

DESARROLLO DE VIDEOJUEGO INDIE EN UNREAL ENGINE 4

FERNANDO JOSÉ GARCÍA PADILLA

Trabajo fin de Grado

Supervisado por Dr. Pablo Trinidad Martín-Arroyo



Universidad de Sevilla

julio 2017

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

fergarpad@alum.us.es

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:

Desarrollo de videojuego indie en Unreal Engine 4

Lo cual firmo,

Fdo. D. Fernando José García Padilla
en la Universidad de Sevilla
04/07/2017

*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 aventurare 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	7
1.1.2.	El auge del videojuego independiente	10
1.1.3.	La irrupción de los trofeos / logros y su impacto	12
1.2.	Estado del arte	14
2.	Objetivos del proyecto	17
2.1.	Motivación	18
2.2.	Listado de objetivos	20
2.2.1.	Objetivos del producto	20
2.2.2.	Objetivos individuales	22
II	Organización del proyecto	23
3.	Metodología	25
3.1.	Estructura organizacional del proyecto	26
3.2.	Metodología de desarrollo	27
3.2.1.	Presentación de la metodología	27

ÍNDICE GENERAL

3.2.2. Resumen de la metodología	27
3.2.3. Adaptación de la metodología	29
4. Planificación	31
4.1. Resumen temporal del proyecto	32
4.2. Planificación inicial	32
4.3. Informe de tiempos del proyecto	38
5. Costes	41
5.1. Resumen de costes del proyecto	42
5.2. Costes de personal	42
5.3. Costes materiales	45
5.4. Costes indirectos	47
III Desarrollo del proyecto	51
6. Arranque	53
6.1. Lista de características	54
6.2. Diseño arquitectónico	57
6.2.1. Módulos de juego	58
6.2.2. Plugins	59
6.2.3. Clases de gameplay	60
7. Iteración cero	63
7.1. Inicialización del proyecto	64
7.2. Puesta a punto y primeros pasos	65
7.3. Directrices generales y reseñas	65

7.3.1. Uso de animaciones profesionales «Mixamo»	66
7.3.2. Implementación del sistema de animación del personaje	66
7.3.3. Control de que el personaje no abandone el mapa	66
8. Primera iteración	67
8.1. Características a desarrollar	68
8.2. Diseño	70
8.3. Implementación	76
8.4. Pruebas	81
8.5. Despliegue	82
9. Segunda iteración	83
9.1. Características a desarrollar	84
9.2. Diseño	84
9.3. Implementación	84
9.4. Pruebas	84
9.5. Despliegue	84
10. Tercera iteración	85
10.1. Características a desarrollar	86
10.2. Diseño	86
10.3. Implementación	86
10.4. Pruebas	86
10.5. Despliegue	86

IV Cierre del proyecto	87
11. Manual de usuario	89
11.1. Sección libre	90
12. Conclusiones	91
12.1. Informe post-mortem	92
12.1.1. Lo que ha ido bien	92
12.1.2. Lo que ha ido mal	92
12.1.3. Discusión	92
12.2. Trabajos futuros	92
V Appendices	93
A. Glosario	95
B. Documento de diseño	99
B.1. Visión de conjunto	100
B.1.1. Plataformas de destino	100
B.1.2. Requisitos mínimos objetivo	100
B.1.3. Estilo gráfico	101
B.1.4. Estilo sonoro	101
B.2. Comienzo	101
B.2.1. Menú principal	101
B.2.2. Comienzo del juego e introducción	101
B.2.3. In-game HUDs y menús	101
B.3. Interfaz de usuario	101

B.3.1.	Menú principal	101
B.3.2.	In-game HUD	102
B.3.3.	In-game Options	103
B.3.4.	Pantalla de fin de juego	103
B.3.5.	Pantalla de selección de nivel	104
B.4.	Gameplay	104
B.4.1.	Mecánicas	104
B.4.2.	Controles	107
B.4.3.	Controles con teclado y ratón	108
B.4.4.	Controles con mando	108
B.4.5.	Modos de juego	109
B.4.6.	Ganando el juego	109
B.4.7.	Logros / Trofeos	109
B.5.	Assets	112
B.5.1.	Personajes	114
B.5.2.	Enemigos	115
B.5.3.	Animaciones	115
B.5.4.	Equipo y mejoras	117
B.5.5.	Entorno	117
B.5.6.	Audio	118
Referencias bibliográficas		118

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

1.1. Beneficios de los videojuegos en Europa Occidental (datos en billones americanos) [14]	4
1.2. Evolución gráfica de los videojuegos	5
1.3. Relación entre el dinero generado por los videojuegos, la música grabada y el cine durante 2015 en España [7]	6
1.4. Audiencia y previsión de crecimiento de los espectadores de deportes electrónicos [13]	7
1.5. Comparativa, beneficios y previsiones entre los juegos para móvil y el resto de aplicaciones (datos en billones americanos) [12]	8
1.6. Relevancia del programa «Greenlight» en los lanzamientos indies [22]	10
1.7. Ejemplos de videojuegos indies se hicieron un hueco en el mercado	11
1.8. Captura del juego «E-Motion» (1990), para la videoconsola «Amiga», uno de los primeros juegos en implementar algún tipo de logro	12
1.9. Imagen del sistema de logros interno de «Minecraft» (2011), que se mantuvo hasta la versión 1.11	13
1.10. Imagen del juego «E.T.» para la «Atari 2600», uno de los máximos detonantes de la crisis del videojuego [15]	14
1.11. Fotograma de «La abadía del crimen» (1987), uno de los máximos estandartes de «la edad de oro del software español»	15
1.12. Imagen de la primera demostración técnica en tiempo real de Unreal Engine 4, titulada «Infiltrator»	16
2.1. Fotograma de «Bioshock» (2007), uno de los pocos «First Person Shooter» («FPS») que se supieron diferenciar dentro de la saturación del género	18

ÍNDICE DE FIGURAS

2.2. Captura de «Enter the gungeon» (2016), ejemplo de la forma más tradicional de bullet hell en dos dimensiones	19
3.1. Organigrama del proyecto	26
3.2. Esquema del desarrollo basado en funcionalidades	28
3.3. Esquema del desarrollo basado en funcionalidades adaptado para el proyecto	30
4.1. Equivalencias fases-metodología	34
6.1. Arquitectura Unreal Engine 4	57
7.1. Ventana del creador de proyectos de Unreal Engine	65

ÍNDICE DE CUADROS

4.1. Tabla resumen de tiempos y planificación	32
4.2. Planificación temporal de fases e iteraciones (planificación)	35
4.3. Planificación temporal detallada de fases e iteraciones	36
4.4. Tabla del calculo de horas por rol en iteraciones	37
4.5. Tabla resumen de horas de trabajo por rol	37
4.6. Tabla de tiempos empleados en fases e iteraciones detallados	38
4.7. Tabla de desviación de tiempo por iteración	39
4.8. Tabla de retrasos del proyecto	39
5.1. Tabla resumen de costes	42
5.2. Tabla salarial BOE 2009 (simplificada)	43
5.3. Tabla salarial del proyecto por hora	44
5.4. Tabla sueldos netos	45
5.5. Tabla de características del dispositivo sobremesa	46
5.6. Tabla de características del dispositivo portátil	46
5.7. Tabla de amortización de materiales	47
5.8. Tabla de costes indirectos	48
6.1. Prefijos de clases de gameplay en Unreal Engine	60
8.1. Memorando técnico 0001	70
8.2. Memorando técnico 0001	71

8.3. Memorando técnico 0001	72
8.4. Memorando técnico 0001	73
8.5. Memorando técnico 0001	74
8.6. Memorando técnico 0001	75
B.1. Logro historia - Juego	110
B.2. Logro historia - Tutorial	110
B.3. Logro historia - Capítulo I	111
B.4. Logro historia - Capítulo II	111
B.5. Logro prueba de tiempo - Capítulo I	112
B.6. Logro prueba de tiempo - Capítulo II	112
B.7. Logro colecciónables - 1 colecciónable	113
B.8. Logro colecciónables - 5 colecciónables	113
B.9. Logro colecciónables - 10 colecciónables	114
B.10. Logro colecciónables - 50% de colecciónables	114
B.11. Logro colecciónables - 100% de colecciónables	115
B.12. Logro misceláneos - 100 saltos	115
B.13. Logro misceláneos - 1000 saltos	116
B.14. Logro misceláneos - 10000 saltos	116
B.15. Logro 100%	117
B.16. Assets de personajes	117

PARTE I

INTRODUCCIÓN

CONTEXTO

1

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

2

3

4

5

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 [3, 14] 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 en el pasado año [14].

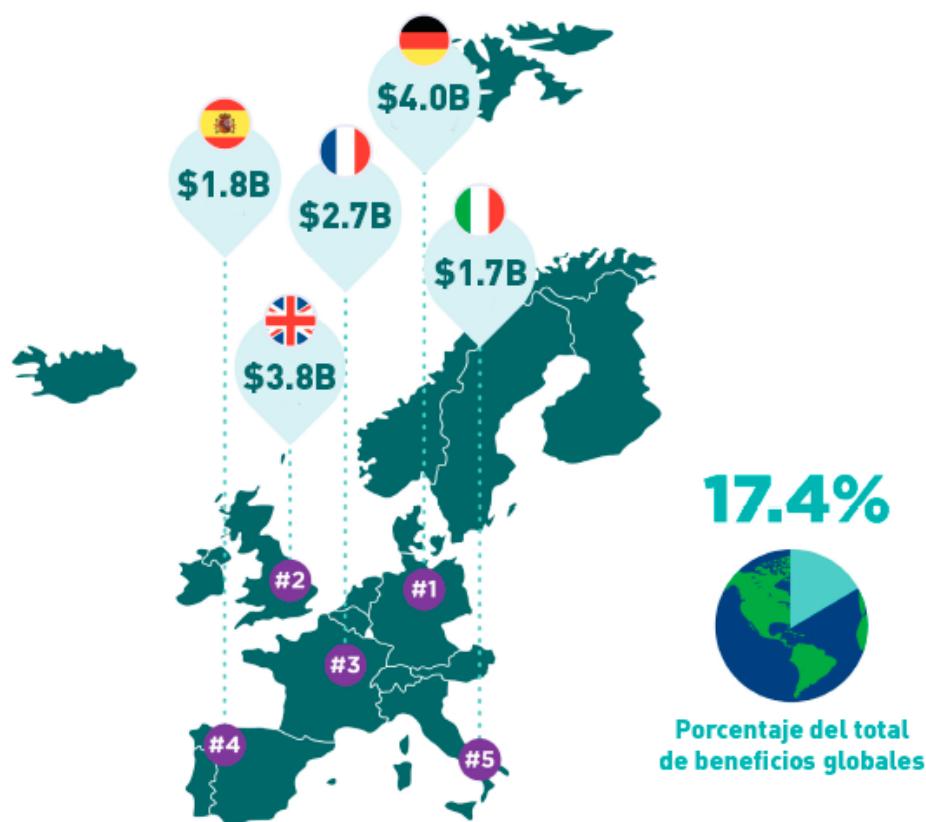


Figura 1.1: Beneficios de los videojuegos en Europa Occidental (datos en billones americanos) [14]

La transformación del sector no tiene sólo que ver con cifras económicas y no me gustaría pasar por alto algo que bajo mi punto de vista es bastante más interesante y que, aunque bien daría para un trabajo aparte, describe perfectamente el giro tan drástico al que se está viendo sometida la industria en los últimos años: hace no tanto tiempo los videojuegos eran un territorio un tanto desconocido, lo que provocaba que en algunas ocasiones fuesen vistos con cierto recelo por una parte de la población. Esta tendencia se ha revertido vertiginosamente en un breve lapso de tiempo, e incluso se podría decir que los videojuegos, en gran medida, han pasado a formar parte de la cultura popular: las grandes empresas textiles hacen camisetas con referencias a videojuegos o videoconsolas reconocidas; se realizan grandes eventos mediáticos en torno videojuegos o personas relacionadas con los videojuegos, a las que se les da el trato de auténticas estrellas del rock; se crean canales de televisión en grandes plataformas audiovisuales dirigidos únicamente a los videojuegos y podríamos decir que toda esta vorágine alcanza su súmmum cuando en un evento de tanta proyección internacional como es el acto de clausura de los juegos olímpicos, el primer ministro del país que recibe el testigo para organizarlos en cuatro años, en este caso Japón, aparece disfrazado de un reconocido personaje de videojuegos: «Mario».



Figura 1.2: Evolución gráfica de los videojuegos

Aún queda un largo camino por recorrer, es un problema que persigue al sector desde su creación, pero en los últimos años hemos vivido una apertura de los videojuegos a la sociedad de tal magnitud que una parte relevante de la misma lo ha pasado a ver de una forma distinta, dejando de lado ese citado recelo o cualquier connotación negativa, lo cual sin duda no puede ser más que favorable para la industria y su crecimiento.

1 Estableciendo un símil, podríamos comparar en algunos aspectos la fase en la que
2 se encuentra la industria del videojuego con aquella en la que se podría encontrar la
3 industria cinematográfica hace unas décadas: el sector está en etapa de expansión, cada
4 vez más universidades empiezan a formar en ámbitos relacionados directamente con
5 los videojuegos, se crean cada vez más puestos de trabajo bajo su cobijo [20] e incluso
6 están creciendo verdaderas celebridades en torno a la industria, ya que es habitual que
7 las personas que son consumidores de la industria conozcan directores de juegos, com-
8 positores o jugadores profesionales, por citar unos ejemplos. Igualmente, no todos los
9 aspectos el sector cinematográfico está más avanzado, ya que no podemos comparar
10 las industrias del videojuego y del cine pasando por alto que la primera, pese a su más
11 que notable juventud, consigue doblar en beneficios a la segunda en nuestro país [7].

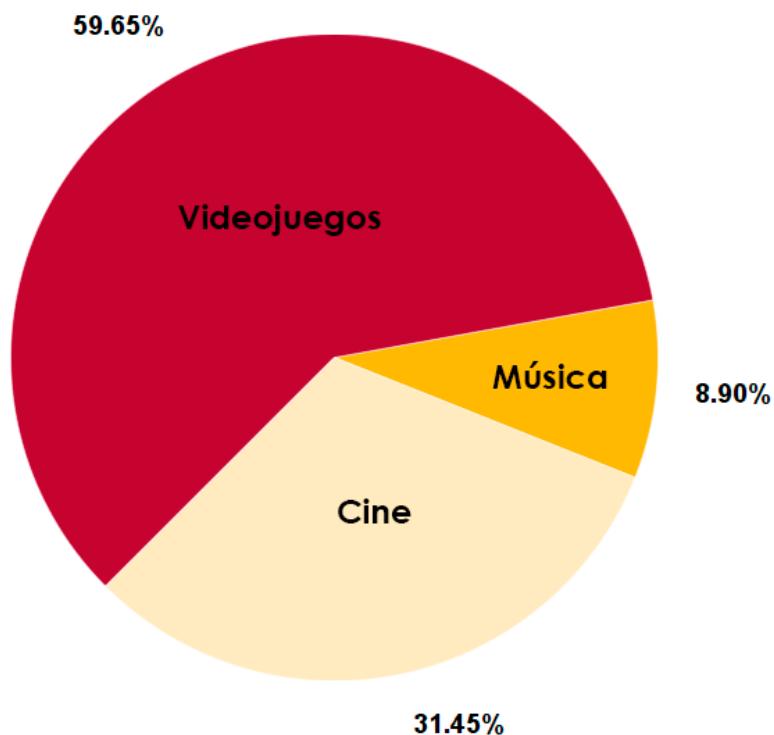


Figura 1.3: Relación entre el dinero generado por los videojuegos, la música grabada y el cine durante 2015 en España [7]

12 En conclusión, es cuestión de tiempo que a los videojuegos se le otorgue socialmen-
13 te el mismo estatus que al cine: ser calificados de cultura o arte, y remarco socialmente
14 porque en países como el nuestro ya están catalogados como cultura, pero ese mensaje
15 aún está por calar en muchos estratos de la sociedad.

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

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

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. Actualmente sufren un vertiginoso ascenso y muestra de ello son las cifras de espectadores, que suben año tras año. Actualmente se cifran en 385 millones las personas que siguen los deportes electrónicos (entre espectadores ocasionales y entusiastas, con una proporción similar para ambos), con el pronóstico de llegar a 589 millones de espectadores en 2020, lo que supone un incremento de más del 50% en 3 años [13].

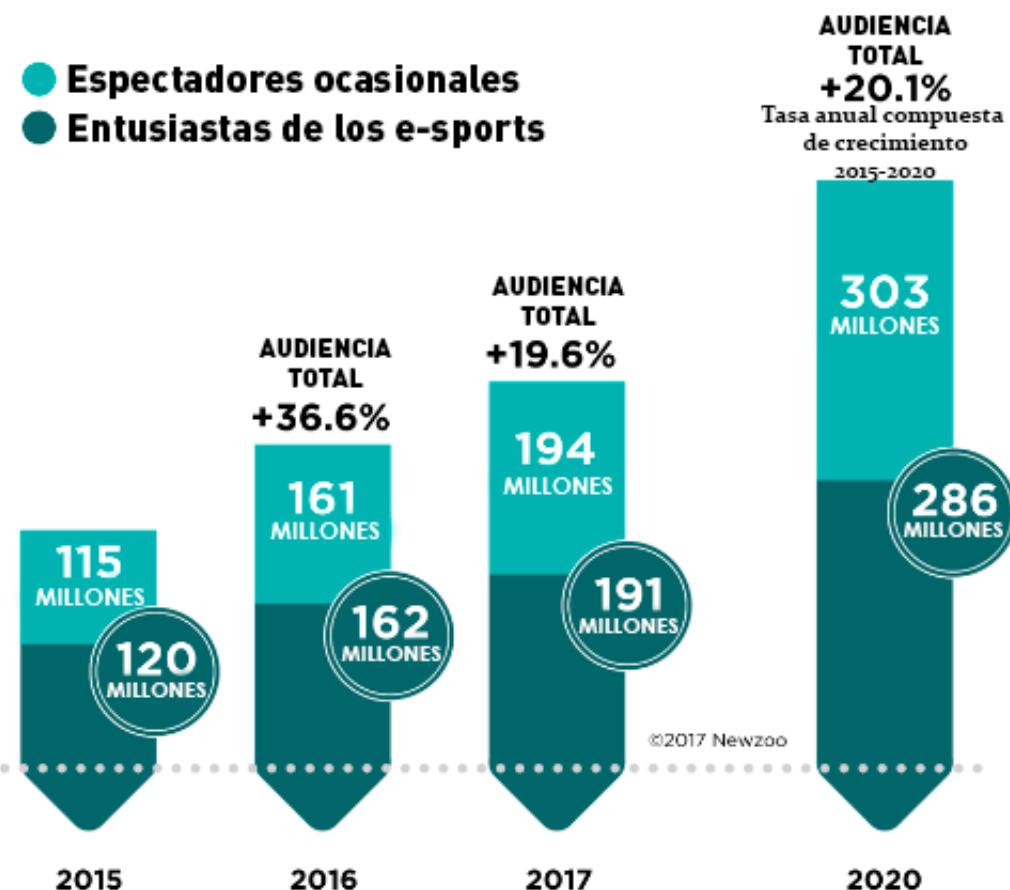


Figura 1.4: Audiencia y previsión de crecimiento de los espectadores de deportes electrónicos [13]

1 Igualmente, como se describía en la sección anterior, cadenas de ámbito nacional
 2 ya dedican algunos de sus diales a tiempo completo a los videojuegos y, realmente,
 3 casi la totalidad de estos suelen estar ligados a los e-sports, algo que actualmente es
 4 tendencia en los estados de nuestro entorno y algo normal desde hace años en algunos
 5 países concretos como Corea del Sur.

6 Aún así, la fuerza de los e-sports va más allá: los deportes electrónicos mueven
 7 millones de personas en todo el mundo, consiguen llenar estadios con sus eventos,
 8 reparten millones en premios y, quizás algo que es mucho más importante, generan
 9 un fenómeno fan que, aunque pueda impresionar si no se conoce a fondo este sector,
 10 está en vías de convertirse en uno tan poderoso como el que podría ser a día de hoy
 11 el fútbol: como se comentaba previamente los espectadores de este tipo de entreteni-
 12 miento no paran de crecer, pero no sólo eso, un amplio sector conoce los equipos, sus
 13 jugadores y, de manera similar como ocurre en los deportes tradicionales, muchos de
 14 ellos marcan en el calendario el siguiente torneo de su juego preferido o el próximo
 15 partido de su equipo favorito. El fenómeno es tal que incluso se realizan realities de
 16 cómo estos equipos de deportes electrónicos viven o entrenan.

17 Este fenómeno coincide en el tiempo con la expansión del mercado del videojuego
 18 en los terminales móviles, el cuál genera una importante cantidad de dinero para la
 19 industria y, de igual manera que pasa con los deportes electrónicos, está en alza con
 20 beneficios que superan los 36.000 millones de dólares en 2016 [12, 14], lo que supone
 21 en torno al 37% de los beneficios totales de la industria obtenidos el pasado año [14].



Figura 1.5: Comparativa, beneficios y previsiones entre los juegos para móvil y el resto de aplicaciones (datos en billones americanos) [12]

Además, los dispositivos móviles están ayudando a que la industria llegue a un número mucho mayor de personas y, lo que es mucho más importante, de una manera más indirecta: ya no es necesario que el usuario tenga el deseo específico de jugar a videojuegos y realice un desembolso para adquirir una consola u ordenador gaming, algo que sin duda puede hacer que muchos usuarios potenciales se pierdan en el proceso, sino que cualquiera que disponga de un teléfono móvil, sea cual sea su plataforma, puede gozar de un amplio catálogo de juegos y de una manera muy accesible, por lo que tarde o temprano la mayoría probará lanzar algún videojuego en su dispositivo y parte de ellos pasarán a ser usuarios habituales de la industria, con las repercusiones que ello conlleva.

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

1.1.2 El auge del videojuego independiente

Centrándonos más en el proyecto que nos ocupa, cabe destacar el fenómeno de los videojuegos indie: juegos desarrollados con poco presupuesto, por un equipo de desarrollo pequeño y que, generalmente, se atreven a arriesgarse más que la media puesto que, tradicionalmente, los juegos desarrollados por grandes empresas y que cuentan con un gran presupuesto detrás tienden a asegurar más, a ser un producto similar a lo que ya está triunfando en ese momento en el mercado. En definitiva, no 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. Además, cabe destacar que gran parte de la culpa de su irrupción la tuvo la consolidación distribución digital, que abrió puertas a muchos desarrolladores y abarató costes, ya que es más rentable para el desarrollador que negociar con una distribución tradicional.

Juegos publicados por año

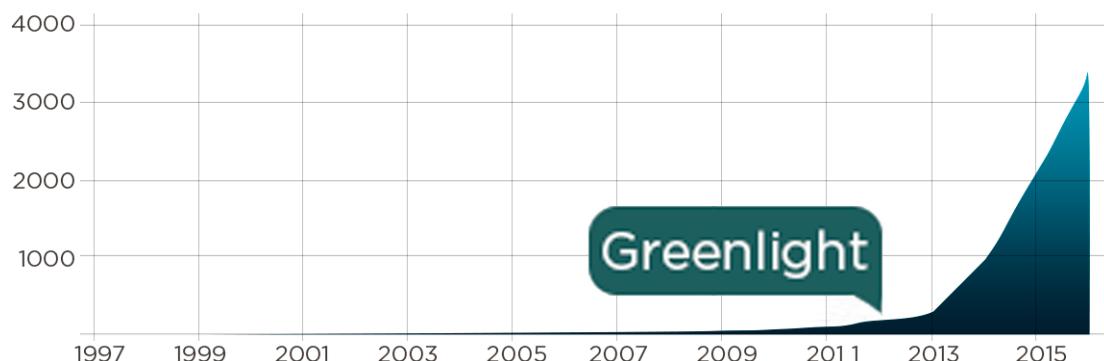


Figura 1.6: Relevancia del programa «Greenlight» en los lanzamientos indies [22]

Si bien se podría decir que nacieron en la clandestinidad, desde hace años gozan de una enorme aceptación y prueba de ello es que cada vez tienen mayor visibilidad: «Steam», la principal plataforma de distribución digital en PC, 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, pasar a 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 es un hecho que cada vez se apuesta más por ellos y se fomenta la aparición de nuevos [4, 11]. Más recientemente ha sido Nintendo quien no ha podido dejar pasar el tren de los videojuegos indies incorporándolos al catálogo de lanzamiento de Nintendo Switch.

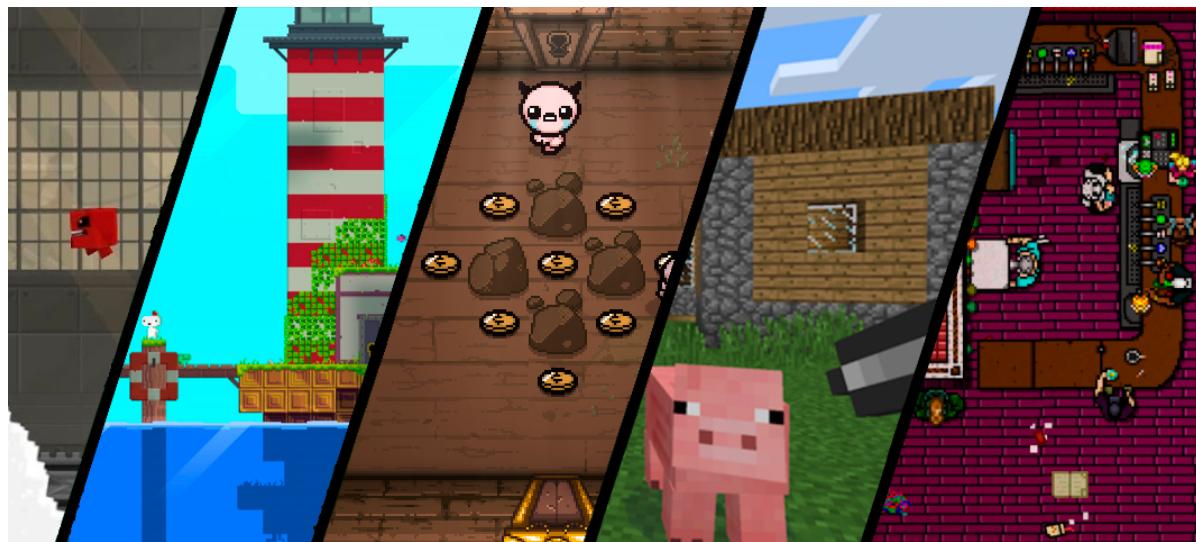


Figura 1.7: Ejemplos de videojuegos indies se hicieron un hueco en el mercado

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 [16, 17].

En la actualidad, existen un gran número de indies que han saltado a la fama, el caso más conocido es el de «Minecraft» (2011), pero existen otros muchos títulos como «Super Meat Boy» (2010), «The Binding Of Isaac» (2011 y su remasterización, «Rebirth», lanzada en 2014), «FEZ» (2012) o «Hotline Miami» (2012).

En conclusión, estamos en un momento en el que existe un público dentro de la comunidad que valora desde hace años el soplo de aire fresco que supone el contenido independiente en la industria, que permanecen atentos al panorama indie y no sólo a los lanzamientos triple A (o AAA), y en muchos casos valoran este tipo de contenido por encima de una superproducción que va dirigida a un público mucho menos específico.

1.1.3 La irrupción de los trofeos / logros y su impacto

Aunque en la década de los 90 ya aparecieron los primeros videojuegos con algún atisbo de sistema de desafíos no fue hasta hace muy poco, con la aparición de la consola «Xbox 360» y su sistema de logros (asociados a una «Gamertag»), que no se redefinieron hasta como los conocemos hoy en día: los logros son diseñados e implementados en cada juego por los desarrolladores, pero una vez que obtienen se quedan registrados en el perfil del usuario. Esto permite al jugador exhibir los logros que más le gusten, los más raros, mostrar el número total de ellos, el número de juegos con todos los logros desbloqueados, el porcentaje de obtención de logros medio, etc. En resumen, con esta renovación pasaron a tener un aspecto mucho más social.

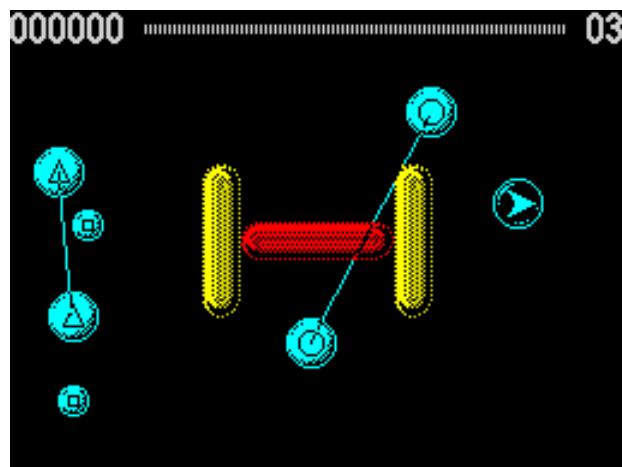


Figura 1.8: Captura del juego «E-Motion» (1990), para la videoconsola «Amiga», uno de los primeros juegos en implementar algún tipo de logro

Cabe destacar que su propósito inicial se ha ido desvirtualizando con el tiempo: un logro, como si nombre indica, originariamente era otorgado al jugador cuando este realizaba algún tipo de proeza en el juego o, por ejemplo, cuando completaba el mismo. Actualmente también se usan en muchos casos para mostrar la progresión del jugador pero por otra parte existen cada vez más títulos, conocidos como «achievement machines» (que podría traducirse como «máquinas de logros»), que recompensan excepcionalmente al jugador (por ejemplo, cada vez que realiza una acción básica como saltar, disparar, o bien múltiples logros por completar la misma acción) [9] y además permiten completar el juego en cuestión de minutos. Aunque en primera instancia pudiera parecer lo contrario, este tipo de juegos, por el mero hecho de permitirte obtener un gran número de logros en poco tiempo o logros que te permitan decorar el perfil, tiene un público y va creciendo [10].

A día de hoy existen numerosas plataformas destinadas a los «achievement hunters» o «cazadores de logros» (como pueden ser «AStats» o «AchievementStats» en «Steam»), que elaboran numerosas estadísticas con los logros y confeccionan un ranking de jugadores atendiendo a los mismos, y otras muchas comunidades con guías o tutoriales para obtener logros específicos (como podrían ser «XBoxAchievements» en «Xbox» o «PlayStationTrophies» en «PlayStation»).



Figura 1.9: Imagen del sistema de logros interno de «Minecraft» (2011), que se mantuvo hasta la versión 1.11

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.



Figura 1.10: Imagen del juego «E.T.» para la «Atari 2600», uno de los máximos detonantes de la crisis del videojuego [15]

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 ningúun 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.

Además, debemos hacer mención que en España justo después de la crisis se entró en un periodo denominado «la edad de oro del software español», que se extendió entre 1983 y 1992, en la que el país se convirtió en uno de los principales creadores de videojuegos de Europa.



Figura 1.11: Fotograma de «*La abadía del crimen*» (1987), uno de los máximos estandartes de «la edad de oro del software español»

Volviendo a nuestros días y centrándonos en lo que la industria 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 [20]. 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 4, CryEngine 3 o Unity.

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

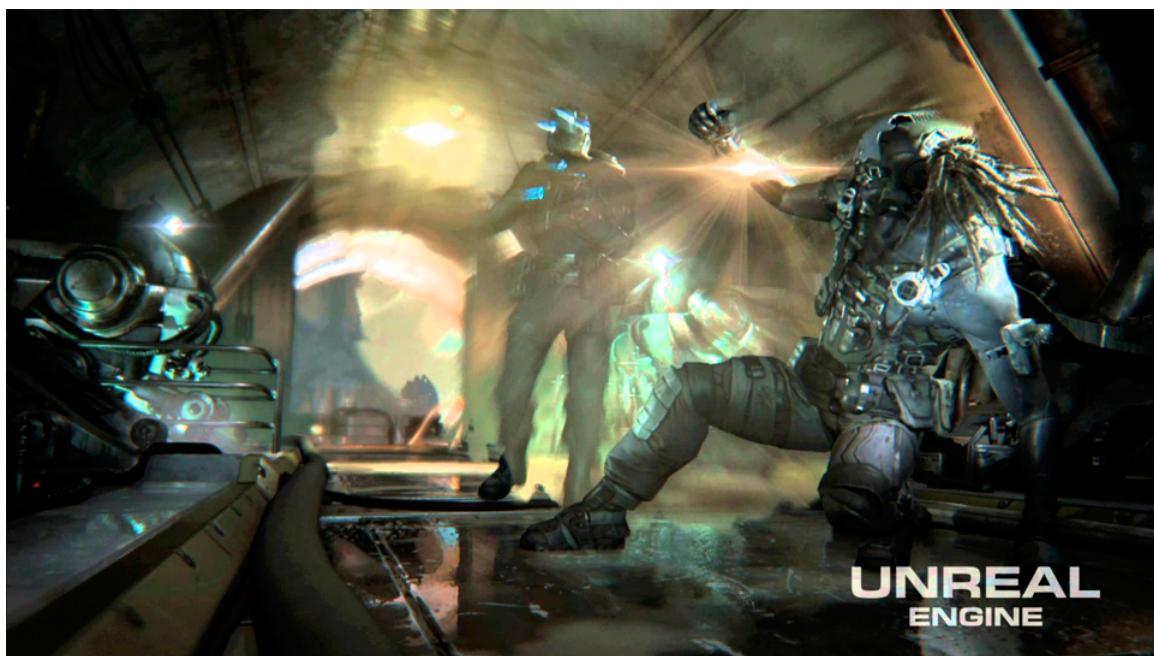


Figura 1.12: Imagen de la primera demostración técnica en tiempo real de Unreal Engine 4, titulada «Infiltrator»

Podemos concluir que la industria pasa por un buen momento, está en continuo crecimiento y no hay ningún indicio que nos haga pensar que se pueda volver a repetir una crisis como la que se originó en los ochenta, y mucho menos por la misma razón: la desinformación, premisa que es impensable que se pueda volver a dar en nuestros días. Por otro lado, los motores gráficos actuales son un avanzado conjunto de herramientas que permiten a los desarrolladores crear sus productos de una manera más sencilla y rápida que antaño, lo que repercute en los tiempos de ejecución y permite abordar cada vez proyectos más extensos o ambiciosos influyendo, por lo tanto, de manera directa en el acabado final del producto conforme van evolucionando.

OBJETIVOS DEL PROYECTO

1

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

2

3

4

2.1 MOTIVACIÓN

Dada la saturación del mercado con productos similares o, directamente, 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, que 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.



Figura 2.1: Fotograma de «Bioshock» (2007), uno de los pocos «First Person Shooter» («FPS») que se supieron diferenciar dentro de la saturación del género

El gameplay se basaría en la repetición y memorización de niveles, esquive 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.

Esquive 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 colecciónables repartidos por el mismo.

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

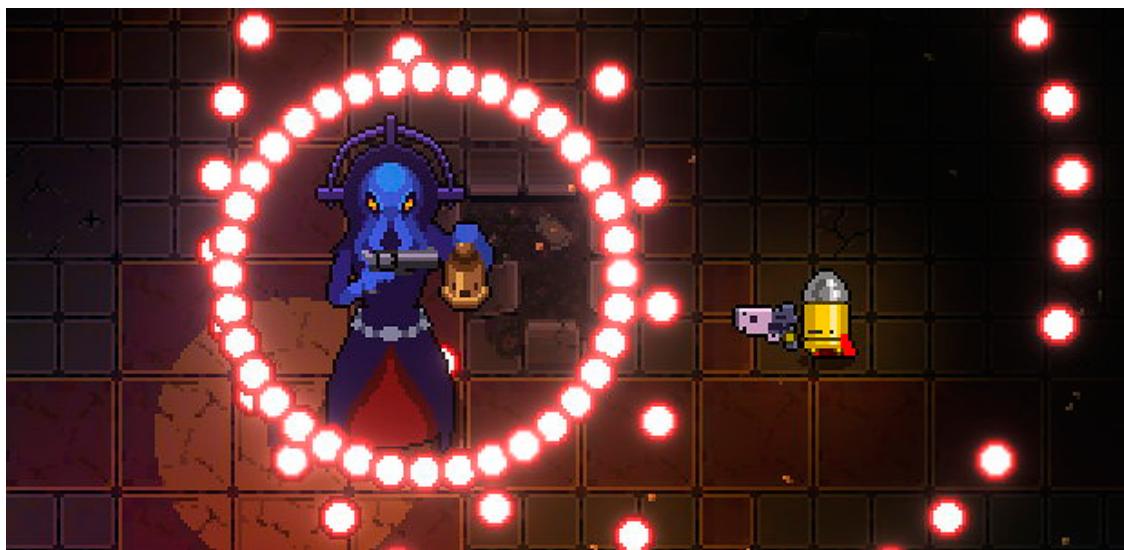


Figura 2.2: Captura de «Enter the gungeon» (2016), ejemplo de la forma más tradicional de bullet hell en dos dimensiones

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

Para una mayor claridad se ha decidido dividir esta sección en dos partes, los objetivos que se pretenden alcanzar individualmente y los objetivos que se pretende que alcance, una vez finalizado, 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.

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.	1 2
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).	3 4 5
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.	6 7 8 9
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.	10 11 12 13 14 15 16 17 18 19 20
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.	21 22 23 24 25

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.

PARTE II

ORGANIZACIÓN

DEL PROYECTO

METODOLOGÍA

1

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.

2

3

4

3.1 ESTRUCTURA ORGANIZACIONAL DEL PROYECTO

Dado que el proyecto no se realiza en grupo no se podría decir que existe una estructura organizacional interna 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.

Sin embargo, como se verá en las posteriores secciones del capítulo, se desempeñarán a la vez tres roles: «Jefe de proyecto», «Diseñador» y «Programador», ya que aunque el equipo de proyecto esté formado por una única persona se intentarán simular estos tres roles.

A raíz de lo descrito surge una consecuencia directa, y es que algunos roles o actividades de la metodología 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.

Teniendo en cuenta todo lo descrito, el organigrama resultante y que se usará en la realización del proyecto es el siguiente:



Figura 3.1: Organigrama 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), metodología 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 [6, 18].

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, 2]:

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:** Despues de una fase de diseño satisfactoria se procede a la construcción total del proyecto.



Figura 3.2: Esquema del desarrollo basado en funcionalidades

Además, la metodología FDD consta de 6 **roles clave**, los cuales se describen a continuación [2, 19]:

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 [1], 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.



Figura 3.3: Esquema del desarrollo basado en funcionalidades adaptado para el proyecto

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

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

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

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

PLANIFICACIÓN

1

En 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.

2

3

4

1 4.1 RESUMEN TEMPORAL DEL PROYECTO

Resumen del proyecto	
Fecha de inicio	20/02/2017
Fecha de fin	27/08/2017
Periodicidad de las revisiones (promedio)	2 semanas
Carga de trabajo semanal (promedio)	16,31 horas
Horas totales previstas	310 horas
Horas finales	360 horas

2 Cuadro 4.1: Tabla resumen de tiempos y planificación

2 4.2 PLANIFICACIÓN INICIAL

3 Para su planificación, se ha decidido dividir el proyecto en siguientes fases o itera-
 4 ciones, las cuales se describen a continuación junto con los roles que participan princi-
 5 palmente:

- 6 ■ **Fase de estudio:** En esta primera etapa interviene únicamente el rol **jefe de pro-**
 7 **yecto** (aunque en un contexto real intervendrían más personas, como por ejemplo
 8 el diseñador jefe), que será el encargado de estudiar las diferentes tecnologías que
 9 tiene a su disposición y de poner los cimientos más básicos de la ruta que seguirá
 10 el proyecto.
- 11 ■ **Fase de inicio del proyecto (o iteración 0):** Es la etapa en la que se realizan las
 12 siguientes tareas:
 - 13 • Las primeras fases de la metodología y buena parte de la elaboración de
 14 la memoria, salvando los apartados finales (por ejemplo, el manual o las
 15 conclusiones) y las secciones centradas en el desarrollo, principalmente.
 - 16 • El diseño general del videojuego y, por tanto, el documento de diseño del
 17 videojuego.
 - 18 • La inicialización del proyecto, preparación del repositorio, etc.

19 Teniendo en cuenta lo descrito, en esta fase intervienen el **jefe de proyecto** (do-
 20 cumentación del proyecto), el **diseñador** (realización del «Documento de diseño
 21 del juego» o «GDD») y el **programador** (inicialización del proyecto).

- **Iteraciones 1-6:** Son las etapas más puramente de desarrollo. Intervienen tanto el **diseñador**, 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), además del **jefe de proyecto** que tendrá una labor de supervisión (principalmente gestionar tiempos empleados y retrasos). A continuación, se describen muy escuetamente las iteraciones (para más información, consultar el documento de diseño §B):
 - **Iteración 1. Mecánicas básicas y poderes:** La primera iteración se implementarán las mecánicas más básicas, como saltar, agacharse, correr, etc. y, además, las mecánicas de las que podrá hacer uso el personaje cuando esté potenciado.
 - **Iteración 2. Mecánicas de escalada:** En la segunda iteración se implementarán las mecánicas de «plataformeo» avanzadas.
 - **Iteración 3. HUD y menú:** Durante la tercera iteración se creará el menú principal y el HUD del que dispondrá el usuario.
 - **Iteración 4. Peligros del entorno:** La cuarta iteración tratará sobre la implementación los peligros del entorno que deberá esquivar el personaje.
 - **Iteración 5. Inteligencia artificial y sistema de guardado:** Durante la quinta iteración se implementará la IA de los enemigos y el sistema de guardado.
 - **Iteración 6. Diseño de escenarios:** En la sexta y última iteración se llevarán a cabo la creación de los escenarios de los que dispondrá, a modo de demostración, el videojuego a entregar. Esto incluye el diseño de un tutorial.
- **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 posteriormente realice la presentación mediante la cual se expondrán los resultados del proyecto.

1 A tenor de lo descrito y de la metodología establecida en el capítulo anterior, pode-
2 mos establecer la siguiente relación:

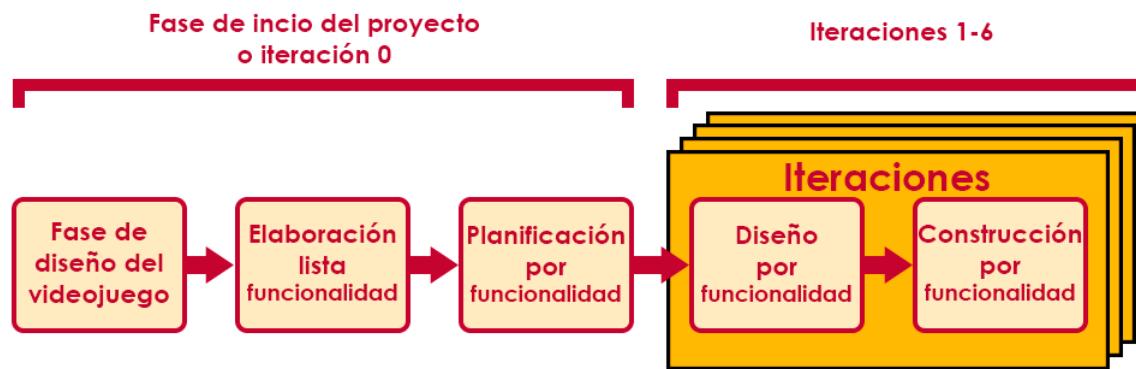


Figura 4.1: Equivalencias fases-metodología

3 Además, analizando la diferentes fases **desde el punto de vista de los roles** pode-
4 mos concluir que:

- 5 ■ **Jefe de proyecto:** Realiza la mayor parte del esfuerzo en las primeras etapas del
6 proyecto (fase de estudio y fase de inicio) y, en menor medida, al final del mismo
7 (fase de cierre). Entre dichas fases, en las iteraciones, tiene un papel mucho más
8 testimonial en el proyecto que nos ocupa.
- 9 ■ **Diseñador:** La mayor parte del esfuerzo en este caso recae en la fase de inicio del
10 proyecto, donde se tienen que sentar las bases del juego y diseñar el documento
11 de diseño del juego. Durante las iteraciones tiene un papel mucho menos relevante
12 que el programador pero más que el jefe de proyecto y, al igual que pasaba
13 con éste último, también crece su esfuerzo al final del proyecto (fase de cierre).
- 14 ■ **Programador:** Únicamente concentra su esfuerzo durante las iteraciones, durante
15 el resto del proyecto no está presente en el escenario planteado.

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

Resumen de fases e iteraciones (planificación)	
Fase de estudio	20/02/17 a 12/03/17 (3 semanas)
Fase de inicio del proyecto	13/03/17 a 02/04/17 (3 semanas)
Iteración 1	03/04/17 a 16/04/17 (2 semanas)
Iteración 2	24/04/17 a 14/05/17 (3 semanas)
Iteración 3	15/05/17 a 28/05/17 (2 semanas)
Iteración 4	03/07/17 a 09/07/17 (1 semana)
Iteración 5	10/07/17 a 23/07/17 (2 semanas)
Iteración 6	24/07/17 a 06/08/17 (2 semanas)
Fase de cierre del proyecto	07/08/17 a 13/08/17 (1 semana)
Red de seguridad	14/08/17 a 31/08/17 (2.5 semanas)

Cuadro 4.2: Planificación temporal de fases e iteraciones (planificación)

Como podemos extraer de la tabla, durante la ejecución del proyecto hay planificadas dos paradas, la primera es de sólo una semana y la segunda de un mes. Se describen a continuación:

- **Semana del 17 al 23 de Abril:** Esta semana es coincidente con los primeros parciales.
- **Mes de Junio (semanas entre el 29 de Mayo y el 02 de Julio):** Coincidente con los segundos parciales, exámenes finales y entrega y posterior defensa de proyectos de diversas asignaturas.

Además, como también figura en la tabla, se ha decidido establecer un periodo denominado «**Red de seguridad**» a modo de medida de contingencia para paliar cualquier retraso que surja durante la ejecución del proyecto. Esto es una medida recomendable en todos los proyectos pero más antes el que nos encontramos, ya que es el primer proyecto de este tipo / tecnología. En concreto tendremos un colchón de seguridad de dos semanas y media.

- ¹ A continuación se muestra una tabla detallada de las iteraciones, mostrando qué rol
- ² interviene, el número de horas de trabajo que participa y el número de horas totales de
- ³ cada iteración:

Planificación de fases e iteraciones detalladas			
Iteración	Roles implicados	Horas est. / rol	Horas estimadas
Fase de estudio	Jefe de proyecto	30 horas	30 horas
	Jefe de proyecto	30 horas	
Fase inicio proyecto	Diseñador	20 horas	60 horas
	Programador	10 horas	
Iteración 1	Jefe de proyecto	4 horas	
	Diseñador	6 horas	34 horas
	Programador	24 horas	
Iteración 2	Jefe de proyecto	6 horas	
	Diseñador	9 horas	51 horas
	Programador	36 horas	
Iteración 3	Jefe de proyecto	4 horas	
	Diseñador	6 horas	34 horas
	Programador	24 horas	
Iteración 4	Jefe de proyecto	2 horas	
	Diseñador	4 horas	18 horas
	Programador	12 horas	
Iteración 5	Jefe de proyecto	4 horas	
	Diseñador	6 horas	34 horas
	Programador	24 horas	
Iteración 6	Jefe de proyecto	4 horas	
	Diseñador	6 horas	34 horas
	Programador	24 horas	
Fase cierre proyecto	Jefe de proyecto	10 horas	
	Diseñador	5 horas	15 horas
TOTAL			310 horas

Cuadro 4.3: Planificación temporal detallada de fases e iteraciones

Recordemos que es un proyecto realizado con una tecnología, que genera un producto nuevo y del que, por tanto, no se tienen referencias previas por lo que estimar las horas es especialmente difícil. Respecto a esto no hay mucho que podamos hacer, tan solo se han podido calcular las estimaciones sobre las horas dedicadas de cada rol durante las **fases de iteración** en base a las siguientes normas, para conseguir que sean homogéneas:

Calculo de horas por rol en iteraciones	
Rol	Horas estimadas
Jefe de proyecto	2 horas por semana
Diseñador	3 horas por semana
Programador	12 horas por semana

Cuadro 4.4: Tabla del calculo de horas por rol en iteraciones

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

Resumen de horas de trabajo por rol	
Jefe de proyecto	94 horas
Diseñador	62 horas
Programador	154 horas

Cuadro 4.5: Tabla resumen de horas de trabajo por rol

4.3 INFORME DE TIEMPOS DEL PROYECTO

Durante el desarrollo del proyecto se han obtenido los siguientes resultados respecto a tiempo:

Tiempos empleados en fases e iteraciones detallados			
Iteración	Roles implicados	Horas reales / rol	Horas reales
Fase de estudio	Jefe de proyecto	25 horas	25 horas
	Jefe de proyecto	58 horas	
Fase inicio proyecto	Diseñador	16 horas	80 horas
	Programador	6 horas	
	Jefe de proyecto	3 horas	
Iteración 1	Diseñador	6 horas	27 horas
	Programador	18 horas	
	Jefe de proyecto	5 horas	
Iteración 2	Diseñador	9 horas	59 horas
	Programador	45 horas	
	Jefe de proyecto	TODO horas	
Iteración 3	Diseñador	TODO horas	TODO horas
	Programador	TODO horas	
	Jefe de proyecto	TODO horas	
Iteración 4	Diseñador	TODO horas	TODO horas
	Programador	TODO horas	
	Jefe de proyecto	TODO horas	
Iteración 5	Diseñador	TODO horas	TODO horas
	Programador	TODO horas	
	Jefe de proyecto	TODO horas	
Iteración 6	Diseñador	TODO horas	TODO horas
	Programador	TODO horas	
	Jefe de proyecto	TODO horas	
Fase cierre proyecto	Diseñador	TODO horas	TODO horas
	Jefe de proyecto	TODO horas	
TOTAL			TODO horas

Cuadro 4.6: Tabla de tiempos empleados en fases e iteraciones detallados

4.3. INFORME DE TIEMPOS DEL PROYECTO

La tabla que se muestra a continuación muestra la **desviación de tiempo** por iteración que se ha obtenido (en la cuál el símbolo «+» representará una desviación de tiempo negativa, es decir, que la fase ha llevado más tiempo del planeado y por contrapartida el símbolo «-» representará que se ha realizado en menos tiempo del previsto):

Desviación de tiempo por iteración		
Fase	Desviación de la iteración	Desviación acumulada
Fase de estudio	-5 horas	-5 horas
Fase de inicio del proyecto	+20 horas	+15 horas
Iteración 1	-7 horas	+8 horas
Iteración 2	+8 horas	+16 horas
Iteración 3	TODO horas	TODO horas
Iteración 4	TODO horas	TODO horas
Iteración 5	TODO horas	TODO horas
Iteración 6	TODO horas	TODO horas
Fase de cierre del proyecto	TODO horas	TODO horas
Desviación total		TODO horas

Cuadro 4.7: Tabla de desviación de tiempo por iteración

La siguiente tabla muestra, por períodos, el **retraso acumulado** del proyecto:

Retrasos del proyecto	
Periodo	Retraso acumulado
20/02/17 a 20/05/17	Periodo sin retrasos
21/05/17 a Actualidad	1 semana de retraso

Cuadro 4.8: Tabla de retrasos del proyecto

Justificación de retrasos:

- **Retraso del 21/05/15 a TODO:** .

TODO: Conclusiones, por ejemplo, el rol de jefe de proyecto durante las iteraciones trabajó más horas de las proyectadas, etc.

COSTES

1

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

2

3

4

5

6

En este capítulo abordaremos una de las partes más importantes en la planificación de un proyecto: los costes, y para ello los dividiremos en costes de personal, de material e indirectos.

7

8

9

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

1 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.

8 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 11 §3.2.3).

12 A partir de la tabla anterior, debemos calcular los honorarios por hora de cada uno 13 de los roles. Para realizar esto debemos dividir el salario anual entre la jornada anual:

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

14 La jornada máxima anual del sector TIC está fijada en 1.800 horas, por tanto, una 15 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

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

18 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 19 20 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:

1

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

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**.

2

TODO: Impuestos y costes sociales.

3

5.3 COSTES MATERIALES

4

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.

5

6

7

8

La amortización se realizaría de la siguiente manera:

9

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

Para la ejecución del proyecto se cuenta con:

10

- 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.

11

12

13

14

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

10 Para la realización del proyecto se dispone de dos dispositivos, un ordenador so-

11 bremesa y un ordenador portátil (a partir de ahora sobremesa y portátil) en el que se

12 realizarán las labores de producción, preproducción y pruebas.

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

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

Una vez realizado esto, sólo faltaría multiplicar el coste material al mes por la duración del proyecto en meses, de esta forma:

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

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

5.4 COSTES INDIRECTOS

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

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

1 Existen gran variedad de costes indirectos, como podrían ser licencias, seguros,
 2 bancos, contratación de servicios profesionales (como podrían ser abogados para po-
 3 sibles temas legales o gestoras para, por ejemplo, llevar a cabo la contabilidad o las
 4 declaraciones a hacienda), alquileres, etc.

5 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

6 [*]: En cuanto a licencias debemos tener en presente que Unreal Engine es un motor
 7 de uso gratuito pero, sin embargo, la compañía propietaria del motor, en este caso Epic
 8 Games, establece en su EULA una cláusula por la que habrá que abonar cierta cantidad,
 9 un loyalty, si se superan los 3.000\$ de beneficio por cuatrimestre. Esta cantidad en gira
 10 en torno a un 5% de los beneficios que pasen de 3.000\$: por ejemplo, si se generan
 11 5000\$ de beneficio habrá que remitirle a Epic Games aproximadamente 100\$ (el 5% de
 12 2000\$, resultado de la resta de 5.000\$ - 3.000\$).

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

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

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

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

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

1
2

PARTE III

DESARROLLO DEL PROYECTO

ARRANQUE

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

Durante este capítulo de la memoria se presentará la lista de características que conforman el proyecto, así como el diseño arquitectónico de Unreal Engine, el motor 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 §B), 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.

6.1. LISTA DE CARACTERÍSTICAS

■ Asignación de teclas y botones vinculados la mecánica.	1
■ Desarrollo de la animación del personaje relacionada con la mecánica.	2
■ Gestión de adrenalina relacionada con la mecánica.	3
7. Mecánica: Parpadeo (o «Blink»).	4
■ Implementación del código de la mecánica.	5
■ Asignación de teclas y botones vinculados la mecánica.	6
■ Desarrollo de la animación del personaje relacionada con la mecánica.	7
■ Gestión de adrenalina relacionada con la mecánica.	8
■ Implementación del código del indicador de blink disponible.	9
8. .	10
9. .	11
10. .	12
11. .	13
12. .	14
13. .	15
14. .	16
15. .	17
16. .	18
17. .	19
18. .	20
19. .	21
20. .	22
21. .	23
22. .	24
23. .	25

CAPÍTULO 6. ARRANQUE

1 24. .

2 25. .

3 26. .

4 27. .

5 28. .

6 29. .

7 30. .

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 [8].

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

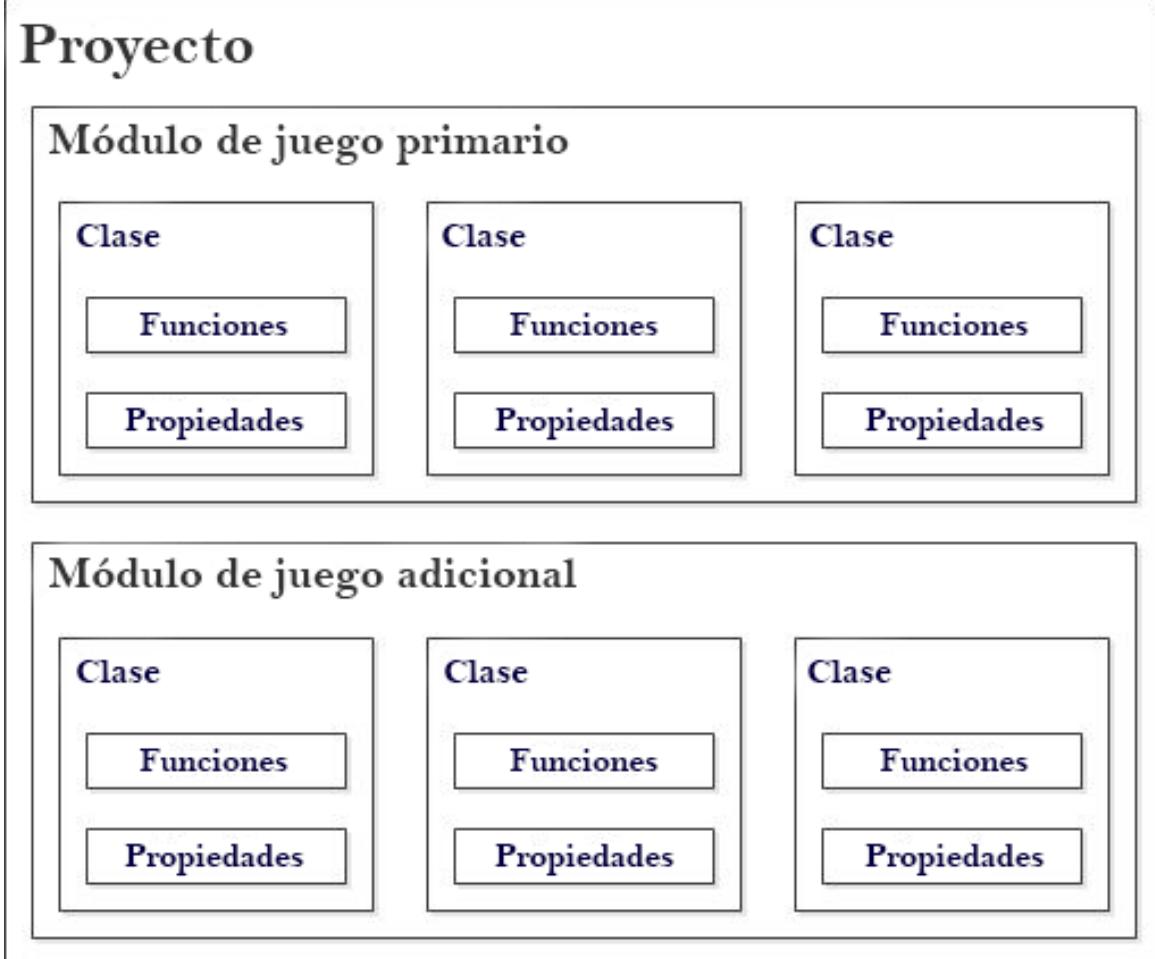


Figura 6.1: Arquitectura Unreal Engine 4

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

¹ también pueden ser definidas en su propia interfaz, permitiendo que puedan ser im-
² plementadas en diferentes clases.

³ Cuando programamos en Unreal Engine, es posible tener clases, funciones y varia-
⁴ bles estándar del lenguaje C++, usando la sintaxis estándar de C++. Sin embargo, tam-
⁵ bién podemos usar las instrucciones «UCLASS()», «UFUNCTION()» y «UPROPERTY()»
⁶ para que Unreal Engine conozca nuevas clases, funciones y propiedades, respectiva-
⁷ mente. Por ejemplo, una variable cuya declaración está precedida por «UPROPERTY()»
⁸ puede ser reconocida por el motor y ser mostrada y editada en el editor de Unreal En-
⁹ gine. También existen «UINTERFACE()» y «USTRUCT()» y palabras claves para cada
¹⁰ una que pueden ser usadas para especificar el comportamiento de una clase, función,
¹¹ propiedad, interfaz o estructura en Unreal Engine.

¹² 6.2.1 Módulos de juego

¹³ De la misma forma en la que el propio motor está compuesto por una colección
¹⁴ de módulos, cada juego está compuesto por uno o más módulos de juego. Estos son
¹⁵ similares a los paquetes de las versiones anteriores del motor, en las cuales estos eran
¹⁶ contenedores de una colección de clases relacionadas. En Unreal Engine 4, ya que está
¹⁷ basado en C++, **los módulos son en realidad DLLs**, en lugar de paquetes.

¹⁸ Los módulos de juego deben contener, como mínimo, un archivo de cabecera («.h»),
¹⁹ un archivo C++ («.cpp») y un archivo de compilación («*.Build.cs»).

²⁰ El archivo de cabecera debe estar localizado en la carpeta «Public» de la localización
²¹ del módulo, por ejemplo: «[NombreDelJuego]/Source[NombreDelMódulo]/Public».
²² Este archivo contiene cualquier archivo de cabecera, incluyendo las cabeceras auto-
²³ generadas del módulo, necesario para compilar las clases que incluye el módulo.

²⁴ El archivo «C++», localizado en la carpeta «Privada» de la localización del módulo
²⁵ («[NombreDelJuego]/Source[NombreDelMódulo]/Private»), implementa el módulo.

²⁶ El archivo de compilación («build») es está localizado en la raíz del directorio del
²⁷ módulo («[NombreDelJuego]/Source[NombreDelMódulo]») y contiene información usa-
²⁸ da por la herramienta de compilación de Unreal Engine («UnrealBuildTool») para com-
²⁹ pilar el módulo.

Módulos de juego múltiples

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

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

Limitaciones

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

6.2.2 Plugins

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

Anatomía

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

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

5 Los plugins pueden tener su propia carpeta de contenido («Content») que con-
 6 tendrá assets específicos para ese plugin.

7 6.2.3 Clases de gameplay

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

13 Las clases en Unreal Engine tienen una estructura de nombres estandarizadas, por
 14 la que se podrá conocer instantáneamente qué tipo de clase es simplemente mirando a
 15 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».

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

16 Clases de cabecera

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

en la versión actual del motor).

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

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

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

Declaración de clases

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

```
UCLASS([ especificador , especificador , ... ] ,
        [meta( clave=valor , clave=valor , ... )])
class NombreDeLaClase : public NombreDeLaClasePadre
{
    GENERATED_BODY()
}
```

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

Especificadores de clases

Cuando declaramos una clase podemos añadir especificadores a la declaración para controlar cómo es el comportamiento de la clase en varios aspectos del motor y del 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

1 Implementación de clases

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

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

9 Constructores de clases

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

ITERACIÓN CERO

1

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.

2

3

Ernest Hemingway (1899–1961),

4

Novelist

5

6

T ODO TODO TODO

7

¹ 7.1 INICIALIZACIÓN DEL PROYECTO

² 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, ⁴ <https://www.unrealengine.com/>.

⁵ Con esto habremos obtenido el «launcher» de Epic Games / Unreal Engine por ⁶ lo que una vez descargado e instalado, queda un paso más: descargar una versión ⁷ concreta del motor, en nuestro caso elegiremos la versión «4.16».

⁸ Una vez descargada e instalada la versión deseada del motor, procedemos con el ⁹ siguiente paso: para la confección del trabajo vamos a crear un nuevo proyecto de ¹⁰ Unreal Engine en «C++» y sin contenido adicional, es decir, vamos a crear un proyec-¹¹ to limpio en «C++». Para ello, en el Launcher de Unreal Engine, hacemos click en el ¹² botón «Iniciar» y se nos abrirá una ventana de selección de proyecto. Como queremos ¹³ crear uno nuevo, hacemos click en la pestaña «New Project» o «Nuevo Proyecto» y una ¹⁴ vez en esa pestaña hacemos click en la pestaña «C++», seleccionamos «Basic Code» o ¹⁵ «Código básico» y configuramos el proyecto de la siguiente manera:

¹⁶ ■ «Desktop / Console» (o «Escritorio / Consola»): Puesto que nuestro proyecto ¹⁷ está destinado a dispositivos sobremesa, y no a móviles o tablets, es la opción ¹⁸ que debemos marcar cuando creamos nuestro proyecto.

¹⁹ ■ «Maximum quality» (o «Máxima calidad»): La calidad, como se detalla en el do-²⁰ cumento de diseño anexo a esta memoria, no es una prioridad para el proyecto, ²¹ pero intentaremos que el resultado final luzca lo máximo posible.

²² ■ «No starter content» (sin contenido inicial): Como se detallaba previamente, va-²³ mos a crear un proyecto en «C++» con el mínimo contenido, para conocer nues-²⁴ tro proyecto el máximo posible, y si en algún momento necesitamos material del ²⁵ contenido inicial lo añadiremos manualmente.

²⁶ Una vez configurado, hacemos click en «Create Project» (o «Crear Proyecto»).

²⁷ Con esto, habremos terminado la creación de nuestro proyecto y tendremos casi ²⁸ lista la base sobre la que empezar a trabajar.

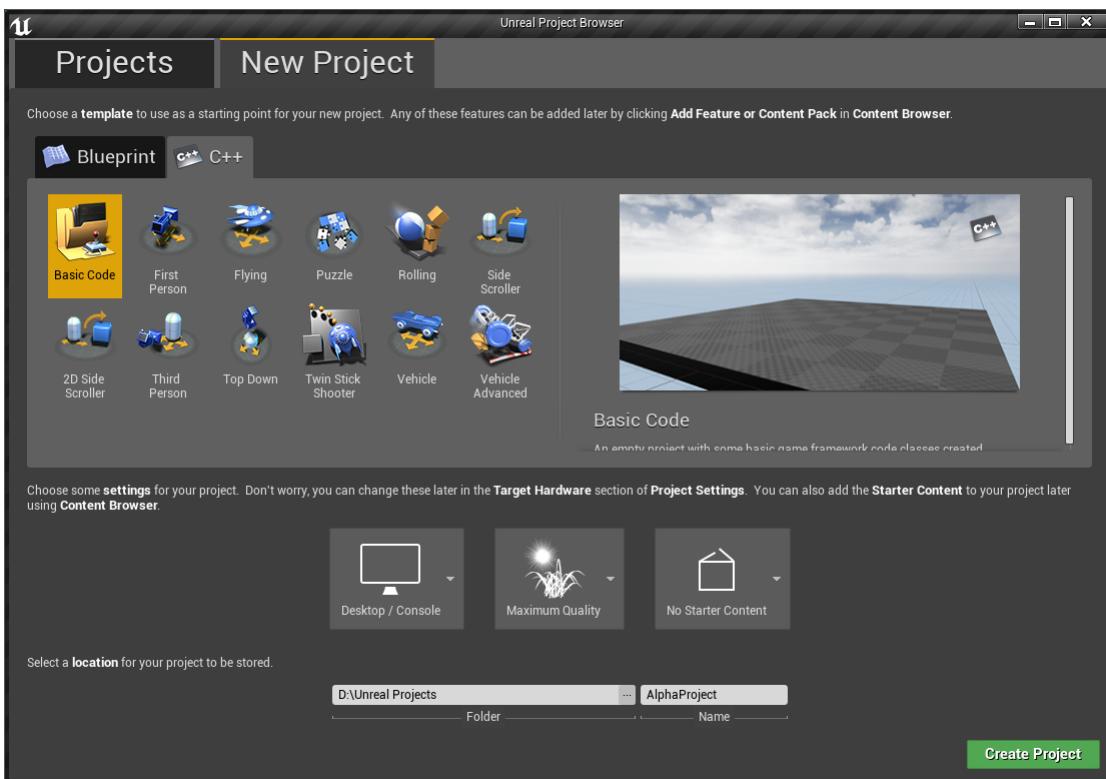


Figura 7.1: Ventana del creador de proyectos de Unreal Engine

7.2 PUESTA A PUNTO Y PRIMEROS PASOS

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

TODO.

7.3 DIRETRICES 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»

1 Mixamoquoting

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

6 Las animaciones son de máxima calidad, tanto es así que se usan en juegos con alto
7 presupuesto o ingresos: el juego más vendido del momento en PC, «Playerunknown's
8 Battlegrounds» [23], utiliza varias de las animaciones de «Mixamo» y las sigue incor-
9 porando junto con nuevas funcionalidades en sus actualizaciones mensuales.

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

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

18 1. Se

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

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

22 **7.3.2 Implementación del sistema de animación del personaje**

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

24 TODO.

25 **7.3.3 Control de que el personaje no abandone el mapa**

26 TODO.

PRIMERA ITERACIÓN

1

*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.*

2

3

Ernest Hemingway (1899–1961),

4

Novelist

5

6

T ODO TODO TODO

7

8.1 CARACTERÍSTICAS A DESARROLLAR

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: Rotación del personaje vinculada a un rate.
 - Implementación del código de la mecánica.
 - Asignación de teclas y botones vinculados la mecánica.
4. 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.
5. 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.
6. 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.
7. Mecánica: Tiempo bala (o «Time dilation»).

8.1. CARACTERÍSTICAS A DESARROLLAR

■ Implementación del código de la mecánica.	1
■ Asignación de teclas y botones vinculados la mecánica.	2
■ Desarrollo de la animación del personaje relacionada con la mecánica.	3
■ Gestión de adrenalina relacionada con la mecánica.	4
8. Mecánica: Parpadeo (o «Blink»).	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
■ Gestión de adrenalina relacionada con la mecánica.	9
■ Implementación del código del indicador de blink disponible.	10
Además, durante esta iteración se realiza la inicialización y preparación del proyecto.	11 12

¹ 8.2 DISEÑO

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*

² **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 ³ características a implementar no originan conflictos entre sí, no existen modificaciones ⁴ que afecten al diseño. 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 para que 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 derealismo, 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 agarra do 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 ($\text{input} = 0$), B) moverlo al máximo ($\text{input} = 1$) 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 ($\text{input} = 0$), B) moverlo al máximo ($\text{input} = 1$) 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

Identificador	Descripción de la acción de alto nivel					
TODO	Añade movimiento en la dirección del personaje					
Métodos de alto nivel						
[void] MoveForward (Value:float)						
Pasos						
1.	Cuando se hace uso del enlace asignado al movimiento frontal / trasero del personaje (llamado «MoveForward»), comprueba que el valor de 'Value' sea distinto de 0.					
2.	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. IU Técn.			
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						
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.					
4	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 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. IU Técn.			
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					
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			
			Técn.				
1	APawn	[void] AddControllerYawInput (Value:float)	TODO	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					
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.	IU			
			Técn.				
1	APawn	[void] AddControllerPitchInput (Value:float)	TODO	TODO			

CAPÍTULO 8. PRIMERA ITERACIÓN

Identificador	Descripción de la acción de alto nivel
TODO	Añade rotación horizontal a la cámara
Métodos de alto nivel	
[void] TurnAtRate (Rate:float)	
Pasos	
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
			Técn.	
1	APawn	[void] AddControllerYawInput (Value:float)	TODO	TODO
1	UWorld	[Float] GetDeltaSeconds ()	TODO	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	
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
			Técn.	
1	APawn	[void] AddControllerPitchInput (Value:float)	TODO	TODO
1	UWorld	[Float] GetDeltaSeconds ()	TODO	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			
	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.			
	2			
Identificador	Descripción de la acción de alto nivel			
TODO	Interactúa con el entorno / mensajes			
	Métodos de alto nivel			
	[void] OnAction ()			
	Pasos			
	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.	IU
1	UGameplayStatics	SetGamePaused (WorldContextObject: UObject, bPaused:bool)	TODO	TODO
			4	

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
	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»).
	8

CAPÍTULO 8. PRIMERA ITERACIÓN

Identificador	Descripción de la acción de alto nivel
TODO	El personaje entra en modo caminar
	Métodos de alto nivel
	[void] OnSprintReleased ()
1	Pasos
	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»).
2	

Identificador	Descripción de la acción de alto nivel			
TODO	El personaje salta			
	Métodos de alto nivel			
	[void] OnJump ()			
	Pasos			
3	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.	IU
4	ACharacter	[void] Jump ()	TODO	TODO

5 **IMPORTANTE:** Se deja preparada esta mecánica para el futuro, ya que se verá
6 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			
7	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.	IU
8	ACharacter	[void] StopJumping ()	TODO	TODO

8.4 PRUEBAS

1

TODO: Descripción de las pruebas realizadas al software

2

¹ 8.5 DESPLIEGUE

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

SEGUNDA ITERACIÓN

1

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.

2

3

Ernest Hemingway (1899–1961),

4

Novelist

5

6

T ODO TODO TODO

7

¹ **9.1 CARACTERÍSTICAS A DESARROLLAR**

² **9.2 DISEÑO**

³ **9.3 IMPLEMENTACIÓN**

⁴ **9.4 PRUEBAS**

⁵ **9.5 DESPLIEGUE**

TERCERA ITERACIÓN

1

*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.*

2

3

Ernest Hemingway (1899–1961),

4

Novelist

5

6

T ODO TODO TODO

7

¹ **10.1 CARACTERÍSTICAS A DESARROLLAR**

² **10.2 DISEÑO**

³ **10.3 IMPLEMENTACIÓN**

⁴ **10.4 PRUEBAS**

⁵ **10.5 DESPLIEGUE**

PARTE IV

CIERRE DEL PROYECTO

MANUAL DE USUARIO

1

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.

2

3

Ernest Hemingway (1899–1961),

4

Novelist

5

6

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

7

8

11.1 SECCIÓN LIBRE

2 Estructurar en función del proyecto.

CONCLUSIONES

1

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

2

3

4

5

6

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

7

8

1 12.1 INFORME POST-MORTEM

2 Qué es un informe post-mortem

3 12.1.1 Lo que ha ido bien

4 ■ Argumento a favor 1.

5 ■ Argumento a favor 2.

6 ■ Argumento a favor 3.

7 12.1.2 Lo que ha ido mal

8 ■ Argumento en contra 1.

9 ■ Argumento en contra 2.

10 ■ Argumento en contra 3.

11 12.1.3 Discusión

12 En función de lo anterior, qué cambiaría si empezara hoy el proyecto de nuevo.

13 12.2 TRABAJOS FUTUROS

14 Enumera los puntos abiertos y que no se han resuelto. Indica si darían lugar a otro

15 proyecto y de qué forma se podría acotar.

PARTE V

APPENDICES

GLOSARIO

Asset

TODO.

Bullet hell

Traducido al español como “infarto de balas”, bullet hell es un subgénero de videojuegos que se caracteriza por llenar el entorno (“la pantalla”) de peligros (tradicionalmente balas, de ahí su nombre) que matan, dañan o provocan algún tipo de alteración al personaje.

Distribuidora

En el mundo de los videojuegos es una empresa que distribuye juegos, ya sean propios o de terceros, y que normalmente se encarga también de aspectos propios de una editorial, como de llevar a cabo la promoción de la obra.

Escenario

Escena o escenario es la manera en la que se suele denominar un mapa o escenario de un videojuego (en inglés, se suele denominar “scenario”).

Frame

Los frames, o fotogramas en nuestro idioma, son típicamente cada una de las imágenes en las que se divide un vídeo. Cuando hablamos de frames en animaciones nos referimos a un instante de la animación (la posición, rotación y escala de cada uno de los huesos que forman el esqueleto).

Gameplay	Forma en la que se juega a un videojuego, incluyendo las reglas, los objetivos y la forma de alcanzarlos.
Gaming	Hace referencia a que el dispositivo está diseñado o enfocado para el uso relacionado con videojuegos.
Independiente / indie	Generalmente se denomina videojuego independiente o indie a aquel realizado por un número reducido de personas y/o con poco presupuesto, sin el respaldo de una gran compañía.
Jugabilidad	Experiencia del jugador durante la interacción con las distintas mecánicas, controles y situaciones del videojuego.
Landscape	En un videojuego, un landscape es un plano, generalmente con diferentes alturas, que se utiliza principalmente para la recreación de un paisaje natural pudiendo contener montañas, valles, etc. Principalmente se usa como base para un escenario y en él se emplazarán árboles, edificios y modelados de todo tipo.
Mecánica	Son cada una de las reglas del universo del videojuego y conjunto de acciones que puede realizar el jugador. Por ejemplo: Una mecánica sería que el personaje muriese al caer desde una determinada altura o bien que el personaje, simplemente, pueda correr o saltar.

Metodología SCRUM	Metodología ágil orientada a equipos autogestionados que, muy resumidamente, se caracteriza por sus reuniones diarias de corta duración (15 minutos como máximo) para hablar de lo que se ha hecho, lo que se va a hacer y los problemas que se han encontrado y basada en sprints: la realización de un conjunto de funcionalidades (Sprint Backlog), extraídas del Product Backlog (el conjunto de funcionalidades totales).
Metodologías ágiles	Las metodologías ágiles surgieron como contraposición a las metodologías tradicionales. Son aquellas que se basan en el desarrollo iterativo y donde los requisitos y soluciones evolucionan a lo largo del tiempo según lo necesite el proyecto.
Modelado	Un modelado o malla (del inglés mesh) es la representación mediante polígonos de un determinado objeto, que normalmente tiene asociado un material o textura, pudiendo también tener asociado varios.
Motor gráfico	Es un framework diseñado para desarrollar videojuegos que le ofrece al programador al menos las características básicas para llevar a cabo esa tarea, tales como podrían ser un motor de renderizado para visualizar gráficos en pantalla, detección de colisiones entre objetos, creación de animaciones, etc.
Plataformas	Género de videojuegos en el que el personaje tendrá que hacer uso de mecánicas básicas como salto, sprint o escalada para ir sorteando obstáculos presentes en el mapa y superar el nivel.

Renderizar	Proceso de generar un espacio que normalmente consta de tres dimensiones (3D), pero también se utiliza para generar espacios de dos (2D), representando en el proceso objetos, materiales, luces, etc.
Retargeting	Técnica usada cuando se trabaja con animaciones para adaptar las animaciones creadas sobre un esqueleto a otro. Consiste en encontrar similitudes entre los huesos de uno y otro esqueleto. Además, para que la técnica funcione ambos esqueletos deben estar en la misma pose de inicio (si no lo están, se deberá modificar la pose del esqueleto destino).
Timing	Realizar una acción específica en el momento preciso. Normalmente está sujeto a la repetición de niveles, es decir, si el jugador realiza las mismas acciones en el mismo momento pasará el nivel o una parte del mismo (siempre que no haya elementos aleatorios involucrados).
Triple A (AAA)	Se denominan así a los videojuegos lanzados por grandes empresas, con un gran presupuesto detrás tanto para su producción como para su promoción. El motivo de esto es que las tres aes (AAA), tradicionalmente, significaban un diez en gráficos, un diez en sonido y un diez en jugabilidad, pero el término ha ido evolucionando hasta llegar al descrito anteriormente, sin tener que ser necesariamente un juego de diez en ningún apartado.

DOCUMENTO DE DISEÑO

Un 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.

B.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).

B.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.

B.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.

B.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.

B.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.

B.2 COMIENZO

B.2.1 Menú principal

B.2.2 Comienzo del juego e introducción

B.2.3 In-game HUDs y menús

B.3 INTERFAZ DE USUARIO

B.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.

B.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.

B.3.3 In-game Options

TODO.

B.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.

B.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.

B.4 GAMEPLAY

B.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.

B.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.

B.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): Comutador de tiempo bala.
- **Botón derecho** del ratón: Blink.

B.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**: Comutador de tiempo bala.
- Disparador **LT**: Blink.

B.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.

B.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.

B.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 B.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 B.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 jugador obtendrá un logro en los siguientes casos:

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 B.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 B.4: *Logro historia - Capítulo II*

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 B.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 B.6: Logro prueba de tiempo - Capítulo II

B.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 colecciónables - 1 colecciónable		
		
Logo		
Obtención		Otorgado al jugador por encontrar un colecciónable (cuálquiera de ellos)
Idioma	Nombre	Descripción
Español	Colecciónables 1	Encuentra tu primer colecciónable
Inglés	Collectibles 1	Find your first collectible

Cuadro B.7: Logro colecciónables - 1 colecciónable

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

Cuadro B.8: Logro colecciónables - 5 colecciónables

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

Cuadro B.9: *Logro colecciónables - 10 colecciónables*

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

Cuadro B.10: *Logro colecciónables - 50 % de colecciónables*

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.

B.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 colecciónables - 100 % de colecciónables		
Logo		
Obtención	Otorgado al jugador por encontrar todos los colecciónables	
Idioma	Nombre	Descripción
Español	Colecciónables 100 %	Encuentra todos los colecciónables
Inglés	Collectibles 100 %	Find all the collectibles

Cuadro B.11: Logro colecciónables - 100 % de colecciónables

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 B.12: Logro misceláneos - 100 saltos

B.5.2 Enemigos

TODO

B.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 su descarga desde el Marketplace de Unreal Engine), animaciones de la plantilla por

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 B.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 B.14: Logro misceláneos - 10000 saltos

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 tanto ha sido necesario realizar un **retargeting** en cada una de ellas para adaptarlo al

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 B.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 B.16: Assets de personajes

esqueleto que del personaje del que se hace uso (que sí usa un esqueleto compatible con el esqueleto por defecto de Unreal Engine 4).

B.5.4 Equipo y mejoras

TODO: Poderes

B.5.5 Entorno

TODO

B.5.6 Audio

TODO

BIBLIOGRAFÍA

- [1] A comparison of fdd and scrum, 2011. (pages 27, 29).
- [2] Metodología fdd - feature driven development / desarrollo basado en funciones, 2012. (pages 27, 28).
- [3] AEVI. El videojuego en el mundo. Technical report, Asociación española del videojuego (AEVI), 2017. (page 4).
- [4] Buhomag. Xbox apuesta fuerte por los videojuegos indies españoles, 2015. (page 10).
- [5] U. E. Developer. Fortnite developer fireside on unreal engine 4, 2014. (page 16).
- [6] DZone. An introduction to feature-driven development, 2009. (page 27).
- [7] E. Economista. Los videojuegos facturaron más de 1.000 millones, el doble que la industria del cine, 2016. (pages xi, 6).
- [8] EpicGames. *Documentación Unreal Engine*. Epic Games, 2017. (page 57).
- [9] P. Gamer. A game with over 4,000 achievements has just launched on steam, 2017. (page 12).
- [10] T. P. O. V. Games. Why do achievements, trophies, and badges work?, 2016. (page 12).
- [11] E. Mundo. Playstation premia el talento de los videojuegos españoles, 2016. (page 10).
- [12] Newzoo. 2017 global mobile market report. Technical report, Newzoo, 2017. (pages xi, 8).
- [13] 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, 7).

- [14] Newzoo. Global games market 2016 report. Technical report, Newzoo, 2017. (pages XI, 4, 8).
- [15] E. Press. La increíble historia de et, el videojuego que hundió a atari antes de ser enterrado, 2017. (pages xi, 14).
- [16] SoftZone. Qué es steam direct y en qué se diferencia de steam greenlight, 2017. (page 11).
- [17] Steam. Tasa de steam direct y próximas actualizaciones de la tienda, 2017. (page 11).
- [18] Step-10. Fdd: History, 2014. (page 27).
- [19] Step-10. Fdd: People, 2014. (page 28).
- [20] U-Tad. La industria del videojuego en españa continúa con su imparable crecimiento, 2015. (pages 6, 15).
- [21] Xataka. Las chicas también juegan a videojuegos, y cada vez más, 2014. (page 9).
- [22] ZehnGames. El desarrollo español en steam: Una lectura desde los datos. Technical report, Zehn Games, 2017. (pages xi, 10).
- [23] ZonaRed. Playerunknown's battlegrounds continúa en el trono de las ventas semanales de steam, 2017. (page 66).