Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Desarrollo de un videojuego comercial para plataformas móviles en Unity

Proyecto Final de Carrera

Ingeniería Informática

Autor: Javier Belenguer Faguás

Director: Jorge Belenguer Faguás y Juan José Lull Noguera

25/04/2013

Resumen

El proyecto consiste en el desarrollo y comercialización de una app (aplicación móvil) para iOS y Android mediante el entorno de desarrollo de videojuegos 3D Unity. Se trata de un juego del género First Person Shooter con ambientación militar moderna en el que debes de eliminar a todos los enemigos utilizando las distintas armas de fuego a tu disposición mientras recoges botiquines y munición para ayudarte a sobrevivir.

Palabras clave: Storm Gunner, Unity, plataformas móviles, videojuego, UML, RevMob, Tapgage.

Agradecimientos

A mi familia, por apoyarme todos estos años y por su comprensión y paciencia.

A mis amigos y compañeros, por todos los buenos momentos que hemos compartido.

A Jorge y Juan, por haberme brindado esta oportunidad y por su apoyo y ayuda.

A Gopango Networks S.L. por su ayuda en el desarrollo de este proyecto y por su apoyo.

Tabla de contenidos

1. Introducción

En este capítulo se introduce el proyecto final de carrera, Storm Gunner, un videojuego en 1ª persona de ambientación militar moderna en el que has de sobrevivir matando a todos los enemigos con las armas de fuego a tu disposición.

1.1. Motivación

En los últimos años se han popularizado los teléfonos inteligentes o "smartphones", teléfonos móviles construidos sobre una plataforma informática móvil, lo que les proporciona una mayor capacidad de computación y conectividad, siendo algunas de sus características comunes el acceso a internet, GPS, función multitarea, y la descarga, instalación y uso de aplicaciones de terceros. Esta última característica ha abierto un nuevo mercado a las empresas desarrolladoras de software, que han convertido o desarrollado software para estas nuevas plataformas. Uno de los tipos de aplicaciones más populares en estos dispositivos son los videojuegos, llegando algunos de los más exitosos a acumular millones de descargas.

El desarrollo de videojuegos es una tarea compleja y costosa, llegándose a invertir años y millones de dólares en su desarrollo, y esto ocurre también, aunque en menor medida, en el desarrollo de videojuegos para dispositivos móviles inteligentes. Estos costes y tiempos de desarrollo se ven aumentados por la existencia de varias plataformas distintas dentro de los "smartphones", siendo las principales iOS y Android. La aplicaciones para la plataforma iOS se escriben en el lenguaje de programación Objective-C, mientras que las aplicaciones para Android se programan mediante Java, aunque también permite el uso de otros lenguajes como C/C++. Además, existen cada vez más modelos de dispositivos, con distintos tamaños de pantalla, resoluciones, CPUs y GPUs y memoria RAM, lo que dificulta comprobar el correcto funcionamiento de la aplicación en todos los teléfonos móviles.

Para intentar solventar este problema han surgido varios SDKs (Software Development Kit, o kit de desarrollo de software) y motores de videojuegos multiplataforma, permitiendo reutilizar la mayor parte del código al desarrollar para las distintas plataformas que soportan, algunos ejemplos de este tipo de SDK son Corona SDK, Marmalade, o GameSalad.

Otro SDK cuya popularidad se ha ido incrementando rápidamente de Unity, habiendo sido usado para juego como Bad Piggies de Rovio o Wasteland 2 de inXile Entertainment. Algunas característica que diferencia este SDK de los demás es que cuando con un editor visual para crear lo niveles y gestionar las escenas, una tienda de librería y extensiones de terceros tanto gratuitas como de pago, la gran cantidad facilidades para el desarrollo como shaders, sombras, iluminación e sistemas de partículas fácilmente integrables en los proyectos y el gran número de plataformas que soporta, como Android, iOS, Windows, Mac OS X, Linux, Web, PS3, etc.

1.2. Objetivos

El propósito de este proyecto es la implantación de una app (aplicación móvil) que será comercializada, de principio a fin, contando con la ayuda del equipo de la empresa Gopango Networks, S.L.

La app en cuestión en un juego, el cuál será desarrollado mediante Unity, entorno de desarrollo de aplicaciones 3D, que permite la creación de versiones para iOS (dispositivos Apple), Android, Mac y PC entre otros. Se trata de un juego en primera persona en el que el usuario ha de defenderse de una serie de enemigos disparándoles con diversas armas, pudiendo únicamente girar y otear en diferentes direcciones, sin posibilidad de moverse de la posición inicial. Los objetos de la escena son tridimensionales y son facilitados por la empresa.

Los pasos del proyecto son los siguientes:

- Negociación con el cliente de las características que tendrá el juego. Generación de hoja de requerimientos.
- Creación de versión alfa, que será mostrada al cliente.
- Modificación del proyecto con las aportaciones del cliente.
- Generar fase de beta-testing, incorporando información de los testeadores y discusión con el equipo de la empresa sobre puntos críticos, mejoras importantes y mejoras deseables para el futuro.
- Creación de la versión final del proyecto.
- Generación del material necesario para publicar la app en la tienda de Apple: crear versiones del icono, capturas de pantalla (screenshots). Introducir información necesaria en la plataforma de juego social Game Center. Participación en la creación del texto de descripción de la app, creación de palabras clave.

En cuanto al propio desarrollo de la app, se utilizará como herramienta de gestión del proyecto de software GitHub, tanto para compartir el proyecto con otros desarrolladores del equipo como para plasmar los hitos del proyecto y las tareas necesarias para cada hito del mismo.

Las tareas de desarrollo, a grandes rasgos, son las siguientes:

- Creación de una escena en Unity
- Incorporación del elemento de terreno y skydome.
- Crear la funcionalidad a los soldados enemigos: flujo de estados unidos a acciones y a la animación del objeto.
- Generación del comportamiento del personaje principal. Funcionalidad de rotación de vista, apuntado y disparos. Creación de los controles de manejo.
- Pantalla de configuración (volumen, selección de controles).
- Introducción del objeto caja (vida, munición) y funcionalidad.
- Creación de luces y sombras de la escena.
- Generación de splash screen o pantalla introductoria.

- Pantalla de créditos.
- Creación de menú de niveles.
- Creación de parámetros de dificultad para los distintos niveles de dificultad.
- Asignación de sonidos (efectos especiales) y música al juego.
- Introducción del HUD (Head-Up Display, información en pantalla durante la partida).

2. Fundamentos básicos de Unity

Unity, como se ha mencionado antes, es un motor para videojuegos e IDE que facilita el trabajo de desarrollar videojuegos. El motor de Unity cuenta con sistema de física para calcular colisiones, simular la gravedad y otras fuerzas sobre los objetos del videojuego, etc., tiene un sistema de partículas para renderizar humo, explosiones y demás efectos visuales, tiene un sistema de audio, pathfinding, red, soporta shaders y otras muchas características. Unity es compatible con 3 lenguajes de programación, C#, Boo y una variante de javascript llamada UnityScript de características y sintaxis similar al lenguaje JScript.NET de Microsoft.

Por otro lado, Unity cuenta con un editor, el cual sirve como editor de niveles, permitiendo situar los elementos en la escena tal y como aparecerán en el juego al iniciarse el juego, pudiendo tener multiples escenas para representar distintos niveles o menus del juego, pudiendo cargar otra escena en cualquier momento. Unity representa los elementos del juego mediante “GameObjects”, que son elementos que cuentan solo con una posición, rotación y escalado, a los que se les puede asociar comportamientos en forma de clases que heredan de la clase MonoBehaviour, los cuales actúan como la lógica del “GameObject”, además se les pueden asociar otros “GameObjects” y también componentes, como pueden ser caja de colisión para la física, modelos en 3D y otras opciones, cuya posición, rotación y escalado serán relativos al “GameObject”. Desde el editor se puede modificar el valor inicial de todas las variables públicas de los scripts asociados a cada GameObject, pudiendo ver los resultados en el editor conforme se modifican los valores.

Aunque no es necesario heredar de la clase MonoBehavior para realizar algunas tareas en Unity, el desarrollo está orientado para su uso, ya que la ejecución de la aplicación se basa en el flujo de llamadas de MonoBehavior, el cual facilita la organización del código y su ejecución en el momento apropiado al contar con diveras funciones que se ejecutan cuando el objeto, se crea, se destruye, durante la actualización de la lógica o de la física y otras situaciones. Por otro lado, la utilización de MonoBehavior permite la creación de corutinas, es decir, subrutinas que pueden de pausadas y posteriormente resumidas cuando sea apropiado, lo que permite estructurar el código de forma similar a si se utilizaran multiples hilos sin tener que preocupares de establecer semáforos y otras medidas para evitar el bloqueo de los hilos.

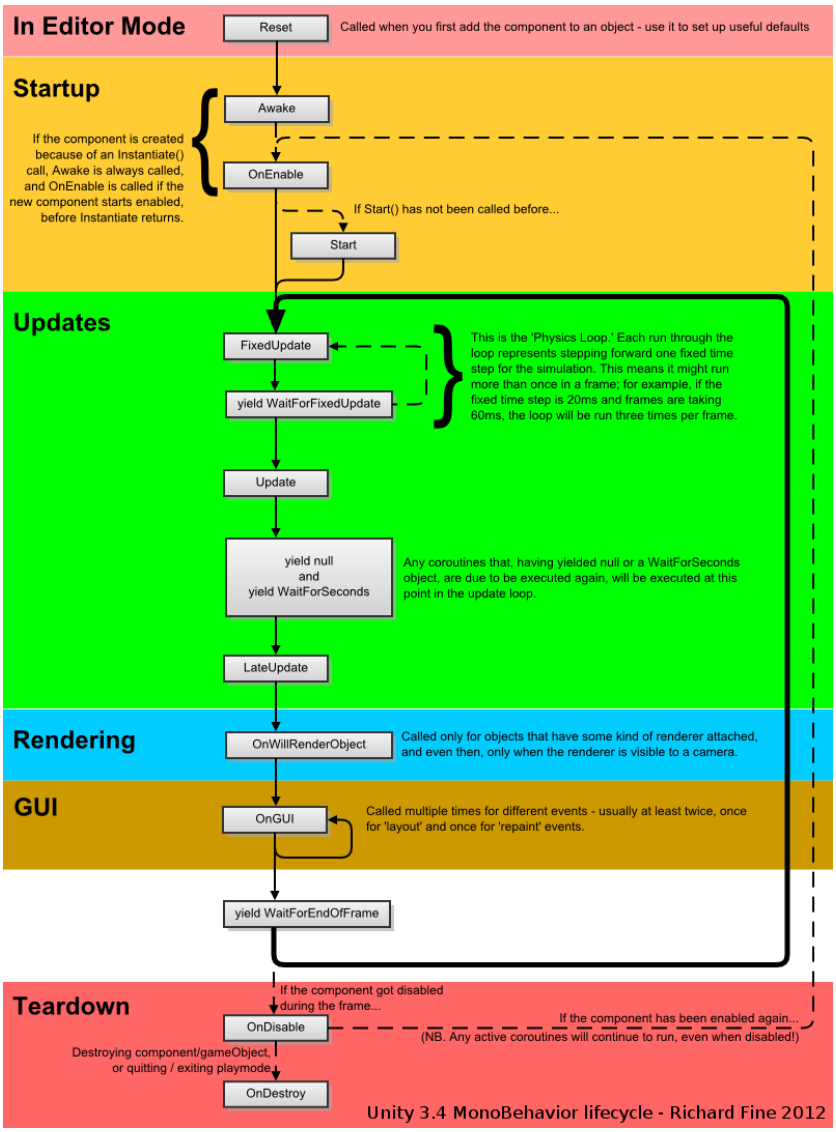

Figura 1. Ciclo de vida de MonoBehavior

Además, tal y como se ha mencionado antes, Unity funciona para múltiples plataformas, siendo iOS, Android, Mac OS X, Windows, ps3 y Xbox 360 algunas de ellas. El editor puede emular el aspecto del videojuego en los distintos dispositivos y compilar y ejecutar el videojuego en el dispositivo, pudiendo configurar las opciones de compilación y personalizando el videojuego para cada dispositivo, además de reutilizar la mayor parte del código del juego.

Unity utiliza el IDE (Integrated Development Environment) llamado MonoDevelop para la programación del código del juego, este IDE multiplataforma permite desarrollar software usando lenguajes .NET y que se pueda compilar para Windows, Mac y Linux, aunque Unity utiliza una versión modificada propia que compila el código directamente desde el editor de Unity. Dado que es el IDE para el que está pensado Unity se ha utilizado este IDE para evitar problemas de compatibilidad y demás conflictos al compilar.

Por último, Unity puede ser ampliado mediante plug-ins creados por los usuarios para incrementar las funcionalidades tanto del motor como del editor,

siendo algunos gratis y otros de pago, los cuales se pueden obtener desde la tienda oficial de Unity o como archivo con extensión ".unitypackage" a través de terceros.

3. Análisis y diseño

3.1. Características del videojuego

Storm Gunner es, tal y como se ha mencionado antes, un First Person Shooter (FPS), de ambientación militar de época contemporánea en el que te defiendes de los enemigos que te atacan, no puedes moverte de tu posición pero puedes girar, apuntar a los enemigos y dispararles.

El juego se ha dividido en 2 modos de juego, el modo clásico y el modo infinito. En el modo clásico, existen múltiples niveles, que se han de completar en orden, siendo el modo progresivamente más difícil ya que la habilidad del jugador aumenta conforme juega y no supondría un desafío y siendo los primeros niveles fáciles para que el jugador pueda practicar. Para mejorar la experiencia de juego y evitar que se vuelva monótono, además de facilitar el aumento progresivo de la dificultad, existen varios tipos de enemigos que van apareciendo conforme se avanza de nivel. Para compensar esto y añadir más variedad de juego el jugador cuenta con nuevas armas en los niveles posteriores. Para mejorar la variedad de los niveles hay 2 tipos distintos, los niveles "Deathmatch", en los cuales hay un número determinado de enemigos de cada tipo y el objetivo es acabar con todos, y los niveles "Survival" en los que hay un número infinito de enemigos de ciertos tipos dependiendo del nivel y el objetivo es sobrevivir durante un tiempo determinado. Este es el modo principal de juego ya que facilita el aprendizaje por parte del jugador de las mecánicas de juego y de las distintas armas y enemigos y porque proporciona una experiencia de juego mejor organizada al ser niveles planeados por el equipo de desarrollo.

Por otro lado, el modo infinito cuenta con un solo nivel, en el que hay una cantidad ilimitada de enemigos y cuyo objetivo es aguantar el mayor tiempo posible vivo, disponiendo de todas las armas del juego desde el principio, y apareciendo nuevos tipos de enemigos progresivamente. Este modo de juego sirve para extender la rejugabilidad, ya que siempre puedes hacerlo un poco mejor y aguantar más tiempo, y por lo tanto hacer que jueguen más tiempo al juego y puedan seguir divirtiéndose con él aunque hayan completado el modo principal.

El juego cuenta con 3 tipos de armas el rifle de asalto, la ametralladora y el bazuca, las cuales tienen munición limitada. El rifle de asalto es el arma inicial, tiene el daño más bajo, velocidad de recarga rápida y no permite mantener el botón de disparo para disparar continuamente, por lo que es un arma de propósito general. La ametralladora en cambio sí que permite disparo continuo, provoca más daño al enemigo y dispara muy rápido y no requiere recargar, pero la munición se acaba rápidamente si se dispara de continuo y tarda en empezar

a disparar, por lo que resulta más útil contra grupos de enemigos. Por último, el bazuca es el arma que más daño provoca, pero tiene una velocidad de recargado muy lenta, hay que recargar tras cada disparo, tiene poca munición y la velocidad de los misiles es mucho menor que las balas, ya que es un arma pensada para ser usada contra los enemigos más poderosos.

Dado que la munición es un recurso limitado y es necesaria para poder matar a los enemigos, el quedarse sin munición es, junto con quedarse sin vida, condición de fracaso de la partida. Para poder recuperar vida y munición cada cierto tiempo aparecen cajas de munición y primeros auxilios. Dado que no te puedes mover de tu posición para cogerlas, se tomó la decisión de que se consiguiera la vida o munición al destruir la caja disparándole, lo que además evita introducir una nueva mecánica de juego solo para poder conseguir las cajas, lo cual resultaría molesto a los jugadores al tener que cambiar de una a otra según si se está matando a enemigos o recogiendo cajas. Si las cajas no se utilizan estas se autodestruyen tras un tiempo para evitar que el jugador las acumule hasta el momento en que la necesite y para aumentar la emoción de encontrar una caja, sobre todo cuando el jugador necesita para sobrevivir.

En cuanto a los enemigos, existen 3 tipos distintos, los soldados, los hummers y los tanques. El soldado es primer tipo de enemigo contra el que el jugador se enfrenta y el más común, además de ser el que menos vida tiene y menos daño provoca al jugador. Los soldados, al igual que las cajas de vida y munición, aterrizan en paracaídas, y se mueven a tu alrededor disparándote o lanzándote granadas en intervalos variables. Las granadas provocan más daño que los disparos, pero las granadas pueden ser destruidas en el aire por el jugador a disparos previniendo el daño, con la ventaja de ser espectacular y emocionante para el jugador, los disparos en cambio solo se pueden evitar matando al enemigo antes de que acabe de apuntar.

Los hummers son el siguiente tipo de unidad que aparece en el juego, tienen más vida, provocan más daño al jugador y siempre están en movimiento, incluso mientras disparan con sus ametralladoras, aunque hay muchos menos en cada nivel. Los hummers, al igual que los tanques, se acercan desde el horizonte en vez que aterrizar en paracaídas. Para matar a este enemigo se requiere más puntería y habilidad, especialmente con el bazuca.

Finalmente, los tanques tienen más vida y provocan más daño que los hummers, aunque son más lentos moviéndose y apuntando y se paran para apuntar y disparar, por lo que son más fáciles de acertar con el bazuca. Estos son los enemigos son muy resistentes, lo que a menudo provoca que se acumulen los soldados y desaparezcan las cajas mientras el jugador se ocupa de destruirlo, por lo que requieren una mayor planificación por parte del jugador.

Además de esto, el juego cuenta con un sistema de puntuaciones y estadísticas para los jugadores con más experiencia, teniendo en cuenta su puntería, vida restante al acabar la partida y enemigos matados, consiguiendo puntos extra si murieron en el aire para premiar la rapidez. Este sistema permite a los

jugadores intentar mejorar su resultado cada vez que juegan, aumentando la rejugabilidad del juego y complementándose con el modo infinito, donde siempre se puede conseguir una mejor puntuación.

Se quiso potenciar el aspecto online y en grupo del juego, muy popular en los dispositivos móviles y que potencia el que los usuarios den a conocer el juego a sus amigos, pero dado que el juego no está pensado para ser multijugador, se decidió añadir logros y rankings de puntuaciones online, donde se puede competir intentando conseguir los mejores resultados y desbloquear todos los logros. Los logros son proezas o hitos que se pueden alcanzar en el juego, como matar a cierto número de enemigos o de cierta forma, mientras que los rankings muestran a los jugadores con mejores puntuaciones en los distintos niveles y modos.

Para facilitar el proceso de aprendizaje el juego cuenta con un tutorial, en el cual se enseña cómo funciona la cámara, que hacen las cajas de munición y vida, y se muestran varios tipos de armas y como cambiar de una a otra, de modo que el jugador no se sienta perdido al empezar a jugar y abandone el juego.

El juego cuenta con 2 modos de control distintos entre los que el usuario podrá elegir según sus preferencias. El primero consiste en mantener pulsado cerca del borde de la pantalla en el lado hacia el que se quiere girar la cámara, estando la mira de apuntado justo encima del dedo de donde de pulsa el usuario, que será hacia donde disparará el arma. El segundo modo es mediante un pad virtual similar al que se encuentra en los mandos de las consoles, pero dibujado en pantalla, el cual aparecerá donde pulse el usuario, y que servirá como eje de coordenadas para calcular hacia donde mover la cámara, de modo que si se arrastra el dado hacia la derecha del pad la cámara se moverá hacia la derecha, etc.

En cuanto a disparar, hay un botón en pantalla que al ser pulsado provoca que se dispare el arma, pudiendo mantenerse para seguir disparando. El juego no requiere recargar el arma, salvo en el caso del lanzamisiles el cual se recarga automáticamente, ya que volvería más complejo el juego sin aportar diversión al usuario, además de requerir al usuario acordarse de recargar lo cual podría resultar molesto.

Para cambiar de arma el juego cuenta con un botón, que al ser pulsado, muestra el resto de arma disponible en pantalla a los lados del botón, pudiendo desplazar el dedo en dirección contraria al arma deseada, lo que arrastrará las armas en esa dirección, y si deja de pulsar cuando el arma se encuentra encima del botón se cambiará a esa arma. De este modo funciona de forma similar a las listas en los dispositivos con pantalla táctil, facilitando el aprendizaje al usuario.

3.2. Modelado UML

Se ha utilizado el lenguaje de modelado UML (Unified Modeling Language) para facilitar la organización y estructuración del desarrollo del juego mediante un diagrama de clases, el cual muestra las clases que forman el código de la

aplicación y como se relacionan entre sí, además también ha ayudado a definir la organización de los menús del juego y el flujo del mismo mediante un diagrama de estados.

Debido a que Unity se hace cargo de la mayor parte de las tareas relacionadas con el renderizado de la escena, la física, el sonido, etc., no es necesario programar clases que se ocupen de gestionar esto, por lo que el diagrama de clases está más centrado en el gameplay y la interfaz. Además, se ha aprovechado el ciclo de vida de MonoBehaviour para la programación de las clases, se han aprovechado las funciones de inicialización, actualización, etc. de MonoBehaviour.

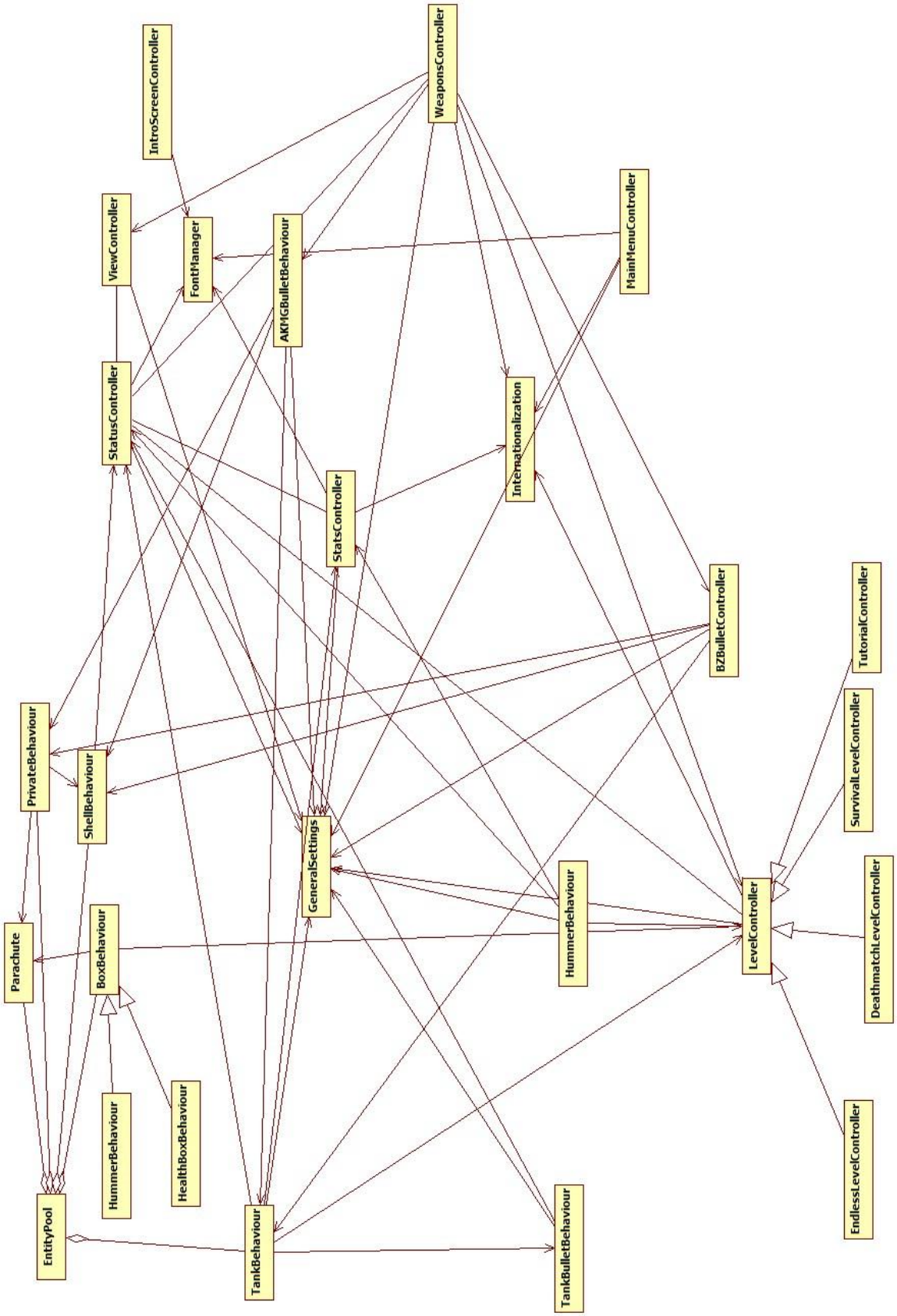

Figura 2. Diagrama de clases UML

Las clases de la figura 2 se pueden dividir en su mayor parte en 2 categorías principales según su propósito, las clases cuyo nombre acaba en “Behaviour” contienen la lógica y el comportamiento de las distintas entidades que forman el juego, es decir, los enemigos, las balas, los power-ups y el paracaídas. En el caso de los enemigos, estas clases gestionan la inteligencia artificial, el movimiento, las animaciones y el disparo al jugado, mientras que si son balas, power-ups, etc. simplemente se ocupan del movimiento, las colisione y lo que sucede en tal caso.

La segunda categoría de clases son aquellas cuyo nombre acaba en “Controller”, las cuales gestionan la escenas, los niveles y al jugador. LevelController y las clases que heredan de ella gestionan los niveles, ocupándose de crear a los enemigos y colocarlos en el mapa, además de acabar la partida cuando se cumpla el objetivo de la misma. ViewController gestiona la cámara del jugador, mientras que StatusController gestiona la interfaz y los controles, StatsController registra los logros y puntuaciones del jugador y WeaponsController el disparo y recarga del arma y sus animaciones, además de la munición y el cambio de arma. IntroScreenController gestiona la escena inicial en la que se muestran los logos de las compañías, haciendo la transición de un logo al siguiente, mientras que MainMenuController gestiona la escena principal en la que se encuentran el menú de opciones, el de selección de nivel, etc. interpretando las pulsaciones del jugadores y animando el fondo y las transiciones entre menus.

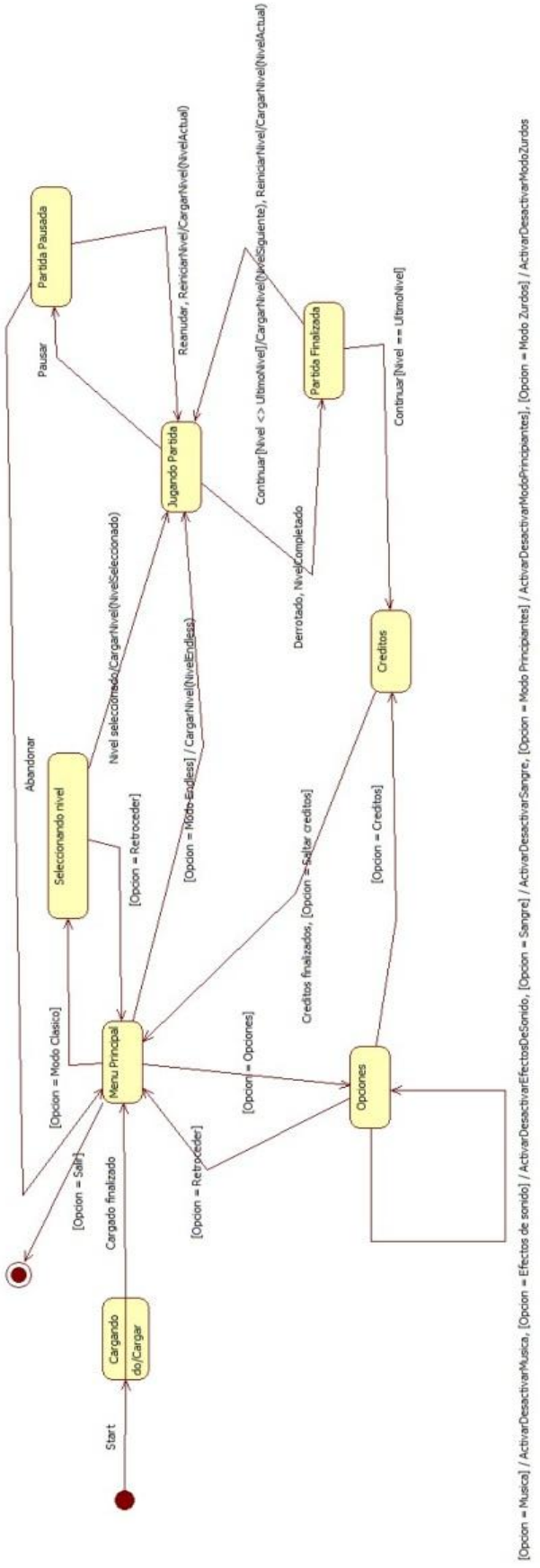

Figura 3. Diagrama de estados UML

Tal y como se puede ver en la figura 3, cuando inicias el juego comienza el cargado, mostrándose la escena de introducción con los logos de las compañías. Cuando se acaba de cargar se muestra el menú principal, desde el cual se puede elegir entre el modo clásico o el modo infinito, además de jugar el tutorial, e ir a opciones, desde donde se puede ir a los créditos. Si se selecciona el modo clásico se muestra el menú de selección de nivel, y al seleccionar un nivel comienza el juego, mientras que si se selecciona el modo infinito el juego comienza inmediatamente. Durante la partida se puede pausar el juego y desde el menú de pausa se puede volver al menú principal además de reanudar o reiniciar la partida.

3.3. Diseño funcional

Se realizó un diseño funcional de la interfaz para facilitar al diseñador gráfico el diseño y creación de los gráficos de la interfaz.

Figura 4. Menú Principal

En el menú principal podemos encontrar los botones que van a los diferentes modos de juego y menús con los que cuenta el juego, estando separados del botón de "Otras apps" ya que este lleva a una pantalla con publicidad que se explicará más adelante, para distinguir de este modo claramente los botones relacionados con la funcionalidad del juego del botón de publicidad. Los botones se encuentran situado a los lados ya que se desea que se aprecie mejor la animación de fondo que se va a implementar y de este modo llamar la atención al usuario y tenga una buena primera impresión del juego.

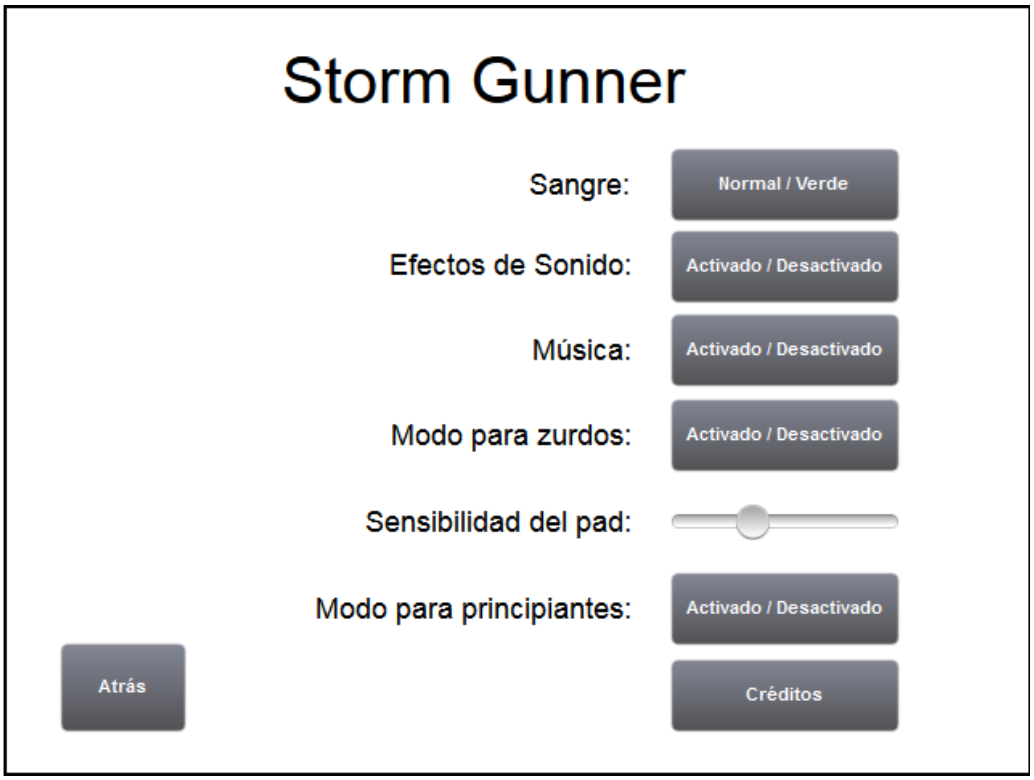

Figura 5. Menú de opciones

El menú de opciones cuenta con los controles para ajustar el sonido, los modos de control del juego y para desactivar la sangre, de modo que puedas configurar todo el juego desde un solo menú sin necesidad de tener que el usuario tenga buscar donde se encuentra la opción que desea. Además se ha situado el botón que lleva al menú de créditos ya que no es lo suficientemente importante para aparecer en el menú principal.

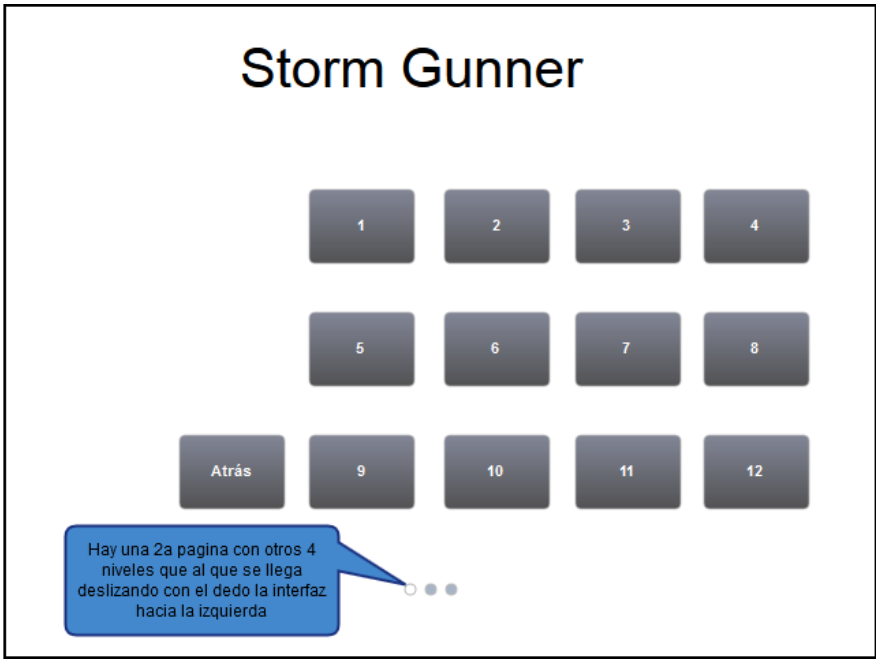

Figura 6. Menú de selección de nivel del modo clásico

En el menú de selección de nivel del modo clásico se encuentras los botone para ir a los distintos niveles del juego, pudiendo deslizar la interfaz hacia la izquierda para ver la segunda página de niveles.

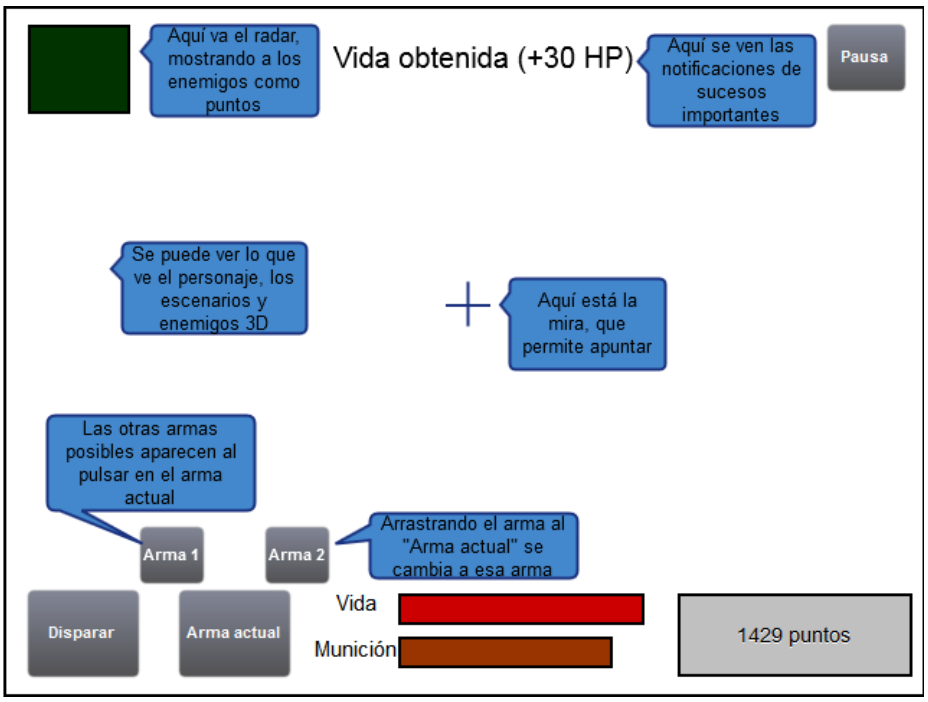

Figura 7. Partida del juego

Durante la partida se puede ver el entorno 3D sobre el que se desarrolla el juego, con la interfaz de juego representada encima, mostrando datos relevantes para el usuario como la vida, la munición, la posición de los enemigos, etc., tal y como se puede ver en la figura 7.

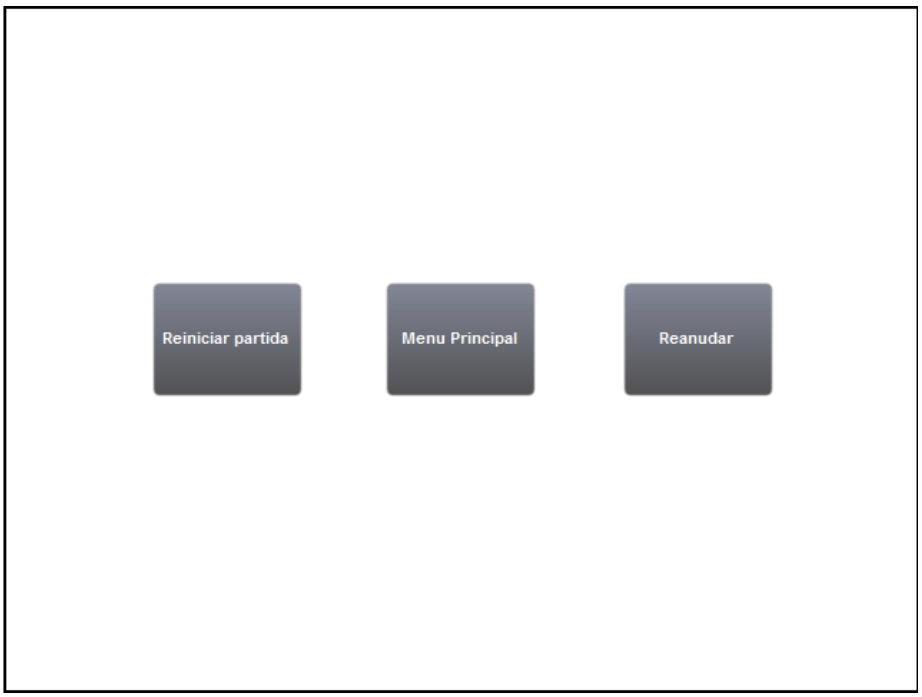

Figura 8. Menú de pausa

Al pulsar en el botón de pausa durante la partida, el cual se puede ver en la esquina superior derecha en la figura 7, el juego se pausa y se muestra el menú de pausa sobre la partida, pudiendo reiniciar o reanudar la partida y volver al menú principal.

Como se puede ver, el juego se ha dividido en 4 menús además de la interfaz de juego, el propósito es reducir el tiempo de desarrollo dedicado a la implementación de la interfaz y que esta sea sencilla e intuitiva, utilizando elementos de interfaz. y gestos táctiles comunes en los dispositivos móviles con pantalla táctil.

4. Implementación

A partir del análisis y diseño antes explicado se procedió a implementar el juego. Para que reducir el tiempo de desarrollo dedicado a la interfaz y mejorar la reusabilidad los elementos de la interfaz tienen tamaños y posiciones proporcionales al tamaño de la pantalla, por lo que la interfaz tiene la misma distribución en distintas resoluciones y relaciones de aspecto, como puede ser 4:3 o 16:9.

Al comienzo del juego se muestran en sucesión los logos de las compañías que han participado en el desarrollo del juego, pudiendo acelerar el proceso pulsando en la pantalla. Mientras carga se sigue mostrando el último logo cuando con un texto indicando que el juego está cargando para que el usuario sepa que el juego responde.

Para la realización del menú principal se creó un escena 3D con el cielo, el soldado y el parapeto que se ven en el fondo colocados estratégicamente para obtener un resultado vistoso mediante una cámara 3D sobre la se representa la interfaz. En cuanto a los botones, se han utilizado textos renderizados como texturas, los cuales cambian de color cuando se pone detecta que se está pulsando encima de ellos para proporcionar “feedback” al usuario. Para optimizar la renderización de los textos se han utilizado fuentes bitmap, que consiste en crear imágenes que contienen todos los caracteres de la fuente, realizándose una imagen para cada tamaño distinto de fuente que se va a utilizar.

Para dar un aspecto más impactante al juego se ha añadido un efecto de “lens flare” cuando se mira al sol, el cual se puede ver también durante la partida, que consiste en una serie de brillos que se suele producir en las cámaras y otros sistemas de lentes.

Las transiciones entre menús se realizan volviendo gradualmente transparentes los elementos del menú anterior y opacos los elementos del nuevo menú mientras que se mantiene el mismo fondo, mejorando de esta forma el “feedback” al usuario al indicar claramente el cambio de menú y mejorando el

aspecto visual del juego, además de tratarse de una característica muy común en las aplicaciones para “smartphones” y a lo que los usuario están acostumbrados.

En el menú de selección de niveles del modo clásico se pueden ver los botones correspondientes a los diferentes niveles, mostrándose de un color más oscuro los botones de los niveles aún no desbloqueados para que resalten menos y se diferencien de los botones de niveles desbloqueados. Tal y como se menciona en el diseño funcional se ha implementado el sistema común en dispositivos móviles de desplazar con el dedo la interfaz para ver el resto de la interfaz, mostrándose acabado como puntos cuantas “páginas” hay y como un punto blanco la página actual, intentando de este modo que la interfaz sea intuitiva para el usuario.

En cuanto a la partida, se ha utilizado el mismo sistema de cámara 3D con la interfaz de juego superpuesta que en los menús antes mencionados, salvo que en este caso la cámara muestra el mapa de juego. Los modelos y terrenos del juego son modelos 3D, mientras que el cielo es un “skybox”, es decir, un cubo que envuelve el mapa de juego y que en las caras internas se le han aplicado texturas que representan el cielo, siendo una forma eficiente de implementar cielo. Se pueden ver los brazos y el arma del jugador para un mayor realismo y para que el usuario sepa que está viendo a través de los ojos del personaje, además de proporcionar “feedback” al usuario cuando recarga o dispara, ayudando la inmersión en el juego.

Dado que el renderizado de sombras realistas es demasiado costoso para la mayoría de dispositivos móviles, se ha realizado una versión simplificada de las sombras proyectando sobre el suelo una elipse negra con suavizado de borde usando como origen de la proyección el suelo para que esté correctamente orientada con respecto a la luz. También se ha añadido una luz direccional en donde se encuentras situado el sol en el “skybox” para emular la luz proveniente del sol.

Las explosiones y las salpicaduras de sangre se han implementado como sistemas de partículas, de este modo nunca son exactamente iguales, dándole un aspecto más realista al juego y menos repetitivo.

Para mejorar el rendimiento del juego, lo cual es importante para que funcione correctamente en el mayor número de dispositivos posibles, en vez de crear y destruir los objetos que representan a los enemigos, paracaídas, granadas cada vez se ha implementado “object pooling”, es decir, se reutilizan los objetos, siendo desactivados cuando no se están utilizando, y siendo activados y reiniciados a su estado original cuando se van a utilizar de nuevo, por lo que se reduce el tiempo de procesamiento dedicado a la gestión de memoria para la creación de los objetos, lo cual podría producir picos de latencia cuando aparece un nuevo enemigo, granada, etc.

El juego cuenta con efectos de sonido para los disparos, explosiones, plegado de los paracaídas y para los movimientos de los vehículos y soldados. Además

también cuenta con música para los menús y otra diferente durante la partida más apropiada para la acción. Para reducir el coste del videojuego se optó por utilizar sonidos con licencia para uso comercial gratuita con atribución, los cuales fueron obtenidos de diversas webs dedicadas a ello.

También se ha añadido una escena de créditos al juego, la cual se puede ver al completar el ultimo nivel del modo clásico o accediendo desde el menú de opciones. Dado que no caben todos los créditos en pantalla a la vez, los créditos se desplazan hacia arriba lentamente desde la parte inferior de la pantalla, ocultándose los créditos ya mostrados y apareciendo los créditos que faltan progresivamente, de forma similar a como se muestra en muchas películas. En los créditos, además de aparecer los participantes en el desarrollo, también se muestran los autores de los efectos de sonidos y la música utilizadas en el juego, ya que es un requisito para su uso atribuir su autoría, tal y como se ha mencionado antes.

Para obtener información sobre los usuarios y su experiencia de juego y de este modo poder mejorar el gameplay y el marketing del juego se ha integrado GameAnalytics en el juego. GameAnalytics es un sistema de estadísticas y análisis que permite saber cuántos usuarios tiene el juego, de que país son, cuantos siguen teniendo la aplicación instalada, cuantos están jugando en ese momento, etc. Por otro lado, también permite establecer eventos dentro del juego que puedan ser relevantes para el desarrollador para saber cuántas veces se da dicho evento. En el caso del Storm Gunner se establecieron 2 eventos, uno que se da cuando el usuario acaba el tutorial la primera vez que lo juega y otro que se cumple cuando el jugador salta el tutorial la primera vez. Estos eventos permiten saber lo ameno que es el tutorial para el usuario y cuantos usuarios empiezan a jugar sin saber los mecanismos de juego. Además de todo esto, GameAnalytics también permite ver los errores de ejecución que han sufrido los usuarios y cuantas veces han ocurrido para poder arreglarlos y saber cuáles son prioritarios.

Para facilitar la comprobación de errores y el balanceo del gameplay se ha utilizado TestFlight, un servicio que permite subir un aplicación de iOS a sus servidores y que sea descargado directamente desde el móvil por las personas a las que les hayas dado permiso, de modo que puedan jugar al juego sin necesidad de usar un ordenador para compilar y transferir la aplicación al móvil.

5. Monetización

Existen 3 opciones principalmente para la monetización de una aplicación para móviles inteligentes, que son:

- Publicar la aplicación como una “app” de pago.
- Mediante compras “in-app”, es decir, permitir comprar contenido y características adicionales desde la propia aplicación.
- Poner publicidad en el juego

Algunas de estas opciones pueden combinarse, siendo habitual en algunos casos.

Dado que el principal motivo para jugar durante un tiempo prolongado es el modo infinito y los rankings online, no es apropiada la implementación de compras “in-app” de contenido adicional ya que daría ventaja a los jugadores que pagaran, lo que sería considerado injusto por los jugadores y provocaría que el juego recibiera valoraciones negativas.

Tradicionalmente en los juegos para móviles inteligentes los juegos de pago no cuentan con publicidad, ya que se considera que el usuario ha pagado por la eliminación de publicidad, lo que hace que estas 2 opciones sean excluyentes, salvo que se planteen como 2 opciones de uso de la aplicación para el usuario, una gratis con publicidad y otra de pago sin publicidad. En este caso en vez de publicar 2 versiones de la aplicación, una de pago y otra gratis, se puede publicar como una aplicación gratis con publicidad e implementar una compra “in-app” que permita eliminarla, lo cual tiene la ventaja de no requerir 2 revisiones de la aplicación por parte de Apple al publicar la aplicación en la App Store, pudiendo una de ellas se rechazada y la otra aceptada.
Finalmente se decidió publicar la aplicación gratis con publicidad, ya que en juegos anteriores publicados por la compañía se había comprobado que publicando como “app” de pago se obtenían hasta 10 veces menos beneficio, además de producirse hasta 1000 veces menos descargas, lo que provocaba que la publicidad por “boca a boca”, es decir, la que hacen los usuarios al hablar sobre el juego a sus conocidos, fuera significativamente menor, reduciendo también los beneficios a largo plazo.

En cuanto a los servicios de publicidad usados, se eligió utilizar RevMob y Tapgage ya que ambos servicios de publicidad habían sido usado anteriormente por la empresa en otros juegos y aplicaciones con buenos resultados, ambos cuentan con anuncios de juegos, que son los que es más probable que interesen a los usuarios y que pulsen, produciendo mayores beneficios, y que cuentan 2 diversas opciones que anuncio que se adaptan a los distintos casos y necesidades. Se han utilizado 2 servicios de publicidad para evitar que no aparezcan anuncios en el juego si uno de los 2 servicios no cuenta con anuncios para el país del usuario en ese momento, lo que provocaría que se produjeran beneficios. Además de todo esto, ambos servicios cuentan con plug-ins para integrarlos en juegos creados en Unity, por lo que el tiempo y coste de

desarrollo dedicado a la implementación de la publicidad en el juego es muy bajo.

Tapgage es un servicio de publicidad de juegos y otras aplicaciones, en la que se gana dinero cuando el usuario pulsa en el anuncio. La cantidad de dinero obtenida por cada pulsación es baja, por lo que está pensado para ser usado en aplicaciones con muchos usuarios. Tapgage cuenta con 2 tipos de publicidad, los anuncios a pantalla completa y los "offer walls", los cuales son listas de aplicaciones, en las cuales el usuario puede elegir la aplicación que desee, apareciendo una aplicación como destacada.

Figura 9. Ejemplo de anuncio a pantalla completa de Tapgage

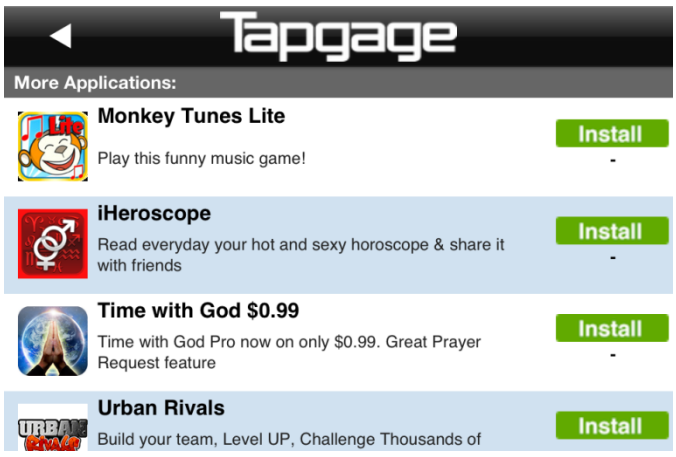

Figura 10. Ejemplo de "offer wall" de Tapgage

Por otro lado, RevMob es un servicio de publicidad sólo de juegos, en el que se gana dinero cuando el usuario se instala el juego publicitado. El dinero obtenido

por instalación es muy alto comparado con otros servicios de publicidad, a cambio el número de gente que se instala los juegos es menor que el de gente que pulsa en un anuncio, por ello este servicio requiere de menos usuarios para obtener beneficio, aunque los beneficios diarios pueden variar más. RevMob cuenta con 4 tipos de publicidad, los anuncios a pantalla completa, los pop-ups, los banners y los "ad links", que son los anuncios personalizados por el desarrollador, pudiendo representarlo como quieras.

Figura 11. Ejemplo de anuncio a pantalla completa de RevMob

Para evitar que la publicidad moleste demasiado a los usuarios, se ha establecido que deben transcurrir al menos 30 segundos entre anuncios, para evitar que los anuncios vuelvan a aparecer si un usuario vuelve inmediatamente a un menú con anuncio al que acababa de acceder. Por otro lado, los anuncios aparecen en menus concretos del juego, que en el caso de RevMob son el menú principal y el de resultados de la partida, y en el caso de tapgage el menú de pausa del juego, siendo esto así para evitar que aparezcan anuncios durante la partida, interrumpiendo la partida al jugador. Además de esto, el anuncio del menú principal no aparece la primera vez que inicias el juego, para que no sea lo primero que ves al iniciar el juego por primera vez y de una mala impresión.

6. Estado del arte

En la actualidad los juegos para "smartphones" cuentan con una gran popularidad, siendo un género muy común entre ellos el "shooter" de supervivencia, en el que el jugador ha de sobrevivir contra continuas oleadas de enemigos hasta acabar con todos, o en otros casos, hasta morir. Storm Gunner cuenta con muchas similitudes a este tipo de juegos, como se puede ver en los siguientes ejemplos:

Dead on Arrival es un shooter de supervivencia ambientado en un apocalipsis zombie, en el cual el jugador ha de luchar contra las hordas de zombies, consiguiendo dinero que se puede gastar para desbloquear nuevas armas y otras

herramientas. A diferencia del Storm Gunner, este juego cuenta con una perspectiva cenital, lo que proporciona una visión más amplia del entorno sin necesidad de un radar sacrificando la inmersión que aporta una perspectiva en primera persona, y también hace más sencillo matar a los enemigos al ser más rápido apuntar a los enemigos.

Figura 12. Captura de pantalla del juego Dead on Arrival

Otro ejemplo de shooter de supervivencia es Zombieville USA 2, se trata de otro juego de zombies, aunque este con un estilo gráfico similar a los dibujos animados, dándole un aspecto más humorístico y menos realista. Por otro lado, el juego pese a su estilo gráfico cuenta con violencia explícita, mostrándose los cerebros y vísceras de los enemigos muertos, lo que lo hace inapropiado para los más pequeños, mientras que el Storm Gunner, aunque cuenta con gráficos realistas, lo único que muestra es pequeñas cantidades de sangre, que puedes desactivarse para que no aparezcan. Por otro lado el juego cuenta con una vista lateral 2D de cerca, por lo que el usuario solo ve los enemigos que se encuentran muy cerca de él, lo que reduce la complejidad estratégica del juego dado que el usuario se preocupa solo del momento actual, sin planear para más adelante.

Figura 13. Captura de pantalla del juego Zombieville USA 2

Por último, el juego más parecido a Storm Gunner es The Last Defender, un shooter de supervivencia de ambientación militar y perspectiva en primera persona similar al Storm Gunner, aunque cuenta con algunas diferencias, como el hecho de que en Storm Gunner puede quedarte sin munición, lo que provoca que pierdas la partida, además de contar con power-ups que aparecen durante un tiempo limitado, por lo que el juego requiere una mayor táctica, al tener que considerar cuando ir a por la munición y cuando matar a los enemigos, además de requerir buen puntería para no malgastar munición.

Figura 14. Captura de pantalla del juego The Last defender

Como se puede ver, el género de los shooters de supervivencia es variado, contando con multitud de títulos con distintos planteamientos y mecánicas, y Storm Gunner intenta aportar otro nuevo planteamiento de juego a este género, centrándose

menos en matar grandes cantidades de enemigos como es habitual en el género y planteando un estilo de juego donde tengas que planear tus acciones y pensar rápido sin perder el frenetismo propio del género.

7. Conclusiones

Durante el desarrollo de Storm Gunner, Unity ha facilitado de forma significativa el desarrollo multiplataforma del juego, permitiendo compartir prácticamente la totalidad del código desarrollado utilizando solamente la API de Unity y, aunque en el caso de los plug-ins y extensiones depende del plug-in concreto y como haya sido desarrollado, la mayoría de ellos son multiplataforma.

Además, Unity ahorró mucho tiempo de desarrollo debido a la gran cantidad de características típicas de videojuegos que ya lleva implementadas, como los proyectores usados en el juego para las sombras, raycast y colisiones para detectar cuando una bala acierta a un enemigo o cuando un enemigo en paracaidas toca el suelo, API común tanto para el input recibido de un raton y como el recibido de una pantalla táctil, etc.

Por otro lado, Unity también permitió la fácil integración de distintos servicio de publicidad y análisis de datos a través de plug-ins.

En cuanto a los aspectos negativos de Unity, los ejecutables ocupan más que los generados por otros entornos de desarrollo debido al sistema utilizado para que el código funcione de forma prácticamente idéntica en todas las plataformas, y que, aunque no es importante en plataformas como Windows o Linux, en plataformas móviles es una diferencia significativa.

Otro aspecto negativo es el hecho de que el código interno de Unity se ejecuta obligatoriamente en un solo hilo, siendo por lo tanto poco eficiente en procesador con múltiples núcleos. Esto se ve reducido debido a que el código implementado por el usuario si que puede utilizar múltiples hilos.

8. Trabajo futuro

En cuanto a mejoras de juego, en futuras versiones se podría ampliar la variedad de armas, aumentando el número de opciones para el jugador y provocando que el juego sea menos repetitivo, lo que alargaría el tiempo de juego medio de los jugadores.

Otra mejora a implementar sería añadir nuevos tipos de enemigos, lo que aumentaría la complejidad y variedad de los niveles, además de permitir añadir más niveles al juego, lo que alargaría la duración del modo clásico y daría más variedad al modo infinito. Este cambio y el anterior además conllevarían ganancias de dinero dado que la fuente de ingresos del juego es la publicidad, y

dado que los jugadores juegan más tiempo, ven más anuncios y es más probable que lo pulsen.

Por otro lado, se podría mejorar la monetización del juego implementado "in-apps", es decir, añadiendo la posibilidad de comprar diferentes cosas para el juego desde dentro del mismo usando dinero real, como podría ser eliminar la publicidad del juego, obtener armas muy poderosas de forma temporal o con una cantidad limitada de munición, o conseguir ataques especiales, ralentizar el tiempo y otras mejoras que hagan el juego más fácil. De este modo se añade una nueva forma de monetización al juego que puede atraer a un grupo distinto de usuarios de los que no se obtenía dinero a través de la publicidad y también se obtendría más dinero de los usuarios que ya usaban la publicidad, aumentado los ingresos del juego.

Bibliografía

ArtisTech Media LLC. (s.f.). ccMixter. Obtenido de http://ccmixter.org/

Corona Labs Inc. (s.f.). *Corona SDK*. Obtenido de http://www.coronalabs.com/products/corona-sdk/

Fine, R. J. (s.f.). *Unity MonoBehaviour Lifecycle*. Obtenido de http://www.richardfine.co.uk/2012/10/unity3d-monobehaviour-lifecycle/

Game Analytics. (s.f.). *Game Analytics*. Obtenido de http://www.gameanalytics.com/

GameSalad Inc. (s.f.). *GameSalad*. Obtenido de http://gamesalad.com/

Ideaworks3D Ltd. (s.f.). *Marmalade*. Obtenido de http://www.madewithmarmalade.com/

inXile Entertainment, Obsidian Entertainment. (s.f.). *Wasteland 2*. Obtenido de http://wasteland.inxile-entertainment.com/

Microsoft Corporation. (s.f.). *JScript.Net*. Obtenido de http://msdn.microsoft.com/en-us/library/72bd815a%28v=vs.71%29.aspx

Microsoft Corporation. (s.f.). *Microsoft C#*. Obtenido de http://msdn.microsoft.com/en-us/library/aa287558%28v=vs.71%29.aspx

Mika Mobile Inc. (s.f.). *Zombieville USA 2*. Obtenido de https://itunes.apple.com/es/app/zombieville-usa-2/id454781476?mt=8

mountain lion. (s.f.). *The Last Defender HD*. Obtenido de https://itunes.apple.com/us/app/the-last-defender-hd/id413923180?mt=8&ign-mpt=uo%3D4

N3V Game Pty Ltd. (s.f.). *Dead On Arrival*. Obtenido de https://itunes.apple.com/es/app/dead-on-arrival/id445751461?mt=8

Novell, Inc., Mono community. (s.f.). *MonoDevelop*. Obtenido de http://monodevelop.com/

Object Management Group, Inc. (s.f.). *Unified Modeling Language (UML)*. Obtenido de http://www.uml.org/

RevMob. (s.f.). *RevMob*. Obtenido de https://www.revmob.com/

Rovio Entertainment Ltd. (s.f.). *Bad Piggies*. Obtenido de http://www.badpiggies.com/

SoundBible.com. (s.f.). *SoundBible*. Obtenido de http://soundbible.com/

Tapgage Inc. (s.f.). *Tapgage*. Obtenido de http://www.tapgage.com/

TestFlight App Inc. (s.f.). *TestFlight*. Obtenido de https://testflightapp.com/

The Codehaus. (s.f.). *Boo*. Obtenido de http://boo.codehaus.org/

Unity Technologies. (s.f.). *Unity*. Obtenido de http://unity3d.com/

Anexo 1. Capturas de pantalla

Figura 15. Captura de pantalla del menú principal

Figura 16. Captura de pantalla del menú de opciones

Figura 17. Captura de pantalla del menú selección de nivel del modo clásico

Figura 18. Captura de pantalla del tutorial

Figura 18. Captura de pantalla del menú de pausa

Figura 19. Captura de pantalla del juego con la selección de arma desplegada

Figura 20. Captura de pantalla del juego en la que se ve a varios enemigos