior para no verse desbordado y perder la partida.

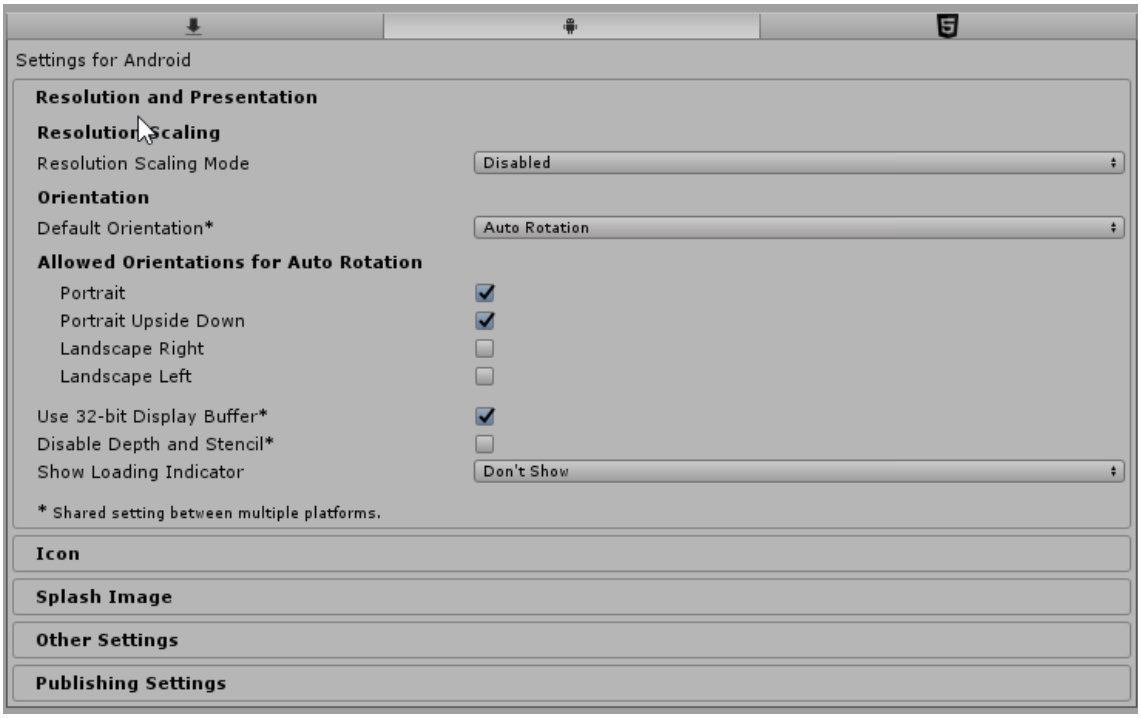
2.6 Publicación en Google Play.

Como colofón final, se ha publicado el juego en Google Play y a continuación se describen los pasos necesarios para hacerlo posible:

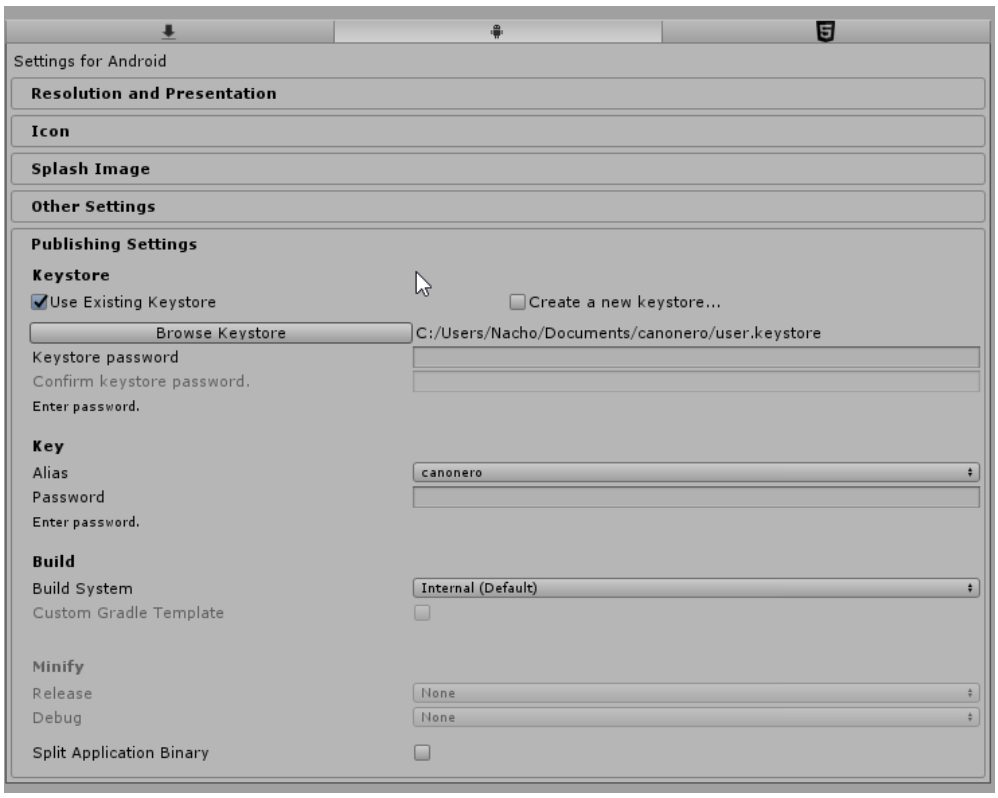
1. Configuración en Unity.
   Lo primero es configurar el tipo de build de Unity como proyecto de Android. Esto recompilara los archivos necesarios de forma automática para adaptarse a los requisitos de esta plataforma.



   Después, a través de la opción Project Settings - > Player habrá que configurar las características generales del juego, como la orientación (si es que se quiere tener fija), como el escalado de resolución.

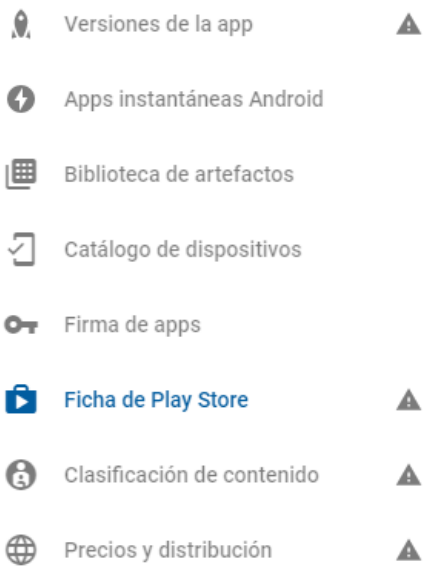


Por último, habrá que crear un nuevo keystore (si es que no se tiene ya uno), con el objetivo de firmar el APK con una clave válida y que se pueda reutilizar en el futuro (por ejemplo, para realizar actualizaciones).



Una vez terminada la configuración, solo resta pulsar sobre build para obtener nuestro APK y poder distribuirlo.

2. Creación de la aplicación en nuestro espacio de desarrollador de Google Play.
   Después de identificarnos en play.google.com tendremos que pulsar sobre “Crear nueva aplicación”. Elegiremos un nombre y aparecerá una nueva pantalla donde se indica todo lo que tendremos que rellenar para tener nuestra aplicación disponible en la tienda.

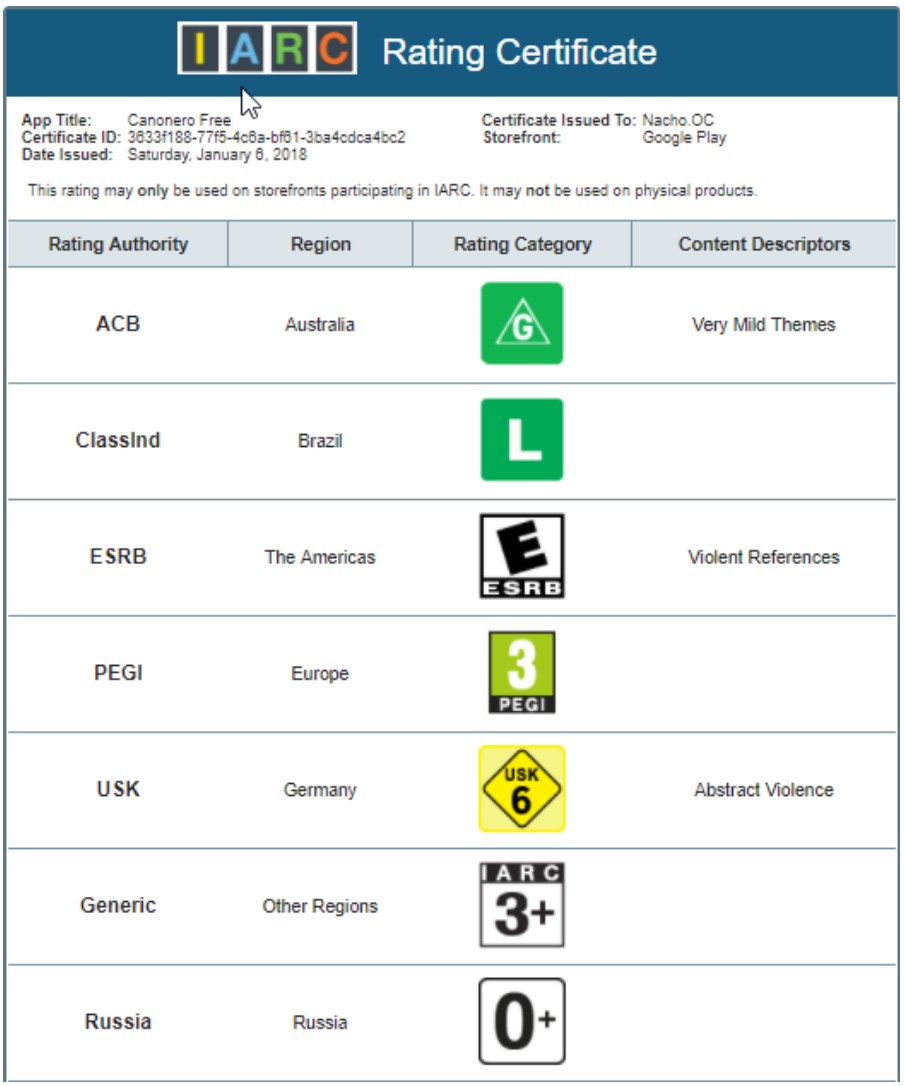


Como podemos observar, es obligatorio rellenar varios pasos:

- **Precio:** En nuestro caso la aplicación es gratuita, así que es tan sencillo como marcarlo de ese modo y habremos terminado.

- **Clasificación y contenido:** Este paso sirve para avisar sobre los contenidos del juego, haciendo un énfasis especial en la violencia o los contenidos para adultos mostrados en el juego. Una vez que se rellena el formulario propuesto, el servicio nos otorgará un certificado del juego para los diferentes países. En nuestro caso hemos obtenido el siguiente:

Por ejemplo, el certificado indica que para Europa, la edad mínima recomendada para jugar a nuestro juego es de 3 años.



IARC Rating Certificate

App Title: Canonero Free
Certificate ID: 3633f188-77f5-4c8a-bf81-3ba4cdca4bc2
Date Issued: Saturday, January 6, 2018

Certificate Issued To: Nacho.OC
Storefront: Google Play

This rating may only be used on storefronts participating in IARC. It may not be used on physical products.

Rating Authority	Region	Rating Category	Content Descriptors
ACB	Australia	G	Very Mild Themes
ClassInd	Brazil	L	
ESRB	The Americas	E ESRB	Violent References
PEGI	Europe	3 PEGI	
USK	Germany	USK 6	Abstract Violence
Generic	Other Regions	IARC 3+	
Russia	Russia	0+	

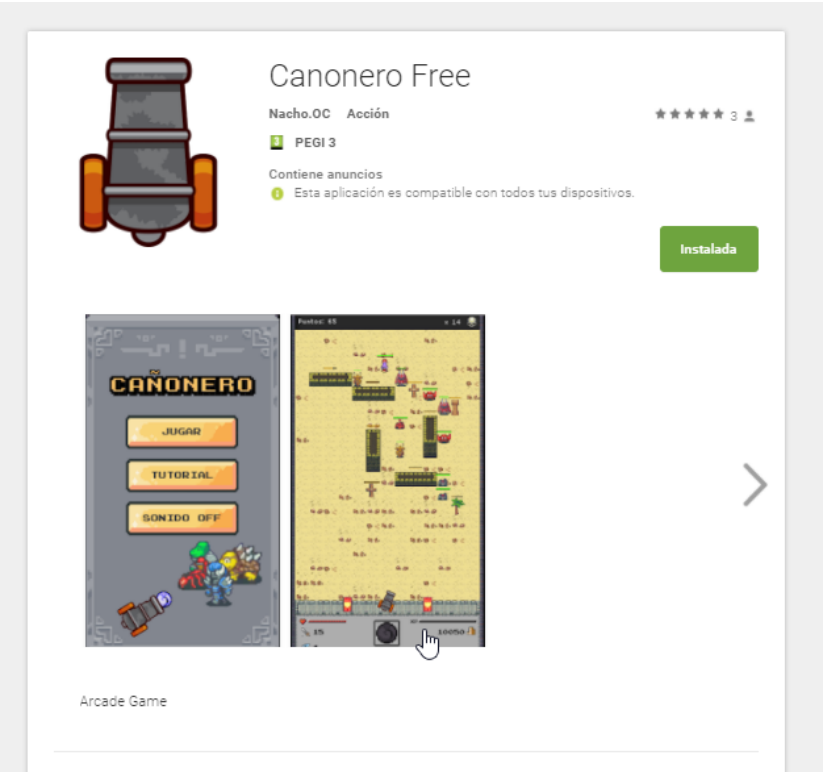
- **Ficha del Play Store:** En este apartado debemos de rellenar todos los datos relacionados con el juego, temática, capturas de pantalla promocionales, iconos… La mayoría de los campos son opcionales, pero desde Google recomienda rellenar la mayor cantidad posible ya que ayudará a que nuestro juego sea mas atractivo y mas sencillo de localizar en la tienda.

- **Versiones de la APP:** Este es el último paso y en esta pantalla deberemos subir nuestro APK generado por Unity. Existen 3 formas de lanzar nuestra aplicación, en fase Beta, Alpha y Producción. Las dos primeras fases están destinadas a pruebas, y solo usuarios seleccionados por nosotros mismos pueden descargar el juego de la tienda. La última fase, Producción, es la destinada a alojar el APK final que queremos que este disponible para el público en

general. Nuestro APK paso por las 3 fases antes de ser puesto finalmente en Producción.

Una vez terminados todos los pasos, podremos ver nuestro juego publicado en la tienda de Google Play, como se observa en la siguiente imagen:

https://play.google.com/store/apps/details?id=com.ordorica.canonerofree



3. Conclusiones

En computo general estoy contento con el resultado del juego. Creo que la parte fundamental para haber conseguido finalizarlo a tiempo a sido la planificación de cada uno de los hitos.
Gracias al consejo del consultor, cree un diagrama de Gantt con las partes a desarrollar y una estimación del tiempo necesario.
A pesar de no haberlo cumplido al 100%, ya que algunos desarrollos fueron mas difíciles y costosos de lo esperado, me ha venido muy bien para tener siempre una visión global de lo que faltaba por terminar.
(Diagrama de Gantt del proyecto:
https://www.tomsplanner.es/public/canonero)

Ademas de la planificación, conocer la herramienta de desarrollo también es muy importante. En este caso no tenia mucha práctica con ella y eso ha influido en algún retraso en el desarrollo, aunque la idea era aprender a utilizar Unity, así que los problemas en este apartado están justificados.

Mención especial al apartado artístico del juego, ya que me fuerte no es el diseño sino la programación y me ha costado mucho completarlo, a pesar de disponer de hojas de sprites que ya había adquirido previamente. El tratamiento de los sprites como la creación de las animaciones es un proceso lento, ya que hay que realizar muchos retoques para que funcione como quieras.

También es importante estar abierto a modificaciones y creo que una ellas ha sido clave para que el juego fuese divertido; me estoy refiriendo a la adición del selector de habilidades, sugerido por varios compañeros que pudieron probar una versión parcial del juego. Tras esta entrega me gustaría seguir mejorando el juego, es más, actualmente el código ya tiene nuevas funcionalidades que por falta de tiempo no están implementadas, como la aparición de objetos de bonificación (cofres, cristales preciosos…).

Por último, me gustaría reseñar que en mi opinión, crear un videojuego una persona es un trabajo muy difícil, ya que tienes que abarcar muchos campos (diseño, sonido, gráficos, programación, inventiva), por lo que me gustaría realizar mis siguientes procesos con un grupo reducido de personas y que cada uno tuviera su especialización. Aun así, creo que la creación de este juego me ha servido para mejorar en muchas áreas que casi no había tocado y en general a sido muy positivo para mi carrera como desarrollador.

4. Glosario

Sprite: Mapa de bits en 2D que se dibujan directamente en un destino de representación sin usar la canalización de transformaciones, iluminación o efectos.

Diagrama de Gantt: Es una herramienta gráfica cuyo objetivo es exponer el tiempo de dedicación previsto para diferentes tareas o actividades a lo largo de un tiempo total determinado.

Testing: Son pruebas de software o una fase del desarrollo de un producto.

APK: Es el formato utilizado por el sistema Android para poder instalar en el aplicaciones.

Prefab: Es un objeto contenedor especifico de Unity que sirve para encapsular varios objetos en uno solo. Por ejemplo, un prefab Player podría encapsular el script, sprite y animaciones asociados a él.

Script: Es un archivo de ordenes que contiene una lógica de programación. En nuestro caso se refiere a los archivos de C# que contienen la lógica del juego.

Frame: Es un fotograma del juego.

FixedUpdate: Es un método especifico de Unity que se ejecuta una vez por cada fotograma del juego.

Raycast: Es un objeto de Unity que simula el lanzamiento de un rayo hacia un punto y que nos da información sobre las colisiones producidas hasta llegar a ese punto.

Trigger: Es un disparador y en nuestro proyecto se refiere al evento que se ejecuta cuando ocurre determinada acción. Por ejemplo, cuando un disparo toca un enemigo.

Instancia: Es la realización especifica de una determinada entidad. En nuestro caso, cada vez que aparece un enemigo en pantalla se dice que es una instancia de un enemigo.

Build: En nuestro caso se refiere al tipo de compilación y construcción del ejecutable del juego y va en función del entorno final elegido.

Keystore: Es un almacén virtual de certificados con claves privadas.

5. Bibliografía

- Sprites y animaciones
  http://timefantasy.net/
- Instalación y uso del entorno de desarrollo Unity
  https://unity3d.com/es
- Sonidos
  https://freesound.org/
- Diagrama de Gantt
  https://www.tomsplanner.es/
- Consultas sobre programación y scripts
  https://es.stackoverflow.com/
- Consultas sobre la creación y administración de elementos en Unity.
  https://forum.unity.com/

6. Código Fuente

Debido al tamaño del código fuente del proyecto no se ha adjuntado en la entrega pero se puede descargar libremente en:

https://drive.google.com/open?id=1BIrCBheO-QsDK4k1FfQ-7pTlDDL2GG7w

7. Presentación

El video de la presentación pueden encontrarse en:

https://vimeo.com/250026216?utm_source=email&utm_medium=vimeo-cliptranscode-201504&utm_campaign=29220