ARShooter

Nombre Estudiante: Jordi Pérez Sandonís
Máster Universitario en Desarrollo de Aplicaciones para Dispositivos Móviles

Nombre Consultor/a: Francesc D´Assís Giralt Queralt
Profesor/a responsable de la asignatura: Carles Garrigues Olivella

03/01/2018

FICHA DEL TRABAJO FINAL

Título del trabajo:	*ARShooter*
Nombre del autor:	*Jordi Pérez Sandonís*
Nombre del consultor/a:	*Francesc D´Assís Giralt Queralt*
Nombre del PRA:	*Carles Garrigues Olivella*
Fecha de entrega (mm/aaaa):	01/2018
Titulación::	Máster Universitario en Desarrollo de Aplicaciones para Dispositivos Móviles
Idioma del trabajo:	*Castellano*
Palabras clave	*Realidad Aumentada, Aplicación, Smartphone*

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

ARShooter es un juego de disparos/acción con toques del estilo “defender la torre” para dispositivos móviles (tablet, smartphone) realizado en Unity con el soporte de ARToolkit. El juego utiliza marcadores físicos con símbolos específicos para determinar la posición de la cámara y los utiliza como puntos de referencia para sobre-impresionar imágenes sintéticas (3D) sobre la imagen real del vídeo capturado por la cámara del dispositivo. La finalidad última del proyecto es la de realizar un producto de entretenimiento diferente a lo que está acostumbrado el mercado utilizando técnicas relativamente nuevas y experimentar con ellas. La elección de utilizar librerías y un SDK para juegos existente en el mercado se debe a que realizar todo el desarrollo desde la base se hace imposible en el tiempo disponible para la realización del proyecto.

Abstract (in English, 250 words or less):

ARShooter is a mobile shooter/action game with a little component of a “defense the tower” made with Unity with the support of the ARToolkit library. The game uses physical markers with specific symbols to determine the camera’s position and uses those markers as reference points in space for mix 3D images with real video feed captured with the device’s camera. The purpose of this project is to develop an entertainment product a little bit different from the average using relatively new techniques and experiment with them. The choice of using existing libraries and a game SDK it’s because the big complexity of developing all the subsystems from the ground up.

Índice

1. Introducción

1.1 Contexto y justificación del Trabajo

Dado el aumento de las capacidades de los dispositivos móviles y el gran mercado que representa el ocio electrónico, el mundo de los juegos para dispositivos móviles ha evolucionado en todos los aspectos. Ya no solo con mejores gráficos o mejor sonido sino modificando la manera en que las personas interaccionamos con los productos. Un ejemplo de esta evolución es el uso de la realidad aumentada, que consiste en mezclar elementos del mundo real (generalmente imágenes capturadas con una cámara) con elementos virtuales (generalmente imágenes generadas a partir de modelos tridimensionales o fuentes externas de datos).

La realidad aumentada no es una técnica ligada a los dispositivos móviles de uso más común como los smartphones y tablets, han aparecido otros dispositivos como las Google Glass, capaces de mostrar información gráfica sobre sus cristales, mezclando así las imágenes del otro lado del cristal con datos procedentes de fuentes de datos externos. Esto ha permitido crear todo tipo de aplicaciones de asistencia en distintas tareas del ámbito personal y profesional además de multitud de aplicaciones de ocio.

Un ejemplo de esto es la aplicación Pokémon Go, en la que se puede ver ver la superposición de criaturas del universo Pokémon sobre el video que entra por la cámara del dispositivo en tiempo real. El juego utiliza los datos del GPS y el giroscopio para posicionar la cámara virtual y mostrar las criaturas geolocalizadas que se encuentran en frente del dispositivo. Esta aplicación ha tenido una gran repercusión en los medios de comunicación por la cantidad de jugadores y la revelación que supuso en su momento.

Pokémon Go no ha sido la única aplicación en utilizar estas técnicas, hay muchas otras aplicaciones que utilizan la realidad aumentada para mostrar trayectos, instrucciones en ruta, traducir carteles, mostrar modelos tridimensionales a escala del cuerpo humano o edificios históricos, etc.

Dada la situación actual en el campo de la realidad aumentada, su estrecha relación con los dispositivos móviles y la relevancia económica y cultural de los videojuegos hacen especialmente interesante realizar un proyecto final de máster que involucre todos estos conceptos.

1.2 Objetivos del Trabajo

El objeto de este proyecto es un juego de disparos en primera persona para smartphone y tablet en el que el jugador deberá evitar que sus enemigos destruyan unos objetos presentes en cada nivel a la vez que acumula puntos eliminando el máximo de enemigos posible.

El juego utilizará técnicas de realidad aumentada para mover el punto de vista del jugador y mostrar un mundo tridimensional sobre una superficie real. La técnica utilizada para el posicionamiento y orientación del punto de vista consistirá en el reconocimiento óptico de un marcador físico (tarjeta impresa con un código bidimensional) que deberá estar presente en el video capturado por la cámara del dispositivo. Este marcador se puede encontrar en los anexos de la memoria y además se incluye en un archivo ".pdf" junto al resto de entregables del proyecto.

La interacción del usuario con el juego se dará a través del movimiento del dispositivo (traslación y rotación) con respecto al marcador mencionado y mediante toques en la pantalla táctil sobre los botones y otros controles que ofrecerá el juego.

El jugador deberá poder utilizar distintas armas y modificadores (powerups) durante el juego.

El juego utilizará gráficos 3D para representar el escenario de juego y gráficos en 2D para el HUD (información en pantalla, menús, …). Además se utilizarán sistemas de partículas para representar las explosiones e impactos de bala.

El juego deberá estar acompañado de música y efectos de sonido que sonarán como respuesta a distintas situaciones.

El juego contemplará las colisiones entre objetos y utilizará un sistema de física que incluirá el efecto de la gravedad y la reacción de los objetos a las fuerzas de los impactos de los disparos.

Se proveerá de una inteligencia artificial simple que controlará a los enemigos haciéndoles buscar caminos entre los obstáculos hasta su objetivo.

El juego deberá guardar un registro de las mejores puntuaciones obtenidas por los jugadores.

Los niveles del juego estarán almacenados en archivos externos que podrán cargarse durante la ejecución. En el proyecto se incluye la especificación de este tipo de archivos pero no se entrará en la programación de una herramienta para crearlos ya que añadiría una cantidad de trabajo extra que no aportaría nada significativo al producto central del proyecto.

1.3 Enfoque y método seguido

Superponer gráficos sobre imágenes reales requiere que la aplicación determine la posición y orientación del observador con respecto al mundo real. Algunas de las técnicas que se utilizan para determinar la posición y orientación son:

1. Analizar imágenes desde el punto de vista del observador en busca de puntos de referencia como pueden ser marcadores con formas, dibujos o colores característicos.

2. Analizar los datos de los sensores del dispositivo:
   - GPS
   - giroscopio
   - brújula
   - …

Estas técnicas son perfectamente combinables para conseguir el resultado que mejor se adapte a las necesidades de la aplicación a desarrollar.

Dada la cantidad de tiempo y la complejidad que podría suponer el desarrollo de las funcionalidades relacionadas con el análisis de la imagen capturada por la cámara y los cálculos para obtener la posición de la cámara virtual, se hace necesario partir de un trabajo previo, en este caso de alguna de las librerías existentes que resuelven estas situaciones como:

- ARToolkit (https://artoolkit.org)
- Kudan (https://www.kudan.eu)
- Vuforia (https://www.vuforia.com)
- Wikitude (https://www.wikitude.com)

Las restricciones de tiempo y recursos hacen que desarrollar un motor de juego desde la base no sea una opción viable. Para solventar este problema se han buscado motores que permitan su utilización de forma gratuita, que se adapten a las necesidades de este proyecto y que además generen ejecutables para distintas plataformas móviles. Las opciones que se han tenido en cuenta son:

- libGDX (https://libgdx.badlogicgames.com)
- Unity (https://unity3d.com)
- Unreal Engine (https://www.unrealengine.com)

En conclusión, el enfoque más adecuado para desarrollar el juego es partir de la base de un motor de juegos existente y extenderlo con una librería orientada a la realidad aumentada. Teniendo en cuenta los costes (dando prioridad a la opción menos costosa) y la experiencia previa con algunas de las herramientas necesarias para la elaboración del proyecto (Unity, lenguaje de programación c#, ...), la combinación Unity + ARToolkit es la que mejor se adapta a las necesidades de este proyecto.

1.4 Planificación del Trabajo

El tiempo que se dedicará al proyecto dependerá del día de la semana:

- 1’5h en días laborables
- 4h en días festivos

resultando en 15’5h semanales.

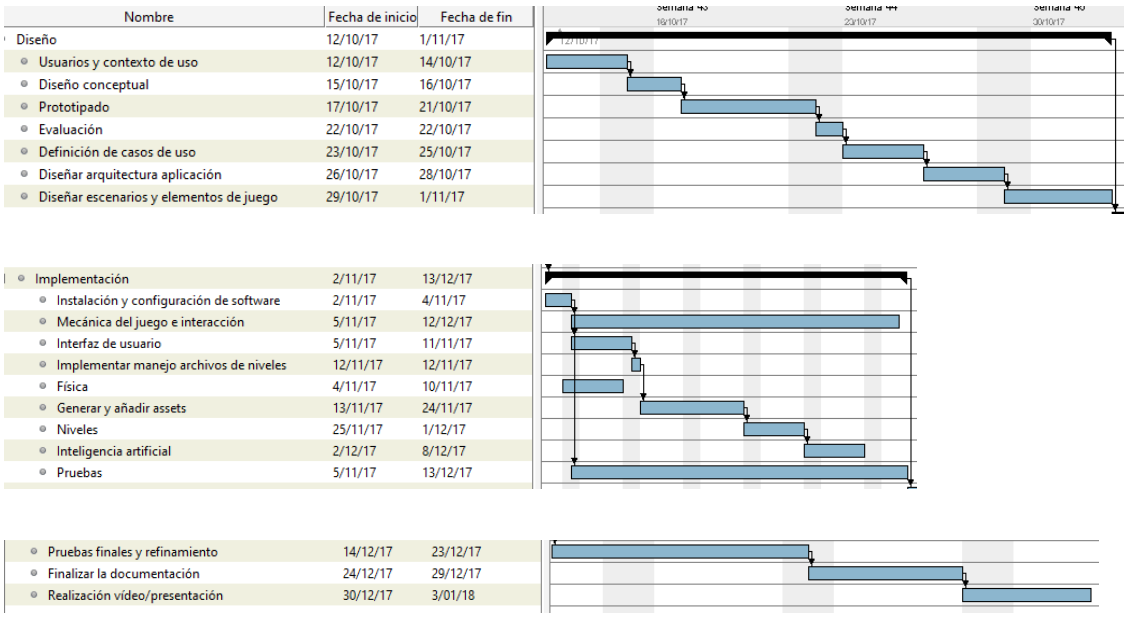
Recursos necesarios:

- PC con el software necesario para programar el juego (Unity, ARToolkit, Android SDK, Visual Studio Community, …)
- Dispositivo Android para realizar las pruebas

Tareas a realizar:

- Diseño y documentación [48h]
  - Usuarios y contexto de uso [6h]
  - Diseño conceptual [6h]
  - Prototipado [10h]
  - Evaluación [3h]
  - Definición de los casos de uso [5h]
  - Diseño de la arquitectura de la aplicación [10h]
  - Diseño de escenarios y elementos de juego [8h]
- Implementación y documentación [77,5h]
  - Instalación y configuración de software [7h]
  - Mecánicas del juego e interacción [77,5h*]
  - Interfaz de usuario [15,5h*]
  - Fichero de niveles/escenas [4h*]
  - Física [15,5h*]
  - Generar y añadir assets [23h*]
  - Niveles [15,5h*]
  - Inteligencia artificial [15,5h*]
  - Pruebas [77,5h*]
- Pruebas finales y refinamiento [20h]
- Finalizar la documentación [14h]
- Realización del video/presentación [14h]

* Los recuentos de horas que incluyen un asterisco representan horas compartidas con otras tareas



Nombre	Fecha de inicio	Fecha de fin
Diseño	12/10/17	1/11/17
Usuarios y contexto de uso	12/10/17	14/10/17
Diseño conceptual	15/10/17	16/10/17
Prototipado	17/10/17	21/10/17
Evaluación	22/10/17	22/10/17
Definición de casos de uso	23/10/17	25/10/17
Diseñar arquitectura aplicación	26/10/17	28/10/17
Diseñar escenarios y elementos de juego	29/10/17	1/11/17
Implementación	2/11/17	13/12/17
Instalación y configuración de software	2/11/17	4/11/17
Mecánica del juego e interacción	5/11/17	12/12/17
Interfaz de usuario	5/11/17	11/11/17
Implementar manejo archivos de niveles	12/11/17	12/11/17
Física	4/11/17	10/11/17
Generar y añadir assets	13/11/17	24/11/17
Niveles	25/11/17	1/12/17
Inteligencia artificial	2/12/17	8/12/17
Pruebas	5/11/17	13/12/17
Pruebas finales y refinamiento	14/12/17	23/12/17
Finalizar la documentación	24/12/17	29/12/17
Realización vídeo/presentación	30/12/17	3/01/18

Esta planificación puede ser modificada o revisada durante la ejecución del proyecto. Si durante la fase de implementación se detecta que la planificación no se ajusta a las necesidades del proyecto debido a su complejidad o problemas que puedan surgir, se optaría por tomar alguna de las siguientes medidas:

- reducir funcionalidades que no sean básicas para el juego:
  - limitar la cantidad y complejidad de los powerups
  - limitar la cantidad y complejidad de las armas
  - limitar la complejidad de la inteligencia artificial

- invertir una parte del tiempo de la siguiente fase en completar las funcionalidades que falten. En este caso no podrá ser más de una semana y siempre teniendo en cuenta que durante esa fase hay días festivos en los que el trabajo en el proyecto será complicado.

1.5 Breve sumario de productos obtenidos

- Aplicación Android (juego)
- Documento digital con el marcador utilizado por la aplicación para imprimirlo y poder utilizar el juego
- Memoria del proyecto
- Presentación en video

2. Temática e historia del juego

ARShooter es un juego futurista de acción en primera persona que se desarrolla en un futuro ficticio en el que los humanos se enfrentan a dos grandes problemas: el agua potable escasea y los robots, cansados del trato abusivo de los humanos, deciden destruirlos. El plan de los robots es clonarse para formar el ejército más grande jamás visto y destruir todas las reservas de agua que quedan en la Tierra.

La acción del juego se desarrolla en el interior de una depuradora, el jugador deberá disparar a los robots que se acerquen a los tanques de almacenamiento para destruirlos a todos o morirá inevitablemente como el resto de los miembros de su especie.

Cada nivel se representa en un escenario tridimensional y tiene una duración limitada. Para apuntar se utiliza el punto de mira del centro de la pantalla.

3. Herramientas utilizadas

El desarrollo de ARShooter ha requerido un conjunto de software especializado tanto para la programación de la aplicación como para la generación y manipulación de recursos multimedia. En todos los casos se ha optado por software gratuito cuyas licencias permitieran su uso en este tipo de trabajos académicos y la publicación de los mismos.

Las herramientas utilizadas han sido las siguientes:

- Unity 2017.2.0f3 Personal: entorno de desarrollo y motor de videojuegos multiplataforma. Ha servido de base para la realización del proyecto.

- ARToolkit 5.3.2: herramienta opensource para la realización de proyectos de realidad aumentada en distintas plataformas. Se ha utilizado el plugin específico para Unity y las librerías para Android proporcionadas con el plugin.

- Visual Studio Community 2015: IDE de programación gratuito de Microsoft que ontiene todo lo necesario para programar en C# y depurar los proyectos. La programación del proyecto se ha realizado íntegramente con esta herramienta.

- Blender: herramienta de modelado 3D que permite exportar diseños a formatos aceptados por Unity. Se ha utilizado para crear los modelos tridimensionales utilizados en el proyecto.

- Inkscape: herramienta de dibujo vectorial. Se ha utilizado para crear el título y el icono de la aplicación además de otros gráficos sencillos.

- Gimp: herramienta de dibujo y manipulación de imagen. Ha servido para generar ciertos efectos en el título de la aplicación.

- GitLab: servidor Git para gestionar los cambios durante el desarrollo de proyectos de mediana y gran envergadura. Ha permitido llevar un control de las modificaciones y revertir los cambios en situaciones anómalas.

- SourceTree: cliente de control de versiones Git. Se ha utilizado durante todo el desarrollo para guardar las modificaciones del código en el servidor de GitLab.

- ArgoUML: herramienta de modelado de diagramas UML programada en java. Este software se ha utilizado para generar los diagramas UML durante la fase de diseño del proyecto.

- LibreOffice: suite ofimática libre que se ha utilizado para generar la memoria de este proyecto.

4. Diseño

4.1 Diseño centrado en el usuario (DCU)

4.1.1 Usuarios y contexto de uso [Análisis]

Durante la fase de análisis se han realizado una serie de reuniones en grupo y entrevistas individuales con usuarios habituales de dispositivos móviles. La mayoría de los entrevistados eran aficionados a los videojuegos de distintos géneros que además utilizaban distintas plataformas (videoconsolas de salón, videoconsolas portátiles, PC, teléfonos móviles y tabletas).

Los datos recogidos durante esta fase de exploración y análisis indican que el tipo de usuario que utilizará la aplicación objeto de este proyecto tiene el siguiente perfil:

Nombre	Jugador
Características	• edad comprendida entre los 14 y los 40 años aunque el rango de edad puede variar ligeramente por ambos extremos • puede tener cierta preferencia por las actividades dinámicas frente a las actividades estáticas • utiliza el dispositivo móvil diariamente de forma intensiva, eso le lleva a tener mucha soltura con estos dispositivos • aficionado a los videojuegos con preferencia por los del género de acción • le gusta experimentar nuevas maneras de manejar y sentir los productos de ocio electrónico
Contextos de uso	Principalmente en espacios interiores (su casa, la casa de un amigo) durante su tiempo libre.
Análisis de tareas	1. Iniciar una partida. El usuario requiere de menús para configurar la partida (selección de nivel (pantalla), cambiar la dificultad,...) 2. Configurar la aplicación. El usuario estará interesado en modificar distintos aspectos relacionados con la percepción del juego (modificar el

	nivel del volumen de la música de fondo y los efectos de sonido) 3. Conseguir los objetivos propuestos en el juego (jugar, interactuar). El usuario requiere de un sistema que permita dar órdenes al juego. 4. Guardar las puntuaciones. El jugador deseará saber si ha mejorado.
Características o elementos descubiertos que tienen que ser presentes en la interfaz de la aplicación	1. Menú principal. Debe permitir: ◦ iniciar partidas ◦ configurar la aplicación ◦ acceder a puntuaciones ◦ salir del juego 2. Menú de opciones. Debe permitir la modificación de: ◦ el volumen de música de fondo y efectos de sonido ◦ el nombre del usuario que se mostrará en el ranking ◦ la dificultad del juego 3. Marcadores e indicadores (HUD). Se debe informar al usuario de forma gráfica sobre sus progresos, los marcadores y las posibilidades que tiene en cada momento 4. Controles para interacción del usuario. Se debe proporcionar una forma de interacción. 5. "Canvas" de visualización de la escena del juego. El juego debe mostrar el mundo virtual en el cual ocurre la acción. 6. Música de fondo y sonidos FX. Además de los gráficos, el sonido forma parte del mundo virtual y tiene un peso muy importante en la experiencia de juego.

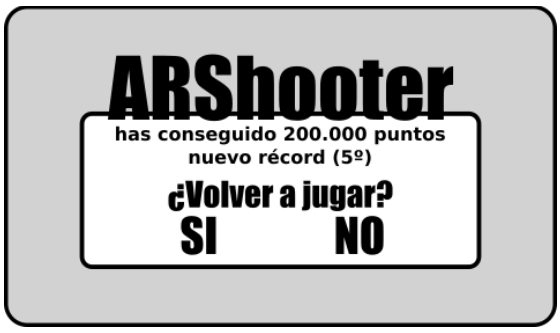
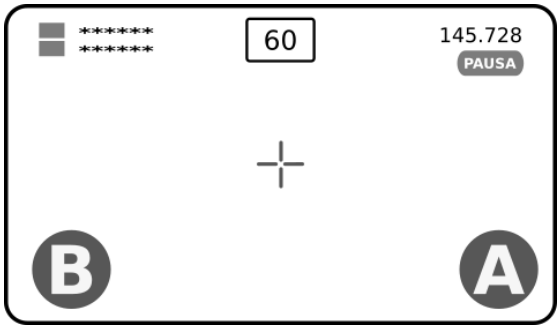
4.1.2 Diseño conceptual (escenarios de uso) [Diseño]

- Juan, un joven de 20 años aficionado a los FPS (first person shooter), dispone de un par de horas de ocio que quiere dedicar a los videojuegos. Esta vez ha decidido intentar batir su último récord en ARShooter. Se sienta cómodamente en el sofá frente a una mesita, sitúa el marcador en la mesa, saca su teléfono móvil, inicia la aplicación, carga el nivel que quiere jugar e inicia la partida. Disparo a disparo va batiendo su récord. Cuando decide finalizar, sale de la aplicación.

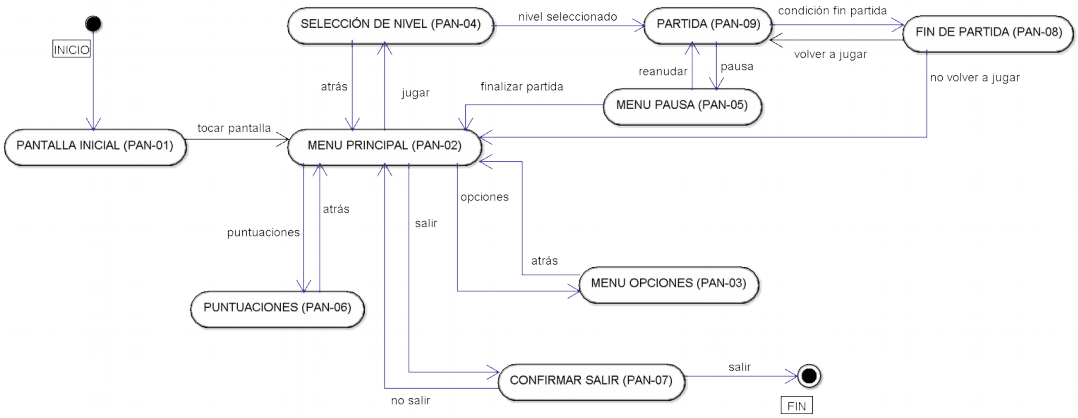
4.1.3 Prototipado [Diseño]

A continuación se muestra una tabla con esbozos y explicaciones de las pantallas que debe incluir el juego para cumplir con los requisitos mencionados en el apartado anterior.

	**PAN-01** Pantalla inicial a modo de presentación del juego. El usuario tiene que tocar la pantalla para continuar.
	**PAN-02** Menú principal del juego. Ofrece las siguientes opciones: 1. Iniciar una partida 2. Acceder a las opciones 3. Ver las puntuaciones 4. Salir de la aplicación
	**PAN-03** Menú de opciones del juego. Ofrece las siguientes opciones: 1. Cambiar el nombre del jugador 2. Modificar el volumen de la música 3. Modificar el volumen de los efectos de sonido 4. Modificar la dificultad del juego
	**PAN-04** Menú de selección de nivel (pantalla). El usuario debe tocar una de las imágenes en miniatura que representan los niveles. Aparece tras seleccionar la opción “jugar” del menú principal.

	**PAN-05** Menú “Pausa”. Aparece al pausar el juego. Desde esta pantalla el usuario puede: 1. Reanudar la partida (quitar pausa) 2. Finalizar la partida (y volver al menú inicial) 3. Modificar las opciones de sonido
	**PAN-06** Panel de récords. Muestra las mejores puntuaciones obtenidas en el juego. Se accede seleccionando la opción “Puntuaciones” del menú principal.
	**PAN-07** Mensaje para que el usuario confirme la acción de salir del juego. Evita salidas accidentales al presionar la opción “Salir” del menú principal.
	**PAN-08** Pantalla de fin de la partida. Muestra un mensaje indicando la puntuación obtenida, informa si se ha superado un récord y permite volver a jugar una partida en el mismo nivel y con la misma dificultad o volver al menú inicial.
	**PAN-09** Pantalla de juego. Muestra el escenario de juego. La cruz del centro se utiliza para apuntar a los enemigos. Los botones “A” y “B” disparan las armas primaria y secundaria respectivamente. En la parte superior se informa de la munición, el tiempo de la partida y la puntuación. El juego se pone en pausa presionando el botón “PAUSA” de la parte superior derecha de la pantalla.

El flujo de interacciones se muestra en la siguiente figura:



4.1.4 Evaluación [Evaluación]

Para la correcta evaluación del prototipo primeramente se realizará una ficha con datos relevantes sobre usuario/evaluador:

- ¿Como se llama?
- ¿Qué edad tiene?
- ¿Tiene conocimientos sobre tecnologías y gadgets?
- ¿Tiene experiencia jugando a videojuegos? ¿De qué género? ¿Qué le han parecido?
- ¿Utiliza aplicaciones en smartphones y/o tablets?
- En caso afirmativo, ¿Ha utilizado alguna vez un juego que involucre realidad aumentada?

Las tareas que debería realizar el usuario/evaluador son:

- Iniciar una partida
- Modificar las opciones del juego
- Mover el punto de vista del juego (cambiar la posición de la cámara)
- Intentar conseguir una alta puntuación

Tras la evaluación el usuario deberá contestar a unas preguntas sobre su reciente experiencia con el producto:

- ¿Ha tenido alguna dificultad en la navegación de los menús?
- ¿Cree que la dificultad del juego es correcta?

- ¿Le ha parecido un juego entretenido?
- ¿Ha experimentado problemas de posicionamiento de la cámara en cuanto a precisión o pérdida de posición? ¿En qué condiciones? (Detectar problemas de seguimiento del marcador)
- ¿Ha experimentado algún problema de rendimiento?

4.2 UML

4.2.1 Definición de casos de uso

CASO DE USO	
Id	CU-01
Nombre	INICIAR PARTIDA
Descripción	Inicia el trámite para comenzar una nueva partida
Precondición	-
Secuencia principal	1. El usuario selecciona la opción de jugar una nueva partida
Errores/Alternativas	-
Postcondición	El sistema espera a que se seleccione el nivel
Notas	

CASO DE USO	
Id	CU-02
Nombre	SELECCIONAR NIVEL
Descripción	El usuario debe seleccionar un nivel al que jugar
Precondición	El usuario ha seleccionado la opción de jugar una nueva partida
Secuencia principal	1. El usuario selecciona el nivel en el que jugará
Errores/Alternativas	-
Postcondición	El sistema muestra la pantalla de juego, la partida está en curso en el nivel seleccionado.
Notas	

CASO DE USO	
Id	CU-03
Nombre	JUGAR
Descripción	El usuario interactúa con los controles del juego
Precondición	La partida está iniciada
Secuencia principal	1. El usuario apunta con el dispositivo
Errores/Alternativas	1a El usuario dispara el arma principal 2a El usuario dispara el arma secundaria
Postcondición	La partida está en curso y se ha realizado la acción solicitada por el usuario con posibles consecuencias
Notas	

CASO DE USO	
Id	CU-04
Nombre	PAUSAR PARTIDA
Descripción	El usuario puede pausar una partida en curso
Precondición	La partida está iniciada, no está en pausa
Secuencia principal	1. El jugador presiona el botón de pausa/menú
Errores/Alternativas	-
Postcondición	Se muestra el menú “pausa”, el juego pasa al estado “en pausa”
Notas	-

CASO DE USO	
Id	CU-05
Nombre	REANUDAR PARTIDA
Descripción	El usuario puede reanudar la partida después de pasar por el estado “en pausa”
Precondición	La partida está iniciada y en estado “en pausa”está en pausa
Secuencia principal	1. El jugador presiona la opción “Reanudar” en el menú
Errores/Alternativas	-
Postcondición	La partida vuelve al estado “en ejecución”. El menú “pausa” ya no se muestra
Notas	-

CASO DE USO	
Id	CU-06
Nombre	IR AL MENÚ
Descripción	El usuario vuelve al menú principal
Precondición	La partida no está en el estado “en ejecución”
Secuencia principal	1. El usuario selecciona “atrás” en una ventana con esta opción o “no volver a jugar” al finalizar una partida.
Errores/Alternativas	-
Postcondición	El sistema muestra el menú principal
Notas	

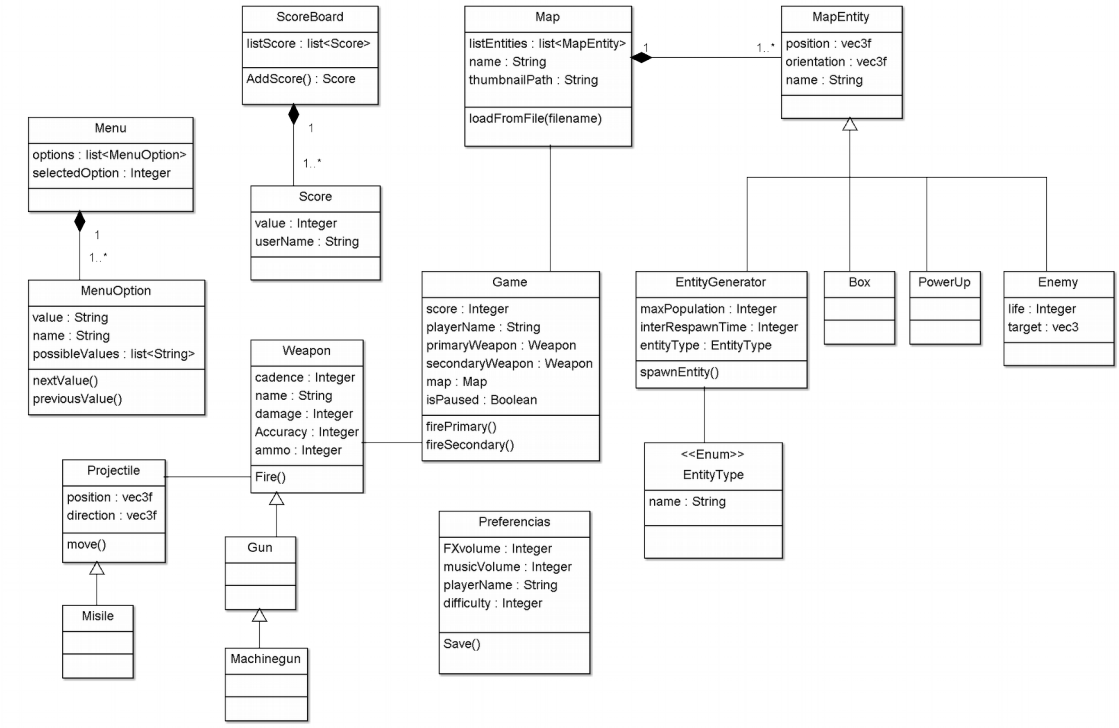
CASO DE USO	
Id	CU-07
Nombre	MODIFICAR OPCIONES
Descripción	El usuario puede modificar las opciones del juego
Precondición	Se muestra el menú “opciones” o el menú “pausa”
Secuencia principal	1. El usuario selecciona una opción del menú 2. El usuario toca el botón correspondiente para que se modifique el valor de dicha opción tantas veces como sea necesario para establecer el valor que desea
Errores/Alternativas	
Postcondición	La opción puede haber cambiado de estado
Notas	

CASO DE USO	
Id	CU-08
Nombre	VER PUNTUACIONES
Descripción	El usuario puede ver las puntuaciones del juego
Precondición	Se muestra el menú principal
Secuencia principal	1. El usuario selecciona la opción “ver puntuaciones”
Errores/Alternativas	
Postcondición	Se muestra la lista de puntuaciones
Notas	

CASO DE USO	
Id	CU-09
Nombre	SALIR
Descripción	El usuario puede salir completamente del juego
Precondición	Se muestra el menú principal
Secuencia principal	1. El jugador selecciona la opción salir 2. Se muestra el mensaje de confirmación para salir del juego 3. El jugador confirma que desea salir del juego
Errores/Alternativas	3a. El jugador contesta “No” 4. Se vuelve a mostrar el menú principal
Postcondición	El juego ya no está en ejecución
Notas	

4.2.2 Diseño de la arquitectura de la aplicación

El diagrama de clases simplificado correspondiente a las clases del negocio de ARShooter son las siguientes:



En este diagrama no se muestran las clases referentes a los scripts necesarios para controlar el motor de Unity, los controladores de las ventanas/pantallas y componentes especiales propios de Unity (GameObjects y MonoBehaviours).

5. Implementación

5.1 Pantallas de la aplicación

A continuación se expone el acabado final de las pantallas de la aplicación junto con la referencia que se le dio en el esbozo del apartado de diseño:

PAN-01 (Pantalla Inicial)

PAN-02 (Menú principal)

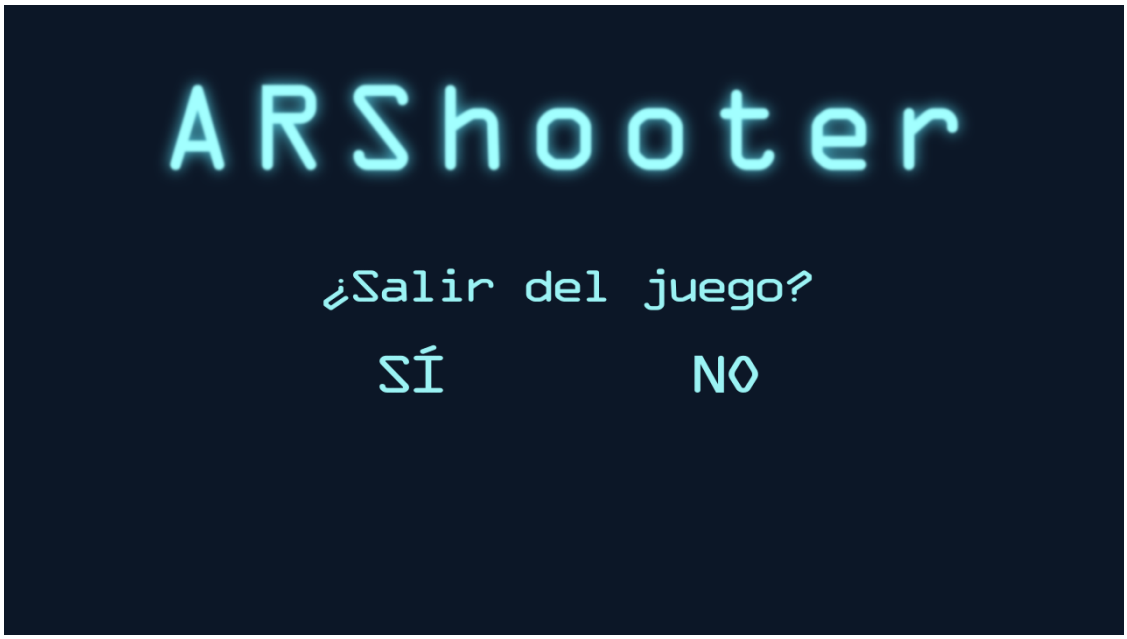
PAN-03 (Menú Opciones)

PAN-04 (Selección De Nivel)

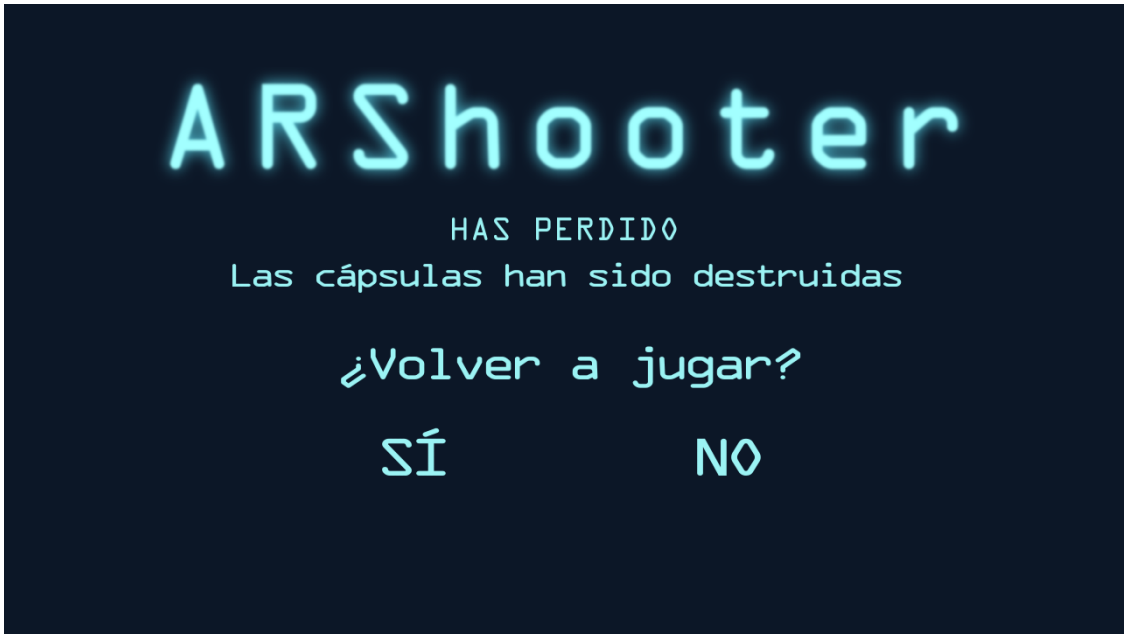
PAN-05 (Menú Pausa)

PAN-06 (Tabla De Puntuaciones)

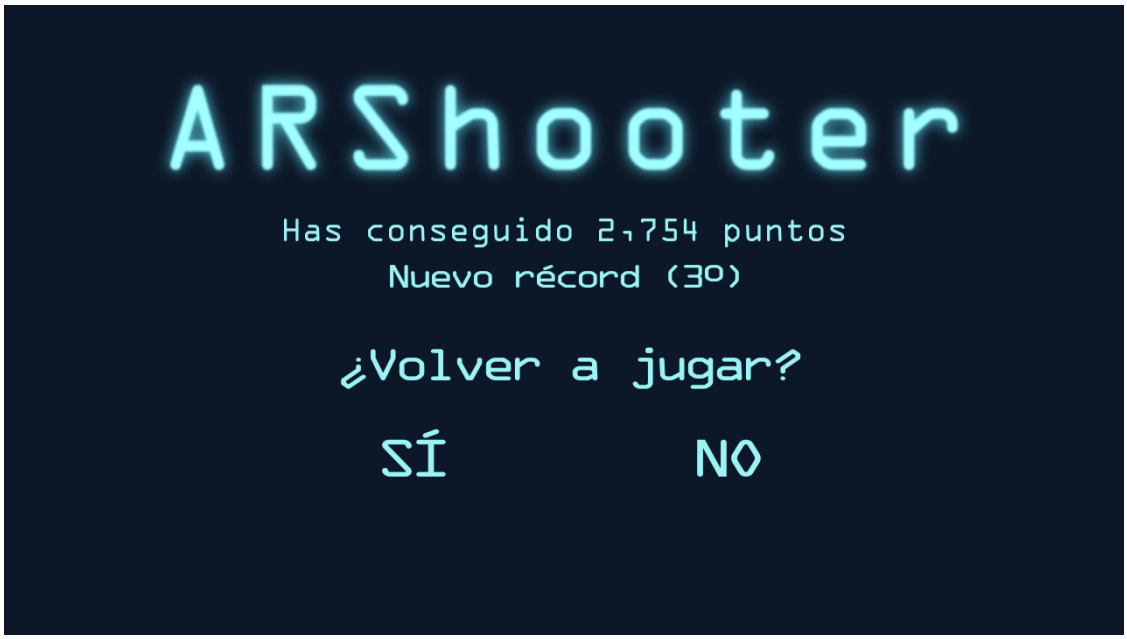
PAN-07 (Confirmación Salir)



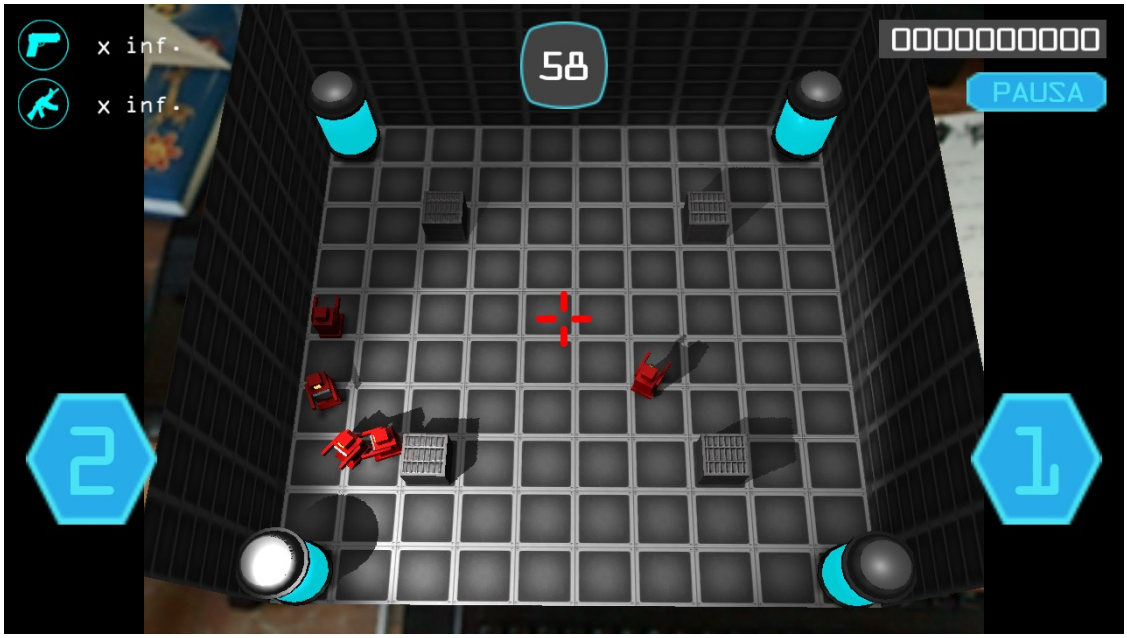
PAN-08 (Fin De Partida - Perdedor)



PAN-08 (Fin De Partida - Ganador)



PAN-09 (Pantalla De Juego)



5.2 Decisiones tomadas durante la implementación

- Todas las clases que representan entidades o deben tenerse en cuenta en el bucle de ejecución de Unity derivan de la clase MonoBehavior de Unity. Esto ha liberado a las clases que debían codificarse de tener que guardar los datos relativos a posiciones y rotaciones y tener que implementar complejas operaciones con vectores y matrices ya que de esta manera vienen dadas por Unity.

- Se ha intentado, en la medida de lo posible, separar las pantallas de la aplicación en escenas de Unity independientes y se han creado scripts separados para controlar el comportamiento de cada una de ellas a modo de controlador (patrón MVC).

- Los archivos de datos que utiliza la aplicación para guardar preferencias, ranquing y niveles de juego se almacenan en la memoria externa del dispositivo. Esto es debido a que el "player" de Unity no permite acceder a la memoria interna con permisos de escritura de forma fácil. Para solventarlo se ha utilizado una estrategia que consiste en dos pasos:

    1. distribuir los archivos en el APK de la aplicación con sus valores por defecto (carpeta "/Resources")
    2. durante la ejecución, almacenar los datos en la memoria externa donde la aplicación podrá manipularlos sin problemas

- Se ha decidido que los archivos de datos se almacenen en texto plano en vez de en archivos estructurados como xml o json ya que no se precisan elementos anidados y dichos elementos no deben contener demasiados atributos. Así, finalmente, se ha guardado un ítem por línea y se han utilizando los carácteres ':' y ',' como separadores.

- Se ha creado un control personalizado reutilizable que permite seleccionar un valor entre varios posibles mediante la navegación secuencial de dichos valores. Este control no existe en Unity, la alternativa disponible más parecida en Unity es un control de tipo "dropdown" que no se ajusta al diseño inicial de las pantallas del proyecto. La facilidad que ofrece Unity para crear nuevos componentes a través de la composición ha hecho fácil la creación de dicho control personalizado que finalmente se ha utilizado en los siguientes casos:

    1. selección del volumen de la música (valores de 0 a 10)
    2. selección del volumen de los sonidos FX (valores de 0 a 10)
    3. selección del nivel de dificultad del juego (valores: "Fácil", "Normal", "Difícil")

- Al iniciar el proyecto se había planteado crear un sistema de inteligencia artificial que permitiera a los enemigos encontrar caminos dentro de una escena cambiante (algunos objetos podrían cambiar de posición debido a que reciben impactos, los enemigos podrían obstruir el camino de otros enemigos, etc.). Crear el algoritmo necesario para que todo funcionara correctamente supondría un consumo de tiempo demasiado elevado. La solución la aportó el propio Unity, los objetos del juego se pueden configurar de modo que el mismo motor crea una malla de navegación de la escena (NavMesh) que se actualiza en tiempo real teniendo en cuenta las modificaciones de la escena y es capaz de mover los actores dentro de esta evitando obstáculos. Este es el sistema que se ha utilizado finalmente debido a su facilidad de uso y los buenos resultados que ofrece.

- Para el sistema de física no ha habido ninguna duda ya que Unity integra uno que funciona muy bien y es muy sencillo de configurar. Simplemente se tuvo que especificar la forma de la superficie de impacto de los objetos (generalmente una forma más simple que la geometría del objeto para facilitar los cálculos) y algunos condicionantes como bloqueos en la rotación en algún eje, su peso, etc.

- El apartado de iluminación inicialmente dio bastantes problemas debido a que las primeras pruebas se realizaron en PC y posteriormente se probó con dispositivos Android pudiéndose observar grandes diferencias utilizando los mismo parámetros debido a las particularidades de los dispositivos. Este fue el caso de una luz a modo de linterna que que estaba situada en la cabeza de los robots, la acumulación de estas luces en PC daba lugar a un punto sobre-iluminado en el suelo mientras que en el dispositivo Android de prueba quemaba la totalidad del suelo de la escena (no solo la zona de aplicación). Finalmente se optó por eliminar dicha luz, esta decisión agilizó el juego restando carga a la CPU y la GPU.

5.3 Problemas encontrados durante la implementación

Como se comenta en apartados anteriores, el proyecto utiliza la API de realidad aumentada ARToolkit mediante un plugin para Unity. El comportamiento normal del plugin es que cuando inicia la app, se realiza una conexión con un servidor de ARTookit para descargar los parámetros de calibrage de la cámara del dispositivo. Así se evita que cada usuario tenga que calibrar su cámara, ya que es un proceso un poco largo y complicado. En caso de fallar este paso, debería utilizar unos parámetros por defecto.

El servidor proporcionado por ARToolkit ha dejado de funcionar correctamente durante las últimas semanas de la fase de implementación (o no devuelve datos o devuelve datos incorrectos) y el plugin no resuelve bien esta situación dando como resultado que el dispositivo no procese el video de la cámara. Esto anula totalmente la realidad aumentada y muestra una pantalla totalmente negra en su lugar.

Hay algunos hilos en el foro oficial y en StackOverflow sobre esta situación pero no hay una respuesta resolutiva por parte de los responsables del proyecto, todo indica que están acabando la nueva versión de la API (ARToolkit6) y no están invirtiendo recursos en la versión actual (ARToolkit 5.3.x).

Según el foro de soporte del proyecto en GitHub, por ahora solo hay dos soluciones posibles:

1. Modificar el plugin impidiendo que intente descargar el archivo con los datos para calibrar la cámara y en su lugar cargar los datos por defecto

2. Activar el “modo vuelo” del dispositivo móvil

En este proyecto se ha optado por desactivar manualmente las conexiones del dispositivo utilizando el “modo vuelo” de Android. Esta no es la mejor solución (aunque ha funcionado perfectamente en los terminales de prueba) y sería preferible modificar el plugin para saltar este paso en el caso de que no resuelvan el problema del servidor, pero esto comportaría invertir demasiado tiempo y recursos en algo que no forma parte del desarrollo que se planteó en el TFM.

Se puede obtener información detallada sobre este error en los siguientes enlaces:

1. https://github.com/artoolkit/arunity5/issues/190 (Hilo del foro de GitHub de ARToolkit donde se menciona el problema con intervención de un responsable de ARToolkit)

2. https://github.com/artoolkit/artoolkit5/issues/90 (Hilo del foro de GitHub de ARTookit donde se menciona el "workarround" modificando el código del API. Solución propuesta por un responsable de ARToolkit)

5.4 Estado del proyecto al finalizar la fase de implementación (13/12/2017)

Se completaron la mayoría de tareas relacionadas con la implementación de la aplicación. Concretamente se cumplieron los siguiente objetivos:

- Reconocimiento del marcador y representación de escena 3D desde el punto de vista del dispositivo (observador)
- Desarrollo de las distintas pantallas y menús
- Generación de modelos 3D
- Aplicación de luces y materiales
- Carga y salvado de preferencias de usuario
- Carga y salvado de puntuaciones
- Carga de niveles de juego a partir de archivos de texto
- Reproducción de efectos sonoros y música de fondo
- Posibilidad de modificar el volumen de los sonidos
- Pausar y reanudar partida
- Disparar a los elementos del escenario
- Mover enemigos por la escena desde un punto inicial hasta su objetivo
- Control de puntuaciones y duración de la partidas
- Implementación de 2 armas, una principal y otra secundaria
- Aplicación de leyes físicas en objetos de la escena 3D

Los objetivos sin cumplir fueron los siguientes:

- Completar el comportamiento de los enemigos
- Añadir el arma “Misil”
- Añadir power-ups para mejorar la puntuación

5.5 Acciones necesarias para finalizar correctamente el proyecto

En el planteamiento inicial del proyecto se reservaron unos días de la siguiente fase para terminar de refinar la aplicación y realizar pruebas. Puesto que ya se verificó el correcto funcionamiento de las funcionalidades implementadas, los siguientes días se utilizan íntegramente para completar la mayoría de los objetivos que faltan y a la vez comprobar que estas últimas implementaciones funcionen correctamente.

5.6 Pruebas realizadas

La depuración de la aplicación se realizó inicialmente conectando la ejecución la aplicación en PC con el depurador de Visual Studio pero este proceso dejó de funcionar por causas que no se llegaron a descubrir. Al no encontrar una solución de forma rápida se optó por implementar un sistema de mensajes por pantalla en tiempo real en la misma aplicación que puede activarse y desactivarse desde el proyecto de Unity.

Durante el desarrollo se realizaron las siguientes comprobaciones:

- Verificar los impactos de los disparos (coordenadas y objetos afectados). Resultados satisfactorios tras las últimas pruebas.

- Comprobar los atributos de los objetos antes y después de recibir impactos. Resultados satisfactorios tras las últimas pruebas.

- Comprobar la correcta asignación de los objetivos de los enemigos. Resultados satisfactorios tras las últimas pruebas.

- Comprobar que el control del tiempo funcionara correctamente tras pausar y reanudar el juego varias veces. Resultados satisfactorios tras las últimas pruebas.

- Comprobar que todos los elementos del juego se pausaran y reanudaran correctamente incluidos los afectados por la física. Resultados satisfactorios tras las últimas pruebas.

- Verificar la correcta aplicación de las texturas sobre los objetos tridimensionales. Resultados satisfactorios tras las últimas revisiones visuales.

- Verificar que los elementos gráficos adoptaran una posición correcta al cambiar a pantallas de distintos tamaños. Resultados satisfactorios tras las últimas revisiones visuales.

- Comprobar que el rendimiento del juego fuera el adecuado con terminales un poco antiguos. Resultados satisfactorios tras las últimas pruebas.

- Comprobar que la reproducción de varios sonidos no produzcan demasiada distorsión. Resultados satisfactorios tras las últimas revisiones auditivas.

- Utilizar el marcador con distintos tipos de iluminación y moviendo el terminal a distintas velocidades. Resultados relativamente satisfactorios tras las últimas pruebas visuales. Se puede mejorar el sistema tratando la imagen capturada antes de pasarla a ARToolkit y utilizando una aproximación de la posición del marcador en función de los últimos movimientos detectados.

- Verificar que los resultados de las partidas que deberían entrar en el tablón de puntuaciones se guarden y recuperen correctamente. Resultados satisfactorios tras las últimas pruebas.

- Verificar que los valores de las opciones del juego se guarden y recuperen correctamente. Resultados satisfactorios tras las últimas pruebas.

- Comprobar la carga correcta de los niveles del juego. Resultados satisfactorios tras las últimas pruebas.

En el código fuente de la aplicación se pueden observar algunas de las pruebas comentadas.

6. Instrucciones de compilación y ejecución

6.1 Código fuente. El proyecto de Unity

En el archivo "zip" de la entrega se adjunta el proyecto completo realizado en Unity. Dicho proyecto contiene los archivos necesarios para la utilización de ARToolkit (plugin para realidad aumentada) tanto para la compilación como para la ejecución en dispositivo Android.

6.2 Preparación del entorno

La preparación del entorno de desarrollo y pruebas consiste en los siguientes pasos:

- Instalar y configurar Android SDK
- Instalar y configurar Java Development Kit (JDK)
- Instalar en el PC los drivers de un dispositivo Android y configurar en dicho dispositivo las opciones de depuración
- Instalar Unity con licencia “Personal” (durante el desarrollo del proyecto he utilizado la versión 5.5.0f3 y posteriormente 2017.2.0f3) y configurar las rutas para utilizar "Android SDK" y "Java Development Kit (JDK)" (esto último lo pedirá Unity automáticamente)

Las instalaciones y configuraciones de los pasos mencionados anteriormente se realizan de forma sencilla y no requieren de mayor explicación. Para más información sobre cada uno de los pasos dirigirse a las URLs indicadas para cada programa en el siguiente apartado.

6.3 Compilación

6.3.1 Requisitos

La compilación se realizará en un PC con sistema operativo Windows (versiones: 7 SP1+, 8, 10) utilizando Unity. Los pasos para realizarla en Mac (versiones: Mac OS X 10.8+) no deberían ser distintos a los especificados en este documento.

Requisitos hardware: Unity no ofrece información extensa sobre este aspecto, solo remarca los datos de la GPU mencionando que requiere una tarjeta gráfica con DX9 (modelo de shader 3.0) o DX11 con capacidades de funciones de nivel 9.3.

Requisitos software:

- Unity 2017.2.0f3 “Personal” (http://unity3d.com). Esta es la versión que he utilizado durante el desarrollo aunque también debería funcionar en versiones superiores
- Android SDK (https://developer.android.com/studio/index.html)
- Java Development Kit (JDK) (http://www.oracle.com/technetwork/java/javase/downloads/index.html)

6.3.2 Proceso de compilación

Paso1: cargar el proyecto en Unity mediante la opción del menú
"File" > "Open project ..."

Paso2: especificar la plataforma destino utilizando el menú
"File" > "Build Settings ..."

Paso3: en la ventana "Build Settings" que se ha abierto en el paso anterior, presionar el botón "Build"

6.4 Ejecución

6.4.1 Requisitos

La ejecución se realizará en un dispositivo físico Android (versiones: Android 2.3.1 Gingerbread (API Level 9) o superior) aunque podría ejecutarse en una variedad de sistemas bastante amplia (véase la documentación de Unity). El dispositivo debe disponer de cámara y capacidades de renderizado 3D mediante OpenGL ES.

Para este proyecto se recomienda la utilización de dispositivos Android actuales de gama media o alta. Esto dará más potencia y se traducirá en mayor fluidez ya que el juego hace un uso considerable de los procesadores (CPU, GPU) y de la memoria del dispositivo.

Al ser un producto de realidad aumentada requiere de unos marcadores que permitan posicionar el mundo virtual sobre el mundo real. En este caso el marcador es un patrón de imagen que se incluye en el apartado “12 Patrón de imagen (marcador)”. Dicho patrón deberá imprimirse en un papel para mostrarlo a la cámara del dispositivo durante la ejecución de la partida como se explica en el apartado “7.1 Antes de jugar”.

6.4.2 Proceso de ejecución

La ejecución desde Unity se puede realizar por cualquiera de los dos métodos siguientes:

-Método1:

Paso1: cargar el proyecto en Unity mediante la opción del menú
"File" > "Open project ..."

Paso2: especificar la plataforma destino utilizando el menú
"File" > "Build Settings ..."

Paso3: en la ventana "Build Settings" que se ha abierto en el paso anterior, presionar el botón "Build And Run"

-Método 2

Paso1: cargar el proyecto en Unity mediante la opción del menú "File" > "Open project ..."

Paso2: Seleccionar la opción del menú "File" > "Build And Run"

-Método 3

Con la aplicación instalada, tocar el icono de la aplicación:

7. Manual de usuario

7.1 Objetivo del juego

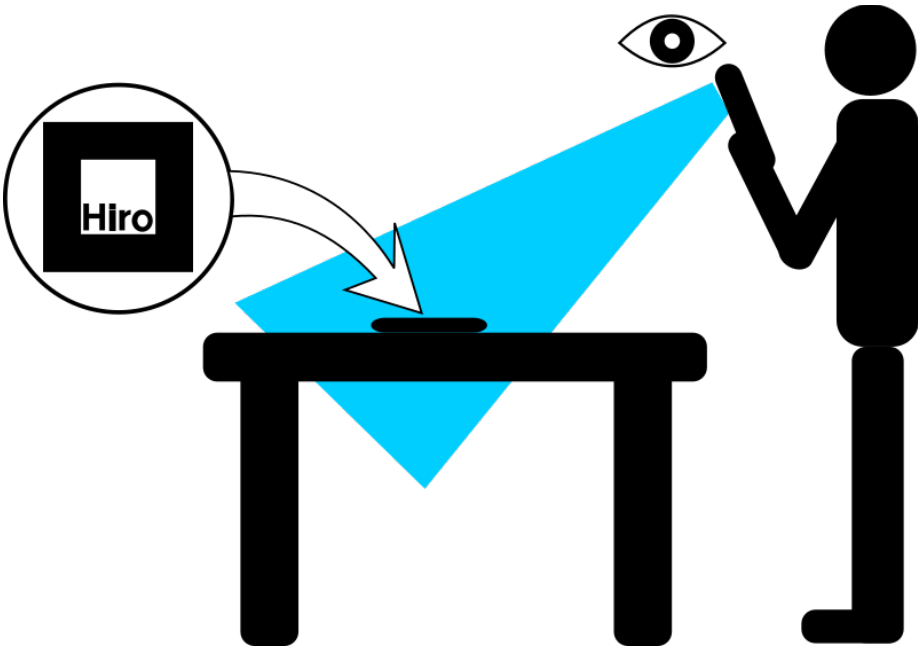
El jugador debe impedir que los robots que aparezcan en la escena destruyan las cápsulas azules del escenario. Cada disparo certero otorga puntos al jugador en función del daño infligido al enemigo. Para terminar la partida con éxito, el jugador debe conservar al menos una cápsula al finalizar los 60 segundos que dura la partida.

7.2 Antes de jugar

Debido al error expuesto en el apartado "5.3 Problemas encontrados durante la implementación" es imperativo que se active el "modo vuelo" de Android antes de ejecutar la aplicación. En caso contrario el jugador verá una pantalla negra donde debería ver la pantalla de juego.

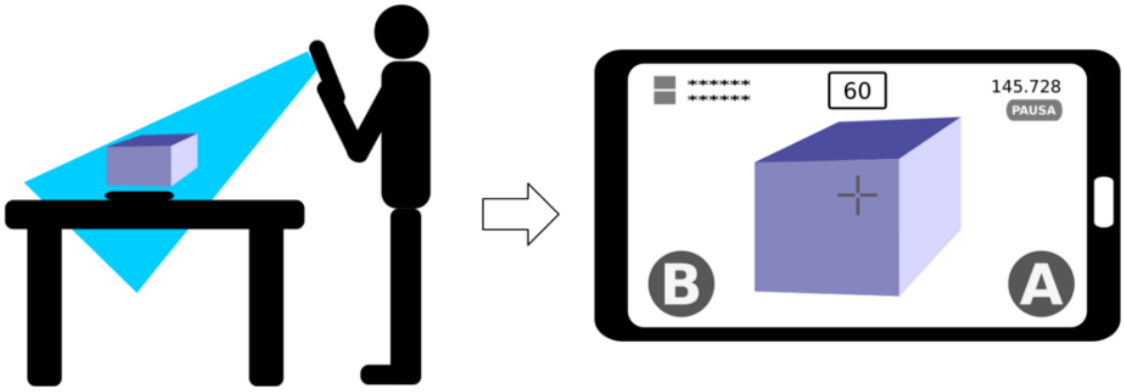
Para poder utilizar ARShooter, el usuario debe imprimir la imagen de referencia (en adelante "marcador") que se adjunta en el apartado "12 Patrón de imagen (marcador)". También se adjunta un archivo ".pdf" con la imagen.

El marcador debe situarse sobre una superficie horizontal (por ejemplo una mesa) y tiene que estar dentro del campo de visión de la cámara trasera del dispositivo móvil como se muestra en la siguiente imagen:



7.3 Apuntar a los enemigos

Durante el juego, el sistema analiza la imagen de la cámara para determinar la posición y orientación del usuario respecto al marcador y dibuja el escenario 3D en la pantalla del dispositivo como si éste saliera del marcador y fuera observado desde la cámara del dispositivo.



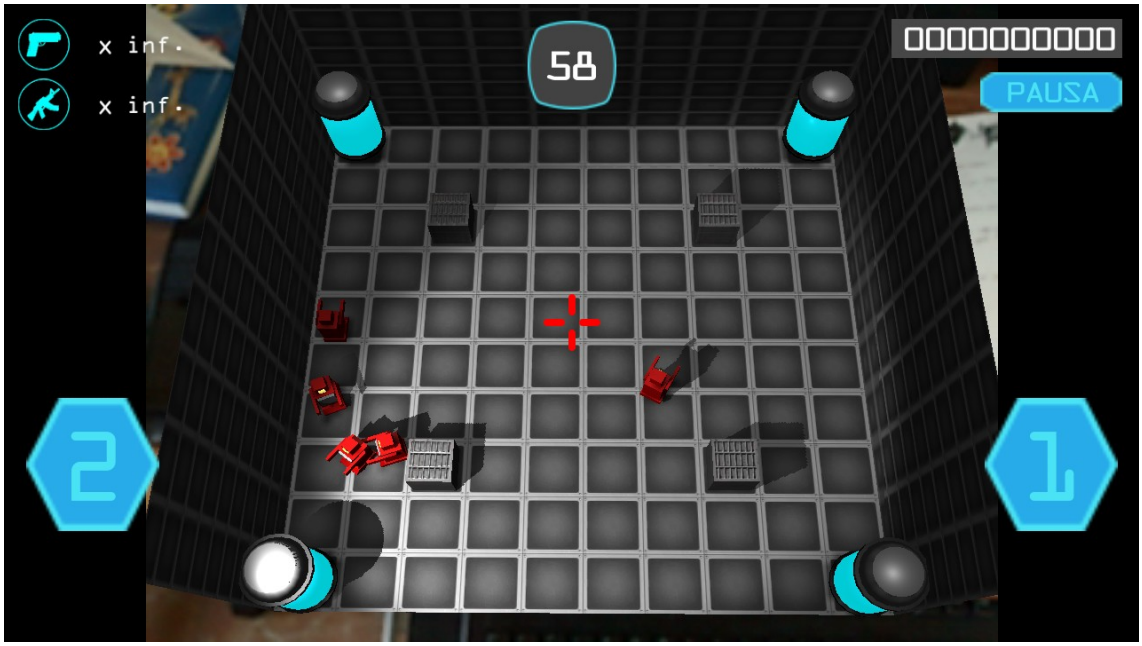
Teniendo en cuenta lo anterior, para apuntar basta con desplazar o girar el dispositivo ligeramente manteniendo el marcador dentro del campo de visión de la cámara. Estos movimientos hacen que la cámara virtual que enfoca la escena 3D se desplace imitando los movimientos que se realizan con el dispositivo móvil. De esta manera se puede hacer coincidir el punto de mira del centro de la pantalla con un enemigo.

Ejemplos de movimiento de cámara:

- Acercar el dispositivo hacia el marcador mostrará la escena de más cerca (acerca la cámara virtual)
- Girar el dispositivo sobre el eje vertical hacia un lado u otro es similar a girar la cabeza sobre el eje del cuello (gira la cámara virtual)

7.4 Controles e indicadores visuales

La pantalla de juego está provista de diversos indicadores visuales que ofrecen información sobre la partida e incluye tres botones de acción para que el usuario pueda interactuar con el juego.



Elementos visuales:

1. Contador de tiempo restante: muestra el tiempo que falta para terminar la partida. Está situado en la parte superior central de la pantalla.

2. Contador de puntuación: muestra la puntuación obtenida por el jugador durante la partida. Está situado en la parte superior derecha de la pantalla.

3. Punto de mira: se utiliza para apuntar, indica el punto exacto donde se realizará el disparo. Está situado en la parte central de la pantalla.

4. Indicadores de armas. Se muestran en la esquina superior izquierda de la pantalla. Hay dos indicadores de este tipo:

    1. Información del arma primaria: muestra un icono que representa el arma seguido de la cantidad de munición. En este caso indica que el arma primaria es la pistola y tiene munición infinita. Es el indicador superior.

    2. Información del arma secundaria: muestra un icono que representa el arma seguido de la cantidad de munición. En este caso indica que el arma es la ametralladora y tiene munición infinita. Es el indicador inferior.

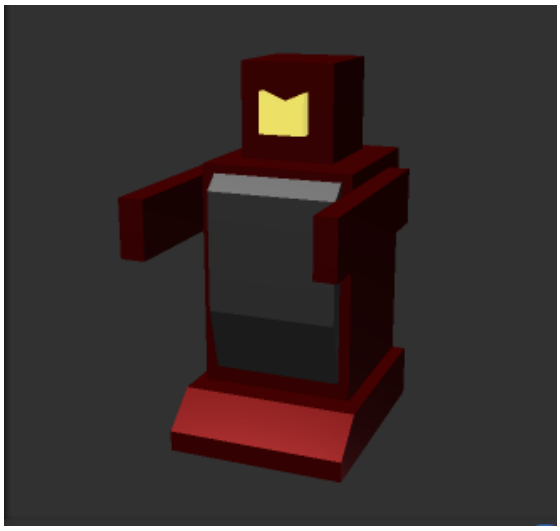
Controles:

1. Botón “Pausa”: detiene el juego temporalmente y muestra un menú para modificar las opciones de sonido, reanudar la partida y salir de la partida. Está situado en la parte superior derecha de la pantalla bajo la puntuación.

2. Botón de disparo del arma primaria: efectúa un disparo con el arma primaria (pistola). Está situado en la parte inferior derecha de la pantalla.

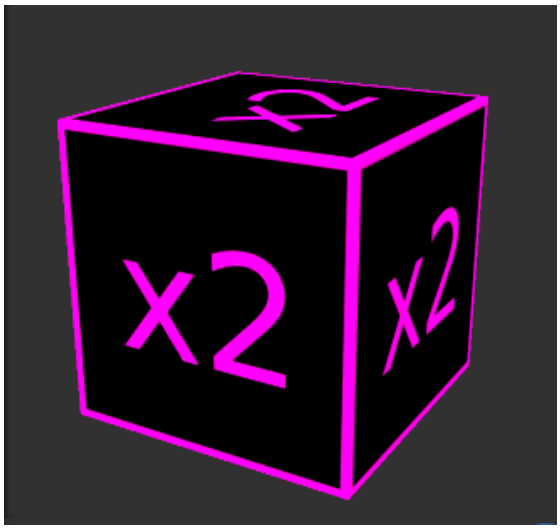
3. Botón de disparo del arma secundaria: efectúa disparos con el arma secundaria (ametralladora). Está situado en la pare inferior izquierda de la pantalla

7.5 Objetos 3D del escenario de juego

Enemigos	
	Son unos robots de color rojo que irán apareciendo en la escena con el único objetivo de destruir las cápsulas de agua presentes en la pantalla. Buscarán caminos hacia su objetivo sorteando las cajas que encuentren por medio y se inmolarán al llegar a su objetivo.

Cápsulas de agua	
	Son los objetos que debe defender el jugador. Perderlas todas hará que finalice la partida. Recibirán daños cada vez que un robot se inmole cerca de ellas y cada vez que reciban un disparo del jugador. Se vuelven de color rojo cuando están a punto de ser destruidas.

Cajas	
	Son obstáculos que impiden el paso de los robots enemigos. Están afectadas por las leyes físicas y si reciben el impacto de un disparo rebotarán por la pantalla. Al recibir muchos impactos desaparecerán.

PowerupX2	
	Modificador de doble daño. Multiplica por dos el daño que infligen los disparos de las armas. Aparece en un lugar aleatorio a mitad de la partida. Se recoge al recibir un disparo. Desaparece cinco segundos después de aparecer. Es una buena ventaja si los disparos aciertan en los enemigos pero una gran desventaja si los disparos terminan impactando en las cápsulas de agua.

8. Formato de archivo de los niveles del juego

Los niveles del juego se almacenan en archivos de texto con codificación UTF-8. Contienen las definiciones de todos los elementos que aparecen en la escena. Se especifica un elemento por línea y cada línea contiene todos los datos necesarios para definir el objeto (su tipo, posición, rotación y otras características específicas de cada tipo de elemento), todos los valores están separados por comas.

Los elementos que se pueden encontrar en los archivos de niveles son:

- Cápsula
- Punto de regeneración de enemigos
- Caja
- Línea de comentarios

La especificación de cada elemento se muestra en las siguientes tablas:

Nombre: Cápsula
Descripción: Cápsula que el jugador deberá defender de los enemigos
Atributos: • TipoEntidad: “CAPSULA” • PosiciónX: float • PosiciónY: float • PosiciónZ: float • RotaciónY: float
Formato: *CAPSULA, PosiciónX, PosiciónY, PosiciónZ, RotaciónY*
Ejemplo: *CAPSULA, -0.18, 0.0, 0.18, 0.0*

Nombre: Caja
Descripción: Caja que impedirá el paso de los enemigos obligándoles a buscar otro camino para llegar a su objetivo
Atributos: • TipoEntidad: “CAJA” • PosiciónX: float • PosiciónY: float • PosiciónZ: float • RotaciónY: float
Formato: *CAJA, PosiciónX, PosiciónY, PosiciónZ, RotaciónY*
Ejemplo: *CAJA, 0.1, 0.0, 0.1, 0.0*

Nombre: Punto de regeneración de enemigos
Descripción: Lugar en el que irán apareciendo nuevos enemigos durante la partida. Este elemento tiene tres limitaciones: 1. Nunca habrá más enemigos vivos generados en este punto que los especificados por el atributo “PoblaciónMáxima” 2. El punto de regeneración dejará de ser útil al alcanzar el número especificado por el atributo “GeneracionesMáximas” 3. No se generarán nuevos enemigos en un tiempo inferior al especificado por el atributo “TiempoRecarga”
Atributos: • TipoEntidad: “PUNTO_REGENERACION_ENEMIGOS” • PosiciónX: float • PosiciónY: float • PosiciónZ: float • RotaciónY: float • PoblaciónMáxima: entero • GeneracionesMáximas: entero • TiempoRecarga: float (especificado en segundos)
Formato: *PUNTO_REGENERACION_ENEMIGOS, PosiciónX, PosiciónY, PosiciónZ, RotaciónY, PoblaciónMáxima, GeneracionesMáximas, TiempoRecarga*
Ejemplo: *PUNTO_REGENERACION_ENEMIGOS, -0.12, 0.0, 0.0, 0.0, 2, 10, 3.0*

Nombre: Comentario
Descripción: Línea que contiene un comentario. Estos no se procesan al cargar el nivel.
Atributos: • Marca de comentario: “#” • Comentario: texto del comentario
Formato: *#texto_del_comentario*
Ejemplo: *#esto es un comentario*

A continuación se muestra como ejemplo el contenido de un archivo de nivel de juego:

```
#TipoEntidad, posicionX, posicionY, posicionZ, rotacionY

CAPSULA, -0.18, 0.0, 0.18, 0.0
CAPSULA, -0.18, 0.0, -0.18, 0.0
CAPSULA, 0.18, 0.0, 0.18, 0.0
CAPSULA, 0.18, 0.0, -0.18, 0.0
PUNTO_REGENERACION_ENEMIGOS, -0.12, 0.0, 0.0, 0.0, 2, 10, 3.0
PUNTO_REGENERACION_ENEMIGOS, 0.0, 0.0, 0.12, 0.0, 2, 10, 3.0
PUNTO_REGENERACION_ENEMIGOS, 0.0, 0.0, 0.0, 0.0, 2, 10, 3.0
PUNTO_REGENERACION_ENEMIGOS, 0.0, 0.0, -0.12, 0.0, 2, 10, 3.0
PUNTO_REGENERACION_ENEMIGOS, 0.12, 0.0, 0.0, 0.0, 2, 10, 3.0
CAJA, 0.1, 0.0, 0.1, 0.0
```

```
CAJA, -0.1, 0.0, 0.1, 0.0
CAJA, 0.1, 0.0, -0.1, 0.0
CAJA, -0.1, 0.0, -0.1, 0.0
```

9. Estado actual del proyecto

Siendo estrictos con la definición inicial del proyecto, el estado actual del mismo a día 3 de Enero de 2018 es del 95% completado. Ha faltado implementar algunas armas más para que el usuario tuviera un mayor abanico de posibilidades y se enriqueciera el gameplay. Además ha faltado implementar algunos powerups que modificaran un poco el ritmo del juego. Aún así, las funcionalidades más importantes del juego han sido implementadas y funcionan correctamente.

Se tuvo que optar por la reducción de funcionalidades en el momento en que aparecieron problemas con la API ARToolkit y parte del tiempo disponible para la implementación se tuvo que dedicar a buscar soluciones para poder ejecutar la aplicación.

10. Líneas de trabajo futuro

Teniendo en cuenta el estado del proyecto y las posibilidades que tiene, sería lógico pensar que los siguientes pasos que deberían darse sean:

- añadir nuevas armas:
  - Misiles
  - Granadas
  - Escopeta
  - …

- añadir nuevos powerups:
  - Acelerar el juego
  - Desacelerar el juego
  - Duplicar el tiempo
  - Inmortalidad de las cápsulas durante un tiempo limitado
  - …

- añadir nuevos niveles dotando la pantalla de distintas alturas, rampas, escalones, …

- mejorar la inteligencia artificial y utilizar las nuevas funciones de Unity para buscar caminos

- añadir nuevos elementos en el juego:
  - enemigos de distintos tipos

11. Conclusiones

Pese a la experiencia previa con Unity en la asignatura “Desarrollo de videojuegos para dispositivos móviles” y conocer el lenguaje C#, la curva de aprendizaje con la API de Unity y la API de ARToolkit han supuesto una dificultad mayor a la esperada inicialmente.

La unión de Unity y ARToolkit es una alternativa gratuita y muy potente en el desarrollo de juegos y realidad aumentada para dispositivos móviles Android y para otras plataformas. Se integran fácilmente y es realmente sencillo de utilizar una vez se tiene el entorno correctamente preparado. Es una lástima que la versión 5 de ARToolkit haya dejado de funcionar y los creadores no hayan dado una solución fácil y definitiva al problema y solo quede esperar a que aparezca la versión 6 que parece muy prometedora.

Este proyecto me ha servido para adentrarme en el mundo de los videojuegos para dispositivos móviles con una plataforma excelente como es Unity y ha supuesto un reto duro pero gratificante.

12. Patrón de imagen (marcador)