Universitat Oberta de Catalunya
uoc.edu

Rush the Cosmos

Angel Camilo Palacios Garzón
Máster en Diseño y desarrollo de videojuegos
Unity 2D, Multiplayer

Helio Tejedor Navarro
Joan Arnedo Moreno

Barcelona, 14 de junio de 2020

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Rush the Cosmos*
Nombre del autor:	*Angel Camilo Palacios Garzón*
Nombre del consultor/a:	*Helio Tejedor Navarro*
Nombre del PRA:	*Joan Arnedo Moreno*
Fecha de entrega (mm/aaaa):	06/2020
Titulación::	*Máster en Diseño y desarrollo de videojuegos*
Área del Trabajo Final:	*Programación en Unity 2D, Juegos multijugador*
Idioma del trabajo:	*Castellano*
Palabras clave	*Android,Unity 2D, space shooter.*

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

Rush the Cosmos es un juego de espacial en 2D para varios jugadores inicialmente creado para dispositivos Android.

El objetivo es permitir que hasta 4 jugadores puedan unirse a una partida y competir en tiempo real por conquistar la mayor parte del escenario. Para ello, tienen que conquistar los planetas que componen el mapa.

El juego utiliza el clásico método de aplicación *server* + *client*. Un jugador crea la partida (*room*), este jugador actúa como *host* y cliente a la vez. El resto de jugadores se unen a esta *room* como clientes.

El uso del motor *Photon Bolt* para la gestión de *rooms* y la sincronización de los objetos de red han demostrado ser fiable y fluida. Su uso además es bastante similar al de la *HLAPI* de *Unity* utilizada durante el máster.

Las pruebas realizadas con jugadores voluntarios han mostrado un *engagement* positivo y proporcionado gran cantidad de *feedback* sobre nuevas funcionalidades que les gustaría ver en el juego.

La arquitectura del proyecto está pensada para que el juego sea iterable y, sin duda, esta primera versión se verá ampliada en los meses posteriores a la presentación y defensa del proyecto. Un ejemplo sería el modo campaña para un jugador, leaderboards globales, nuevos modelos de naves, armas y estructuras, etc.

Abstract (in English, 250 words or less):

Rush the Cosmos is a 2D space multiplayer game for Android devices.

The aim is to allow up to 4 players to join a match and compete in real time to conquer as much of the scene as possible. To do so, they must claim the planets scattered across the map.

The game uses a typical server + client application, where one of the players creates a room for the other players to join. This first player acts as host and client at the same time. The other players are clients on this host.

The use of the Photon Bolt engine to handle the matchmaking and the synchronization of the network objects proved smooth and reliable. Its use is also very similar to that of Unity's HLAPI that we already learnt during this master's program.

During the testing process with volunteers the game showed positive engagement and a lot of feedback from these users was noted about new features that they would like to see in the game.

This project's architecture has been thought keeping iterations in mind. This first version of the game will certainly be extended in the months following its presentation and defense. One example could be the Campaign mode, global leaderboards, new space ships, weapons and structures, etc.

Índice

1. Introducción

1.1 Contexto y justificación del Trabajo

Esta última década ha visto resurgir una segunda edad de oro de los videojuegos 2D. En la actualidad existe una larga lista de *space shooters 2D* tradicionales del estilo de los míticos *Galaga* y *Asteroids* que se pueden descargar tanto para PC como para dispositivos móviles. En la mayoría de casos, estos juegos se centran en la acción y en la habilidad del jugador para pilotar la nave. Pocos de estos títulos ofrecen un componente de estrategia al jugador y, ninguno que haya podido encontrar, ofrece acción en tiempo real para varios jugadores.

El objetivo del juego es ofrecer a los jugadores la posibilidad de competir desde sus dispositivos Android entre amigos o contra otros jugadores de todo el mundo. Organizando partidas relativamente cortas (15 minutos) en un entorno 2D dónde el jugador debe tener en cuenta logística, estrategia y habilidades para pilotar su nave.

1.2 Objetivos del Trabajo

1. Definir el *scope* del proyecto: qué entra y qué queda marcado como opcional para futuras iteraciones del juego.
2. Probar con éxito las mecánicas del juego en un dispositivo Android (prototipo jugable del proyecto). Este prototipo de un jugador permite probar que los controles del juego son intuitivos y que la jugabilidad es fluida.
3. Integrar la lógica de red: menú de lobby, gestión de partidas creadas, jugadores conectados, sincronización del estado del juego en los diferentes dispositivos. Una vez completado este punto se puede comenzar con el desarrollo del grueso de *features* y contenidos del juego.
4. Completar el desarrollo para una partida completa. Desde el Logo de inicio hasta la pantalla final mostrando los resultados de una partida con varios participantes.
5. Realizar con éxito una sesión de test con usuarios reales para obtener *feedback* y descubrir bugs.
6. Definir un calendario a corto-medio plazo para las futuras iteraciones del juego.

1.3 Enfoque y método seguido

Esta mezcla de subgéneros de acción 2D y estrategia supone que el producto será desarrollado desde cero. Aunque puede suponer más trabajo que adaptar un producto existente, también da libertad para definir la arquitectura del código con total libertad.

Una vez definida y probada la primera versión jugable del juego, el siguiente paso es hacer que esta funcione con más de un jugador adaptando el código para incorporar el *network engine* elegido.

El desarrollo teniendo en cuenta una única instancia del juego es similar al desarrollo con varias instancias en una misma partida, pero hay importantes diferencias que deben tenerse en cuenta. La más importante es que, en una partida de un juego *single-player* la instancia del juego es dueña de

absolutamente todos los objetos que existen (personaje principal, enemigos, disparos, explosiones). Por el contrario, en una partida multijugador, cada instancia del juego tendrá *ownership* (será dueña) de algunos de los objetos. Esta lógica permite que un jugador sólo pueda controlar y modificar el estado de su personaje principal, sin poder hacer lo mismo con los personajes de los demás jugadores.

Debido a que el juego, por el momento, es puramente orientado a multijugador. Una vez completada la integración del *network engine* en la primera versión jugable del proyecto, todo el desarrollo se ha realizado teniendo en cuenta esta lógica distribuida en red. Si desde el principio se hubiera querido tener un modo puramente *single player*, se podría haber desarrollado su código en paralelo, sin embargo, adaptar código desde una versión *multiplayer* a *single player* resulta más fácil que hacerlo en sentido inverso.

1.4 Planificación del Trabajo

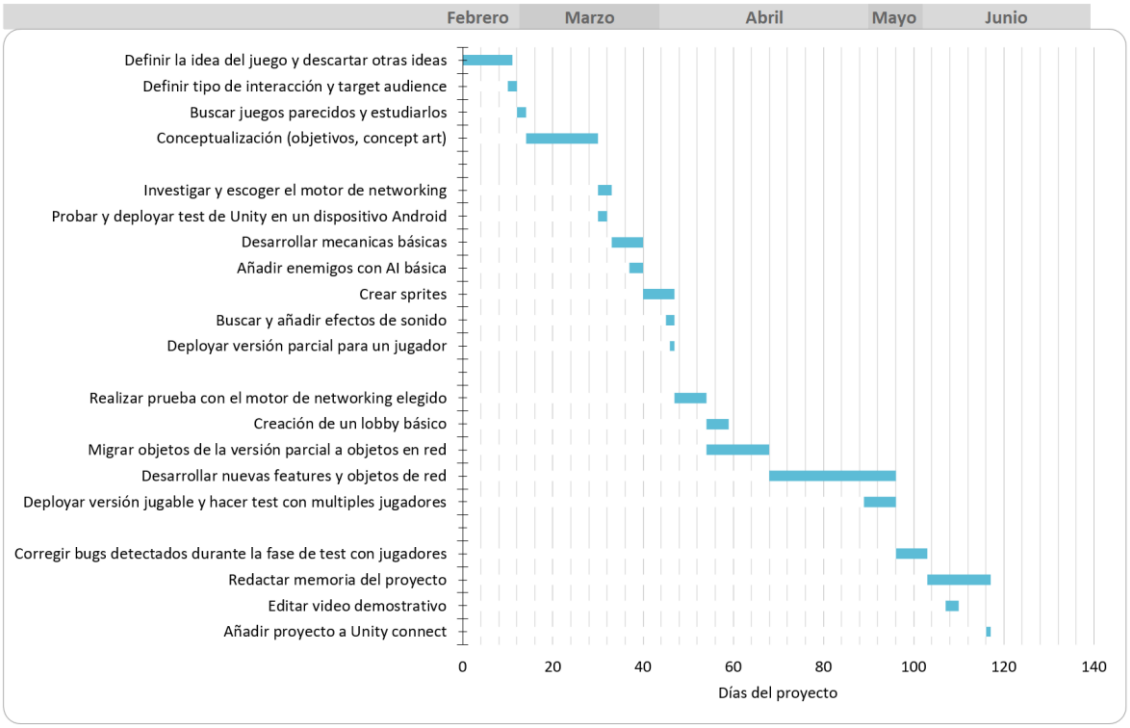
El proyecto está dividido en cuatro grandes bloques de tiempo que definen los hitos del desarrollo del juego.

Un primer bloque va destinado a la conceptualización y diseño del juego. Una vez finalizado el primer bloque le siguen tres bloques de desarrollo: uno para la prueba de concepto, otro para la creación de la primera versión jugable y finalmente un bloque para la creación de la versión final.

El siguiente diagrama agrupa las principales tareas desarrolladas dentro de cada uno de estos bloques.

Diagrama de Gantt del desarrollo del proyecto

TAREA	FECHA DE INICIO	FECHA DE FINALIZACIÓN	DÍA DE INICIO	DURACIÓN (EN DÍAS)
Diseño del videojuego				
Definir la idea del juego y descartar otras ideas	19-2	29-2	0	11
Definir tipo de interacción y target audience	29-2	1-3	10	2
Buscar juegos parecidos y estudiarlos	2-3	3-3	12	2
Conceptualización (objetivos, concept art)	4-3	19-3	14	16
Versión parcial				
Investigar y escoger el motor de networking	20-3	22-3	30	3
Probar y deployar test de Unity en un dispositivo Android	20-3	21-3	30	2
Desarrollar mecanicas básicas	23-3	29-3	33	7
Añadir enemigos con AI básica	27-3	29-3	37	3
Crear sprites	30-3	5-4	40	7
Buscar y añadir efectos de sonido	4-4	5-4	45	2
Deployar versión parcial para un jugador	5-4	5-4	46	1
Versión jugable				
Realizar prueba con el motor de networking elegido	6-4	12-4	47	7
Creación de un lobby básico	13-4	17-4	54	5
Migrar objetos de la versión parcial a objetos en red	13-4	26-4	54	14
Desarrollar nuevas features y objetos de red	27-4	24-5	68	28
Deployar versión jugable y hacer test con multiples jugadores	18-5	24-5	89	7
Versión final				
Corregir bugs detectados durante la fase de test con jugadores	25-5	31-5	96	7
Redactar memoria del proyecto	1-6	14-6	103	14
Editar video demostrativo	5-6	7-6	107	3
Añadir proyecto a Unity connect	14-6	14-6	116	1



1.5 Breve sumario de productos obtenidos

- Prueba de concepto: permite ver que los conceptos de la nave, los controles y el escenario con planetas es viable y la jugabilidad es fluida para el usuario. En caso de fallos claros en la jugabilidad se puede corregir de cara al siguiente producto.
- Primera versión jugable: incorpora toda la jugabilidad y capacidad para multijugador. Permite recibir feedback de un primer grupo de usuarios para corregir errores y añadir mejoras.
- Versión final: incluye los detalles obtenidos durante la fase de pruebas de la primera versión jugable. Este producto es la primera versión completa del juego, aunque su jugabilidad se puede ir extendiendo mediante iteraciones de manera indefinida.

1.6 Breve descripción de los otros capítulos de la memoria

- Estado del arte: breve análisis del proyecto en comparación con títulos parecidos.
- Definición del juego: explicación del juego (contexto, objetivos, personajes, etc.).
- Diseño técnico: explicación sobre las herramientas empleadas en el proyecto, recursos y arquitectura del proyecto.
- Diseño de niveles: descripción del escenario principal en el que se realiza la acción del juego.
- Manual de usuario: guía a la que los jugadores pueden acudir para iniciarse o conocer algún elemento concreto del juego.
- Conclusiones: resultados obtenidos y futuro del proyecto.
- Glosario: explicación sobre algunos términos utilizados en este documento.
- Bibliografía: referencias a otros proyectos.

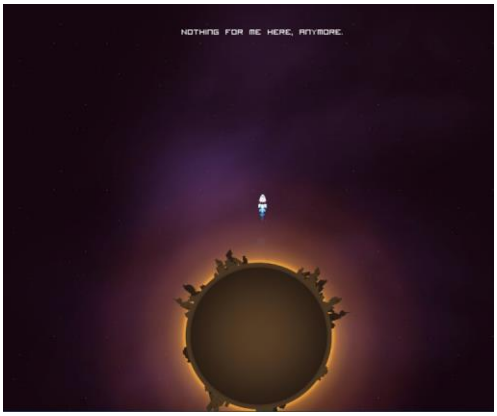
2. Estado del arte

Como se comenta brevemente en el capítulo de introducción. Ninguno de los juegos analizados coincide en sus dinámicas con las de este proyecto. Sin embrago, hay algunos juegos que guardan similitudes en sus mecánicas y controles. A continuación analizamos algunos de ellos.

2.1 Last Horizon

Plataforma: PC (Windows), Mac OS X
Desarrollador: Pixeljam
Lanzamiento: 18/11/2015

Ofrece una experiencia de exploración y supervivencia espacial para un jugador. El objetivo es encontrar un nuevo planeta en el que habitar. Dispone de un tiempo limitado para conseguir su siguiente objetivo a medida que las reservas de oxigeno de la nave van disminuyendo.



2.2 SLI-FI: 2D Planet Platformer

Plataforma: PC (Windows), Mac OS X,SteamOS + Linux
Desarrollador: Etaxoft
Lanzamiento: 08/09/2017

En lugar de una nave, el personaje principal es un pequeño ser, parecido al protagonista del popular juego *Super Meat Boy*, que va dando saltos de un planeta a otro. El jeugo nos permite explorar varios plaentas sólo o en modo coperativo con otro jugador.

2.3 Endless Sky

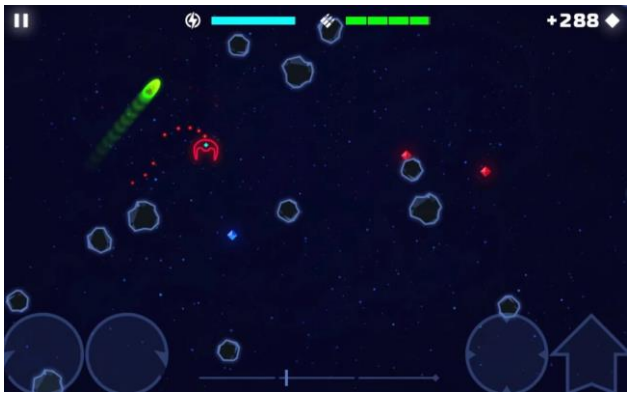
Plataforma: PC (Windows), Mac OS X, SteamOS + Linux
Desarrollador: Michael Zahniser
Lanzamiento: 30/10/2015

Enfocado en el comercio y el combate. Este juego para un solo jugador permite mejorar la nave y librar grandes batallas contra NPCs aunque tiene también un componente táctico muy grande. En este juego se mezcla acción en tiempo real y mapa de la galaxia. Al llegar a un planeta no aterrizamos físicamente. En su lugar, accedemos a un menú en el que gestionamos las acciones que podemos realizar en ese planeta.

2.4 Neon Spaceships

Plataforma: Android
Desarrollador: TAY-117
Fecha de lanzamiento: 05/08/2018

Es el único juego de la lista desarrollado para jugar en Android. Está fuertemente inspirado en el clásico *Asteroids* añadiendo la posibilidad de mejorar la nave. Este juego sigue el modelo *freemium* que está muy de moda en el sector de los videojuegos para dispositivos móviles. Podemos mejorar la nave jugando y acumulando las divisas del juego o con dinero real a través de pagos dentro de la app.



Tras analizar los títulos anteriores concluimos que ninguno de ellos, ni ningún otro título encontrado en el mercado durante el análisis, cumple con todas las características siguientes:

- Space shooter 2D
- Desarrollado para plataformas móviles (específicamente Android)
- Acción en tiempo real
- Multi-jugador (hasta 4 jugadores en una misma partida)
- Con aspectos del genero de estrategia

Existe, por tanto, un espacio del mercado que este proyecto puede llenar y que todavía no está tan saturado como otros géneros de videojuegos.

3. Definición del juego

El juego consiste en una aventura espacial en 2D. La acción del juego transcurre en el espacio exterior, en un único y extenso escenario en dos dimensiones compuesto por varios planetas y zonas por explorar. El jugador controla una nave espacial y compite por controlar el sector y sus recursos contra otros jugadores. El escenario contiene un sistema de planetas que los jugadores deberán colonizar. Una vez colonizado un planeta, el jugador puede crear estructuras que le ayuden a producir recursos y a defender el planeta de los otros jugadores. Tendremos que enfrentarnos a planetas hostiles y otros jugadores. El objetivo del juego es expandir nuestra influencia por el sector, esta se mide en puntos de influencia. El jugador debe acumular recursos, conquistar planetas y usar esos recursos para construir edificaciones en los planetas bajo su control. Estos edificios proporcionan nuevos recursos (economía) y defensas ante amenazas (defensa). Gana la partida el jugador con mayor número de puntos de influencia al terminar el límite de tiempo de la partida.

3.1 El jugador

Antes de entrar en la partida, el jugador elige un nombre con el que mostrarse en el juego que lo identifica del resto de jugadores. A continuación toma el mando de una nave y su tripulación. La acción transcurre en tercera persona en un mundo 2D. La cámara sigue en todo momento a la nave del jugador.

3.2 Interacción entre los actores del juego

Por un lado tenemos a los jugadores rivales. La partida puede tener hasta 4 jugadores, estos jugadores son hostiles los unos a los otros. En caso de encontrarnos con uno de ellos por el escenario, podemos optar por huir o enfrentarnos a ellos utilizando el armamento de nuestra nave. En caso de ser destruida, la nave volverá a aparecer en el punto de inicio de la partida. Un jugador rival puede optar también por atacar y conquistar uno de nuestros planetas. Para evitarlo, debemos construir defensas planetarias que atacarán automáticamente a cualquier enemigo que se aproxime

3.3 Objetivos

El objetivo es expandir nuestra influencia por todo el escenario conquistando planetas. Los planetas proporcionan también recursos y sus estructuras pueden mejorarse a lo largo de la partida. Para colonizar un planeta, el jugador aterrizar en él. La partida tiene un límite de tiempo y un contador de puntos de influencia. Gana el jugador que acumule más puntos al final de la partida.

4. Diseño técnico

4.1 Plataforma de destino

Desde un principio la decisión ha sido desarrollar el proyecto para dispositivos *Android* por motivos de presupuesto. El motivo principal es no dispongo de un *iPhone*, de un ordenador *Mac* ni de una cuenta *Apple Developer* (requisito para poder deployar y publicar el proyecto en iPhone). En el caso de Android dispongo de tres dispositivos en los que probar, además de los dispositivos virtualizados. Finalmente, una pequeña encuesta en mi círculo de conocidos sobre posibles *beta-testers* para el proyecto me permitió ver que tenía más *testers* potenciales con dispositivos Android que con iPhone.

4.2 Entorno de desarrollo

Durante la fase de definición del proyecto tuve en cuenta dos entornos como posibles candidatos para desarrollar el juego: Unity y Godot.

4.2.1 Unity

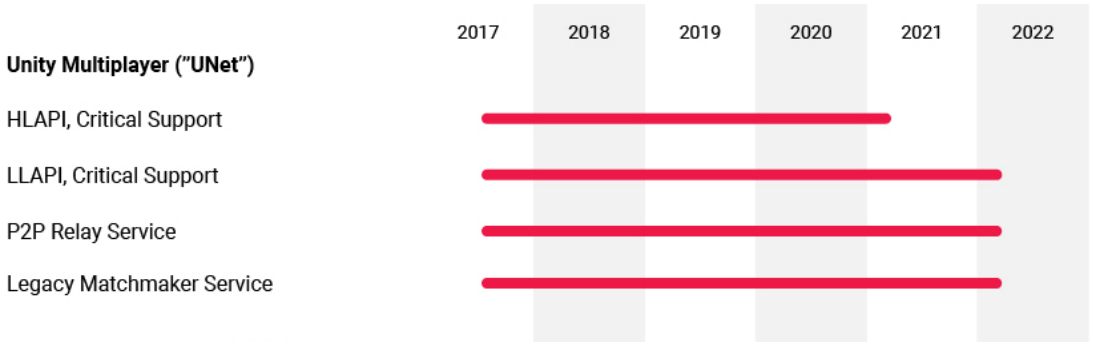
Es el entorno que más hemos utilizado a lo largo del programa de este máster y el que tiene la comunidad más grande en estos momentos. La curva de aprendizaje para trabajar con este entorno no es especialmente rápida pero es el motor con el que tengo más experiencia. Por este motivo y el que veremos en el punto XXX Unity es el motor elegido para la realización del proyecto.

4.2.1 Godot

Ofrece un desempeño excepcional en entornos 2D, por encima del que ofrece Unity. Tiene una comunidad creciente. Una gran ventaja que ofrece a los desarrolladores, y que lo hace especialmente atractivo para desarrolladores independientes y pequeños estudios, es que es un motor libre y de código abierto. Todo el contenido creativo hecho con este motor pertenece únicamente al desarrollador. A diferencia de Unity, todos los beneficios son para el desarrollador, independientemente del número de copias vendidas. No he optado por este motor para este proyecto pero sin duda es una herramienta potente y con una comunidad creciente.

4.3 Motor de red

Durante este programa de master hemos trabajado con UNet de Unity. Esta solución proporciona dos APIs (HLAPI y LLAPI) con funcionalidades muy potentes de lógica de *networking* para sincronizar el estado de nuestros juegos a través de varios protocolos de red. UNet lleva varios años marcada como obsoleta y prevé dejará de prestar servicios a los juegos que la utilizan actualmente en 2022.



Por este motivo quedó descartada para este proyecto y en su lugar decidí utilizar Photon Bolt.

4.3.1 Photon Bolt

Existen cada vez más soluciones en el mercado de *back-end as a service* que liberan al desarrollador de tener que gestionar toda la infraestructura que hay detrás de un juego para tareas de sincronización de datos, *matchmaking*, persistencia de datos, pagos, etc. Una de estas empresas es *Photon*. Ofrecen varias soluciones distintas que se adaptan a distintos tipos de juego. Para este proyecto he optado por utilizar *Photon Bolt*. Este motor se integra al entorno de *Unity* y permite la creación de partidas *hosteadas* por uno de los jugadores (no necesita un servidor dedicado) y ofrece *matchmaking* en la nube. La cuenta gratuita usada para este proyecto permite hasta 20 jugadores conectados simultáneamente.

4.4 Recursos

4.4.1 Arte y sprites

Todos los sprites son propios. Varios de ellos han sido reciclados de un antiguo proyecto anterior llamado *Mars Uprising* que realicé con algunos compañeros durante el grado de ingeniería informática. Se trataba de un juego estilo *tower defense* para dispositivos Android. Doy mi especial agradecimiento a mi amigo Enric García por realizar unos retoques para las versiones finales de la nave y los planetas.

4.4.2 Efectos de sonido

La totalidad de los efectos de sonido han sido descargados de la base de datos de la web https://freesound.org/ bajo la licencia *Creative Commons License*.

4.4.3 Música

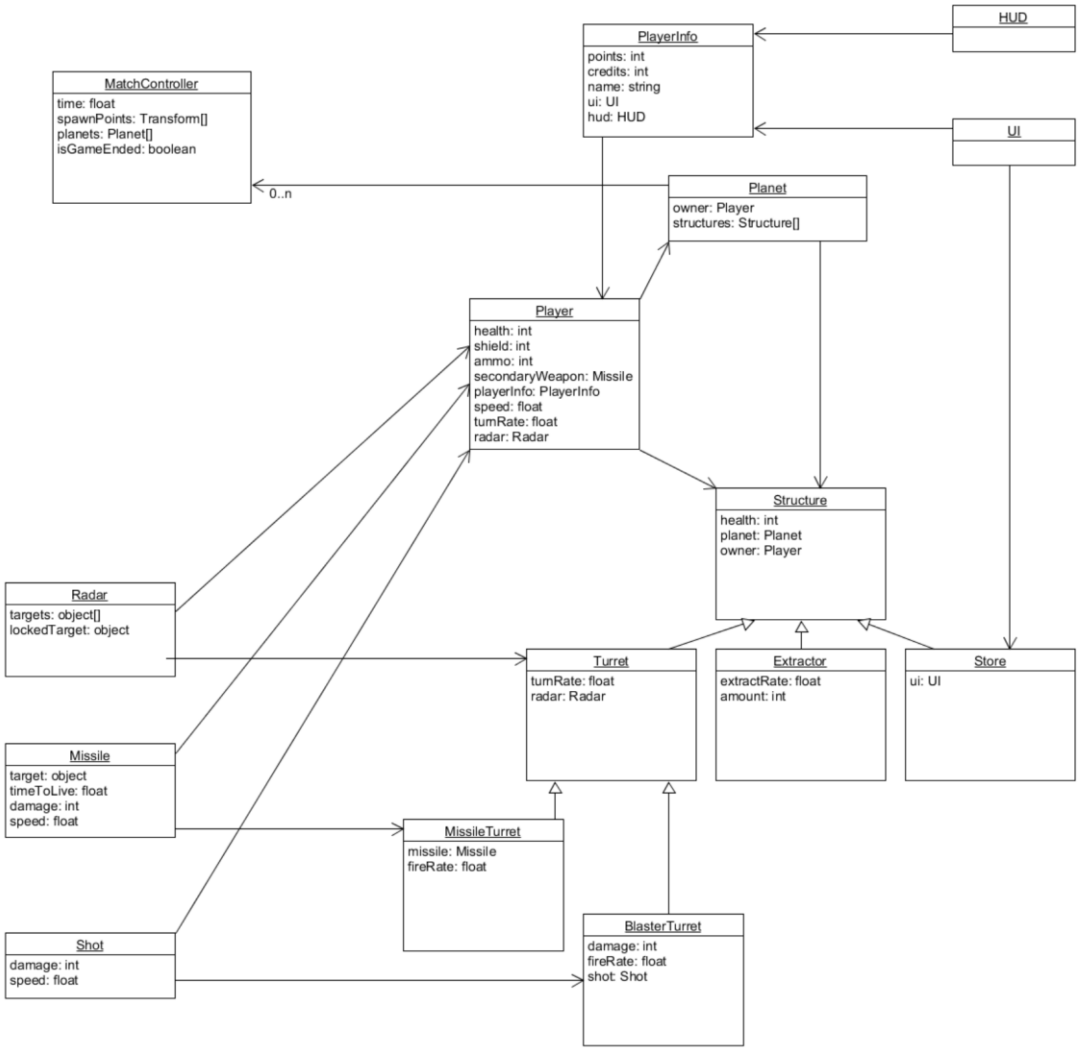
- Menú principal: The biggest discovery – Patrick De Arteaga
- Escena principal: Not giving up – Patrick De Arteaga

4.5 Otras herramientas

- Adobe Photoshop CS6: para la edición y creación de sprites.
- PlayerPrefs: funcionalidad de *Unity* para persistir datos. Usada para guardar el nombre del jugador en el dispositivo cuando se cierra la app.
- Visual Studio Code: IDE usado para programar. El lenguaje de programación del proyecto es C#.

4.6 Esquema de arquitectura

El siguiente diagrama de clases muestra las principales interacciones de los objetos del juego durante la partida.



4.6.1 MatchController

Se encarga de poblar el escenario con los objetos estáticos (planetas, cajas de power-ups) cuando empieza la partida. Sólo el *MatchController* del jugador que hace de *host* hace esta acción.

Controla también el contador del final de la partida. Y contiene las coordenadas de los puntos de *spawn* para los jugadores.

4.6.2 Player

Contiene la lógica de la nave que controla el jugador y su información acerca de la salud de la nave, el nivel de carga del escudo, la cantidad de munición del arma secundaria y el tipo arma principal y secundaria que tiene asignados.

4.6.3 PlayerInfo

Contiene información del jugador sobre su estado en la partida: puntos, créditos y su nickname.

4.6.4 HUD

Esta clase es utilizada para enseñar información del estado del jugador y para recibir los inputs que controlan la nave (joystick y botones de disparo).

4.6.5 UI

Esta clase se utiliza para acciones del jugador que se realizan sin los controles de la nave (menú in-game, interfaz de construcción de estructuras, interfaz de compras en la tienda).

4.6.6 Planet

Contiene la lógica del planeta. Guarda información sobre quién es su propietario, qué estructuras tiene. Contiene también la lógica con las reglas para conquistar el planeta.

4.6.7 Structure

Esta clase genérica define cualquier estructura que se puede construir en un planeta. Guarda información acerca de sus puntos de salud, en qué planeta está construida y quién es el usuario propietario.

4.6.8 Turret

Esta clase guarda lógica sobre cómo se comporta una torreta. Utiliza su radar para elegir objetivos. Sabe qué jugador es su propietario y por tanto no lo ataca.

4.6.9 MissileTurret

Esta clase hereda de *Turret*. Tiene asignado un tipo de misil y una cadencia de disparos. Instanciará un objeto de la clase *Missile* asignándole como objetivo el objeto devuelto por el radar.

4.6.10 BlasterTurret

Similar a la clase anterior. En lugar de tener asignado un tipo de *Missile* tiene una referencia a la clase *Shot.* Instanciará los objetos *Shot* sólo cuando el cañón de la torreta está orientado hacia su objetivo.

4.6.11 Extractor

Genera recursos periódicamente según los valores de sus atributos *extractRate* y *amount* y se los pasa al *Player* que tenga asignado como *owner*.

4.6.12 Store

Gestiona la lógica para activar la interfaz de compras y enlazar el resultado de esas interacciones con el *owner* de la tienda, por ejemplo recargar sus misiles.

4.6.13 Radar

Contiene lógica sobre qué objetos se consideran objetivos y cuáles no. Elige un objetivo de entre los que haya disponibles para que el dueño del radar pueda realizar acciones, como lanzar un misil u orientar el cañón de una torre blaster hacia ese objetivo.

4.6.14 Missile

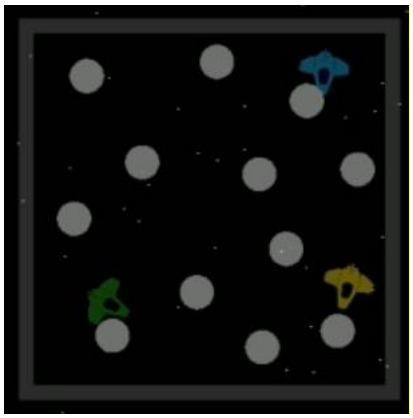
Define un misil y su comportamiento. Guarda una referencia del objetivo, define cuánto tiempo puede el misil rastrear su objetivo, el daño que causa al impactar, su velocidad y su radio de giro.

4.6.15 Shot

Similar a la clase *Missile* pero con una lógica más sencilla. Una vez instanciado, el disparo continua en línea recta hasta impactar con algo. A diferencia de los misiles, y ya que el comportamiento del disparo es tan determinista, este objeto no es un objeto de red (*Bolt entity*). Cada instancia del juego (el juego en cada dispositivo) crea una instancia local del disparo. Esto reduce enormemente la cantidad de objetos de red que el motor *Photon Bolt* tiene que tener en cuenta, ya que los disparos se instancian con mucha más frecuencia que los misiles.

5. Diseño de niveles

El juego contiene un único nivel principal con planetas repartidos por diferentes zonas. Todos estos planetas aparecen en gris al iniciar la partida indicando que no pertenecen a ningún jugador. Cada uno de los cuatro jugadores aparecerá en una esquina diferente del mapa. El planeta más cercano a su punto de *respawn** está completamente deshabitado. Esto está hecho así para que cada jugador pueda reclamar un planeta rápidamente al iniciar la partida si lo desea.



Por todo el mapa se generan también diferentes cajas de *power-ups** que los jugadores podrán aprovechar para mejorar su nave. El siguiente capítulo contiene información sobre los diferentes tipos de *power-up* que el jugador se puede encontrar por el mapa.

Algunos de los planetas neutrales contienen estructuras defensivas. Estos planetas repelen a cualquier jugador que se acerque y tardan más tiempo en ser conquistados mientras sus estructuras sigan en pie.

Unos bordes invisibles impiden al jugador abandonar el escenario del nivel. Estos bordes también son detectados por la cámara principal del juego que se detendrá al llegar detectar que el jugador ha llegado a un borde. De este modo no se enseña lo que hay más allá de los límites del escenario.

6. Manual de usuario

6.1 Requerimientos de Hardware

- **Versión de Android**: el juego está desarrollado para cualquier dispositivo Android con versión 4.4 *Kitkat* o posterior.

- **Pantalla**: el juego ha sido desarrollado para una pantalla con un *aspect ratio** 16:9 (1280x720, 1920x1080, 2960x1440) aun que se adapta también a otros tamaños de pantalla.

6.2 Instalación

Para instalar el juego es necesario descargar el archivo *RushTheCosmos.apk* disponible en el repositorio del proyecto dentro de la carpeta *Installer*. Una vez descargado, el dispositivo detectará el archivo y guiará al usuario durante la instalación.

6.3 Guía del juego

6.3.1 Menú principal

Desde este menú tenemos la posibilidad de convertirnos en el *host** de una partida, unirnos a una partida ya creada y configurar nuestro nombre de usuario, este es el nombre que se muestra en el *leaderboard** al final de una partida junto a la puntuación obtenida.

6.3.2 Configurar nombre del jugador

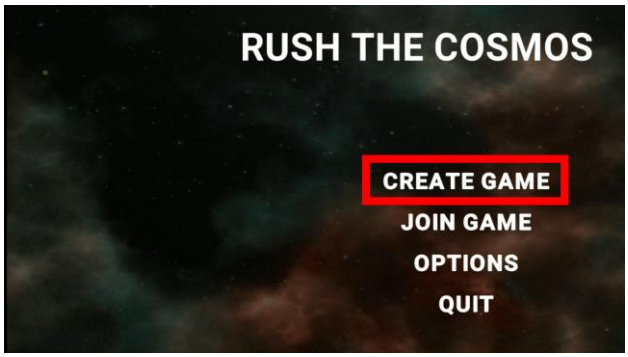
Se accede desde el menú principal, tocando el botón "*OPTIONS"*.

Es recomendable realizar esta acción la primera vez que ejecutamos el juego. Sólo es necesario realizar esta acción una vez. El nombre de usuario quedará guardado en el dispositivo. Al tocar el campo de texto se desplegará el teclado del dispositivo que nos permite escribir el nombre de usuario que queramos.

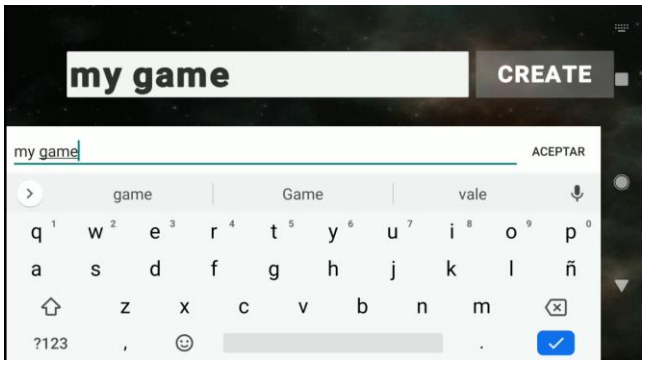
Una vez introducido el nombre podemos volver al menú principal tocando el botón '*BACK*' en la parte inferior de la pantalla

6.3.3 Crear una partida

Al menú para crear partida se accede desde el menú principal tocando el botón '*CREATE GAME'*.

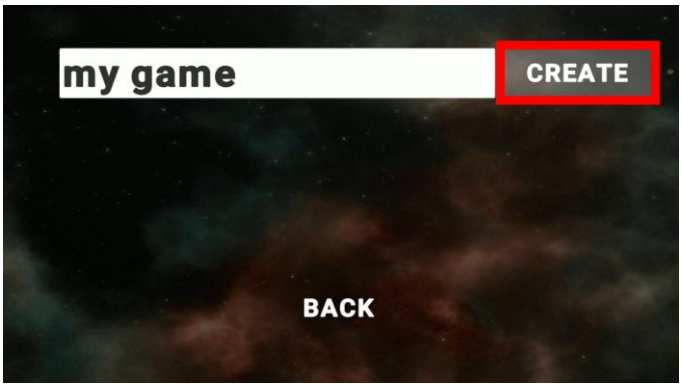


A continuación debemos escribir en el campo de texto un nombre con el que permitir a los otros usuarios identificar nuestra partida.



NOTA: en caso de no darle un nombre a la partida se le asigna un identificador aleatorio.

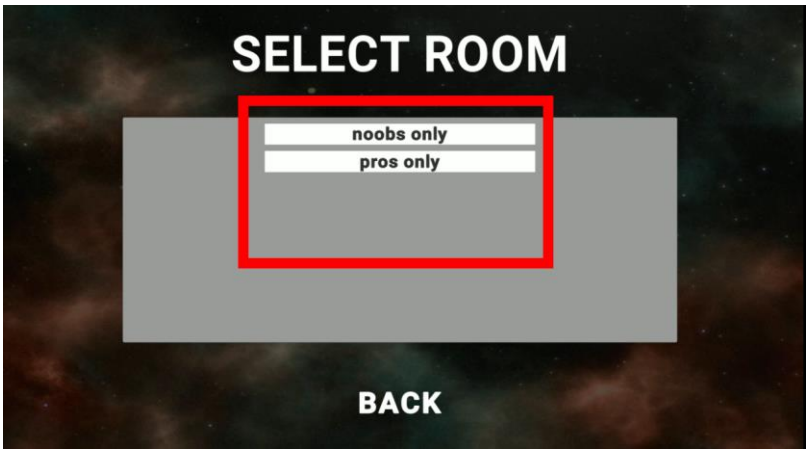
Finalmente creamos la partida tocando el botón '*CREATE*'.



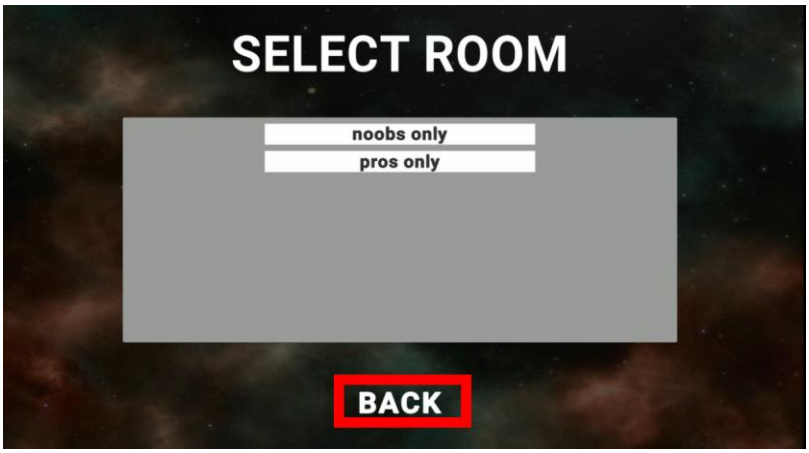
6.3.4 Unirse a una partida

Para acceder al listado de partidas en curso tocamos el botón '*JOIN GAME*' del menú principal.

Cada partida se muestra en forma de botón dentro de una lista. Para unirse a la partida basta con tocar uno de los botones de la lista.



En caso de querer volver al menú principal encontramos el botón '*BACK*' en la parte inferior de la pantalla.



6.3.5 HUD & controles

El *HUD (head-up display)* es como se conoce a todos los elementos que se proyectan por pantalla encima del escenario del juego. Estos nos proporcionan información importante y, en el caso de los dispositivos móviles, también actúan como controles táctiles.

6.3.5.1 Minimapa

Sirve para orientarse por el escenario y proporciona información acerca de la ubicación de los demás jugadores y de a quién pertenece cada planeta.

6.3.5.2 Contador de puntos y créditos

Mantiene actualizados los puntos del jugador y la cantidad de créditos que tiene acumulados.

6.3.5.3 Timer

Muestra el tiempo que queda para que termine la partida.

6.3.5.4 Barras de escudo y de vida (integridad del casco de la nave)

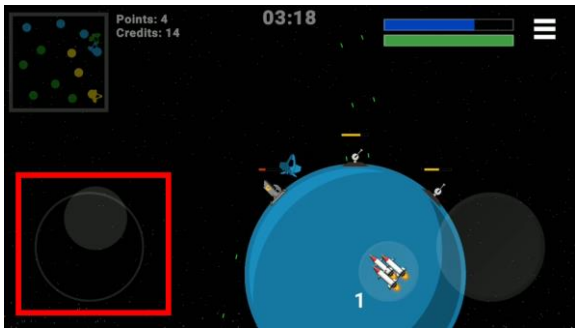
La barra superior muestra en azul el estado de los generadores de escudo de la nave. La barra inferior muestra el estado del casco de la nave. Se muestra en verde cuando la nave se encuentra en buen estado, pasando a amarillo cuando la integridad del casco decae y finalmente en rojo cuando el casco está en estado crítico.

6.3.5.5 Controles

Los controles de la nave ocupan las esquinas inferior izquierda y derecha de la pantalla. Son lo suficientemente grandes para que el usuario pueda dar con ellos sin dejar de prestar atención a lo que pasa en la partida pero buscan obstaculizar lo menos posible la visibilidad del jugador con un tono semitransparente.

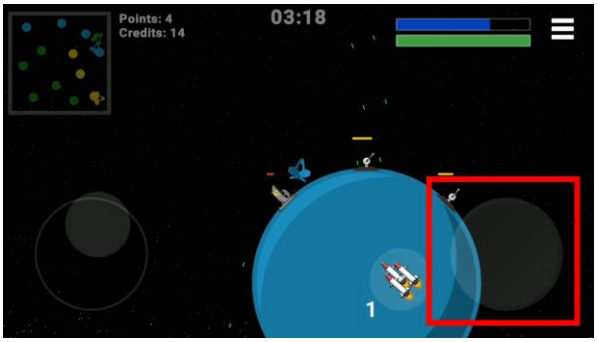
6.3.5.5.1 Joystick

Se encuentra en la esquina inferior izquierda y controla la dirección de la nave. Esta se desplazará en la dirección en la que se mueva el joystick. El morro de la nave también girará automáticamente para orientarse en la dirección del joystick.



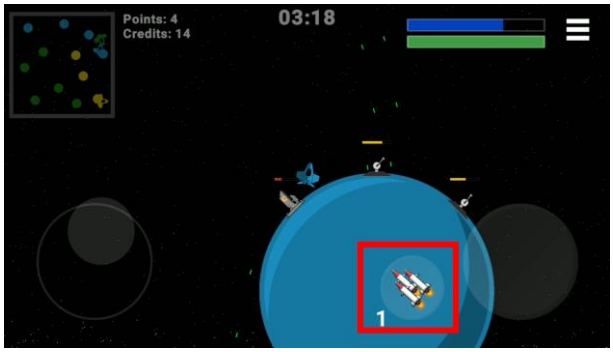
6.3.5.5.2 Botón de arma principal

Este botón se encuentra en la esquina inferior derecha, simétricamente opuesto al joystick. Dispara los blasters principales de la nave. Esta arma tiene munición ilimitada aunque provoca menos daños que el arma secundaria.



6.3.5.5.3 Botón de misiles

Los misiles, o arma secundaria, se utilizan pulsando este botón. A diferencia del arma principal, el arma secundaria tiene munición limitada. La cantidad de misiles disponibles se muestra junto al botón. Si se utilizan todos los misiles restantes el botón queda deshabilitado hasta que el jugador reponga municiones.



6.3.6 Combate

Esta es la parte de la partida donde los jugadores miden su destreza con los controles de la nave. La habilidad en el combate puede proporcionar una gran ventaja al jugador en el resultado de la partida.

El método más efectivo de destruir al rival es situarse en su retaguardia y disparar la mayor cantidad de armas posibles.

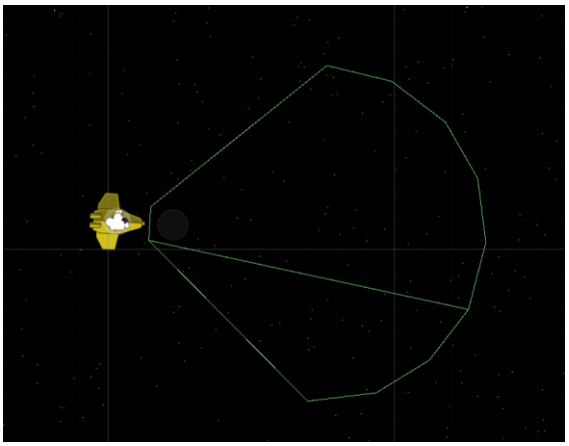
6.3.6.1 Cañones blaster

Munición: infinita
Puntos de daño: 5

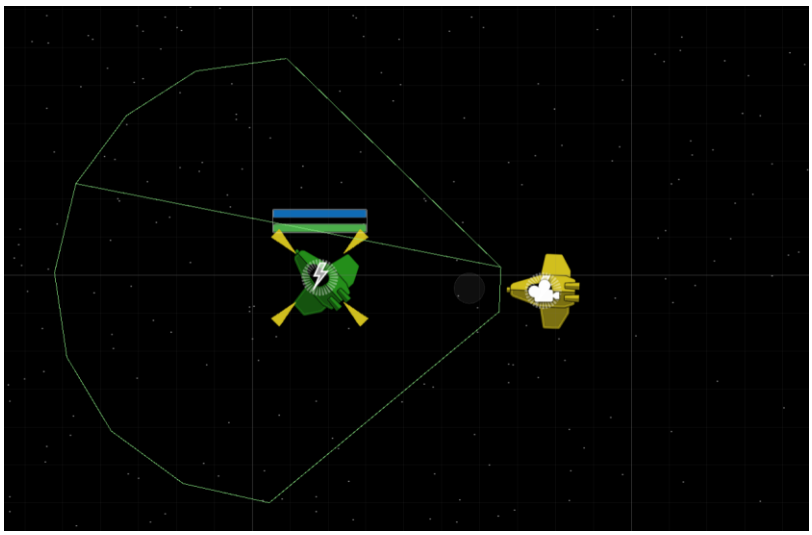
El arma principal del jugador consiste en dos cañones blaster automáticos que disponen de munición indefinida. Cada disparo produce un daño leve en el casco del rival y en las estructuras enemigas. Los pilotos más hábiles consiguen acertar con ambos cañones, duplicando así la cantidad de daño infligido al rival.

6.3.6.2 Radar de la nave

El radar de la nave permite al jugador fijar un objetivo para sus misiles dirigidos. Este radar tiene forma de cono y está enfocado hacia el frente desde el morro de la nave. El jugador, por tanto, debe orientar el morro hacia su objetivo, siendo imposible fijar naves y estructuras que se encuentre detrás de la nave.



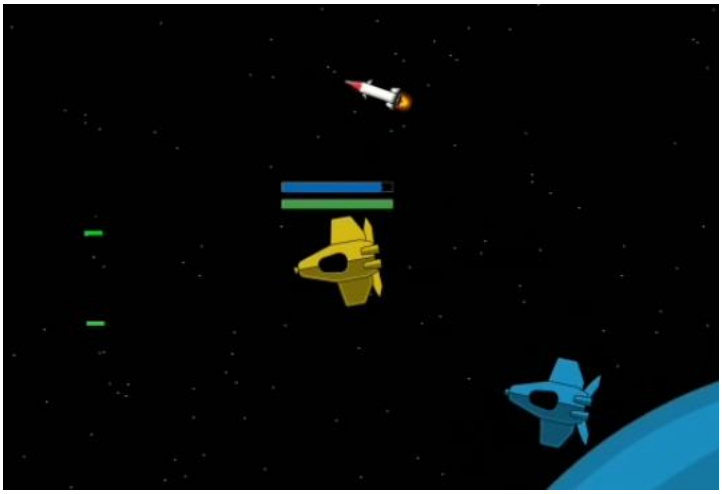
El área del radar no es visible durante la partida, sin embargo, un objetivo que ha sido fijado por el radar se marca con una cruz del color del jugador que lo tiene fijado. Esta señal es visible para todos los jugadores y sirve para saber cuándo lanzar nuestros misiles como para saber cuándo un rival nos tiene fijados en su radar.



6.3.6.3 Misiles dirigidos

Munición inicial: 3
Munición máxima: 10
Puntos de daño: 40

Cada jugador inicia la partida con tres misiles en su inventario. Esta poderosa arma secundaria causa mucho más daño que los cañones blaster. Para asegurar el disparo, el jugador debe intentar fijar el objetivo en el centro de su radar. Un piloto con buenos reflejos es capaz de esquivar los misiles enemigos. El misil es más rápido que una nave pero hace que su ratio de giro sea menor. Un misil que ha pasado de largo su objetivo tiene pocas probabilidades de poder dar media vuelta y acertar en su objetivo. El misil mantendrá su objetivo fijado durante unos segundos, tras los cuales volará a la deriva y, finalmente, explotará al llegar al límite de su autonomía.

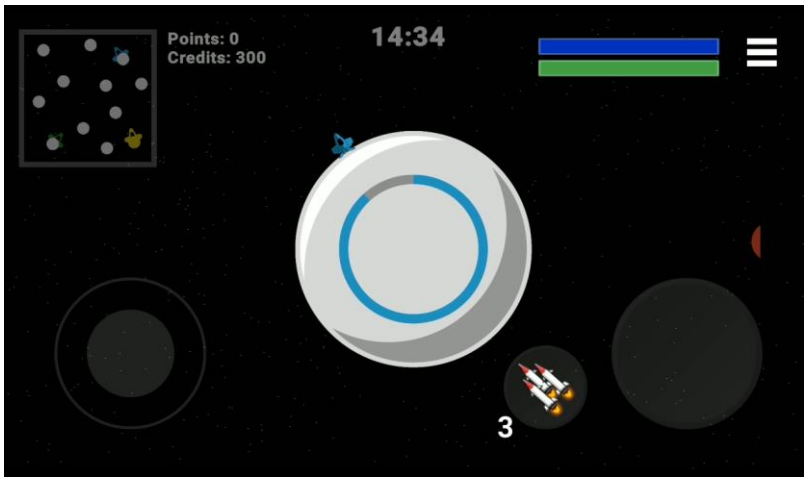


6.3.7 Loot

El riesgo del combate proporciona unos grandes beneficios para el jugador que sale victorioso. Al ser destruido, todo jugador deja tras de sí el total de sus misiles junto con la mitad de los créditos que tuviera acumulados en el momento de su derrota. El primer jugador en llegar a ese *loot* puede conseguir una importante ventaja sobre sus rivales.

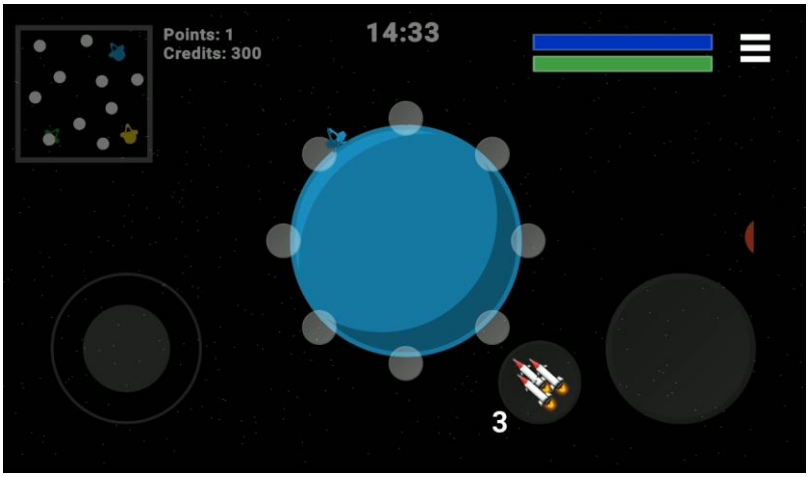
6.3.8 Conquistar un planeta

Es el objetivo principal del juego. Para reclamar el planeta para sí, un jugador debe aterrizar en su superficie. La nave se verá atraída hacia el planeta por su gravedad si se encuentra lo suficientemente cerca. Al detectar un campo gravitatorio se desplegará automáticamente el tren de aterrizaje de la nave. Una vez en la superficie del planeta, el jugador deberá permanecer en ella durante un tiempo determinado. Este tiempo es representado mediante una barra de progreso que se muestra dentro del planeta.



Nota: cuantas más estructuras tenga un planeta, más tiempo requerirá su conquista.

Una vez conquistado el planeta, su superficie adquirirá el color del jugador y el jugador podrá ver los botones de construcción de estructuras en la superficie del planeta.



6.3.9 Estructuras, defensa y economía

Un planeta vacío no proporciona mucho valor al jugador y es susceptible de ser robado con facilidad. Los jugadores deben procurar la defensa de sus nuevas conquistas así como la construcción de una economía que les proporcione recursos con los que expandirse con más facilidad. Las siguientes estructuras están disponibles para todos los jugadores.

6.3.9.1 Torre blaster

Precio: 100c
Puntos de daño por disparo: 1

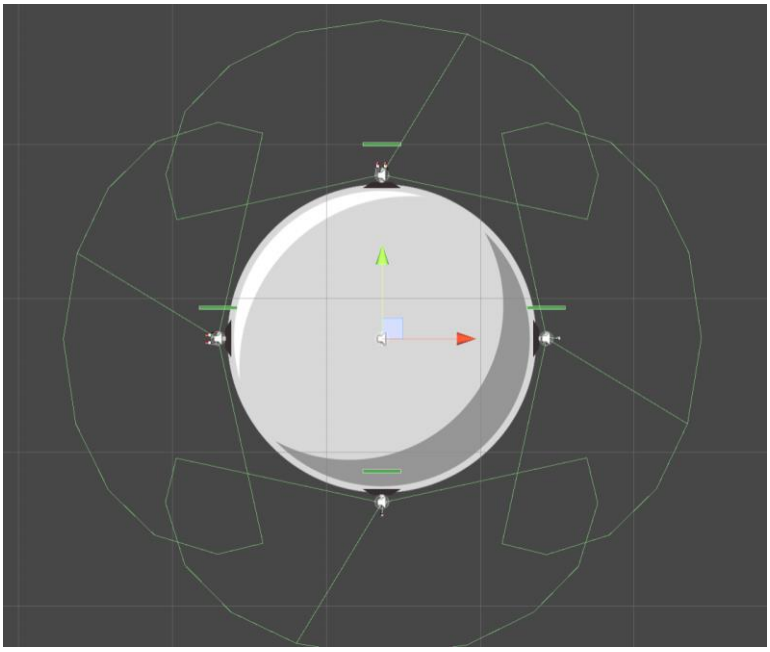
Esta torre es la mejor en relación calidad/precio. Disparan una ráfaga ilimitada de blasters a cualquier objetivo hostil que entre en su radio de acción. Cada disparo inflige un daño menor pero la cadencia de disparo (disparos por minuto) es bastante alta. Su velocidad de giro es limitada, por lo que conviene atacar estas torres por el lado contrario al que la torre está orientada y retirarse en cuanto la torre reoriente el cañón.

6.3.9.2 Torre de misiles

Precio: 200c
Puntos de daño por disparo: 25

Esta torre es el doble de cara que una torre blaster pero su potencial de daño es muy superior. También posee la ventaja de poder empezar a disparar antes de que los lanzadores estén orientados hacia el objetivo. El comportamiento de sus misiles es el mismo que el de los misiles lanzados desde una nave. Una vez disparado el misil, se dirigirá hacia su objetivo. Aunque la torre pueda detectar naves en todas direcciones, es aconsejable atacarla por el lado contrario al que esté orientada, ya que el recorrido del misil en vuelo será mayor.

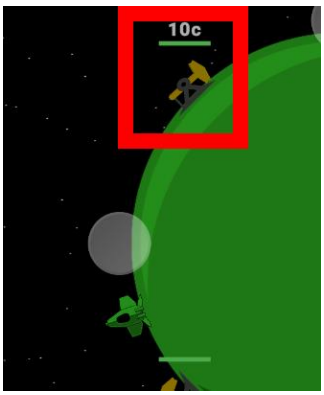
Nota: las torres disponen de un radar similar al de la nave pero con un área mayor. El área no es visible durante la partida. La torre elegirá aleatoriamente un objetivo de entre todos los que se encuentren dentro del área del radar.



6.3.9.3 Extractor

Precio: 150c
Producción por minuto: 120c

Extrae recursos del planeta y los convierte en créditos que se ingresan en la cuenta del dueño del planeta. Cada 5 segundos se producirá un ingreso de 10 créditos con una notificación que sólo es visible para el jugador beneficiado.



6.3.9.4 Tienda

Precio: 100

Este edificio da acceso al menú de compras de para la nave. Utilizado de forma correcta puede suponer una gran ventaja para su propietario. Los artículos a la venta son los siguientes.

6.3.9.4.1 Recargar misil

Coste: 25c

Compra un misil que se añade de inmediato al inventario del jugador. Para comprar varios misiles el jugador puede pulsar repetidas veces este botón. Cuando se alcance la capacidad máxima de misiles que puede llevar el jugador este botón quedará deshabilitado. Lo mismo pasa si el jugador no tiene suficientes créditos.

6.3.9.4.2 Reparar la nave

Coste: 100c

Permite reparar al completo la nave. Si la nave ya está al máximo de salud o el jugador no tiene suficientes créditos, este botón aparecerá deshabilitado.

6.3.9.4.3 Recargar escudo

Coste: 75c

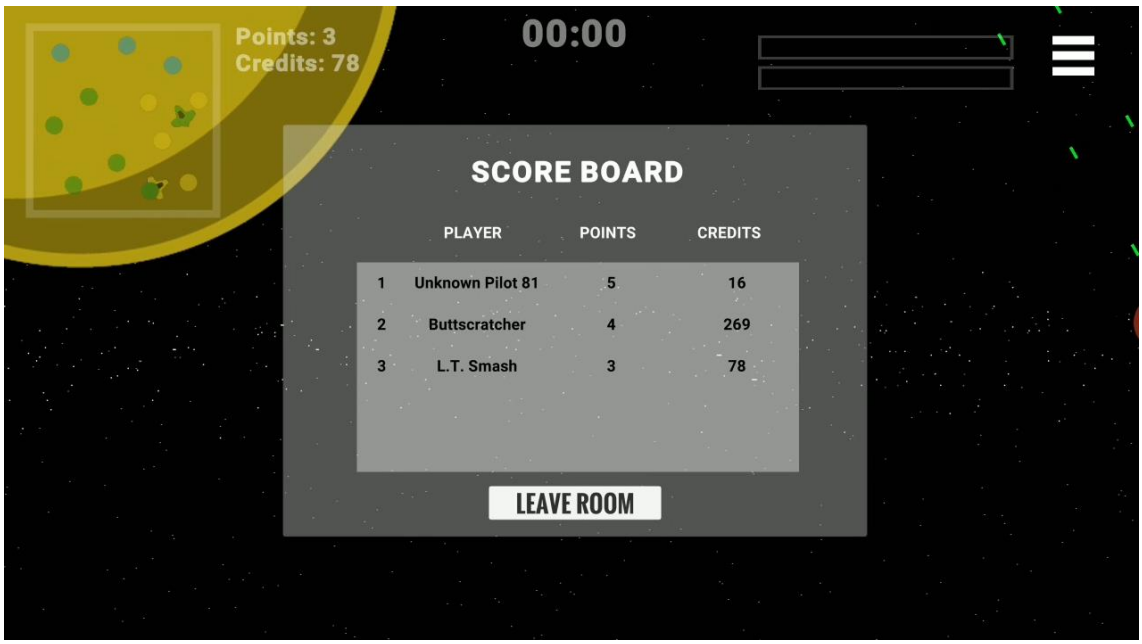
Permite recargar por completo el escudo de la nave. Si el escudo ya está al máximo de carga o el jugador no tiene suficientes créditos, este botón aparecerá deshabilitado.

6.3.10 Respawn

Cada vez que nuestra nave es destruida volveremos al punto de partida habiendo perdido la mitad de nuestros créditos y todos los misiles que tuviéramos. Esta operación no es instantánea y nos hace perder valiosos segundos. Aunque no hay un límite al número de veces que un jugador puede hacer *respawn*, debemos evitar esta situación todo lo posible.

6.3.11 Condiciones de victoria

Al terminar la partida se nos muestra el score*board* final con el ranking de jugadores, sus puntos (planetas bajo control) y sus créditos. El nombre que se muestra es el mismo que hemos introducido en *OPCIONES* del menú principal. Si un jugador no ha introducido un nombre personalizado se mostrará como *Unknown Pilot* seguido de dos dígitos aleatorios (eg. *Unknown Pilot 24*).



6.3.12 Menú in-game

Este elemento de UI aparece durante la partida en la esquina superior derecha de la pantalla.

La partida no se detiene mientras accedemos al menú in-game. Los demás jugadores aún pueden atacarnos. Este menú nos permite auto-destruir nuestra nave (puede ser útil en caso de quedarnos atascados o por cualquier otra razón). También desde este menú podemos abandonar la partida y volver al menú principal o cerrar completamente la aplicación.

Nota: si el jugador host abandona la partida se perderá la conexión con la sala y los demás jugadores no podrán continuar jugando.

7. Conclusiones

Realmente, el trabajo ahorrado al utilizar una solución externa para la gestión de la capa de red supone una increíble reducción de la complejidad del proyecto y un gran ahorro de tiempo que puede ser dedicado al desarrollo del contenido del juego. Sin embargo, el alto nivel de integración del motor y del entorno de desarrollo hace que sea difícil abstraer esta capa y liga fuertemente el proyecto al motor de *Photon*. Obviamente, si el servicio cae, el juego es incapaz de crear y listar partidas.

El *scope* original del proyecto incluía un modo de un jugador, finalmente el *scope* quedó reducido y sólo incluye el modo *multi-player*, que es el que ofrece un mayor atractivo. Teniendo en cuenta el punto explicado en el párrafo anterior, la creación de un modo *single-player* toma importancia y sería sin duda el siguiente hito en el desarrollo del juego.

Como suele ocurrir, empezar a documentar el juego en una etapa temprana podría haber desvelado aspectos que resultaron ser más complejos de lo que parecían en un principio.

El test con usuarios también debería haber iniciado en una fase más temprana del desarrollo. Algunos de los consejos y críticas por parte de los usuarios han resultado muy valiosas y, aunque algunas han podido entrar en la versión final, algunas otras se han quedado fuera.

El juego ha sido desarrollado teniendo en mente que sea un proyecto extensible. Aparte del modo para un jugador, hay infinitas mejoras que se pueden hacer al modo multijugador: *upgrades* en la nave, nuevos tipos de naves, misiles y estructuras, persistencia de los progresos y de una divisa virtual, creación de rankings, etc. La única manera de lanzar con éxito este juego al mercado y mantener el *engagement* de un número suficiente de usuarios es ofreciendo nuevos contenidos de manera constante a los jugadores.

8. Glosario

Conjunto de términos usados con frecuencia en este documento:

- **Room (sala):** una partida activa creada por uno de los jugadores.
- **Host:** creador de la partida que además es un jugador.
- **Engagement:** voluntad del usuario a querer continuar jugando el juego.
- **Feedback:** comentarios por parte de usuarios que han probado el juego.
- **Leaderboard:** tabla que muestra en orden descendente a los mejores jugadores.
- **Scoreboard:** tabla con el resultado de la partida actual.
- **Space shooter**: juego de disparos ambientado en el espacio.
- **Scope:** conjunto de objetivos que se pretende abarcar en el proyecto.
- **Network engine:** motor que se encarga de todas las comunicaciones de capa de red.
- **Single-player:** juego para un único jugador, sin comunicación con el exterior.

- **Multi-player:** pensado para permitir a varios jugadores interactuar entre ellos.
- **Ownership:** ser dueño de algo. Usado en el proyecto para referirse al dueño de un objeto compartido a través de la red.
- **Freemium:** modelo de negocio empleado por muchas grandes empresas de videojuegos, sobre todo para dispositivos móviles. Se basa en el uso de micro pagos para acelerar el progreso del jugador o darle acceso a contenidos exclusivos.
- **Deployar:** compilar y ejecutar el juego en un dispositivo.
- **Dispositivo virtualizado:** dispositivo simulado en el ordenador. Útil para hacer pruebas con varios tipos de dispositivos.
- **Beta-tester:** persona que se encarga de probar el juego antes de estar listo para su lanzamiento, detectar bugs y aportar *feedback* a los desarrolladores.
- **Back-end:** lógica ejecutada fuera del dispositivo (cliente) que proporciona un servicio.
- **Matchmaking:** se refiere a hacer accesibles las partidas creadas por los usuarios y toda la lógica para permitir a un usuario crear o unirse a una partida.
- **Sprites:** es como se conoce comúnmente a los diseños y dibujos en 2D que utiliza el proyecto.
- **Tower defense:** genero de juegos en los que hay que repeler oleadas de ataques poniendo defensas y obstáculos por el camino.
- **Power-up:** elemento del juego que proporciona una mejora al jugador. Por ejemplo: reponer munición o restaurar la salud.
- **Spawn / Respawn:** es la acción del jugador de aparecer por primera vez o reaparecer en la partida después de haber sido destruido.
- **In-game:** que ocurre durante una partida.
- **Loot:** equipamiento perdido por un jugador al ser destruido que queda abandonado y puede ser arrebatado por otro jugador.
- **Upgrade:** mejora aplicada a un elemento del juego (e.g.: mayor capacidad de carga de misiles).

9. Bibliografía y enlaces

(Base de datos de sonidos) Freesound
https://freesound.org/

(Juego) Título: Last Horizont, Desarrolladora: Pixeljam
Steam: https://store.steampowered.com/app/394220/Last_Horizon/

(Juego) Título: SLI-FI: 2D Planet Platformer, Desarrolladora: Etaxoft
Steam: https://store.steampowered.com/app/668230/SLIFI_2D_Planet_Platformer/

(Juego) Título: Endless Sky, Desarrolladora: Michael Zahniser
Steam: https://store.steampowered.com/app/404410/Endless_Sky/

(Juego) Título: Endless Sky, Desarrolladora: Michael Zahniser
Google Play:
https://play.google.com/store/apps/details?id=com.cocoon.theneonspaceships&hl=en_US

(Game Engine) Godot
https://godotengine.org/

(Juego) Título: Galaga, Desarrolladora: Namco
[Consulta: 14 de junio de 2020]
https://en.wikipedia.org/wiki/Galaga

(Juego) Título: Asteroids, Desarrolladora: Atari, Inc.
[Consulta: 14 de junio de 2020]
https://en.wikipedia.org/wiki/Asteroids_(video_game)

(Network Engine) Photon Bolt
[Consulta: 14 de junio de 2020]
https://doc.photonengine.com/en-us/bolt/current/getting-started/overview

(Network Engine) Photon Bolt
[Consulta: 14 de junio de 2020]
https://docs.unity3d.com/Manual/UNet.html

(Proyecto) Mars uprising
[Consulta: 14 de junio de 2020]
https://sites.google.com/site/marsuprising/inicio