

TÍTULO DEL TRABAJO FIN DE GRADO

FERNANDO JOSÉ GARCÍA PADILLA

Trabajo fin de Grado

Supervisado por Dr. Pablo Trinidad Martín-Arroyo



Universidad de Sevilla

junio 2017

Publicado en junio 2017 por
Fernando José García Padilla
Copyright © MMXVII

fergarpad@alum.us.es

Pon aquí cuestiones acerca del copyright

Yo, D. Fernando José García Padilla con NIF número 47214882E,

DECLARO

mi autoría del trabajo que se presenta en la memoria de este trabajo fin de grado que tiene por título:

Título del Trabajo Fin de grado

Lo cual firmo,

Fdo. D. Fernando José García Padilla
en la Universidad de Sevilla
28/06/2017

Dedicado a todos los que me han apoyado
Y en especial a...



AGRADECIMIENTOS

No olvides añadir una nota de agradecimiento a quienes hayan contribuido emocionalmente al proyecto fin de Grado.



RESUMEN

El objetivo de este proyecto es la realización, haciendo uso de la ingeniería del software, de un videojuego independiente que sea exigente con sus mecánicas y se aventure combinando varios de los géneros existentes en el mercado con el propósito de llegar a un público específico dentro del mismo. Para su ejecución se ha escogido hacer uso del motor gráfico Unreal Engine, un potente framework que nos ofrece las herramientas que vamos a necesitar durante el transcurso del desarrollo, cuyo uso es gratuito y además está muy extendido en la actualidad.

ÍNDICE GENERAL

I	Introducción	1
1.	Contexto	3
1.1.	El mundo del videojuego	4
1.1.1.	El boom de los e-sports y los juegos para móvil	6
1.1.2.	El auge del videojuego independiente	7
1.1.3.	La irrupción de los trofeos / logros y su impacto	9
1.2.	Estado del arte	10
2.	Objetivos del proyecto	13
2.1.	Motivación	14
2.2.	Listado de objetivos	15
2.2.1.	Objetivos del producto	15
2.2.2.	Objetivos individuales	17
II	Organización del proyecto	19
3.	Metodología	21
3.1.	Estructura organizacional del proyecto	22
3.2.	Metodología de desarrollo	22
3.2.1.	Presentación de la metodología	22

3.2.2.	Resumen de la metodología	22
3.2.3.	Adaptación de la metodología	24
4.	Planificación	27
4.1.	Resumen temporal del proyecto	28
4.2.	Planificación inicial	28
4.3.	Informe de tiempos del proyecto	29
5.	Costes	33
5.1.	Resumen de costes del proyecto	34
5.2.	Costes de personal	34
5.3.	Costes materiales	37
5.4.	Costes indirectos	39
III	Desarrollo del proyecto	43
6.	Arranque	45
6.1.	Lista de características	46
6.2.	Diseño arquitectónico	49
6.2.1.	Módulos de juego	50
6.2.2.	Plugins	51
6.2.3.	Clases de gameplay	52
7.	Iteración cero	55
7.1.	Inicialización del proyecto	56
7.2.	Puesta a punto y primeros pasos	57
7.3.	Directrices generales y reseñas	57

7.3.1. Uso de animaciones profesionales «Mixamo»	58
7.3.2. Implementación del sistema de animación del personaje	58
8. Primera iteración	59
8.1. Características a desarrollar	60
8.2. Diseño	62
8.3. Implementación	68
8.4. Pruebas	73
8.5. Despliegue	74
9. Segunda iteración	75
9.1. Características a desarrollar	76
9.2. Diseño	76
9.3. Implementación	76
9.4. Pruebas	76
9.5. Despliegue	76
10. Tercera iteración	77
10.1. Características a desarrollar	78
10.2. Diseño	78
10.3. Implementación	78
10.4. Pruebas	78
10.5. Despliegue	78
IV Cierre del proyecto	79
11. Manual de usuario	81

11.1. Sección libre	82
12. Conclusiones	83
12.1. Informe post-mortem	84
12.1.1. Lo que ha ido bien	84
12.1.2. Lo que ha ido mal	84
12.1.3. Discusión	84
12.2. Trabajos futuros	84
V Appendices	85
A. Documento de diseño	89
A.1. Visión de conjunto	90
A.1.1. Plataformas de destino	90
A.1.2. Requisitos mínimos objetivo	90
A.1.3. Estilo gráfico	91
A.1.4. Estilo sonoro	91
A.2. Comienzo	91
A.2.1. Menú principal	91
A.2.2. Comienzo del juego e introducción	91
A.2.3. In-game HUDs y menús	91
A.3. Interfaz de usuario	91
A.3.1. Menú principal	91
A.3.2. In-game HUD	92
A.3.3. In-game Options	93
A.3.4. Pantalla de fin de juego	93

A.3.5. Pantalla de selección de nivel	94
A.4. Gameplay	94
A.4.1. Mecánicas	94
A.4.2. Controles	97
A.4.3. Controles con teclado y ratón	98
A.4.4. Controles con mando	98
A.4.5. Modos de juego	99
A.4.6. Ganando el juego	99
A.4.7. Logros / Trofeos	99
A.5. Assets	102
A.5.1. Personajes	104
A.5.2. Enemigos	105
A.5.3. Animaciones	105
A.5.4. Equipo y mejoras	107
A.5.5. Entorno	107
A.5.6. Audio	108
Referencias bibliográficas	108

ÍNDICE DE FIGURAS

1.1. Beneficios de los videojuegos en Europa Occidental (datos en billones americanos) [4]	4
1.2. Audiencia y previsión de crecimiento de los espectadores de deportes electrónicos [3]	6
1.3. Lanzamientos indies a partir de Steam Greenlight [7]	8
6.1. Arquitectura Unreal Engine 4	49
7.1. Ventana del creador de proyectos de Unreal Engine	57

ÍNDICE DE CUADROS

4.1. Tabla resumen de tiempos y planificación	28
4.2. Planificación temporal de fases e iteraciones	28
4.3. Planificación temporal detallada de iteraciones	30
4.4. Tabla resumen de horas de trabajo por rol	30
4.5. Planificación temporal de iteraciones	31
5.1. Tabla resumen de costes	34
5.2. Tabla salarial BOE 2009 (simplificada)	35
5.3. Tabla salarial del proyecto por hora	36
5.4. Tabla sueldos netos	37
5.5. Tabla de características del dispositivo sobremesa	38
5.6. Tabla de características del dispositivo portátil	38
5.7. Tabla de amortización de materiales	39
5.8. Tabla de costes indirectos	40
6.1. Prefijos de clases de gameplay en Unreal Engine	52
8.1. Memorando técnico 0001	62
8.2. Memorando técnico 0001	63
8.3. Memorando técnico 0001	64
8.4. Memorando técnico 0001	65
8.5. Memorando técnico 0001	66

8.6. Memorando técnico 0001	67
A.1. Logro historia - Juego	100
A.2. Logro historia - Tutorial	100
A.3. Logro historia - Capítulo I	101
A.4. Logro historia - Capítulo II	101
A.5. Logro prueba de tiempo - Capítulo I	102
A.6. Logro prueba de tiempo - Capítulo II	102
A.7. Logro coleccionables - 1 coleccionable	103
A.8. Logro coleccionables - 5 coleccionables	103
A.9. Logro coleccionables - 10 coleccionables	104
A.10. Logro coleccionables - 50% de coleccionables	104
A.11. Logro coleccionables - 100% de coleccionables	105
A.12. Logro misceláneos - 100 saltos	105
A.13. Logro misceláneos - 1000 saltos	106
A.14. Logro misceláneos - 10000 saltos	106
A.15. Logro 100%	107
A.16. Assets de personajes	107

TODO LIST

PART I

INTRODUCCIÓN

CONTEXTO

The good parts of a book may be only something a writer is lucky enough to overhear or it may be the wreck of his whole damn life – and one is as good as the other.

*Ernest Hemingway (1899–1961),
Novelist*

***E**n este primer capítulo abordaremos el entorno que rodea al producto software que pretendemos crear, la industria del videojuego, con el objetivo de situarnos en el mismo para conocer las oportunidades que ofrece el mercado y los riesgos, relacionados con el mismo, que debemos tener presentes.*

1.1 EL MUNDO DEL VIDEOJUEGO

La industria del videojuego vive, desde hace años, una etapa de popularización y expansión. Según datos de la asociación española de videojuegos (AEVI) y del informe anual de Newzoo (2016) [1, 4] el pasado año el sector creció un 8.5%, lo que representa pasar de 91.800 millones de dólares en 2015 a 99.600 millones de dólares en 2016. Además, esta tendencia es global y aumenta en cada uno de los cinco continentes, siendo la región geográfica que representan África y Oriente medio la que más prosperó en el transcurso del año.

Teniendo siempre presente que estamos en un mercado global y que lanzar un producto desarrollado en nuestro país, si se satisfacen necesidades como la localización del producto, puede repercutir en ventas en todo el mundo, España se sitúa en la octava posición del ranking mundial, gracias a la nada despreciable cifra de 1.812 millones de dólares de beneficios generados en 2016, pero lejos de los dos principales países, China y Estados Unidos, que sobrepasan con holgura los 20.000 millones de beneficios el pasado año [4].

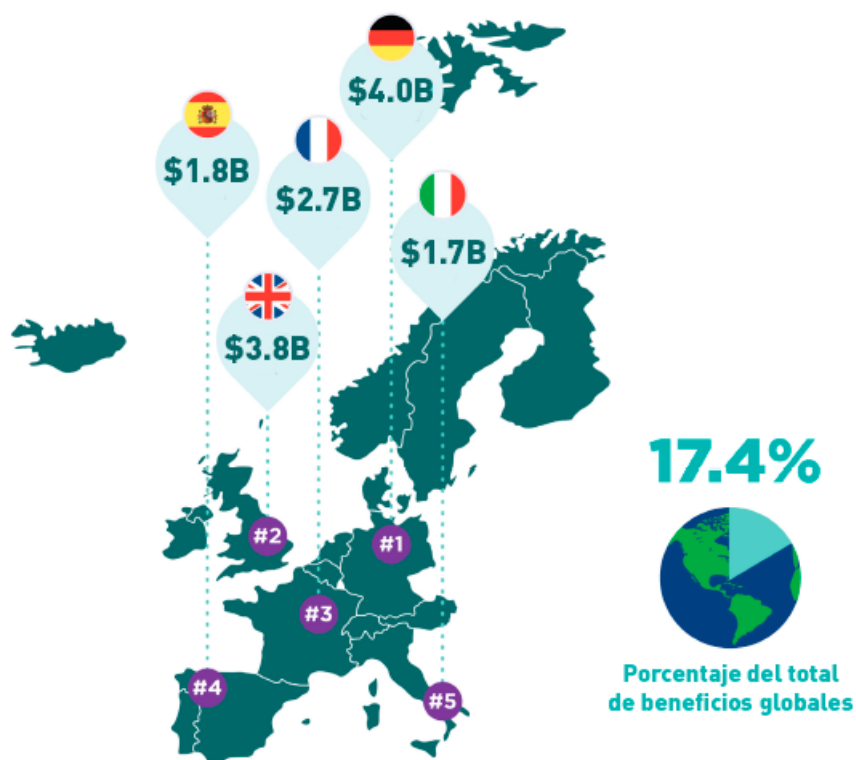


Figura 1.1: Beneficios de los videojuegos en Europa Occidental (datos en billones americanos) [4]

1 La transformación del sector no tiene sólo que ver con cifras económicas y no me
2 gustaría pasar por alto algo que bajo mi punto de vista es bastante más interesante
3 y que, aunque bien daría para un trabajo aparte, describe perfectamente el giro tan
4 drástico al que se está viendo sometida la industria en estos momentos: hace no tanto
5 tiempo jugar a videojuegos era un territorio un tanto desconocido, lo que provocaba
6 que a veces fuese visto con cierto recelo socialmente por una parte de la población.
7 Esta tendencia se ha revertido vertiginosamente en un breve lapso de tiempo e incluso
8 se podría decir que los videojuegos, en gran medida, han pasado a formar parte de la
9 cultura popular: las grandes empresas textiles hacen camisetas con referencias a video-
10 juegos o videoconsolas reconocidas; se realizan grandes eventos mediáticos en torno
11 a personas relacionadas con los videojuegos, a los que se les tratan como a auténticas
12 estrellas del rock; se crean canales de televisión en grandes plataformas audiovisua-
13 les dirigidos únicamente a los videojuegos y toda esta vorágine alcanza su sùmmum
14 cuando en un evento de tanta proyección internacional como es el acto de clausura de
15 los juegos olímpicos, el primer ministro del país que recibe el testigo para organizar-
16 los en cuatro años, en este caso Japón, aparece disfrazado de un reconocido personaje
17 de videojuegos: Mario. Aún queda un largo camino por recorrer, es un problema que
18 persigue al sector desde su creación, pero en los últimos años hemos vivido una aper-
19 tura de los videojuegos a la sociedad de tal magnitud que una parte relevante de la
20 misma lo ha pasado a ver de una forma distinta, dejando de lado ese citado recelo o
21 cualquier connotación negativa, lo cual sin duda no puede ser más que favorable para
22 la industria y su crecimiento.

23 Estableciendo un símil, podríamos comparar en algunos aspectos la fase en la que
24 se encuentra la industria del videojuego con aquella en la que se podría encontrar la
25 industria cinematográfica hace ya unas décadas: el sector está en etapa de expansión,
26 cada vez más universidades empiezan a formar en ámbitos relacionados directamente
27 con los videojuegos, se crean cada vez más puestos de trabajo bajo su cobijo e incluso
28 están creciendo verdaderas celebridades en torno a la industria, ya que es habitual que
29 las personas que son consumidores de la industria conozcan directores de juegos, com-
30 positores o jugadores profesionales, por citar unos ejemplos. Igualmente, no todos los
31 aspectos el sector cinematográfico está más avanzado, ya que no podemos comparar
32 las industrias del videojuego y del cine pasando por alto que la primera, pese a su más
33 que notable juventud, ya consigue doblar en beneficios a la segunda. En conclusión, es
34 cuestión de tiempo que a los videojuegos se le otorgue socialmente el mismo estatus
35 que al cine: ser calificados de cultura o arte, y remarco socialmente porque en países
36 como el nuestro ya se les consideran cultura, pero ese mensaje aún está por calar en

muchos estratos de la sociedad.

1

1.1.1 El boom de los e-sports y los juegos para móvil

2

Aunque ninguno de los dos sectores estén estrictamente relacionados con lo que nos ocupa en este proyecto, son dos importantes partes de la industria que debemos analizar para contextualizarnos y entender plenamente la magnitud del mercado de los videojuegos actualmente.

3

4

5

6

Los deportes electrónicos, también denominados e-sports, son prueba del auge que está viviendo la industria, de los puestos de trabajo que genera en diversos sectores y de las posibilidades de negocio que hay alrededor del éxito que están viviendo actualmente.

7

8

9

10

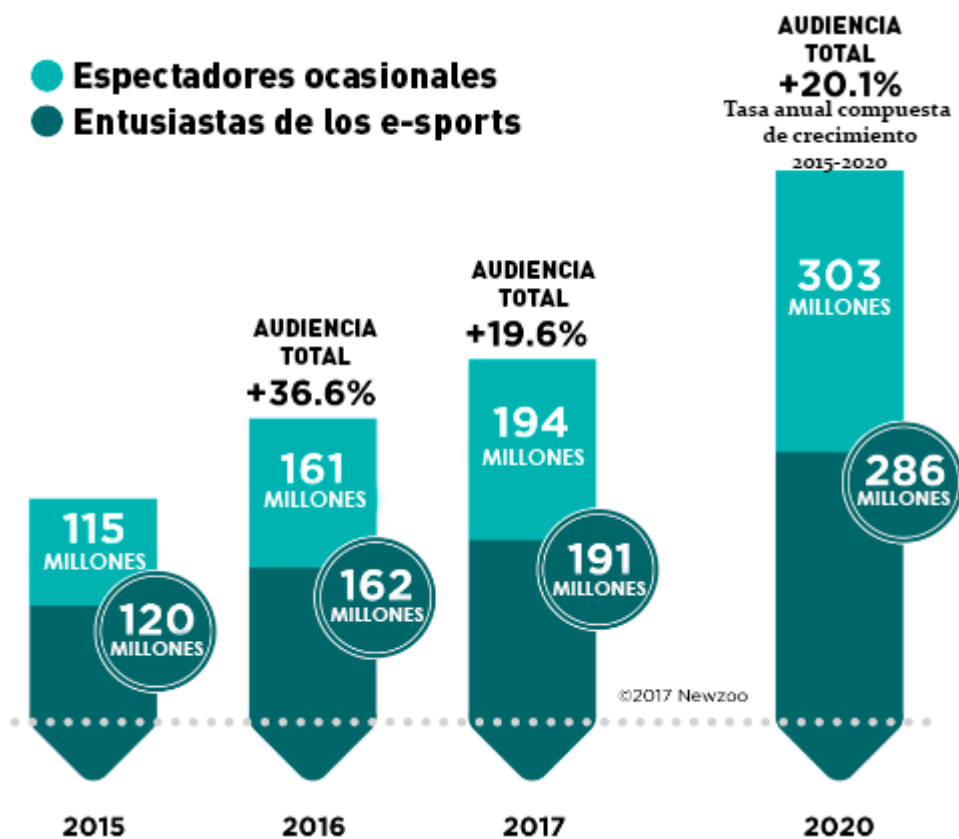


Figura 1.2: Audiencia y previsión de crecimiento de los espectadores de deportes electrónicos [3]

Como se describía en la sección anterior, cadenas de ámbito nacional ya dedican algunos de sus canales a tiempo completo a los videojuegos y, realmente, casi la tota-

11

12

1 lidad de estos suelen estar ligados a los e-sports, algo que actualmente es tendencia
2 en los estados de nuestro entorno y algo normal desde hace años en algunos países
3 concretos como Corea del Sur, pero el poderío de los e-sports va más allá: los deportes
4 electrónicos mueven millones de personas en todo el mundo, consiguen llenar esta-
5 dios, reparten millones en premios y, quizá algo que es mucho más importante, ge-
6 neran un fenómeno fan que, aunque pueda impresionar si no se conoce a fondo este
7 sector, está en vías de convertirse en uno tan poderoso como el que podría ser a día
8 de hoy el fútbol: los espectadores de este tipo de entretenimiento no paran de crecer,
9 conocen los equipos, sus jugadores y, de manera similar como ocurre en los deportes
10 tradicionales, muchos de ellos marcan en el calendario el siguiente torneo de su juego
11 preferido o el próximo partido de su equipo favorito. El fenómeno es tal que incluso se
12 realizan realities de cómo estos equipos de deportes electrónicos viven o entrenan.

13 Por otra parte, los dispositivos móviles están ayudando a que la industria llegue
14 a un número mucho mayor de personas y, lo que es mucho más importante, de una
15 manera más indirecta: ya no es necesario que el usuario sienta esa necesidad, deseo o
16 impulso específico de jugar a videojuegos y realice un desembolso para adquirir una
17 consola, algo que sin duda puede hacer que muchos usuarios potenciales se pierdan en
18 el proceso, sino que cualquiera que disponga de un teléfono móvil puede gozar de un
19 gran catálogo de juegos sea cual sea su plataforma y de una manera muy accesible, por
20 lo que tarde o temprano la mayoría probará lanzar algún videojuego en su dispositivo.

21 Este cambio en la accesibilidad amplía no sólo el espectro de jugadores que llegarán
22 a poseer una consola o un ordenador gaming para tener más títulos a su disposición,
23 generando beneficios para la industria en el proceso, sino que llega a sectores de la po-
24 blación a los que antes era difícil acceder, como el de las personas de mediana edad, y
25 provocan cambios en las tendencias tales como que las mujeres superen a los hombres,
26 en un sector tradicionalmente de los últimos, en el número total de jugadores.

27 1.1.2 El auge del videojuego independiente

28 Centrándonos más en el proyecto que nos ocupa, cabe destacar el fenómeno de
29 los videojuegos indie: juegos desarrollados con poco presupuesto, por un equipo de
30 desarrollo pequeño y que, generalmente, se atreven a arriesgarse un poco más que la
31 media puesto que, tradicionalmente, los juegos desarrollados por grandes empresas y
32 que cuentan con un gran presupuesto detrás tienden a asegurar más, a ser un producto
33 similar a lo que ya está triunfando en ese momento en el mercado. En definitiva, no
34 suelen estar dirigidos al gran público sino que suelen ser un producto de nicho dirigido

a un sector muy específico dentro de la comunidad de jugadores.

Si bien se podría decir que nacieron en la clandestinidad, desde hace años están teniendo una enorme aceptación y prueba de ello es que cada vez tienen mayor visibilidad: Steam, la principal plataforma de distribución digital en PC y que es propiedad de «VALVE», apoya los videojuegos independientes gracias a su «Steam Greenlight», permitiendo a los usuarios de su plataforma valorar juegos desconocidos y, si la aceptación es buena, incluirlos en su catálogo. Sony con su videoconsola PlayStation y Microsoft con su homónima Xbox incorporan desde hace ya algunos años indies en su catálogo, y si bien en un principio eran bastante escasos y sujetos a una serie de restricciones (por ejemplo, en Xbox se apoyaban los juegos realizados con su propio creador de juegos) cada vez son más, y recientemente ha sido Nintendo quien no ha podido dejar pasar el carro de los videojuegos indies incorporando estos a su catálogo de lanzamiento de Nintendo Switch.

Juegos publicados por año

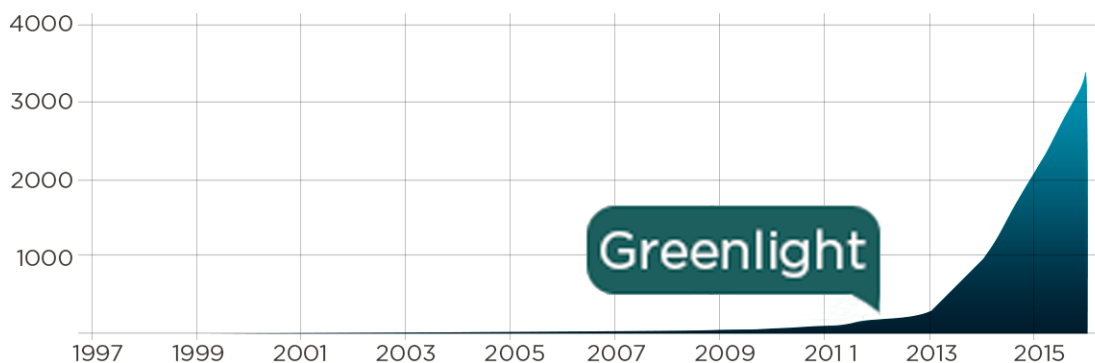


Figura 1.3: Lanzamientos indies a partir de Steam Greenlight [7]

Cabe remarcar que recientemente, desde Junio de 2017, el programa de «VALVE» para sus juegos independientes de «Steam» denominado «Steam Greenlight» no acepta nuevas aportaciones. Esto es porque la compañía está haciendo algunas modificaciones y el sistema pasará a denominarse «Steam Direct». El principal cambio será que los desarrolladores tendrán que abonar una fianza 100\$ por juego, en lugar de realizar un único pago como en «Greenlight» y que permitía publicar tantos juegos en el programa como se quisiese, y al recaudar 1000\$ en beneficios la fianza será devuelta. El principal motivo de esto es mejorar la calidad de los juegos que se incorporan finalmente a la plataforma al salir del programa de lanzamientos [5, 6].

En conclusión, estamos en un momento en el que existe un público dentro de la

1 comunidad que valora desde hace años el soplo de aire fresco que supone el contenido
 2 independiente en la industria, que permanecen atentos al panorama indie y no sólo
 3 a los lanzamientos triple A (o AAA), y en muchos casos valoran este tipo de conteni-
 4 do por encima de una superproducción que va dirigida a un público mucho menos
 5 específico.

6 1.1.3 La irrupción de los trofeos / logros y su impacto

7 Aunque en la década de los 90 ya aparecieron los primeros videojuegos con algún
 8 sistema de desafíos (en concreto el juego «E-Motion» para la videoconsola «Amiga»),
 9 no fue hasta hace muy poco, con la aparición de la consola «Xbox 360» y su sistema
 10 de logros (asociados a una «Gamertag»), que no se redefinieron hasta como los cono-
 11 cemos hoy en día: los logros son definidos en cada juego por los desarrolladores, pero
 12 una vez que obtienen se quedan registrados en el perfil del usuario. Esto permite al
 13 jugador exhibir los logros que más le gusten, los más raros, mostrar el número total de
 14 ellos, el número de juegos con todos los logros completados, el porcentaje de obtención
 15 de logros medio en sus juegos... En resumen, con esta renovación pasaron a tener un
 16 aspecto mucho más social.

17 Además cabe destacar que su propósito inicial se ha ido desvirtualizando con el
 18 tiempo: un logro, como si propio nombre indica, originariamente era otorgado al ju-
 19 gador cuando este realizaba algún tipo de proeza en el juego o cuando completaba el
 20 mismo. Actualmente también se usan en muchos casos para mostrar la progresión del
 21 jugador en el juego pero por otra parte existen cada vez más títulos, conocidos como
 22 «achievement machines» (que podría traducirse como «máquinas de logros»), que re-
 23 compensan excesivamente al jugador (por ejemplo, cada vez que realiza una acción
 24 básica como saltar, disparar, o bien múltiples logros por completar la misma acción) y
 25 además permiten completar el juego en cuestión de minutos, y aunque pudiera pare-
 26 cer lo contrario este tipo de juegos, por el mero hecho de permitirte obtener un gran
 27 número de logros en poco tiempo o logros que te permitan decorar el perfil, tiene un
 28 público y va creciendo.

29 A día de hoy existen numerosas plataformas destinadas a los «achievement hun-
 30 ters» o «cazadores de logros» (como pueden ser «AStats» o «AchievementStats» en
 31 «Steam»), que elaboran numerosas estadísticas con los logros y confeccionan un ran-
 32 king de jugadores atendiendo a los mismos, y otras muchas comunidades con guías o
 33 tutoriales para obtener logros específicos (las más importantes de habla inglesa podrían
 34 ser «XBoxAchievements» en «Xbox» o «PlayStationTrophies» en «PlayStation»).

En definitiva, en la actualidad los logros tienen un peso importante para un sector considerable de los jugadores, de tal manera que algunos incluso dejan de comprar un título por carecer de logros o retrasan la compra hasta que incorporen los mismos.

1.2 ESTADO DEL ARTE

La industria de los videojuegos, aunque es difícil determinar su origen exacto (los primeros videojuegos surgieron tras la Segunda Guerra Mundial, pero a un ritmo lento y no son más que los albores del sector que conocemos hoy en día), es una industria joven. A pesar de esta juventud, y poniéndonos en antecedentes, es un sector que ya ha pasado por una grave crisis a principios de los ochenta y que lo llevó al borde de su extinción. Esta crisis fue motivada, principalmente, por el descontrol que originó la coexistencia en el mercado de demasiadas consolas y, sobre todo, el lanzamiento al mercado de un gran número de videojuegos de baja calidad, sin ningún tipo de control, mermando la confianza del usuario y haciendo descender las cifras de ventas.

La crisis se extendió hasta mediado de los ochenta y, desde entonces, la industria goza de salud y ha ido en crecimiento sin atravesar de nuevo ningún valle, por el contrario, es un sector que desde aquel momento ha ido en una clara tendencia positiva y que actualmente sigue en pleno auge.

Centrándonos en lo que supone en nuestro país, cada vez las universidades y academias ofertan más titulaciones o cursos relacionados con los videojuegos, y esto ayuda a que poco a poco el tejido empresarial relacionado con la industria del videojuego vaya creciendo. La existencia de poco tejido empresarial no ha impedido que en el pasado, durante la corta historia de la industria, en nuestro país hayan aparecido títulos de primer orden a nivel internacional: el caso más conocido es el de la saga «Commandos», que hizo su aparición a finales de los noventa y, más recientemente, un título que podríamos catalogar de súper producción, «Castlevania: Lord of the Shadows» (2010).

Aunque hay excepciones, la notable la mayoría de los juegos actuales son realizados a partir de un motor gráfico comercial ya definido. Estos motores suelen ser propiedad de la compañía, o bien de terceros bajo acuerdo, pero en los últimos años han surgido varios motores de uso gratuito que han ganado enorme popularidad y que hacen más accesible el desarrollo de videojuegos, entre los que encontramos Unreal Engine, CryEngine o Unity.

1 Tomando como ejemplo Unreal Engine, que es el motor que se usará para llevar a
2 cabo este proyecto, es una potente herramienta que más allá de satisfacer las necesida-
3 des básicas del desarrollador (renderizado 2D/3D, detección de colisiones, texturizado
4 de objetos, sistema de luces, etc.) incorpora importantes herramientas en su haber tales
5 como «Persona», que permite al desarrollador crear o realizar cambios en animaciones
6 o «UMG», que permite crear una interfaz de usuario de una manera sencilla dentro
7 del motor. Además, permite utilizar tanto C++ como programación gráfica (a lo que se
8 suele hacer referencia como «Blueprints») siendo la segunda forma unas 10 veces peor
9 en rendimiento que la primera, según la propia Epic Games, desarrolladores de Unreal
10 Engine.

OBJETIVOS DEL PROYECTO

The secret of happiness is not in doing what one likes, but in liking what one does.

Sir James Matthew Barrie (1860–1937),

Novelist

La siguiente sección presenta más detalladamente el proyecto, la razón de ser del mismo y un listado de objetivos, tantos del producto como individuales, que se pretenden alcanzar en el transcurso de su realización.

2.1 MOTIVACIÓN

Dada la saturación del mercado con productos similares o clónicos se pretende crear un videojuego que, siguiendo la filosofía indie que se describía previamente, consiga ser innovador mezclando géneros, que se salga del esquema predominante y rete al jugador siendo desafiante con sus mecánicas y haciéndole pensar, haga uso de la principalmente primera persona para tener la posibilidad de aprovechar el auge de la tecnología de realidad virtual y que, premeditadamente, deje bastante de lado la historia, personajes o desarrollo de los mismos y, por tanto, se centre plenamente en la jugabilidad. El jugador deberá ir haciendo uso de todas las mecánicas de las que dispone para sortear obstáculos, esquivar trampas, solucionar puzzles y hallar el camino correcto para ir avanzando en el juego.

El gameplay se basaría en la repetición y memorización de niveles, esquivar y timing, así como en la gestión de adrenalina y, en menor medida, en la resolución de pequeños puzzles y exploración:

Repetición y memorización de niveles: Los peligros del entorno (balas, flechas, láseres, etc. y, por supuesto, caídas al vacío) matarían instantáneamente al jugador, lo que le obligaría a empezar el nivel desde el último punto de control o, en el caso de no disponer de ninguno, desde cero.

Esquivar y timing: No se le daría la posibilidad al jugador de defenderse o eliminar peligros, por lo que el jugador deberá ir avanzando por el escenario haciendo uso de las mecánicas para sortear peligros en el momento justo sin morir en el intento.

Gestión de adrenalina: Cada acción que realice el jugador tendrá un peso asociado, por tanto, en todo momento el jugador deberá vigilar la barra de adrenalina y, así pues, pensar antes de realizar una acción.

Resolución de puzzles: A menudo el jugador deberá encontrar un botón / palanca / llave para avanzar o bien tendrá que encontrar un orbe dorado para acceder a una determinada zona que antes era inaccesible.

Exploración: No sólo deberá encontrar el camino correcto, o la forma de desbloquear o pasar por un camino, sino que por el mundo habrá atajos que interconecten el mapa con antiguos puntos de control, así como coleccionables repartidos por el mismo.

Podríamos concluir que, principalmente, mezcla los géneros de primera persona, aventura / exploración, plataformas, bullet hell y puzzle.

Teniendo en cuenta lo descrito anteriormente sería un producto que no va dirigido al gran público, sino que el público objetivo sería más bien un público de nicho, que busque algo diferente e intente poner a prueba sus capacidades.

2.2 LISTADO DE OBJETIVOS

He decidido dividir esta sección en dos partes, los objetivos que pretendo alcanzar individualmente y los objetivos que pretendo que alcance el producto:

2.2.1 Objetivos del producto

Sistema de plataformeo: Realización de un sistema de mecánicas básicas de plataformeo como saltar, agacharse, correr, manejar al personaje en el aire, etc. teniendo en cuenta que, sobre todo ésta última, sean amigables con el usuario y permitan llevarlas al límite sin frustración por parte del usuario.

Sistema de escalada: Elaboración de un sistema de escalada que maximice la interacción del personaje con el entorno y permita acciones tales como escalar paredes, desplazarse verticalmente o dejarse caer entre cornisas, moverse por ellas, saltar de una a otra, etc.

Sistema de poderes: Producción de un sistema de poderes coherente, en el que el personaje pueda ganar poderes tales como el de la teletransportación sin que entre en conflicto con las mecánicas, más básicas, de plataformeo y escalada, es decir, que sólo los pueda usar durante un área determinada para poder avanzar en el juego pero que al abandonarla vuelva a tener que hacer uso de las mecánicas de plataformeo y escalada.

Elaboración del tutorial: Elaborar un nivel de tutorial en el que se le presenten al jugador las mecánicas básicas.

Elaboración de niveles prototipo: Realización uno o una serie de niveles prototipo (dependiendo de la longitud de los mismos) de lo que sería un escenario del juego real.

Sistema de animaciones: Creación de una máquina de estados que permita pasar de una animación a otra sin bugs y de una manera sutil, sin saltos entre animaciones.	1 2
Bullet hell en 3D: Conseguir adaptar el género del bullet hell a un entorno 3D de manera adecuada, ya que este género es típico de entornos 2D donde, al eliminar toda una dimensión, es mucho más sencillo de implementar porque se reducen drásticamente los puntos en los que los peligros y el jugador pueden coincidir. En otras palabras, llenar el entorno de peligros en un escenario más amplio es una tarea mucho más difícil de diseñar.	3 4 5 6 7 8
Realidad virtual: Diseñar el juego de manera que fuese amigable con la innovadora tecnología de realidad virtual, es decir, enfocarlo hacia ella y que pudiese ser utilizado con ésta sin problemas (sin prescindir del teclado / ratón o mando).	9 10 11
Sistema de guardado / cargado: El producto debe ser capaz de guardar y cargar los avances del jugador cuando así se requiera. En concreto se hará uso del autoguardado cuando el jugador entre en contacto con la superficie asociada a un punto de control.	12 13 14 15
Sistema interno de desafíos / logros: Se guardará también el progreso del jugador en materia de desafíos en su archivo de guardado. Esto es independiente de cualquier sistema de logros que se pudiesen implementar en el futuro en el juego (logros de Steam, trofeos de PlayStation, logros de Xbox, etc.), pero serían los mismos desafíos por lo que sólo faltaría la comunicación con el sistema de logros externo pertinente. Un sistema como éste es el que usan actualmente muchos títulos del mercado, como «The Binding of Isaac: Rebirth» o «Payday: The Heist»: cuando el usuario alcanza un requisito para un logro se marca el desafío como completado internamente (archivo de guardado) y a la vez lo comunican al servidor de logros pero, si esto por algún motivo falla, lo comunican de nuevo al iniciar otra vez el juego.	16 17 18 19 20 21 22 23 24 25 26
Optimización: Optimizar lo máximo posible realizando todo el contenido posible en C++ ya que como explicaba anteriormente (tomando la propia Epic Games como fuente, desarrolladora de Unreal Engine), los tiempos de ejecución de ese código se reducen en torno a 10 veces usando el lenguaje de programación en C++ sobre la implementación en programación gráfica.	27 28 29 30 31

2.2.2 Objetivos individuales

Uno de los objetivos principales que me he marcado es simplemente aprender y disfrutar, conocer tanto los aspectos organizativos como los de desarrollo que envuelven al mundo de la creación de videojuegos, puesto que es algo que siempre me ha parecido muy interesante. Es un objetivo muy fácil de satisfacer, y no podría decir que cumpliendo sólo este ya estuviese satisfecho, pero sin duda me es indispensable.

Aprender a utilizar de una forma fluida el entorno y el conjunto de herramientas que conforman Unreal Engine.

Ampliar mis conocimientos en el lenguaje de programación C++, aprovechando que es el lenguaje en el cual se programa Unreal Engine.

Adquirir conocimientos en programación visual, ya que es un aspecto que en la titulación se ha abordado de manera muy escueta y habiendo elegido Unreal Engine para la ejecución del proyecto es algo que voy a necesitar en algún momento del desarrollo.

Formarme en diversas aptitudes que no entran dentro del ámbito de la titulación de Ingeniería del Software, como pueden ser la creación de entornos 3D, la texturización e iluminación de los mismos, creación de landscapes, la animación de personajes, la creación de interfaces, diseño de niveles, etc.

PART II

ORGANIZACIÓN DEL PROYECTO

METODOLOGÍA

The good parts of a book may be only something a writer is lucky enough to overhear or it may be the wreck of his whole damn life – and one is as good as the other.

*Ernest Hemingway (1899–1961),
Novelist*

A continuación se presenta la estructura organizacional del proyecto, la metodología que se ha elegido seguir para el desarrollo de este software, un breve resumen de la misma y la forma en la que se va a adaptar a nuestro problema específico.

3.1 ESTRUCTURA ORGANIZACIONAL DEL PROYECTO

Dado que el proyecto no se realiza en grupo no se podría decir que existe una estructura organizacional propiamente dicha, o al menos carecería de sentido establecer una estructura organizacional para una sola persona, ya todas las responsabilidades y roles posibles recaerían sobre una única persona, salvando la figura del tutor.

A raíz de lo descrito surge una consecuencia directa, y es que algunos roles o actividades desaparecerían del proyecto, todos los que tengan que ver con la interacción entre dos o más personas, al carecer de sentido en un proyecto con un único miembro en él. Ejemplo de esto podrían ser todas aquellas actividades que tengan que ver con la sincronización / puesta en común con el resto de miembros del equipo o cualquier rol relacionado con la resolución de conflictos dentro del equipo del proyecto.

3.2 METODOLOGÍA DE DESARROLLO

3.2.1 Presentación de la metodología

Para la realización de este proyecto se hará uso de la metodología «Feature-Driven Development» (FDD), denominada «Desarrollo basado en funcionalidades» en español.

Fue creada por Jeff De Luca y Peter Coad a finales del siglo XX para salvar un importante proyecto que había sido declarado como irrealizable y tiene su razón de ser en la calidad y el monitoreo constante del proyecto.

3.2.2 Resumen de la metodología

FDD es una metodología englobada en el grupo de las denominadas ágiles, constituye un proceso iterativo e incremental de desarrollo software y consta de 5 partes:

1. **Desarrollo del modelo global:** La realización de un proyecto siguiendo el esquema dictado por FDD empieza por la realización de un modelo global de alto nivel que tiene en cuenta el contexto y el alcance del sistema. El modelo se subdivide en partes más pequeñas que se van completando y se confecciona un diagrama de clases por cada una. Finalmente cada una de las partes en las que se subdividió se van agrupando para formar el modelo global final.

2. **Elaboración de la lista de funcionalidades:** Seguidamente, utilizando el conocimiento obtenido durante el desarrollo del modelo global, se confecciona una lista de funcionalidades que posteriormente se dividirán en funcionalidades más específicas.
3. **Planificación por funcionalidad:** Nuevamente partiendo del punto anterior, tomamos la lista de funcionalidades y esta vez las ordenamos según su prioridad y teniendo en cuenta su dependencia.
4. **Diseño por funcionalidad:** Se elige un conjunto de funcionalidades de la lista y se procede a diseñarlas y desarrollarlas mediante un proceso iterativo.
5. **Construcción por funcionalidad:** Después de una fase de diseño satisfactoria se procede a la construcción total del proyecto.

Además, la metodología FDD consta de 6 **roles clave**, los cuales se describen a continuación:

1. **Jefe de proyecto:** Es el responsable de gestionar el presupuesto, el tiempo, el espacio y los recursos del proyecto. También es el responsable de transmitir el progreso del proyecto a los altos cargos de la empresa.
2. **Arquitecto jefe:** Es el responsable del diseño global del sistema. Tiene la última palabra en todas las cuestiones de diseño.
3. **Jefe de desarrollo:** Es el responsable del desarrollo en el día a día, de evitar situaciones de bloqueo y de solucionar conflictos en el proyecto.
4. **Programadores jefes:** Son desarrolladores experimentados que se encargan de diseñar los requisitos a alto nivel y trabajan junto a otros programadores jefes para resolver las dificultades a las que se enfrenta el proyecto día a día.
5. **Encargados de clases:** Desarrolladores que junto a su pequeño equipo de trabajo y bajo la supervisión de un programador jefe diseñan, programan, prueban y documentan las funcionalidades que se implementan en el sistema.
6. **Expertos de dominio:** Son los encargados, haciendo uso de su conocimiento del negocio, de detallarle minuciosamente a los desarrolladores las características que debe tener el producto.

3.2.3 Adaptación de la metodología

Teniendo en consideración que FDD es una metodología pensada para grupos de personas de un tamaño considerable, grupos en los que incluso otras metodologías ágiles como SCRUM no tendrían cabida al ser imposible realizar una autogestión, y enfocada al desarrollo de un software más convencional, donde se podría diseñar un diagrama de clases con facilidad (como podrían ser las aplicaciones de escritorio, móvil, web, etc.) y por contrapartida este proyecto será llevado a cabo por una única persona y el producto generado en su desarrollo será un videojuego, es necesario hacer algunas modificaciones en nuestra metodología:

1. **Fase de diseño del videojuego:** La principal diferencia radica aquí puesto que, dada la naturaleza del proyecto, al no poder organizar un diagrama de clases tenemos que buscar alternativas. En este caso se ha optado por elaborar una documentación con las decisiones de diseño del videojuego, esto vendría a coincidir con el conjunto de las mecánicas que va a contener, las características que incluiría el videojuego y las distintas singularidades que conformarían el mismo. En la industria se denomina comúnmente «Game Design Document» («Documento de diseño del videojuego» en español) y se podrá consultar en el anexo de este documento.
2. **Elaboración de la lista de funcionalidades:** Al igual que en la metodología FDD tradicional, se elabora una lista de funcionalidades a partir del paso anterior y éstas se desglosan a su vez en funcionalidades más específicas.
3. **Planificación por funcionalidad:** Nuevamente como en la metodología FDD tradicional, ordenamos las funcionalidades resultantes según su prioridad y teniendo en cuenta sus dependencias.
4. **Diseño por funcionalidad:** Otra gran diferencia sería que, puesto que el grupo de desarrollo está formado por una sola persona, se irían eligiendo funcionalidades de la lista de una en una y se diseñaría e implementaría.
5. **Construcción por funcionalidad:** Igual que en la metodología FDD se procedería a la construcción total del proyecto.

La **documentación** se generaría de forma constante durante todo el transcurso del proyecto y lo haría de forma iterativa tanto cada vez que se empezara a diseñar una funcionalidad como cuando se terminase el diseño de la misma.

1 Durante el trascurso del proyecto se utilizarán dos **repositorios**, uno para gestionar
2 la documentación (en LaTeX) y otro para almacenar el código del producto en sí.

3 Para finalizar, los **roles** también necesitan adaptarse al proyecto, ya que muchos de
4 ellos dejan de tener sentido por las mismas razones que se citaban anteriormente. En
5 este caso es necesario una simplificación, que llevaría a que los roles de programador
6 se aglutinaran en uno solo y lo mismo pasaría con los roles de diseño, lo que daría
7 lugar a:

- 8 ■ **Jefe de proyecto:** Es el responsable de gestionar el presupuesto, el tiempo, el
9 espacio y los recursos del proyecto. En el caso que nos ocupa sería el encargado
10 de la elaboración de la memoria en términos generales.
- 11 ■ **Diseñador:** Es el responsable del diseño global del sistema y de la elicitación de
12 requisitos tanto a alto nivel como de posteriormente detallarlos. En resumidas
13 cuentas, sería un analista pero esta vez centrado en el mundo del videojuego.
- 14 ■ **Programador:** Diseña, programa, prueba y documenta las funcionalidades que
15 se implementan en el sistema.

PLANIFICACIÓN

The good parts of a book may be only something a writer is lucky enough to overhear or it may be the wreck of his whole damn life – and one is as good as the other.

*Ernest Hemingway (1899–1961),
Novelist*

*E*n la siguiente sección se expone la planificación del proyecto y el informe de tiempos generado durante el transcurso del mismo que, comparado con lo anterior, nos permitirá ver la desviación en cuanto a tiempo del proyecto.

4.1 RESUMEN TEMPORAL DEL PROYECTO

1

Resumen del proyecto	
Fecha de inicio	TODO/TODO/2017
Fecha de fin	TODO/TODO/2017
Periodicidad de las revisiones (promedio)	TODO semanas
Carga de trabajo semanal (promedio)	TODO horas
Horas totales previstas	TODO horas
Horas finales	TODO horas

Cuadro 4.1: Tabla resumen de tiempos y planificación

4.2 PLANIFICACIÓN INICIAL

2

La siguiente tabla muestra un resumen de las iteraciones junto con su fecha de comienzo y fin:

3

4

Resumen de fases e iteraciones	
Fase de estudio	20/02/17 a 12/03/17
Fase de inicio del proyecto	13/03/17 a 02/04/17
Iteración 1	03/04/17 a 16/04/17
Iteración 2	24/04/17 a 14/05/17
Iteración 3	01/05/17 a 14/05/17
Iteración 4	15/05/17 a 28/05/17
Iteración 5	29/05/17 a 11/06/17
Iteración 6	12/06/17 a 25/06/17
Fase de cierre del proyecto	26/06/17 a 30/06/17

Cuadro 4.2: Planificación temporal de fases e iteraciones

Observando la tabla, podemos apreciar que podríamos establecer una clara diferenciación entre las iteraciones:

5

6

- **Fase de estudio:** En esta primera etapa interviene principalmente el jefe de proyecto, que será el encargado de estudiar las diferentes tecnologías que tiene a su

7

8

disposición y de poner los cimientos más básicos de la ruta que seguirá el proyecto.

- **Fase de inicio del proyecto o iteración 0:** Es la etapa en la que se realizan las primeras fases de la metodología y buena parte de la elaboración de la memoria, salvando los apartados finales (por ejemplo, el manual o las conclusiones) y las secciones centradas en el desarrollo, principalmente. Teniendo en cuenta lo descrito, en esta fase intervienen el jefe de proyecto y el diseñador.

- **Iteraciones 1-6:** Son las etapas más puramente de desarrollo. Intervienen tanto el analista, que será el encargado de preparar las iteraciones, como el programador, que será quien codifique y genere documentación de las iteraciones (como ya se explicó en la sección §3.2.3).

- **Fase de cierre del proyecto:** En esta iteración intervienen el diseñador, encargado de elaborar el manual de la aplicación, y el jefe de proyecto, que será quien finalice los últimos apartados de la memoria, haga las últimas modificaciones sobre la misma y realice la presentación mediante la cuál se expondrán los resultados del proyecto.

Nota: En prácticamente todas las fases del proyecto intervendrían los tres roles, pero dado el escenario en el que se enmarca el proyecto (con una única persona para tres roles), en cada fase debemos focalizar más en uno u otro.

TODO: Explicar cómo se han decidido las fechas, interacción con fechas importantes y situaciones personales.

A continuación se muestra una tabla detallada de las iteraciones, mostrando qué rol interviene, el número de horas de trabajo que participa y el número de horas totales de cada iteración:

Y, por último, de la anterior tabla podemos extraer el número total de horas de trabajo de cada rol, lo cuál nos será útil en el próximo capítulo cuando elaboremos la documentación relativa a los costes de personal (ver sección §5.2):

4.3 INFORME DE TIEMPOS DEL PROYECTO

TODO: Lo mismo que el anterior pero con datos reales.

Iteraciones detalladas			
Iteración	Roles implicados	Horas est. / rol	Horas estimadas
Fase de estudio	Jefe de proyecto	30 horas	30 horas
Fase inicio proyecto	Jefe de proyecto	30 horas	40 horas
	Diseñador	10 horas	
Iteración 1	Diseñador	4 horas	24 horas
	Programador	20 horas	
Iteración 2	Diseñador	4 horas	24 horas
	Programador	20 horas	
Iteración 3	Diseñador	4 horas	24 horas
	Programador	20 horas	
Iteración 4	Diseñador	4 horas	24 horas
	Programador	20 horas	
Iteración 5	Diseñador	4 horas	24 horas
	Programador	20 horas	
Iteración 6	Diseñador	4 horas	24 horas
	Programador	20 horas	
Fase cierre proyecto	Diseñador	0 horas	0 horas
	Jefe de proyecto	0 horas	
TOTAL			X horas

Cuadro 4.3: Planificación temporal detallada de iteraciones

Resumen de horas de trabajo por rol	
Jefe de proyecto	X horas
Diseñador	X horas
Programador	X horas

Cuadro 4.4: Tabla resumen de horas de trabajo por rol

TODO: Justificar los retrasos de forma detallada aquí para cada una de las iteraciones. Explicar las razones.

1
2

Resumen de iteraciones	
Iteración 1	10/10/14 a 21/10/14
Iteración 2	21/10/14 a 15/11/14
...	dd/mm/aa a dd/mm/aa

Cuadro 4.5: Planificación temporal de iteraciones

COSTES

1

2 *The good parts of a book may be only something a writer is lucky enough to overhear or it may be the wreck*
3 *of his whole damn life – and one is as good as the other.*

4

Ernest Hemingway (1899–1961),

5

Novelist

6

7 *E* n este capítulo abordaremos una de las partes más importantes en la planificación de
8 un proyecto: los costes, y para ello los dividiremos en costes de personal, de material e
9 indirectos.

5.1 RESUMEN DE COSTES DEL PROYECTO

Uno de los aspectos más importantes para que un proyecto finalice con éxito es que esté bien presupuestado desde su inicio. No obstante, es un objetivo difícil de alcanzar, por tanto lo que se pretende con la realización de este capítulo es, primeramente, intentar que se ajuste lo máximo a la realidad y, en caso de que se desvíe demasiado de la misma, aprender de los errores cometidos para futuros proyectos.

La tabla que se muestra a continuación anticipa los costes a los que tendremos que hacer frente durante la ejecución del proyecto:

Resumen del proyecto	
Costes de personal	TODO €
Sueldo neto	TODO €
Impuestos	TODO €
Costes sociales	TODO €
Costes materiales	TODO €
Costes indirectos	TODO €
TOTAL	TODO €

Cuadro 5.1: Tabla resumen de costes

En las siguientes secciones veremos en detalle cómo están calculados cada uno de los resultados mostrados.

5.2 COSTES DE PERSONAL

Los costes de personal son, con diferencia, los más importantes en un presupuesto de un proyecto del sector TIC: por regla general, en un proyecto de este sector, aproximadamente el 90% de los costes se puede catalogar como costes de personal, y dentro de ese porcentaje alrededor de un 5% son impuestos.

En este caso, el proyecto tendría un único trabajador pero debemos tener en cuenta que desarrollaría diversas tareas dentro del mismo, por tanto el salario no puede ser idéntico para una labor u otra. Tal y como se ha descrito previamente se han establecido 3 roles: jefe de proyecto, diseñador y programador (para más detalles, revisar sección §3.2.3).

Con el fin de calcular el coste que supondrían los honorarios nos apoyamos en el Boletín Oficial del Estado (BOE), en este caso el boletín número 82 del sábado 4 de abril de 2009, que especifica el salario en el sector de tecnologías de la información y la comunicación, en el que como se citaba se enmarca el proyecto. Para hallar los datos que buscamos debemos fijarnos en la tabla salarial de 2009 de dicho boletín, a continuación se muestran los roles de nuestro proyecto junto con los roles que nos interesan de los descritos en el BOE:

Tabla salarial BOE 2009 (simplificada)			
Categoría	Categoría BOE	Salario mes (x14)	Salario año
Jefe de proyecto	Jefe/a de Operación	1.103,04 €	15.442,56 €
Analista	Analista y A. de sistemas	1.569,25 €	21.969,50 €
Programador	Programador/a senior	1.103,04 €	15.442,56 €

Cuadro 5.2: Tabla salarial BOE 2009 (simplificada)

Nótese que nomenclatura de los roles de nuestro proyecto no coinciden plenamente con la que utiliza el BOE, sino que en la tabla anterior jefe de proyecto se corresponde con «jefe/a de operación», analista con «analista y analista de sistemas» y programador con «programador/a senior». A continuación se justifica, para cada uno de ellos, la relación de correspondencia, por tanto citando al BOE:

- **Jefe/a de Operación:** Es el trabajador que, con personal a su cargo, tiene bajo su responsabilidad: la planificación del trabajo a realizar en cada uno de los ordenadores y turnos; la asignación de recursos humanos en cada puesto de trabajo y en cada momento; la coordinación con los integrantes de las distintas áreas del proyecto; el conocimiento, en todo momento, del sistema operativo y del trabajo para tomar decisiones en las dudas que se presenten en operación; y, por último, la conservación de los manuales de operación.
- **Analista de Sistemas:** Le corresponde el diseño, puesta a punto y mantenimiento de los sistemas operativos a utilizar en los procesos de mecanización; formarse e informarse de todo lo concerniente al proceso de datos; y, finalmente, asesorar y coordinar con todo el personal de la empresa sobre las posibilidades de proceso de datos.
- **Programador/a Senior:** Es el trabajador que debe tener un conocimiento profundo de las técnicas y recursos que maneja, enfocado principalmente a los lenguajes

de programación existentes en el ordenador que utiliza así como de las facilidades y ayuda que le presta al software para la puesta a punto de programas, correspondiéndole estudiar los problemas complejos definidos por los analistas, confeccionando organigramas detallados de tratamiento. Le corresponde redactar programas en el lenguaje de programación que le sea indicado. Asimismo, confecciona juegos de ensayo, pone a punto los programas y completa los expedientes técnicos de los mismos.

Como podemos apreciar en el texto extraído del BOE (que se ha presentado de forma resumida) nuestros roles coinciden en un amplio porcentaje con los roles descritos en él (de nuevo, para más detalles sobre los roles de este proyecto, revisar sección §3.2.3).

A partir de la tabla anterior, debemos calcular los honorarios por hora de cada uno de los roles. Para realizar esto debemos dividir el salario anual entre la jornada anual:

$$\frac{\text{Salario}_{\text{Anual}}}{\text{Jornada Laboral Anual}_{\text{Horas}}} = \text{Salario}_{\text{Hora}}$$

La jornada máxima anual del sector TIC está fijada en 1.800 horas, por tanto, una vez aplicado esto a cada rol, obtendríamos los siguientes resultados:

Tabla salarial del proyecto por hora	
Categoría	Salario por hora
Jefe de proyecto	8,58 €
Diseñador	12,20 €
Programador	8,58 €

Cuadro 5.3: Tabla salarial del proyecto por hora

Dentro de los costes de personal debemos diferenciar entre el sueldo neto del trabajador, los impuestos y los costes sociales.

Para calcular el sueldo neto del trabajador, debemos tomar la planificación del proyecto que hemos visto previamente (sección §4.1) y multiplicar el total de horas de cada rol por su coste a la hora que acabamos de calcular, así pues:

$$\text{Sueldo}_{\text{Neto}} = \text{Salario}_{\text{Hora}} * \text{Horas}_{\text{Proyecto}}$$

Y, a su vez, podemos concluir que:

$$TotalSueldos_{Neto} = \sum_{i=1}^{n^{\circ}empleados} SueldoNeto_i = TODO$$

Sueldos netos			
Rol	Total de horas	Sueldo neto por hora	Sueldo neto
Jefe de proyecto	TODO horas	8,58 €	TODO €
Costes materiales	TODO horas	12,20 €	TODO €
Costes indirectos	TODO horas	8,58 €	TODO €
TOTAL			TODO €

Cuadro 5.4: Tabla sueldos netos

Por tanto, el coste del proyecto en cuanto a sueldos netos asciende a **TODO**.

TODO: Impuestos y costes sociales.

5.3 COSTES MATERIALES

Los costes materiales son costes independientes de un proyecto en concreto, por lo que el coste que supone para este proyecto no se corresponde directamente su valor de adquisición, sino que es necesario realizar una amortización teniendo en cuenta la duración de este proyecto respecto a la vida útil de cada elemento.

La amortización se realizaría de la siguiente manera:

$$Coste_{Mes} = \frac{CosteCompra}{VidaUtil_{Meses}}$$

Para la ejecución del proyecto se cuenta con:

- Dos equipos, un sobremesa y un portátil (más detalles sobre los equipos a continuación) ambos con una vida útil estimada de 3 años ya que el sobremesa, aún siendo más antiguo, ha sido actualizado recientemente.
- Una mesa de trabajo, con una vida útil de 4 años.

- Una pequeña mesa de reuniones que se utilizaría en caso de tener que mantener una reunión con una persona interesada en el proyecto y que contaría con una vida útil de 6 años. 1
2
3
- Cinco sillas: una de escritorio utilizada en la mesa de trabajo y cuatro, más simples, utilizadas en la mesa de reuniones. La utilizada en la mesa de trabajo costaría con una vida útil de 4 años y el resto contarían con un vida útil de 6 años. 4
5
6
- Dos repisas, ambas con una vida útil de 6 años. 7
- Una estantería, con una vida útil de 6 años. 8
- Dos papeleras, con una vida útil de 10 años. 9

Para la realización del proyecto se dispone de dos dispositivos, un ordenador sobremesa y un ordenador portátil (a partir de ahora sobremesa y portátil) en el que se realizarán las labores de producción, preproducción y pruebas. 10
11
12

Características sobremesa	
Sistema/s operativo/s	Windows 10
Procesador	Intel Core i7 930 2.80Ghz
Placa base	Asus P6X58D-E
Memoria RAM	12GB DDR3 1600Mhz PC3-12800 CL6 (2x4GB + 2x2GB)
Disp. de almacenamiento	SSD 120GB + Disco duro 3TB SATA3 7200rpm
Tarjeta gráfica	Sapphire Radeon HD 7950 OC 3GB GDDR5

Cuadro 5.5: Tabla de características del dispositivo sobremesa

Características portátil	
Sistema/s operativo/s	Windows 10
Procesador	Intel Core i7 4712MQ 2.3 GHz
Placa base	(Dato no proporcionado)
Memoria RAM	8GB DDR3 SODIMM (1x8GB)
Disp. de almacenamiento	Disco duro 1TB SATA 5400rpm
Tarjeta gráfica	Nvidia GeForce GT820M 2GB GDDR3

Cuadro 5.6: Tabla de características del dispositivo portátil

- 1 La tabla que se muestra a continuación muestra los materiales que van a cumplir
 2 una utilidad en la realización del proyecto, su vida útil, su valor y el coste real al mes
 3 que supone en el proyecto:

Amortización de materiales			
Material	Valor	Vida útil	Coste real mes
Sobremesa	1100 €	3 años (36 meses)	30,55 €
Portátil	700 €	3 años (36 meses)	19,44 €
Mesa trabajo	200 €	4 años (48 meses)	4,16 €
Mesa reuniones	150 €	6 años (72 meses)	2,08 €
Silla trabajo	60 €	4 años (48 meses)	1,25 €
Silla reuniones x 4	40 €	6 años (72 meses)	0,55 * 4 = 2,22 €
Repisa x 2	30 €	6 años (72 meses)	0,41 * 2 = 0,83 €
Estantería	80 €	6 años (72 meses)	1,11 €
Papelera x 2	5 €	10 años (120 meses)	0,04 * 2 = 0,08 €
TOTAL (mes)			59,63 €

Cuadro 5.7: Tabla de amortización de materiales

- 4 Una vez realizado esto, sólo faltaría multiplicar el coste material al mes por la du-
 5 ración del proyecto en meses, de esta forma:

$$\text{CosteMaterial}_{\text{proyecto}} = \text{CosteAmortizado} * \text{MesesProyecto}$$

- 6 Teniendo en cuenta el número de meses del proyecto, en este caso TODO meses, el
 7 coste material total es de **TODO**.

8 5.4 COSTES INDIRECTOS

- 9 Los costes indirectos son, sin lugar a dudas, los más complicados de calcular a la
 10 hora de elaborar un presupuesto.

- 11 Siguiendo su definición, un coste indirecto es aquel afecta al proceso productivo en
 12 general de uno o más productos, en este caso proyectos, y que por tanto es complicado
 13 asignar una parte de ese coste a un proyecto en concreto sin elaborar algún criterio,
 14 sistemático y estable en el tiempo, que nos permita hacerlo.

Existen gran variedad de costes indirectos, como podrían ser licencias, seguros, bancos, contratación de servicios profesionales (como podrían ser abogados para posibles temas legales o gestoras para, por ejemplo, llevar a acabo la contabilidad o las declaraciones a hacienda), alquileres, etc.

En el caso de este proyecto, al mes, son los siguientes:

Costes indirectos	
Motivo del coste	Coste mes
Licencias	0 €*
Alquiler	400 €
Internet (fibra óptica)	40 €
Luz	60 €
Agua	20 €
Material fungible	20 €
Servicios de limpieza	150 €
TOTAL (mes)	TODO €

Cuadro 5.8: Tabla de costes indirectos

[*]: En cuanto a licencias debemos tener en presente que Unreal Engine es un motor de uso gratuito pero, sin embargo, la compañía propietaria del motor, en este caso Epic Games, establece en su EULA una cláusula por la que habrá que abonar cierta cantidad, un loyalty, si se superan los 3.000\$ de beneficio por cuatrimestre. Esta cantidad en gira en torno a un 5% de los beneficios que pasen de 3.000\$: por ejemplo, si se generan 5000\$ de beneficio habrá que remitirle a Epic Games aproximadamente 100\$ (el 5% de 2000\$, resultado de la resta de 5.000\$ - 3.000\$).

En este caso, el coste indirecto al mes no sería más que el sumatorio de todos los costes indirectos al mes:

$$CosteMaterial_{Mes} = \sum_{i=1}^{n^o \text{costes}} CosteIndirecto_i$$

Por consiguiente, el coste indirecto total del proyecto sería el coste indirecto por mes multiplicado por el número de meses del proyecto:

$$CosteMaterial_{Proyecto} = CosteAmortizado * MesesProyecto$$

- ¹ Por último, teniendo presente el número de meses del proyecto, en el caso que nos
- ² ocupa TODO meses, el coste indirecto total del proyecto asciende a **TODO**.

PART III

DESARROLLO DEL PROYECTO

ARRANQUE

1

2 *The good parts of a book may be only something a writer is lucky enough to overhear or it may be the wreck*
3 *of his whole damn life – and one is as good as the other.*

4

Ernest Hemingway (1899–1961),

5

Novelist

6

7 *D*urante este capítulo de la memoria se presentará la lista de características que con-
8 forman el proyecto, así como el diseño arquitectónico de Unreal Engine, el motor
9 gráfico elegido para llevar a cabo el proyecto.

6.1 LISTA DE CARACTERÍSTICAS

La lista de características está confencionada a partir del documento de diseño, incorporado como anexo a este documento (Ver apéndice §A), por tanto, para encontrar una descripción más detallada tendremos que remitirnos a dicho anexo.

1. Mecánica: Movimiento del personaje.

- Implementación del código de la mecánica.
- Asignación de teclas y botones vinculados la mecánica.
- Desarrollo de la animación del personaje relacionada con la mecánica.

2. Mecánica: Rotación del personaje.

- Implementación del código de la mecánica.
- Asignación de teclas y botones vinculados la mecánica.
- Desarrollo de la animación del personaje relacionada con la mecánica.

3. Mecánica: Sprint.

- Implementación del código de la mecánica.
- Asignación de teclas y botones vinculados la mecánica.
- Desarrollo de la animación del personaje relacionada con la mecánica.

4. Mecánica: Salto.

- Implementación del código de la mecánica.
- Asignación de teclas y botones vinculados la mecánica.
- Desarrollo de la animación del personaje relacionada con la mecánica.

5. Mecánica: Agachado.

- Implementación del código de la mecánica.
- Asignación de teclas y botones vinculados la mecánica.
- Desarrollo de la animación del personaje relacionada con la mecánica.

6. Mecánica: Tiempo bala (o «Time dilation»).

- Implementación del código de la mecánica.

- Asignación de teclas y botones vinculados la mecánica.
- Desarrollo de la animación del personaje relacionada con la mecánica.
- Gestión de adrenalina relacionada con la mecánica.

7. Mecánica: Parpadeo (o «Blink»).

- Implementación del código de la mecánica.
- Asignación de teclas y botones vinculados la mecánica.
- Desarrollo de la animación del personaje relacionada con la mecánica.
- Gestión de adrenalina relacionada con la mecánica.
- Implementación del código del indicador de blink disponible.

8. .

9. .

10. .

11. .

12. .

13. .

14. .

15. .

16. .

17. .

18. .

19. .

20. .

21. .

22. .

23. .

24. .	1
25. .	2
26. .	3
27. .	4
28. .	5
29. .	6
30. .	7

6.2 DISEÑO ARQUITECTÓNICO

En el momento que hablamos de diseño arquitectónico en el marco de este proyecto estamos refiriéndonos al diseño arquitectónico del motor que estamos utilizando, «Unreal Engine 4». La información que se muestra a continuación está extraída de fuentes oficiales de Epic Games, propietaria de «Unreal Engine», en concreto de la documentación (en inglés) del motor [2].

Cuando programamos elementos del gameplay (elementos del juego) usando código C++, cada módulo («gameplay module») puede contener muchas clases en C++.

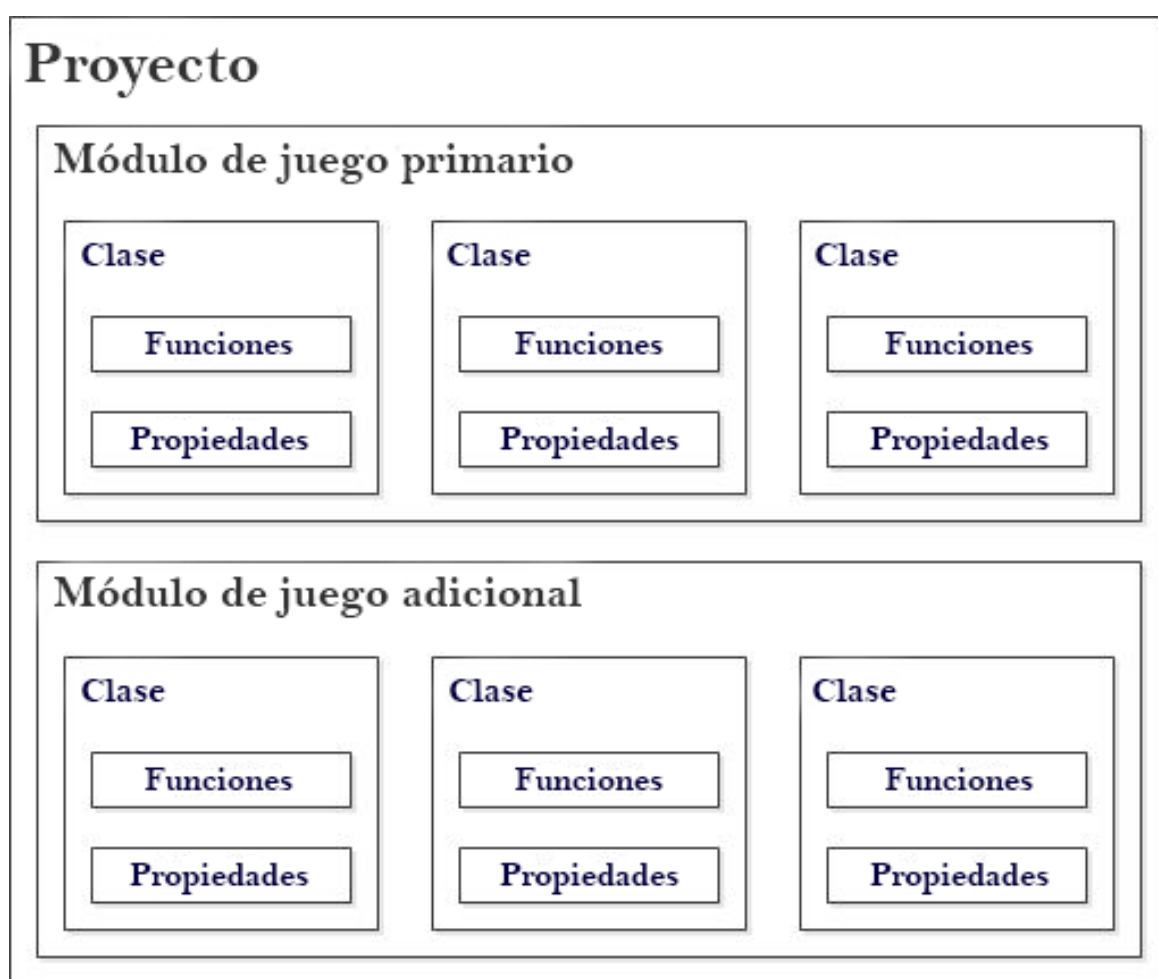


Figura 6.1: Arquitectura Unreal Engine 4

Cada clase determina una plantilla para un nuevo actor u objeto. En el archivo de encabezado de clase («.h») se declaran la clase y cualquier función o propiedades de la clase. Las clases también pueden contener «structs», estructuras de datos que ayudan

en la organización y manipulación de las propiedades relacionadas. Las estructuras también pueden ser definidas en su propia interfaz, permitiendo que puedan ser implementadas en diferentes clases.

Cuando programamos en Unreal Engine, es posible tener clases, funciones y variables estándar del lenguaje C++, usando la sintaxis estándar de C++. Sin embargo, también podemos usar las instrucciones «UCLASS()», «UFUNCTION()» y «UPROPERTY()» para que Unreal Engine conozca nuevas clases, funciones y propiedades, respectivamente. Por ejemplo, una variable cuya declaración está precedida por «UPROPERTY()» puede ser reconocida por el motor y ser mostrada y editada en el editor de Unreal Engine. También existen «UINTERFACE()» y «USTRUCT()» y palabras claves para cada una que pueden ser usadas para especificar el comportamiento de una clase, función, propiedad, interfaz o estructura en Unreal Engine.

6.2.1 Módulos de juego

De la misma forma en la que el propio motor está compuesto por una colección de módulos, cada juego está compuesto por uno o más módulos de juego. Estos son similares a los paquetes de las versiones anteriores del motor, en las cuales estos eran contenedores de una colección de clases relacionadas. En Unreal Engine 4, ya que está basado en C++, **los módulos son en realidad DLLs**, en lugar de paquetes.

Los módulos de juego deben contener, como mínimo, un archivo de cabecera («.h»), un archivo C++ («.cpp») y un archivo de compilación («*.Build.cs»).

El archivo de cabecera debe estar localizado en la carpeta «Public» de la localización del módulo, por ejemplo: «[NombreDelJuego]/Source[NombreDelMódulo]/Public». Este archivo contiene cualquier archivo de cabecera, incluyendo las cabeceras auto-generadas del módulo, necesario para compilar las clases que incluye el módulo.

El archivo «C++», localizado en la carpeta «Privada» de la localización del módulo («[NombreDelJuego]/Source[NombreDelMódulo]/Private»), implementa el módulo.

El archivo de compilación («build») es está localizado en la raíz del directorio del módulo («[NombreDelJuego]/Source[NombreDelMódulo]») y contiene información usada por la herramienta de compilación de Unreal Engine («UnrealBuildTool») para compilar el módulo.

1 Módulos de juego múltiples

2 Hay diferentes puntos de vista respecto a la separación en DLL. Separar el juego en
3 un montón de archivos DLL puede tener más inconvenientes que beneficios, pero es
4 una decisión que debe ser hecha por cada equipo de desarrollo basada en sus necesi-
5 dades. Usar múltiples módulos de juego puede conducir en mejores tiempos de enlace
6 e iteración de código más rápida, pero con más módulos será necesario lidiar con ex-
7 portaciones de DLL e interfaces más a menudo. Esta concesión es la correcta para el
8 motor y el editor, pero es cuestionable para el gameplay.

9 También se puede crear un módulo de juego primario y un número indeterminado
10 de módulos de juego adicionales, para ello es necesario crear los archivos «*.Build.cs»
11 para los citados módulos adicionales, además de añadir las referencias pertinentes
12 en el archivo «Target.cs». Además, el módulo principal deberá contener la etiqueta
13 «IMPLEMENT_PRIMARY_GAME_MODULE» y el resto de módulos secundarios la eti-
14 queta «IMPLEMENT_GAME_MODULE». La herramienta de compilación de Unreal
15 Engine deberá entonces descubrir automáticamente los módulos y compilar los DLL
16 adicionales.

17 Limitaciones

18 Unreal Engine soporta la creación de módulos que son dependientes entre sí (es de-
19 cir, que ambos contienen importaciones y exportaciones que hacen referencia al otro),
20 pero no es ideal para los tiempos de compilación. Los módulos sin dependencias cru-
21 zadas son difíciles de diseñar y mantener, pero a cambio el código es más limpio por
22 ello.

23 6.2.2 Plugins

24 Muchos subsistemas de Unreal Engine fueron diseñados para ser extensibles, per-
25 mitiéndole al usuario añadir nuevas funcionalidades y modificar la funcionalidad exis-
26 tente sin modificar el código del propio motor directamente. Se pueden crear nuevos
27 tipos de archivos, agregar nuevos elementos en el menú, nuevos comandos a la barra
28 de herramientas o incluso agregar nuevas funciones y sub-modos de editor.

29 Anatomía

30 Los plugins con código tendrán una carpeta fuente («Source»). Esta carpeta con-
31 tendrá un o más directorios con el código fuente del código para el plugin. Nótese que
32 los plugins a menudo contienen código, pero no tienen por qué contenerlo.

Para plugins con módulos de código, el plugin tendrá su propia carpeta de binarios («Binaries») que contendrá el código compilado para ese plugin. También se guardarán archivos de compilación temporales en la carpeta «Intermediate» dentro del directorio del plugin.

Los plugins pueden tener su propia carpeta de contenido («Content») que contendrá assets específicos para ese plugin.

6.2.3 Clases de gameplay

Cada clase de gameplay en Unreal Engine está compuesta por una clase cabecera («.h») y una clase de archivo fuente («.cpp»). La clase cabecera contiene las declaraciones de la clase y sus miembros, como variables y funciones, mientras que la clase de archivo fuente es donde la funcionalidad de la clase se define implementando las funciones que pertenecen a la clase.

Las clases en Unreal Engine tienen una estructura de nombres estandarizadas, por la que se podrá conocer instantáneamente qué tipo de clase es simplemente mirando a la primera letra, o prefijo. Los prefijos de las clases de gameplay son:

Prefijos de clases de gameplay en Unreal Engine	
Prefijo	Significado
A	Extiende desde la clase base «spawneable». Son los denominados «actores», y pueden ser emplazados directamente en el mundo o escenario del juego.
U	Extiende desde la clase base de todos los objetos de juego. No pueden ser instanciados directamente en el mundo, deben pertenecer a un «actor». Son generalmente objetos como «componentes».

Cuadro 6.1: Prefijos de clases de gameplay en Unreal Engine

Clases de cabecera

Las clases de juego o «gameplay» en Unreal Engine generalmente tienen archivos de cabecera por separado y únicos. Estos archivos suelen ser nombrados para coincidir con la clase que se está definiendo, menos por el prefijo «A» o «U», y usando la extensión «.h». Por tanto, la clase de cabecera para la supuesta clase «AAActor» sería «Actor.h». Nótese que aunque se recomienda hacer uso de esta nomenclatura no se establece ninguna relación formal entre los archivos por definirlos de ese modo (al menos

1 en la versión actual del motor).

2 Las clases de cabecera usan la sintaxis estándar de «C++», que se pueden combinar
3 con instrucciones especializadas como ya se vio anteriormente al inicio de la sección
4 (§6.2)).

5 Al principio de cada clase de cabecera de una clase de gameplay, el archivo de
6 cabecera generado (creado automáticamente) necesita incluirse. Por tanto, al principio
7 de la clase «ClasePrueba.h» deberá aparecer la siguiente línea:

8 `#include "ClasePrueba.generated.h"`

9 Declaración de clases

10 La declaración de clase indica el nombre de la clase, de qué clase hereda y, además,
11 cualquier función o variable que herede.

```
12 UCLASS([ especificador , especificador , ... ] ,
13         [ meta(clave=valor , clave=valor , ...) ])
14 class NombreDeLaClase : public NombreDeLaClasePadre
15 {
16     GENERATED_BODY()
17 }
```

18 La declaración consiste en una declaración estándar de «C++» para una clase, pero
19 además de la declaración estándar, como se ve en la pieza de código, se usa la ins-
20 trucción «UCLASS()» para añadir especificadores de clase y metadatos. Así mismo, la
21 etiqueta «GENERATED_BODY()» debe ser colocada justo al principio de la clase.

22 Especificadores de clases

23 Cuando declaramos una clase podemos añadir especificadores a la declaración para
24 controlar cómo es el comportamiento de la clase en varios aspectos del motor y del
25 editor. La siguiente lista muestra estos especificadores:

- | | |
|--------------------------|----------------------|
| ■ Abstract | ■ BlueprintType |
| ■ AdvancedClassDisplay | ■ ClassGroup |
| ■ AutoCollapseCategories | ■ CollapseCategories |
| ■ AutoExpandCategories | ■ Config |
| ■ Blueprintable | ■ Const |

- ConversionRoot
- CustomConstructor
- DefaultToInstanced
- DependsOn
- Deprecated
- DontAutoCollapseCategories
- DontCollapseCategories
- EditInlineNew
- HideCategories
- HideDropdown
- HideFunctions
- Intrinsic
- MinimalAPI
- NoExport
- NonTransient
- NotBlueprintable
- NotPlaceable
- PerObjectConfig
- Placeable
- ShowCategories
- ShowFunctions
- Transient
- Within

Implementación de clases

1

Todas las clases de juego (o «gameplay») para poder ser implementadas correctamente deben hacer uso de la etiqueta «GENERATED_BODY». Esto se hace, como acabamos de ver, en la clase de cabecera («.h») que define la clase y todas sus funciones y variables.

2
3
4
5

Los archivos de código fuente («.cpp») deben incluir los archivos de cabecera («.h») que contiene la declaración C++ de la clase, que normalmente es generada automáticamente pero puede ser creada manualmente si así se desea.

6
7
8

Constructores de clases

9

Los objetos en Unreal Engine usan constructores para asignar valores por defecto a propiedades así como realizar otras inicializaciones necesarias. El constructor de clase está emplazado en la clase de implementación y, por ejemplo, para la clase de implementación «AActor.cpp» el método del constructor será «AActor::AActor».

10
11
12
13

ITERACIÓN CERO

1

2 *The good parts of a book may be only something a writer is lucky enough to overhear or it may be the wreck*
3 *of his whole damn life – and one is as good as the other.*

4

Ernest Hemingway (1899–1961),

5

Novelist

6

7

T ODO TODO TODO

7.1 INICIALIZACIÓN DEL PROYECTO

El motor gráfico sobre el que vamos a trabajar, como se decía al inicio de esta memoria, es gratuito y para obtenerlo solamente tenemos que acceder a su página oficial, <https://www.unrealengine.com/>.

Con esto habremos obtenido el «launcher» de Epic Games / Unreal Engine por lo que una vez descargado e instalado, queda un paso más: descargar una versión concreta del motor, en nuestro caso elegiremos la versión «4.16».

Una vez descargada e instalada la versión deseada del motor, procedemos con el siguiente paso: para la confección del trabajo vamos a crear un nuevo proyecto de Unreal Engine en «C++» y sin contenido adicional, es decir, vamos a crear un proyecto limpio en «C++». Para ello, en el Launcher de Unreal Engine, hacemos click en el botón «Iniciar» y se nos abrirá una ventana de selección de proyecto. Como queremos crear uno nuevo, hacemos click en la pestaña «New Project» o «Nuevo Proyecto» y una vez en esa pestaña hacemos click en la pestaña «C++», seleccionamos «Basic Code» o «Código básico» y configuramos el proyecto de la siguiente manera:

- «Desktop / Console» (o «Escritorio / Consola»): Puesto que nuestro proyecto está destinado a dispositivos sobremesa, y no a móviles o tablets, es la opción que debemos marcar cuando creamos nuestro proyecto.
- «Maximum quality» (o «Máxima calidad»): La calidad, como se detalla en el documento de diseño anexo a esta memoria, no es una prioridad para el proyecto, pero intentaremos que el resultado final luzca lo máximo posible.
- «No starter content» (sin contenido inicial): Como se detallaba previamente, vamos a crear un proyecto en «C++» con el mínimo contenido, para conocer nuestro proyecto el máximo posible, y si en algún momento necesitamos material del contenido inicial lo añadiremos manualmente.

Una vez configurado, hacemos click en «Create Project» (o «Crear Proyecto»).

Con esto, habremos terminado la creación de nuestro proyecto y tendremos casi lista la base sobre la que empezar a trabajar.

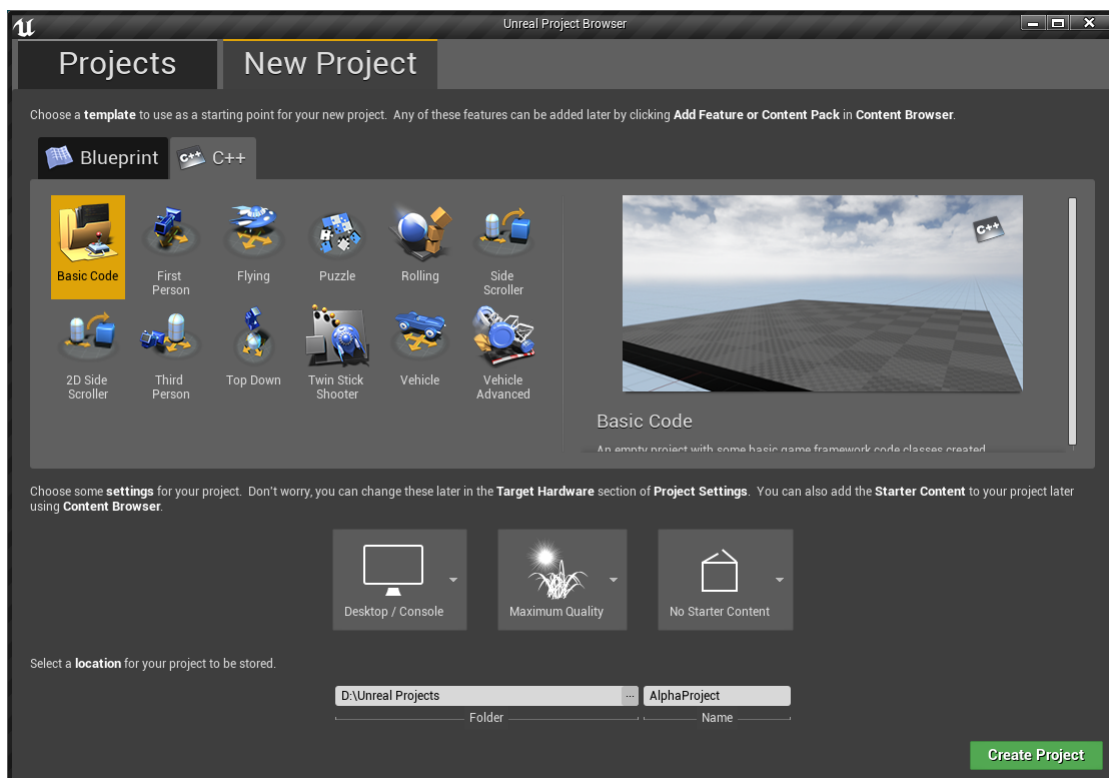


Figura 7.1: Ventana del creador de proyectos de Unreal Engine

7.2 PUESTA A PUNTO Y PRIMEROS PASOS

Antes de empezar a trabajar tenemos que realizar una serie de modificaciones:

TODO.

7.3 DIRECTRICES GENERALES Y RESEÑAS

Durante la ejecución del proyecto se seguirán una serie de pautas, métodos y en definitiva trabajos que no se van a ver reflejados en su totalidad en la documentación de las siguientes iteraciones. Es por ello que esta sección pretende explicarlos y que ese trabajo, difícil de documentar ya que en muchos casos es paralelo al trabajo principal de cada iteración o engloba más de una iteración, esté documentado.

7.3.1 Uso de animaciones profesionales «Mixamo»

Mixamoquoting

1

«Mixamo» es una empresa filial de la conocida «Adobe» dedicada al mundo de la animación 3D y que se centraba en generar estas animaciones para su posterior comercialización. Actualmente, todas sus animaciones son de libre uso, por lo que supone una gran oportunidad para un proyecto indie como este.

2

3

4

5

Las animaciones son de máxima calidad, tanto es así que se usan en juegos con alto presupuesto o ingresos: el juego más vendido del momento en PC, «Playerunknown's Battlegrounds» [8], utiliza varias de las animaciones de «Mixamo» y las sigue incorporando junto con nuevas funcionalidades en sus actualizaciones mensuales.

6

7

8

9

Sin embargo, hay un gran problema derivado del uso de estas animaciones en concreto: el personaje que se usa en este proyecto utiliza el esqueleto estándar de Unreal Engine, y hace ya muchas versiones del motor gráfico «Mixamo» retiró el soporte para personajes que usen el esqueleto «por defecto» de Unreal Engine.

10

11

12

13

La manera de solucionar esto es la incorporación de un esqueleto auxiliar, que no se utiliza en el proyecto salvo para este cometido, que nos servirá para realizar un «retargeting» a la animación. El proceso, una vez que tenemos listo el esqueleto destino y el esqueleto auxiliar es el siguiente:

14

15

16

17

1. Se

18

Hay que repetir todos los pasos del 2 al X para cada una de las animaciones que se quieren incorporar al proyecto. TODO.

19

20

TODO: Añadir al glosario retargeting. TODO: Capturas Retargeting.

21

7.3.2 Implementación del sistema de animación del personaje

22

Para la ... se utilizará una máquina de estados...

23

TODO.

24

PRIMERA ITERACIÓN

1

2 *The good parts of a book may be only something a writer is lucky enough to overhear or it may be the wreck*
3 *of his whole damn life – and one is as good as the other.*

4

Ernest Hemingway (1899–1961),

5

Novelist

6

7

T ODO TODO TODO

8.1 CARACTERÍSTICAS A DESARROLLAR

1. Mecánica: Movimiento del personaje. 1
 - Implementación del código de la mecánica. 2
 - Asignación de teclas y botones vinculados la mecánica. 3
 - Desarrollo de la animación del personaje relacionada con la mecánica. 4
2. Mecánica: Rotación del personaje. 5
 - Implementación del código de la mecánica. 6
 - Asignación de teclas y botones vinculados la mecánica. 7
 - Desarrollo de la animación del personaje relacionada con la mecánica. 8
3. Mecánica: Rotación del personaje vinculada a un rate. 9
 - Implementación del código de la mecánica. 10
 - Asignación de teclas y botones vinculados la mecánica. 11
4. Mecánica: Sprint. 12
 - Implementación del código de la mecánica. 13
 - Asignación de teclas y botones vinculados la mecánica. 14
 - Desarrollo de la animación del personaje relacionada con la mecánica. 15
5. Mecánica: Salto. 16
 - Implementación del código de la mecánica. 17
 - Asignación de teclas y botones vinculados la mecánica. 18
 - Desarrollo de la animación del personaje relacionada con la mecánica. 19
6. Mecánica: Agachado. 20
 - Implementación del código de la mecánica. 21
 - Asignación de teclas y botones vinculados la mecánica. 22
 - Desarrollo de la animación del personaje relacionada con la mecánica. 23
7. Mecánica: Tiempo bala (o «Time dilation»). 24

- Implementación del código de la mecánica.
- Asignación de teclas y botones vinculados la mecánica.
- Desarrollo de la animación del personaje relacionada con la mecánica.
- Gestión de adrenalina relacionada con la mecánica.

8. Mecánica: Parpadeo (o «Blink»).

- Implementación del código de la mecánica.
- Asignación de teclas y botones vinculados la mecánica.
- Desarrollo de la animación del personaje relacionada con la mecánica.
- Gestión de adrenalina relacionada con la mecánica.
- Implementación del código del indicador de blink disponible.

Además, durante esta iteración se realiza la inicialización y preparación del proyecto.

8.2 DISEÑO

1

Memorando técnico 0001	
Asunto	¿Cuál es el problema?
Resumen	¿Cuál es la solución propuesta?
Factores causantes	Descripción pormenorizada del problema
Solución	Descripción pormenorizada de la solución propuesta
Motivación	¿Por qué propone esta solución?
Cuestiones abiertas	Factores a tener en cuenta en la solución cuya dimensión se reconoce.
Alternativas	Otras soluciones consideradas y la razón por la que se excluyeron.

Cuadro 8.1: Memorando técnico 0001

Nota: Al tratarse de la primera iteración (por tanto, no existían soluciones anteriores que se pudiesen ver afectadas en esta iteración) y teniendo en cuenta que las características a implementar no originan conflictos entre sí, no existen modificaciones que afecten al diseño.

2
3
4
5

Memorando técnico 0001

Asunto	Creación del personaje controlable haciendo uso del «true first person camera»
Resumen	El modo «true first person camera» (cámara en primera persona real) implica que tengamos un modelo de personaje completo con una cámara anexa a la cabeza del modelo (por contraposición a la típica cámara en primera persona que utiliza normalmente un modelo en el que sólo se muestran las manos del personaje, impidiendo ver sus pies o su sombra).
Factores causantes	-
Solución	Creamos nuestro personaje extendiendo la clase ACharacter y usamos la propiedad «Mesh», heredada de la clase padre y del tipo «USkeletalMeshComponent», para guardar nuestro modelo de personaje. Por otra parte, creamos un «camera arm» (o brazo de cámara) para unir la cabeza del personaje con nuestra cámara, que también crearemos. Ajustamos el brazo de cámara para que la cámara quede justo a la altura de los ojos del personaje y por delante del mismo y modificamos la propiedad «ProbeSize» del brazo de cámara, que por defecto prácticamente es inservible para nuestro propósito, para que cuando la cámara colisione un obstáculo retroceda en lugar de traspasarlo (pero sin que llegue a meterse dentro de la cabeza del modelo del personaje).
Motivación	Esta decisión genera muchos problemas, como que la cámara pueda traspasar paredes o introducirse dentro del personaje si no está correctamente calibrado, pero a la vez le da una mayor sensación de realismo, mejorando la experiencia de juego del usuario (especialmente en un título encaminado a una adaptación, en un futuro, a la realidad virtual).
Cuestiones abiertas	En futuras iteraciones debe ser revisado ya que si bien con estas mecánicas iniciales el valor de «ProbeSize» es idóneo, con mecánicas que abordaremos en iteraciones posteriores en las que el personaje debe estar más pegado a obstáculos (por ejemplo cuando esté colgando de una cornisa) el valor actual de dicha variable puede ocasionar que la cámara se introduzca dentro del modelo del personaje.
Alternativas	La alternativas principales son dos: usar cámara en primera persona sin modelo o usar una cámara en primera persona con un modelo brazos y manos. La primera, aunque es por mucho la solución más fácil de implementar, queda descartada rápidamente en este género ya que por ejemplo debemos saber cuándo el personaje está agarrado a un saliente, cuándo está escalando, etc. La segunda sí podría

Memorando técnico 0001	
Asunto	Movimiento del personaje.
Resumen	En función de input proporcionado por el jugador se añade movimiento en esa dirección.
Factores causantes	-
Solución	Al detectar un input por parte del jugador relacionado con una las cuatro acciones de movimiento (delante, atrás, izquierda o derecha) se añade movimiento al personaje en la dirección especificada. Se diferencia movimiento delante-atrás e izquierda-derecha, siendo el resultado final la combinación de ambos movimientos.
Motivación	Solución propuesta por seguir el estándar.
Cuestiones abiertas	Debemos tener en cuenta que la solución generada deberá ser modificada en las siguientes iteraciones ya que no queremos que nuestro personaje pueda moverse en todo momento (por ejemplo, no queremos que el personaje se mueva cuando esté escalando un saliente).
Alternativas	-

Cuadro 8.3: Memorando técnico 0001

Memorando técnico 0001	
Asunto	Rotación de cámara y personaje
Resumen	Debemos permitir al usuario modificar la rotación de la cámara y, a su vez, alterar la rotación horizontal del personaje para que siempre esté mirando en la dirección de la cámara.
Factores causantes	-
Solución	La solución consta de dos partes muy diferenciadas. Por un lado cuando se detecta un cambio en los ejes asignados al movimiento de la cámara se añade rotación a la misma (se añade por separado respecto al eje X e Y, es decir, si el usuario está moviendo el eje X de la cámara con un valor de «10» hacia arriba y el eje Y de la misma con un valor de «5» hacia la izquierda se añadirá por una parte rotación hacia arriba de «10» puntos y por otra rotación hacia la derecha de «-5» puntos) y, por otro lado, marcamos como verdadera la variable booleana «bUsePawnControlRotation» de la cámara primera persona del jugador para que controle la rotación del personaje.
Motivación	Solución propuesta por seguir el estándar.
Cuestiones abiertas	Debemos tener en cuenta que la solución generada deberá ser modificada en las siguientes iteraciones ya que no queremos que nuestra cámara pueda moverse con total libertad en según qué circunstancias (por ejemplo, no queremos que se mueva cuando el personaje esté agarrando un saliente, ya que alteraría la posición del personaje).
Alternativas	-

Cuadro 8.4: Memorando técnico 0001

Memorando técnico 0001	
Asunto	Señal de rotación proveniente de stick analógico
Resumen	Existen dos métodos típicos de movimiento de cámara: el producido al desplazar un ratón y el que se origina al mover un stick analógico de un mando. En el primero no hay problema, puesto que el usuario es capaz de regular en todo momento la entrada que quiere dar dependiendo de la velocidad a la que mueva el ratón, y lo puede hacer prácticamente sin límites (o sin que esto resulte un problema). Al utilizar un stick analógico de un mando eso no es del todo así, puesto que el usuario puede elegir entre A) no mover el stick (input = 0), B) moverlo al máximo (input = 1) o C) moverlo en un estado intermedio (input mayor que 0 y menor que 1), por lo que debemos ponderar esa entrada proporcionada por el usuario, lo que sería equivalente a establecerle una sensibilidad.
Factores causantes	El factor causante en este caso no es una característica del producto, sino que se debe a la naturaleza de los sticks analógicos.
Solución	El remedio en este caso es ayudarnos de la solución obtenida para mover la cámara con el ratón y multiplicarle un «rate» (sensibilidad).
Motivación	Solución propuesta por seguir el estándar.
Cuestiones abiertas	-
Alternativas	-

Cuadro 8.5: Memorando técnico 0001

Memorando técnico 0001	
Asunto	Señal de rotación proveniente de stick analógico
Resumen	Existen dos métodos típicos de movimiento de cámara: el producido al desplazar un ratón y el que se origina al mover un stick analógico de un mando. En el primero no hay problema, puesto que el usuario es capaz de regular en todo momento la entrada que quiere dar dependiendo de la velocidad a la que mueva el ratón, y lo puede hacer prácticamente sin límites (o sin que esto resulte un problema). Al utilizar un stick analógico de un mando eso no es del todo así, puesto que el usuario puede elegir entre A) no mover el stick (input = 0), B) moverlo al máximo (input = 1) o C) moverlo en un estado intermedio (input mayor que 0 y menor que 1), por lo que debemos ponderar esa entrada proporcionada por el usuario, lo que sería equivalente a establecerle una sensibilidad.
Factores causantes	El factor causante en este caso no es una característica del producto, sino que se debe a la naturaleza de los sticks analógicos.
Solución	El remedio en este caso es ayudarnos de la solución obtenida para mover la cámara con el ratón y multiplicarle un «rate» (sensibilidad).
Motivación	Solución propuesta por seguir el estándar.
Cuestiones abiertas	-
Alternativas	-

Cuadro 8.6: Memorando técnico 0001

8.3 IMPLEMENTACIÓN

1

Identificador	Descripción de la acción de alto nivel			
TODO	Añade movimiento en la dirección del personaje			
Métodos de alto nivel				
[void] MoveFoward (Value:float)				
Pasos (Usar Pseudocódigo o similar)				
1. Cuando se hace uso del enlace asignado al movimiento frontal / trasero del personaje (llamado «MoveFoward»), comprueba que el valor de 'Value' sea distinto de 0.				
1.1. Si es distinto de 0, añade movimiento equivalente al valor de 'Value' en la dirección del vector dirección del actor. Si el valor de 'Value' es positivo el personaje avanzará en la dirección indicada y si es negativo retrocederá.				
1.2. Si es igual a 0, no se hace nada.				
Métodos de bajo nivel necesarios				
Paso	Clase	Método	Mem. Téc.	IU
1.1	APawn	[void] AddMovementInput (Direction:FVector, Value:float)	TODO	TODO

3

Identificador	Descripción de la acción de alto nivel			
TODO	Añade movimiento a izquierda o derecha de la dirección del personaje			
Métodos de alto nivel				
[void] MoveRight (Value:float)				
Pasos (Usar Pseudocódigo o similar)				
1. Cuando se hace uso del enlace asignado al movimiento lateral del personaje (llamado «MoveRight»), comprueba que el valor de 'Value' sea distinto de 0.				
1.1. Si es distinto de 0, añade movimiento equivalente al valor de 'Value' en la dirección del vector derecha del actor. Por tanto, si el valor de 'Value' es positivo el personaje ⁴ avanzará hacia la derecha y si es negativo hacia la izquierda.				
1.2. Si es igual a 0, no se hace nada.				
Métodos de bajo nivel necesarios				
Paso	Clase	Método	Mem. Técn.	IU
1.1	APawn	[void] AddMovementInput (Direction:FVector, Value:float)	TODO	TODO

5

Identificador	Descripción de la acción de alto nivel		
TODO	Añade rotación horizontal a la cámara		
Métodos de alto nivel			
[void] Turn (Value:float)			
Pasos (Usar Pseudocódigo o similar)			
1. Cuando se hace uso del enlace asignado al movimiento horizontal de la cámara (llamado «Turn»), se delega en el método «AddControllerYawInput» de la clase APawn, puesto que no es necesario definir una función en la clase de nuestro personaje para modificar o controlar alguna faceta del resultado.			
Métodos de bajo nivel necesarios			
Paso	Clase	Método	Mem. IU Téc.
1	APawn	[void] AddControllerYawInput (Value:float)	TODO

Identificador	Descripción de la acción de alto nivel			
TODO	Añade rotación vertical a la cámara			
Métodos de alto nivel				
[void] LookUp (Value:float)				
Pasos (Usar Pseudocódigo o similar)				
1. Cuando se hace uso del enlace asignado al movimiento vertical de la cámara (llamado «LookUp»), se delega en el método «AddControllerPitchInput» de la clase APawn, puesto que no es necesario definir una función en la clase de nuestro personaje para modificar o controlar alguna faceta del resultado.				
Métodos de bajo nivel necesarios				
Paso	Clase	Método	Mem. Técn.	IU
1	APawn	[void] AddControllerPitchInput (Value:float)	TODO	TODOC

Identificador	Descripción de la acción de alto nivel		
TODO	Añade rotación horizontal a la cámara		
Métodos de alto nivel			
[void] TurnAtRate (Rate:float)			
Pasos (Usar Pseudocódigo o similar)			
1. Cuando se hace uso del enlace asignado al movimiento horizontal de la cámara proveniente de un mando o similar (llamado «TurnRate»), se delega en el método «AddControllerYawInput» de la clase APawn, pero esta vez le mandamos la multiplicación del (1) Rate que envía el usuario (porcentaje, entre «-1.0» y «1.0» que mueve, por ejemplo, el eje del joystick del mando asignado), (2) de la sensibilidad horizontal (definida como «BaseTurnRate») y (3) por el tiempo en el que se renderiza cada frame.			
Métodos de bajo nivel necesarios			
Paso	Clase	Método	Mem. IU Téc.
1	APawn	[void] AddControllerYawInput (Value:float)	TODO
1	UWorld	[Float] GetDeltaSeconds ()	TODO

Identificador	Descripción de la acción de alto nivel		
TODO	Añade rotación vertical a la cámara		
Métodos de alto nivel			
[void] LookUpAtRate (Rate:float)			
Pasos (Usar Pseudocódigo o similar)			
1. Cuando se hace uso del enlace asignado al movimiento vertical de la cámara proveniente de un mando o similar (llamado «LookUpRate»), se hace eso del método «AddControllerPitchInput» de la clase APawn, pero esta vez le mandamos la multiplicación del (1) Rate que envía el usuario (porcentaje, entre «-1.0» y «1.0» que mueve, por ejemplo, el eje del joystick del mando asignado), (2) de la sensibilidad vertical (definida como «BaseLookUpRate») y (3) por el tiempo en el que se renderiza cada frame.			
Métodos de bajo nivel necesarios			
Paso	Clase	Método	Mem. IU Téc.
1	APawn	[void] AddControllerPitchInput (Value:float)	TODO
1	UWorld	[Float] GetDeltaSeconds ()	TODO

Identificador	Descripción de la acción de alto nivel
TODO	Comunica a la animación que el personaje está rotando
Métodos de alto nivel	
[void] RotationTrick (Value:float)	
Pasos (Usar Pseudocódigo o similar)	
1. Cuando se detecta rotación horizontal de la cámara de cualquier tipo (ya sea proveniente de una fuente que no necesita un «rateo», como un ratón, o de una que sí lo necesita, como un mando) se cambia el valor de «RotationInput» de la instancia de animación del protagonista.	

Identificador	Descripción de la acción de alto nivel			
TODO	Interactúa con el entorno / mensajes			
Métodos de alto nivel				
[void] OnAction ()				
Pasos (Usar Pseudocódigo o similar)				
1. Cuando se hace uso del enlace «Action» se comprueba si el juego está pausado				
1.1. Si el juego está pausado, lo reanuda.				
1.2. Si el juego no está pausado, lo pausa.				
Métodos de bajo nivel necesarios				
Paso	Clase	Método	Mem. Técn.	IU
1	UGameplayStatics	SetGamePaused (WorldContextObject:UObject, bPaused:bool)	TODO	TODO

IMPORTANTE: En el futuro se modificará esta mecánica, ya iteración no disponemos de todos los elementos necesarios para desarrollarla completamente.

Identificador	Descripción de la acción de alto nivel
TODO	El personaje entra en modo sprint
Métodos de alto nivel	
[void] OnSprintPressed ()	
Pasos (Usar Pseudocódigo o similar)	
1. Mientras se hace uso del enlace «Sprint» se cambia la velocidad a la que se mueve el personaje («MaxWalkSpeed») al valor de la velocidad de correr (guardada en la constante «SprintSpeed»).	

Identificador	Descripción de la acción de alto nivel
TODO	El personaje entra en modo caminar
Métodos de alto nivel	
[void] OnSprintReleased ()	
Pasos (Usar Pseudocódigo o similar)	
1. Cuando se deja de hacer uso del enlace «Sprint» se cambia la velocidad a la que se mueve el personaje («MaxWalkSpeed») al valor de la velocidad de andar (guardada en la constante «WalkSpeed»).	

Identificador	Descripción de la acción de alto nivel			
TODO	El personaje salta			
Métodos de alto nivel				
[void] OnJump ()				
Pasos (Usar Pseudocódigo o similar)				
1. Cuando se hace uso del enlace «Jump» se llama a la función «Jump» de la clase «ACharacter»				
Métodos de bajo nivel necesarios				
Paso	Clase	Método	Mem. Técn.	IU
1	ACharacter	[void] Jump ()	TODO	TODO

IMPORTANTE: Se deja preparada esta mecánica para el futuro, ya que se verá afectada en la próxima iteración.

Identificador	Descripción de la acción de alto nivel			
TODO	El personaje deja de saltar			
Métodos de alto nivel				
[void] StopJumping ()				
Pasos (Usar Pseudocódigo o similar)				
1. Cuando se deja de hacer uso del enlace «Jump» se llama a la función «StopJumping» de la clase «ACharacter»				
Métodos de bajo nivel necesarios				
Paso	Clase	Método	Mem. Técn.	IU
1	ACharacter	[void] StopJumping ()	TODO	TODO

¹ 8.4 PRUEBAS

² TODO: Descripción de las pruebas realizadas al software

8.5 DESPLIEGUE	1
TODO: Breve resumen de cómo se han desplegado los cambios en el sistema de producción.	2
	3

SEGUNDA ITERACIÓN

1

2 *The good parts of a book may be only something a writer is lucky enough to overhear or it may be the wreck*
3 *of his whole damn life – and one is as good as the other.*

4

Ernest Hemingway (1899–1961),

5

Novelist

6

7

T ODO TODO TODO

9.1	CARACTERÍSTICAS A DESARROLLAR	1
9.2	DISEÑO	2
9.3	IMPLEMENTACIÓN	3
9.4	PRUEBAS	4
9.5	DESPLIEGUE	5

TERCERA ITERACIÓN

1

2 *The good parts of a book may be only something a writer is lucky enough to overhear or it may be the wreck*
3 *of his whole damn life – and one is as good as the other.*

4

Ernest Hemingway (1899–1961),

5

Novelist

6

7

T ODO TODO TODO

10.1	CARACTERÍSTICAS A DESARROLLAR	1
10.2	DISEÑO	2
10.3	IMPLEMENTACIÓN	3
10.4	PRUEBAS	4
10.5	DESPLIEGUE	5

PART IV

CIERRE DEL PROYECTO

MANUAL DE USUARIO

The good parts of a book may be only something a writer is lucky enough to overhear or it may be the wreck of his whole damn life – and one is as good as the other.

*Ernest Hemingway (1899–1961),
Novelist*

*R*esumen de lo que va a ocurrir en el capítulo. ¿Cuál es el objetivo que tenemos con este capítulo?

11.1 SECCIÓN LIBRE	1
Estructurar en función del proyecto.	2

CONCLUSIONES

The good parts of a book may be only something a writer is lucky enough to overhear or it may be the wreck of his whole damn life – and one is as good as the other.

*Ernest Hemingway (1899–1961),
Novelist*

*R*esumen de lo que va a ocurrir en el capítulo. ¿Cuál es el objetivo que tenemos con este capítulo?

12.1	INFORME POST-MORTEM	1
	Qué es un informe post-mortem	2
12.1.1	Lo que ha ido bien	3
	■ Argumento a favor 1.	4
	■ Argumento a favor 2.	5
	■ Argumento a favor 3.	6
12.1.2	Lo que ha ido mal	7
	■ Argumento en contra 1.	8
	■ Argumento en contra 2.	9
	■ Argumento en contra 3.	10
12.1.3	Discusión	11
	En función de lo anterior, qué cambiaría si empezara hoy el proyecto de nuevo.	12
12.2	TRABAJOS FUTUROS	13
	Enumera los puntos abiertos y que no se han resuelto. Indica si darían lugar a otro proyecto y de qué forma se podría acotar.	14
		15

PART V

APPENDICES

DOCUMENTO DE DISEÑO

*U*n documento de diseño, conocido en la industria como GDD por sus siglas en inglés (Game Design Document), es un documento que recoge las características que conforman el videojuego, es decir, una síntesis del mismo.

Teniendo en cuenta las características de nuestro proyecto, no se pretende elaborar un documento de diseño al uso sino un documento cuyo principal fin sea permitir generar, a partir del mismo, los requisitos del producto software que se va a desarrollar.

Además, atendiendo de nuevo a las características específicas de nuestro proyecto, numerosas secciones del documento de diseño se verían solapadas con la documentación del proyecto ya descrita en el cuerpo de este documento. En caso de que se produzca dicho solapamiento, se citará la sección correspondiente.

A.1 VISIÓN DE CONJUNTO

La visión de conjunto describe de manera global videojuego, su género, la forma de jugarlo y los elementos que lo componen.

Este apartado se describe en la sección «Motivación», donde mediante frases cortas se describen los elementos de los que consta el videojuego cuyo desarrollo está ligado a este proyecto (Ver sección §2.1).

A.1.1 Plataformas de destino

Las plataformas de destino son aquellas para las cuales se está desarrollando el software o se tiene previsto su desarrollo.

A pesar de que nuestro motor gráfico permite la exportación de nuestro producto a diversas plataformas tales como PlayStation 4, Xbox One, Nintendo Switch, Linux o Mac (entre otras), la única plataforma de destino en la que nos centraremos durante la realización del proyecto será Microsoft Windows.

A.1.2 Requisitos mínimos objetivo

Los requisitos mínimos hacen alusión a las características que, como mínimo, deberá tener un equipo para hacer funcionar el videojuego.

El proyecto que nos ocupa no someterá a ninguno de los componentes a una carga alta de trabajo: el poligonaje de los modelos no será alto, las texturas usadas no serán de gran resolución, por norma general no se hará un uso avanzado de la iluminación (por ejemplo, uso excesivo de sombras dinámicas), etc. Asimismo, el motor gráfico usado se caracteriza por no hacer un uso agresivo de los recursos disponibles.

Los requisitos mínimos, o en este caso los requisitos mínimos objetivo, coinciden con los dispositivos descritos en la sección «Diseño arquitectónico» (Ver sección §6.2). Los requisitos mínimos como tal no podrán ser calculados hasta la finalización del proyecto.

A.1.3 Estilo gráfico

El estilo gráfico del producto se podría enmarcar como realista: tanto el personaje como el escenario que se usará como tutorial se clasificarían en este estilo. Sin embargo, debemos tener presente que los niveles prototipo que se pretenden utilizar para recrear la forma en la que se jugaría el juego se desmarcarían de este estilo, ya que usarían formas y texturas mucho más básicas al tratarse de niveles prototipo.

Dentro de este estilo realista, se podría clasificar el universo donde tiene lugar el juego como fantástico.

A.1.4 Estilo sonoro

La música del juego... TODO.

Los efectos de sonido (utilizados cuando el personaje anda, salta, cae, etc.) se podrían clasificar como realistas.

Respecto a las voces y diálogos el producto carecerá de los mismos.

A.2 COMIENZO

A.2.1 Menú principal

A.2.2 Comienzo del juego e introducción

A.2.3 In-game HUDs y menús

A.3 INTERFAZ DE USUARIO

A.3.1 Menú principal

En el menú principal se le mostrará al jugador las siguientes opciones:

- **Continuar:** Carga el último nivel jugado y sitúa al personaje en el último punto de control.

- **Nuevo juego:** Carga el nivel del tutorial y sitúa al personaje al inicio del mismo.
- **Opciones:** Muestra las opciones del juego, que son:
 - TODO.
- **Salir:** Al seleccionar esta opción la aplicación se cerrará.

A.3.2 In-game HUD

Durante el tiempo de juego, al jugador se le mostrará la siguiente información. TODO.

El HUD se compone de los siguientes elementos:

- **Indicador de adrenalina:**
 - **Tipo:** Barra.
 - **Funcionalidad:** Muestra la cantidad de adrenalina restante.
 - **Posición:** Adyacente al límite inferior de la pantalla.
 - **Tamaño:** Ancho de la pantalla.
 - **Visibilidad:** Se muestra si la adrenalina no está cargada por completo.
- **Punto de referencia:**
 - **Tipo:** Imagen.
 - **Funcionalidad:** Imagen (punto) de referencia que se usa principalmente para evitar el «motion sickness» o «cinetosis», en otras palabras, para suavizar mareos en jugadores con propensión a tenerlos.
 - **Posición:** Equidistante de todos los lados de la pantalla.
 - **Tamaño:** N/A.
 - **Visibilidad:** Siempre.
- **Indicador de habilidad «blink»:**
 - **Tipo:** Imagen.
 - **Funcionalidad:** Indica cuando el jugador puede ejecutar la habilidad blink.

- **Posición:** Alrededor del punto de referencia.
- **Tamaño:** N/A.
- **Visibilidad:** Sólo cuando el personaje pueda realizar la habilidad blink y esté mirando a una superficie propicia para ello.

■ **Caja de tutorial:**

- **Tipo:** Caja y texto.
- **Funcionalidad:** Caja que muestra un texto con información correspondiente al tutorial que está en ese momento activo.
- **Posición:** Centro de la pantalla.
- **Tamaño:** N/A.
- **Visibilidad:** Se mostrará siempre que el personaje esté inmerso en un tutorial, mostrando el texto relativo a cada tutorial.

■ **Mensaje de fin de juego:**

- **Tipo:** Texto.
- **Funcionalidad:** Muestra el mensaje de fin de juego.
- **Posición:** Centro de la pantalla.
- **Tamaño:** N/A.
- **Visibilidad:** Aparecerá al alcanzarse cualquiera de las condiciones de fin de juego.

A.3.3 In-game Options

TODO.

A.3.4 Pantalla de fin de juego

Cuando el personaje entre en un estado de fin de juego, la pantalla pasará a negro y se mostrará el mensaje de final de juego sobre la misma hasta que el jugador presione la tecla o botón indicado para reiniciar el nivel desde el último punto de control.

A.3.5 Pantalla de selección de nivel

TODO.

Tipos de nivel

El producto consta de dos tipos de nivel, muy diferenciados:

- Nivel menú: Utilizado para mostrar el menú del juego.
- Nivel tipo tutorial: Enseñará al jugador a usar las distintas mecánicas del juego.
- Nivel tipo regular: Se recrea un escenario real de juego.

Lista completa de niveles

La etapa de desarrollo que tendrá lugar durante el desarrollo de este proyecto generará 3 niveles en total:

- Nivel tutorial.
- Nivel prototipo 1.
- Nivel prototipo 2.

El nivel del tutorial tendrá un nivel de acabado representativo del acabado final del producto (por tanto estará correctamente iluminado, texturizado, decorado, etc.), mientras que el resto de niveles serían niveles de ejemplo y tendrán un acabado más simple.

A.4 GAMEPLAY

A.4.1 Mecánicas

Las mecánicas son el conjunto de acciones que puede realizar el jugador y las reglas por las que se rige el universo del videojuego. Atendiendo a este proyecto, podemos establecer claros grupos entre las mismas:

- **Mecánicas de bajo perfil:** En este proyecto, nos referimos a mecánicas de bajo perfil cuando hablamos de mecánicas dependientes del jugador y que podrá usar en cualquier momento.
- **Mecánicas de alto perfil:** Se denominan así al conjunto de mecánicas que el jugador podrá utilizar cuando disponga de poderes específicos.
- **Mecánicas de escalada:** El conjunto de mecánicas relacionadas con la escalada. Son mecánicas dependientes del jugador y que se pueden llevar a cabo en cualquier momento, por lo que podrían englobarse en las mecánicas de bajo perfil pero se categorizan por separado debido a la importancia que tienen dentro del proyecto y al número de las mismas.
- **Mecánicas del entorno o universo:** Son mecánicas que tienen relación con cómo se relaciona el personaje con el entorno.

Mecánicas de bajo perfil

1. **Desplazamiento:** El jugador podrá desplazar el personaje por el entorno pulsando las teclas, botones o joysticks asignados para ello.
2. **Rotación:** El jugador podrá rotar al personaje y, a su vez, a la cámara utilizando el movimiento del ratón o el joystick asignado.
3. **Interaccionar:** El personaje podrá interaccionar con determinados elementos prefijados del entorno pulsando la tecla o el botón asignado para ello. Esta acción provocará por tanto cambios en el jugador o en el entorno.
4. **Saltar:**
 - El personaje podrá saltar, ganando altura hasta una altura máxima, mientras el jugador mantiene pulsada la tecla de salto.
 - Si la altura máxima de salto se ha alcanzado o el jugador cesa en su acción de presionar la tecla de salto el personaje dejará de ganar altura en el salto y comenzará a caer.
5. **Agachado:**
 - Mientras que el jugador mantenga pulsada la tecla vinculada a la acción de agacharse el personaje se agachará, cambiando la posición de la cámara y reduciendo su caja de colisión, permitiéndole acceder a zonas por las que no podría pasar erguido o bien esquivar peligros.

- Cuando el jugador deje de presionar dicha tecla el personaje volverá a estar de pie.

6. Correr:

- Mientras que el jugador mantenga pulsada la tecla de sprint el personaje pasará a desplazarse por el entorno con una velocidad mayor, acelerando además su animación.
- Cuando el jugador deje de presionar la tecla relacionada el personaje volverá a su velocidad habitual.

Mecánicas de alto perfil

1. **Tiempo bala (o time dilation):** El jugador podrá, presionando la tecla o el botón correspondiente, ralentizar el tiempo del entorno y del personaje (manteniendo el escenario levemente más ralentizado que el personaje, para de esta manera ganar una pequeña ventaja) para así poder sortear obstáculos que de otra forma le serían complicado o imposible de salvar.
2. **Parpadeo (o blink):** El jugador podrá teletransportar al personaje a la posición a la que está mirando si no hay obstáculos de por medio.

Mecánicas de escalada

1. **Wall running:** El personaje podrá correr verticalmente por una pared, hasta una altura máxima, siempre que sea una superficie adecuada para ello.
2. **Salto en wall running:** El personaje podrá utilizar la pared que está escalando para impulsarse hacia el lado contrario, cambiando de esta forma su orientación. Si el personaje combina esta mecánica con la anterior (es decir, si tenemos dos superficies en la que hacer wall running una en frente de otra) para encadenar varios wall runnings y permitirle subir a posiciones muy elevadas.
3. **Agarrarse a un saliente:** El personaje podrá permanecer agarrado a un saliente siempre que se reúnan las condiciones adecuadas.
4. **Escalar un saliente:**
 - Si está agarrado y no hay obstáculos que no le permitan subir, pulsando el botón de saltar podrá terminar de escalar el saliente.

- Si no está agarrado pero el saliente es de baja altura automáticamente escalará el saliente en vez de agarrarse a él.
5. **Desplazarse por un saliente:** Siempre que no haya obstáculos el personaje podrá desplazarse lateralmente por un saliente.
 6. **Dejarse caer de un saliente:** Presionando el botón de agacharse el jugador puede hacer que el personaje abandone el saliente al que estaba agarrado.
 7. **Salto lateral desde un saliente:** Cuando el personaje esté al final de un saliente, si no hay obstáculos de por medio, podrá saltar lateralmente desde ese saliente, permitiéndole así agarrarse a salientes que no estaban conectados con el original o llegar a zonas previamente inaccesibles.
 8. **Salto reverso desde un saliente:** Cuando el personaje esté agarrado a un saliente podrá impulsarse desde el mismo y realizar un salto en la dirección opuesta a la del saliente.

Mecánicas del entorno

1. Si el personaje entra en contacto con algún peligro del escenario (como balas, láseres, etc.) morirá instantáneamente.
2. El personaje, al caer...
 - ... Si la altura de caída es lo suficientemente pequeña, no pasará nada.
 - ... Si la altura de caída es lo suficientemente grande, no podrá moverse por unas décimas de segundo.
 - ... Si cae al vacío morirá instantáneamente.
3. El personaje morirá instantáneamente al abandonar el área de juego.

A.4.2 Controles

En esta sección se definen los enlaces entre las diferentes entradas al alcance del jugador y su traducción al universo del juego.

El producto ofrece al jugador dos métodos de entrada: «teclado y ratón» y «mando». Ambos métodos no son excluyentes entre sí, y si se dispone de los dos métodos de entrada en cualquier momento se podrá pasar de uno a otro, sin ser necesario reiniciar el videojuego.

A.4.3 Controles con teclado y ratón

Esquema

TODO.

Listado detallado

- Tecla **W**: Desplazamiento del jugador hacia el frente.
- Tecla **A**: Desplazamiento del jugador hacia atrás.
- Tecla **S**: Desplazamiento del jugador hacia la izquierda.
- Tecla **D**: Desplazamiento del jugador hacia la derecha.
- Eje **X** del ratón: Regula el movimiento horizontal de la cámara.
- Eje **Y** del ratón: Regula el movimiento vertical de la cámara.
- Tecla **E**: Botón de acción.
- Tecla **Shift Izquierdo**: Sprint.
- **Barra espaciadora**: Salto / Wall running / Salto en wall running / Escalar o saltar de saliente.
- Tecla **Ctrl Izquierdo**: Agachado / Abandonar saliente.
- Tecla **TAB** (Tabulador): Conmutador de tiempo bala.
- **Botón derecho** del ratón: Blink.

A.4.4 Controles con mando

Esquema

TODO.

Listado detallado

- Joystick izquierdo Eje **Y+**: Desplazamiento del jugador hacia el frente.
- Joystick izquierdo Eje **Y-**: Desplazamiento del jugador hacia atrás.

- Joystick izquierdo **Eje X-**: Desplazamiento del jugador hacia la izquierda.
- Joystick izquierdo **Eje X+**: Desplazamiento del jugador hacia la derecha.
- Joystick derecho **Eje X**: del ratón: Regula el movimiento horizontal de la cámara.
- Joystick derecho **Eje Y**: Regula el movimiento vertical de la cámara.
- Botón **A**: Botón de acción.
- Disparador **RT**: Sprint.
- Botón **B**: Salto / Wall running / Salto en wall running / Escalar o saltar de saliente.
- Botón **X**: Agachado / Abandonar saliente.
- Botón **Y**: Conmutador de tiempo bala.
- Disparador **LT**: Blink.

A.4.5 Modos de juego

Aunque el jugador sólo perciba uno, en realidad se establecerán dos modos de juego: el primero se utilizará para navegar por los menús y el segundo será la forma de jugar al juego en sí.

A.4.6 Ganando el juego

La única forma de ganar el juego es completando la fase, es decir, cuando el personaje entre en contacto con la caja de colisión deseada al final de cada nivel.


A.4.7 Logros / Trofeos

Los logros son medallas otorgadas al jugador al realizar un avance específico dentro del juego y que dan una idea del progreso dentro del mismo.


Relacionados con la historia

El jugador los obtendrá al hacer progresos en lo que se podría denominar «la historia», en este caso al finalizar un nivel.

[...]

Logro historia - Juego		
Logo		
Obtención	Otorgado al jugador por completar el juego	
Idioma	Nombre	Descripción
Español	Completado	Completa el juego
Inglés	Finished	Finish the game

Cuadro A.1: Logro historia - Juego

Logro historia - Tutorial		
Logo		
Obtención	Otorgado al jugador por completar el tutorial	
Idioma	Nombre	Descripción
Español	Puesta a punto	Completa el tutorial
Inglés	Setup	Finish the tutorial

Cuadro A.2: Logro historia - Tutorial


Pruebas de tiempo

Son aquellos otorgados al jugador al superar un escenario por debajo de un determinado tiempo preestablecido.


[...]

Coleccionables

El jugador obtendrá logros al recoger objetos repartidos por el universo del juego que el personaje podrá recolectar al entrar en contacto con ellos. Concretamente el

Logro historia - Capítulo I		
Logo		
Obtención	Otorgado al jugador por completar el primer nivel	
Idioma	Nombre	Descripción
Español	Capítulo I	Completa el capítulo I
Inglés	Chapter I	Finish the chapter I

Cuadro A.3: Logro historia - Capítulo I

Logro historia - Capítulo II		
Logo		
Obtención	Otorgado al jugador por completar el segundo nivel	
Idioma	Nombre	Descripción
Español	Capítulo II	Completa el capítulo II
Inglés	Chapter II	Finish the chapter II

Cuadro A.4: Logro historia - Capítulo II

jugador obtendrá un logro en los siguientes casos:

Misceláneos

Además de los indicados, el jugador será recompensado con un logro al realizar las siguientes acciones satisfactoriamente:

100 %

Verifica que el jugador ha completado el resto de logros.

Logro prueba de tiempo - Capítulo I

Logo



Obtención Otorgado al jugador por completar el primer nivel por debajo del tiempo establecido

Idioma	Nombre	Descripción
Español	Prueba de tiempo: Capítulo I	Completa el capítulo I por debajo del tiempo establecido
Inglés	Time trial: Chapter I	Finish the chapter I under the time limit

Cuadro A.5: Logro prueba de tiempo - Capítulo I

Logro prueba de tiempo - Capítulo II

Logo




Obtención Otorgado al jugador por completar el segundo nivel por debajo del tiempo establecido

Idioma	Nombre	Descripción
Español	Prueba de tiempo: Capítulo II	Completa el capítulo II por debajo del tiempo establecido
Inglés	Time trial: Chapter II	Finish the chapter II under the time limit


Cuadro A.6: Logro prueba de tiempo - Capítulo II

A.5 ASSETS

Para la elaboración del producto se usan gran cantidad de assets y de variada clasificación, la inmensa mayoría procedente de proyectos de libre uso de Unreal Engine

Logro coleccionables - 1 coleccionable		
Logo		
Obtención	Otorgado al jugador por encontrar un coleccionable (cualquiera de ellos)	
Idioma	Nombre	Descripción
Español	Coleccionables 1	Encuentra tu primer coleccionable
Inglés	Collectibles 1	Find your first collectible


Cuadro A.7: Logro coleccionables - 1 coleccionable

Logro coleccionables - 5 coleccionables		
Logo		
Obtención	Otorgado al jugador por encontrar cinco coleccionables cualquiera	
Idioma	Nombre	Descripción
Español	Coleccionables 5	Encuentra cinco coleccionables en total
Inglés	Collectibles 5	Find five collectibles in total


Cuadro A.8: Logro coleccionables - 5 coleccionables

creados por Epic Games (como por ejemplo assets usados en demos técnicas o los pertenecientes a packs de assets liberados).

Teniendo en cuenta esto, esta sección no pretende elaborar una lista detallada de todos los assets que se han usado en la creación del videojuego, sino los assets que se salgan de esa norma: assets de uso gratuito no pertenecientes a Epic Games o assets sin licencia de uso gratuito (de los cuales se disponga de sus derechos de uso, ya sea mediante su adquisición o por el permiso explícito de su creador, o bien assets realizados por el equipo del proyecto).

Logro coleccionables - 10 coleccionables		
Logo		
Obtención	Otorgado al jugador por encontrar diez coleccionables cualquiera	
Idioma	Nombre	Descripción
Español	Coleccionables 5	Encuentra diez coleccionables en total
Inglés	Collectibles 5	Find ten collectibles in total

Cuadro A.9: Logro coleccionables - 10 coleccionables


Logro coleccionables - 50 % de coleccionables		
Logo		
Obtención	Otorgado al jugador por encontrar el 50% de los coleccionables	
Idioma	Nombre	Descripción
Español	Coleccionables 50%	Encuentra la mitad de los coleccionables
Inglés	Collectibles 50%	Find a half of the collectibles

Cuadro A.10: Logro coleccionables - 50 % de coleccionables

En cada uno de los casos se explicará el uso específico que se ha hecho de cada asset, es decir, cómo ha repercutido al proyecto.

A.5.1 Personajes

Tratándose de un juego de un solo jugador (y sin selector de personajes ni ningún otro método de intercambio de personajes jugables) sólo se usa un asset, que es el siguiente:

Logro coleccionables - 100 % de coleccionables		
Logo		
Obtención	Otorgado al jugador por encontrar todos los coleccionables	
Idioma	Nombre	Descripción
Español	Coleccionables 100 %	Encuentra todos los coleccionables
Inglés	Collectibles 100 %	Find all the collectibles

Cuadro A.11: Logro coleccionables - 100% de coleccionables

Logro misceláneos - 100 saltos		
Logo		
Obtención	Otorgado al jugador por saltar 100 veces	
Idioma	Nombre	Descripción
Español	Salto 100	Salta 100 veces
Inglés	Jump 100	Jump 100 times

Cuadro A.12: Logro misceláneos - 100 saltos

A.5.2 Enemigos

TODOS

A.5.3 Animaciones

Las animaciones usadas en el proyecto tienen cuatro orígenes principalmente: animaciones extraídas del pack de animaciones gratuito de Epic Games (disponible para

Logro misceláneos - 1000 saltos		
Logo		
Obtención	Otorgado al jugador por saltar 1000 veces	
Idioma	Nombre	Descripción
Español	Salto 1000	Salta 1000 veces
Inglés	Jump 1000	Jump 1000 times


Cuadro A.13: Logro misceláneos - 1000 saltos

Logro misceláneos - 10000 saltos		
Logo		
Obtención	Otorgado al jugador por saltar 10000 veces	
Idioma	Nombre	Descripción
Español	Salto 10000	Salta 10000 veces
Inglés	Jump 10000	Jump 10000 times

Cuadro A.14: Logro misceláneos - 10000 saltos

su descarga desde el Marketplace de Unreal Engine), animaciones de la plantilla por defecto de tercera persona de Unreal Engine 4 (al crear esta plantilla, se generan estas animaciones), animaciones propiedad de Mixamo (cuyo uso es gratuito desde hace unos años) y animaciones realizadas específicamente para el proyecto generalmente a partir de un frame de una animación.

Una buena parte de las animaciones usadas, sea de la fuente que sea, ha tenido que ser modificada de alguna u otra forma para su uso en el proyecto pero, más específicamente, las animaciones de Mixamo tienen añadido un trabajo extra: dichas animaciones no están creadas a partir de un esqueleto compatible con Unreal Engine 4, por

Logro 100 %		
		
Logo		
Obtención	Otorgado al jugador por obtener el resto de logros	
Idioma	Nombre	Descripción
Español	100 %	Completa el juego y acaba con el resto de retos
Inglés	100 %	Finish the game and all the challenges

Cuadro A.15: Logro 100 %

Assets de personajes			
ID	Personaje	Procedencia	Beneficios aportados
#1	Protagonista	Adquisición	Estrictamente estéticos
Notas: Cambio meramente estético respecto al maniquí por defecto de Unreal Engine. Se compone de un modelado y textura (ya que usa el mismo esqueleto que el maniquí por defecto).			

Cuadro A.16: Assets de personajes

tanto ha sido necesario realizar un **retargeting** en cada una de ellas para adaptarlo al esqueleto que del personaje del que se hace uso (que sí usa un esqueleto compatible con el esqueleto por defecto de Unreal Engine 4).

A.5.4 Equipo y mejoras

TODO: Poderes

A.5.5 Entorno

TODO

A.5.6 Audio

TODO

BIBLIOGRAFÍA

- [1] AEVI. El videojuego en el mundo. Technical report, Asociación española del videojuego (AEVI), 2017. (page 4).
- [2] EpicGames. *Documentación Unreal Engine*. Epic Games, 2017. (page 49).
- [3] Newzoo. Esports revenues will reach 696 million dollars this year and grow to 1.5 billion dollars by 2020. Technical report, Newzoo, 2017. (pages XI, 6).
- [4] Newzoo. Global games market 2016 report. Technical report, Newzoo, 2017. (pages XI, 4).
- [5] SoftZone. Qué es steam direct y en qué se diferencia de steam greenlight, 2017. (page 8).
- [6] Steam. Tasa de steam direct y próximas actualizaciones de la tienda, 2017. (page 8).
- [7] ZehnGames. El desarrollo español en steam: Una lectura desde los datos. Technical report, Zehn Games, 2017. (pages XI, 8).
- [8] ZonaRed. Playerunknown's battlegrounds continúa en el trono de las ventas semanales de steam, 2017. (page 58).