

Desarrollo de un videojuego para AppleTV



Grado en Ingeniería Multimedia

Trabajo Fin de Grado

Autor:

Guillermo Dorado Vide

Tutor/es:

Fidel Aznar Gregori

Septiembre 2016



Universitat d'Alacant
Universidad de Alicante

Contenido

Índice de ilustraciones	6
Índice de tablas.....	9
AGRADECIMIENTOS	12
PROPUESTA Y OBJETIVOS.....	13
Propuesta	13
Objetivos.....	13
¿Qué se pretende conseguir?	13
INTRODUCCIÓN.....	15
Entretenimiento digital	15
Visualización de contenido	16
Videojuegos.....	17
Aplicaciones, el futuro	18
ANÁLISIS DEL PROYECTO.....	21
AppleTV	21
¿Por qué AppleTV?	21
¿Qué es AppleTV?	22
Historia	23
Funciones.....	25
Accesibilidad	28
AppleTV y videojuegos.....	30
Creación de un videojuego	34
Proceso de creación de un videojuego y etapas de desarrollo.....	34
Requerimientos para desarrollar un videojuego	36
Método de trabajo.....	38

Requisitos del proyecto.....	40
Tiempo.....	41
Conocimiento y aptitudes	42
Material	43
AppleTV	43
Costes	47
HERRAMIENTAS PARA EL PROYECTO	49
Control y gestión del proyecto.....	49
Trello	49
Google Drive	52
Dropbox.....	54
TrackingTime y TimeTune	55
Apartado de programación	55
Unity	55
MonoDevelop	57
Notepad++	58
Unity remote 4.....	58
Xcode.....	59
Apartado gráfico	60
Blender	60
AwesomeBump	61
EL JUEGO (Vega: Factions War)	63
Inspiración	63
Concepto y género	64
Audiencia.....	64

Características	64
Jugabilidad y mecánicas	64
Aspecto visual	65
Historia	68
Limitaciones	69
Estudio de mercado y viabilidad	70
Como se juega a Vega: Factions War.....	71
Funcionalidades y casos de uso	72
Requisitos funcionales.....	72
Requisitos no funcionales.....	85
Casos de uso	87
DESARROLLO	93
Metodología de trabajo	93
Unity3D.....	95
Funcionamiento	96
Unity remote 4.....	106
Diseño del videojuego (Vega: Factions War)	107
Conceptos de programación	107
Estructura del proyecto.....	115
Proceso de desarrollo	126
Implementación	130
Prototipos y mecánicas	132
CONCLUSIONES.....	157
Producto final.....	157

Conclusiones	162
DOCUMENTACIÓN	166
Definiciones	166
Bibliografía.....	167

Índice de ilustraciones

ILUSTRACIÓN 1 – EVOLUTION [1], IMAGEN DE GIUSEPPE MILO	15
ILUSTRACIÓN 2 – TABLETAS ELECTRÓNICAS, EL ORDENADOR DE BOLSILLO [5].....	16
ILUSTRACIÓN 3 – CONSOLAS DE NINTENDO [17]	17
ILUSTRACIÓN 4 – UNA RECREATIVA DE AKIHABARA, BY ROB SHERIDAN [18]	18
ILUSTRACIÓN 5 – APLICACIONES MÓVILES [23]	19
ILUSTRACIÓN 6 – ESCRITORIO DE GOOGLE DRIVE, APLICACIÓN WEB [26].....	20
ILUSTRACIÓN 7 – LOGO APPLETIV [31].....	21
ILUSTRACIÓN 8 - APPLETIV 4 ^a GENERACIÓN, BY RAYUKK [32]	23
ILUSTRACIÓN 9 – STREAMING DE APPLETIV CON UN TELÉFONO MÓVIL, BY SINCHEN LIN [38].....	26
ILUSTRACIÓN 10 – FOX LOGO, REFERENCIA IMAGEN DE DOMINIO PÚBLICO [49]	27
ILUSTRACIÓN 11 – TUMBLR, BY ZLATKO NAJDENOVSKI [55].....	27
ILUSTRACIÓN 12 – CONTROLADOR REMOTO SIRI, BY ANDREAS LAKSO [57].....	28
ILUSTRACIÓN 13 – BADLAND [92]	33
ILUSTRACIÓN 14 – SKETCHPARTYTV [79].....	33
ILUSTRACIÓN 15 – XENOWERKTV [82]	33
ILUSTRACIÓN 16 - BENEATH THE LIGHTHOUSE [93]	33
ILUSTRACIÓN 17 - GALAXXY ON FIRE [94].....	33
ILUSTRACIÓN 18 - LUMINO CITY [87]	33
ILUSTRACIÓN 19 - TRANSISTOR [89]	34
ILUSTRACIÓN 20 - GUITAR HERO LIVE [95]	34
ILUSTRACIÓN 21 – EQUIPO DE TRABAJO DE RESPAWN, CREADORES DE TITANFALL [101]	38
ILUSTRACIÓN 22 – ICONO DE TRELLIO [139].....	49
ILUSTRACIÓN 23 – TABLERO DE TRELLIO. TAREAS REPARTIDAS EN LISTAS.....	50
ILUSTRACIÓN 24 – TARJETA DE TRELLIO, DESCRIPCIÓN Y FUNCIONALIDADES.....	51
ILUSTRACIÓN 25 – TRELLIO. MODIFICACIÓN DE TARJETAS Y LISTAS EN CUALQUIER MOMENTO	51
ILUSTRACIÓN 26 – ESCRITORIO VIRTUAL GOOGLEDRIVE	53
ILUSTRACIÓN 27 – APLICACIONES INTEGRADAS EN GOOGLEDRIVE.....	53
ILUSTRACIÓN 28 – ESCRITORIO WEB DE DROPBOX	54
ILUSTRACIÓN 29 – LOGO DE UNITY [150]	56
ILUSTRACIÓN 30 – ALGUNAS DE LAS PLATAFORMAS A LAS QUE DA SOPORTE UNITY	56
ILUSTRACIÓN 31 – MONODEVELOP. FUNCIÓN DE AUTOCOMPLETADO DE LAS INSTRUCCIONES DE UNITY	58
ILUSTRACIÓN 32 – UNITY REMOTE 4 FUNCIONANDO CON UNITY VERSIÓN DE ESCRITORIO EN EL PROTOTIPO DE CONTROL DE MOVIMIENTO DE VEGA: FACIONS WAR.....	59
ILUSTRACIÓN 33 – ICONO DE BLENDER [165]	60

ILUSTRACIÓN 34 – VENTANA DE TRABAJO DE BLENDER. MODELANDO LA PRIMERA ITERACIÓN DE LA NAVE DEL JUGADOR	61
ILUSTRACIÓN 35 – AWESOMEBUMP4.0BETA. OBTENIENDO EL NORMALMAP DE UNA TEXTURA	62
ILUSTRACIÓN 36 – IMAGEN DE LA CAJA DEL JUEGO STARFOX64 [171].....	63
ILUSTRACIÓN 37 – ASPECTO VISUAL DE VEGA: FACTIONS WAR.....	68
ILUSTRACIÓN 38 – DIAGRAMA DE CASOS DE USO DE LA APLICACIÓN VEGA:FACTIONS WAR	88
ILUSTRACIÓN 39 – UNITY, CREACIÓN DE UN GAMEOBJECT Y COMO AÑADIR NUEVOS COMPONENTES	99
ILUSTRACIÓN 40 – UNITY, EJEMPLO DE UN GAMEOBJECT QUE SIMULA UN PUNTO DE LUZ Y SUS COMPONENTES	100
ILUSTRACIÓN 41 – INTERFAZ UNITY, VISTA GENERAL	101
ILUSTRACIÓN 42 – INTERFAZ UNITY, PROJECT WINDOW.....	101
ILUSTRACIÓN 43 – INTERFAZ UNITY, SCENE VIEW.....	102
ILUSTRACIÓN 44 – INTERFAZ UNITY, HIERARCHY E INSPECTOR	103
ILUSTRACIÓN 45 – INTERFAZ UNITY, GAME SCREEN	103
ILUSTRACIÓN 46 – UNITY, EJEMPLO DE USO DE UN PREFAB	105
ILUSTRACIÓN 47 – EJEMPLO DE PROCESO DE DESARROLLO. TRABAJANDO EN EL SISTEMA DE DAÑOS DE LOS ENEMIGOS.....	107
ILUSTRACIÓN 48 – EJEMPLO DE CÓDIGO EN UNITY, COMPONENTES COMO VARIABLES	110
ILUSTRACIÓN 49 – EJEMPLO DE CÓDIGO EN UNITY. RELACIONES ENTRE COMPONENTES DENTRO DE LA JERARQUÍA	111
ILUSTRACIÓN 50 – EJEMPLO DEL USO DE LOS COMPONENTES COMO VARIABLES. ASIGNACIÓN	112
ILUSTRACIÓN 51 – OBJETO SELECCIONADO EN EL HIERARCHY DE UNITY	113
ILUSTRACIÓN 52 – VISUALIZACIÓN DE LA RELACIÓN ENTRE OBJETOS PADRE E HIJOS EN EL HIERARCHY DE UNITY	113
ILUSTRACIÓN 53 – EJEMPLO DE ASIGNACIÓN DE UN OBJETO CÓMO VARIABLE MEDIANTE LA INTERFAZ DE UNITY	113
ILUSTRACIÓN 54 – DIAGRAMA DE FLUJO Y DISTRIBUCIÓN GENERAL DE LA APLICACIÓN VEGA: FACTIONS WAR	115
ILUSTRACIÓN 55 – ESTRUCTURA DEL JUEGO EN UNITY, PARTES QUE LO COMPONEN	117
ILUSTRACIÓN 56 – ESQUEMA DE LA ENTIDAD TPLAYER	119
ILUSTRACIÓN 57 - ESQUEMA DE LA ENTIDAD PLAYER.....	120
ILUSTRACIÓN 58 - COMPONENTES PRINCIPALES DE LA ENTIDAD PLAYER	120
ILUSTRACIÓN 59 – COMPONENTES SCRIPTS PRINCIPALES DE PLAYER.....	121
ILUSTRACIÓN 60 – COMPONENTES QUE FORMAN LA ENTIDAD PLAYERMODEL.....	121
ILUSTRACIÓN 61 – ESQUEMA DE LA ENTIDAD ENEMIGO	122
ILUSTRACIÓN 62 – COMPONENTES PRINCIPALES DE LA ENTIDAD ENEMYMODEL.....	123
ILUSTRACIÓN 63 – COMPONENTES PRINCIPALES DE LAS ENTIDADES QUE CONTROLAN EL SISTEMA DE APUNTADO	124
ILUSTRACIÓN 64 - COMPONENTES DE LA ENTIDAD PATHMANAGER	125

ILUSTRACIÓN 65 - COMPONENTES DE LA ENTIDAD WAYPOINT	125
ILUSTRACIÓN 66- DISEÑO CONCEPTUAL DEL PRIMER NIVEL DEL JUEGO.....	127
ILUSTRACIÓN 67 – DISEÑO CONCEPTUAL DE NAVES.....	127
ILUSTRACIÓN 68 – DISEÑO CONCEPTUAL DE LA INTERFAZ DEL JUEGO	128
ILUSTRACIÓN 69 – DISEÑO CONCEPTUAL DE NAVES.....	128
ILUSTRACIÓN 70 – DISEÑO DE LOS MODELOS DEL JUEGO	129
ILUSTRACIÓN 71 – DISEÑO CONCEPTUAL DE LOS ENEMIGOS TERRESTRES DEL JUEGO.....	129
ILUSTRACIÓN 72 – SISTEMAS DE ESCUDOS DE LA NAVE DEL JUGADOR.....	130
ILUSTRACIÓN 73 – PROTOTIPO INICIAL DE LA IDEA DEL VIDEOJUEGO	133
ILUSTRACIÓN 74 – PROTOTIPO INICAL DE MOVIMIENTO DEL JUGADOR	134
ILUSTRACIÓN 75 – SCRIPT DE FUNCIONAMIENTO BÁSICO DE LA CÁMARA	134
ILUSTRACIÓN 76 – SISTEMA DE APUNTADO Y DE DISPARO	135
ILUSTRACIÓN 77 - CÓDIGO QUE CALCULA LA POSICIÓN DE COLISIÓN DEL RAYCAST	136
ILUSTRACIÓN 78- SCRIPT QUE APUNTA LOS CAÑONES DE LA NAVE AL PUNTO DE DISPARO OBTENIDO	136
ILUSTRACIÓN 79 – CONJUNTO DE LOS PRIMEROS ENEMIGOS DISEÑADOS	137
ILUSTRACIÓN 80 – BUCLE PRINCIPAL DEL SISTEMA DE PUNTOS DE VIDA DEL JUGADOR	138
ILUSTRACIÓN 81 –EXPLOSIÓN AL SER DESTRUÍDO UN ENEMIGO CON EL SISTEMA DE VIDAS YA IMPLEMENTADO	138
ILUSTRACIÓN 82 – CAPTURA DE COMO SE MUESTRA EL SCRIPT DE ATAQUE DE LOS ENEMIGOS.....	140
ILUSTRACIÓN 83 – PARTE DEL CÓDIGO QUE DETECTA LA COLISIÓN DE LOS ATAQUES ENEMIGOS EN EL JUGADOR	140
ILUSTRACIÓN 84 – FUNCIONES PRINCIPALES QUE CONTROLAN LOS ESCUDOS DEL JUGADOR.....	141
ILUSTRACIÓN 85 – COMPONENTE SCRIPT DE LA NUEVA FORMA DE MOVIMIENTO LINEAL.....	141
ILUSTRACIÓN 86 – CÓDIGO QUE PERMITE DESPLAZAR LINEALMENTE UN OBJETO ENTRE DOS PUNTOS	142
ILUSTRACIÓN 87 – COMPONENTE SCRIPT DE LA NUEVA FORMA DE MOVIMIENTO “HOVER”, MÁS PARAMETRIZABLE	142
ILUSTRACIÓN 88 – CÓDIGO DE UNA DE LAS VARIANTES DE MOVIMIENTO DEL NUEVO TIPO DE MOVIMIENTO “HOVER”	142
ILUSTRACIÓN 89 – PARTE DEL CÓDIGO PARA CALCULAR LOS PUNTOS INTERMEDIOS DE UNA CURVA	144
ILUSTRACIÓN 90– VISUALIZACIÓN DEL SISTEMA DE PUNTOS DE RUTA. CAMINO ENTRE DOS WAYPOINTS	144
ILUSTRACIÓN 91 – PARTE DEL CÓDIGO DEL CONTROLADOR DE UN NIVEL. CÁLCULO DE WAYPOINTS	145
ILUSTRACIÓN 92 – PARTE DEL CÓDIGO DEL CONTROLADOR DE UN NIVEL. FUNCIONES PRINCIPALES.	145

ILUSTRACIÓN 93 – PRUEBAS REALIZADAS PARA ADAPTAR EL CONTROL DE MOVIMIENTO MEDIANTE UNITY REMOTE4.....	146
ILUSTRACIÓN 94 – MODELO DE UN NUEVO ENEMIGO, SAETA	147
ILUSTRACIÓN 95 – MODELO DE UN NUEVO ENEMIGO, CAZA	147
ILUSTRACIÓN 96 – CREACIÓN DE LA PRIMERA ZONA DEL PRIMER NIVEL.....	148
ILUSTRACIÓN 97 – REFERENCIA INICIAL DEL ESTILO VISUAL Y CARGA DE PRIMEROS MODELOS.	148
ILUSTRACIÓN 98 - NUEVOS TIPOS ENEMIGOS. EL CAZA Y LA CORBETA.	149
ILUSTRACIÓN 99 – NUEVO CÓDIGO Y COMPONENTES PARA LOS ENEMIGOS.....	149
ILUSTRACIÓN 100 – NUEVO ENEMIGO, SAETA	150
ILUSTRACIÓN 101 – NUEVA FUNCIONALIDAD A LAS TORRETAZOS, CAPACIDAD DE DESPLIEGUE.....	150
ILUSTRACIÓN 102 – CÓDIGO DE FUNCIONAMIENTO DEL DESPLIEGUE DE LAS TORRETAZOS.....	151
ILUSTRACIÓN 103 – COMPONENTE SCRIPT NUEVO PARA EL DESPLIEGUE DE LAS TORRETAZOS	151
ILUSTRACIÓN 104 – CÓDIGO PARA EL PROTOTIPO ESPECÍFICO.	152
ILUSTRACIÓN 105 – ADAPTACIÓN DEL SISTEMA DE APUNTADO AL CONTROL TÁCTIL DEL MANDO SIRI	152
ILUSTRACIÓN 106 – TRABAJO DE CÁLCULO DE LOS UV Y TEXTURIZADO DEL PRIMER NIVEL.....	153
ILUSTRACIÓN 107 – MEJORAS EN EL ASPECTO VISUAL. MATERIALES Y TEXTURAS.....	154
ILUSTRACIÓN 108 – CÁLCULOS DE LAS OCLUSIONES DEL PRIMER NIVEL	154
ILUSTRACIÓN 109 – CÓDIGO QUE ACTIVA EN LA ESCENA A LOS ENEMIGOS QUE ESTÁN CERCA DEL JUGADOR	155
ILUSTRACIÓN 110 – CÓDIGO QUE PERMITE CARGAR LA PUNTUACIÓN Y POSICIÓN DE UN CHECKPOINT DE UN NIVEL.....	155
ILUSTRACIÓN 111 – CREACIÓN DE LA INTERFAZ DE LA PANTALLA DE SELECCIÓN DE NIVELES	155
ILUSTRACIÓN 112 – TÍTULO DEL VIDEOJUEGO CREADO, VEGA: FACTIONS WAR.....	156
ILUSTRACIÓN 113- CAZA	157
ILUSTRACIÓN 114- CORBETA.....	157
ILUSTRACIÓN 115 - FRAGATA	158
ILUSTRACIÓN 116 – EXOARMADURA DE COMBATE	158
ILUSTRACIÓN 117 – MÍNA DE PROXIMIDAD.....	158
ILUSTRACIÓN 118 - SAETA	158
ILUSTRACIÓN 119 – TANQUE AERODESLIZADOR	158
ILUSTRACIÓN 120 – TORRETA DEPLEGABLE.....	159
ILUSTRACIÓN 121 – CRUCEROS DE BATALLA	159
ILUSTRACIÓN 122 – JEFES FINALES	159
ILUSTRACIÓN 123 – PRIMER NIVEL DEL JUEGO, HUIDA	160
ILUSTRACIÓN 124 – SEGUNDO NIVEL DEL JUEGO, ATAQUE CALCULADO	160
ILUSTRACIÓN 125 – TERCER NIVEL DEL JUEGO, GUERRA DE FACCIONES	160
ILUSTRACIÓN 126 – SEGUNDO NIVEL DEL VIDEOJUEGO, ENTRANDO EN LA BASE DE LOS ENEMIGOS	161

Índice de tablas

TABLA 1 COSTES DEL PROYECTO, TIEMPO EN CRÉDITOS	48
TABLA 2 COSTES DEL PROYECTO, TIEMPO DE DESARROLLO REAL	48
TABLA 3 RF1	73
TABLA 4 RF2	73
TABLA 5 RF3	73
TABLA 6 RF4	74
TABLA 7 RF5	74
TABLA 8 RF6	74
TABLA 9 RF7	75
TABLA 10 RF8	75
TABLA 11 RF9	75
TABLA 12 RF10	76
TABLA 13 RF11	76
TABLA 14 RF12	77
TABLA 15 RF13	77
TABLA 16 RF14	77
TABLA 17 RF15	78
TABLA 18 RF16	78
TABLA 19 RF17	79
TABLA 20 RF18	79
TABLA 21 RF19	79
TABLA 22 RF20	80
TABLA 23 RF21	80
TABLA 24 RF22	80
TABLA 25 RF23	81
TABLA 26 RF24	81
TABLA 27 RF25	82
TABLA 28 RF26	82
TABLA 29 RF27	82
TABLA 30 RF28	83
TABLA 31 RF29	83
TABLA 32 RF30	83
TABLA 33 RF31	84
TABLA 34 RF32	84
TABLA 35 RF33	84
TABLA 36 RF34	85

TABLA 37 RNF35	85
TABLA 38 RNF36	86
TABLA 39 RNF37	86
TABLA 40 RNF38	86
TABLA 41 RNF39	87
TABLA 42 CASO DE USO, EJECUTAR LA APLICACIÓN	89
TABLA 43 CASO DE USO, NAVEGAR POR LOS MENÚS	89
TABLA 44 CASO DE USO, MODIFICAR LAS OPCIONES	90
TABLA 45 CASO DE USO, JUGAR UN NIVEL	91
TABLA 46 CASO DE USO, CONTROLAR LA NAVE	91

AGRADECIMIENTOS

Han sido años de trabajo y superación que me han aportado multitud experiencias. Pero durante este recorrido uno nunca se encuentra solo, y esta sección está dedicada a todos los que han ofrecido su apoyo.

Primero, a mi tutor y a todo el profesorado de Ingeniería Multimedia, por la paciencia y los conocimientos que han impartido. Además, quiero destacar su esfuerzo y dedicación puestos en mejorar la carrera cada año. Gracias.

También, dar las gracias al resto de alumnos y compañeros de la carrera, por compartir las penas y glorias y, sobre todo, por el buen ambiente que se ha creado gracias a ellos.

Finalmente, agradecer el gran apoyo de las personas más cercanas y queridas. A mi hermano, que siempre se interesaba por mis estudios y ofrecía su consejo y ayuda en cualquier trabajo. A mis padres que han soportado mis quejas y me animaban para que siguiera adelante. Y, finalmente, a mi pareja, que sin su apoyo psicológico y su sonrisa no hubiese sido lo mismo.

¡A todos los mencionados, de nuevo, Gracias!

PROUESTA Y OBJETIVOS

Propuesta

Desarrollar de manera individual un videojuego completo con todos sus elementos, desde su fase conceptual hasta su forma final, en un corto periodo de tiempo. Se desarrollará para la plataforma AppleTV, que dispone de un sistema de control propio llamado Siri, usando el motor de desarrollo Unity.

Objetivos

A continuación, se desarrollará y se pondrá en contexto la propuesta y la ideología del proyecto, y cuáles son las metas que se quieren obtener.

¿Qué se pretende conseguir?

- 1- Desarrollar un videojuego
 - a. Organizar y planificar todo el proceso de creación de un videojuego, desde el concepto, diseño y estructura hasta la lógica de programación y acabado visual.
 - b. Crear el videojuego planificado del punto anterior, poniendo especial atención en conseguir que tenga una estructura interna bien construida y que funcione de manera óptima.
 - c. Crear dicho videojuego involucrándose en todas las facetas del desarrollo de videojuegos haciendo uso de la multidisciplinariedad de los Ingenieros Multimedia. Participando en la programación, diseño, modelado, diseño de niveles, etc.
 - d. Desarrollar el videojuego teniendo en mente la plataforma para la que se desea publicar.

- e. Crear dicho videojuego usando la herramienta de desarrollo de videojuegos Unity.
- f. Aprender a utilizar Unity y todas las funcionalidades que ofrece en el campo de creación de videojuegos.

2- Adaptarlo a AppleTV

- a. Realizar un estudio y análisis sobre el dispositivo AppleTV y sobre lo que éste puede ofrecer.
- b. Analizar e investigar el mercado de los videojuegos desarrollados para AppleTV desde su lanzamiento.
- c. Tener en cuenta los requisitos y límites de AppleTV a la hora de desarrollar el videojuego mencionado en el punto 1.
- d. Adaptar el control del videojuego al mando remoto Siri y la arquitectura de AppleTV.
- e. Usar Unity y sus distintas herramientas para configurar el uso del mando remoto Siri en el videojuego.
- f. Obtener mediante Unity la versión de compilación necesaria para crear el videojuego en dispositivos iOS.
- g. Crear la aplicación del videojuego para ser usada en un dispositivo AppleTV.

INTRODUCCIÓN

Este apartado tratará sobre el entretenimiento digital y su evolución con el paso del tiempo. Además, se explicará que es una aplicación, se hablará sobre sus ventajas y sobre cómo representan el futuro.

Entretenimiento digital

Ya que este trabajo final de grado tiene que ver con el ocio digital, en concreto sobre la creación de un videojuego, vamos a hablar brevemente sobre qué es y cómo ha evolucionado el entretenimiento digital.



Ilustración 1 – Evolution [1], imagen de Giuseppe Milo

Con el paso del tiempo, los medios, las herramientas y la forma de entretenimiento van cambiando [2] [3]. En pleno apogeo de la era electrónica, las nuevas tecnologías y los nuevos medios multimedia [4]¹² forman parte del

¹ Multimedia, aparato o software capaz de utilizar varias formas o medios, tanto físicos como digitales, para reproducir o comunicar información. [4]

² Multimedia en la electrónica, se refiere a cualquier dispositivo capaz de almacenar y mostrar contenido multimedia. [4]

día a día de la sociedad, cambiando la manera que tiene de comunicarse y relacionarse, así como su propio comportamiento.



Ilustración 2 – Tabletas electrónicas, el ordenador de bolsillo³ [5]

El mayor cambio actual proviene del fácil acceso de la sociedad a dispositivos electrónicos como teléfonos móviles, portátiles, tabletas electrónicas, gafas y relojes inteligentes, etc. todos ellos conectados entre sí mediante la red de redes, internet. Con estos nuevos dispositivos circulando, siempre conectados a la red, y sus casi infinitas posibilidades de desarrollo, se abren nuevos mercados en todos los sectores, incluyendo al sector del entretenimiento que está creciendo en cantidad y variedad [6] [7] [8] [9] [10].

Visualización de contenido

Uno de los sectores que más crece y que más cambios está experimentando es la visualización de contenido multimedia. En la actualidad se pueden ver series y películas, mediante distintas plataformas o aplicaciones como Youtube [11] [12] o Netflix [13] [14] [15] [16]. Gracias a internet y a las

³ Creative Commons Zero (CC0) license.

nuevas tecnologías se puede tener acceso a ellas en casi cualquier dispositivo, desde televisiones y tabletas hasta móviles.

Videojuegos

El otro medio que está creciendo exponencialmente es el sector de los videojuegos. Empezó siendo una forma de entretenimiento al que solo unos pocos podían acceder, ya fuera por cuestiones de ingresos o por el estigma social negativo que se asociaba con el sector y con sus usuarios. Con el paso del tiempo, ha ido cambiando el concepto y la percepción de los videojuegos hasta alcanzar el estado actual, siendo mejor aceptado por la sociedad (siempre y cuando se haga un consumo responsable de los mismos). Cualquiera puede jugar a un videojuego u obtener un dispositivo que tenga acceso a ellos.



Ilustración 3 – Consolas de Nintendo [17]⁴

Antiguamente, en sus inicios, para poder jugar a videojuegos se tenía que ir a locales especializados, como las recreativas o salones de ocio. Con el tiempo se pudo, si se disponía del suficiente dinero, comprar un ordenador o consola de sobremesa para jugar en el hogar. En la actualidad se juegan a videojuegos en cualquier lugar mediante el móvil, ordenadores, tabletas digitales, televisión, consolas y, realmente, en cualquier dispositivo que tenga

⁴ Creative Commons Attribution-Share Alike 3.0 Unported

acceso a internet o que sea capaz de soportar el software del mismo. No hay limitaciones al lugar o herramienta.



Ilustración 4 – Una recreativa de Akihabara, by Rob Sheridan [18]⁵

Destacar que, no solo ha evolucionado la cantidad y propagación de los videojuegos, sino también ha aumentado la variedad de ofertas y géneros [19] [20]. Existen juegos de aventura, acción, plataformas, narrativos, puzzles, e incluso videojuegos educativos y promocionales. El sector está en auge, y su futuro es prometedor.

Aplicaciones, el futuro

El concepto de aplicación [21] [22] se da a conocer a la sociedad con el desarrollo de la tecnología móvil y de las tabletas electrónicas, pero en realidad, ¿Qué es una aplicación?

⁵ Attribution-NonCommercial-ShareAlike 2.0 Generic



Ilustración 5 – Aplicaciones móviles⁶ [23]

Una aplicación es cualquier software [24] [25] desarrollado con la finalidad de cumplir con ciertas tareas y funcionalidades. Es decir, cualquier programa de un ordenador se puede considerar una aplicación software⁷. Por ejemplo, el famoso Microsoft Word, los reproductores de música o el propio navegador web cuentan como aplicaciones. Lo que hay que tener en cuenta es que todos los anteriores son aplicaciones software, mientras que las conocidas como App⁸, son aplicaciones específicas para móvil o web que han evolucionado en funcionamiento y forma como contraparte de su formato para escritorio⁹.

¿Qué aporta una aplicación?

En la actualidad las aplicaciones dominan el mundo electrónico y de la web. Todas las empresas desarrollan sus propias aplicaciones para sus productos o servicios y parece que toda idea o concepto tiene que tener su versión en App. Esto tiene una razón de ser: los beneficios y ventajas que aportan las aplicaciones. Una App supone un acceso rápido y normalmente específico a una aplicación software encargada de realizar una serie de funciones. Al poderse usar desde un dispositivo móvil o desde la página web,

⁶ Creative Commons Attribution-Share Alike 4.0 International

⁷ Aplicación software [25], Puede estar vinculada a un ordenador y su sistema operativo o puede ser publicada de manera independiente, o como open-source [217] [164]. Las aplicaciones desarrolladas para móviles se denominan aplicaciones móviles y las propias de un navegador aplicaciones web.

⁸ App [22]: Término abreviado para definir a las aplicaciones móviles

⁹ Una aplicación para escritorio es la versión conocida específica para ordenadores. Por ejemplo, Skype cuenta con su versión de escritorio y Twitter con su versión móvil.

además de su facilidad de manejo debido al tipo de dispositivo para el que están desarrolladas, hacen que trabajar con ellas sea más sencillo.

El futuro es de las aplicaciones

Como se ha mencionado antes, las App son la evolución directa de las aplicaciones software. Esto quiere decir que no solo son una versión de un programa desarrollado para poder usarse en dispositivos con características o carencias diferentes a las de un ordenador de sobremesa, sino que el propio concepto y funcionalidades del desarrollo de aplicaciones móviles han hecho que las aplicaciones software cambien.

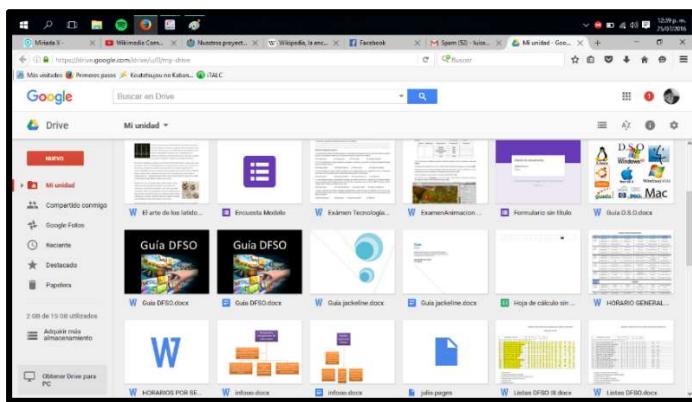


Ilustración 6 – Escritorio de Google Drive, aplicación web¹⁰ [26]

Tanto en cómo se gestionan a cómo se utilizan, las aplicaciones han evolucionado y van a seguir evolucionando, hasta el punto en el que no será necesario instalarlas en los dispositivos, sino que se podrá acceder a ellas directamente desde internet. Esto último es parecido a lo que Google está haciendo con su navegador Google Chrome y sus App nativas.

Para leer posibles evoluciones y usos sobre las aplicaciones se pueden consultar las siguientes referencias [27] [28] [29] [30].

¹⁰ Creative Commons Attribution-Share Alike 4.0 International

ANÁLISIS DEL PROYECTO

En esta sección se va a realizar un análisis y estudio sobre el dispositivo AppleTV y sobre el proceso de la creación de un videojuego.

AppleTV

Este apartado está dedicado a explicar AppleTV y por qué se ha decidido crear un videojuego para este dispositivo.



Ilustración 7 – Logo AppleTV¹¹ [31]

¿Por qué AppleTV?

AppleTV [32] [33] es una plataforma que a nivel de desarrollo de videojuegos aún no está muy explorada y, por tanto, puede ofrecer muchas posibilidades a cualquier proyecto nuevo. Al ser un mercado pequeño un producto con un buen acabado y que sea divertido y sencillo puede ser un éxito. Además, desarrollar para AppleTV supone un reto y permitirá poner a prueba los conocimientos y habilidades conseguidas a lo largo de la carrera. Adicionalmente, al usar una plataforma con la que no se está acostumbrado a trabajar, como es el caso del sistema operativo de Apple, y que además cuenta con su propia arquitectura y sus referencias de control de calidad, añaden más valor a la experiencia que se puede obtener.

¹¹ Referencia imagen de dominio público.

¿Qué es AppleTV?

Ya hemos comentado por qué hemos decidido usar AppleTV, pero ahora hay que responder a la pregunta, ¿Qué es AppleTV?

Las apps son el futuro de la electrónica, y Apple ha pensado que también es el futuro de la televisión. Esa es la razón por la que la compañía creó AppleTV: un receptor digital multimedia que se encarga de conectar todo tipo de aplicaciones y herramientas digitales entre sí y con un televisor. El dispositivo puede reproducir todo tipo de contenido multimedia y usar las plataformas YouTube, Flickr, Vimeo, Netflix y, como no, el iTunes Store.

Básicamente, es un sistema operativo o plataforma centrada en el uso de aplicaciones, desarrollada para usarse directamente con la pantalla de la televisión. Se podría decir que es un ordenador preparado específicamente para usarse con la televisión, aportando facilidad de manejo y simplificando el resto de elementos del uso cotidiano de un ordenado. Hay que destacar que AppleTV es HDMI compatible¹², y para conectarse a un televisor necesita hacerlo a través de un cable HDMI.

Además, también ofrece la posibilidad de conectarse y reproducir contenido de un ordenador, ya sea con uno que tenga el sistema operativo Mac OS X o un ordenador que tenga acceso a iTunes¹³.

¹² HDMI [244] es una interfaz de transferencia de datos, que cuenta con su propia tecnología de transmisión, que permite visualizar contenido en alta definición

¹³ Aunque sea de otro sistema operativo como Windows mientras pueda acceder a iTunes puede funcionar.



Ilustración 8 - AppleTV 4^a generación, by Rayukk¹⁴ [32]

El aparato en sí es muy sencillo y no cuenta con botones o controles propios.

Por tanto, para utilizarlo se necesita un controlador externo, ya sea empleando un mando remoto diseñado por Apple para AppleTV¹⁵, vinculando otro mando mediante conexión por infrared o bluetooth¹⁶ o con la aplicación iTunes Remote¹⁷ que se controla mediante la función Wi-Fi del dispositivo.

Además, al contar con conexión Wi-Fi puede recibir contenido directamente desde iTunes Store o AirPlay.

En las referencias [34] [35] se pueden acceder a un enlace con todas las especificaciones y características de AppleTV. Para consultar la documentación en castellano véase [36].

Historia

Para entender AppleTV en la actualidad, primero hay que revisar su historia y evolución con el paso del tiempo.

¹⁴ Creative Commons Attribution-Share Alike 4.0 International.

¹⁵ El mando “Siri” de la 4th generación de AppleTV cumple esta función.

¹⁶ Infrared [65] y Bluetooth [243] sirven para transmitir datos en cortas distancias.

¹⁷ La aplicación se puede descargar desde cualquier producto de Apple mediante de la App Store.

Primera generación

En septiembre de 2006 se anunció como una propuesta en desarrollo el “iTВ”, el prototipo del primer AppleTV. Usaba un AppleRemote y necesitaba un ordenador con iTunes para ofrecer todos sus servicios.

En 2007 se lanzó la primera versión con una capacidad de 40GB, revisada poco después con una versión de 160GB y capacidad HDD.

En enero de 2008 se introdujo una actualización gratuita que independiza el dispositivo de estar conectado a un ordenador mediante iTunes. Esta actualización permitía adquirir o alquilar directamente de la tienda de iTunes y acceder al servicio de streaming de fotos de MobileMe y Flickr.

A lo largo de 2008 se siguieron sacando actualizaciones que hicieron compatible el dispositivo con iPhone y el iPodTouch.

Segunda generación

En septiembre de 2010 empieza la segunda generación de AppleTV. El dispositivo cuenta con su propia variante del sistema operativo iOS y, además, se reduce su tamaño y se cambia la forma de su carcasa.

Este nuevo modelo cuenta con un disco duro interno y 8GB de almacenamiento flash interno. Se potenció la capacidad de realizar streaming del dispositivo, permitiendo conexiones mediante iTunes entre distintos ordenadores iOS y dispositivos mediante AirPlay.

Tercera generación

En marzo de 2012 se lanzó la tercera generación. En aspecto era idéntica a la versión anterior, pero con un procesador mejorado y podía soportar y visualizar contenido a 1080p, full HD.

En 2015 Apple declara obsoleta las primeras generaciones de AppleTV.

Cuarta generación

En septiembre de 2015 Apple anuncia la cuarta generación de AppleTV, que se lanzará al mercado en octubre de ese mismo año. Vuelve a ser una revisión con mejora sustancial del dispositivo. En esta generación es cuando Apple utiliza su famoso eslogan *“El futuro de la televisión son las aplicaciones.”*

El aparato es ligeramente más alto que la versión anterior, pero esta vez cuenta con un nuevo mando táctil remoto, denominado “Siri”, que mejoraba su control notablemente. Además, esta cuarta generación parte con un nuevo sistema operativo llamado tvOS vinculado a la App Store, que permite la descarga de aplicaciones de terceros, ya sean de reproducción de videos, videojuegos u otros tipos de contenido.

Aunque en sus inicios contaba con poco contenido, la cuarta generación ha ido creciendo poco a poco con la salida de APIs de desarrollo específicas para el dispositivo. Como curiosidad, comentar que como competidores directos con su dispositivo FireTV, Amazon.com dejó de vender AppleTV en su sección dedicada a productos Apple.

Funciones

AppleTV cuenta con muchas funcionalidades que la hacen muy apetecible al consumidor [37], a continuación, hablaremos de las más importantes.



Ilustración 9 – Streaming de AppleTV con un teléfono móvil, by Sinchen.Lin ¹⁸ [38]

Sincronización y streaming

AppleTV puede realizar streaming¹⁹ de cualquier contenido multimedia que reciba a una pantalla u ordenador compatibles. Además, también puede sincronizarse con cualquier ordenador que permita ejecutar iTunes, con lo que no solo se limita a dispositivos iOS, sino que es posible compartir el contenido con distintas plataformas. Esto permite, entre otras cosas, usar AppleTV como receptor y reproductor de otro dispositivo dónde se almacena el contenido.

La función de streaming se puede usar mediante la descarga de la app dedicada Livestream [39] desde la App Store, o mediante el uso de Apple Airplay [40].

AirPlay [41] [42] [43] es un protocolo stack/suite²⁰ desarrollado por Apple que permite el intercambio de contenidos entre dispositivos iOS y un

¹⁸ Attribution 2.0 Generic.

¹⁹ Es el término que se utiliza al hacer referencia a cualquier contenido multimedia que está siendo enviado y recibido continuamente casi al mismo tiempo [209]. El emisor está grabando y emitiendo el contenido al mismo tiempo, y el receptor está descargando el contenido en tiempo real.

²⁰ Suite es la propia definición de los protocolos y el stack es la implementación software de esos protocolos. Suite es más conocido como los protocolos network [210] TCP/IP. Para saber que son los protocolos stack/suite, más información en los siguientes enlaces enlaces. Stack [219] Suite [220]

televisor o reproductor de música de manera inalámbrica²¹, usando una red Wi-Fi [44]. Para su uso el emisor tiene que ser un dispositivo iOS o con acceso a iTunes. Si se quiere usar con un televisor es necesario tener un AppleTV conectado a la pantalla para poder hacer funcionar el servicio.

Aplicaciones

El gran reclamo de venta de AppleTV son las aplicaciones. Las aplicaciones a las que puede acceder incluyen las que se encuentran accesibles para todos los dispositivos móviles o tablets, como por ejemplo Youtube, Tumblr [45] [46], visualizadores de galaxias, HBO [47] [48], Fox [49] [50] o incluso programas para hacer ejercicio [51].

Algunos consejos y formas de uso de AppleTV se pueden encontrar en las referencias [52] [53] [54].



Ilustración 10 – Fox logo, referencia imagen de dominio público [49]



Ilustración 11 – Tumblr, By Zlatko Najdenovski [55]²²

²¹ Transferencia de datos o energía entre dos puntos sin necesidad de conexión por cable o directa [218]. Wi-Fi [44] es la tecnología que permite la conexión a Internet de manera inalámbrica. En el caso de AirPlay, el dispositivo emisor y el receptor hacen de puertos para establecer la conexión de datos.

²² Creative Commons (Attribution 3.0 Unported)

Accesibilidad

Siri

Es el controlador remoto que te viene de serie con la 4º Generación de AppleTV. Siri ha sido diseñado específicamente para ser usado con el AppleTV, con el objetivo de mejorar la interacción del usuario con la plataforma. Su tamaño no pasa de los 12,4 cm altura por 3,8 cm de anchura, con un peso de alrededor de los 47 gramos [56].



Ilustración 12 – controlador remoto Siri, by Andreas Lakso²³ [57]

Con estas características, Apple ha pretendido poner a disposición de los usuarios un mando muy sencillo y ligero de usar, con el que acceder a todas las funciones de la plataforma. Tiene el número de botones suficientes para poder actuar como un mando, pero con las funcionalidades añadidas de tener un controlador táctil incorporado con control multidireccional, así como control por movimiento al contar con acelerómetro y giroscopio.

Una de las funcionalidades banderas de Siri es el uso del micrófono incorporado para interactuar con la plataforma y dar órdenes sin hacer uso de los controles del mando. Algunas de las opciones que permite realizar comprenden desde búsqueda de contenido mediante voz hasta interactuar con

²³ Creative Commons Attribution-Share Alike 4.0 International.

el contenido que se está visualizando, como activar o desactivar subtítulos, avanzar la reproducción u obtener información del contenido. Ejemplos de frases de interacción con Siri:

- "Avanza dos minutos."
- "¿Quién protagoniza esta película?"
- "Reproduce éxitos de los años 90".
- "Busca videos divertidos de gatos en YouTube".
- "Abre el App Store".

El mando de Siri aún sigue actualizando sus funcionalidades, para ver más ejemplos de uso consultar [58] [59] [60] [61] [62].

Es conveniente destacar que AppleTV solo puede tener vinculado un mando Siri, con lo utilizar varios mandos o el juego multijugador no son posibles con Siri²⁴.

Aunque como mando para controlar el AppleTV es muy eficiente y ofrece variedad de uso, para el desarrollo de videojuegos Siri [63] tiene carencias que suponen un problema o dificultad añadida. El número de botones del mando simplemente no es suficiente, obligando a ser creativo con el método elegido para interactuar con el videojuego [64]. Así que, algunos tipos de juegos, sobre todo los que requieran de un control más complejo o permitan realizar varias acciones a la vez, son más difíciles de crear o adaptar a la plataforma AppleTV.

²⁴ Para jugar con varios jugadores sería necesario usar otro controlador iOS o cualquier controlador compatible con la plataforma [66].

Otros controladores

Como ya se ha comentado con anterioridad, se puede conectar a AppleTV otros controladores remotos de terceros mediante infrarrojos [65], así que se pueden usar también mandos para la televisión e incluso reproductores DVD como sustitutos. Para conectarlos se tiene que activar una opción en la configuración del dispositivo para detectar controladores remotos. En la página de soporte de Apple [66] están explicados los pasos detallados para lograrlo.

AppleTV y videojuegos

Tras la salida de la cuarta generación en 2015, una de las características que se destacan de AppleTV es su capacidad para funcionar también como “consola” [67]. Es cierto que tiene cierto potencial como plataforma de videojuegos, pero no está exenta de inconvenientes y todavía es temprano para realizar un veredicto final.

Los puntos negativos de la plataforma son las limitaciones de diseño de Apple con respecto a AppleTV [68] [69]. Las más destacadas están relacionadas con el control e interacción del usuario con el juego y con el propio límite de tamaño de las aplicaciones.

Es obligatorio que los videojuegos funcionen usando únicamente el controlador Siri como mando principal y con un requisito de tamaño máximo de la aplicación de 200mb, aunque la restricción de tamaño no es tan importante²⁵.

Ambas limitaciones suponen un problema a la hora de crear un producto grande y ambicioso, que se salga de los cánones de lo que se puede encontrar

²⁵ La limitación de tamaño de la aplicación se puede solventar, ya que existen métodos y recursos para poder subir juegos de mayor tamaño. Se pueden usar métodos de On-Demand Resources [113] [114] para tener dividida la aplicación en pequeñas partes que el dispositivo va cargando o eliminando según las necesite.

en videojuegos para móviles, como los que se pueden ver en las videoconsolas y ordenadores actuales.

Estado actual

A pesar de que ya cuenta con numerosos títulos a disposición de los usuarios [70] [71], la gran mayoría de ellos, al menos los más destacados, han sido adaptados a AppleTV desde otros sistemas iOS.

Los juegos que han sido adaptados o “videojuegos *port*²⁶” no han sido diseñados teniendo en mente inicialmente el concepto que ofrece AppleTV como plataforma de videojuegos. Por un lado, son productos acabados de gran calidad, pero, por otro, ofrecen una experiencia a medias, tanto de impresión final como de jugabilidad y control. Desde el punto de vista de la plataforma realmente no ofrecen el potencial de un juego dedicado desde su diseño al dispositivo. Mediante estos “videojuegos port”, las compañías desarrolladoras aprovechan el nuevo mercado para volver a sacar su producto y de paso probar la viabilidad de la nueva plataforma. Es bueno que se consigan este tipo de videojuegos en AppleTV, pero no que su futuro como plataforma de videojuegos dependa exclusivamente de ellos.

Afortunadamente, también hay varios videojuegos (entre los que se puede incluir el proyecto desarrollado en este Trabajo final de grado) diseñados, o con la previsión de ser lanzados, para AppleTV. Es cierto que algunos son pequeños desarrollos o pruebas realizados por empresas para ver cómo funciona el nuevo AppleTV, pero otros tantos son producciones más grandes. Entre estos también se encuentran opciones interesantes y de calidad,

²⁶ El término Port en el mundo de los videojuegos hace referencia al término Porting [211], que consiste en adaptar un programa para que funcione en un sistema operativo nuevo diferente al sistema original. Un Port de un juego es un videojuego que ha sido modificado para poder funcionar correctamente bajo las características del nuevo sistema.

pero aún no terminan de llamar al interés del consumidor o de tener ese valor y carisma que tienen las sagas conocidas de los videojuegos como Mario, Angry birds o Candy crush Saga.

Es necesaria una mayor cantidad de juegos, con más variedad y orientados a distinto público para poder lanzar la plataforma a un nivel superior del mercado de los videojuegos.

Posibilidades futuras

Apple está realizando cambios en sus políticas y limitaciones [72] [73] [74] impuestas para AppleTV, con lo que el futuro puede ser prometedor.

Entre ellas destaca que no será necesario el uso de Siri, sino que se podrá usar cualquier mando o controlador “third party²⁷” sin necesidad de adaptar el control también a Siri. Por tanto, se conseguirá que juegos existentes de gran calidad, que no se podían portear a AppleTV por las limitaciones del control, sean una realidad. Los nuevos juegos desarrollados para AppleTV podrán presentar mayor variedad y profundidad en su control y jugabilidad.

En general, el estado de AppleTV como plataforma de videojuegos, como ya se ha comentado, está aún por determinar [69] [75] [76]. El potencial está ahí, solo hay que descubrir hasta dónde se puede llevar.

²⁷ Las empresas o desarrolladoras third party [223] crean productos para otras compañías o plataformas que no son propias sin ningún tipo de restricción de exclusividad de la compañía para la que están trabajando.

Ejemplos de juegos en AppleTV

Para que se pueda apreciar el potencial de la plataforma, a continuación vamos a listar una serie de juegos que están disponibles en la actualidad en la AppStore para AppleTV.

- Badland [77] [78]
- SketchParty TV [79] [80]
- Xenowerk TV [81] [82]
- Beneath the Lighthouse [83]
- Galaxy on Fire [84] [85]
- Lumino City [86] [87]
- Transistor [88] [89]
- Guitar Hero Live [90] [91]



Ilustración 13 – Badland [92]

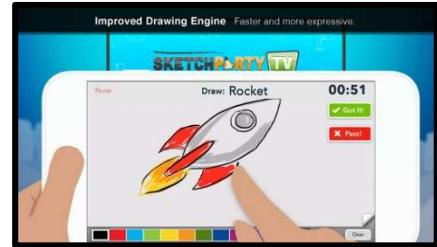


Ilustración 14 – SketchPartyTV [79]



Ilustración 15 – XenowerkTV [82]



Ilustración 16 - Beneath the lighthouse [93]



Ilustración 17 - Galaxy on fire [94]



Ilustración 18 - Lumino City [87]



Ilustración 19 - *Transistor* [89]



Ilustración 20 - *Guitar Hero Live* [95]

Creación de un videojuego

Para poder crear un videojuego, o desarrollar cualquier otro proyecto es necesario tener los recursos suficientes y las ganas de llevarlo a buen término. Pero el resultado final y los gastos producidos durante el desarrollo vienen condicionados por otros elementos, como son la planificación, grupo de personas que participan, material de trabajo, etc.

Para poder tener un producto final aceptable, evitar altos costes de producción y fallos en el producto final o, simplemente, para evitar el fracaso absoluto del proyecto y quedarse por el camino, normalmente se debe seguir un método de trabajo o planificación inicial.

En este apartado vamos a hablar del proceso que se sigue para crear un videojuego y los requisitos necesarios y las metodologías de trabajo que se pueden adoptar.

Para más información sobre el proceso o el desarrollo de videojuegos consultar [96] [97] [98].

Proceso de creación de un videojuego y etapas de desarrollo

La creación de un videojuego se puede dividir en varias fases. Aunque el proceso puede variar de muchas maneras, la etapa inicial se caracteriza por

definir el concepto del juego con el que se quiere trabajar y la idea de lo que se quiere conseguir con él.

Primera fase: Concepto e idea

Esta fase se resume en decidir el tipo de juego que se quiere hacer, puzzle, plataformas, aventura, etc y definir sus características básicas como las mecánicas jugables y aspecto visual. Además, también se tienen que tener en cuenta y concretar otros aspectos como las plataformas de salida, público objetivo, metas económicas y repercusión²⁸ que se quiere obtener.

Al final de esta etapa se suele realizar un prototipo²⁹ simple de videojuego para ver si puede funcionar la idea. Si el prototipo no termina de convencer se vuelve a revisar el concepto buscando mejorarlo.

Segunda fase: Producción y objetivos

Una vez terminado con el concepto, se pasa a la producción del juego en sí mismo. Primero se organiza el desarrollo entero del proyecto definiendo los objetivos a cumplir a lo largo del tiempo y el resultado final esperado. Después se plantean cada una de las etapas de producción y se les asignan fechas de entrega. Finalmente se divide todo en tareas y se asignan a los miembros del equipo que le correspondan.

Mientras se crea el videojuego en esta fase, se realizan prototipos para probar los elementos del juego que se van implementando. El objetivo de esta

²⁸ No es lo mismo desarrollar un videojuego comercial buscando que lo compren las masas, que crear un juego para aprender cómo funciona una nueva plataforma de desarrollo, o aprender un nuevo lenguaje de programación, o para practicar y mejorar tus habilidades y conocimientos dentro del mundo de los videojuegos.

²⁹ Un prototipo [212] en el desarrollo de software es una versión temprana del producto que aún no cuenta con todos sus elementos, pero que sirve para ver la idea que hay detrás del producto, pudiendo ver de primera mano si va a funcionar, si les interesa a los clientes o si se puede mejorar en algo el producto antes de empezar con el desarrollo del producto final.

tarea es encontrar fallos, observar el estado del juego y comprobar que el la jugabilidad funcione como se planteó inicialmente.

Tercera fase: Post-Producción y lanzamiento

Una vez terminado el juego y sacado al mercado, existe un periodo de mantenimiento del producto durante el cual se realiza un servicio de atención al cliente, supervisando el funcionamiento correcto del juego y arreglando los defectos que puedan aparecer.

Para una información más completa sobre las etapas y proceso de creación de software, véase [96] [97] [98] [99] [100].

Requerimientos para desarrollar un videojuego

Para crear un videojuego se necesitan tres recursos principales.

El primero es el tiempo. Se necesita invertir un periodo de tiempo trabajando en el producto. El tiempo de desarrollo suele estar ligado al coste de producción del juego, con lo que cuanto menos tiempo se invierta más barato saldrá el desarrollo.

El segundo recurso lo constituyen las herramientas de trabajo o de desarrollo, que son las que se utilizan para crear cada uno de los elementos que conforman el videojuego, cómo son el sonido, los modelos de los personajes, los escenarios, etc., y van desde ordenadores y tabletas de diseño gráfico hasta equipos de grabación de sonido y captura de movimiento.

El último recurso son los propios trabajadores y sus perfiles profesionales. En el desarrollo de un videojuego intervienen una gran variedad de actividades como, por ejemplo, diseñar, ilustrar, programar, producción, organización, etc. Todas estas actividades se encargan de distintos tipos de

roles dentro del proyecto que conforman el equipo de desarrollo. Los roles básicos son:

Diseñador: Crean el concepto inicial del juego, su jugabilidad y mecánicas.

Diseñador de niveles: Desarrollan la interacción entre el jugador y cada uno de los niveles. Proponen los retos que el jugador va a superar usando las mecánicas del juego.

Escritor (guionista, historia): Crean la historia y el trasfondo del videojuego. Incluye los diálogos, textos de pantalla y manuales dentro del juego.

Programadores (junior, senior): Se encargan de crear la lógica del juego y todo el software que le dará forma.

Artistas 2D/3D, conceptuales y animadores (Diseñadores gráficos): Crean el aspecto visual del juego y todos sus componentes.

Diseñador y compositor de audio: Crean todos los sonidos y música del juego.

Testers: Realizan pruebas constantes del juego e informan de los problemas y errores que se van encontrando.

Gestión del proyecto(Productores): Se encargan de los temas legales, planificación, coordinación, publicación y representación de la compañía.



Ilustración 21 – Equipo de trabajo de Respawn, creadores de Titanfall³⁰ [101]

Para completar el desarrollo del videojuego, cada uno de los trabajadores tendrá uno o varios de estos perfiles. Según la especialización de cada trabajador podrá haber una persona dedicada a un solo perfil, como programador, o dicha persona será capaz de cumplir varios perfiles, como programador y diseñador de niveles. En otras situaciones, debido a las circunstancias de desarrollo, una persona tendrá que cumplir varios o todos los roles. En cualquier caso, todos³¹ los perfiles profesionales son importantes para poder desarrollar todos los aspectos de un juego.

Método de trabajo

A la hora de realizar un trabajo o proyecto de cierta envergadura, ya sea en tiempo, magnitud o ambas, es recomendable utilizar una metodología [102] [103] de trabajo para organizar el desarrollo del proyecto y que llegue a buen puerto. Y, el mundo de los videojuegos, no es una excepción ya que la base del proceso de desarrollo es igual al de cualquier tipo de producto software. Básicamente una metodología sirve para dividir en una secuencia de pasos y

³⁰ Referencia de imagen acceso público.

³¹ Aunque es cierto que algunos de ellos no son estrictamente necesarios para tener un videojuego terminado, desde luego ayudan a tener un buen producto acabado y sin fallos. Cuantos menos roles distintos participan más problemas pueden aparecer o más probabilidades de que el resultado final no sea el deseado o con la calidad que se buscaba.

tareas el proyecto, que una vez terminadas da como resultado la finalización del producto.

Los tipos de metodologías han ido evolucionando a lo largo de los años, desde un enfoque más tradicional sin ningún tipo de organización a las nuevas metodologías ágiles para proyectos de corta duración o de cambio constante.

El beneficio de usar una metodología de trabajo

El objetivo de las metodologías es definir los requisitos del proyecto, organizar el tiempo del desarrollo y establecer una forma de trabajo para completar cada una de las etapas en el tiempo establecido, todo manteniendo unos costes aceptables de producción. Una de las ventajas de usar una metodología consiste en que permite encontrar problemas o defectos, ya sea en la organización o en el propio proyecto, más rápidamente y posibilita identificar y evaluar los daños de dichos problemas.

Para elegir una metodología adecuada para un proyecto se tiene que tener en cuenta tanto el tamaño del grupo de trabajo como el tipo de proyecto que se va a realizar.

Tipos de metodologías

En cascada: Metodología de desarrollo lineal en el que las fases de desarrollo³² se ejecutan en secuencia, una detrás de otra.

Prototipado: Desarrollo iterativo en el que se permite observar la funcionalidad básica del producto sin tener terminadas todas las características del mismo. Permite ver una versión temprana del producto.

³² Fases de desarrollo: Análisis, diseño, implantación, pruebas, integración y mantenimiento.

Incremental: Combinación de desarrollo lineal e iterativo. Se combinan ambos desarrollos para reducir el riesgo del proyecto dividiéndolo en pequeñas partes durante todo el proceso de desarrollo y tratándolas de manera independiente.

Espiral: Combinación de lineal e iterativo. Se basa en la ejecución de un ciclo de proceso de desarrollo de manera iterativa en la que cada vez se repiten todos los pasos hasta terminar el producto.

Rapid application Development(RAD) o metodología ágil: Desarrollo iterativo. Se centra en la creación rápida de prototipos en lugar de una planificación exhaustiva inicial. Los prototipos van creciendo en complejidad incluyendo las funcionalidades anteriores.

Las metodologías de desarrollo clásicas como la de tipo cascada no son las que mejor funcionan para desarrollar software y videojuegos. En estos casos funcionan mejor las metodologías más dinámicas y que permiten revisar más a menudo el estado de los proyectos, como las metodologías ágiles.

Requisitos del proyecto

Para poder realizar este proyecto y completar sus objetivos va a ser necesario cumplir con una serie de requerimientos de los que poder partir. Se debe tener unas habilidades o conocimientos básicos iniciales para poder afrontar el proyecto, los cuales, sino se poseen, será necesario aprender y desarrollar. Una vez empezado el proyecto y, conforme se va desarrollando, estos requisitos pueden cambiar o aumentar, teniendo entonces que adaptarse el proyecto a las nuevas circunstancias. Por eso es importante realizar un análisis y estudio previo del proyecto, para poder planificar mejor su desarrollo y sus posibles alteraciones.

Tiempo

En la universidad el valor de las asignaturas y del trabajo final de grado, a pesar de que el método docente es distinto³³, viene medido por una unidad de medida académica propia, los créditos [104]. Estos créditos, de tipo ECTS (European Credit Transfer System), equivalen al esfuerzo que debe realizar el estudiante para superar una asignatura. En este esfuerzo que se presupone por parte del alumno se tiene en cuenta el tiempo de asistencia a clase, horas de estudio, realizar trabajos, proyectos y horas de preparación en general. La equivalencia en número de horas de 25 horas de trabajo por crédito.

El trabajo final de grado cuenta con un valor en créditos de 12. Por lo tanto, la cantidad de tiempo que se le debería dedicar para superarlo con éxito rondaría las 300 horas de trabajo. Cómo se ha mencionado antes, el trabajo final de grado se desarrolla y evalúa de manera distinta, pero su equivalencia en horas se suele mantener en estas 300 horas.

Las diferencias fundamentales entre el resto de asignaturas y el trabajo final de grado son la metodología docente. Primero se realiza un seguimiento con tutorización del trabajo. Y segundo, el periodo de tiempo del que se dispone para cumplir los objetivos y entregar los resultados³⁴ varía según las circunstancias.

En el caso en el que nos encontramos, el proyecto se decidió que comenzara en junio de 2016, con la entrega límite pensada para la convocatoria

³³ El trabajo final de grado forma parte de una prueba para medir el nivel y los conocimientos aprendidos del alumno, así como su capacidad para aplicar todo lo aprendido y comprobar cómo se desenvuelve en un proyecto real relacionado con su ámbito académico de estudios.

³⁴ El periodo de desarrollo del trabajo final de grado depende del alumno ya que él decide la convocatoria a la que debe presentarse y el tiempo que quiere dedicarle al proyecto. Normalmente se dispone de un cuatrimestre o un curso académico completo, aunque se pueden elegir periodos de tiempo más cortos para poner a prueba las capacidades propias o simplemente para completar la carrera académica cuanto antes y poder enfrentarse al mercado laboral. Información extra sobre el trabajo final de grado y las condiciones sobre las que se está trabajando [222].

de septiembre del mismo año, así que el tiempo del que se ha dispuesto para desarrollar el proyecto y terminar el trabajo final de grado ronda los tres meses.

Tres meses es un tiempo bastante limitado para realizar el proyecto y en el caso que nos ocupa, un estudio e implementación de un videojuego en una plataforma nueva no conocida para el alumno. Esto supone una gran prueba de las capacidades para la toma de decisiones y organización. No obstante, se ha tenido en cuenta esta limitación a la hora de definir los objetivos principales y el nivel de desarrollo del proyecto.

Conocimiento y aptitudes

Para realizar este proyecto se necesitan una serie de conocimientos y habilidades de las cuales, si el alumno no dispone de ellas en el momento de empezar el trabajo tendrá que aprender y estudiar mientras lo desarrolla.

- Conocimientos de la herramienta de Unity y cómo funciona.
- Conocimientos de cómo gestionar y organizar el desarrollo de un videojuego.
- Conocimientos de cómo crear un videojuego, su estructura y la adaptación de la misma al motor de Unity.
- Conocimientos de AppleTV, sus características y sus limitaciones.
- Informarse del estado de AppleTV, su mercado y su valor como plataforma de juegos.
- Conocimientos sobre la adaptación de videojuegos a distintas plataformas, a los distintos tipos de control, específicamente para AppleTV.
- Conocimientos de programación, tipos de lenguajes y estructuras de programación.

- Conocimientos de modelado y diseño, tanto en 3D como en 2D. Aquí también se incluyen animaciones, efectos, texturas y aspecto visual de los mismos.
- Habilidades para buscar información y resolver problemas aportando soluciones características de los Ingenieros.
- Capacidad de toma de decisión frente a un proyecto para obtener un buen resultado final.
- Capacidad de autocrítica y superación para aprender de los errores y mejorar como ingeniero y cómo persona.

Material

Para poder afrontar el trabajo se necesitará acceso a una serie de herramientas y materiales que, si no se disponen, se tendrán que adquirir para completar con éxito el proyecto. Las herramientas a utilizar elegidas se enumeran en otra sección de la memoria, pero el equipo de trabajo constará de:

- Acceso a un ordenador para instalar los programas y realizar el desarrollo del videojuego, así como la redacción de la memoria y la búsqueda de información.
- Acceso a un AppleTV, dispositivo indispensable ya que el trabajo se centra sobre el mismo. Necesario para poder realizar pruebas de campo.
- Tener acceso a una conexión a internet estable³⁵.

AppleTV

Como este proyecto se basa en crear un videojuego para AppleTV, hay que añadir los requerimientos de la plataforma a la hora de planificar su

³⁵ Hoy en día el acceso a una buena conexión a Internet estable es indispensable para poder realizar casi cualquier trabajo de ingeniería, sobre todo si el tipo de trabajo es desarrollo de software.

desarrollo. Debido a sus características, los requisitos de AppleTV son más parecidos a los requisitos de los dispositivos móviles que de los ordenadores o consolas de sobremesa. Algunos ya se han mencionado con anterioridad, pero aquí vamos a especificar las características.

Publicación y calidad

Para poder publicar un producto, sobre todo si es para usar un servicio proporcionada por otra empresa, hay que seguir unos requisitos y guías para controlar que las necesidades de calidad y funcionamiento se cumplan.

Para publicar para Apple se usa el servicio de la App Store. Habrá que cumplir una serie de pautas y estándares que aseguren que el funcionamiento del producto es el adecuado según las directrices de Apple [105] [106].

Para poder publicar productos en la App Store es necesario darse de alta como desarrollador de iOS [107], con un coste de 99\$, teniendo que obtener una cuenta de desarrollador y un certificado [108]. También hay que seguir una serie de pasos para poder subir productos a la tienda [109] [110].

Apple dispone de guías y secciones web dedicadas a explicar todo este proceso [111] [112].

Requisitos de AppleTV

- El tamaño de las aplicaciones está limitado a 200MB. Esto parece reducir inicialmente la envergadura del proyecto, pero existen formas de poder subir aplicaciones de mayor peso a la Apple Store, solo que requieren de conocimientos y métodos más específicos. Esta opción se llama On-Demand Resources.

- Requerimiento de que la aplicación funcione siempre con el mando “Siri”. Aunque acepte otros tipos de controladores es necesario que se pueda acceder a todas las funcionalidades del producto mediante el mando de AppleTV.

On-Demand Resources

Debido a la importancia para poder desarrollar juegos de todos los tipos y calibres posibles, vamos a dedicar un pequeño apartado de los requerimientos a explicar esta funcionalidad.

Usando On-Demand Resources [113] [114] se puede aumentar el tamaño de la aplicación que se quiere subir a AppleTV hasta a 20GB [115]. A pesar de la restricción de 200Mb, la aplicación realmente puede utilizar al mismo tiempo 2.2GB de espacio, con el tope de almacenamiento en 20GB.

El funcionamiento de esta técnica consiste en dividir la aplicación en partes más pequeñas y en aprovechar que nunca se accede a todos los elementos del juego a la vez, con lo que hay partes del mismo que no se cargan hasta que no se utilizan activamente, ayudando con esto a poder repartir el peso de manera más dinámica.

Aunque el tamaño total permitido final no llega al de las grandes producciones modernas [116] [117] [118] [119], sí que hay suficiente espacio para hacer casi cualquier tipo de juego viable en la plataforma. Solo habría que tener en cuenta el límite de tamaño a la hora de definir y plantear las características del juego y el nivel de detalle.

Algunos de los ejemplos referenciados, que ocupan entre 30GB y 50GB de espacio en el disco duro, son de los juegos más modernos, con mejores calidades gráficas y que, por tanto, son de mayor tamaño. Pero existen otros

juegos que entran dentro del límite, como por ejemplo Super Mario 3D World, que no llega a los 20GB del límite con sus 18 GB de peso [120]. El videojuego de tipo MOBA³⁶ League of Legends necesita 8GB [121], el Counter-Strike: Global Offense solo necesita 8GB de espacio en disco [122] y el pasado Battlefield 3, requiere justo 20GB de espacio en disco [123].

Básicamente, el límite de 200Mb está para reducir el peso de descarga de la aplicación, reduciendo los tiempos de descarga y lo que ocupa en el dispositivo mientras no se está usando activamente.

Adaptación del control

Para hacer que funcione el juego en AppleTV habrá que adaptar el control al mando remoto “Siri” y al sistema operativo. Para facilitar esto, Apple aporta unas pautas para adaptar los controles a “Siri” [124] y ayudas al desarrollo y formato a seguir con el control [125].

Unity da soporte a los dispositivos iOS, y appleTV también está incluido. Proporciona directrices para poder gestionar e incluir el controlador en los juegos [126] [127]. Cabe destacar que la información que proporciona Unity es genérica de cómo mapear³⁷ los controles de un sistema a otro, no de cómo funciona internamente, con lo que conseguir el control deseado supone un problema y una inversión de tiempo extra.

³⁶

³⁷ Mapping o data mapping [224] consiste en traducir/transformar datos de un sistema a otro. También puede hacer referencia a crear un sistema que interprete los datos de un sistema para que los pueda interpretar o usare el otro sistema. En este caso, mapear los controles se refiera a que teclas del mando “Siri” se pueden referenciar dentro del entorno de Unity.

Costes

En el mundo laboral, desempeñar cualquier trabajo o realizar un proyecto tiene asociados unos costes. Este proyecto es un caso especial, pero aun así se debe realizar una estimación de costes de lo que sería desarrollar el proyecto para poder darle otro valor a la inversión de tiempo que se va a realizar.

Ya se ha calculado con anterioridad que el número de horas que se le deben de dedicar al proyecto son 300 horas. Por lo tanto, vamos a considerar que para el proyecto se contrata a un empleado para realizar un trabajo de duración estimada en 300 horas.

Las características del contrato también son especiales, ya que el empleado debe tener un gran abanico de habilidades y conocimientos como se ha mencionado antes. Aunque en esta situación no se debería considerar un sueldo normal dado que el desarrollo es de ámbito independiente y que la experiencia de trabajo es nula. Así que, después de analizar los salarios base de las distintas disciplinas [128] [129] [130] [131], y de estudiar brevemente la situación del desarrollo independiente³⁸ [132] [133] [134] [135] [136], se va a tomar como referencia el salario estándar de desarrollo software como base, sin tener en cuenta el cariz multidisciplinar del trabajo.

El salario medio de un desarrollador ronda los 18€ euros la hora, pero debido a las condiciones especiales del empleado, nula experiencia de trabajo y formato independiente del desarrollo, vamos a estimar que se va a cobrar el salario mínimo de desarrollador, unos 11€ euros la hora [137].

³⁸ El desarrollo independiente sigue como siempre ha sido a excepción de que hay más posibilidades de distribuir y exponer tu trabajo. Tiene la ventaja de que puedes controlar el resultado del producto y desarrollarlo a tu gusto. Las desventajas por otro lado son numerosas, la gran cantidad de competencia en el mismo campo, la competencia que viene de las grandes compañías, depender enteramente de que tu juego se haga notar por encima del de los demás y que triunfe, cantidad de ingresos variables, etc.

No se va a incluir el coste del material o las herramientas, ya que se ha tomado la decisión de que todo el desarrollo se realice con material gratuito o de libre acceso.

Teniendo en cuenta todos estos factores, el coste estipulado en el contrato por realizar este trabajo sería:

Tabla 1 costes del proyecto, tiempo en créditos

HORAS	€/HORAS	TOTAL
300 horas trabajo	11 €/h	3.300€

- El coste total del contrato fijo sería de 3.300€.

Este cálculo se ha realizado teniendo en cuenta un contrato prefijado tomando como medida el valor de créditos definidos por la universidad.

Ahora vamos a realizar el cálculo teniendo en cuenta el número de horas según la duración real total del desarrollo del proyecto, tres meses de desarrollo empezando en el mes de junio de 2016. Así que ajustando el contrato a las nuevas circunstancias de tiempo y teniendo en cuenta una jornada laboral de 6 horas, trabajando 6 días a la semana durante 3 meses, la estimación de trabajo en horas ronda las 432h. Por tanto, el total se quedaría como sigue:

Tabla 2 costes del proyecto, tiempo de desarrollo real

Meses	Semana laboral	Jornada laboral	€/hora	TOTAL
3	6 días	6 horas	11€	4.752€

- El coste total del proyecto serían 4.752€.

HERRAMIENTAS PARA EL PROYECTO

Serán necesarias una serie de herramientas para cada etapa del desarrollo del producto, así como de las áreas en que se divide la creación de un videojuego como contenido multimedia.

Control y gestión del proyecto

Para gestionar y estructurar el proyecto con sus etapas se van a usar las siguientes herramientas:

Trello

Trello [138] es una aplicación web que sirve para gestionar y organizar proyectos de una manera más visual. Se puede utilizar tanto para trabajos individuales como en grupo.



Ilustración 22 – Icono de Trello³⁹ [139]

Consiste en la disposición de un tablero en formato web, en el cual, se pueden colocar diversas tarjetas y agruparlas por temas como si de post-it se tratasesen.

Está disponible tanto en versión Web como en versión aplicación para distintos dispositivos, entre ellos Android, iPhone, iPad y Kindle.

³⁹ Creative Commons Attribution-Share Alike 3.0 Unported.

Funcionamiento

Trello funciona alrededor de lo que los creadores denominan tarjetas. Las tarjetas pueden tener un nombre y se puede escribir en ellas. Además, permiten incluir comentarios, subir archivos y vienen con una serie de funciones especiales como poder añadir etiquetas, incluir checklist, fechas, cambiar color, etc. También permite agrupar estas tarjetas en listas, pudiendo de esta manera repartir las tarjetas según categorías.

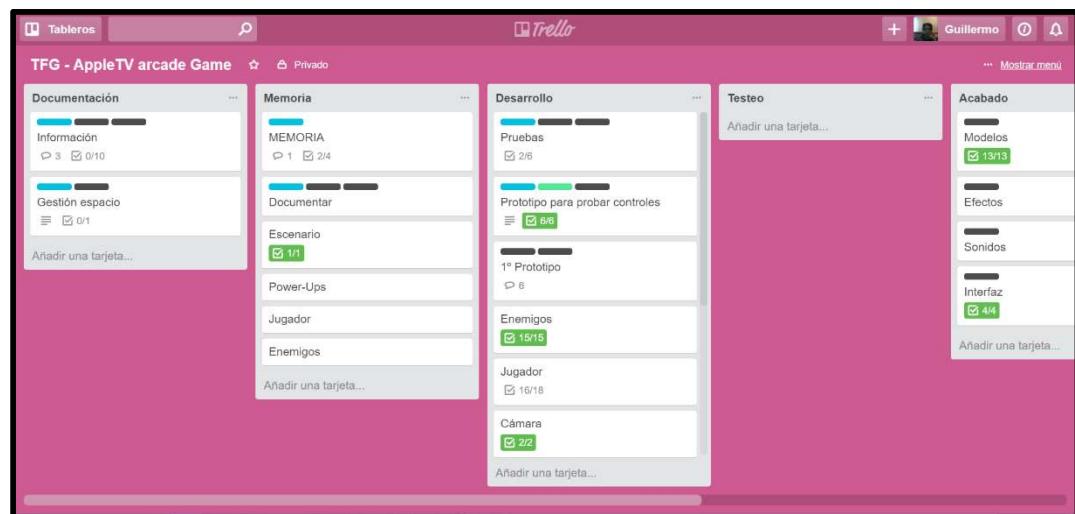


Ilustración 23 – Tablero de Trello. Tareas repartidas en listas

Mediante el uso de las tarjetas y listas se puede organizar un proyecto entero. Cada tarjeta es una tarea o responsabilidad, y las listas se pueden organizar y clasificar cómo el usuario de la aplicación considere oportuno. Por ejemplo, se podría listar la persona que tiene que trabajar en esas tareas, conformando cada lista una parte del proyecto.

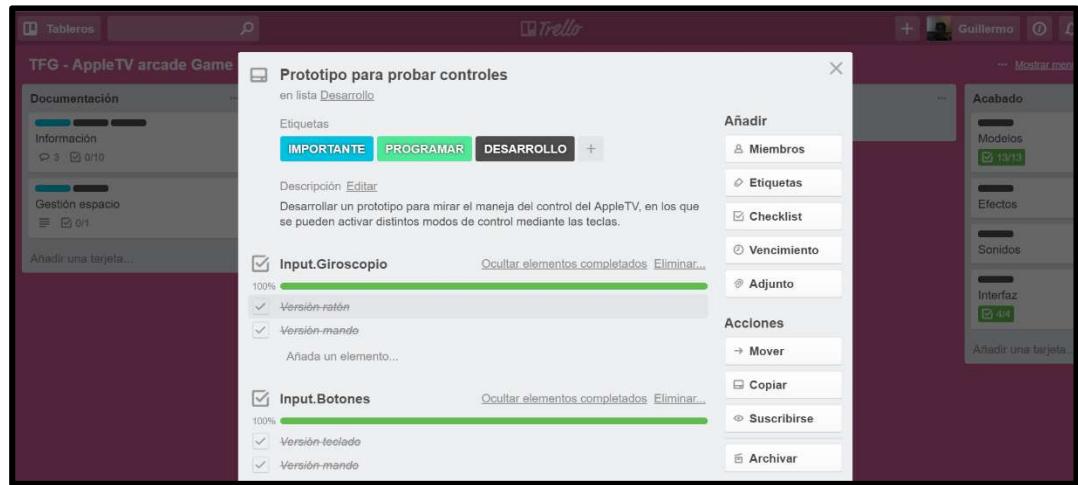


Ilustración 24 – Tarjeta de Trello, descripción y funcionalidades

Como se ha mencionado, Trello también sirve para organizar un proyecto con varios miembros. Simplemente vinculando el proyecto a otras cuentas de Trello permite compartir el tablero con el resto, y todos pueden interactuar con él. Se pueden arrastrar o vincular personas a tarjetas, y cuenta con un sistema de notificaciones integrado.

Otro de los elementos interesantes de Trello es que permite cambiar al gusto tanto las tarjetas como las listas, ofreciendo la posibilidad de modificar y adaptar el desarrollo del proyecto según va cambiando.



Ilustración 25 – Trello. Modificación de tarjetas y listas en cualquier momento

Básicamente Trello es como tener un tablero en el que se pegan post-it, pero de manera virtual, en la que en cualquier momento y lugar se puede acceder a él y actualizarlo.

Disponibilidad de la aplicación

Se ofrece una versión gratuita, que cuenta con todas las funciones mencionadas con anterioridad disponibles. También tiene varias versiones de pago que añaden nuevas funcionalidades y mejoras, como aplicaciones integradas, mejoras de seguridad, más espacio de almacenamiento, etc.

Uso de Trello en el proyecto

Se utilizará principalmente para estructura, organizar y llevar constancia del desarrollo del proyecto gracias a su facilidad de manejo y la visibilidad que otorga del estado del proyecto.

Google Drive

Google Drive [140] es un sistema de almacenamiento en la nube⁴⁰ que te permite guardar todos tus archivos y acceder a ellos desde cualquier dispositivo. Creado por la compañía Google⁴¹, viene integrado en el navegador Google Chrome⁴² haciendo posible acceder a todos los datos desde el navegador web. Los archivos se guardan tanto en la nube como en el propio ordenador, pudiendo acceder a ellos a través de una carpeta en este último caso.

⁴⁰ El almacenamiento en la nube [226] consiste en guardar una copia virtual de todos los elementos, a los que puedes acceder desde cualquier dispositivo con acceso a internet. Los datos se suelen almacenar en servidores externos, proporcionados por regla general por una compañía que puede o no cobrar por este servicio.

⁴¹ Google [225] es la compañía responsable de crear el motor de búsqueda de internet más usado del mundo, Google. Está especializada también en proporcionar productos y servicios software relaciones con Internet.

⁴² Google Chrome [227] es el navegador web creado por Google. Está centrado y diseñado para soportar y gestionar todas las aplicaciones y herramientas diseñadas por Google.

Hay que tener en cuenta que si no se accede a la aplicación mediante el navegador no se podrá acceder a todas las funcionalidades y ventajas que ofrece.

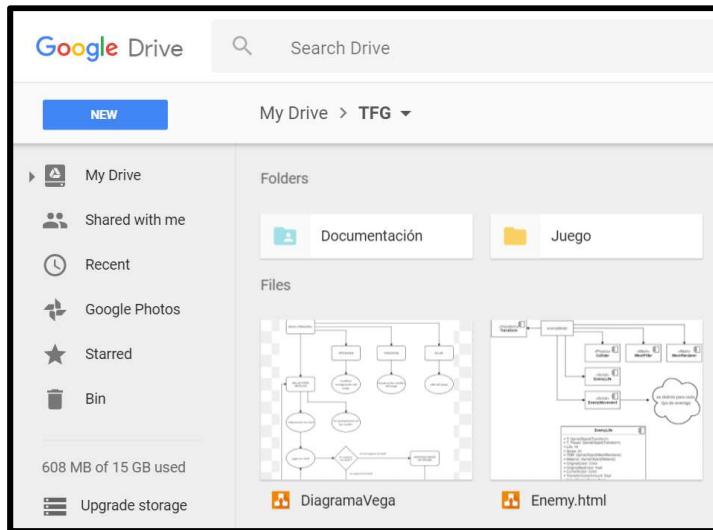


Ilustración 26 – Escritorio virtual GoogleDrive

Drive te permite almacenar y gestionar tus archivos creando carpetas y ofreciendo una gran variedad de funcionalidades, como compartir y editar archivos fácilmente. También te permite crear más contenido desde el propio navegador mediante las aplicaciones integradas [141], desde documentos de texto y presentaciones hasta hojas de cálculo, pudiéndose usar todos ellos desde el navegador sin necesidad de instalación previa.

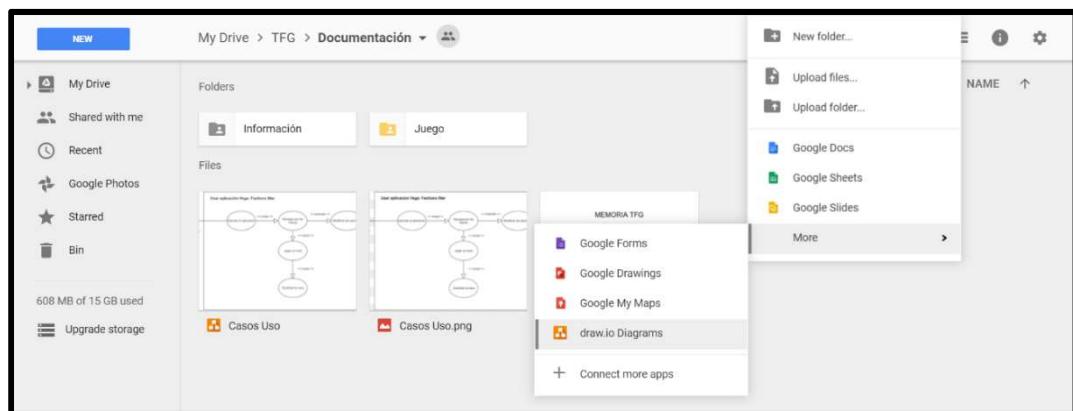


Ilustración 27 – Aplicaciones integradas en GoogleDrive

Cuenta con una capacidad inicial de almacenamiento gratuita de 15GB, dando la posibilidad de ser ampliada mediante pago [142].

Uso de Google Drive en el proyecto

Se utilizará junto a Dropbox como uno de los medios de almacenamiento para guardar la documentación y gestionar los archivos y este mismo documento.

Dropbox

Dropbox [143] [144] es otra aplicación que funciona como un sistema de almacenamiento en la nube parecido a Google Drive, almacenando y sincronizando archivos en línea. Cuenta con una versión gratuita con una cantidad de espacio limitada ampliable mediante la compra de la versión de pago [145].

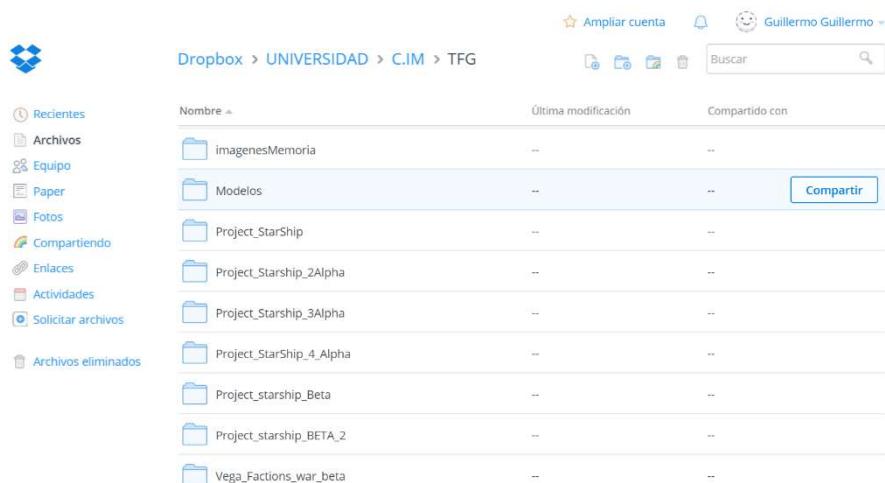


Ilustración 28 – Escritorio web de Dropbox

Uso de Dropbox en el proyecto

De la misma manera que Google Drive, se usará para gestionar los archivos del trabajo final de grado. Mientras que Google Drive se utilizará por

sus funcionalidades añadidas, Dropbox se usará como almacén para copias de seguridad de todo el proyecto.

TrackingTime y TimeTune

TrackingTime [146] y TimeTune [147] son aplicaciones que sirven para gestionar y organizar el tiempo de trabajo. Se pueden configurar para que te notifiquen de la duración de las actividades o para llevar la cuenta de las horas dedicadas a cada tarea. Útiles para gestionar y controlar tu manejo del tiempo.

Uso de TrackingTime y TimeTune en el proyecto

Se utilizarán para llevar la cuenta de las horas trabajadas a lo largo del día y para gestionar y distribuir los descansos manteniendo el horario planificado.

Apartado de programación

Las herramientas de trabajo principales para el desarrollo del videojuego y de todas sus funcionalidades serán:

Unity

¿Qué es Unity?

Unity [148] [149] es un motor de desarrollo de videojuegos y aplicaciones que permite crearlos de forma rápida, sencilla y con una interfaz gráfica muy intuitiva que facilita mucho todo el proceso.



Ilustración 29 – Logo de Unity⁴³ [150]

Unity soporta el desarrollo en multitud de plataformas [151], desde Windows y MAC, incluyendo Android, PS4, Xbox360, WiiU...

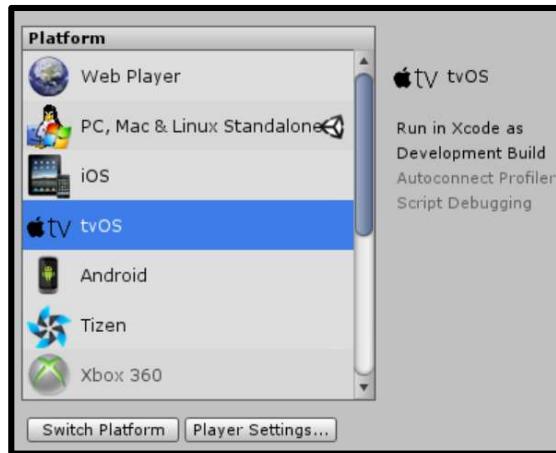


Ilustración 30 – Algunas de las plataformas a las que da soporte Unity

Cuenta con varias versiones disponibles de cara al usuario [152]. Vamos a destacar algunas de ellas:

- Unity Personal: Gratuita. Esta versión ofrece lo necesario para poder desarrollar cualquier videojuego casi sin ningún tipo de restricción, con lo que para usuarios que quieren empezar en el sector o para realizar pruebas o uso personal es una buena elección.
- Unity Profesional: 1500 dólares. Acceso a todas las funcionalidades de Unity, poder personalizar la interfaz y acceder a otras herramientas externas, así como la obtención de la licencia de Unity.

⁴³ Referencia imagen acceso público.

¿Por qué Unity?

Unity, en su versión gratuita, es una muy buena elección dentro de las herramientas gratuitas que se pueden encontrar en la actualidad para realizar prototipos de forma rápida y sin mucha complicación. Para la propuesta del Trabajo Final de Grado se proponía abordar todas las etapas del desarrollo, con lo que una herramienta que facilita el trabajo y permite centrarse en otros elementos del proceso de desarrollo es ideal para este proyecto. Esto, junto con el hecho de haber trabajado anteriormente con la herramienta y conocer de primera mano sus ventajas y desventajas han sido los factores determinantes para elegir Unity como herramienta de trabajo.

Para más características de lo que aporta Unity como herramienta, véase [153].

MonoDevelop

Con Unity se pueden usar distintas herramientas de programación y desarrollo⁴⁴, ya que se puede vincular a sus proyectos la gran mayoría de ellas, pero para este trabajo y aprovechando que con Unity ya te viene su propia herramienta de programación interna, MonoDevelop, será la que se use.

MonoDevelop [154] [155] no tiene todas las funciones que se desearían de una herramienta completa e independiente para programar, como **Netbeans** [156] o **Eclipse** [157], así que se usará Notepad++ para complementar dichas faltas.

⁴⁴ Las herramientas de desarrollo o IDE [228] facilitan el desarrollo de software mediante una serie de herramientas que facilitan la programación, la gestión de la aplicación y el control de fallos.

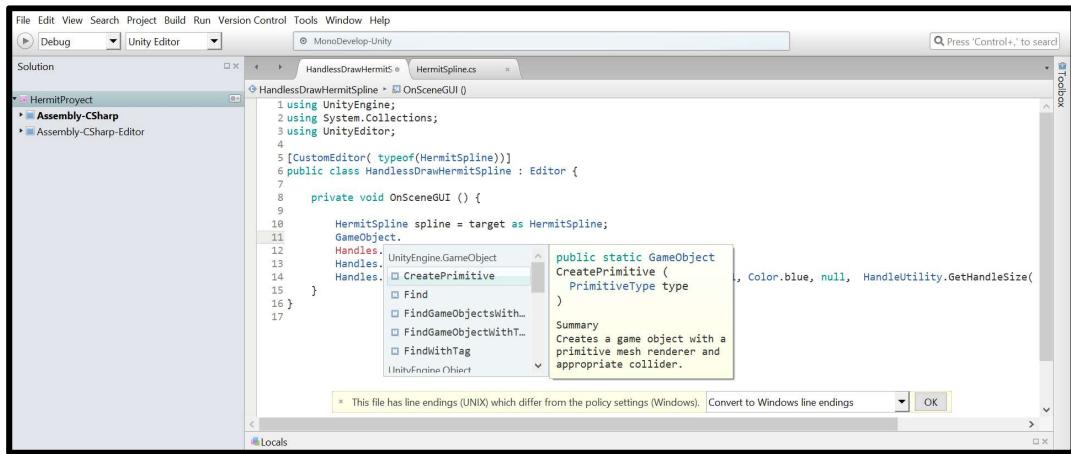


Ilustración 31 – MonoDevelop. Función de autocompletado de las instrucciones de Unity

Aun así, la implementación dentro de Unity de monoDevelop y las características extra que esto aporta a la hora de programar, como el autocompletado con las funciones, clases y objetos propias de Unity, la convierten en una buena herramienta para iniciarse en la programación con Unity.

NotePad++

NotePad++ [158] es una herramienta de desarrollo de código fuente gratuita con soporte a varios tipos de lenguaje de programación. Está basado en su anterior versión NotePad. Al ser una herramienta de uso gratuita, pero con varias funciones muy útiles para la programación, incluyendo algunas que normalmente tienen las herramientas de pago, hacen que su uso sea popular.

Unity remote 4

Unity remote 4 [159] es una aplicación que permite conectar el proyecto Unity en ejecución con un dispositivo Android o iOS para realizar pruebas de rendimiento, control y ejecución sin tener que compilar o adaptar completamente el proyecto previamente a dicha plataforma.

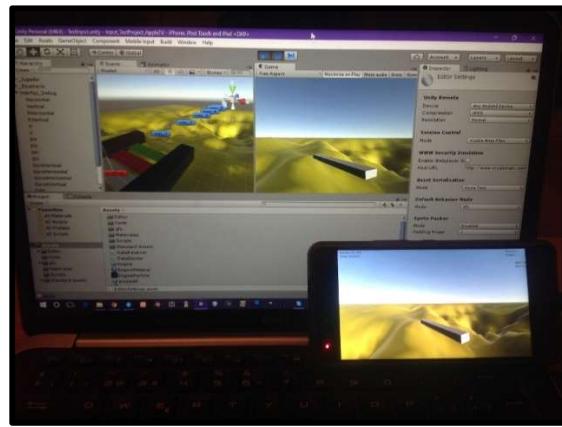


Ilustración 32 – Unity remote 4 funcionando con Unity versión de escritorio en el prototipo de control de movimiento de Vega: Factions War

Xcode

¿Qué es Xcode?

Xcode [160] [161] es una aplicación desarrollada por Apple que funciona como un entorno de desarrollo y programación IDE. Xcode se utiliza para crear software específico en sistemas operativos iOS y permite adaptar el software a sus características. Tiene soporte para varios tipos de lenguajes de programación como C, C++, Java, Python, Ruby y otros. Proporciona muchas funcionalidades como herramienta de desarrollo [162].

Uso de Xcode para el proyecto

Como entorno de desarrollo IDE para iOS, se usará para transformar y adaptar la versión de compilación⁴⁵ del videojuego obtenida con Unity al sistema operativo iOS y crear la aplicación final que ejecutará el videojuego en el AppleTV.

⁴⁵ Unity te permite compilar [229] o adaptar el videojuego a distintas plataformas y sistemas operativos, ya sea Android, iOS o Windows, mediante un simple “click”. Para el caso concreto de iOS, está primera compilación solamente adapta el proyecto para que pueda ser interpretado en un entorno iOS, pero se debe recompilar con Xcode otra vez para crear la versión final de la aplicación que funcione como tal.

Apartado gráfico

Para el apartado artístico y diseño del videojuego se van a usar las siguientes herramientas:

Blender

¿Qué es Blender?

Blender [163] es un programa de modelado y renderizado de escenas en 3D⁴⁶. Ha sido desarrollado con la mentalidad open source [164], así que es de acceso gratuito y todo el contenido creado con el mismo pertenece al autor.



Ilustración 33 – Icono de Blender⁴⁷ [165]

Cuenta con dos motores de renderizado internos y herramientas para animar, texturizar⁴⁸, editar videos y multitud de opciones para modelar. Además, al ser software open source te permite crear tus propias herramientas e incluirlas en la aplicación.

⁴⁶ Los modelos 3D [246] son representaciones tridimensionales por computador de cualquier objeto. Para crearlos se usa software especializado.

⁴⁷ Referencia imagen acceso público.

⁴⁸ Texturizar [213] es el proceso por el cual a un modelo 3D se le aplican imágenes a su superficie, simulando en esa superficie el aspecto de la imagen. Ejemplo: Si sobre un cubo se coloca en sus caras una imagen de una pared de ladrillos, dará la impresión de que el cubo está hecho de ladrillos.

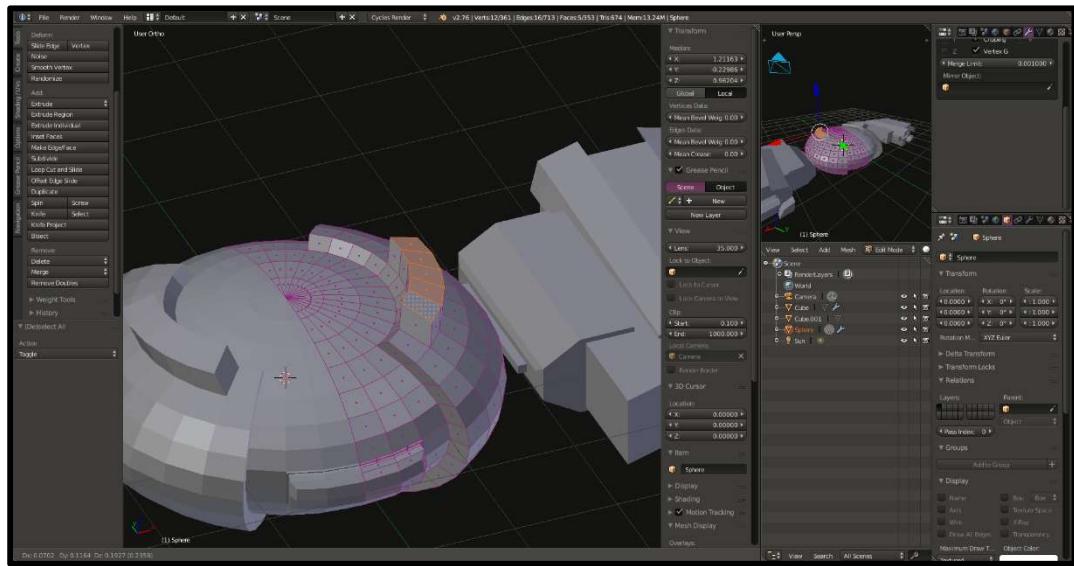


Ilustración 34 – Ventana de trabajo de Blender. Modelando la primera iteración de la nave del jugador

Su elección como herramienta de trabajo viene dada porque ya se había trabajado con Blender con anterioridad. Además, como se ha mencionado antes, es de software libre y gratuito con lo que no habrá ningún problema legal o económico.

Manual oficial de Blender [166].

Uso de Blender en el proyecto

Se utilizará para crear y texturizar todos los modelos3D y entornos del videojuego.

AwesomeBump

AwesomeBump [167] es una herramienta de desarrollo para generar texturas de tipo normal, altura, especular, occlusiones y otras a partir de una imagen original. Con esta herramienta se pueden obtener, con un resultado aceptable, los distintos tipos de mapas de texturas necesarios para los modelos de un videojuego a partir de una simple fotografía o imagen.

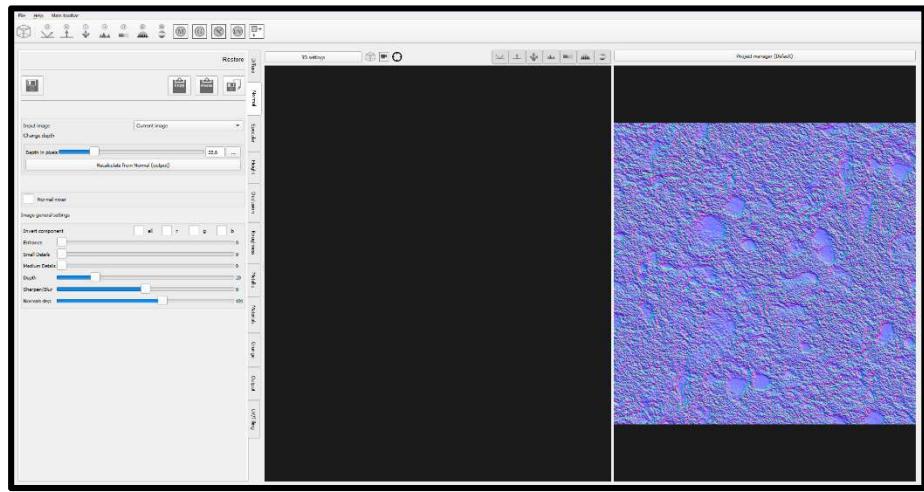


Ilustración 35 – AwesomeBump4.0Beta. Obteniendo el normalMap de una textura

Esto facilita todo el proceso de obtención de dichas texturas sin depender de un programa de pago o del proceso y conocimientos de edición propios de imágenes requeridos para obtenerlas mediante herramientas de edición de imágenes.

EL JUEGO (Vega: Factions War)

En esta sección vamos a explicar en qué consiste el juego que se va a desarrollar en este proyecto. Además, se explicarán sus fundamentos y características, así como la manera de jugar.

Inspiración

Vega: Factions War está basado en todos los juegos de género arcade sobre raíles⁴⁹ [168], siendo inspirado mayoritariamente por la saga de juegos StarFox [169] creada por Nintendo⁵⁰. En concreto, la entrega de la saga que más ha inspirado la creación del videojuego ha sido **StarFox64** [170], la versión desarrollada para la consola de sobremesa Nintendo64 en 1997⁵¹.



Ilustración 36 – Imagen de la caja del juego StarFox64⁵² [171]

⁴⁹ Género de jugabilidad rápida e intensa que es muy recurrido en dispositivos móviles y tablets, ya que los usuarios pueden jugar partidas cortas. Es un tipo de videojuego que suele atraer al público, pero la calidad de los mismos varía mucho de unos a otros. Existe un género específico sin el componente arcade, centrado en recorrer un escenario esquivando obstáculos y recogiendo objetos para aumentar la puntuación [234] [235].

⁵⁰ Nintendo [233] es una empresa multinacional fundada en 1889 dedicada a la industria de los videojuegos. Es reconocida por sus famosas sagas de juegos, como Mario, Zelda. También desarrolla consolas como el éxito de ventas en sobremesa Wii, o las consolas portátiles DS y 3DS.

⁵¹ La Nintendo 64 [230], con fecha de salida 1996, fue la tercera consola de sobremesa desarrollada por Nintendo, que contaba con un procesador de 64 bits y los juegos tenían formato cartucho. Es conocida por juegos famosos como Super Mario 64 [231] y The Legend Of Zelda: Ocarina of Time [232] y por apostar fuerte por el desarrollo 3D de los videojuegos.

⁵² Licencia: Se puede usar como imagen de baja resolución para identificar el juego.

StarFox64 fue un videojuego de disparos en 3D sobre raíles, centrándose la acción en combates de naves espaciales. Desarrollado para la plataforma de videojuegos Nintendo64 en 1997 como remake de la saga adaptada a la tecnología 3D, mantiene su jugabilidad y aprovecha el 3D y la mayor potencia de la consola Nintendo64 para mejorar la experiencia de juego.

Concepto y género

El videojuego será de género arcade sobre raíles, en el que el jugador controla una nave espacial que debe recorrer una serie de escenarios destruyendo a los enemigos que se encuentra en el camino, recogiendo power-ups⁵³ e intentando obtener la mayor puntuación en cada una de las fases.

Audiencia

Este juego va dirigido a los usuarios poseedores de dispositivos iOS y AppleTV, a aquellos que disfruten con juegos de acción y de partidas cortas que supongan un reto y diversión. Aun así, la idea inicial es que el juego sea atractivo para el máximo número de personas posible. Para ello se intentará centrar el valor del juego en las mecánicas jugables y un aspecto del juego carismático para atraer a distintos perfiles de jugador.

Características

Jugabilidad y mecánicas

El juego es para un único jugador y, se controla a través del mando de AppleTV “Siri”. Tras empezar un nivel el usuario manejará la nave desplazándola en los ejes horizontal y vertical. La nave podrá disparar para

⁵³ Un power-up [245] suele ser un ítem o característica especial que otorga un beneficio al jugador, ya sea constante o de manera temporal.

derrotar a los enemigos que se aproximen. El jugador tendrá que controlar la dirección a la que apuntan sus armas usando el mando para mover una cruceta, que hace de apuntador, a lo largo de la pantalla.

Cada nivel cuenta con un sistema de puntuación que se irá incrementando según se vayan derrotando enemigos. Uno de los objetivos del juego es conseguir la mayor puntuación posible dentro de cada uno de los niveles. Las mejores puntuaciones se guardarán y mostrarán para cada nivel.

Los enemigos del juego podrán dañar al jugador mediante distintos patrones de ataques con sus armas. Para resistir estos ataques la nave del jugador dispone de un escudo y la propia resistencia de la nave que suponen el daño que puede soportar antes de ser destruida. Si la nave perdiere todos los puntos de escudo entonces empezaría a perder sus puntos de vida. Si el jugador se queda sin puntos de vida se pierde el nivel y se tendrá que repetir otra vez.

A lo largo de cada nivel, aparte de encontrarte con multitud de enemigos, aparecerán una serie de power-ups que permiten recuperar tanto puntos de vida como puntos de escudo.

Aspecto visual

La apariencia de un juego es muy importante para llamar la atención de los usuarios y de cara a dotarle de su propia personalidad [172].

Existen distintas formas de representar el estilo gráfico de los juegos [173], entre ellos el que predomina en la actualidad y el que se ha decidido utilizar para este juego es la vista en 3D [174].

El aspecto visual de cualquier juego, sea 2D o 3D, viene definido por dos elementos principales.

El primero son los modelos o representación de los elementos del juego. Estos forman el aspecto y les otorgan forma, dotándoles de la apariencia que los identifican. Estas formas pueden ser más realistas y pulidas o más abstractas y fantásticas.

El segundo elemento característico lo definen las texturas y materiales que son aplicados en los modelos. Son los encargados de dar color y matiz. Los materiales y texturas aportan el detalle y acabado final del aspecto visual, ya sea buscando el realismo, o un estilo único y diferente.

Existen muchas combinaciones y acabados visuales, así que elegir el aspecto adecuado para tu juego puede ser difícil, y eso sin contar con la dificultad que puede aportar el conseguir dicho aspecto y que acabe dando la sensación contraria a lo que se buscaba [175]. Aun así, el abanico de posibilidades es enorme y se puede conseguir dotar a los juegos de mucha personalidad solo con su apariencia visual [176] [177].

Para Vega: Factions War se ha optado por un estilo simple pero colorido. Que se encuentre entre un aspecto realista y un estilo algo más cartoon. Al ser un juego de temática sci-fi, se ha tenido en cuenta a la hora de crear los modelos y elegir los materiales, así como los efectos y texturas⁵⁴.

Ha habido dos factores principales a la hora de elegir como sería el acabado final.

El primero consiste en las propias circunstancias del proyecto. Hay un tiempo limitado, el peso del juego no puede superar unos límites y el conocimiento a la hora de realizar los modelos en 3D y texturizado es reducido.

⁵⁴ Todas las texturas son imágenes de libre uso obtenidas de esta página [215].

El segundo es la identidad y el aspecto que se tenía en mente del juego. Formas reconocibles pero que tengan el suficiente detalle para dar profundidad a los modelos.

Modelos

Los modelos se inspiran en figuras geométricas básicas sencillas, pero cada una de ellas tiene un mínimo de detalle para dotarles de más personalidad. A pesar de ser un juego de aspecto futurista, se han buscado formas y figuras reconocibles para todos los modelos, pero añadiendo toques sci-fi.

Materiales y texturas

Para reducir el peso final de la aplicación se ha intentado usar el mínimo número de texturas posibles y utilizar materiales con colores base para los modelos. A la hora de diferenciar los distintos elementos del juego, se ha seguido un criterio especial para los materiales y texturas, que, además, acaban formando parte de las características visuales del videojuego.

El escenario y sus elementos cuentan con texturas básicas con un efecto estilo pincel. El jugador y el resto de elementos con los que interactúa, como los enemigos y power-ups, usan materiales con colores base sin textura.

El resultado final dota a Vega: Factions War de un estilo visual característico, sencillo pero que llama lo suficiente la atención.



Ilustración 37 – Aspecto visual de Vega: *Factions War*

Historia

Tras siglos de conflictos, la humanidad se unificó para conquistar su sueño, la colonización espacial. La Unión, el nombre que se otorgó a la humanidad unificada, empezó colonizando el sistema solar y los sistemas solares vecinos como Procyon y Altair. Durante la última década la expansión se centró Vega, el último sistema en ser colonizado. Pero, para entonces, los conflictos que se habían detenido durante los inicios de la colonización espacial empezaron a resurgir y, con el tiempo, el descontento provocado por los problemas derivados de la exploración espacial, el uso y gestión de recursos y la diferencia en ideologías, han hecho que se sobrepasará el límite. Se desencadenó lo inevitable: Guerra.

En lugar de la Unión, ahora la humanidad se ha fragmentado en facciones que luchan por el control. Los conflictos han estallado en varios sectores...

Como miembro de lo que queda de la flota de la Unión, estás dispuesto a luchar por defender lo que queda de ella. Te encuentras en el centro del foco de los conflictos actuales, **Vega**. Sigue sus órdenes y acaba con sus enemigos para quizás, algún día, terminar con todos estos conflictos...

Limitaciones

Debido a las características del proyecto y a los elementos mencionados en la sección de requisitos, se ha decidido tener en cuenta una serie de limitaciones en la creación del videojuego. El desarrollo del proyecto se centrará en tener la base del producto terminada, con su jugabilidad y mecánicas definidas, así como su aspecto visual terminado con un mínimo nivel de calidad, pero sin llegar a altos niveles de producción. Esto se traduce en crear un juego más sencillo, pero con sus aspectos jugables planteados completos.

Se buscará la forma de que la aplicación pese lo menos posible, y para conseguir esto se tendrá que usar modelos sencillos con pocos polígonos, efectos visuales simples y utilizar pocas texturas. Esto repercutirá en el aspecto final del juego, menos vistoso y con menos personalidad.

El número de fases se reducirá al mínimo posible, teniendo en cuenta el tiempo de producción y desarrollo limitado⁵⁵. El videojuego, por tanto, será corto y no tendrá mucho contenido ni opciones adicionales de base.

Con respecto a las limitaciones de control se ha tomado la decisión de usar controles simplificados con respecto a otros juegos del género. Se puede desplazar la nave moviendo el mando con el control de movimiento, mientras que el apuntador usará el control táctil. Se había planteado inicialmente incluir más acciones, pero finalmente se ha optado por un control más simplificado.

Por un lado, acciones extra podrían incrementar la dificultad del juego de una manera artificial y menos fluida. Por otro lado, las propias limitaciones

⁵⁵ Tal y como se ha decidido planificar el proyecto, una vez realizada la entrega del Trabajo final de grado, la base jugable estaría terminada con lo que el contenido del juego podría ser ampliado mucho más rápido.

del mando harían que la implementación de estas nuevas acciones, aunque posibles, complicarán el manejo dando una sensación de control más forzada.

Estudio de mercado y viabilidad

AppleTV empieza a contar con una gran cantidad y variedad de títulos a pesar de no terminar de convencer como plataforma de videojuegos. Como ya se ha mencionado, muchos son ports de otras plataformas, pero también hay juegos desarrollados inicialmente para AppleTV.

Empieza a existir variedad de títulos, pero aún no tienen la suficiente cantidad como para saturar el mercado como ocurre con los otros dispositivos con sistema iOS o Android⁵⁶.

Por lo tanto, sacar cualquier producto con un cierto nivel de calidad es viable. Si el videojuego es entretenido, está pulido y con un buen acabado final, las posibilidades de que la gente lo quiera probar son altas. Aun no existe un mar de videojuegos en la aplicación y se puede aprovechar la situación para que los nuevos videojuegos tengan visibilidad⁵⁷. Es una de las razones por las que se eligió AppleTV como plataforma destino del videojuego, el nuevo mercado que ofrece.

⁵⁶

⁵⁷ Cuando existen muchas aplicaciones y se crean miles de ellas nuevas cada día, encontrar una aplicación entre todas es difícil.

Como se juega a Vega: Factions War

Tras empezar el videojuego en la pantalla de título, dónde se muestra una introducción histórica del mismo, el usuario puede elegir acceder a distintos menús.

Desde ver los créditos del juego, que muestra diversa información del título, a cambiar y configurar las opciones del juego.

También puede acceder al menú de selección de niveles dónde se muestra una lista de los niveles desbloqueados, así como sus mejores puntuaciones. Los niveles se irán desbloqueando al completar las distintas fases.

Una vez seleccionada una fase empieza la partida. El jugador deberá sortear los ataques de los enemigos para no sufrir daños mientras destruye al mayor número posible de ellos.

A lo largo del recorrido el jugador podrá activar puntos de guardado o checkpoints⁵⁸, que permitirán empezar el nivel desde ese punto si es derrotado en lugar de tener que empezarlo desde el principio.

Una vez alcanzado el final del nivel se tendrá que enfrentar a un boss final que deberá derrotar para completar el nivel con éxito.

Con el jefe final derrotado, se pasará a la pantalla de puntuación que mostrará la puntuación obtenida en ese recorrido.

⁵⁸ Un Checkpoint es un punto a lo largo de un nivel en el cual, se guardan los datos y estado del jugador para poder volver a ese momento concreto del juego. Se suele usar para guardar el progreso del jugador y poder cargarlo cuando pierde una partida. Así, el contenido superado por el jugador no se tiene que volver a repetir si pierde la partida.

Al salir de la pantalla de puntuación se volverá a la pantalla de selección de niveles. Si la puntuación obtenida entra dentro de las tres mejores está se guardará y se mostrará en los datos del nivel.

Funcionalidades y casos de uso

Analizar un proyecto, sus funcionalidades y sus características es uno de los pasos más importantes del desarrollo.

Primero, permite definir los elementos principales del mismo y organizar la planificación del proyecto.

Segundo, analiza el proyecto de una manera más técnica, pudiendo observar su complejidad y detectando si algún elemento no está lo suficientemente bien definido.

Este proyecto va a seguir una metodología ágil de desarrollo rápido e iterativo, con lo que un análisis en profundidad es contraproducente. Por tanto, se van a analizar los requisitos funcionales, los no funcionales y los diagramas de casos de uso. Son los más importantes para definir el proyecto y ayudarán en su desarrollo.

A cada uno de los requisitos vamos a otorgarle un identificador y un nivel de prioridad de 1 a 5. La prioridad representa la importancia en el desarrollo del proyecto y cuánto podría afectar alguna complicación en la implantación de ese requisito. Un 1 equivale a una prioridad baja, mientras que 5 indica prioridad crítica.

Requisitos funcionales

Los requisitos funcionales representan una función que debe cumplir el proyecto. Indican las cosas que el usuario podría o no podría hacer.

Tabla 3 RFI

Identificador	RF1
Nombre	Ejecutar la aplicación Vega: Factions War
Descripción	Se puede ejecutar la aplicación en el dispositivo instalado. Al ejecutarse se cargan todos los recursos necesarios.
Prioridad	5

Tabla 4 RF2

Identificador	RF2
Nombre	Carga de un nivel
Descripción	Al seleccionar un nivel la aplicación deberá cargar y ejecutar el nivel seleccionado.
Prioridad	5

Tabla 5 RF3

Identificador	RF3
Nombre	Sistema de puntuación
Descripción	La aplicación contará con un sistema de puntuaciones para los niveles, guardando las 3 mejores puntuaciones obtenidas en cada nivel.
Prioridad	3

Tabla 6 RF4

Identificador	RF4
Nombre	Mostrar puntuación durante una partida
Descripción	La aplicación mostrará y actualizará en la interfaz del juego la puntuación que lleva acumulada el usuario en ese momento de la partida
Prioridad	4

Tabla 7 RF5

Identificador	RF5
Nombre	Mostrar puntuaciones en la selección de nivel
Descripción	En el menú de selección de niveles se mostrará las 3 mejores puntuaciones de cada nivel guardadas previamente.
Prioridad	3

Tabla 8 RF6

Identificador	RF6
Nombre	Control de movimiento de la nave

Descripción	El usuario debe poder controlar la nave del juego mediante el control remoto de AppleTV “siri”.
Prioridad	5

Tabla 9 RF7

Identificador	RF7
Nombre	Detección de los inputs del controlador remoto
Descripción	La aplicación deberá detectar y reconocer las pulsaciones de botones y el control táctil del control remoto de AppleTV “siri”.
Prioridad	5

Tabla 10 RF8

Identificador	RF8
Nombre	Detección del control por movimiento del controlador remoto
Descripción	La aplicación deberá detectar y reconocer el giroscopio y acelerómetro del control remoto de AppleTV “siri”.
Prioridad	5

Tabla 11 RF9

Identificador	RF9
Nombre	Muerte del jugador
Descripción	Al perder todos los puntos de vida el jugador debe perder esa partida y mostrar una pantalla de derrota.
Prioridad	5

Tabla 12 RF10

Identificador	RF10
Nombre	Sistema de escudos de la nave del jugador
Descripción	La nave del jugador tiene que tener un sistema de escudos que le protegen de los ataques enemigos. Según recibe ataques los puntos del escudo bajan. Si no recibe daños se recuperan los puntos con el tiempo.
Prioridad	5

Tabla 13 RF11

Identificador	RF11
Nombre	Sistema de vida de la nave del jugador
Descripción	La nave del jugador tiene que tener un sistema de puntos de vida que le indican cuantos ataques puede recibir del enemigo antes de ser derrotado. Al recibir un ataque

	pierde puntos de vida. Estos puntos de vida son fijos y no pueden aumentar de manera normal.
Prioridad	5

Tabla 14 RF12

Identificador	RF12
Nombre	Derrotar enemigos
Descripción	Mientras recorre un nivel, el usuario podrá atacar a los enemigos que aparezcan y derrotarlos quitándoles todos sus puntos de vida. Derrotar un enemigo otorga puntos al jugador.
Prioridad	5

Tabla 15 RF13

Identificador	RF13
Nombre	Derrotar a los bosses
Descripción	Al final de un nivel el usuario tiene que atacar y derrotar a un boss quitándole todos sus puntos de vida. Derrotar a un boss finalizaría el nivel y cargaría la pantalla de puntuación del mismo.
Prioridad	5

Tabla 16 RF14

Identificador	RF14
Nombre	Pantalla fin de nivel
Descripción	Al terminar un nivel la aplicación deberá mostrar una pantalla de final de nivel en el que se indique la puntuación final obtenida.
Prioridad	5

Tabla 17 RF15

Identificador	RF15
Nombre	Checkpoints
Descripción	Cuando un jugador llegue a ciertos puntos de un nivel desbloqueará un checkpoint. Cada vez que reinicie el nivel tras ser derrotado empezará el nivel desde ese punto con la puntuación que tenía en ese momento.
Prioridad	5

Tabla 18 RF16

Identificador	RF16
Nombre	Desbloquear niveles
Descripción	Al completar un nivel la aplicación deberá desbloquear el siguiente nivel jugable en el menú de selección de niveles.
Prioridad	5

Tabla 19 RF17

Identificador	RF17
Nombre	Items del juego
Descripción	Los niveles tendrán 3 tipos de ítems que el jugador puede recoger. Uno que recupera vida, otro que recupera escudos y otro que aumenta la puntuación del nivel.
Prioridad	5

Tabla 20 RF18

Identificador	RF18
Nombre	Recoger items
Descripción	Al entrar en contacto la nave del usuario con un ítem deberá recogerlo y otorgarle los bonos que le corresponden según el tipo de ítem que era.
Prioridad	5

Tabla 21 RF19

Identificador	RF19
Nombre	Colisiones con objetos y enemigos

Descripción	La nave del jugador, al entrar en contacto con algún elemento del escenario o un enemigo, deberá colisionar y recibir puntos de daño.
Prioridad	5

Tabla 22 RF20

Identificador	RF20
Nombre	Interfaz del juego
Descripción	Durante el juego, se deberá mostrar una interfaz o HUD que se irá actualizando mostrando la información de la nave del jugador y la puntuación actual del nivel.
Prioridad	5

Tabla 23 RF21

Identificador	RF21
Nombre	Menú de pausa
Descripción	Al pulsar la tecla del menú de pausa el videojuego deberá mostrar el menú de pausa con todas sus opciones y detener la ejecución del juego hasta que se reanude el juego.
Prioridad	5

Tabla 24 RF22

Identificador	RF22
Nombre	Menú de pausa. Salir al menú principal
Descripción	Desde el menú de pausa se debe poder acceder al menú principal mediante un botón. Al hacer esto se perderá el progreso del nivel.
Prioridad	5

Tabla 25 RF23

Identificador	RF23
Nombre	Menú de pausa. Salir del juego
Descripción	Desde el menú de pausa se debe poder salir de la aplicación mediante un botón.
Prioridad	5

Tabla 26 RF24

Identificador	RF24
Nombre	Menú de pausa. Reiniciar nivel.
Descripción	Desde el menú de pausa se debe poder reiniciar el nivel desde un botón.
Prioridad	5

Tabla 27 RF25

Identificador	RF25
Nombre	Menú de pausa. Continuar nivel.
Descripción	Desde el menú de pausa se debe poder reanudar la partida pausada mediante un botón.
Prioridad	5

Tabla 28 RF26

Identificador	RF26
Nombre	Transición de menús
Descripción	La aplicación permite desplazarse y navegar por los distintos menús del juego.
Prioridad	5

Tabla 29 RF27

Identificador	RF27
Nombre	Menú principal
Descripción	La aplicación debe tener un menú principal desde el que se puede acceder a todas las funcionalidades del videojuego. Deberá poder acceder al menú de selección de niveles, al menú de opciones y al menú de créditos.
Prioridad	5

Tabla 30 RF28

Identificador	RF28
Nombre	Menú principal. Salir del juego.
Descripción	Desde el menú principal se debe poder salir de la aplicación mediante un botón.
Prioridad	5

Tabla 31 RF29

Identificador	RF29
Nombre	Transiciones
Descripción	Desde cualquier menú la aplicación deberá poder cambiar a otro menú mediante una transición.
Prioridad	3

Tabla 32 RF30

Identificador	RF30
Nombre	Menú opciones
Descripción	La aplicación deberá tener un menú donde poder cambiar la configuración del juego.
Prioridad	5

Tabla 33 RF31

Identificador	RF31
Nombre	Menú créditos
Descripción	La aplicación deberá tener un menú donde se muestren los créditos del juego.
Prioridad	5

Tabla 34 RF32

Identificador	RF32
Nombre	Menú selección de niveles
Descripción	La aplicación deberá tener un menú donde se muestran los niveles disponibles.
Prioridad	5

Tabla 35 RF33

Identificador	RF33
Nombre	Ejecutable en AppleTV
Descripción	Se deberá realizar un ejecutable del juego que funcione en appleTV.
Prioridad	5

Tabla 36 RF34

Identificador	RF34
Nombre	Memoria del proyecto
Descripción	Se deberá redactar una memoria del trabajo final de grado para que se pueda evaluar el trabajo realizado.
Prioridad	5

Requisitos no funcionales

Son las características que debería cumplir el proyecto. No tanto como funciones sino aspecto técnicos y propiedades del mismo.

Tabla 37 RNF35

Identificador	RNF35
Nombre	Usabilidad
Descripción	La aplicación debe ser fácil de usar y facilitar su uso al jugador mediante datos en la interfaz.
Prioridad	3

Tabla 38 RNF36

Identificador	RNF36
Nombre	Accesibilidad
Descripción	La aplicación debe ser lo más accesible posible para que la puedan usar el mayor número de personas posibles.
Prioridad	2

Tabla 39 RNF37

Identificador	RNF37
Nombre	Rendimiento y funcionamiento
Descripción	La aplicación debe funcionar correctamente y con un rendimiento adecuado. Debe mantenerse estable y no tener errores durante su ejecución.
Prioridad	5

Tabla 40 RNF38

Identificador	RNF38
Nombre	Tamaño de la aplicación
Descripción	El tamaño de la aplicación debe ser lo menor posible para facilitar la descarga y la aceptación de la misma en la App Store.
Prioridad	5

Tabla 41 RNF39

Identificador	RNF39
Nombre	Seguridad del uso de la aplicación
Descripción	La aplicación debe ser segura al ser usada y no provocar ningún malfuncionamiento en los dispositivos con los que interactúa.
Prioridad	5

Casos de uso

Los casos de uso sirven para especificar cómo se usará la aplicación por parte del usuario y cuál debería ser su funcionamiento. Los casos de uso son necesarios para explicar la interacción de la aplicación con el usuario, tanto para facilitar el trabajo y la compresión de los requisitos funcionales y la interacción a la hora del desarrollo del producto, como para cuando el producto esté en el mercado.

El producto final es un videojuego, así que la estructura de los esquemas será sencilla. Primero vamos a explicar en qué consiste.

Diagrama

Un caso de uso cuenta con un actor, el que interactúa con el sistema (en este caso el usuario final del producto) y el que inicia el caso de uso, y una situación o escenario, que puede estar formada por otros casos de uso. Para la aplicación desarrollada, las situaciones/eventos son los siguientes:

Usar Aplicación - Diagrama de Casos de Uso general. El resto de situaciones están contenidas en el caso de uso “Usar aplicación”.

El diagrama de casos de uso es el siguiente:

Casos de uso, “usar aplicación”:

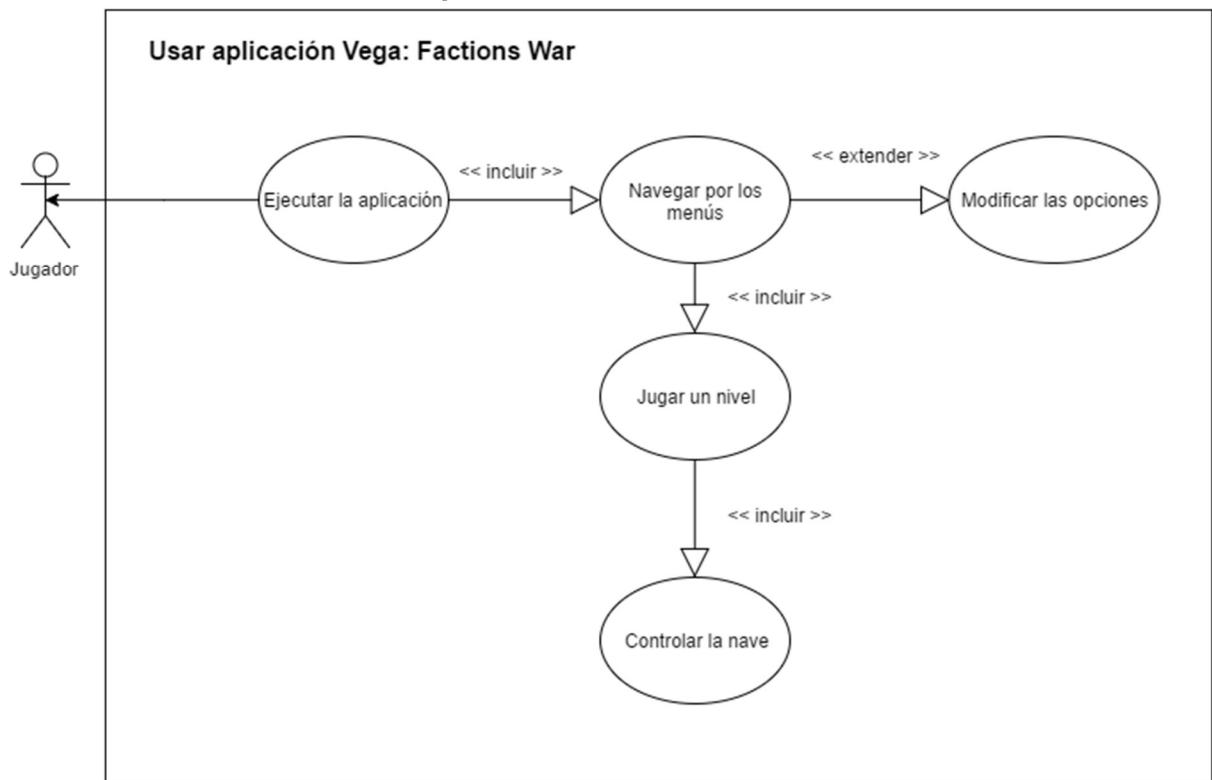


Ilustración 38 – Diagrama de casos de uso de la aplicación Vega:Factions War

Descripción

El formato de la descripción del diagrama va a ser como se menciona a continuación:

Caso de uso: Título o nombre del caso de uso

Actor: El actor que activa o interactúa con el caso de uso

Descripción: Explicación breve e informal del caso de uso.

Precondición y postcondición: Si el caso de uso tiene un paso anterior o un resultado final se mostrarán en este apartado.

Secuencia de flujo: Si el caso de uso cuenta con múltiples partes o se necesita ampliar el apartado de descripción para detallar el caso de uso se incluye este apartado.

Tabla 42 Caso de uso, ejecutar la aplicación

Caso de uso	Ejecutar la aplicación
Actor	Jugador/Usuario
Descripción	Se ejecuta la aplicación en el dispositivo instalado y al ser cargado con éxito permite que se pueda jugar al juego.
Precondición Postcondición	<ul style="list-style-type: none"> - Instalación del juego en el dispositivo. - Juego ejecutado y funcionando en el dispositivo

Tabla 43 Caso de uso, navegar por los menús

Caso de uso	Navegar por los menús
Actor	Jugador/Usuario
Descripción	Interactuar con la aplicación cambiando entre las distintas pantallas que te permiten acceder a las

	distintas partes del juego, como las opciones o la selección de los niveles.
Precondición Postcondición	<ul style="list-style-type: none"> - Que la aplicación se haya ejecutado. - Cambio de una pantalla del juego a otra.
Secuencia de flujo	Desde un menú se puede acceder a otros menús pulsando en el botón correspondiente. Desde cualquier menú que no sea el principal, existe un botón que permite volver al menú principal.

Tabla 44 Caso de uso, modificar las opciones

Caso de uso	Modificar las opciones
Actor	Jugador/Usuario
Descripción	Se cambian los valores de las opciones que ajustan el control de la nave del jugador al gusto del usuario.
Precondición Postcondición	<ul style="list-style-type: none"> - Navegación al menú de opciones. - Guardado de la nueva configuración de los datos y cambios en la forma de controlar la nave del jugador.
Secuencia de flujo	Al entrar en el menú de opciones se pueden cambiar y modificar las características del juego

	mediante elementos interactivos de ese menú. Para guardar los cambios se tiene que pulsar en el botón de guardar que aparece en el menú.
--	--

Tabla 45 Caso de uso, jugar un nivel

Caso de uso	Jugar un nivel
Actor	Jugador/Usuario
Descripción	El jugador juega un nivel, eliminando enemigos, recogiendo objetos y derrotando al enemigo final del nivel para completarlo.
Precondición Postcondición	<ul style="list-style-type: none"> - Desbloquear el nivel. - Guardado de la puntuación y desbloqueo del siguiente nivel.

Tabla 46 Caso de uso, controlar la nave

Caso de uso	Controlar la nave
Actor	Jugador/Usuario
Descripción	El usuario usa el dispositivo controlador a su disposición (siri) para mover la nave del jugador por el entorno e interactuar con el nivel.

Precondición Postcondición	<ul style="list-style-type: none"> - Un nivel ejecutándose y acceso a un dispositivo controlador.
Secuencia de flujo	<p>Mediante el giroscopio se detectan los movimientos de la nave. Con la superficie táctil del mando se mueve el apuntador para dirigir los disparos del jugador.</p>

DESARROLLO

En este apartado se hablará del proceso de desarrollo que se ha seguido, los conceptos e ideas detrás del videojuego y explicar la manera en que se ha trabajado en cada una de las partes del mismo.

Metodología de trabajo

Ya hemos hablado de las metodologías de trabajo y de los diferentes tipos que existen. Ahora, se explicará la metodología escogida para este proyecto.

Como se ha mencionado con anterioridad, para afrontar este trabajo se ha tenido en cuenta el tamaño del grupo de trabajo, el tiempo de desarrollo del que se dispone y el tipo de producto que se va a hacer.

El trabajo consiste en crear un videojuego a pequeña escala, la experiencia que se posee en este tipo de trabajos es muy baja y el tiempo es bastante ajustado, con lo que la mejor opción son las metodologías ágiles [178], que son las más utilizadas para el desarrollo de videojuegos en la actualidad. Al usar este tipo de metodologías, si surge algún problema o retraso con el proyecto, se puede realizar un ajuste de planificación y objetivos, reestructurando el proyecto para completar con éxito el producto.

Al usar períodos de iteración cortos junto con la elaboración de prototipos en cada uno de ellos, la comprobación del estado del producto es constante. De entre todas las metodologías ágiles se ha decidido usar Scrum [179] como base para organizar el desarrollo del proyecto.

Las iteraciones de trabajo o Sprints, usando el término específico para Scrum, se han definido entre una y dos semanas, dependiendo de la funcionalidad a implementar.

Aprovechando el sistema de tutorización del trabajo final de grado, se han interpretado las reuniones con el profesorado de dos maneras. Primero, como la supervisión por parte del cliente del estado del producto, y segundo, como las reuniones al final de un sprint de cada iteración.

En estas reuniones se ha enseñado y evaluado el estado del videojuego, y posteriormente se ha planificado la siguiente iteración teniendo en cuenta los resultados obtenidos.

Aparte de estas reuniones de tutorización, se han realizado Sprints y evaluaciones por parte del alumno para llevar un mejor control del proyecto.

Para la gestión y asignación de tareas se ha utilizado la aplicación Trello, que debido a su funcionamiento parecido al de un tablero equivale al tablero de trabajo de la metodología Scrum.

En este tablero se han creado cada una de las tareas a realizar divididas por componentes del juego. A cada tarea se ha asignado una lista de sub-tareas que representan cada una de las funcionalidades del videojuego.

Según se han ido completando las tareas se han añadido y asignado nuevas mientras se revisaba el estado del proyecto, para mantener un flujo constante de trabajo.

Unity3D

Ya hemos mencionado con anterioridad que es Unity, un motor de desarrollo de videojuegos. Ahora vamos a entrar en detalle en sus funcionalidades y en cómo se puede usar, así como hablar de cómo se ha aprovechado para el desarrollo del proyecto. La herramienta se centra en dos bases fundamentales.

La primera, la portabilidad⁵⁹. Unity permite exportar el videojuego a distintas plataformas usando para ello diversas interfaces de renderizado, que funcionan como una API⁶⁰ de desarrollo. Puede usar Direct3D, nativa del sistema operativo Windows [180] y que se usa para ordenadores Windows y su videoconsola de sobremesa, Xbox360 y XboxOne. También puede usar OpenGL [181] y las APIs de desarrollo específicas para cada consola consolas. Usando Unity, por tanto, permite total libertad de desarrollo para cualquier tipo de plataforma, con lo que con unos pequeños cambios y ajustes se puede tener el videojuego funcionando en un iPad, en un ordenador o en una consola.

La segunda, la accesibilidad y la facilidad de uso. Una de las grandes ventajas que aporta Unity es la interfaz de desarrollo, que permite tener un videojuego funcional en poco tiempo. También es una de las grandes desventajas que aporta, según como se mire. Unity está preparado para trabajar de una forma específica, con lo que si se necesita trabajar o implementar algo de manera distinta se pierde la ventaja que aporta. Una forma de solventar el problema es trabajar y organizar el desarrollo teniendo esto en cuenta desde el

⁵⁹ La portabilidad en este caso es la capacidad de Unity de realizar automáticamente un port del juego diseñado a numerosas plataformas.

⁶⁰ API [207] consiste en una serie de herramientas predefinidas que ayudan en el desarrollo de software. Básicamente funciona como una interfaz de uso y desarrollo entre el lenguaje de programación y el desarrollador, abstrayendo un nivel la implementación de bajo nivel del software, reduciendo la dificultad del trabajo.

principio, centrándose más en la base del código que se está creando y usando el código fuente⁶¹ de Unity, aunque este es poco accesible.

Al contar con experiencias previa con Unity, los problemas con respecto a lo que se ha mencionado, que los ha habidom se han podido solventar o evitar en mayor o menor medida.

Funcionamiento

En este apartado vamos a explicar cómo se trabaja con Unity y en qué consiste su interfaz.

El concepto de GameObject y entidad-componente

Unity funciona con una estructura centrada en GameObjects y sus componentes, basándose en la programación orientada a objetos [182] y la relación entidad-componente.

Vamos a explicar brevemente en qué consiste la programación orientada a objetos para que sea más fácil entender qué es la entidad-componente.

Programación orientada a objetos (OOP)

Es una filosofía de diseño a la hora de programar que se basa en definir objetos. Los objetos contienen datos, atributos y acciones que pueden realizar. Estos objetos se pueden relacionar unos con otros creando una estructura jerárquica. Un ejemplo:

- Un coche es un objeto que está formado por otros objetos, las puertas, las ruedas, el motor, etc. Cada uno de los objetos tiene sus características

⁶¹ El código fuente [236] es la base del funcionamiento de un programa. Son un conjunto líneas de código escritas en un lenguaje de programación que determinan como funciona y como se ejecuta dicho programa.

y funciones, siguiendo el ejemplo, las puertas se abren y el motor se enciende. El objeto que contiene a todos, el objeto padre, tiene como atributos al resto de objetos. Así, el coche, aparte de sus propias características como el color, tendría también cuatro ruedas, un motor, y cinco puertas.

Con este ejemplo explicamos que, en la programación orientada a objetos, los objetos pueden contener a otros objetos y utilizarlos como si fuesen sus atributos.

Para más información sobre la programación orientada a objetos véase las siguientes referencias [183] [184] [185].

Entidad-componente

Una vez entendido el concepto básico de la OOP, pasamos a explicar el otro concepto importante para entender el funcionamiento de Unity, la entidad-componente.

El sistema de entidad-componente [186] [187] (ECS), es un patrón de arquitectura de software usado sobretodo en el desarrollo de videojuegos. Se basa en favorecer la composición de objetos [188] sobre la herencia [189]. Siguiendo el ejemplo del coche, en lugar de tener objetos padre y objetos hijos, tendríamos varios componentes o piezas, que juntos formarían el coche. Las ruedas, puerta y motor, cada uno de ellos es un objeto independiente, pero si los juntas todos formarían el coche. Este coche sería la entidad formada por el resto de componentes, y quitando y añadiendo otros componentes cambiaría el concepto de esta entidad. La diferencia entre OOP y ECS es el acoplamiento de los objetos⁶², ya que en ECS los objetos son más independientes, cumpliendo

⁶² El acoplamiento [238] [239] en informática hace referencia a la dependencia que tienen las partes del código unas con otros. Un bajo acoplamiento implica que son más independientes a nivel estructural y que

funciones más específicas, consiguiendo así más modularidad⁶³ en la estructura del proyecto. Cada entidad en ECS es un objeto en sí mismo, y los componentes [190] las propiedades de ese objeto.

Este sistema tiene sus ventajas, pero también sus inconvenientes, ¿Cómo se relacionan los objetos entre sí? La comunicación entre objetos y componentes se hace más compleja al estar todo más desacoplado, y en general hay que cuidar más la estructura del código para evitar problemas de rendimiento. Y Unity no se libra de estos problemas [191].

En Unity las entidades que formarían los objetos serían los GameObjects.

GameObjects

Los GameObjects [192] [193] son la pieza fundamental con la que Unity te permite crear videojuegos. Un GameObject es la base para representar a los personajes, las partes del escenario y cualquier elemento del videojuego. Pero de partida, un GameObject solo está formado por un componente Transform⁶⁴, que es innato y no se puede eliminar de ninguna manera. Así que, si un GameObject solo es un conjunto de vectores, ¿Cómo se utiliza para conseguir el resto de elementos del videojuego?

Un GameObject es un recipiente o contenedor para componentes. Por si mismos no hacen nada, pero al ir añadiéndole componentes estos son los que incluyen las funcionalidades al objeto y los que lo dotan de sus propiedades. Al

no necesita parte de otro código para funcionar. Ejemplo, una función que realiza la suma de dos números que se le pasan como parámetro tiene bajo acoplamiento. Sin embargo, si esa función necesita que otras funciones realicen parte de los cálculos, entonces están altamente acopladas unas funciones con otras.

⁶³ La programación modular [240] consiste en dividir el código o programa en módulos o subprogramas, haciendo que sea más sencillo de trabajar con él, y más legible. El acoplamiento junto con la programación modular, mejoran el diseño del código de un programa, y permiten, si están bien ejecutados, la reutilización del código.

⁶⁴ El componente Transform define la posición, rotación y escala del GameObject en la escena. En Unity es representado en la escena con la figura de los ejes de coordenadas XYZ [237].

ir añadiendo componentes, el GameObject se puede transformar en un personaje, un efecto especial o un efecto sonoro. Es como si fuese un puzzle, por sí misma una pieza no tiene mucho valor, pero juntando muchas obtienes una imagen.

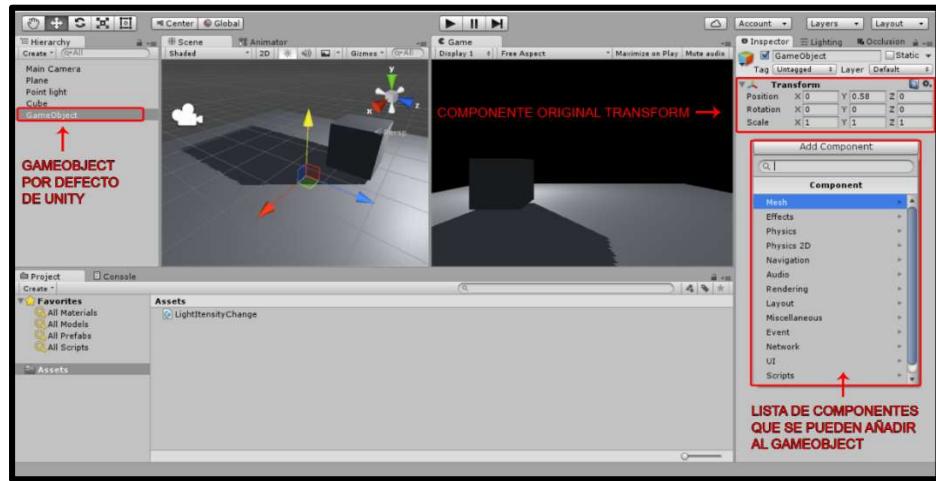


Ilustración 39 – Unity, creación de un GameObject y como añadir nuevos componentes

Por ejemplo, para crear una luz dentro de Unity, a un GameObject hay que añadirle el componente **Light**⁶⁵. Si queremos que, a su vez, la luz emita un sonido, como un chisporroteo, le añadiremos el componente **AudioSource** y en él podremos colocar el archivo de sonido.

Junto con los componentes, los GameObjects pueden crear todos los elementos del juego, desde personajes e interfaces⁶⁶ hasta la propia lógica del juego.

Unity tiene por defecto un conjunto de combinaciones de GameObjects y componentes que se usan como base, pero el usuario puede crear y componer los GameObjects como le interese.

⁶⁵ Los diversos componentes de los que dispone Unity permiten añadir funcionalidades a los GameObjects, desde emitir luz hasta formar parte de un sistema de físicas con gravedad incluida dentro del juego.

⁶⁶ Las interfaces [249] son elementos visuales en los juegos que se usan para mostrar datos al jugador.

Uno de los componentes más importantes que se le puede añadir a un GameObject es el de tipo **Script**. El componente Script vincula código de programación al objeto, permitiendo, siguiendo con el ejemplo de la luz, que la intensidad de la luz parpadee cuando se ejecuta el juego.

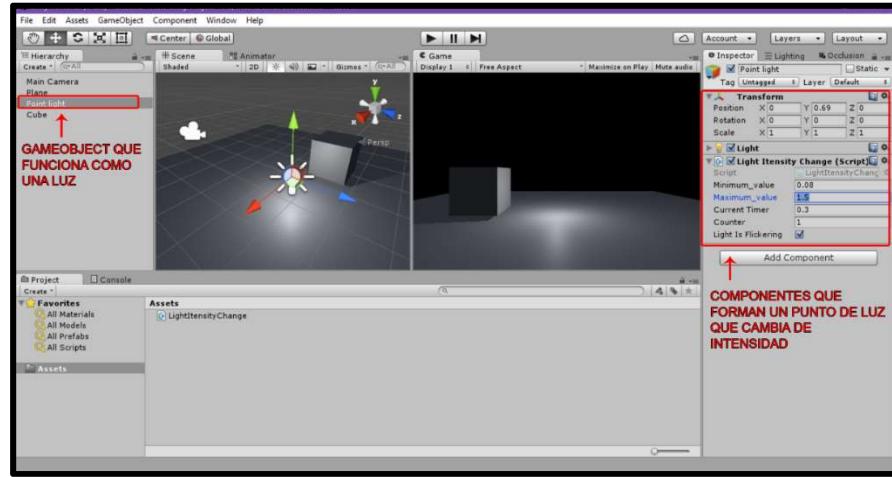


Ilustración 40 – Unity, ejemplo de un GameObject que simula un punto de luz y sus componentes

La interfaz

En Unity, aparte de contar con la barra de herramientas o *toolbar* personalizada, su interfaz está dividida en cinco secciones principales o ventanas. La ventana del proyecto, el inspector, la ventana del editor, la ventana del juego y el *hierarchy*. Todas ellas pueden cambiar de tamaño y de posición, además de contar con otras opciones de personalización, para poder adaptar la disposición de la interfaz al gusto del usuario y de la tarea que se encuentre realizando en ese momento.

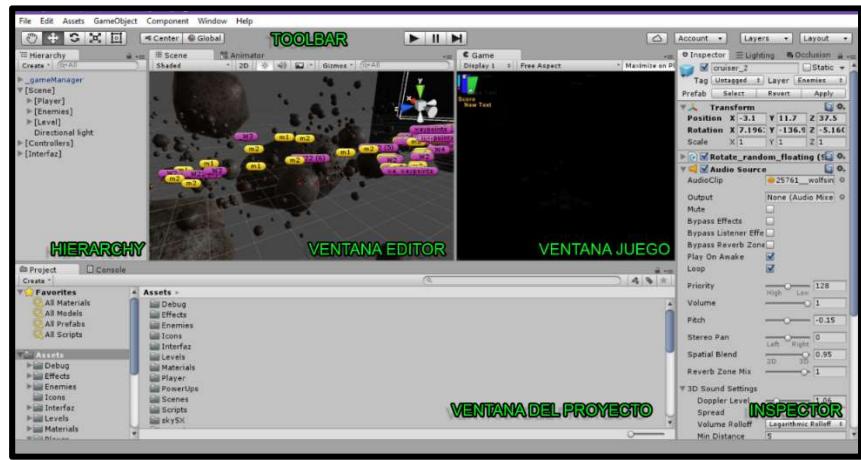


Ilustración 41 – Interfaz Unity, vista general

Para más información y referencias sobre la interfaz del editor de Unity, visite la web de la documentación oficial de Unity [194].

Ventana del proyecto

En la ventana del proyecto es dónde se pueden ver y acceder a todos los recursos o *assets*⁶⁷ que tiene el proyecto, los scripts de programación, los modelos, las texturas, etc. Esta ventana permite organizar el proyecto mediante la creación de carpetas.

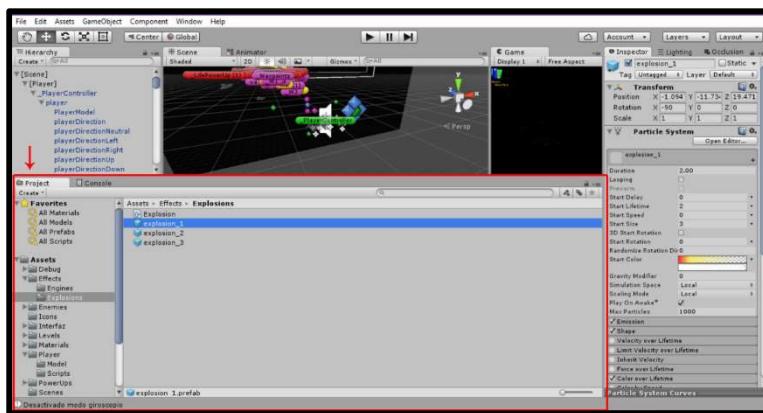


Ilustración 42 – Interfaz Unity, Project window

⁶⁷ Los assets [251] hacen referencia a cualquier recurso usado en la creación de videojuegos, desde modelos y texturas hasta códigos de programación.

Ventana del editor

La ventana del editor muestra la escena actual seleccionada y permite trabajar con los GameObjects navegando por la escena. Es una forma de visualizar las escenas del juego desde el punto de vista del desarrollador, pudiendo interactuar de una manera más visual con los objetos.

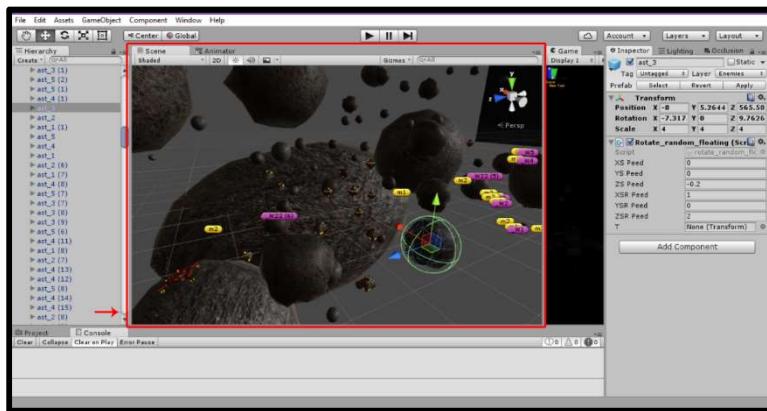


Ilustración 43 – Interfaz Unity, Scene view

Hierarchy e Inspector

La ventana de hierarchy muestra un listado de todos los objetos de la escena. Esta lista indica también la relación de vinculación entre ellos mediante una estructura jerárquica. Los objetos se pueden seleccionar tanto desde la ventana del editor como la del hierarchy⁶⁸, mostrándose entonces sus componentes en el inspector.

⁶⁸ No importa en cuál de las dos ventanas se seleccione el objeto, porque automáticamente se selecciona en la otra ventana y se muestra en el inspector.

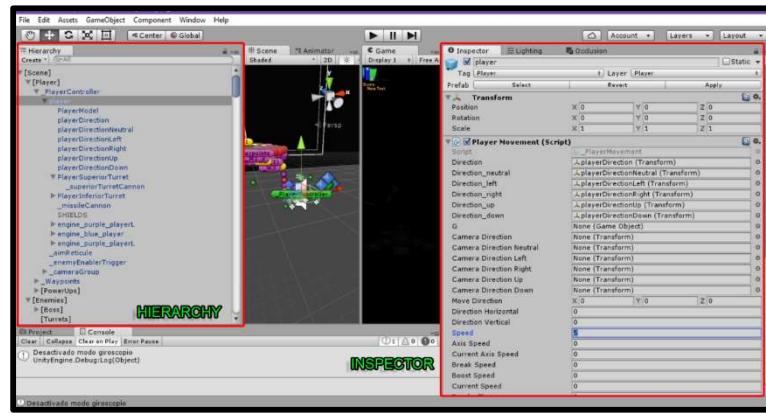


Ilustración 44 – Interfaz Unity, Hierarchy e Inspector

Desde el inspector se pueden ver y editar las propiedades y componentes de los GameObject.

Ventana del juego

La ventana del juego permite ver la ejecución del juego dentro de Unity para poder hacer pruebas en tiempo real.

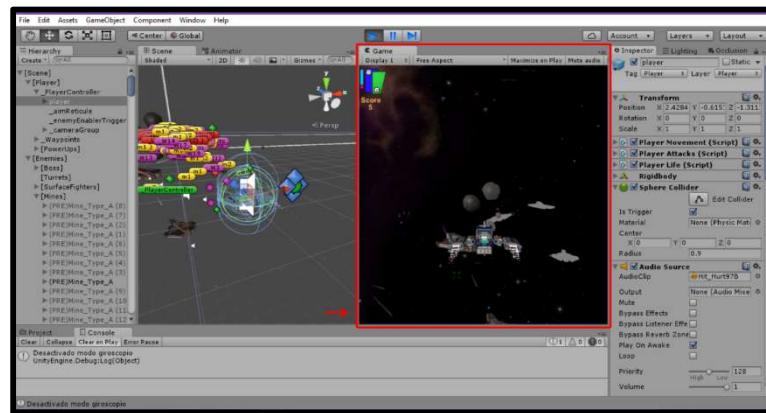


Ilustración 45 – Interfaz Unity, Game screen

Unity, el poder de la interfaz visual

Ya solo queda hablar de cómo funcionan los GameObjects y los componentes junto con la interfaz. Aquí es donde realmente está el poder de Unity. Si la estructura de crear objetos por piezas facilitaba mucho el proceso

de construir elementos, la interfaz de Unity agiliza enormemente el proceso de desarrollo y testeo de un videojuego. No solo permite ver los datos de los scripts y modificarlos desde la interfaz, sino que permite hacerlo en tiempo de ejecución del juego, es decir, mientras se está ejecutando el videojuego puedes hacer que el personaje salte más o menos alto, modificando el parámetro del Script asociado. Además, como todos los elementos de Unity son componentes, puedo simplemente arrastrar nuevos componentes desde la escena, la ventana del proyecto o copiar de otros GameObjects y usarlo en GameObject con el que estoy trabajando.

Prefabs

Para facilitar la construcción del videojuego y de los niveles, Unity cuenta con otra de forma de crear objetos en la escena, los prefabs [195]. Un prefab consiste en guardar un GameObject y sus componentes como una plantilla del que poder hacer copias. Su característica principal es que cualquier cambio que se haga al prefab modificará todas las instancias del mismo. Cuando necesitas que un elemento del juego aparezca varias veces en una escena y tengas que modificar algo, si se usa un prefab no tienes que modificar cada una de las copias que hay, solo el prefab original. Además, estas copias se pueden modificar por separado sin afectar al resto ni al original.

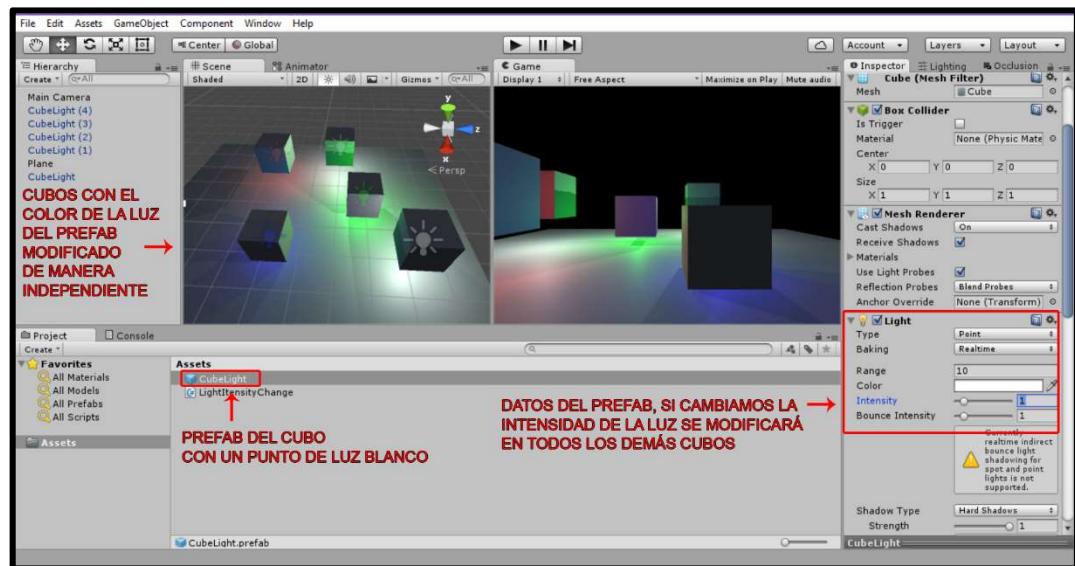


Ilustración 46 – Unity, ejemplo de uso de un prefab⁶⁹

Un ejemplo. Después de definir todos los GameObjects que forman a un gato con sus comportamientos y animaciones, se crea un prefab del mismo. Luego se colocan carios gatos por el escenario, indicando cuanta hambre y ganas de comer ratones tiene cada uno. A cada gato se le puede cambiar el color, tamaño, y hambre por separado, pero si se necesita cambiar su actitud con respecto a comer ratones, es mejor hacerlo desde el prefab para que se modifique en todos a la vez.

Por eso Unity es una herramienta tan eficaz a la hora de realizar prototipos. Todas sus funcionalidades están orientadas a dar rapidez y claridad mientras se desarrollan videojuegos.

⁶⁹ Descripción de la imagen de ejemplo de los prefabs: Se ha creado un cubo que contiene un punto de luz y se ha definido un prefab del mismo. Luego se han colocado varias instancias del prefab en la escena y se ha cambiado el color de la luz de manera individual. Con el prefab original seleccionado, ahora se puede modificar la intensidad de la luz y afectará a todos los cubos, da igual el color que tenga cada uno.

Unity remote 4

Antes de pasar a la siguiente sección de la memoria se va a hablar de una herramienta externa que se ha utilizado para complementar el desarrollo con Unity. Como se ha mencionado antes, se ha usado Unity remote 4 para realizar las primeras pruebas de funcionamiento y de control de movimiento del juego, ya que el controlador de AppleTV “Siri” cuenta con esta opción de control. Una vez probado el juego en AppleTV se ha seguido utilizado para pulir el control o realizar pruebas rápidas.

Para poder usar la aplicación

Se debe instalar [159] la aplicación en el dispositivo que se quiera hacer la prueba, Android o iOS que sea compatible con la misma. También se tiene que instalar en el propio proyecto de Unity, accediendo al Asset Store [196] y descargándola primero. Luego instalando la aplicación mediante la importación como un paquete de Unity en el proyecto y configurarla como se menciona en la documentación y en la Asset Store. Un tutorial de ejemplo de instalación se puede ver aquí. [197]

A tener en cuenta

A pesar de la utilidad clara de la aplicación, cabe destacar que no es infalible y puede generar problemas en el proyecto o fallos inesperados. Durante la elaboración de este TFG se detectaron algunos errores o bugs menores que se producían por usar la aplicación⁷⁰, así que es recomendable realizar las pruebas en un proyecto por separado o en una copia del proyecto.

⁷⁰ No se pudo identificar concretamente el origen de estos pequeños errores, que fueron sobretodos visuales, pero se ha llegado a la conclusión de que fueron debidos a la aplicación puesto que fue en lo único que se trabajó durante el tiempo en que se dieron dichos errores y no se modificó nada más del proyecto.

Diseño del videojuego (Vega: Factions War)

En esta sección se explicará el diseño y estructura general del videojuego. Se mostrarán sus entidades principales, la forma en la que se ha trabajado y algunos de los conceptos de diseño.

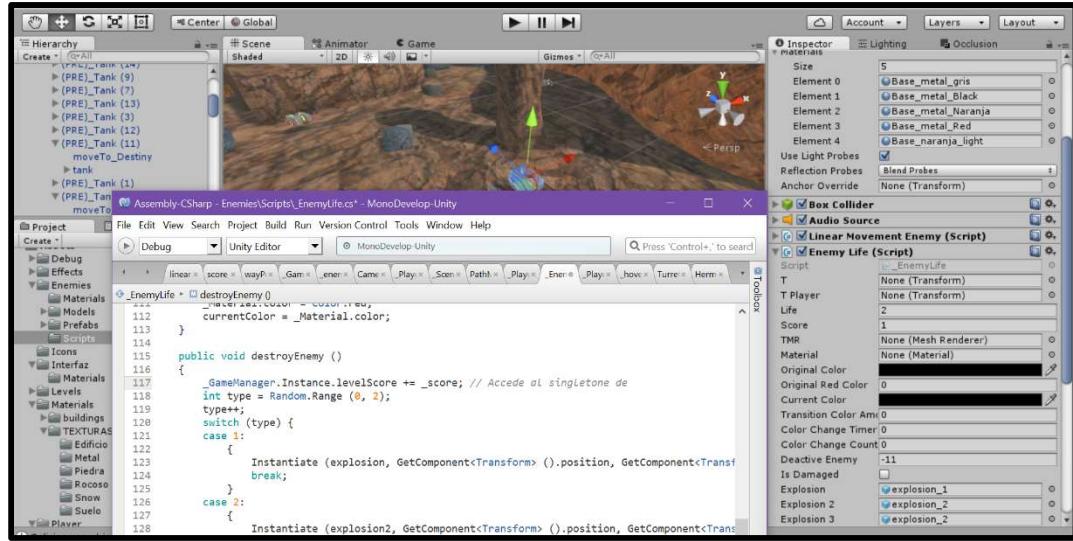


Ilustración 47 – Ejemplo de proceso de desarrollo. Trabajando en el sistema de daños de los enemigos

Conceptos de programación

Se va a comentar una serie de conceptos de programación y diseño necesarios para entender como se ha trabajado en este proyecto.

Patrones de desarrollo

Singletone: [198] Es un patrón de desarrollo que limita el número de instancias de una clase a un único objeto de acceso global. Es muy útil a la hora de mantener el control sobre un tipo de objeto que se va a usar repetidas veces en la aplicación, pero que no se necesitan varias copias del mismo ya que se va a usar como un controlador o coordinar dentro del sistema. Suele usarse en la implementación de otros patrones de desarrollo, como por ejemplo en el patrón fachada o factoría que solo se necesita una instancia de los mismos.

El funcionamiento es sencillo, si al crear o llamar al objeto Singleton ya existe un objeto de ese tipo, se devolverá la referencia a dicho objeto para ser usado. Si, por el contrario, el objeto aún no existe entonces se crea una instancia nueva del objeto. Con esto se controla que solo haya una instancia del objeto en todo momento.

< PSEUDO CÓDIGO >

```
public class Singleton {  
  
    private static Singleton* pinstance = 0;  
  
    public static Singleton* Instance {  
  
        if (pinstance != null) {  
  
            return pinstance;  
  
            /* si ya existe el objeto se devuelve la  
            instancia ya creada*/  
  
        }  
  
        else {  
  
            pinstance = new Singleton();  
  
            /* si no se ha inicializado el objeto se crea  
            una instancia nueva */  
  
        }  
    }  
}
```

Para más información o referencias sobre los patrones de desarrollo en programación véase [199] [200] [201].

técnicas de programación con Unity

Ya se han explicado las bondades de Unity y de su interfaz. Ahora se van a explicar y mostrar sus usos con ejemplos dentro del propio proyecto.

Variables públicas

Cuando en un componente tipo script de Unity, se declara una variable [202] como pública⁷¹, se muestra en el contenido de dicho componente en la ventana del inspector. En la ventana no solo se puede ver el valor de la variable, sino que también se puede modificar sus parámetros. Aparte de los beneficios obvios, sobre todo a la hora de debuggear⁷² y realizar pruebas en tiempo real, existe otra aplicación.

Por la estructura entidad-componente de Unity y su forma de tratar a los GameObjects, se pueden usar estos últimos como variables. Es decir, se pueden coger los datos o componentes de un GameObject y guardarlos en una variable para luego poder acceder a los componentes del mismo.

⁷¹ Una variable pública es visible y accesible en cualquier parte del programa.

⁷² Debuggear [248] o depurar consiste en buscar y corregir errores del código, ya sea a mano o mediante una herramienta de apoyo software.

```

4 public class _PlayerMovement : MonoBehaviour
5 {
6     private Transform _T;
7     private Transform _TGroup;
8     private Transform _TPlayerController;
9     public Transform _direction;
10    public Transform _direction_neutral;
11    public Transform _direction_left;
12    public Transform _direction_right;
13    public Transform _direction_up;
14    public Transform _direction_down;
15    public GameObject _G;
16    public Transform _cameraDirection;
17    public Transform _cameraDirectionNeutral;
18    public Transform _cameraDirectionLeft;
19    public Transform _cameraDirectionRight;
20    public Transform _cameraDirectionUp;
21    public Transform _cameraDirectionDown;
22
23    public Vector3 _MoveDirection;
24
25    public float directionHorizontal;
26    public float directionVertical;

```

Ilustración 48 – Ejemplo de código en Unity, componentes como variables⁷³

Funciona de la siguiente manera:

- Un GameObject es una combinación de distintos componentes más un componente Transform. El componente Transform actúa como objeto padre⁷⁴ del resto de componentes.
- Desde el Transform se puede acceder al resto de componentes del GameObject, y desde cualquier componente del GameObject se puede acceder a su transform.
- Todos los componentes de un GameObject pueden acceder a dicho GameObject o entidad.

⁷³ Transform es el componente padre de las entidades de Unity, y en este caso se está usando como variable para guardar un enlace a otras entidades en este código. Desde ese Transform se podrá acceder a otros componentes de la entidad a la que pertenece. Además de eso, también se está almacenando un GameObject entero. En este caso se está utilizando para guardar el modelo 3D de la nave del jugador.

⁷⁴ El término padre viene dado por una característica de la programación orientada a objetos, en la que un objeto está relacionado con otro en una estructura jerárquica padre-hijo, siendo el padre el que está por encima de los hijos.

```

void Awake ()
{
    _TGroup = gameObject.GetComponent<Transform> ();
    _G = GameObject.Find ("PlayerModel");
    _T = _G.GetComponent<Transform> ();
    _TPlayerController = GameObject.Find ("_PlayerController").GetComponent<Transform> ();

    _direction = GameObject.Find ("playerDirection").GetComponent<Transform> ();
    _direction_neutral = _T.FindChild ("playerDirectionNeutral").GetComponent<Transform> ();
    //Ejemplos de relaciones entre componentes y GameObjects
}

```

Ilustración 49 – ejemplo de código en Unity. Relaciones entre componentes dentro de la jerarquía

En esta imagen se pueden ver varios ejemplos de uso de las propiedades de relación entre componentes mediante código script.

```
_TGroup = gameObject.GetComponent<Transform> ();
```

Usando `gameObject.GetComponent<Transform>()` se accede al componente Transform del GameObject que tiene el script asociado como componente⁷⁵. Esto permite guardarla como una variable y usarlo como referencia más tarde.

```
_G = GameObject.Find ("PlayerModel");
```

`GameObject.Find(String name)`⁷⁶ busca la entidad dentro de la escena con el nombre que se encuentra entre paréntesis. En el ejemplo se está guardando el modelo de la nave del jugador como una variable.

```
_direction_neutral = _T.FindChild ("playerDirectionNeutral").GetComponent<Transform> ();
FindChild(String name) permite buscar entre los hijos del componente Transform que llama al método una entidad hija con el nombre que se encuentra
```

⁷⁵ Esta acción es la relación que se ha comentado antes, desde un GameObject se puede acceder a su Transform, y desde un componente se puede a su vez, obtener el GameObject al que pertenece.

⁷⁶ En el ejemplo se puede ver como se llama a GameObject con mayúsculas, o con minúsculas. En un script de Unity GameObject hace referencia a la clase GameObject, de la que todos los GameObjects han sido creados. Mientras que gameObject hace referencia al GameObject específico al que pertenece el script.

entre paréntesis. En este caso, se está buscando una entidad que almacena la posición neutra de la posición de la nave del jugador.

Como se puede observar, usando estas reglas el usuario dispone de mucha libertad a la hora de programar, y si se plantea una estructura adecuada, solventar algunos de los problemas que se han mencionado de la estructura entidad-componente en Unity.

Este sistema de guardar en variables cualquier componente permite, por ejemplo, almacenar el componente script de un GameObject en otro GameObject distinto. Así, el GameObject que almacena el script como variable puede modificar o ejecutar los métodos y variables públicos del script del otro GameObject.

Pese a la libertad que ofrece esta técnica, también puede generar redundancias de código y desorganización dentro del proyecto, así que hay que planificar su uso con cuidado.

Otro ejemplo práctico.

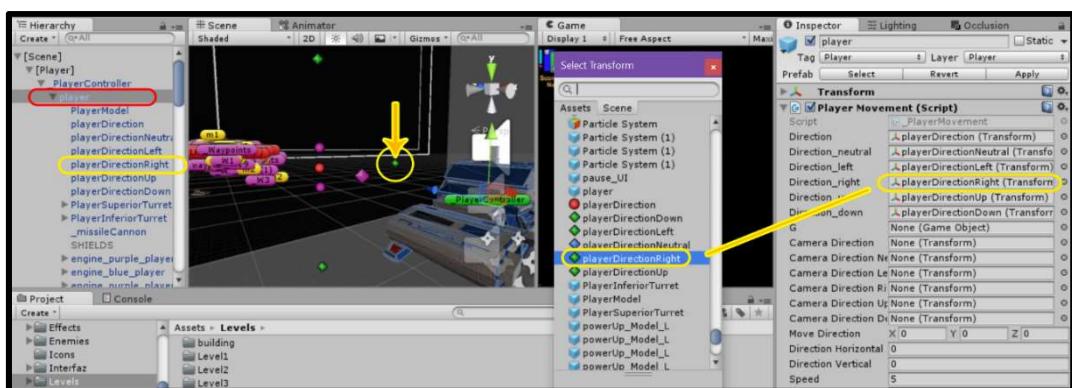


Ilustración 50 – Ejemplo del uso de los componentes como variables. Asignación

En la imagen se puede ver un ejemplo de asignación de un componente de un objeto como variable en el script de otro objeto. El script en cuestión es el que controla el movimiento de la nave por parte del jugador.



Ilustración 51 – Objeto seleccionado en el hierarchy de Unity

En rojo se muestra el objeto al que pertenece el script, en este caso, la entidad player.

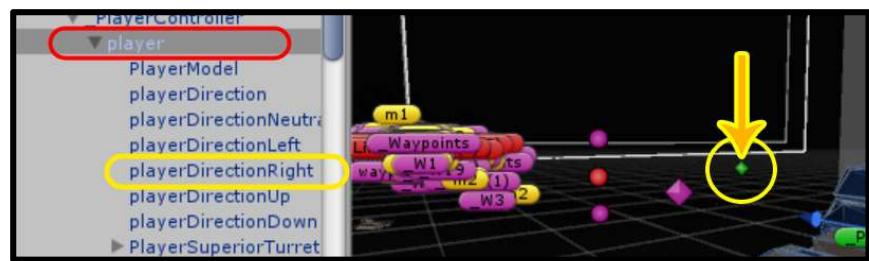


Ilustración 52 – Visualización de la relación entre objetos padre e hijos en el hierarchy de Unity

En amarillo se muestra primero el objeto del que se van a guardar sus datos. El objeto **playerDirectionRight** almacena la posición derecha máxima a la que puede girar la nave del jugador. Además, resulta que el objeto en cuestión es hijo del objeto player.

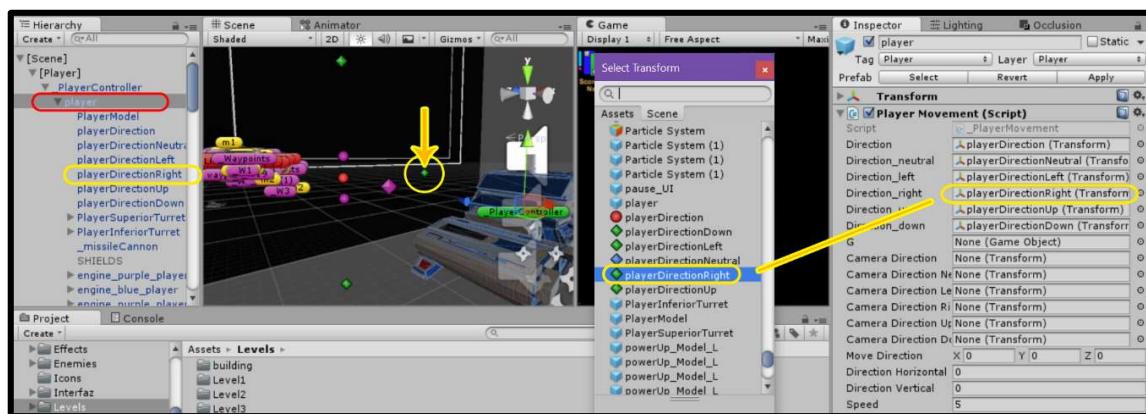


Ilustración 53 – Ejemplo de asignación de un objeto como variable mediante la interfaz de Unity

En **PlayerMovement** se está asignando el componente Transform de **playerDirectionRight** a la variable **direction_right**. Al hacer esto el script de player puede acceder directamente a ese componente del objeto hijo sin necesidad de escribir más código.

Estructura del proyecto

Esquema general

En la siguiente imagen se puede ver el mapa de uso de Vega: Factions War. Es un resumen del flujo de proceso del videojuego y muestra la estructura de la aplicación.

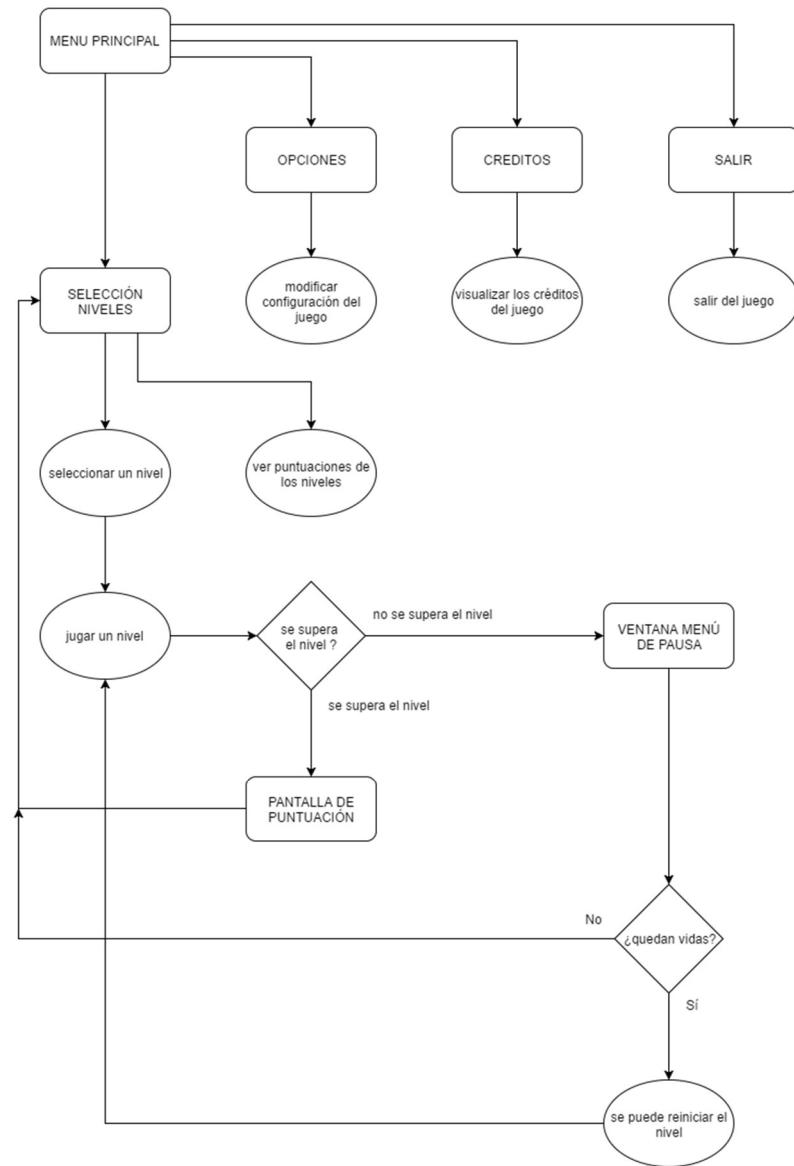


Ilustración 54 – Diagrama de flujo y distribución general de la aplicación Vega: Factions War

Desde el menú principal se puede salir de la aplicación, acceder al menú de opciones, créditos y a la selección de niveles.

En la pantalla de selección de niveles se puede elegir el nivel al que se quiera jugar y se encuentre desbloqueado en ese momento.

Una vez seleccionado el nivel empieza la partida. Si se supera con éxito el nivel se muestra una pantalla de puntuación y se vuelve al menú de selección de niveles. En cambio, si se pierde una vida y se falla el nivel, pueden ocurrir dos situaciones.

La primera situación se da cuando al jugador le quedan vidas. Si aún le quedan vidas se puede repetir el nivel. Además, si se ha desbloqueado un **checkpoint** durante el anterior recorrido del nivel, se continuará a partir de ese punto en el nivel con la puntuación que se consiguió en ese momento.

Si, por el contrario, al jugador no le quedan vidas, se pierde la partida y se volverá a la pantalla de selección de niveles, perdiendo los checkpoints que se hayan desbloqueado en ese nivel.

La estructura del videojuego dentro de Unity se puede observar con la siguiente imagen. Es un esquema de cómo se han construido cada una de las escenas del mismo.

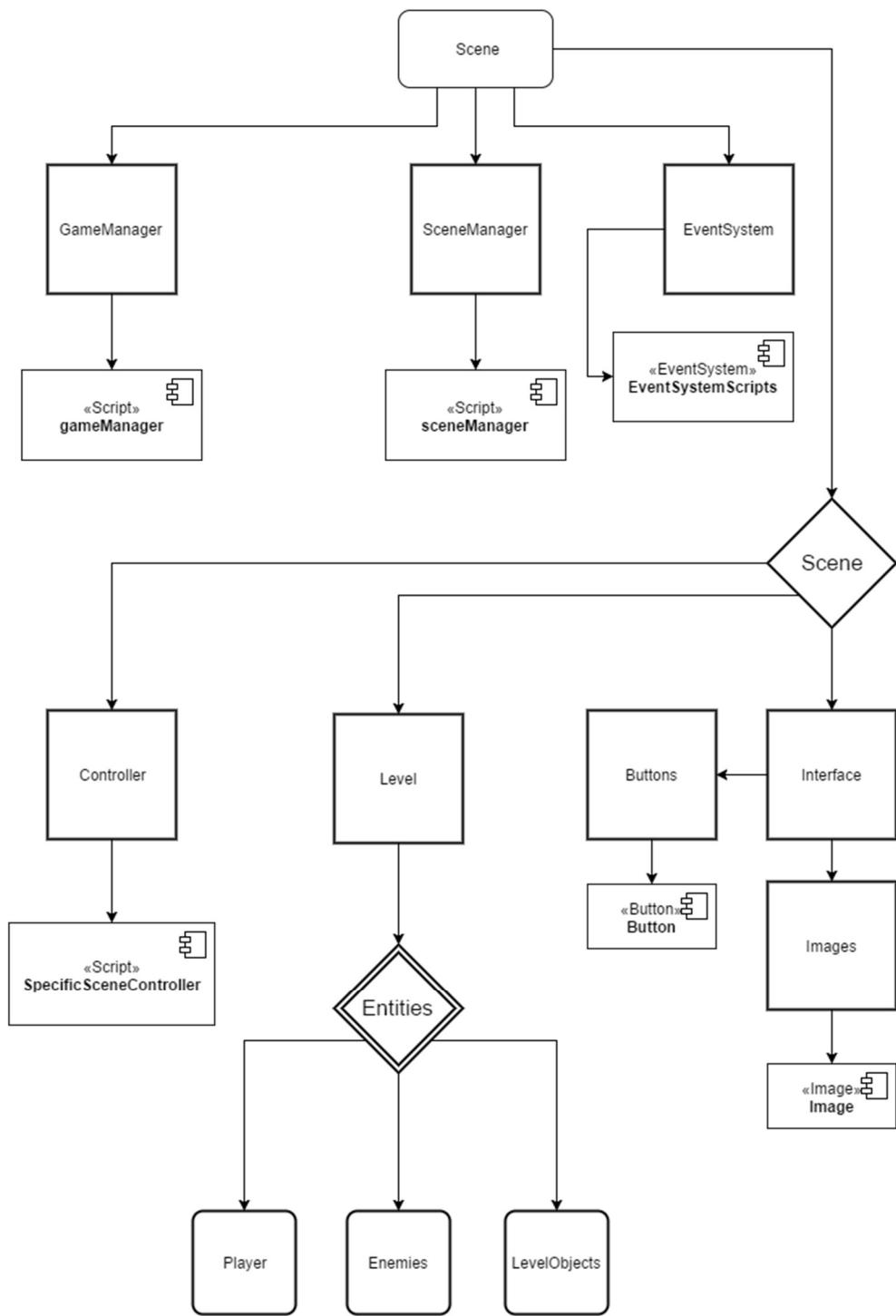


Ilustración 55 – Estructura del juego en Unity, partes que lo componen.

El videojuego se ha dividido en varias escenas. Todas las escenas cuentan con un controller específico⁷⁷ de esa escena y una interface.

El controller gestiona el funcionamiento de la escena. En las pantallas de menús permite mostrar los datos que se necesiten en la interfaz y controlar la navegación entre menús. En los niveles el controller es el singleton **gameManager**, que se encarga de llevar la puntuación, gestionar los enemigos, el menú de pausa, el guardado de datos de la partida,etc.

A parte de los controllers y la interface, existe un singleton **sceneManager** que gestiona los datos del juego, las opciones y la carga de las distintas escenas.

Las escenas que contienen los distintos menús del juego son una versión simplificada de las que forman los niveles, pero su estructura base es la misma. Lo que las simplifica es el hecho de no tener el contenido del nivel y su lógica de funcionamiento. Lo que sería el personaje controlado por el jugador, los enemigos y los componentes y mecánicas del nivel.

Entidades principales

Ahora se va a hablar de las entidades principales que forman el videojuego y a explicar como están compuestas. Se darán más detalles en la sección de proceso de desarrollo.

⁷⁷ Script del código de programación encargada de gestionar y controlar otras partes del código.

La entidad TPlayer

Representa al personaje jugador. Como se puede ver en la imagen, es una entidad que está formada por otras entidades en estructura jerárquica.

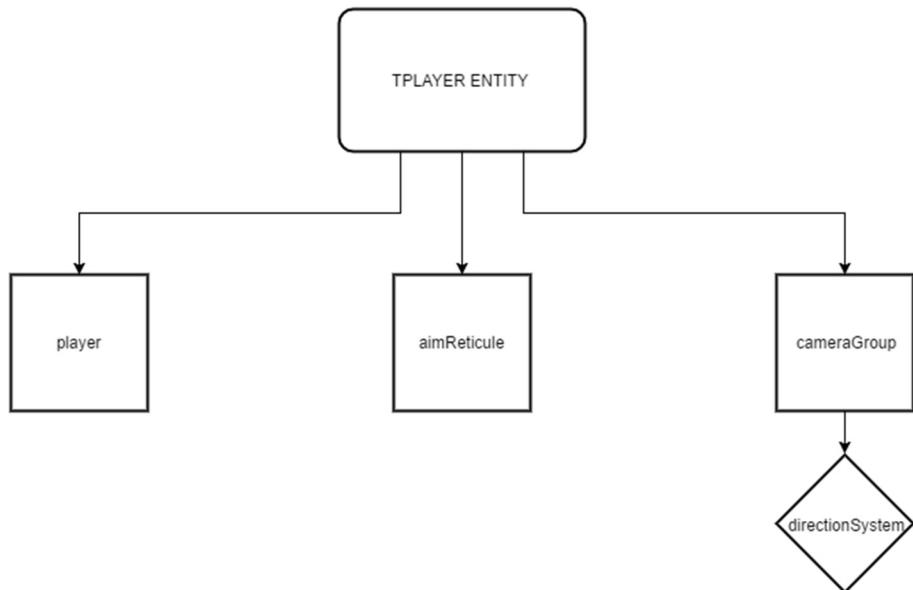


Ilustración 56 – Esquema de la entidad TPlayer

Casi todas las funciones del personaje jugador las controla player, mientras que cameraGroup controla el movimiento de la cámara. Por su lado, aimReticule forma parte del sistema de apuntado para dirigir los ataques del jugador incluido en player, que por motivos de funcionamiento se debe mantener fuera de la entidad.

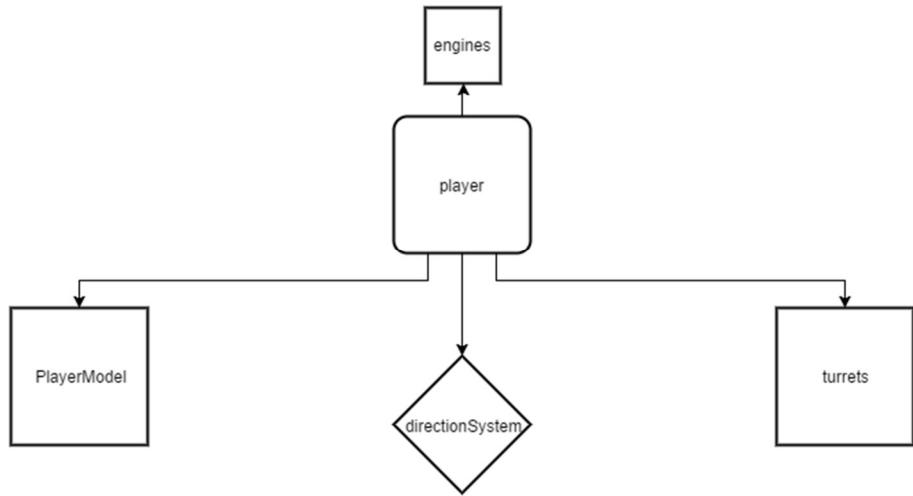


Ilustración 57 - Esquema de la entidad player

La entidad player es la encargada de gestionar el control de la nave por parte del jugador y su movimiento por la pantalla. A su vez, tiene como hijos otras entidades que forman parte del sistema de apuntado y de dirección de la nave.

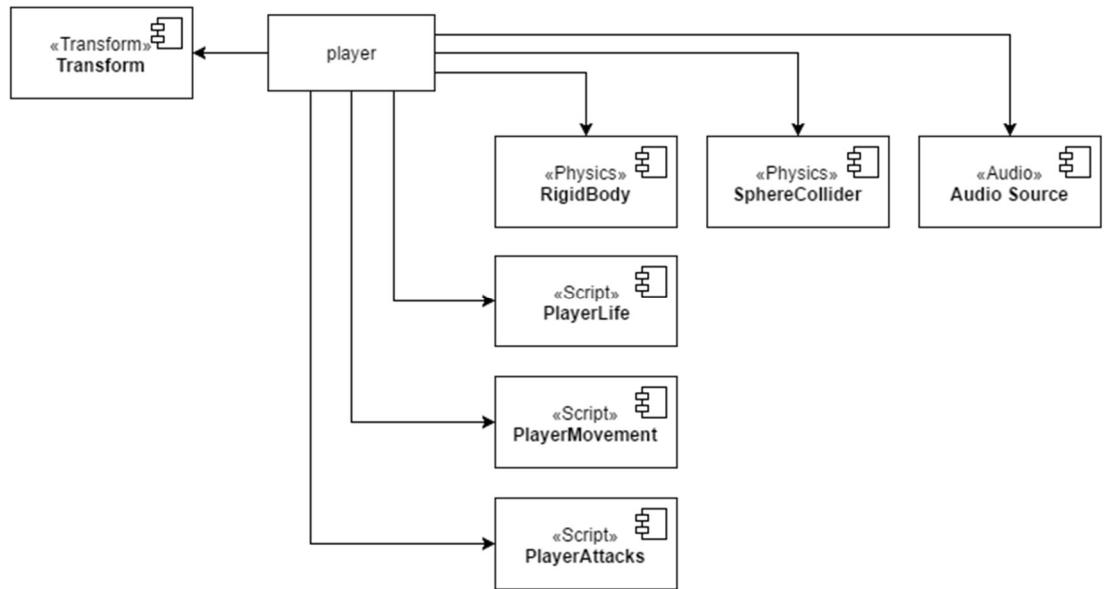


Ilustración 58 - componentes principales de la entidad player

Estas entidades junto con el movimiento de la nave son controladas por los scripts **PlayerMovement**, **PlayerAttacks** y **PlayerLife**.

PlayerMovement	PlayerAttacks	PlayerLife
<pre>+ DirectionHorizontal: float + DirectionVertical: float + Speed: float + AxisSpeed: float + CurrentAxisSpeed: float + CurrentSpeed: float + MaxHorizontalDistance: float + MaxVerticalDistance: float + PathHorizontalPoint: float + PathVerticalPoint: float + PathHorizontalLimit: float + PathVerticalLimit: float + AxisVerticalUpsideDown: bool + Direction: GameObject(Transform) + Direction_neutral: GameObject(Transform) + Direction_left: GameObject(Transform) + Direction_right: GameObject(Transform) + Direction_up: GameObject(Transform) + Direction_down: GameObject(Transform) + G: GameObject + CameraDirection: GameObject(Transform) + CameraDirectionNeutral: GameObject(Transform) + CameraDirectionLeft: GameObject(Transform) + CameraDirectionRight: GameObject(Transform) + CameraDirectionUp: GameObject(Transform) + CameraDirectionDown: GameObject(Transform) + MoveDirection: Vector3</pre>	<pre>+ FireRate: float + ReloadTimer: float + TurretRotationSpeed: float + CanFire: bool + SuperiorCanFire: bool + InferiorCanFire: bool + AudioSourceSuperior: GameObject(AudioSource) + AudioSourceInferior: GameObject(AudioSource) + T: GameObject(Transform) + PlayerProyectil: GameObject + SuperiorTurretCannon: GameObject(Transform) + LayerMask: Mask(Layers) + Screen: Vector3</pre>	<pre>+ Shields: int + Life: int + MaxShields: int + MaxLife: int + Shields: int + Continues: int + ShieldRate: float + ShieldTimer: float + InvulnerabilityTimerDamage: float + InvulnerabilityTimerCollision: float + InvulnerabilityCount: float + Invulnerability: bool + Col_shields: GameObject(Collider) + Col_ship: GameObject(Collider) + T: GameObject(Transform) + TCurrent: GameObject(Transform) + Rb: GameObject(Rigidbody) + Shield(GameObject): GameObject</pre>

Ilustración 59 – Componentes scripts principales de player⁷⁸

Por último, la entidad que muestra el modelo y los efectos de la nave se llama playerModel.

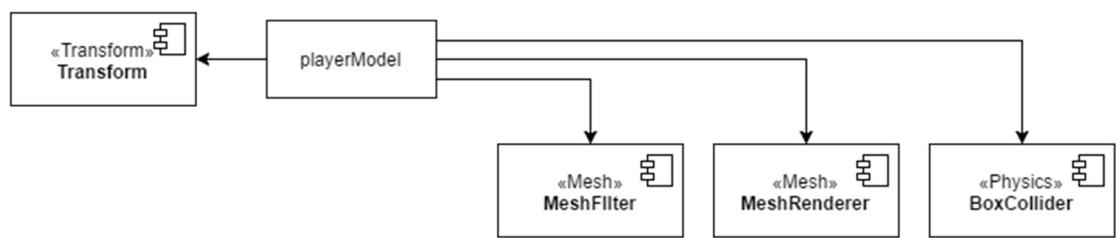


Ilustración 60 – componentes que forman la entidad playerModel

⁷⁸ Para facilitar la lectura y la comprensión general de las entidades, solo se muestran las variables. En apartados posteriores se explicará con más detalle algunas de las funciones que pueden realizar.

Entidad de los enemigos

Los distintos tipos de enemigos están creados cada uno de forma distinta, no es lo mismo un caza que una mina de proximidad. Pero la base de su estructura es la misma y es lo que se va a mostrar a continuación.

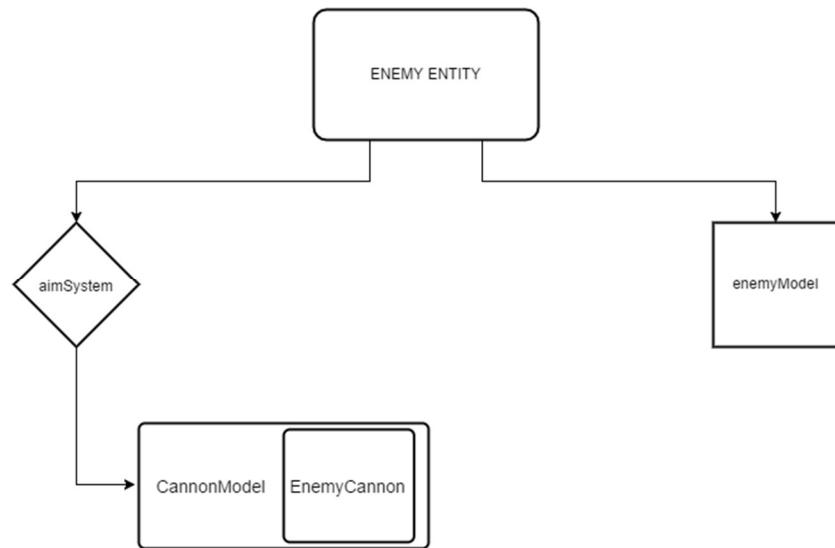


Ilustración 61 – esquema de la entidad enemigo

Por un lado, los enemigos tienen su entidad modelo que contiene los scripts que controlan su movimiento y su vida.

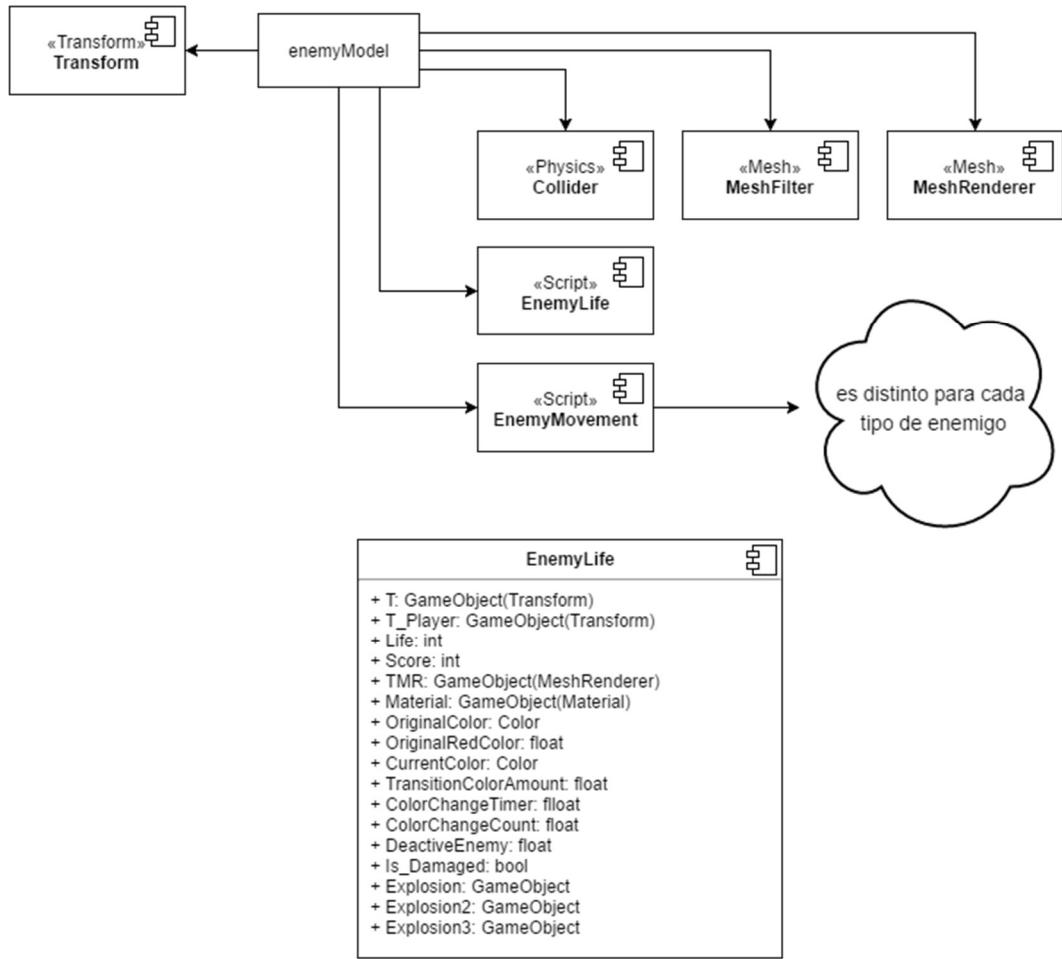


Ilustración 62 – componentes principales de la entidad enemyModel

Por el otro, tienen un sistema de entidades que gestiona el sistema de disparo y apuntado, controlados por los scripts **EnemyFiringSystem** y **EnemyAimingSystem**.

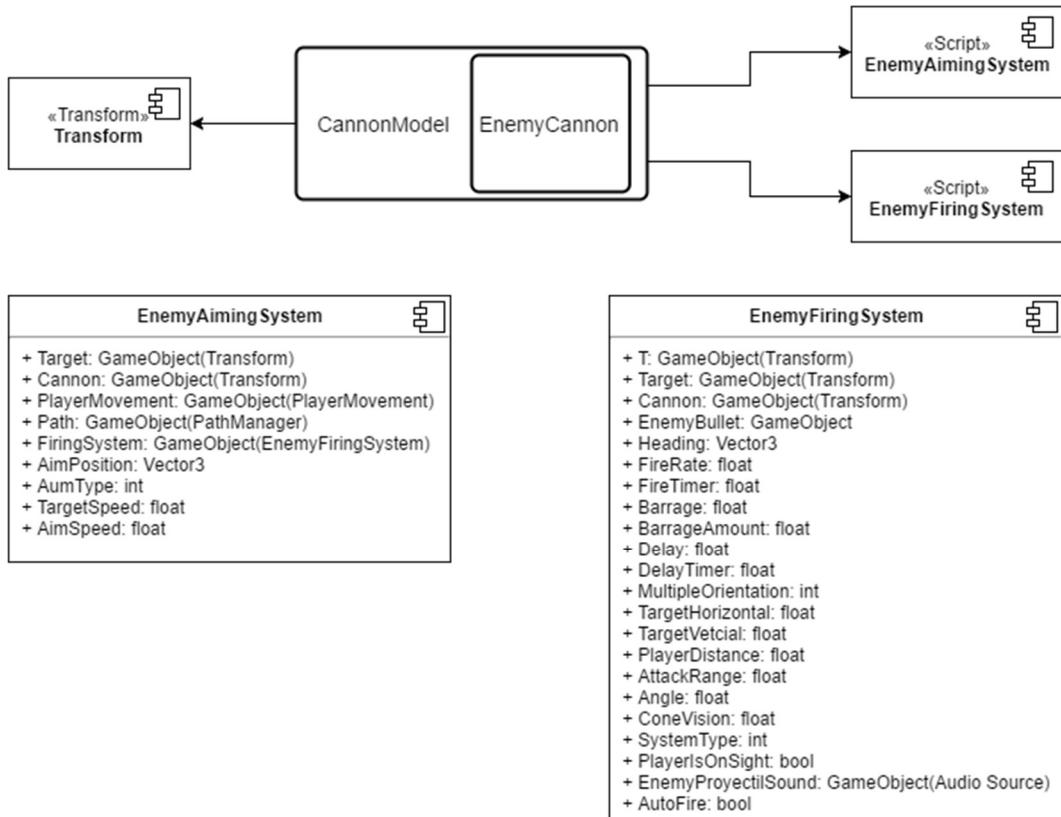


Ilustración 63 – componentes principales de las entidades que controlan el sistema de apuntado

Entidad Waypoint y PathManager

Para terminar, vamos a mostrar las entidades que gestionan el movimiento de la nave por el nivel, PathManager se encarga de mover la nave del jugador por el camino que ha calculado previamente. Para calcular el camino, tiene almacenado una serie de puntos en una lista, ordenados por posición, que indican la ruta que la nave va a seguir. Estos puntos son representados por la entidad Waypoint.

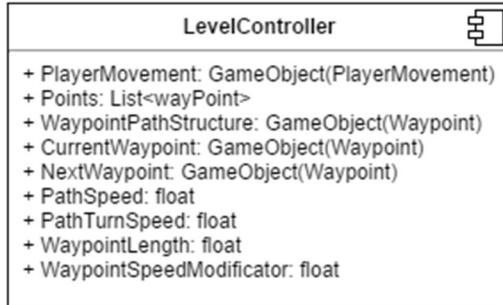


Ilustración 64 - componentes de la entidad PathManager

El funcionamiento de estos Waypoint es sencillo. Cada Waypoint tiene una serie de datos que permiten controlar la dirección y el recorrido al siguiente Waypoint de la lista.

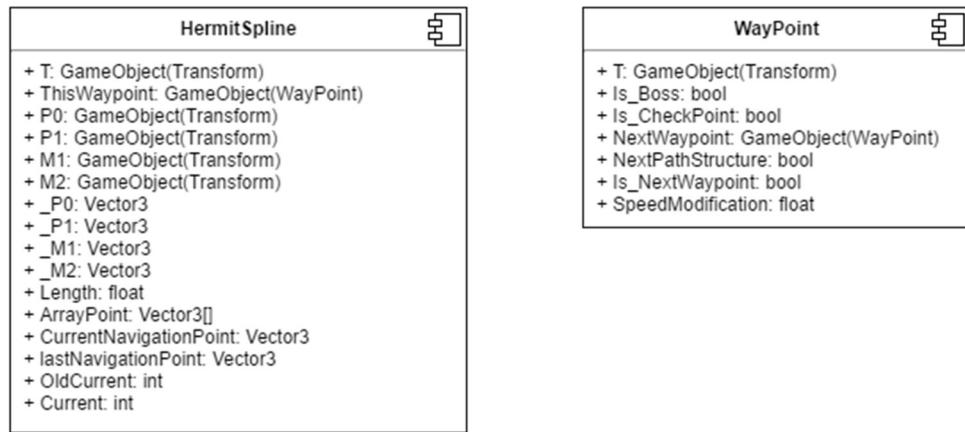
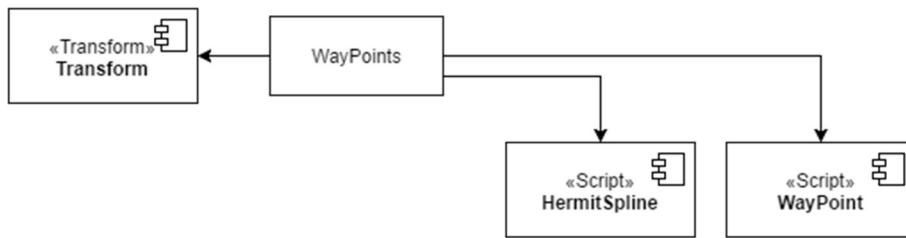


Ilustración 65 - componentes de la entidad Waypoint

Proceso de desarrollo

Para crear el producto final se ha seguido un proceso de desarrollo basado en prototipos, partiendo de unos conceptos y esquemas de ideas iniciales sobre los que se ha ido iterando. Primero se eligió la plataforma para la que se iba a desarrollar el videojuego, evaluando los requisitos y necesidades de la misma, así como las preferencias de los posibles consumidores. Una vez completado este primer paso, uno de los más importantes, se pasó a concretar el tipo de videojuego que se pretendía desarrollar.

La idea inicial era de un juego de acción arcade sobre raíles, un tipo de juego sencillo y divertido que no consume mucho tiempo.

Después se eligió la temática y atmósfera; creando un mundo de ciencia ficción futurista daba libertad a la hora de elegir el aspecto visual y facilita la implementación de mecánicas divertidas.

Finalmente se pensaron en las mecánicas y cómo podía encajar todo junto. Basándose en las características de los juegos arcade sobre raíles se pensó en la mejor manera de implementar el sistema de control, ya que este afecta en gran medida a como se va a jugar y al resto de mecánicas. Finalmente se planteó la interacción con los enemigos y el escenario.

A continuación se pueden ver algunos bocetos de las ideas iniciales del proceso de creación.

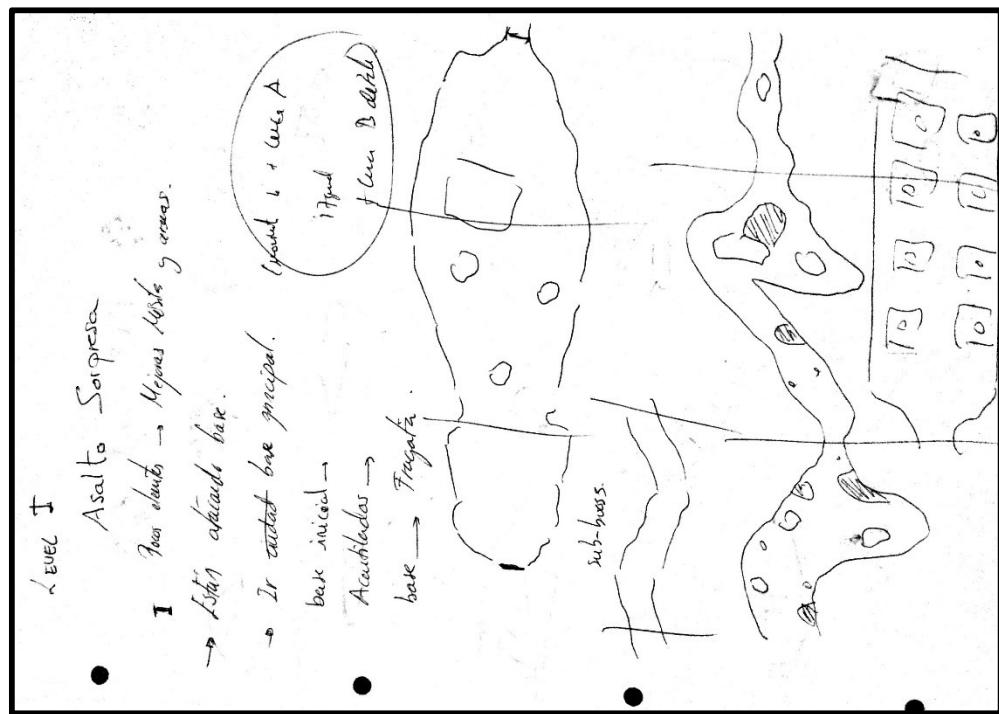


Ilustración 66- Diseño conceptual del primer nivel del juego

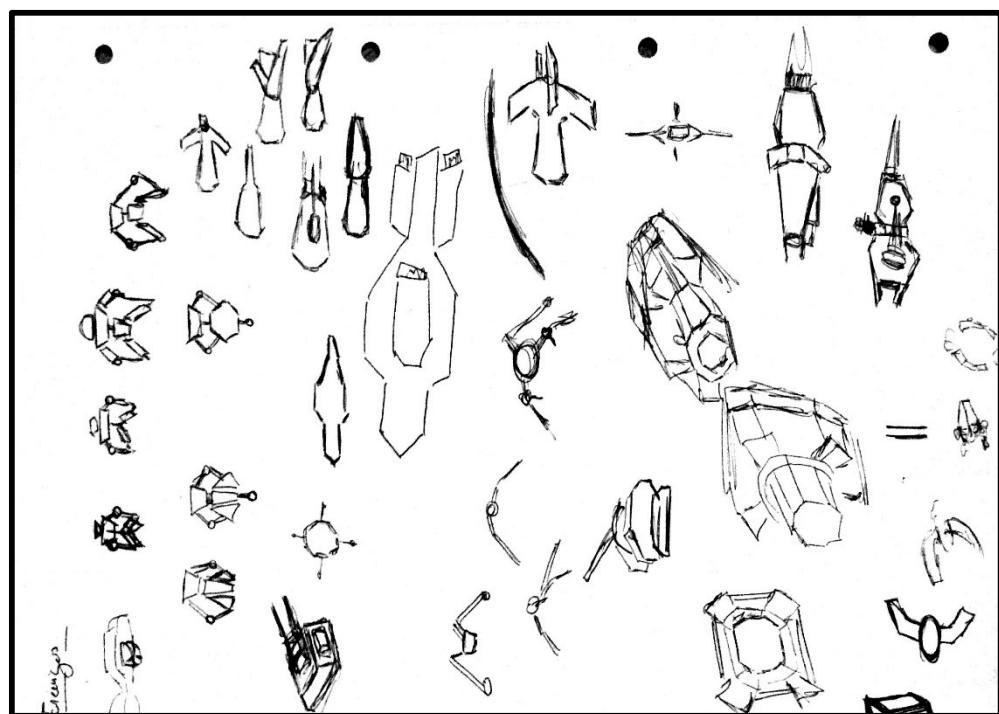


Ilustración 67 – Diseño conceptual de naves

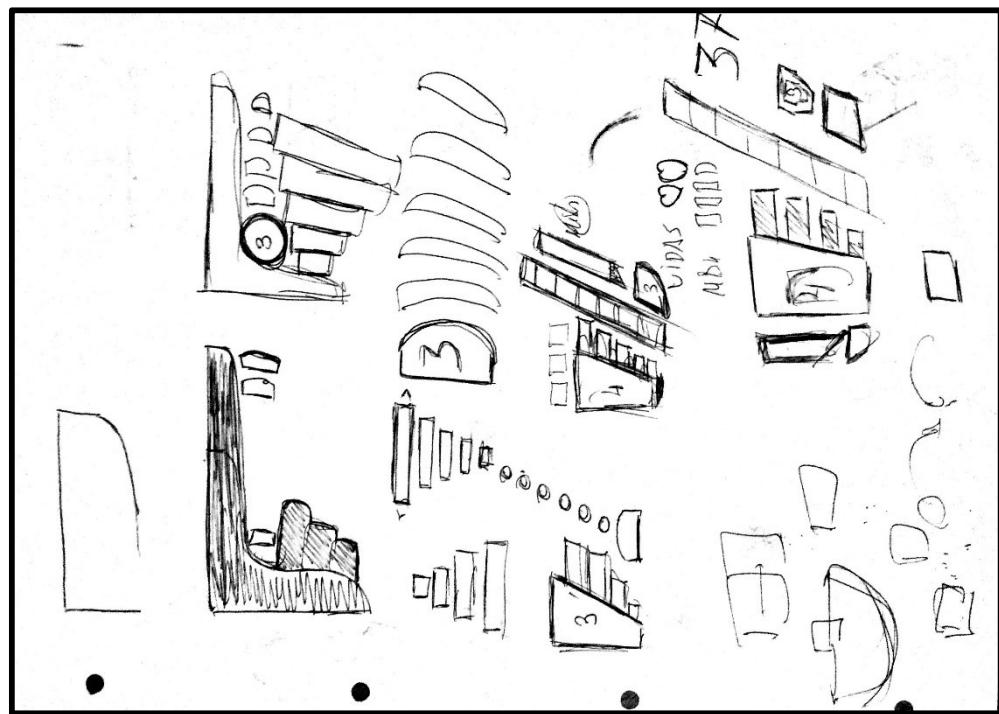


Ilustración 68 – Diseño conceptual de la interfaz del juego

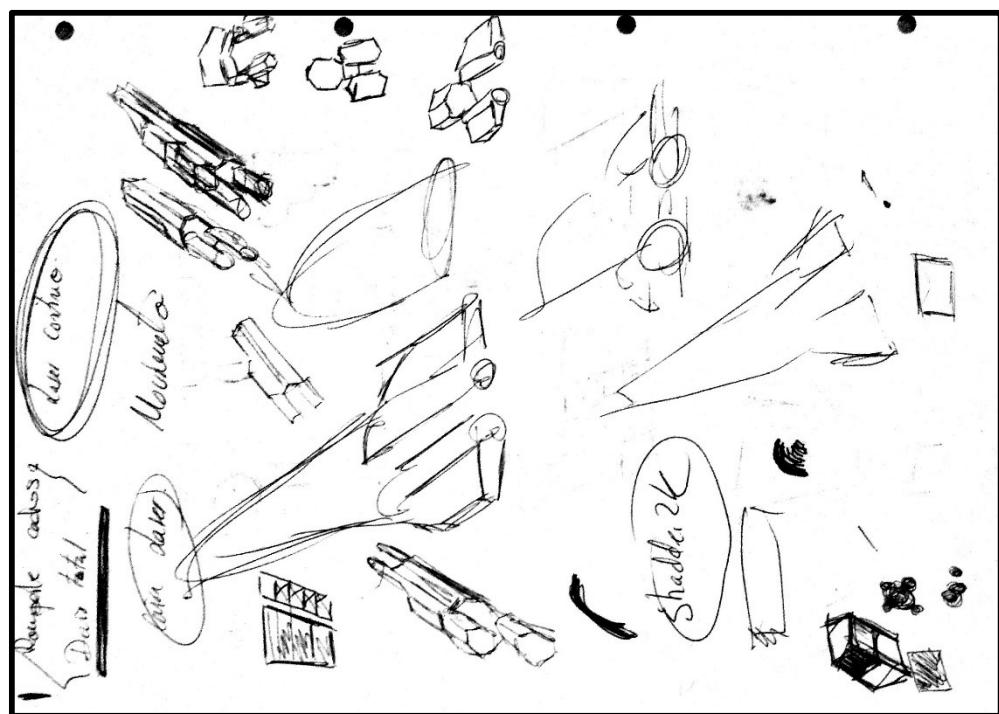


Ilustración 69 – Diseño conceptual de naves

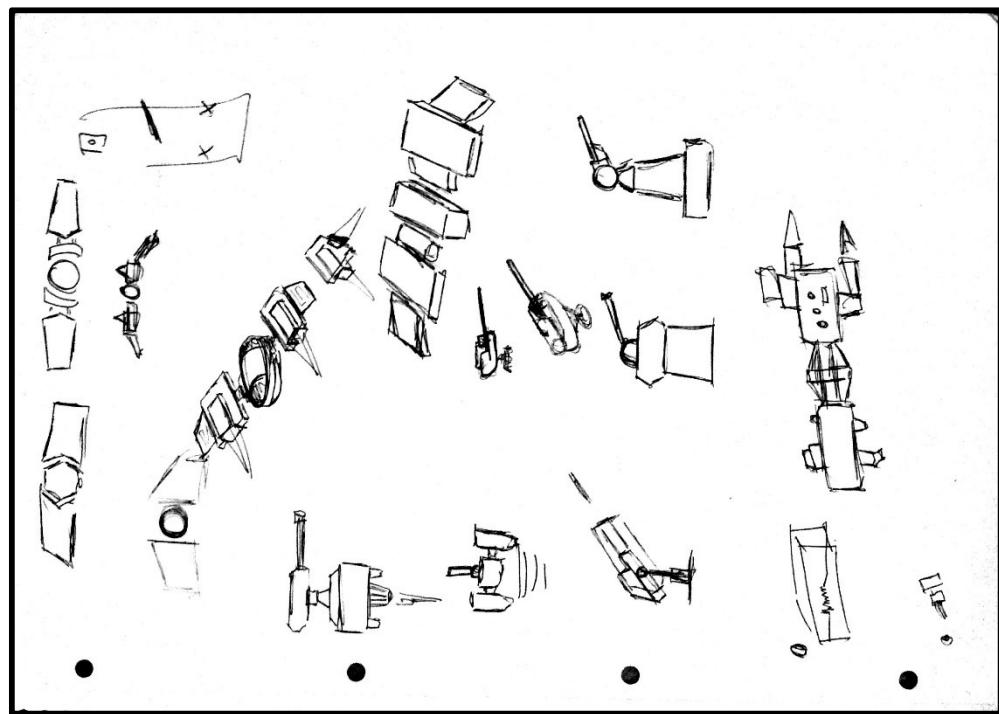


Ilustración 70 – Diseño de los modelos del juego

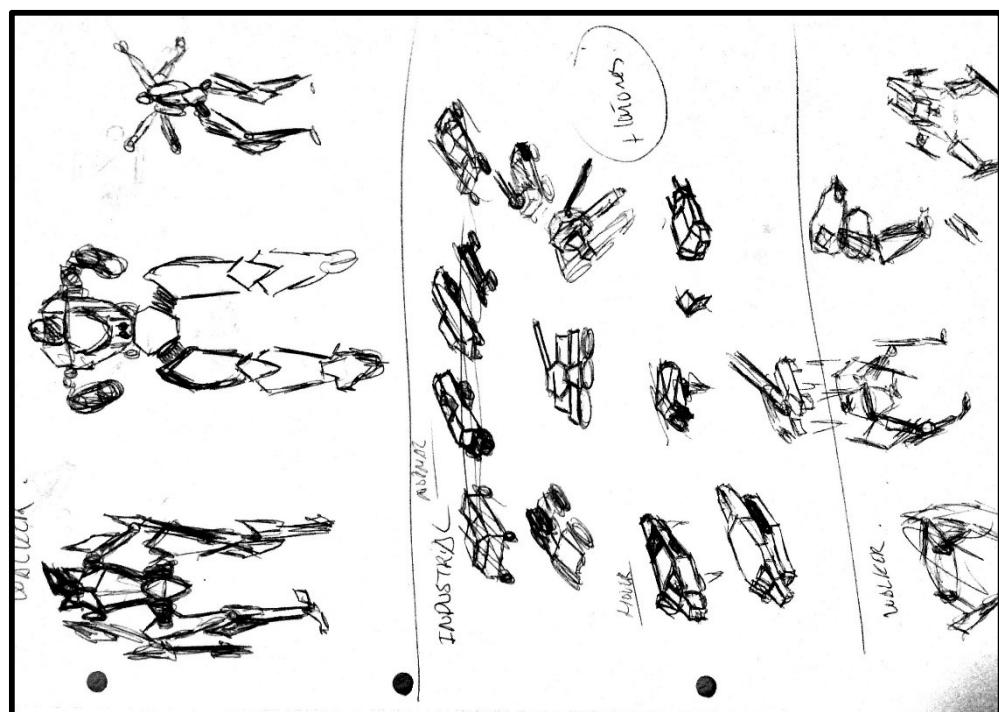


Ilustración 71 – Diseño conceptual de los enemigos terrestres del juego

Implementación

Jugador

El videojuego empezó a desarrollarse por el jugador. Es el elemento más importante y la propia jugabilidad depende en gran medida de cómo se controle la nave. Se decidió un sistema de control sencillo y que permita cierto grado de libertad, como en la mayoría de juegos del mismo tipo. El jugador puede moverse arriba, abajo, izquierda y derecha hasta unos límites de la pantalla, mientras que la nave avanza automáticamente por el recorrido del nivel.

El sistema de puntos de vida

Para añadir mecánicas interesantes que recompensen la habilidad, se diseñó un sistema de puntos de vida de dos niveles. En el primer nivel, se tiene un escudo que se recarga con el tiempo, lentamente. Es la manera de recompensar al jugador por su destreza en el juego y recibir pocos ataques.



Ilustración 72 – Sistemas de escudos de la nave del jugador

Una vez que el jugador ha perdido el escudo empieza el segundo nivel, los puntos de vida que se pierden son los de la nave, que no se pueden recuperar de manera normal y que una vez lleguen a cero se termina la partida.

Enemigos

Los enemigos son uno de los elementos interactivos más importantes del juego. Se desarrollan teniendo en cuenta que puedan ofrecer distintos niveles de desafío al jugador. Deben suponer un reto que al ser derrotados ofrezcan una recompensa al jugado, ya sea por el mero hecho de superarlos o mediante una puntuación.

¿Cómo diseñar el comportamiento de los enemigos?

Para ofrecer variedad se han diseñado varios tipos de enemigos, que actúan de manera diferente y ofrecen una mecánica jugable distinta. Desde los enemigos más sencillos hasta los jefes finales, cada tipo distinto tiene sus patrones de ataque y movimiento diferenciados. Además, se han diseñado de tal manera que dentro de cada tipo pueda haber cierta variación, aumentando el abanico de casos a los que se puede enfrentar el jugador, para mantener la jugabilidad fresca y entretenida.

Niveles

Los niveles son otro de los elementos de interacción más importantes del videojuego. Es dónde se ofrecen los encuentros con los. Además, añaden sus propias mecánicas jugables mediante cambios en el entorno en el que el jugador se debe enfrentar a los enemigos, añadiendo obstáculos, zonas cerradas o abiertas. Se han implementado de tal manera que cada nivel cuente con sus características distintivas y ofrezcan variedad de situaciones dentro del propio nivel.

Menús e interfaz

Se plantearon para que fuesen sencillos y funcionales. La navegación y lectura es fácil, ofreciendo al jugador los datos que necesita. Se ha basado el diseño en el tipo de ambientación de ciencia ficción de la temática del videojuego.

Prototipos y mecánicas

La etapa inicial del desarrollo de Vega: Factions wars se ha centrado en crear varios prototipos de la jugabilidad, aprovechando la facilidad de Unity para crear prototipos rápidamente. Cada prototipo se ha centrado en un aspecto de las bases del juego para ver la viabilidad de las ideas implementadas y sus mecánicas. El proceso seguido ha sido el siguiente:

- Tras terminar el prototipo, si el resultado era bueno, se itera sobre lo que se ha desarrollado para mejorarlo y llevarlo a la versión final del producto.
- Si las ideas no son divertidas o parecen demasiado complicadas o no encajan bien con el resto se desechan y se re imagina la idea o concepto.
- Según se van consiguiendo buenos prototipos, estos se juntan en prototipos más complejos. Así se combinan sus mecánicas y su jugabilidad, con la finalidad de acabar con un prototipo de toda la base del juego del que crear la versión final del mismo.

Prototipos principales desarrollados y su finalidad

En esta sección vamos a listar los principales prototipos que se han desarrollado en el proyecto, explicando su función, los elementos que se han implementado y detallar de ellos las funcionalidades más importantes.

Prototipado inicial de desarrollo

Los primeros prototipos que se hicieron fueron para probar las mecánicas de la jugabilidad y ver la viabilidad del proyecto. Es el equivalente al que realizaría una empresa, normalmente para mostrarse a los clientes, antes de aprobar y empezar con el proyecto en sí. En este caso, el papel de los clientes es realizado por el tutor del proyecto.

En estos prototipos, se implementó un movimiento básico del jugador y un comportamiento sencillo de los enemigos.

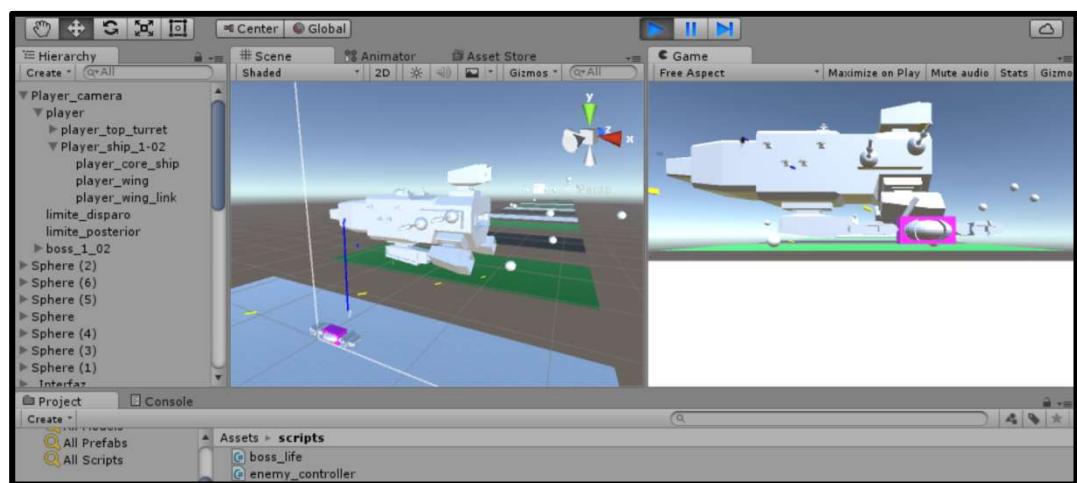


Ilustración 73 – Prototipo inicial de la idea del videojuego

El jugador podía moverse hacia los lados, arriba y abajo, y disparar. Los enemigos disparaban al jugador.

Movimiento del jugador

El primer prototipo de desarrollo del proyecto. Se empezó por el control de la nave del jugador y el movimiento de seguimiento de la cámara. La idea era conseguir que fuese lo más natural y fluido posible, sin que llegaría a marear al jugador.

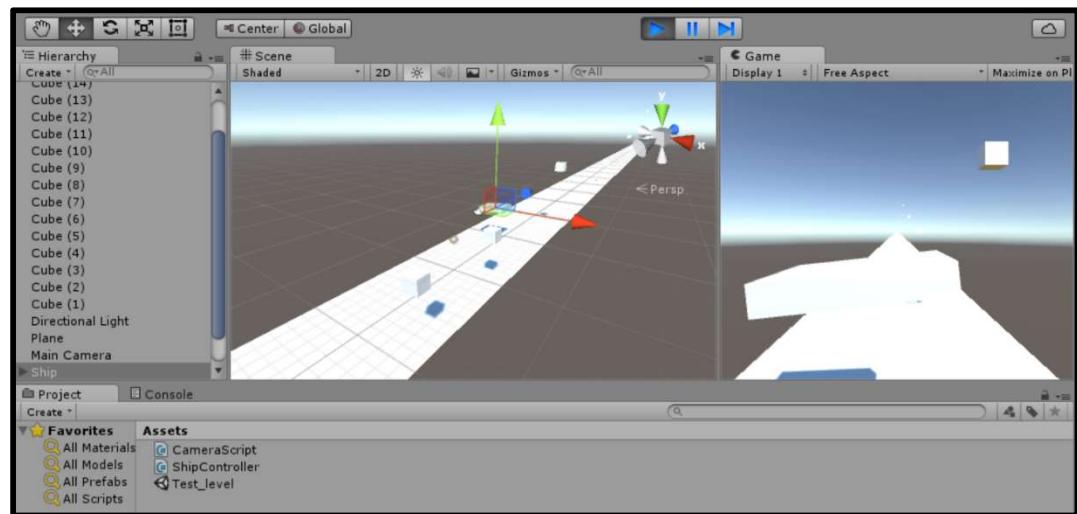


Ilustración 74 – Prototipo inicial de movimiento del jugador

```

46      // Update is called once per frame
47      void Update () {
48          _T.LookAt (_cameraDirection);
49          cameraOffset ();
50          screenCameraBorders ();
51      }

```

Ilustración 75 – Script de funcionamiento básico de la cámara

La cámara sigue a la nave del jugador en todo momento. Para que la sensación de movimiento fuese más natural, el seguimiento de la cámara no sea producía en el momento en el que el jugador se moviera, sino con cierto retraso y aceleración. También se limitó el movimiento de la cámara a lo largo del nivel estableciendo unos bordes límites.

Sistema de apuntado del jugador

Una vez con el control del movimiento del jugador implementado, se desarrolló el sistema de apuntado y disparo de la nave.

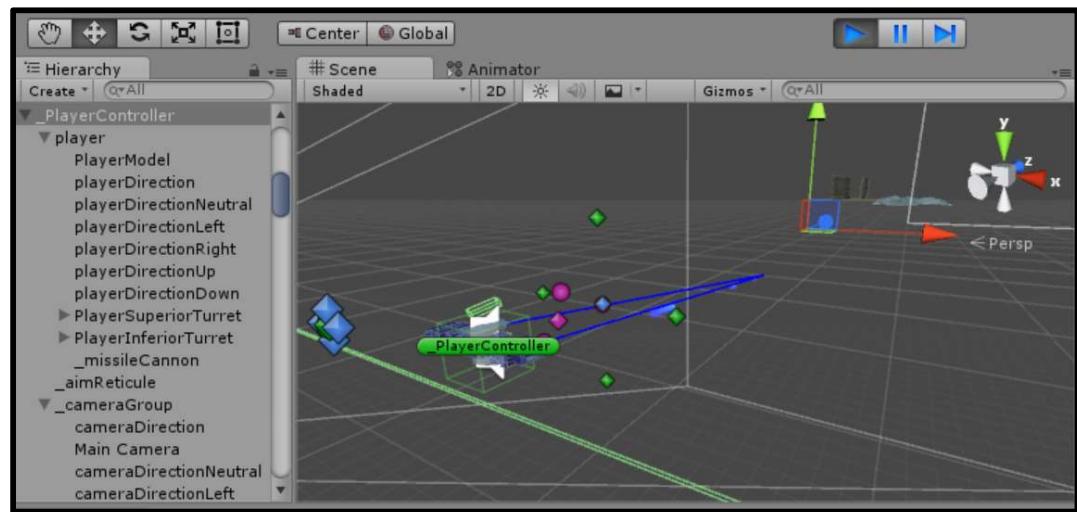


Ilustración 76 – Sistema de apuntado y de disparo

Para poder diseñar el sistema de apuntado se tuvo que resolver un problema derivado de la perspectiva, de la posición de dónde salen los disparos, que no está centrada, y el hecho de que fuera en tres dimensiones.

El indicador que indica dónde está apuntando el jugador se muestra en un entorno 2D, mientras que el lugar real al que se está disparando está ubicado en un entorno 3D. Este hecho genera confusión al jugador sobre el lugar a dónde se está apuntando. Además, como la posición de la nave y, por tanto, de los cañones de dónde salen los disparos, no está centrada, también afecta a este fenómeno visual. Para resolver esta situación se recurrió a lo siguiente:

- La interfaz consistiría en un ícono 2D. Este ícono se colocaría en la posición X e Y, sin tener en cuenta la profundidad, del elemento al que se está apuntando.
- Para detectar qué elemento se está apuntando se lanza un raycast [203] desde la cámara hacia el apuntador. Si colisiona con un objeto se ubica el ícono de la interfaz en la posición XY del objeto con respecto a la cámara. Para evitar que no siempre detecte algún elemento y que se

produczcan fallos, se coloca una caja de colisiones a cierta distancia del jugador que hace de barrera límite para el raycast⁷⁹.

```
if (Physics.Raycast (_aimRay, out hit, Mathf.Infinity, _layerMask)) {
    objectHit = hit.point;
    checkFiringTurrets (objectHit);
    checkMissiles (objectHit);
}
```

Ilustración 77 - Código que calcula la posición de colisión del raycast

Una vez obtenida la posición en 2D y 3D, se orientan los cañones de la nave para que apunten y disparen en esa dirección.

```
165 void checkFiringTurrets (Vector3 point)
166 {
167     if ((superiorTurret.position.y < (point.y + 3))) {
168         _superiorCanFire = true;
169         superiorQ = Quaternion.LookRotation(point - superiorTurret.position);
170         superiorTurret.rotation = Quaternion.RotateTowards(superiorTurret.rotation, superiorQ, turretRotateSpeed * Time.deltaTime);
171     } else {
172         superiorTurret.LookAt (superiorTurretCannon);
173         _superiorCanFire = false;
174     }
175
176     if ((inferiorTurret.position.y > (point.y - 3))) {
177         _inferiorCanFire = true;
178         inferiorQ = Quaternion.LookRotation(point - inferiorTurret.position);
179         inferiorTurret.rotation = Quaternion.RotateTowards(inferiorTurret.rotation, inferiorQ, turretRotateSpeed * Time.deltaTime);
180     } else {
181         inferiorTurret.LookAt (inferiorTurretCannon);
182         _inferiorCanFire = false;
183     }
184 }
```

Ilustración 78- Script que apunta los cañones de la nave al punto de disparo obtenido

Enemigos básicos y sus ataques

Con los controles del jugador implementados, llegó el turno de crear a los enemigos.

⁷⁹ El raycast se muestra representado en la ilustración 43 como dos rayos azules. Ambos rayos convergen en el punto en el punto de colisión con el objeto detectado. En este caso, como no hay ningún objeto en pantalla está chocando con la barrera límite.

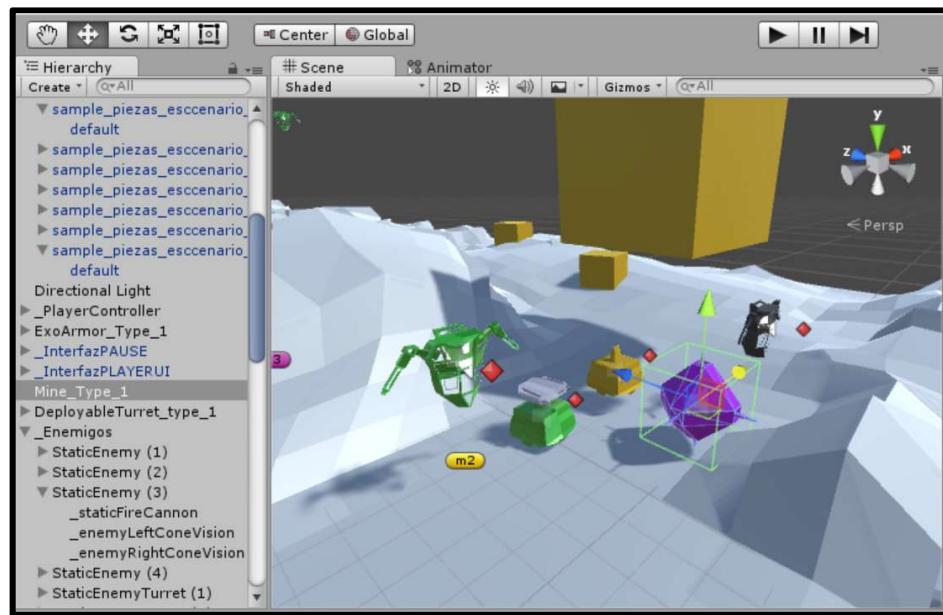


Ilustración 79 – Conjunto de los primeros enemigos diseñados

Inicialmente se planteó diseñar 2 tipos de enemigos distintos. Los de tipo estático, como torretas y bombas flotantes, y los de tipo dinámico que se movían por el escenario, como robots que flotaban y tanques.

Los enemigos atacaban buscando la posición en el mundo 3D de la nave del jugador y disparando ese punto.

Implementación básica de interfaz y menús

Se diseñó un sistema de interfaz básico para construir los menús y la pantalla de pausa, usando para ello el sistema de interfaz de Unity, llamado canvas⁸⁰, y sus características.

⁸⁰ El canvas [252] es la herramienta de Unity que permite crear todas las interfaces del juego. También funciona en base a componentes.

Combinación de los anteriores prototipos: Vida del jugador y de los enemigos

Para terminar la base de la jugabilidad, solo faltaba implementar la vida del jugador y que se pudiesen derrotar a los enemigos. Así que se combinaron todos los prototipos anteriores y se diseñó un sistema de puntos de vida para el jugador y para los enemigos.

Para el jugador se diseñó un sistema híbrido de puntos de vida. Una parte de ellos se regenerarían con el tiempo en forma de un escudo y la otra parte serían los puntos de vida de la nave, que una vez llegarán a cero sería destruida.

```
59      // Update is called once per frame
60      void Update ()
61      {
62          life ();
63          shields ();
64          invulnerabilityStateCalculator ();
65      }
```

Ilustración 80 – Bucle principal del sistema de puntos de vida del jugador

Los enemigos tendrían solamente puntos de vida, y la cantidad de puntos de vida variaría en función del tipo de enemigo.



Ilustración 81 – Explosión al ser destruido un enemigo con el sistema de vidas ya implementado

Para poder crear el sistema de puntos de vida, se tuvo que trabajar con las físicas de Unity. Sin entrar en mucho detalle se va a explicar a grandes rasgos cómo funciona el sistema diseñado.

Para la nave del jugador se optó por usar un RigidBody y hacerlo kinemático⁸¹. De esta manera se detectaban las colisiones del jugador con otras cajas de colisión, pero no se vería afectado por las fuerzas que resultarían de esa colisión. Así, al chocar con un objeto no se quedaría quieto o cambiaría de rumbo, simplemente pasaría a través de él.

Los ataques del jugador y de los enemigos también funcionarían igual, serían RigidBodies y contarían con su propia caja de colisiones. Con esto se podría gestionar la velocidad y comportamiento de cada uno de los ataques de manera independiente, mediante el uso de prefabs. Los enemigos y la nave del jugador instanciarían sus ataques de estos prefabs almacenándolos como variable en sus scripts.

⁸¹ Descripción en el manual de Unity de los objetos kinemáticos [250].

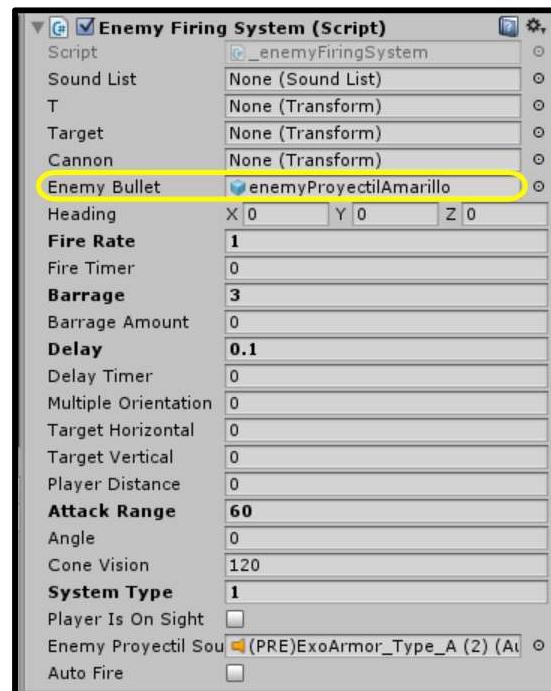


Ilustración 82 – Captura de como se muestra el script de ataque de los enemigos.⁸²

Con este sistema, los enemigos solo necesitarían una caja de colisiones para detectar los ataques recibidos.

```

127 void OnCollisionEnter (Collision col)
128 {
129     Debug.Log (col.gameObject.name);
130     if (col.gameObject.tag == "proyectil") {
131         recibeDamage (col.gameObject.GetComponent<enemyStandardProyectil> ()._damage);
132         damageInvulnerability (0);
133     }
134 }
```

Ilustración 83 – Parte del código que detecta la colisión de los ataques enemigos en el jugador

⁸² Se puede ver destacado en amarillo como está guardado el prefab del ataque en una variable.

```

void shields ()
{
    if (_shields < _MaxShields) {
        _shieldTimer -= Time.deltaTime;
        if (_shieldTimer <= 0.0f) {
            rechargeShield ();
        }
        activeShield ();
    }
}

void activeShield ()
{
    if (_shields == 0) {
        col_shields.enabled = false;
        col_ship.enabled = true;
    } else if (_shields == 1) {
        col_shields.enabled = true;
        col_ship.enabled = false;
    }
}

```

Ilustración 84 – Funciones principales que controlan los escudos del jugador

Por último, se mejoraron los sistemas de ataque y apuntado de los enemigos y del jugador para hacerlos más dinámicos y parametrizables. Para ello, se diseñó un sistema de apuntado dinámico, con varias opciones de ataque que se podían modificar y alterar al gusto. Desde la velocidad de disparo y la cantidad de ataques hasta la elección de dónde y cómo querían disparar al jugador.

Movimiento de los enemigos

El siguiente paso fue desarrollar más los distintos tipos de enemigos. Se hicieron varias mejoras y se diseñaron dos nuevos sistemas de movimiento.



Ilustración 85 – Componente Script de la nueva forma de movimiento lineal

Uno de ellos era un sistema de desplazamiento lineal para los enemigos que se desplazan con movimientos simples, como los de tipo tanque.

```

15 // Update is called once per frame
16 void Update () {
17     _T.position = Vector3.MoveTowards (_T.position, _destiny.position, _Speed * Time.deltaTime);
18 }
19 }
```

Ilustración 86 – Código que permite desplazar linealmente un objeto entre dos puntos

El otro sistema, más complejo, se creó para ser lo más parametrizable posible.

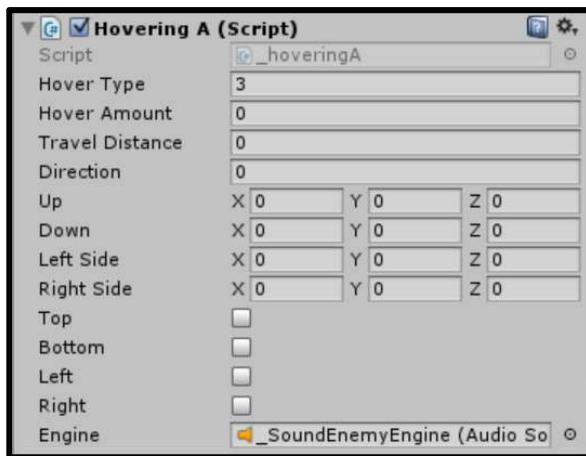


Ilustración 87 – Componente script de la nueva forma de movimiento “hover”, más parametrizable

Permitía a los enemigos desplazarse cortas distancias mediante pequeños movimientos, siguiendo diferentes patrones.

```

void type0 ()
{
    calculateDirection ();
    if (_top) {
        travelDistance = Vector3.Distance (_T.position, up);
        _T.position = Vector3.MoveTowards (_T.position, up, travelDistance * Time.deltaTime);
        if ((travelDistance) < 0.2f) {
            direction = Random.Range (0, 3);
            _top = false;
            _engine.Play ();
        }
    }
    if (_bottom) {
        travelDistance = Vector3.Distance (_T.position, down);
        _T.position = Vector3.MoveTowards (_T.position, down, travelDistance * Time.deltaTime);
        if ((travelDistance) < 0.2f) {
            direction = Random.Range (0, 3);
            _bottom = false;
            _engine.Play ();
        }
    }
}
```

Ilustración 88 – Código de una de las variantes de movimiento del nuevo tipo de movimiento “hover”

Desplazamiento por un nivel, versión básica

Una de las mecánicas que más tiempo llevó definir fue el desplazamiento de la nave del jugador a través del nivel. Se plantearon varias ideas, pero la que finalmente se implementó fue la siguiente.

Para recorrer un nivel, el jugador debería poder navegar por distintos waypoints, permitiendo no recorrerlo de forma lineal mediante giros o cambiando de altura.

Por eso se decidió desarrollar un sistema que se encargaba de calcular puntos de ruta entre estos waypoints, pudiendo personalizar el recorrido entre ellos creando curvas. Para conseguirlo, se tuvieron que implementar varias mecánicas en el sistema que no formaban parte de Unity. Por tanto, se tuvo que trabajar con el código y lenguaje interno de Unity. Tras buscar tutoriales que ayudaran al respecto [204], se desarrolló dicho sistema.

Este nuevo sistema se basaba en las curvas bezier [205] [206], que permite calcular curvas con un punto inicial y otro final, y dos puntos de pivote que definen la forma de la curva. Para los puntos de ruta, se utilizaron una serie de fórmulas para obtener puntos intermedios. Una vez conseguidos estos puntos se guardaba su posición y orientación en una lista.

```

63     void calculatePoints(int segmentQuantity = 100) {
64         length = 0;
65         float[] arcLengths = new float[segmentQuantity];
66         Vector3 oldPoint = getBezierCurve (0);
67
68         for (int x = 1; x < arcLengths.Length; x++) {
69             Vector3 point = getBezierCurve ((float)x / segmentQuantity);
70             arcLengths [x] = Vector3.Distance (oldPoint, point);
71             length += arcLengths [x];
72             oldPoint = point;
73         }
74         //create our points array
75         arrayPoints = new Vector3[segmentQuantity];
76         //arc index is where we got up to in the array to avoid the Shlem
77         int arcIndex = 0;
78         float walkLength = 0; //how far along the path we've walked
79         //segment Length
80         float segmentLength = length/segmentQuantity;
81         oldPoint = getBezierCurve(0);

```

Ilustración 89 – Parte del código para calcular los puntos intermedios de una curva

Una de las herramientas implementadas permitía visualizar el recorrido entre los waypoints mediante líneas en la pantalla del editor que se actualizaban en tiempo real.

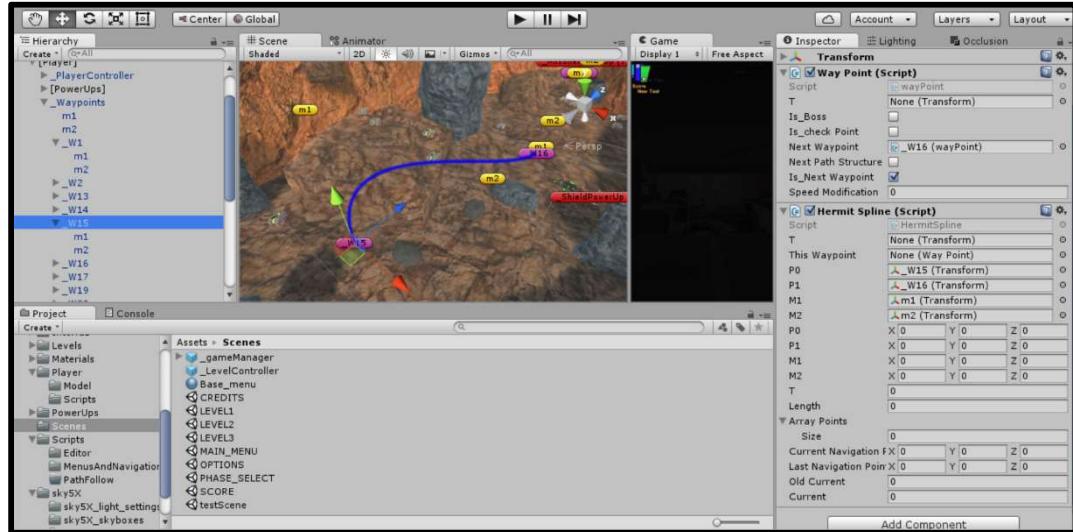


Ilustración 90 – Visualización del sistema de puntos de ruta. Camino entre dos Waypoints

El sistema luego se combinaba con el controlador de cada nivel, que se encargaba de calcular el camino y desplazar la nave del jugador entre cada punto de ruta, ajustando su velocidad y orientación.

```

99      void checkWaypointCompletion () {
100         if (_TPlayer.position == _currentSpline.currentNavigationPoint) {
101             _currentSpline.calculateNextPoint();
102         }
103         if (_TPlayer.position == _nextWaypoint.transform.position) {
104             calculateNextWayPoint ();
105         }
106     }
107
108     public void calculateNextWayPoint() {
109         _currentWaypoint = _nextWaypoint;
110         if (_currentWaypoint.is_checkPoint) {
111             _GameManager.Instance.SaveOnGoingLevelScoreData ();
112         }
113         _currentSpline = _currentWaypoint.GetComponent<HermitSpline> ();
114         getNextWaypoint ();
115     }

```

Ilustración 91 – Parte del código del controlador de un nivel. Cálculo de waypoints

Lo bueno de dedicar tanto tiempo a implementar este sistema fue que era totalmente parametrizable y reutilizable. Así, el código se podía modificar o usar para cubrir distintas necesidades, ya sea en este proyecto o en proyectos futuros.

```

void travel () {
    //calculateCurveTPercentage();
    rotatePlayerPath ();
    playerStep ();
    checkWaypointCompletion ();
}

void rotatePlayerPath() {
    Vector3 _direction = _currentSpline.currentNavigationPoint - _TPlayer.position;
    if (_direction != Vector3.zero) {
        _TPlayer.rotation = Quaternion.Slerp (_TPlayer.rotation ,Quaternion.LookRotation (_direction), Time.deltaTime);
    }
}

void playerStep() {
    _TPlayer.position = Vector3.MoveTowards (_TPlayer.position,_currentSpline.currentNavigationPoint, Time.deltaTime);
}

```

Ilustración 92 – Parte del código del controlador de un nivel. Funciones principales.

En esta fase también se empezaron a realizar las pruebas de control de movimiento.

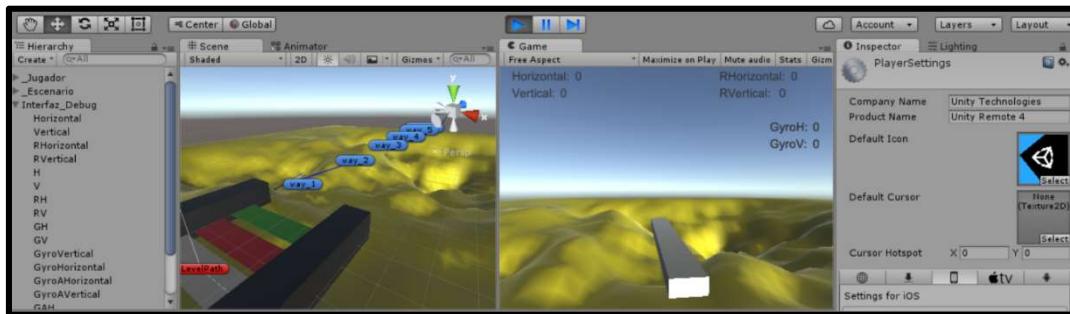


Ilustración 93 – Pruebas realizadas para adaptar el control de movimiento mediante Unity Remote4

Se estudió e investigó la forma de adaptar el manejo del juego a un control por movimiento mediante giroscopio y detección táctil. Se creó un prototipo que serviría a lo largo de todo el desarrollo para adaptar el control de movimiento a AppleTV.

La función de este prototipo consistía en aprender a configurar los inputs tipo *touch* y *giroscopio* del mando “siri”. El prototipo creaba una escena sencilla formada por cajas y planos, para poder ubicar y orientar mejor el movimiento. El objeto que se controlaba con el mando se podía mover y rotar en dos ejes, el X y el Y. También se implementó una sencilla interfaz para mostrar los valores que se estaban usando para controlar el movimiento y el giro. Así, la escena servía para poder probar, configurar y debuggear el sistema de control que se iba a usar en el juego.

Inicialmente se realizaron las pruebas de control con un dispositivo Android. Después de realizar los primeros ajustes, se utilizó un iPad para adaptar el control al sistema iOS.

Diseño de unidades y escenario

Con todas las características principales de la jugabilidad terminadas, el siguiente paso fue diseñar y modelar los elementos del juego, así como definir su aspecto visual. A continuación se muestran algunas ilustraciones con modelos de distintos elementos del juego.

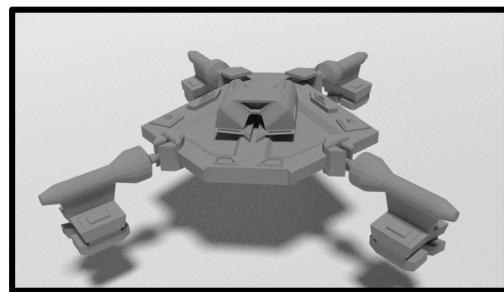


Ilustración 94 – Modelo de un nuevo enemigo, Saeta

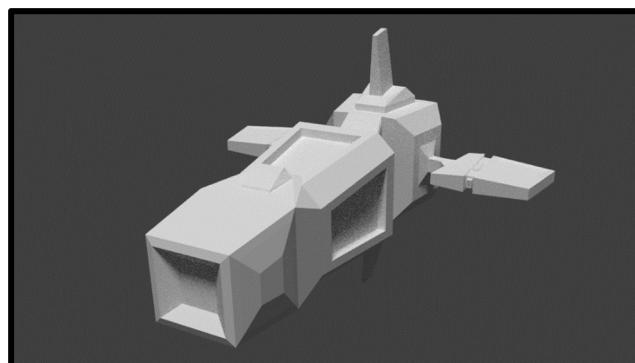


Ilustración 95 – Modelo de un nuevo enemigo, Caza

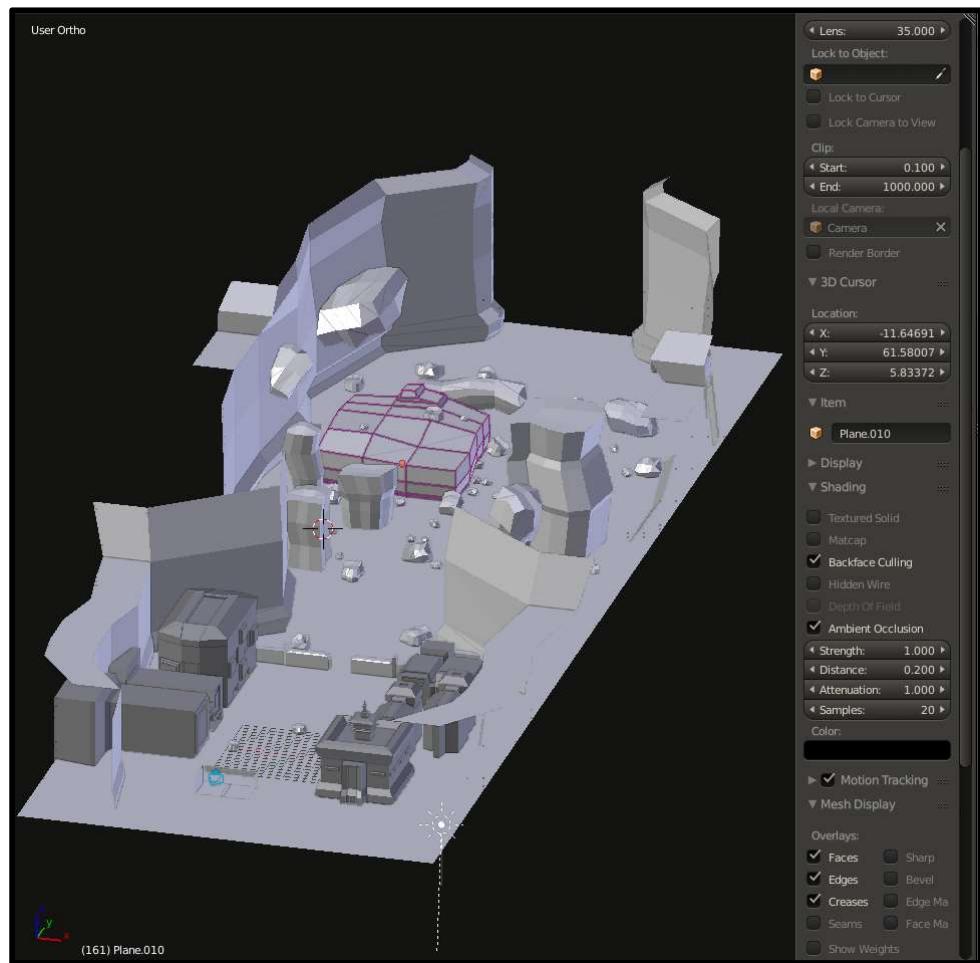


Ilustración 96 – Creación de la primera zona del primer nivel.



Ilustración 97 – Referencia inicial del estilo visual y carga de primeros modelos.

Desarrollo de distintos tipos de enemigos, mecánicas específicas de cada uno de ellos y tipos de ataque

En este prototipo se incluyeron nuevos tipos de enemigos y se mejoraron y definieron todavía más las mecánicas de ataque para cada uno de ellos. Estos enemigos ofrecían mayor variedad de situaciones y tenían patrones de ataque distintos.

Los nuevos enemigos diseñados fueron:

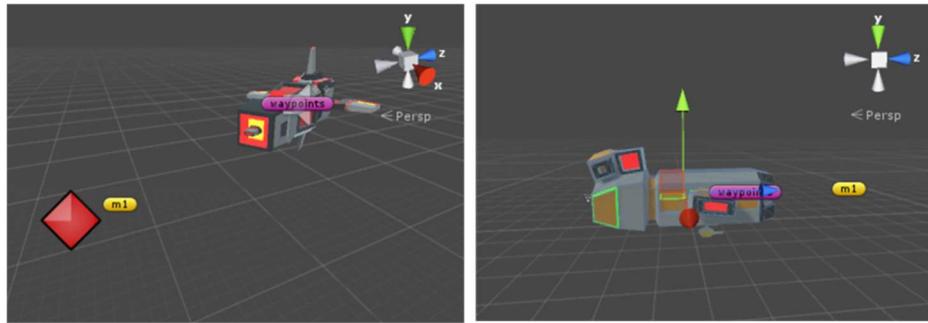


Ilustración 98 - Nuevos tipos enemigos. El caza y la corbeta.

- Dos tipos de naves nuevas, unas rápidos y pequeñas, y las otras grandes y lentas. Para estas naves se diseñó un sistema de movimiento por el escenario basado en el sistema de movimiento del jugador.



Ilustración 99 – Nuevo código y componentes para los enemigos



Ilustración 100 – Nuevo enemigo, Saeta

- Un vehículo híbrido que se podía pegar a cualquier superficie y atacar al jugador en cualquier posición.

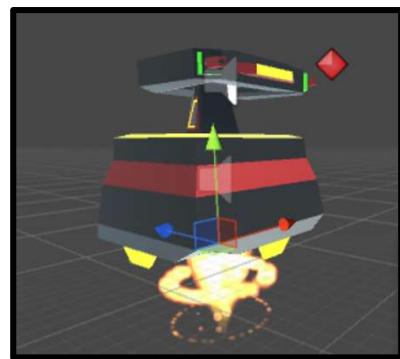


Ilustración 101 – Nueva funcionalidad a las torretas, capacidad de despliegue

También se incluyó una nueva mecánica a las torretas. Algunas de ellas podían ser lanzadas desde el cielo hasta aterrizar en una ubicación, pudiendo sorprender al jugador en cualquier momento.

```

37 // Update is called once per frame
38 void Update ()
39 {
40     if (_Turret != null) {
41         if ((_Turret.position.z - _Player.position.z) < _PlayerDistanceActivaction) {
42             if (_startDescend) {
43                 travelDistance = Vector3.Distance (_Turret.position, _end.position);
44                 if (travelDistance < minTranslateDistance) {
45                     travelDistance = minTranslateDistance;
46                 }
47                 _Turret.position = Vector3.MoveTowards (_Turret.position, _end.position, travelDistance * Time.deltaTime);
48                 if (_Turret.position == _end.position) {
49                     _engine.Stop ();
50                     _startDescend = false;
51                     Destroy (_particlesEngine.gameObject);
52                 }
53             }
54         }
55     }
56 }
57 }
58 }

```

Ilustración 102 – Código de funcionamiento del despliegue de las torretas

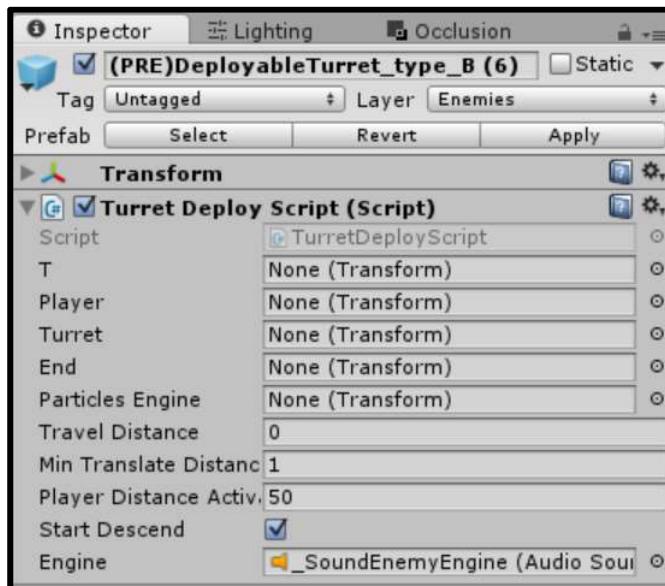


Ilustración 103 – Componente script nuevo para el despliegue de las torretas

Combinación de los anteriores prototipos: Adaptación de los controles a appleTV

Con las nuevas mejoras introducidas se combinaron todos los prototipos en uno. A su vez, se empezó a trabajar en la adaptación final de los controles del juego a AppleTV.

```

132     void checkInput ()
133     {
134         // PRUEBAS PARA EL CONTROL DE MOVIENTO POR SIRI O IPAD
135         if (Input.GetKeyDown(KeyCode.G)) {
136             is_Motion_Input_Active = !is_Motion_Input_Active;
137             if (is_Motion_Input_Active) {
138                 Debug.Log ("Activado modo giroscopio");
139                 //UnityEngine.Apple.TV.Remote.reportAbsoluteDpadValues = true;
140                 UnityEngine.Apple.TV.Remote.allowExitToHome = false;
141             }
142             else {
143                 Debug.Log ("Desactivado modo giroscopio");
144                 //UnityEngine.Apple.TV.Remote.reportAbsoluteDpadValues = false;
145             }
146         }
147     }

```

Ilustración 104 – Código para el prototipo específico.

Usando el prototipo previamente creado se empezaron a realizar pruebas más exhaustivas del control para que, entre otras cosas, funcionase bien con la interfaz.

Luego se volvieron a realizar las pruebas con dispositivo Android e iOS para comprobar los últimos cambios.

Una vez hechas estas pruebas con éxito, se dedicó un periodo de tiempo a terminar de adaptar el control a AppleTV en el videojuego para poder implementarlo correctamente.

```

142     void calculateAimPosition ()
143     {
144         if (_playerMovement.is_Motion_Input_Active) {
145             foreach (Touch _touch in Input.touches) {
146                 //_screen += new Vector3 (_touch.deltaPosition.x,_touch.deltaPosition.y, 0);
147                 _screen.x += _touch.deltaPosition.x;
148                 _screen.y += _touch.deltaPosition.y;
149                 //_screen = new Vector3 (_touch.position.x,_touch.position.y, 0);
150                 if (_touch.tapCount >= 3) {
151                     _GameManager.Instance.pauseGame ();
152                 }
153             }
154             _aimRay = Camera.main.ScreenPointToRay (_screen);
155         }else {
156             _aimRay = Camera.main.ScreenPointToRay (Input.mousePosition);
157         }
158         if (Physics.Raycast (_aimRay, out hit, Mathf.Infinity, _layerMask)) {
159             objectHit = hit.point;
160             checkFiringTurrets (objectHit);
161             checkMissiles (objectHit);
162         }
163     }
164

```

Ilustración 105 – Adaptación del sistema de apuntado al control táctil del mando Siri

Enemigos de tipo Boss y sub-boss, mecánicas

La última mecánica jugable por implementar era el enfrentamiento contra los bosses. Se desarrollaron nuevos métodos de comportamiento y ataque para estos nuevos tipos de enemigos.

También se diseñaron nuevos sistemas de puntos de vida específicos para ellos. El primero de ellos se basaba en que los bosses sólo podían ser dañados si se atacaba a ciertas zonas. El otro consistía en un sistema acumulativo de puntos de vida, que juntaba la vida de distintos enemigos o partes como una sola, teniendo que acabar con todas las partes para derrotar al boss.

Combinación de los anteriores prototipos: Efectos especiales, materiales y mejoras de todo lo diseñado hasta el momento.

Este prototipo se centró en diseñar y concretar el aspecto visual del juego, generando los materiales, las texturas, la iluminación, efectos, etc.

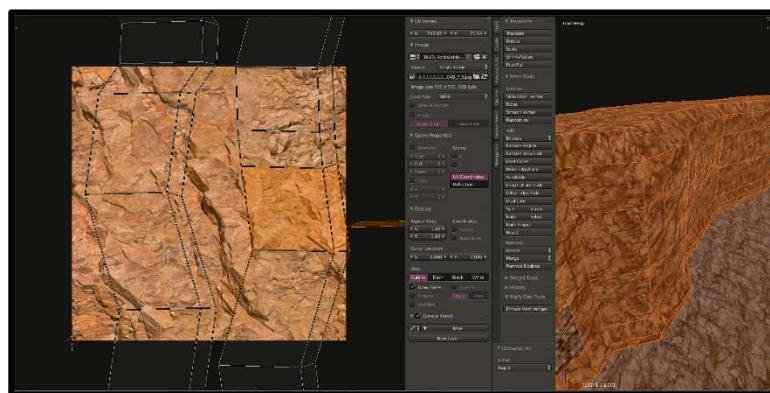


Ilustración 106 – Trabajo de cálculo de los UV y texturizado del primer nivel

Se trabajó con Blender en todos los modelos, calculando sus UV⁸³ para mapear los distintos materiales y texturas.

⁸³ UV mapping [247] es el proceso de proyectar un modelo 3D en una superficie 2D para poder realizar el texturizado. Sería parecido a coger un cubo recortado de papel e ir desplegando todas sus caras en proceso inverso en el que se había montado, hasta tener todas las caras sobre el mismo plano.

Se crearon, diseñaron y gestionaron todos los efectos especiales del juego, como explosiones, sistemas de partículas y los ataques del jugador y enemigos.



Ilustración 107 – Mejoras en el aspecto visual. Materiales y texturas

También se aprovechó este prototipo para mejorar la optimización del juego, controlando las occlusiones y lo que debía o no debía calcular Unity y dibujar por pantalla.

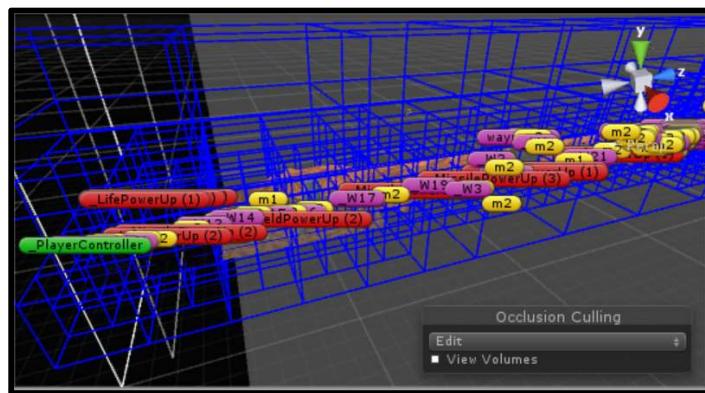


Ilustración 108 – Cálculos de las occlusiones del primer nivel

```

239     private void sortEnemiesByDistance ()
240     {
241         enemies.Sort (delegate(Transform item1, Transform item2) {
242             return (item1.position.z.CompareTo (item2.position.z));
243         });
244     }
245
246     private void checkEnemiesToActive ()
247     {
248         if (enemies.Count != 0) {
249             if ((enemies [0].position.z - _TPlayer.position.z) < 130.0f) {
250                 enemies [0].gameObject.SetActive (true);
251                 enemies.RemoveAt (0);
252             }
253         }
254     }
255

```

Ilustración 109 – Código que activa en la escena a los enemigos que están cerca del jugador

Implementación de menús, navegación entre menús, interfaz y sistemas de ficheros para el guardado de datos

En este prototipo se mejoraron y terminaron todos los menús y la navegación entre ellos.

```

477     public void LoadOnGoingLevelData ()
478     {
479         if (File.Exists (Application.persistentDataPath + "[OnGoing][levelInfo]" + SceneManager.GetActiveScene ().name + ".")
480             BinaryFormatter bf = new BinaryFormatter ();
481             FileStream file = File.Open (Application.persistentDataPath + "[OnGoing][levelInfo]" + SceneManager.GetActiveSc
482             OnGoingLevelData data = (OnGoingLevelData)bf.Deserialize (file);
483             file.Close ();
484             levelScore = data.score;
485             _pathManager._currentWaypoint = GameObject.Find(data._current).GetComponent<wayPoint>();
486             _TPlayer.GetComponent<_PlayerLife> () ._continues = data.continues;
487         }
488     }

```

Ilustración 110 – Código que permite cargar la puntuación y posición de un checkpoint de un nivel

Además, se creó todo el sistema de ficheros para poder guardar y cargar datos dentro del juego, que permitían hacer el sistema de checkpoints y de puntuación del juego.

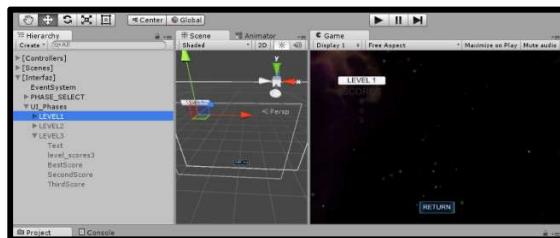


Ilustración 111 – Creación de la interfaz de la pantalla de selección de niveles

Combinación de los anteriores prototipos, pulidos y arreglos. Versión del prototipo final.

VEGA:FACTIONS WAR

Ilustración 112 – Título del videojuego creado, Vega: Factions War

Se creó un último prototipo final con todos los arreglos y mejoras finales. Se pulieron algunas mecánicas y se dejó preparada una versión inicial de lo que sería Vega: Factions War.

De Unity a AppleTV

Para pasar el resultado final del videojuego a formato aplicación para AppleTV, se tuvieron que seguir una serie de pasos.

Primero se terminó de configurar el proyecto para adaptarlo a AppleTV. Despues, se ha hecho una primera compilación del proyecto en Unity, seleccionando la configuración específica para la plataforma de Apple.

Posteriormente, con el resultado de la compilación se usó Xcode en un dispositivo iOS para terminar de crear la aplicación en su versión AppleTV.

CONCLUSIONES

Producto final

Al final de su desarrollo, Vega:Factions wars cuenta con todo los elementos necesarios que constituyen un videojuego. Tiene varios niveles a superar por el jugador y diversos enemigos a los que enfrentarse en cada uno de ellos. Además, el videojuego está preparado y diseñado para poder ampliarlo y continuar su desarrollo si se desea. A continuación, antes de explicar en detalle las conclusiones del trabajo, se hará un resumen del contenido y características de Vega:Factions wars.

- El juego dispone de un total de 13 enemigos, cada uno de ellos con características y comportamientos diferenciados que aportan variedad a la experiencia jugable. En las siguientes ilustraciones se muestran los distintos enemigos.



Ilustración 113- caza

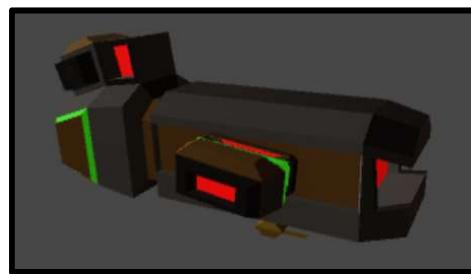


Ilustración 114- corbeta

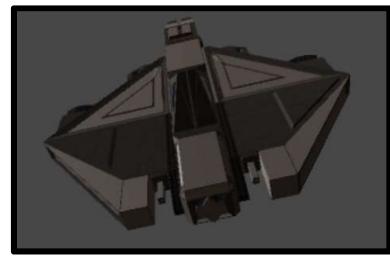


Ilustración 115 - Fragata



Ilustración 116 – Exoarmadura de combate



Ilustración 117 – Mina de proximidad



Ilustración 118 - Saeta



Ilustración 119 – Tanque aerodeslizador



Ilustración 120 – Torreta deplegable

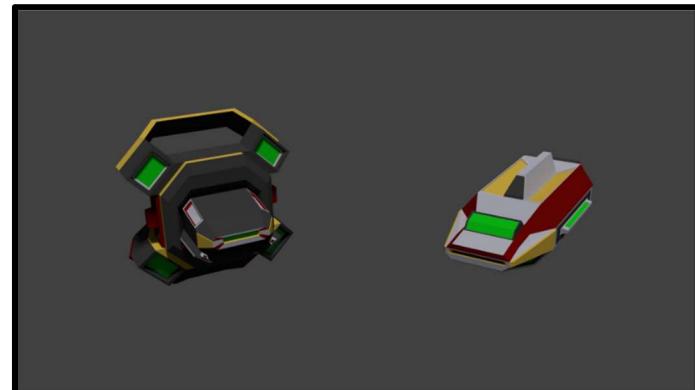


Ilustración 121 – Cruceros de batalla



Ilustración 122 – Jefes finales

- Además, están en desarrollo enemigos adicionales como futuras nuevas incorporaciones.



- El juego también cuenta con tres diferentes niveles jugables. Cada uno de ellos tiene una ambientación y características distintas, aumentando la variedad de situaciones y sensaciones que experimentará el jugador.

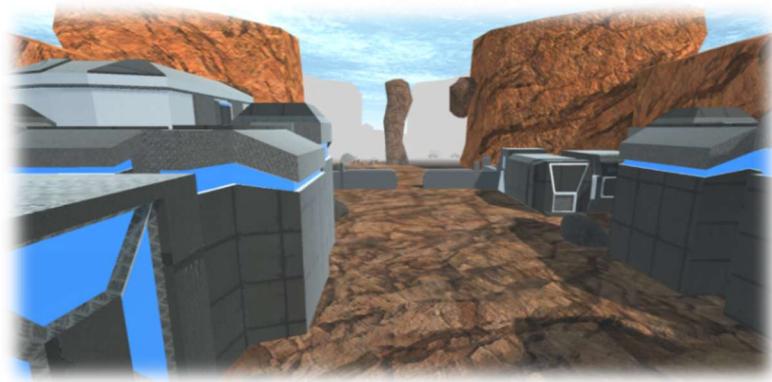


Ilustración 123 – Primer nivel del juego, Huida

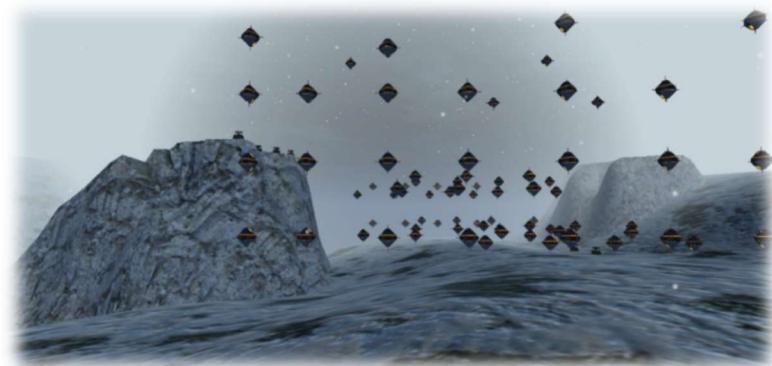


Ilustración 124 – Segundo nivel del juego, Ataque calculado

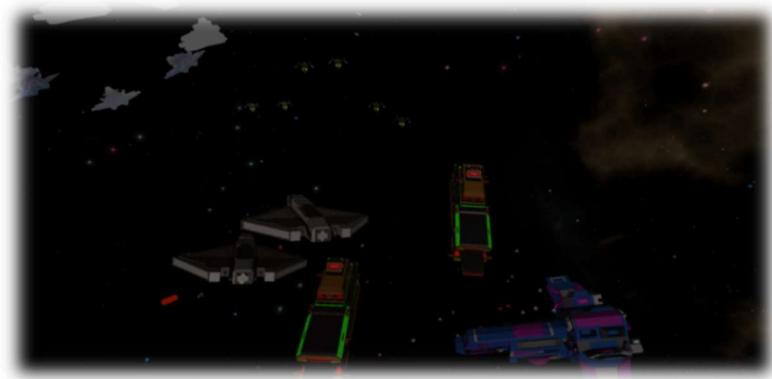


Ilustración 125 – Tercer nivel del juego, Guerra de facciones

- Además, el juego se ha diseñado con una serie de mecánicas jugables sencillas y adictivas.



Ilustración 126 – Segundo nivel del videojuego, entrando en la base de los enemigos

Conclusiones

Este trabajo ha consistido en el desarrollo de un videojuego para la plataforma AppleTV, siguiendo para ello una serie de objetivos propuestos.

Se ha organizado y planificado el proceso de creación del videojuego. Durante el periodo de desarrollo del proyecto se ha intentado mantener la gestión del mismo. Aunque no ha sido posible hacer un seguimiento riguroso y constante en todo momento, la planificación realizada ha facilitado mucho el desarrollo, permitiendo reajustar el proyecto y las tareas según ha sido necesario.

Se ha conseguido crear un videojuego, trabajando para ello en todas las disciplinas necesarias para terminarlo. El rango de tareas en las que se ha trabajado es amplio: la programación y creación de todo el código del videojuego, diseño de su estructura y conceptos y, finalmente, el modelado y texturización de todos los elementos que forman parte del videojuego, definiendo también el aspecto artístico del mismo. Es importante destacar que sólo ha habido un único autor material⁸⁴ durante la creación del videojuego, siendo el responsable de todo el trabajo. Además, el trabajo se ha llevado a cabo con la intención de mantener dentro del proyecto una estructura bien construida y óptima. Aunque es ciertamente mejorable, se ha alcanzado un cierto nivel de calidad en la organización del proyecto y del código, y se tiene la intención de seguir mejorando este aspecto en el futuro. Se ha usado para ello diversas técnicas de programación y diseño, como el uso y nombramiento adecuado de las variables para una correcta lectura e interpretación de su uso, la construcción de métodos de tamaño reducido y función específica para facilitar la lectura e

⁸⁴ El tutor ha participado en la toma de decisiones y es también partícipe de la autoría conceptual del proyecto.

interpretación del código, y en general, crear una relación clara entre los diversos componentes y métodos dentro del proyecto.

La herramienta usada para el desarrollo del videojuego ha sido Unity. Durante la elaboración del juego, se ha aprendido a usar Unity y sus distintas funcionalidades, alcanzando un aceptable grado de entendimiento de sus características y de optimización en su forma de trabajar. Aunque no ha sido posible aprender y dominar todo sobre Unity, los conocimientos adquiridos sobre el funcionamiento de la herramienta y en general del desarrollo de un videojuego han sido muy útiles, y servirán en un futuro para realizar otros proyectos o la obtención de un puesto de trabajo.

Adicionalmente, se ha realizado un trabajo de investigación y análisis sobre el dispositivo AppleTV y sus características y servicios. Gracias a esto se ha comprendido la utilidad de la plataforma y la intención de Apple al desarrollarla. Se ha realizado también una estimación del valor que proporciona como producto y como plataforma de desarrollo. Aunque positiva, en el sector de los videojuegos aún tiene que realizar progresos. El futuro del dispositivo es prometedor, y vale la pena invertir dinero y tiempo si el desarrollador tiene unos objetivos claros.

También, se ha hecho un estudio de mercado de los videojuegos que han salido para AppleTV. Durante el mismo se ha teniendo en mente evaluar la viabilidad de sacar nuevos videojuegos en la plataforma. Los resultados han sido claros, el mercado en la plataforma está creciendo, pero aún necesita un empujón para hacerla viable en el mercado al mismo nivel que los dispositivos móviles o las videoconsolas.

En cuanto al desarrollo del videojuego Vega:Facons wars, se ha realizado teniendo en cuenta los requisitos y limitaciones de la plataforma

AppleTV. El control del videojuego se ha diseñado desde el principio con el mando remoto Siri en mente, ajustando los movimientos del jugador a las posibilidades de Siri.

Ya que no existía una documentación oficial apropiada de cómo programar el control del mando Siri, se ha realizado un trabajo de investigación para comprender como funcionaba el mando y sus posibilidades de adaptación al control del videojuego mediante Unity. Existían varias formas de adaptar el control del mando y se han tenido que probar y testear todas ellas hasta dar con una opción que cumpliera con los requisitos de la jugabilidad.

Destacar también que se ha limitado el peso de la aplicación para facilitar su futura publicación en la App Store, utilizando el menor número de texturas, efectos y modelos posibles. En este campo y en el de la optimización del videojuego dentro de Unity es dónde más problemas ha habido con respecto al diseño. A pesar de haber conseguido ciertos resultados, la falta de conocimientos sobre ambas materias, que pueden abarcar por su cuenta sendos trabajos de investigación, han limitado el desarrollo y optimización del proyecto en ambos frentes, no consiguiendo el grado de implementación deseado. No obstante, gracias a tener que enfrentarse a los problemas surgidos se han adquirido nuevos conocimientos y esto ha despertado un gran interés para seguir aprendiendo sobre ambas materias.

Por último, se ha adaptado el proyecto y se han seguido los pasos necesarios para que la aplicación funcione en un dispositivo AppleTV. Usando Unity y Xcode se ha compilado el proyecto y creado una aplicación que funciona en AppleTV de manera correcta. A nivel de desarrollador es interesante investigar sobre Xcode y cómo se podría optimizar la aplicación para un entorno iOS una vez el proyecto ha sido creado desde Unity.

Tras cumplirse los objetivos anteriormente citados se puede afirmar que se ha aprendido mucho durante la elaboración de este trabajo final de grado. A nivel personal, he disfrutado mucho creando el videojuego y enfrentándome a todos los problemas que han surgido. Han existido momentos de desesperación cuando el código no funcionaba bien o no se conseguía implementar algo o, cuando cosas que antes funcionaban luego dejaban de funcionar. Especial mención a la optimización del juego y a la adaptación del mando Siri a Unity, que han traído varios quebraderos de cabeza y, a pesar de funcionar correctamente, el resultado final se puede mejorar. A pesar de todos estos problemas, he quedado satisfecho con el trabajo realizado. He aprendido mucho y tengo ganas de seguir formándome.

Para terminar, quisiera expresar el orgullo que me produce el haber desarrollado mi proyecto de inicio a fin, observando el fruto de mi trabajo en Vega: Factions War.

DOCUMENTACIÓN

Definiciones

API: API (Application Programming Interface o aplicación de interfaz para programación) [207] es una biblioteca de funciones, procedimientos que ayudan a desarrollar software. Básicamente funciona como una interfaz de uso y desarrollo entre el lenguaje de programación y el desarrollador, abstrayendo un nivel la implementación de bajo nivel del software, reduciendo la dificultad del trabajo.

App: Término abreviado para definir a las aplicaciones móviles o web [22]

Framework: “En el desarrollo de software, un framework o infraestructura digital, es una estructura conceptual y tecnológica de soporte definido, normalmente con artefactos o módulos concretos de software, que puede servir de base para la organización y desarrollo de software.” [208]

Infrared: Es un tipo de radiación electromagnética que no entra dentro del espectro visible que se encuentra por encima del color rojo (de ahí su nombre). Tiene una gran variedad de aplicaciones, se usa en dispositivos de visión nocturna, para la transmisión de datos, en astronomía, climatología, etc. [65]

Streaming: Acción de enviar (por parte de un proveedor) y recibir (por el usuario) constantemente contenido multimedia, como la retransmisión en directo por internet de un evento. [209]

Network: Se traduce en la palabra “red”. Una network en el ámbito de los ordenadores se refiere a la conexión y espacio de comunicación que se genera al conectar varios ordenadores. Una red de telecomunicaciones para compartir datos. La network más famosa conocida es Internet. [210]

Port: Hace referencia al termino Porting [211], que consiste en adaptar un programa para que funcione en un sistema operativo nuevo diferente al sistema original. Un Port de un juego es un videojuego que ha sido modificado para poder funcionar correctamente bajo las características del nuevo sistema.

Prototipo: Un prototipo [212] consiste en desarrollar una versión temprana e incompleta de un producto pero que cuenta con parte de sus funcionalidades principales.

Raycast: Tipo de intersección rayo-superficie usado en geometría computacional para obtener información y resolver problemas con los datos obtenidos. [203]

Texturizar: Texturizar [213] o texture mapping es el proceso que define sobre la superficie de un modelo 3D imágenes o colores.

Bibliografía

Referencias del documento e información adicional y enlaces de interés citados en esta memoria.

- [1] «flickr giuseppemilo,» [En línea]. Available:
<https://www.flickr.com/photos/giuseppemilo/12570992595/in/photolist-k9RC8D>.
- [2] «computer computingnow,» [En línea]. Available:
<https://www.computer.org/web/computingnow/archive/february2015-spanish>.
- [3] «domoticampus hogar-digital,» [En línea]. Available:
<http://www.domoticampus.com/hogar-digital/>.
- [4] «wiki multimedia,» [En línea]. Available:
<https://es.wikipedia.org/wiki/Multimedia>.
- [5] «pexels man-using-stylus-pen-for-touching-the-digital-tablet-screen,» [En línea]. Available: <https://www.pexels.com/photo/man-using-stylus-pen-for-touching-the-digital-tablet-screen-6335/>.
- [6] «eluniversal la-evolucion-del-entretenimiento,» [En línea]. Available:
<http://www.eluniversal.com.mx/articulo/techbit/2016/04/8/la-evolucion-del-entretenimiento>.
- [7] «elperiodico sector-videojuegos-espana-ventas-2015,» [En línea]. Available: <http://www.elperiodico.com/es/noticias/tecnologia/sector-videojuegos-espana-ventas-2015-5013529>.
- [8] «eltiempo universidades-colombianas-le-apuestan-a-la-educacion,» [En línea]. Available: <http://www.eltiempo.com/estilo-de>

vida/educacion/universidades-colombianas-le-apuestan-a-la-educacion/16148915.

- [9] «fundetec la-movilidad-cambia-los-habitos-de-consumo-del-ocio-digital,» [En línea]. Available: <http://www.fundetec.es/reportajes/la-movilidad-cambia-los-habitos-de-consumo-del-ocio-digital/>.
- [10] «interxion impulsando-el-entretenimiento-digital-en-2015,» [En línea]. Available: <http://www.interxion.com/es/blogs/2016/05/los-videojuegos-continuan-impulsando-el-entretenimiento-digital-en-2015/>.
- [11] «wiki youtube,» [En línea]. Available: <https://en.wikipedia.org/wiki/YouTube>.
- [12] «www.youtube.com,» [En línea]. Available: <https://www.youtube.com/>.
- [13] «wiki netflix,» [En línea]. Available: <https://es.wikipedia.org/wiki/Netflix>.
- [14] «netflix,» [En línea]. Available: <https://www.netflix.com/es/>.
- [15] «wikipedia Netflix,» [En línea]. Available: <https://es.wikipedia.org/wiki/Netflix>.
- [16] «netflix,» [En línea]. Available: <https://www.netflix.com/es/>.
- [17] «commons.wikimedia.org NintendoStack,» [En línea]. Available: <https://commons.wikimedia.org/wiki/File:NintendoStack.jpg>.
- [18] «flickr demonbaby,» [En línea]. Available: <https://www.flickr.com/photos/demonbaby/4099434138>.
- [19] «wiki genero_de_videojuegos,» [En línea]. Available: https://es.wikipedia.org/wiki/G%C3%A9nero_de_videojuegos.

- [20] «venturebeat,» [En línea]. Available:
<http://venturebeat.com/2015/02/13/making-money-doing-good-the-new-genre-of-world-games-like-never-alone-interview/>.
- [21] «wiki aplicación informática,» [En línea]. Available:
https://es.wikipedia.org/wiki/Aplicaci%C3%B3n_inform%C3%A1tica.
- [22] «wiki aplicación móvil,» [En línea]. Available:
https://es.wikipedia.org/wiki/Aplicaci%C3%B3n_m%C3%B3vil.
- [23] «commons.wikimedia.org Pitfalls-To-Avoid-In-Mobile-App-Design,» [En línea]. Available: <https://commons.wikimedia.org/wiki/File:8-UX-Pitfalls-To-Avoid-In-Mobile-App-Design.jpg>.
- [24] «wiki software,» [En línea]. Available:
<https://es.wikipedia.org/wiki/Software>.
- [25] «wiki application software,» [En línea]. Available:
https://en.wikipedia.org/wiki/Application_software.
- [26] «commons.wikimedia.org Escritorio_de_Google_Drive_2,» [En línea]. Available:
https://commons.wikimedia.org/wiki/File:Escritorio_de_Google_Drive_2.png.
- [27] «businessdegrees the-future-of-mobile-application,» [En línea]. Available:
<http://businessdegrees.uab.edu/resources/infographics/the-future-of-mobile-application/>.
- [28] «businessinsider,» [En línea]. Available:
<http://www.businessinsider.com/the-future-of-mobile-apps-2014-10>.

- [29] «opera web-apps-mobile-world,» [En línea]. Available: <http://www.opera.com/blogs/news/2016/02/opera-for-android-web-apps-mobile-world-congress-2016/>.
- [30] «readwrite mobile-future-notifications-not-apps,» [En línea]. Available: <http://readwrite.com/2015/03/13/mobile-future-notifications-not-apps/>.
- [31] «commons.wikimedia.org AppleTV,» [En línea]. Available: <https://commons.wikimedia.org/wiki/File:AppleTV.svg>.
- [32] «wikipedia Apple_TV,» [En línea]. Available: https://en.wikipedia.org/wiki/Apple_TV.
- [33] «wiki AppleTV,» [En línea]. Available: https://es.wikipedia.org/wiki/Apple_TV.
- [34] «AppleTV caracteristicas,» [En línea]. Available: <http://www.apple.com/tv/specs/>.
- [35] «everymac specs,» [En línea]. Available: <http://www.everymac.com/systems/apple/apple-tv/index-appletv.html>.
- [36] «appletv skydocu,» [En línea]. Available: <http://appletv.skydocu.com/es/>.
- [37] «gizmodo appleTV,» [En línea]. Available: <http://es.gizmodo.com/el-nuevo-apple-tv-ya-esta-aqui-aplicaciones-streaming-1729608673>.
- [38] «flickr linsinchen,» [En línea]. Available: <https://www.flickr.com/photos/linsinchen/13919304134>.
- [39] «itunes livestream,» [En línea]. Available: <https://itunes.apple.com/es/app/livestream/id493086499?mt=8>.

- [40] «help.livestream Use-Apple-TV-to-Watch-Livestream-Events,» [En línea]. Available: <https://help.livestream.com/hc/en-us/articles/213179127-Use-Apple-TV-to-Watch-Livestream-Events>.
- [41] «wiki airPlay,» [En línea]. Available: <https://en.wikipedia.org/wiki/AirPlay>.
- [42] «support airPlay,» [En línea]. Available: <https://support.apple.com/en-us/HT204289>.
- [43] «hipertextual a fondo que es airPlay,» [En línea]. Available: <https://hipertextual.com/archivo/2010/09/a-fondo-que-es-airplay/>.
- [44] «wiki Wi-Fi,» [En línea]. Available: <https://en.wikipedia.org/wiki/Wi-Fi>.
- [45] «wikipedia Tumblr,» [En línea]. Available: <https://es.wikipedia.org/wiki/Tumblr>.
- [46] «tumblr,» [En línea]. Available: <https://www.tumblr.com/>.
- [47] «hbo,» [En línea]. Available: <http://www.hbo.com/>.
- [48] «www.apple.com hbo-now,» [En línea]. Available: <http://www.apple.com/tv/entertainment/hbo-now/>.
- [49] «wikipedia Fox_Broadcasting_Company,» [En línea]. Available: https://en.wikipedia.org/wiki/Fox_Broadcasting_Company.
- [50] «itunes.apple.com fox-now,» [En línea]. Available: <https://itunes.apple.com/us/app/fox-now/id571096102?mt=8>.
- [51] «www.apple.com entertainment,» [En línea]. Available: <http://www.apple.com/tv/entertainment/>.
- [52] «macworld 30-must-know-secrets-and-shortcuts-for-your-apple-tv,» [En línea]. Available: <http://www.macworld.com/article/3003185/home->

players/30-must-know-secrets-and-shortcuts-for-your-apple-tv.html#slide1.

- [53] «macworld tips-and-tricks-for-the-new-apple-tvs-siri-remote,» [En línea]. Available: <http://www.macworld.com/article/3000918/home-players/tips-and-tricks-for-the-new-apple-tvs-siri-remote.html>.
- [54] «AppleTV Support,» [En línea]. Available: <https://support.apple.com/apple-tv>.
- [55] «www.iconfinder.com blog_social_social_media_tumblr_icon,» [En línea]. Available: https://www.iconfinder.com/icons/317724/blog_social_social_media_tumblr_icon.
- [56] «apple product siri-remote,» [En línea]. Available: <http://www.apple.com/shop/product/MLLC2LL/A/siri-remote>.
- [57] «commons.wikimedia.org Apple_tv_gen_4_remote,» [En línea]. Available: https://commons.wikimedia.org/wiki/File:Apple_tv_gen_4_remote.jpeg.
- [58] «support.apple siri,» [En línea]. Available: <https://support.apple.com/es-es/HT205300>.
- [59] «cultofmac siri-apple-tv-remote-tips,» [En línea]. Available: <http://www.cultofmac.com/411423/siri-apple-tv-remote-tips/>.
- [60] «imore what-can-siri-do-apple-tv,» [En línea]. Available: <http://www.imore.com/what-can-siri-do-apple-tv>.
- [61] «apple apple-tv-gets-new-siri-capabilities-and-single-sign-on,» [En línea]. Available: <http://www.apple.com/newsroom/2016/06/apple-tv-gets-new-siri-capabilities-and-single-sign-on.html>.

- [62] «news.softwarevilla apple-tv-remote-app-for-iphone-releases-with-siri-features,» [En línea]. Available: <https://news.softwarevilla.com/2016/08/02/apple-tv-remote-app-for-iphone-releases-with-siri-features/>.
- [63] «developer.apple ControllingInputontvOS,» [En línea]. Available: https://developer.apple.com/library/tvos/documentation/ServicesDiscovery/Conceptual/GameControllerPG/ControllingInputontvOS/ControllingInputontvOS.html#/apple_ref/doc/uid/TP40013276-CH7-DontLinkElementID_5.
- [64] «developer.apple WorkingwithGameControllers,» [En línea]. Available: https://developer.apple.com/library/tvos/documentation/General/Conceptual/AppleTV_PG/WorkingwithGameControllers.html#/apple_ref/doc/uid/TP40015241-CH18-SW1.
- [65] «wikipedia Infrared,» [En línea]. Available: <https://en.wikipedia.org/wiki/Infrared>.
- [66] «Third party remote, how to - Oficial,» [En línea]. Available: <https://support.apple.com/en-us/HT201856>.
- [67] «AppleTV games-and-more,» [En línea]. Available: <http://www.apple.com/es/tv/games-and-more/>.
- [68] «iphonelife gamer-gaming-apple-tv-leaves-much-to-be-desired,» [En línea]. Available: <https://www.iphonelife.com/content/gamer-gaming-apple-tv-leaves-much-to-be-desired>.
- [69] «geek what-happened-to-games-on-the-apple-tv-and-apple-watch,» [En línea]. Available: <http://www.geek.com/games/what-happened-to-games-on-the-apple-tv-and-apple-watch-1656739/>.

- [70] «macworld the-first-10-apple-tv-games-you-should-play,» [En línea]. Available: <http://www.macworld.com/article/2999007/home-tech/the-first-10-apple-tv-games-you-should-play.html#slide1>.
- [71] «macworld 22-best-apple-tv-games-2015-2016,» [En línea]. Available: <http://www.macworld.co.uk/feature/apple/22-best-apple-tv-games-2015-2016-action-adventure-puzzle-scroller-gaming-3611419/>.
- [72] «idownloadblog tvos-10-games-no-siri-remote,» [En línea]. Available: <http://www.idownloadblog.com/2016/06/14/tvos-10-games-no-siri-remote/>.
- [73] «9to5mac siri-remote-no-longer-required-for-tvos-games,» [En línea]. Available: <https://9to5mac.com/2016/06/13/siri-remote-no-longer-required-for-tvos-games/>.
- [74] «imore apple-tv-about-to-release-gaming-kraken,» [En línea]. Available: <http://www.imore.com/apple-tv-about-to-release-gaming-kraken>.
- [75] «theverge apple-tv-game-console-review,» [En línea]. Available: <http://www.theverge.com/2015/11/10/9680146/apple-tv-game-console-review>.
- [76] «thenextweb game-developers-take-serious-look-apple-tv,» [En línea]. Available: <http://thenextweb.com/dd/2016/05/17/game-developers-take-serious-look-apple-tv/#gref>.
- [77] «itunes.apple.com badland,» [En línea]. Available: <https://itunes.apple.com/es/app/badland/id535176909?mt=8>.
- [78] «badlandgame,» [En línea]. Available: <http://badlandgame.com/>.
- [79] «itunes.apple.com sketchparty,» [En línea]. Available: <https://itunes.apple.com/es/app/sketchparty-tv/id500175028?mt=8>.
- [80] «sketchparty,» [En línea]. Available: <http://sketchparty.tv/>.

- [81] «itunes.apple.com xenowerk,» [En línea]. Available: <https://itunes.apple.com/us/app/xenowerk-tv/id1047229854?mt=8>.
- [82] «pixelbite xenowerk,» [En línea]. Available: <http://www.pixelbite.se/more-games/xenowerk/>.
- [83] «itunes.apple.com beneath-the-lighthouse,» [En línea]. Available: <https://itunes.apple.com/es/app/beneath-the-lighthouse/id1024522751?mt=8>.
- [84] «itunes.apple.com galaxy-on-fire-manticore-rising,» [En línea]. Available: <https://itunes.apple.com/es/app/galaxy-on-fire-manticore-rising/id1032131368?mt=8>.
- [85] «gof3manticore,» [En línea]. Available: <http://www.gof3manticore.com/>.
- [86] «itunes.apple.com lumino-city,» [En línea]. Available: <https://itunes.apple.com/es/app/lumino-city/id958604518?mt=8>.
- [87] «luminocitygame,» [En línea]. Available: <http://www.luminocitygame.com/press/>.
- [88] «itunes.apple.com transistor,» [En línea]. Available: <https://itunes.apple.com/es/app/transistor/id948857526?mt=8>.
- [89] «supergiantgames transistor,» [En línea]. Available: <http://www.supergiantgames.com/games/transistor/>.
- [90] «itunes.apple.com guitar-hero-live,» [En línea]. Available: <https://itunes.apple.com/es/app/guitar-hero-live/id1024764676?mt=8>.
- [91] «guitarhero,» [En línea]. Available: <https://www.guitarhero.com/>.
- [92] «badlandgame blog,» [En línea]. Available: <http://badlandgame.com/blog/>.

- [93] «madewith.unity.com beneath-the-lighthouse,» [En línea]. Available: <https://madewith.unity.com/games/beneath-the-lighthouse>.
- [94] «gof3manticore rising,» [En línea]. Available: <http://www.gof3manticore.com/en/rising>.
- [95] «wikipedia Guitar_Hero_Live,» [En línea]. Available: https://en.wikipedia.org/wiki/Guitar_Hero_Live.
- [96] «wikipedia Video_game_development,» [En línea]. Available: https://en.wikipedia.org/wiki/Video_game_development.
- [97] C. E. N. L. L. F. F. M. y. M. Gerardo Abraham Morales Urrutia, «openjournal.uacj.mx Procesos de desarrollo para videojuegos,» [En línea]. Available: <http://openjournal.uacj.mx/ojs/index.php/culcyt/article/view/299/283>.
- [98] A. M. M. Pereira, «revistas.ucm.es El proceso productivo del videojuego,» [En línea]. Available: <http://revistas.ucm.es/index.php/HICS/article/view/45178/42539>.
- [99] «ecured.cu Desarrollo_de_Videojuegos,» [En línea]. Available: http://www.ecured.cu/Desarrollo_de_Videojuegos.
- [100] «ign the-game-production-pipeline-concept-to-completion,» [En línea]. Available: <http://www.ign.com/articles/2006/03/16/the-game-production-pipeline-concept-to-completion?page=1>.
- [101] «wikipedia Respawn_Entertainment_team_photo_at_E3_2013,» [En línea]. Available: https://en.wikipedia.org/wiki/Titanfall#/media/File:Respawn_Entertainment_team_photo_at_E3_2013.JPG.

- [102] «wiki Metodología de desarrollo de software,» [En línea]. Available: https://es.wikipedia.org/wiki/Metodolog%C3%ADa_de_desarrollo_de_software.
- [103] «wiki software development process,» [En línea]. Available: https://en.wikipedia.org/wiki/Software_development_process.
- [104] «web.ua.es creditos-y-asignaturas,» [En línea]. Available: <http://web.ua.es/es/oiapreguntas/creditos-y-asignaturas.html>.
- [105] «developer.apple.com submit,» [En línea]. Available: <https://developer.apple.com/tvos/submit/>.
- [106] «developer.apple.com guidelines,» [En línea]. Available: <https://developer.apple.com/app-store/review/guidelines/>.
- [107] «34milideas.com publicar-nuestra-app-google-play-app-store,» [En línea]. Available: <http://www.34milideas.com/publicar-nuestra-app-google-play-app-store/>.
- [108] «developer.apple.com AppDistributionGuide ManagingAccounts,» [En línea]. Available: https://developer.apple.com/library/mac/documentation/IDEs/Conceptual/AppDistributionGuide/ManagingAccounts/ManagingAccounts.html#/apple_ref/doc/uid/TP40012582-CH24-SW1.
- [109] «developer.apple.com SubmittingYourApp,» [En línea]. Available: https://developer.apple.com/library/mac/documentation/IDEs/Conceptual/AppDistributionGuide/SubmittingYourApp/SubmittingYourApp.html#/apple_ref/doc/uid/TP40012582-CH9-SW1.
- [110] «developer.apple.com AppDistributionGuide,» [En línea]. Available: <https://developer.apple.com/library/mac/documentation/IDEs/Conceptual/>

AppDistributionGuide/LaunchingYourApponDevices/LaunchingYourAppo
nDevices.html#/apple_ref/doc/uid/TP40012582-CH27-SW1.

- [111] «developer.apple.com AppDistributionGuide,» [En línea]. Available: https://developer.apple.com/library/mac/documentation/IDEs/Conceptual/AppDistributionGuide/Introduction/Introduction.html#/apple_ref/doc/uid/TP40012582-CH1-SW1.
- [112] «developer.apple.com distribute,» [En línea]. Available: <https://developer.apple.com/distribute/>.
- [113] « developer.apple On_Demand_Resources_Guide,» [En línea]. Available: https://developer.apple.com/library/tvos/documentation/FileManagement/Conceptual/On_Demand_Resources_Guide/index.html#/apple_ref/doc/uid/TP40015083.
- [114] «developer.apple techtalks-apple-tv,» [En línea]. Available: <https://developer.apple.com/videos/play/techtalks-apple-tv/5/>.
- [115] «imore how-new-apple-tv-uses-demand-resources-host-great-apps-and-games,» [En línea]. Available: <http://www.imore.com/how-new-apple-tv-uses-demand-resources-host-great-apps-and-games>.
- [116] «en.cdprojektred.com,» [En línea]. Available: <http://en.cdprojektred.com/support/tw3-system-requirements/>.
- [117] «game-debate.com Assasins Creed specs,» [En línea]. Available: http://www.game-debate.com/games/index.php?g_id=8514&game=Assassins%20Creed%205.
- [118] «game-debate.com Battlefield1,» [En línea]. Available: http://www.game-debate.com/games/index.php?g_id=9002&game=Battlefield%201.

- [119] «eu.battle.net overwatch specs,» [En línea]. Available: <https://eu.battle.net/support/es/article/7636>.
- [120] «www.nintendo.es SUPER-MARIO-3D-WORLD,» [En línea]. Available: <https://www.nintendo.es/Juegos/Wii-U/SUPER-MARIO-3D-WORLD-765385.html>.
- [121] «support.riotgames.com Minimum-and-Recommended-System-Requirements,» [En línea]. Available: <https://support.riotgames.com/hc/en-us/articles/201752654-Minimum-and-Recommended-System-Requirements>.
- [122] «store.steampowered.com Counter-Strike, Global Offensive,» [En línea]. Available: <http://store.steampowered.com/app/730/?l=spanish>.
- [123] «systemrequirementslab.com battlefield-3,» [En línea]. Available: <http://www.systemrequirementslab.com/cyri/requirements/battlefield-3/11211>.
- [124] «developer.apple.com tvos human-interface-guidelines,» [En línea]. Available: <https://developer.apple.com/tvos/human-interface-guidelines/>.
- [125] «developer.apple.com WorkingwiththeAppleTVRemote,» [En línea]. Available: https://developer.apple.com/library/tvos/documentation/General/Conceptual/AppleTV_PG/WorkingwiththeAppleTVRemote.html#/apple_ref/doc/uid/TP40015241-CH5-SW4.
- [126] «Unity manual tvOS,» [En línea]. Available: <https://docs.unity3d.com/Manual/tvOS.html>.
- [127] «ocs.unity3d.com iphone-joystick,» [En línea]. Available: <http://docs.unity3d.com/Manual/iphone-joystick.html>.

- [128] «gamesetwatch GAMA14_ACG_SalarySurvey_F,» [En línea]. Available: http://www.gamesetwatch.com/2014/09/05/GAMA14_ACG_SalarySurvey_F.pdf.
- [129] «orcahq.com game-industry-salary-explorer,» [En línea]. Available: <https://orcahq.com/blog/game-industry-salary-explorer>.
- [130] «www.animationarena.com video-game-salary,» [En línea]. Available: <http://www.animationarena.com/video-game-salary.html>.
- [131] «gamesindustrysalarysurvey,» [En línea]. Available: <https://www.gamesindustrysalarysurvey.com/>.
- [132] «gamasutra_key_points_from_the_2014_Indie_Salary_Report,» [En línea]. Available: http://www.gamasutra.com/view/news/221630/6_key_points_from_the_2014_Indie_Salary_Report.php.
- [133] «www.polygon.com signifying-nothing-making-indie-games-in-2015,» [En línea]. Available: <http://www.polygon.com/2015/8/6/9110459/signifying-nothing-making-indie-games-in-2015>.
- [134] «positech.co.uk is-indie-pc-gaming-the-next-mobile,» [En línea]. Available: <http://positech.co.uk/cliffsblog/2015/10/16/is-indie-pc-gaming-the-next-mobile/>.
- [135] «www.quora.com How-do-full-time-indie-game-developers,» [En línea]. Available: <https://www.quora.com/How-do-full-time-indie-game-developers-with-no-income-prepare-for-the-extremely-likely-scenario-that-their-game-will-not-be-profitable>.
- [136] «www.takethis.org brigador-creator-discusses-the-mental-health-costs-of-indie-development,» [En línea]. Available:

<http://www.takethis.org/2016/07/brigador-creator-discusses-the-mental-health-costs-of-indie-development/>.

- [137] «[payscale.com Software_Developer/Salary](http://www.payscale.com/research/US/Job=Software_Developer/Salary),» [En línea]. Available: http://www.payscale.com/research/US/Job=Software_Developer/Salary.
- [138] «[trello](https://trello.com/),» [En línea]. Available: <https://trello.com/>.
- [139] «[commons.wikimedia.org Antu_trello](https://commons.wikimedia.org/wiki/File:Antu_trello.svg),» [En línea]. Available: https://commons.wikimedia.org/wiki/File:Antu_trello.svg.
- [140] «[google drive](https://www.google.com/intl/es_es/drive/),» [En línea]. Available: https://www.google.com/intl/es_es/drive/.
- [141] «[google drive apps](http://www.google.es/drive/apps.html),» [En línea]. Available: <http://www.google.es/drive/apps.html>.
- [142] «[support.google.com drive planes](https://support.google.com/drive/answer/2375123?hl=es),» [En línea]. Available: <https://support.google.com/drive/answer/2375123?hl=es>.
- [143] «[dropbox](https://www.dropbox.com/es_ES/),» [En línea]. Available: https://www.dropbox.com/es_ES/.
- [144] «[wikipedia Dropbox](https://es.wikipedia.org/wiki/Dropbox),» [En línea]. Available: <https://es.wikipedia.org/wiki/Dropbox>.
- [145] «[dropbox plans](https://www.dropbox.com/plans),» [En línea]. Available: <https://www.dropbox.com/plans>.
- [146] «[trackingtime](https://trackingtime.co/es/),» [En línea]. Available: <https://trackingtime.co/es/>.
- [147] «[timetune](http://timetune.center/),» [En línea]. Available: <http://timetune.center/>.
- [148] «[Unity3d web oficial](https://unity3d.com/es),» [En línea]. Available: <https://unity3d.com/es>.
- [149] «[wiki Unity](https://en.wikipedia.org/wiki/Unity_(game_engine)),» [En línea]. Available: [https://en.wikipedia.org/wiki/Unity_\(game_engine\)](https://en.wikipedia.org/wiki/Unity_(game_engine)).

- [150] «commons.wikimedia.org Official_unity_logo,» [En línea]. Available: https://commons.wikimedia.org/wiki/File:Official_unity_logo.png.
- [151] «Unity multiplataforma,» [En línea]. Available: https://unity3d.com/es/unity/multiplatform?_ga=1.110001155.749018176.1470128113.
- [152] «Unity planes producto,» [En línea]. Available: https://store.unity.com/es/?_ga=1.86462750.749018176.1470128113.
- [153] u. editor. [En línea]. Available: <http://unity3d.com/es/unity/editor>.
- [154] «monodevelop,» [En línea]. Available: <http://www.monodevelop.com/>.
- [155] «wiki MonoDevelop,» [En línea]. Available: <https://en.wikipedia.org/wiki/MonoDevelop>.
- [156] «netbeans,» [En línea]. Available: <https://netbeans.org/>.
- [157] «eclipse,» [En línea]. Available: <https://eclipse.org/home/index.php>.
- [158] «Notepad++,» [En línea]. Available: <https://notepad-plus-plus.org/>.
- [159] «Unity remote 4,» [En línea]. Available: <https://docs.unity3d.com/Manual/UnityRemote4.html>.
- [160] «XCode Main,» [En línea]. Available: <https://developer.apple.com/xcode/>.
- [161] «wikipedia Xcode,» [En línea]. Available: <https://en.wikipedia.org/wiki/Xcode>.
- [162] «Xcode Features,» [En línea]. Available: <https://developer.apple.com/xcode/features/>.
- [163] «Blender,» [En línea]. Available: <https://www.blender.org/>.

- [164] «wiki open source,» [En línea]. Available:
https://en.wikipedia.org/wiki/Open-source_software.
- [165] «commons.wikimedia.org Blender_logo_no_text,» [En línea]. Available:
https://commons.wikimedia.org/wiki/File:Blender_logo_no_text.svg.
- [166] «blender online manual,» [En línea]. Available:
<https://www.blender.org/manual/>.
- [167] «awesomebump,» [En línea]. Available:
<http://awesomebump.besaba.com/>.
- [168] «wikipedia Shoot-em-Up,» [En línea]. Available:
https://es.wikipedia.org/wiki/Shoot_%27em_up.
- [169] «wikipedia Star_Fox_(saga),» [En línea]. Available:
[https://es.wikipedia.org/wiki/Star_Fox_\(saga\)](https://es.wikipedia.org/wiki/Star_Fox_(saga)).
- [170] «wiki starFox64,» [En línea]. Available:
https://en.wikipedia.org/wiki/Star_Fox_64.
- [171] «wikipedia StarFox64_N64_Game_Box,» [En línea]. Available:
https://en.wikipedia.org/wiki/File:StarFox64_N64_Game_Box.jpg.
- [172] «www.gamasutra.com the_aesthetics_of_game_art,» [En línea]. Available:
http://www.gamasutra.com/view/feature/185676/the_aesthetics_of_game_art_and_.php?print=1.
- [173] «wikipedia Video_game_graphics,» [En línea]. Available:
https://en.wikipedia.org/wiki/Video_game_graphics.
- [174] «wikipedia 3D_computer_graphics,» [En línea]. Available:
https://en.wikipedia.org/wiki/3D_computer_graphics.

- [175] «www.theguardian.com future-of-video-gaming-visuals-nvidia-rendering,» [En línea]. Available:
<https://www.theguardian.com/technology/2015/feb/12/future-of-video-gaming-visuals-nvidia-rendering>.
- [176] «www.ign.com 13-examples-of-timeless-video-game-graphics,» [En línea]. Available: <http://www.ign.com/articles/2015/11/13/13-examples-of-timeless-video-game-graphics>.
- [177] «nerdburglars.net most-artistic-game-visuals-from-last-gen,» [En línea]. Available: <https://nerdburglars.net/most-artistic-game-visuals-from-last-gen/>.
- [178] «wiki agile software development,» [En línea]. Available:
https://en.wikipedia.org/wiki/Agile_software_development.
- [179] w. Scrum. [En línea]. Available:
[https://es.wikipedia.org/wiki/Scrum_\(desarrollo_de_software\)](https://es.wikipedia.org/wiki/Scrum_(desarrollo_de_software)).
- [180] «wikipedia Direct3D,» [En línea]. Available:
<https://en.wikipedia.org/wiki/Direct3D>.
- [181] «wikipedia OpenGL,» [En línea]. Available:
<https://en.wikipedia.org/wiki/OpenGL>.
- [182] «wikipedia Object-oriented_programming,» [En línea]. Available:
https://en.wikipedia.org/wiki/Object-oriented_programming.
- [183] «www.codeproject.com Introduction-to-Object-Oriented-Programming-Concep,» [En línea]. Available:
<http://www.codeproject.com/Articles/22769/Introduction-to-Object-Oriented-Programming-Concep>.

- [184] «en.wikibooks.org Object_Oriented_Programming,» [En línea]. Available: https://en.wikibooks.org/wiki/Object_Oriented_Programming.
- [185] L. R. Izquierdo, «docplayer.es Introduccion-a-la-programacion-orientada-a-objetos,» [En línea]. Available: <http://docplayer.es/131182-Introduccion-a-la-programacion-orientada-a-objetos.html>.
- [186] «wiki Entity_component_system,» [En línea]. Available: https://en.wikipedia.org/wiki/Entity_component_system.
- [187] «t-machine entity-systems-are-the-future-of-mmog-development,» [En línea]. Available: <http://t-machine.org/index.php/2007/11/11/entity-systems-are-the-future-of-mmog-development-part-2/>.
- [188] «wikipedia Object_composition,» [En línea]. Available: https://en.wikipedia.org/wiki/Object_composition.
- [189] «wikipedia Inheritance,» [En línea]. Available: [https://en.wikipedia.org/wiki/Inheritance_\(object-oriented_programming\)](https://en.wikipedia.org/wiki/Inheritance_(object-oriented_programming)).
- [190] «gameprogrammingpatterns component,» [En línea]. Available: <http://gameprogrammingpatterns.com/component.html>.
- [191] «sebaslab ioc-container-for-unity3d-part-1,» [En línea]. Available: <http://www.sebaslab.com/ioc-container-for-unity3d-part-1/>.
- [192] «docs.unity3d.com GameObjects,» [En línea]. Available: <http://docs.unity3d.com/Manual/GameObject.html>.
- [193] «docs.unity3d.com class-GameObject,» [En línea]. Available: <http://docs.unity3d.com/Manual/class-GameObject.html>.
- [194] «docs.unity3d.com LearningtheInterface,» [En línea]. Available: <http://docs.unity3d.com/Manual/LearningtheInterface.html>.

- [195] «docs.unity3d.com Prefabs,» [En línea]. Available:
<http://docs.unity3d.com/Manual/Prefabs.html>.
- [196] «assetstore remote4,» [En línea]. Available:
<https://www.assetstore.unity3d.com/en/#!/content/18106>.
- [197] «Bud Broesky unity-remote4 install tutorial,» [En línea]. Available:
<https://www.youtube.com/watch?v=9R8dVJr6Asw>.
- [198] «wiki Singleton_pattern,» [En línea]. Available:
https://en.wikipedia.org/wiki/Singleton_pattern.
- [199] «wiki Software_design_pattern,» [En línea]. Available:
https://en.wikipedia.org/wiki/Software_design_pattern.
- [200] «wikibooks Design_Patterns,» [En línea]. Available:
https://en.wikibooks.org/wiki/C%2B%2B_Programming/Code/Design_Patterns.
- [201] «sourcemaking design_patterns,» [En línea]. Available:
https://sourcemaking.com/design_patterns.
- [202] «wikipedia Variable,» [En línea]. Available:
[https://es.wikipedia.org/wiki/Variable_\(programaci%C3%B3n\)](https://es.wikipedia.org/wiki/Variable_(programaci%C3%B3n)).
- [203] «wikipedia Ray_casting,» [En línea]. Available:
https://es.wikipedia.org/wiki/Ray_casting.
- [204] «catlikecoding curves and splines,» [En línea]. Available:
<http://catlikecoding.com/unity/tutorials/curves-and-splines/>.
- [205] «devmag.org.za bzier-curves-a-tutorial,» [En línea]. Available:
<http://devmag.org.za/2011/04/05/bzier-curves-a-tutorial/>.

- [206] «answers.unity3d.com find-direction-of-bezier-curve,» [En línea]. Available: <http://answers.unity3d.com/questions/576584/find-direction-of-bezier-curve.html>.
- [207] «wikipedia Application_programming_interface,» [En línea]. Available: https://en.wikipedia.org/wiki/Application_programming_interface.
- [208] «wiki framework,» [En línea]. Available: <https://es.wikipedia.org/wiki/Framework>.
- [209] «wikiStreaming Streaming_media,» [En línea]. Available: https://en.wikipedia.org/wiki/Streaming_media.
- [210] «wikipedia Computer_network,» [En línea]. Available: https://en.wikipedia.org/wiki/Computer_network.
- [211] «wikipedia Porting,» [En línea]. Available: <https://en.wikipedia.org/wiki/Porting>.
- [212] «wikipedia Prototipo,» [En línea]. Available: <https://es.wikipedia.org/wiki/Prototipo>.
- [213] «wikipedia Texture_mapping,» [En línea]. Available: https://en.wikipedia.org/wiki/Texture_mapping.
- [214] «Unity manual,» [En línea]. Available: <http://docs.unity3d.com/Manual/index.html>.
- [215] «textures,» [En línea]. Available: <http://www.textures.com/>.
- [216] «wiki video_game,» [En línea]. Available: https://en.wikipedia.org/wiki/Video_game.
- [217] «wiki código abierto,» [En línea]. Available: https://es.wikipedia.org/wiki/C%C3%B3digo_abierto.

- [218] «wiki inalámbrica,» [En línea]. Available:
https://es.wikipedia.org/wiki/Red_inal%C3%A1mbrica.
- [219] «wikipedia Protocol_stack,» [En línea]. Available:
https://en.wikipedia.org/wiki/Protocol_stack.
- [220] «wikipedia Internet_protocol_suite,» [En línea]. Available:
https://en.wikipedia.org/wiki/Internet_protocol_suite.
- [221] «apple iOS siri,» [En línea]. Available: <http://www.apple.com/es/ios/siri/>.
- [222] «web.ua.es trabajo-fin-de-grado,» [En línea]. Available:
<http://web.ua.es/es/oia/preguntas/trabajo-fin-de-grado.html>.
- [223] «wikipedia Desarrollador_de_tercera,» [En línea]. Available:
https://es.wikipedia.org/wiki/Desarrollador_de_tercera.
- [224] «wikipedia Data_mapping,» [En línea]. Available:
https://en.wikipedia.org/wiki/Data_mapping.
- [225] «wikipedia Google,» [En línea]. Available:
<https://es.wikipedia.org/wiki/Google>.
- [226] «wikipedia Almacenamiento_en_nube,» [En línea]. Available:
https://es.wikipedia.org/wiki/Almacenamiento_en_nube.
- [227] «wikipedia Google_Chrome,» [En línea]. Available:
https://es.wikipedia.org/wiki/Google_Chrome.
- [228] «wikipedia Integrated_development_environment,» [En línea]. Available:
https://en.wikipedia.org/wiki/Integrated_development_environment.
- [229] «wikipedia Compilador,» [En línea]. Available:
<https://es.wikipedia.org/wiki/Compilador>.

- [230] «wikipedia Nintendo_64,» [En línea]. Available:
https://es.wikipedia.org/wiki/Nintendo_64.
- [231] «wikipedia Super_Mario_64,» [En línea]. Available:
https://es.wikipedia.org/wiki/Super_Mario_64.
- [232] «wikipedia The_Legend_of_Zelda:_Ocarina_of_Time,» [En línea]. Available:
https://es.wikipedia.org/wiki/The_Legend_of_Zelda:_Ocarina_of_Time.
- [233] «wikipedia Nintendo,» [En línea]. Available:
<https://es.wikipedia.org/wiki/Nintendo>.
- [234] «wikipedia Bit.Trip_Runner,» [En línea]. Available:
https://en.wikipedia.org/wiki/Bit.Trip_Runner.
- [235] «halfbrick.com jetpack-joyride,» [En línea]. Available:
<http://halfbrick.com/our-games/jetpack-joyride/>.
- [236] «wikipedia código fuente,» [En línea]. Available:
https://es.wikipedia.org/wiki/C%C3%B3digo_fuente.
- [237] «wikipedia Sistema_de_coordenadas,» [En línea]. Available:
https://es.wikipedia.org/wiki/Sistema_de_coordenadas.
- [238] «wikipedia Acoplamiento_informático,» [En línea]. Available:
https://es.wikipedia.org/wiki/Acoplamiento_inform%C3%A1tico.
- [239] «wikipedia Acoplamiento_en_informática,» [En línea]. Available:
[https://es.wikipedia.org/wiki/Acoplamiento_\(inform%C3%A1tica\)](https://es.wikipedia.org/wiki/Acoplamiento_(inform%C3%A1tica)).
- [240] «wikipedia programación modular,» [En línea]. Available:
https://es.wikipedia.org/wiki/Programaci%C3%B3n_modular.

- [241] «alexcpeterson spacescape,» [En línea]. Available: <http://alexcpeterson.com/spacescape/>.
- [242] «commons.wikimedia.org SNES-Controller,» [En línea]. Available: <https://commons.wikimedia.org/wiki/File:SNES-Controller.jpg>.
- [243] «wikipedia Bluetooth,» [En línea]. Available: <https://es.wikipedia.org/wiki/Bluetooth>.
- [244] «wikipedia High-Definition_Multimedia_Interface,» [En línea]. Available: https://es.wikipedia.org/wiki/High-Definition_Multimedia_Interface.
- [245] «wikipedia Power-up,» [En línea]. Available: <https://en.wikipedia.org/wiki/Power-up>.
- [246] «wikipedia 3D_modeling,» [En línea]. Available: https://en.wikipedia.org/wiki/3D_modeling.
- [247] «wikipedia UV_mapping,» [En línea]. Available: https://en.wikipedia.org/wiki/UV_mapping.
- [248] «wikipedia Depuración programas,» [En línea]. Available: https://es.wikipedia.org/wiki/Depuraci%C3%B3n_de_programas.
- [249] «wikipedia Interfaz,» [En línea]. Available: <https://es.wikipedia.org/wiki/Interfaz>.
- [250] «docs.unity3d.com Rigidbody-isKinematic,» [En línea]. Available: <https://docs.unity3d.com/ScriptReference/Rigidbody-isKinematic.html>.
- [251] «wikipediaDigital_asset,» [En línea]. Available: https://en.wikipedia.org/wiki/Digital_asset.
- [252] «docs.unity3d.com UICanvas,» [En línea]. Available: <http://docs.unity3d.com/es/current/Manual/UICanvas.html>.