2014

Conceptualización y desarrollo de un videojuego RPG para PC

TREBALL FI DE GRAU

AUTOR: ENRIQUE JAVIER LOPEZ VALLEJO

DIRECTOR: Oriol Boira Ricart

Grado en multimedia

Índice

1. Conceptualización

1.1. Resumen

El proyecto abarca la creación de una idea para un videojuego, pasando por el proceso de game design, donde desarrollaremos la idea, obteniendo un guion, especificaremos el target, mecánicas del juego y que dinámicas se generaran, los flujos de partida, los controles del juego, las sensaciones a transmitir durante el juego, los personajes y su trasfondo psicológico y los distintos niveles que aparecerán durante el juego.

Este proyecto también incluye la creación de una banda sonora acorde con el juego, con distintas pistas para diferentes zonas del juego o eventos. Así como un estudio del estado del mercado y la historia de los videojuegos, un análisis de las tecnologías que se han utilizado para el desarrollo del proyecto y la organización que hemos seguido en el proyecto.

En la segunda parte, veremos cómo se han utilizado las tecnologías para dar forma al concepto del juego creando un motor mediante el middleware Unity 3D, con una explicación de los diferentes códigos utilizados para realizar cada parte.

1.2. Objetivos

Generales

- Investigar las fases de desarrollo de un videojuego.
- Estudiar las estrategias de comunicación para un videojuego.
- Conocer los modelos de negocio.
- Desarrollar la conceptualización y un motor para un videojuego.
- Planificar y estructurar un proyecto de un videojuego.
- Conocer las técnicas de usabilidad que se aplican a los videojuegos

Específicos

- Aplicar y relacionar los conocimientos adquiridos en diversas asignaturas del grado en multimedia.
- Conocer a fondo el software Unity 3D, el uso de assets externos y productos de terceros.
- Descubrir y comprender el flujo optimo entre los programas de modelado y Unity.
- Aprender a organizar y optimizar los recursos del proyecto.
- Creación de un videojuego acorde con nuestro target.
- Crear un videojuego con usabilidad y accesibilidad.

1.3. Tecnologías

En este apartado se explica la elección y justificación de las tecnologías que se han elegido para este proyecto, únicamente se incluyen las tecnologías implicadas en la creación del motor.

1.3.1. Unity 3D

Unity 3D es un middleware perteneciente a Unity technologies, es un software que pretende facilitar el desarrollo de los videojuegos aportando una tecnología para la creación de características y funcionalidades intentando ser el máximo de reaprovechable que pueda.

Decidimos usar este programa ya que estamos familiarizados con él, nos permite trabajar de forma bastante ágil y flexible e incluye muchos elementos ya creados que son de gran ayuda, como por ejemplo el Input manager, permitiéndonos crear de forma fácil grupos que reaccionan según la tecla, facilitando el uso de más de una tecla para una misma acción, o la personalización de las teclas, o las funciones pre establecidas como OnGUI que permite crear elementos gráficos.

Además actualmente es uno de los programas más utilizados en estudios pequeños debido a la facilidad de aprenderlo y la cantidad de elementos pre creados que existen y la posibilidad de reciclar elemento de otros proyectos o descargase contenido.

1.3.2. C#

Decidí usar este lenguaje para aprender un nuevo lenguaje, además al ser un lenguaje orientado a objetos es más sencillo para trabajar en videojuegos, deriva de C que es un lenguaje que hemos visto durante el grado, y complemento así la asignatura BETMA III, donde hemos visto algo de JAVA.

También me decidí por este lenguaje ya que es uno de los más utilizados por los desarrolladores en la plataforma Unity.

1.4. Estado de la industria

1.4.1. Estudio de la opinión de los RPG

Decidimos realizar una encuesta a diversos grupos de personas para conocer su opinión sobre los videojuegos, y más concretamente que saben de los videojuegos rpg, conocer estas opiniones es importante para poder entender el estado de la industria y su visión en la sociedad, esta encuesta ha sido realizada mediante la herramienta de google *Formularios* permitiéndonos realizar la encuesta de forma online y organizar los datos de forma sencilla y eficaz, ya que nos da los resultados en una hoja de cálculo y nos permite realizar comprobaciones de los resultados de forma más rápida, además de permitir el envío facilitando que llegue a todos los usuarios que participan y puedan cumplimentar el formulario de forma sencilla y sin ninguna molestia extra.

Los objetivos que perseguimos son poder comparar las respuestas de personas de distinta edad, separándolas en franjas de edad, y ver las diferencias existentes según la edad, también comprar las respuesta según el sexo e incluso la región donde viven estas personas, para poder comparar la percepción social que tienen, y conociendo esta realidad poder adaptarnos mejor a nuestro target o conocer las necesidades de aquellos que no están en él, y poder intentar incluirlo o animarlos a jugar.

Después de realizar la encuesta a X personas, hemos analizado los datos, y llegado a las siguientes conclusiones.

Datos demográficos:

- Hombres: 45%
- Mujeres: 55%
- Entre 16 y 18 años: 5%
- Entre 18 y 25 años: 73%
- Entre 25 y 30 años: 14%
- Entre 30 y 40 años : 5%
- Mayores de 40 años: 5%

Conclusiones extraídas:

La totalidad de los participantes han jugado a algún tipo de videojuego, de forma que podemos decir que ya no son un mero entretenimiento para jóvenes, si no que personas de todas las edades juegan. Al 91% de los encuestados les gustan los videojuegos, y la mayoría, 24%, juegan unas 5-10h a la semana, seguidos por un 19% en 1-3h y menos de 30 min, de esto podemos extrapolar los hábitos de los jugadores, unos son jugadores más hardcore, dispuesto a dedicar más tiempo jugar, y posiblemente a juegos con una mayor dificultad, y después una gran corriente cada vez mayor de personas que no están acostumbradas a jugar, pero que gracias a los juegos casual cada vez juegan más.
En esta muestra de personas podemos ver que el género que más predomina es el RPG, un 33% de las personas encuestadas, seguido por juegos de estrategia, aventura y plataformas, difiriendo de lo que otros estudios dicen, siendo el casual el género más popular.

A la pregunta de si se deben considerar los videojuegos como una obra de arte todas las personas han opinado que sí, y al preguntarles él porque han contestado que por su finalidad comunicativa, son una expresión de ideas y el proceso de creación, que involucra varios medios artísticos, o la combinación de diversos artes y porque son una expresión de sentimientos, creatividad e imaginación de los creadores/desarrolladores.
Un 73% considera que los padres deben informarse antes de ir a la tienda sobre el tipo de juego que es, y si está recomendado o es adecuado para sus hijos, un 18% creen que en la tienda deberían de informar a los padres y solo un 9% no considera necesario informarse sobre el tipo de juego que es.
Al preguntarles porque no siempre se informan, la mayoría han dicho que por pasotismo y tener el hijo contento.
Un 77% conoce PEGI, y de este 77% un 89% cree que no se siguen sus recomendaciones, debido a que la gente se basa en otros aspectos, y no se fijan en estas advertencias, no lo consideran un dato relevante a la hora de elegir el juego.
El 91% reconocieron los símbolos de PEGI, y no consideran que sea necesario añadir nuevos símbolos, ya que los que hay expresan bien las ideas, a pesar de que en la imagen que se les mostraba faltaban algunos símbolos más modernos.

El 100% de los encuestados consideran que los videojuegos pueden ser una herramienta muy positiva para la educación, pero consideran que o deben sustituir un sistema más tradicional y hay que preparar bien los juegos para explotarlos al máximo con fines educativos.

En relación con los juegos de rol el 91% los conoce, y un 73% ha jugado alguna vez a uno, de este grupo la mayoría los considera divertido, como una forma de estimular y mejorar las capacidades mentales, una forma de ser más empáticos y mejorar la comunicación. Todos creen que pueden aplicarse a la educación para ayudar a los niños a desarrollarse y aprender, y reconocieron sus beneficios para el desarrollo intelectual, la mayoría opina que no son juegos violentos ni que hagan gente violenta pese a su fama.

Algunos fueron capaces de identificar el significado de las siglas RPG, y un 55% conoce el subgénero del JRPG, aunque un 60% prefiere el RPG tradicional.
Un 95% dejara jugar a sus hijos a videojuegos, demostrando así la grana aceptación que tiene en las nuevas generaciones y que ya no se ve como algo raro, también un 95% les dejara jugar a juegos de rol, y un 77% participara en la selección de juegos junto con su hijo, aunque la mayoría dicen que solo le harán sugerencias e intentaran vitar que no juego a juegos que puedan ser una mala influencia, o intentar que sean reforzadores educativos. Un 77% jugara con ellos sobre todo para evitar ciertas emociones negativas, y para disfrutar de un momento padre-hijo y poder ayudarlos.

Comparamos estos datos con el informe de Entretainment software association essential facts about the computer and video game industry del año 2014, donde analiza la sociedad Americana y como juega, dice que el 59% de los americanos juegan, una tendencia que crece, como hemos podido ver que cada vez está más aceptado, cada hogar cuanta con dos jugadores por media, la mayoría juegan en consola, seguido de smartphones, que en un año han aumentado su uso en un 22%, la edad media de

jugador es 31 años, debido a la introducción de nuevos jugadores y que la gente que jugaba en los inicios han crecido y han seguido jugando, y un 52% de los jugadores son masculinos, aunque destaca que el número de mujeres que juega con más de 50 años en el último año ha aumentado un 32%, respecto PC la edad media del comprador de juegos es 35 años y a nivel de sexos es igual 50%, los jugadores dicen que aprovechan mejor el dinero invertido en videojuegos que en otro tipo de entretenimiento, el 48% dice que los gráficos, la trama y el boca-boca es lo más interesante para comprar un juego, y el 21% se preocupa por el precio.
El número de jugadores casual ha aumentado en un 55% en el último año, contrastando con nuestro resultado, y siendo este más cercano a la realidad, siendo estos los más jugados con un 30% del total de juegos jugados, y en Smartphone aumenta hasta un 46%.
Hoy en día se dedica más tiempo a videojuegos que a otras formas de entretenimiento, y crece el juego social, sobre todo con la propia familia.
El 88% de los padres consideran que el sistema ESRB es muy eficiente y ayuda a elegir juegos para los niños, y el 85% lo conocen.
El 87% de los padres controla a lo que juegan sus hijos, un resultado similar al de la encuesta realizada, y el 91% de los padres acompaña a su hijo a comprar/alquilar juegos, y un 82% de los niños recibe el permiso de sus padre para comprarse un juego.
El 56% de los padres considera a los juegos beneficiosos para sus hijos.
Las principales razones por las que juegan con sus hijos son la diversión para toda la familia, se lo piden o socializar con sus hijos.
Los juegos más vendidos son los de acción, 30%, los de rol solo representan un 12.3%.

Resultados de la encuesta essential facts about the computer and video game industry: http://www.theesa.com/facts/pdfs/ESA_EF_2014.pdf
Los datos obtenidos en esta encuesta se pueden leer en Anexos 5.4.Datos obtenidos de la encuesta sobre videojuegos y su percepción social

1.4.2. Evolución de los juegos RPG

Los juegos de rol actuales son la evolución de juegos más clásicos, los juegos de rol actual tienen su origen a finales de los años 60, cuando el profesor de sociología William A. Gamson, creo el juego SimSoc, un simulador social en 1966, el objetivo de este juego era educacional, se empelaba para enseñar aspectos de sociología, política y mejorar las habilidades de comunicación.

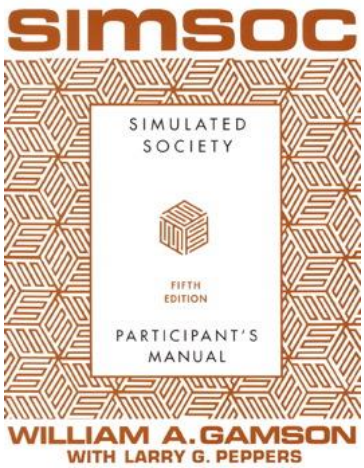


Imagen de la portada de SimSoc

Este juego no tenía tablero, ni fichas, se basa en la interpretación y el dialogo, los autores lo publicaron por su cuenta ya que ninguna editorial creía que pudiese tener éxito, el concepto tomo forma en 1974 con la publicación de Dungeons and Dragons, por Gary Gygax y Dave Arneson, basándose en los juegos de estrategia, wargames, introduciendo elementos de fantasía.

El éxito de estos juegos hizo que a mediados de los años 70 empezarán a aparecer videojuegos de rol, uno de los primero juegos en aparecer fue *Dungeons n Dragons,* publicado por CLOAD en 1980 para TRS-80, el juego tenía una analizar de líneas de comandos, generación de personajes, trampas y una mazmorra.

En los orígenes casi todos los juegos eran adaptaciones de D&D, como lo fueron *Ultima* o *Wizardry,* siendo en primera o tercera persona, o mezclando ambas, uno de los juegos que más ha influido en el desarrollo de los juegos actuales de rol es *Ultima III: Exodus,* publicado en 1983 por Origins Systems, es el primer juego que incluyo personajes animados, la posibilidad de controlar a un grupo de personajes y no únicamente un personaje, con una pantalla aparte de combates, y creando un sistema de combate más elaborado, incluyendo armas y magias.

Pantalla principal de Exodus III

En 1985 salió el juego *Phantasia*, introduciendo el concepto de automapping, proveyendo al jugador de pistas e información de trasfondo. En 1988 apareció *Pool of radiance*, inició de una saga Gold Box, basado en las reglas de *Advance Dungeons & Dragons*, ofreciendo una vista en primera persona, y combates tácticos en 3 persona, se popularizaron los combates por turnos, y se inició la edad dorada de los videojuegos de rol, los años 90, con juegos clásicos como *The Bards Tale, Wastedland, La saga Might and Magic* y la continuación de *La saga Ultima.*

Pantalla de Pool of radiance

A mediados de los 90 incrementaron los ARPG, basado en sprites e isométricos, un ejemplo de estos juegos son *Fallout, Baldurs Gate, Icewind o Diablo.* En esta época aparecieron los primeros motores 3D y con ellos juegos como *The elders scroll: Arena o Mandate of heaven.* Hacia el año 200 se impusieron los motores 3D.

En consolas el primer juego que apareció fue *Dragonstomper* en 1982 para Atari 2600, seguido por *Bokosuka wars* lanzado en 1983 para Sharp X1, este último estableció las bases de los SRPG y después de los ARPG.

Uno de los juegos RPG más importantes de consola es *Zelda II: the adventure of link* (1987) es una de los primero juegos ARPG de la historia, a diferencia del original *The legend of zelda* que a pesar de ser un juego de rol es más cercano a un juego de aventuras-acción.

Pantalla de Zelda II: the adventure of link

En 1986 aparece *Dragon quest,* considerado al base de los juegos de rol en consola, otro de los juegos más importantes es Final Fantasy, que introdujo la vista lateral y fue considerado norma en los juegos.

A principios de los 90 los juegos de consola se basaban más en trama que los de PC, siendo más similares a los juegos de rol tradicionales, tomando mucha fuerza en consolas e influyendo sobre los rpg de PC.

En los 90 apareció el disco óptico, permitiendo misiones más complejas, audio de mejor calidad, personajes más detallados y videos CGI, el mayor ejemplo de esto es *Final Fantasy vii.* A principio de los 90 los RG eran más cercanos a la novela de fantasía, pero hacia final de la década se fueron convirtiendo en algo mucho más cinemático, sobre todo los JRPG.

Imagen del videojuego Final Fantasy VII

Durante la historia de los videojuegos de rol han aparecido muchos subgéneros, la mayoría de ellos fruto de su mezcla con otros géneros, pero hay que destacar dos:

JRPG: son los juegos de rol creados en Japón, con gráficos más brillantes y cercanos al anime, personajes jóvenes e historias lineales con tramas intrincadas y mucho más cinemáticos. En muchas ocasiones con tonos menos serios, son diseñados con un énfasis en la belleza estética, y van dirigidos a un público mayor, suele utilizar batallas por turnos.

WRPG: son los juegos de rol creados en Europa, tienden a tener gráficos más oscuros, con personajes de mayor edad, se centran en la libertad del jugador, el realismo y las mecánicas de juego, es común que permitan al jugador crear su personaje e incluyan un árbol de diálogos que permita al jugador tomar sus propias decisiones respecto a la trama, se han mantenido más fieles al estilo de la novela, pero a finales del 2000 empezaron a hacerse más cinemáticos, con personajes de mayor edad y masculinos, y un público objetivo más reducido, hombres, tanto adolescentes como adultos.

1.4.3. Referentes

Buscamos en el mercado juegos que pudieran aportarnos alguna mejora a nuestro juego, juegos que han triunfado en el mercado similares al nuestro, no únicamente en nuestra plataforma si no en otras plataformas, y también buscar algún juego que no tuvo tanto éxito y analizar el por qué.

En mi caso analice dos juegos, el primero de ellos, Ragnarok Odyssey, un juego actual para PlayStation Vita, un juego destinado a un target que le gusta la acción y superar retos, sin historias paralelas, este grupo está representado en el Bartle test como los Killers. El juego está basado en superar misiones, ordenadas por niveles, con distintos grados de dificultades marcados por una serie de colores, indicando su dificultad. Tras leer los análisis de la prensa sobre el juego, vi que había algunos elementos a tener en cuenta para el desarrollo de nuestro juego.

El segundo juego que analice fue Tales of Symphonia, el primer juego de la saga Tales Of. Un juego lanzado originalmente en Game Cube, este es un clásico de los juegos RPG, este juego está pensado para un target al que le guste explorar mundos abiertos, le gusten historias profundas, conseguir títulos, incluye opciones de personalización y una historia principal muy rica con múltiples historias secundarias, está destinado a los Achivers y Explorers, personas que les guste la abstracción, la aventura y los retos.

De estos juegos extraje una serie de elementos que gustan mucho, y otros que no han funcionado bien, los que hemos tenido en cuenta a la hora de realizar el juego.

Estos elementos han sido:

- La importancia de una buena jugabilidad, con controles sencillos.
- Batallas a tiempo real ayudan a conseguir una mejor inmersión en la historia y la acción, el sistema de combate tiene que ser ágil y fácil de dominar, pero difícil llegar a alcanzar los combos más poderosos.
- Los elementos de personalización mejora la identificación del jugador con el personaje, pero estas opciones deben de ser excesivas ya que pueden crear confusión en el usuario.
- Es buena idea añadir una serie de historia s secundarias para evitar caer en la monotonía y dar una mayor variedad de opciones al jugador, también nos sirven como elemento regulador de dificultad.
- La incorporación de efectos sonoros y una buena banda sonora ayuda en la inmersión del jugador y favorece la credibilidad, estos fallos producen errores muy molestos.
- El personaje tiene que evolucionar junto con la historia y acorde a esta.
- Cada enemigo tiene que tener puntos débiles y fortalezas, obligando al jugador a buscar nuevas estrategias y adaptarse a sus enemigos.
- Tendremos que buscar una calidad gráfica acorde con nuestro target.

Para leer los análisis completos, ir a Anexos 5.1.Análisis completos de referentes y a Anexos 5.2 para leer los análisis de la parte gráfica realizados por Juan Mesa.

1.5. Target

Para este proyecto hemos elegido un target principal de persona joven, que le gustan los videojuegos y tiene suficiente tiempo libre para dedicar varias horas a la semana a jugar, así como que sea una persona que le guste la aventura, y la exploración, descubrir nuevos lugares y obtener nuevos elementos, así como la posibilidad de personalizar su avatar y crearse un personaje a su gusto, personas que les guste superar retos y sean altamente imaginativas, personas que busquen la abstracción del mundo real y sean persistentes hasta que lo logren.

Podemos clasificar nuestro target en achiver y explorers, según el test de Bartle, o como personas eficientes y abiertas en el test de Big five personality trait.

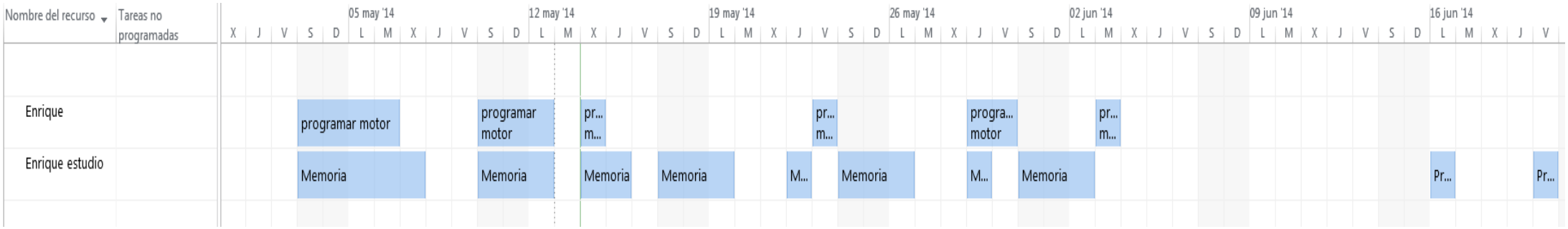
1.6. Planificación

1.6.1. Diagrama de Gant

		Modo de	Nombre de tarea	Trabajo	Duració	Comienzo	Fin
1	✓		**Trabajo final de grad**	**401,05 horas**	**306 días**	**vie 26/04/13**	**vie 27/06/14**
			Amortización PC	*1*		*vie 26/04/13*	*vie 27/06/14*
			Licencias softwar	*1*		*vie 26/04/13*	*vie 27/06/14*
2	✓		**Gestiones**	**0 horas**	**258 días**	**nar 02/07/13**	**vie 27/06/14**
3	✓		Presentación de	0 horas	0 días	nar 02/07/13	nar 02/07/13
4	✓		Entrega del trab	0 horas	0 días	vie 27/06/14	vie 27/06/14
5	✓		**Apredizaje**	**125,3 horas**	**99 días**	**vie 26/04/13**	**nié 11/09/13**
6	✓		Busqueda de in	15,3 horas	16 días	vie 26/04/13	vie 30/08/13
			Enrique estu	*15,3 horas*		*vie 26/04/13*	*vie 30/08/13*
7	✓		Cursos y formac	97 horas	46 días	lun 17/06/13	nié 11/09/13
			Enrique estu	*97 horas*		*lun 17/06/13*	*nié 11/09/13*
8	✓		Busqueda de nu	8 horas	1 día	vie 26/04/13	vie 26/04/13
			Enrique estu	*8 horas*		*vie 26/04/13*	*vie 26/04/13*
9	✓		Aprendizaje de	5 horas	5 días	vie 26/04/13	nié 12/06/13
			Enrique estu	*5 horas*		*nié 01/05/13*	*nié 12/06/13*
10	✓		**Conceptualización**	**60,25 horas**	**145 días**	**nar 08/10/13**	**lun 28/04/14**
11	✓		**Guión**	**25 horas**	**93 días**	**nar 15/10/13**	**jue 20/02/14**
16	✓		**Game design**	**26,75 horas**	**113 días**	**nar 08/10/13**	**jue 13/03/14**
17	✓		Concepto	4,5 horas	3 días	nar 08/10/13	nar 15/10/13
			Enrique	*4,5 horas*		*nar 08/10/13*	*nar 15/10/13*
18	✓		Redacción dc	22,25 horas	19 días	nar 15/10/13	jue 13/03/14
			Enrique	*22,25 horas*		*nar 15/10/13*	*jue 13/03/14*
19	✓		Estudio de la in	8,5 horas	7 días	nar 18/03/14	lun 28/04/14
			Enrique	*8,5 horas*		*nar 18/03/14*	*lun 28/04/14*
20	✓		**Desarrollo del mo**	**191,75 horas**	**248 días**	**vie 21/06/13**	**nar 03/06/14**
21	✓		programar moto	170,25 horas	68 días	nié 26/06/13	nar 03/06/14
			Enrique	*170,25 horas*		*nié 26/06/13*	*nar 03/06/14*
22	✓		Diseñar motor	21,5 horas	15 días	vie 21/06/13	jue 29/08/13
			Enrique	*21,5 horas*		*vie 21/06/13*	*jue 29/08/13*
23	✓		**Memoria + presen**	**23,75 horas**	**64 días**	**nar 01/04/14**	**vie 27/06/14**
24	✓		Memoria	20 horas	15 días	nar 01/04/14	lun 02/06/14
			Enrique estu	*20 horas*		*nar 01/04/14*	*lun 02/06/14*
25	✓		Presentación	3,75 horas	4 días	lun 16/06/14	vie 27/06/14
			Enrique estu	*3,75 horas*		*lun 16/06/14*	*vie 27/06/14*

1.6.2. Calendario de planificación

	Modo de	Nombre de tarea	Duración	Comienzo	Fin	Predecesora	Nombres de los recursos	Costo	Trabajo normal
1		**Trabajo final de grado**	**306 días**	**vie 26/04/13**	**vie 27/06/14**		**Amortización PC[1]; Licencias software[1]**	**11.819,63 €**	**401,05 horas**
2		**Gestiones**	**258 días**	**mar 02/07/1**	**vie 27/06/14**			**0,00 €**	**0 horas**
3		Presentación de la propuesta del trabajo	0 días	mar 02/07/13	mar 02/07/13			0,00 €	0 horas
4		Entrega del trab	0 días	vie 27/06/14	vie 27/06/14			0,00 €	0 horas
5		**Apredizaje**	**99 días**	**vie 26/04/13**	**mié 11/09/1**			**1.879,50 €**	**125,3 horas**
6		Busqueda de información	16 días	vie 26/04/13	vie 30/08/13		Enrique estudio	229,50 €	15,3 horas
7		Cursos y formación en C#	46 días	lun 17/06/13	mié 11/09/13		Enrique estudio	1.455,00 €	97 horas
8		Busqueda de nuevos asets	1 día	vie 26/04/13	vie 26/04/13		Enrique estudio	120,00 €	8 horas
9		Aprendizaje de metodologias de trabajo	5 días	vie 26/04/13	mié 12/06/13		Enrique estudio	75,00 €	5 horas
10		**Conceptualización**	**145 días**	**mar 08/10/1**	**lun 28/04/14**			**2.108,75 €**	**60,25 horas**
11		**Guión**	**93 días**	**mar 15/10/1**	**jue 20/02/14**			**875,00 €**	**25 horas**
16		**Game design**	**113 días**	**mar 08/10/1**	**jue 13/03/14**			**936,25 €**	**26,75 horas**
17		Concepto	3 días	mar 08/10/1	mar 15/10/1		Enrique	157,50 €	4,5 horas
18		Redacción documento	19 días	mar 15/10/13	jue 13/03/14		Enrique	778,75 €	22,25 horas
19		Estudio de la industria	7 días	mar 18/03/14	lun 28/04/14		Enrique	297,50 €	8,5 horas
20		**Desarrollo del motor**	**248 días**	**vie 21/06/13**	**mar 03/06/14**			**6.711,25 €**	**191,75 horas**
21		programar motor	68 días	mié 26/06/13	mar 03/06/14		Enrique	5.958,75 €	170,25 horas
22		Diseñar motor	15 días	vie 21/06/13	jue 29/08/13		Enrique	752,50 €	21,5 horas
23		**Memoria + presentación**	**64 días**	**mar 01/04/14**	**vie 27/06/14**			**356,25 €**	**23,75 horas**
24		Memoria	15 días	mar 01/04/1	lun 02/06/14		Enrique estudio	300,00 €	20 horas
25		Presentación	4 días	lun 16/06/14	vie 27/06/14	24	Enrique estudio	56,25 €	3,75 horas


Nombre del recurso
Tareas no programadas
05 may '14
12 may '14
19 may '14
26 may '14
02 jun '14
09 jun '14
16 jun '14
X J V S D L M X J V S D L M X J V S D L M X J V S D L M X J V S D L M X J V S D L M X J V S D L M
Enrique
programar motor
programar motor
pr... m...
pr... m...
progra... motor
pr... m...
Enrique estudio
Memoria
Memoria
Memoria
Memoria
M...
Memoria
M...
Memoria
Pr...
Pr...

1.6.3. Hoja de recursos

	Nombre del recurso	Tipo	Etiqueta de material	Iniciales	Grupo	Capacidad máxima	Tasa estándar	Tasa horas extra	Costo/U:	Acumu	Calendario base	Cód	Costo	Trabajo
1		Trabajo				100%	0,00 €/hora	0,00 €/hora	0,00 €	Prorrateo	Estándar		0,00 €	0 horas
2	Amortización PC	Material		A_PC			216,67 €		0,00 €	Comienzo			216,67 €	1
3	Enrique	Trabajo		E	1	100%	35,00 €/hora	0,00 €/hora	0,00 €	Prorrateo	Estándar		8.820,00 €	252 horas
4	Enrique estudio	Trabajo		E_est		100%	15,00 €/hora	0,00 €/hora	0,00 €	Prorrateo	Estándar		2.235,75 €	149,05 horas
5	Licencias softwar	Material		L			547,21 €		0,00 €	Prorrateo			547,21 €	1

1.6.4. Coste del proyecto

Para calcular el coste de este proyecto he intentado aplicar todos los parámetros que habría que aplicar a un proyecto real.

El gasto principal de un proyecto como este es el producido por los trabajadores, después de investigar y consultar varias fuentes, he llegado a que el salario aproximado es de 35€/h, de los que el empleado percibe 15€ y el resto son impuesto u otros gastos, para las horas de aprendizaje he utilizado el salario de 15€/h ya que son horas dedicas al proyecto, pero no las contabilizo como trabajo directo.

Para este proyecto también se han tenido en cuenta las amortizaciones del material, como por ejemplo el PC donde se ha trabajado, el cual se espera que tenga una vida útil de 5 años, y decidí amortizarlo a 3 años, de forma que durante dos años se acumulara dinero para comprar el siguiente, el coste fue de 1300€ aprox. Lo que nos deja una amortización de 433,34€/año, por la duración del proyecto vemos que actualmente se podrían abarcar dos proyectos por año, lo que nos deja con una amortización de 216,67€/proyecto.

También hay que tener en cuenta el software, para este proyecto es necesario Unity pro, ya que hemos utilizado algunas de sus características, para el modelado 3D se ha utilizado blender, que no representa coste alguno, la edición de audio se ha realizado en un software libre por parte del compositor, y no ha sido necesario el uso de más software. La licencia de unity pro, sin complementos es de 1500$(1.094,41€) cada año, dejándonos 547,21€ por proyecto en amortización.

Todo esto nos da un total de 11.819,63€, siendo este un presupuesto elevado, este es debido a la cantidad de horas de estudio invertidas, en otros proyecto al necesitar menso horas de estudio, ya que solo tendré que ampliar conocimientos y actualizar otros, el tiempo de producción se reduce en gran medida.

1.7. Guion

1.7.1. Idea

En la aldea de Hjem se celebra de forma anual un torneo de caza, en el que participan todos los hombres del pueblo, el día de la celebración del LXVII torneo de caza, nuestro joven protagonista toma parte en el torneo sin conocer como cambiara su vida, ya que durante el trascurso del torneo al seguir el rastro de una presa llegará a una cueva, donde descubrirá un conjunto de runas con un sello roto, este hecho hará que el protagonista tenga que abandonar la vida en su tranquila aldea para adentrarse en la resolución del misterio de quién está detrás de la liberación de los sellos y cuál es su objetivo, a lo largo de este viaje conocerá personas que le harán evolucionar y descubrir su destino y la verdad que hay oculta tas su mundo.

1.7.2. Focus group

Con la idea y el target pensados, decidimos realizar una técnica de usabilidad llamada focus group, decidimos crear una serie de reactivos y realizar una reunión con 4 posibles jugadores, donde les presentamos los reactivos y así pudimos obtener información de que les gustaba más o que partes no las veían bien, si algún elemento era susceptible de provocar malestar o incomodar a los jugadores.

Para este focus group realizamos los siguientes pasos:

1. Pensar posibles reactivos, buscamos aquellos elementos que podían provocar inconformidad, como puede ser la violencia, o elementos de diseño, durabilidad u otros ítems.
2. Seleccionamos una muestra de 5 posibles jugadores analizando nuestro target, intentado tener la mayor representación posible del target.
3. Citamos a los participantes a un área preparada, donde se pudieran estar relajados, y les presentamos entre ellos, y les hablamos del proyecto y la prueba que iban a hacer.
4. Colocamos a los usuarios en una mesa redonda intentando evitar confrontaciones directas para que se encontraran a gusto y todos se pudieran oír y ver de forma natural.
5. Iniciamos la sesión presentando el primer reactivo y dando la palabra a un usuario, mientras anotábamos las ideas generales que iban surgiendo.
6. Al acabar la sesión repasamos las ideas surgidas con los usuarios, mostrándose estos conformes.

Días después redactamos unas conclusiones de esas ideas generales y las utilizamos como guía a la hora de realizar el guion.

En Anexos 5.3.Informe del focus group se puede consultar el informe generado en este focus group.

1.7.3. Guion literario

Una vez establecida la idea, y con la información extraída del focus group, iniciamos la redacción de una escaleta, donde planificamos los eventos principales del juego, una vez planificados lo eventos principales empezamos a desarrollar las conexiones entre estos, y crear los diálogos principales. También empezamos a definir las líneas psicológicas de algunos de los personajes, finalmente desarrollamos el esqueleto del guion que fuimos completando incluyendo eventos secundarios, y diálogos adicionales, una vez realizado el guion acotamos la parte correspondiente a la demo.

1.7.4. Guion técnico

Para la parte del guion perteneciente a la demo desarrollamos un guion técnico anotando que elementos intervenían en cada parte, tanto assets, clips de audio, texturas o scripts, para delimitar el trabajo a realizar, y centrarnos en esas partes.

1.8. Game design

1.8.1. Concepto

Torchsnow es un juego perteneciente al género RPG, se basa en un mundo abierto, donde el jugador podrá personalizar y desarrollar el personaje a su gusto mediante una historia y una serie de misiones, a la vez que el jugador vive una historia que busca hacerle reflexionar sobre diversos temas como las diferencias entre las personas en el mundo, como madurar y aceptarse a uno mismo o las consecuencias de nuestras acciones.

1.8.2. Objetivos del jugador

El objetivo principal del juego consiste en que el jugador derrote al líder de una secta religiosa que pretende dominar el mundo mediante el uso de unos entes que fueron encarcelados en nuestro mundo, otro de los objetivos más importantes del jugador es el desarrollo de su propio personaje.

El juego también cuenta con objetivos secundarios, como la caza de una serie de entes, que permite conocer más detalles de la historia y mejorar el rango dentro del clan y obtener recompensas, o misiones en las que ayudamos a ciudadanos permitiéndonos acceder a nuevos servicios en la ciudad, o a ítems que no se pueden comprar.

1.8.3. Sensaciones a transmitir

El juego busca la identificación del jugador, para conseguirlo empleamos principalmente los sentimientos, según el momento del juego se emplea un estilo grafico ligeramente distinto, o una paleta de colores diferente, adecuándose a la parte de la historia y el contexto de esa zona.

1.8.4. Sistema de juego

El sistema de juego es a tiempo real, en un mundo abierto.

1.8.5. Controles

Los controles del juego son los siguientes:

Para caminar se puede realizar el input con las teclas WASD, o las teclas de dirección del teclado, para correr apretar una vez la tecla shift y después las teclas de movimiento, al volver a apretar la tecla shift se vuelve a caminar de forma normal. Para desplazarse lateralmente se utilizas las teclas Q y E, para nadar las mismas que para andar, para saltar se utiliza la tecla de barra espaciadora, para atacar se presiona al tecla F, únicamente se realizara el ataque si el jugador tiene marcado el objetivo, y para acceder a la habilidades de combate la tecla k, para acceder al menú del personaje se aprieta al tecla C, y se navega utilizando el mouse, para acceder al inventario se aprieta la tecla I, se seleccionan los objetos con el mousse, dando doble click para utilizarlo, o poner el mouse encima para ver los stats del ítem. El control de cámara se puede realizar con el boto derecho del mouse manteniéndolo clickado y moviendo el mouse, o con el num pad usando las teclas 8 4 6 2.

Elegimos estos controles ya que son parecidos a los de otros juegos, además de guardan relación con lo que hacen, como el menú de personaje C ya que es la inicial

de Character, y queda cerca de la posición de la mano. Pero en el juego en opciones se incluirá la opción de personalizar los controles.

1.8.6. Personajes

Durante el transcurso del juego aparecerán diversos personajes, estos se pueden clasificar en:

1.8.6.1. Protagonista

El protagonista es un héroe silencioso, facilitando la identificación del jugador con él, las características físicas son definidas por el usuario al crearlo, y sus estadísticas, de forma que será un personaje definido por el usuario con las opciones presentes.

1.8.6.2. Enemigos

Dependiendo de la zona variaran los enemigos presentes, adaptándose estos al medio, estos enemigos variaran entre ellos pudiendo tener o no magia, o teniendo ligeras variaciones aleatorias en la vida, fuerza u otros elementos de sus estadísticas.

En la fase inicial encontraremos 2 tipos de enemigos:

- MOB: Son el enemigo base, similares a un gusano, utilizan su cuerpo para golpear, hay tres tipos, uno sin magia de color gris, uno de elemento fuego y otro de elemento agua, que utilizaran ataques físicos pero tiene resistencia a un elemento.
- Dungeon Guardian: es el primer tipo de goblin que encontraremos, tienen poca inteligencia, y son bastante agresivos, únicamente utilizan ataques físicos, las armas que utilizan varía entre una lista de posibles armas

1.8.6.3. NPCs

Los NPC son aquellos personajes que no pueden ser controlados por el jugador, tenemos varias clases:

- Comerciantes: son aquellos con los que el jugador puede interactuar para intercambiar objetos obtenidos en cofres o al eliminar enemigos, por otros.
- Secundarios: personajes relevantes para el desarrollo de la historia.
- Peticionarios: son aquellos que te piden que cumplas algún encargo a cambio de una recompensa
- Gente: Son personas que están repartidas por el mundo y únicamente realizan comentarios sobre la situación de la zona, dando pistas al jugador de posibles secretos.

1.8.7. Niveles de dificultad

La dificultad se regula mediante las misiones, con unas primeras misiones más cortas y sencillas, y a medida que avanza va aumentando la duración y complejidad para superar las misiones, incluyendo más puzles más complejos u objetivos más complejos o compuestos.

Los enemigos también aumentan de dificultad al ir avanzando en el juego, tendrán unos límites de mínimos y máximos de sus stats, y dentro de este margen se crearan de forma aleatoria, haciendo que los enemigos no sean siempre iguales, haciendo que el jugador tenga que adaptarse, ya que una vez un enemigo puede ser más rápido que la anterior, tenga más fuerza o vida.

1.8.7.1.1. Recompensas

Las recompensas de las misiones estarán ligadas a su dificultad, se entregaran ítems útiles para cumplir otras misiones o mejorar al personaje, los enemigos dropearan ítem, cada enemigo tendrá una lista de posibles ítems para dar, una vez eliminado el jugador tendrá la opción de abrir la ventana de loot y recoger ese objeto generado aleatoriamente de la lista de ese enemigo, a excepción de los grandes jefes, los que tendrá una única opción de dropeo.

Además los enemigos recompensaran con un punto de habilidad, que permitirá subir de nivel las habilidades del personaje, cada nivel requerirá un cierto número de puntos de habilidad, según el nivel de las habilidades se asignara un título u otro al jugador.

1.8.8. Level design

El mapa es un mundo abierto, este mundo estará dividido en zonas, cada zona con una estética propia y marcada, desde el inicio que es una pequeña villa de aspecto medieval, a ciudades con aspecto modernista, pantanos o bosques.

Cada zona tendrá un tipo de enemigo, y una dificultad propia de forma que el jugador avance por las zonas según su nivel de dificultad y el avance de la historia.

1.8.9. Objetos

En el juego encontraremos diversos objetos, armas que engloba todo tipo de objeto de ataque, a distancia como arcos, media distancia como lanzas y varas, o corta distancia como son espadas y dagas, también objetos defensivos, escudos y piezas de armadura, como objetos que proporcionaran buffs, pociones, comida y algunos encantamientos.

La mayoría de estos objetos requerirán que el jugador los cree, o comercie para obtenerlos, cambiando objetos dropeados por enemigos en comerciantes.

El intercambio de objetos es la base de la economía, de forma que para conseguir cierto objeto tengas que realizar unas acciones determinadas, y poder controlar mejor la curva de dificultad.

1.8.10. Banda sonora y efectos sonoros

La banda sonora ha sido cedida por diversos artistas, tenemos una canción para campo abierto, en el momento en que se entra en combate cambia a un tema de combate, y finalmente una canción para el pueblo. La intención inicial era crear una banda sonora propia basándonos en instrumentos típicos de zonas del norte

de Europa y la música profana de esas zonas durante un periodo cercano a la edad media, también adaptando estas músicas a las necesidades del juego, ya que cada ciudad o zona del mapa tendrá una ambientación propia, y se tendría que adaptar el tema a la zona, o crear temas para personajes o momentos del juego.

A nivel de efectos sonoros utilizamos sonidos de librerías gratuitas, que estarán reproduciéndose de forma continua y los podrás escuchar al acercarte a la fuente del sonido, creando así un mundo más real.

1.9. Usabilidad

En este apartado explicare lo que tenemos pensado hacer una vez concluyamos el juego, para valorar la usabilidad del juego.

Para ello una vez finalizado realizaremos unas pruebas con usuarios.

1.9.1. Pruebas con usuarios

Con el juego finalizado citaremos a un grupo de usuarios del target, y con ellos testearemos los menús, y la jugabilidad en general, nos ayudaremos de un eyetracker para comprobar que puntos son a los que el jugador mira durante más tiempo y comprobar si tiene que ser así, con esto buscamos conseguir una interfaz gráfica agradable al usuario y que este disponga en todo momento de la información necesaria.

Otra parte de estas pruebas con usuarios será comprobar errores en la jugabilidad, y la detección de posibles bugs, o comportamientos de usuario inesperados, también para comprobar si las instrucciones del juego se entienden o necesitamos añadir información.

Otro punto a valorar será la user Xperience

1.9.2. User Xperince

Para valorar la user Xperience le daremos a cada participante una hoja con varias grid, y explicaremos en que consiste, dividiremos la demo a la que jugaran en tareas y antes y después de completarlas les pediremos que rellenen una grid marcando en qué punto se encuentran.

Cuadro empelado para las valoraciones

El testeo terminara con una entrevista donde recogeremos sus impresiones sobre el juego, que mejoraría, que le ha gustado y como se ha sentido jugando, para obtener una información más detallada, finalmente se agradecerá la participación.

2. Desarrollo

Esta sección explica cómo se ha desarrollado el motor gráfico, analizando algunos de los scripts más importantes que lo forman.

Todos los scripts están adjuntos en el proyecto para su consulta.

2.1. Game settings

Game settings el script utilizado para declarar las constantes del juego tales como distancias de ataque, rutas a elementos (Prefabs), también es donde se asigna la versión del juego, punto de inicio del personaje y todas las demás variables que se mantendrán constantes durante el juego.

Las funciones de este script son diversas, tienen que asignar y crear todo aquello que es necesario para el juego, este script es usado como referencia por muchos otros, que utilizan sus constantes o llaman a sus funciones para crear o modificar los elementos.

Así que podríamos decir que es un script que controla el juego y su desarrollo.

En el tenemos las funciones de guardar la posición actual del jugador en el mundo y cargar dicha posición al reiniciar el juego, aunque finalmente no se ha implementado en la demo, para guardar la posición recibimos un Vector 3 y usando el método PlayerPrefs guardamos los valores X, Y y Z del vector, al iniciar el juego se llama a la función de cargar de posición, donde creamos una variable Vector 3 y le asignamos los valores guardados.

Otras funciones que contiene el script son aquellas relacionadas con el modelo del personaje, incluyendo, modelo, pelo, altura, ancho, color de piel y cara. Todas ellas tiene un funcionamiento similar, al seleccionar las características desde el creador de personaje se llama a estas funciones, cuyo fin es el de guardar las características elegidos por el usuario, para guardar y cargar estas características se valen del método PlayerPref, guardando el dato, o cargándolo al inicio del juego.

Este script también se encarga de guardar y cargar los atributos de personaje, los atributos de vida y las habilidades, de forma similar a las demás funciones.

```
165  #region atributos
166  /* atributos... */
169  public static void SaveAttribute ( atributeNames name, atributte attribute){
170      PlayerPrefs.SetInt(((atributeNames)name).ToString()+BASE_VALUE,attribute.BaseValue);
171      PlayerPrefs.SetInt(((atributeNames)name).ToString()+EXP_LV, attribute.ExpToLvl);
172  }
173
174  public static void LoadAttribute (atributeNames name){
175
176      PlayerCharacter2.Instance.GetPrimaryAttribute( (int)name).BaseValue = PlayerPrefs.GetInt(((atributeNames)name).ToString() + BASE_VALUE,0);
177      PlayerCharacter2.Instance.GetPrimaryAttribute( (int)name).ExpToLvl = PlayerPrefs.GetInt(((atributeNames)name).ToString() + EXP_LV,atributte.STARTING_EXP_COST);
178
179  }
180
181  public static void SaveAttributes(atributte[] attribute){
182      for(int cnt = 0; cnt < attribute.Length; cnt++){
183          SaveAttribute((atributeNames)cnt, attribute[cnt]);
184      }
185  }
186
187  public static void LoadAttributes(){
188
189      for( int cnt=0; cnt< Enum.GetValues(typeof(atributeNames)).Length; cnt++){
190          LoadAttribute((atributeNames)cnt);
191      }
192
193  }
194  #endregion
```

2.2. Personaje principal y scripts de creación

2.2.1. Base character script

En este script asignamos los gameObjects que serán los padres de los props que tendrá el personaje, como es el pelo, los cascos, armas y escudos.

También es donde se crea el nombre del personaje, llamando mediante un setter & getter las funciones de carga del nombre, este script también controla niveles y la experiencia disponible, siendo esta una variable del tipo uint para evitar que pueda ser negativa. El script también controla el timer de ataque del personaje, si esta en combate o no, los atributos y si tiene algún objeto equipado.

Al iniciarse la función vaciamos el nombre, nivel, experiencia y comprobamos los atributos, una vez acabado esto llamamos a las funciones para crear los valores.

Tenemos una regios de setter & getters, que es una función donde obtenemos el valor de una variable (get) y lo asignamos (set). También incluirá un calculador par al experiencia, a medida que el nivel aumenta será necesaria más experiencia para aumentar el nivel de la habilidad.

Después tenemos las funciones que se encargan de los atributos, vida y habilidades, creándolos y llamando a una función que los modifica, donde calculamos le valor que tiene que tener ese valor, obtiene el valor de un atributo, y mediante un ratio calcula cuantos puntos hay que asignar a vida, mana, o nivel de una habilidad.

```
private void SetupVitalModifiers(){
    //vida
        //GetVital((int)VitalName.Salud).addModifier(new ModifyingAttribute{attribute =
        ModifyingAttribute healthModifier = new ModifyingAttribute();
        healthModifier.attribute = GetPrimaryAttribute((int)atributeNames.Constitucion);
        healthModifier.ratio=.5f;
        GetVital((int)VitalName.Salud).addModifier(healthModifier);
    //energia
        ModifyingAttribute energyModifier = new ModifyingAttribute();
        energyModifier.attribute = GetPrimaryAttribute((int)atributeNames.Constitucion);
        energyModifier.ratio=1;
        GetVital((int)VitalName.Estamina).addModifier(energyModifier);

    //mana
        ModifyingAttribute manaModifier = new ModifyingAttribute();
        manaModifier.attribute = GetPrimaryAttribute((int)atributeNames.Sabiduria);
        manaModifier.ratio=1;
        GetVital((int)VitalName.Mana).addModifier(manaModifier);
}
```

Finalmente tenemos un setter&getter para asignar objetos a la mano principal, el el get comprobamos si tiene algo equipado y lo ponemos, en el set miramos el valor guardado en get, en caso de tener algún hijo, eliminamos el gameObject, y creamos mediante un Instantiate la nueva malla cargándola desde resources, y dándole los mismos valores que tiene el transform del padre.

```
public Item EquipedWeapon{
    get{ return _equipment [(int)EquipmentSlot.ManoPrincipal];}
    set{
        _equipment [(int)EquipmentSlot.ManoPrincipal] = value;
        //_equipedWeapon = value;


        if(weaponMount.transform.childCount>0){
            Destroy(weaponMount.transform.GetChild(0).gameObject);
        }
        //if(_equipedWeapon != null){
        if(_equipment [(int)EquipmentSlot.ManoPrincipal] != null){
        GameObject mesh = Instantiate( Resources.Load(GameSettings2.MELEE_WEAPONS_MESH_PATH + _equipment [(int)EquipmentSlot.ManoPrincipal].Name),weaponMount.transform.position, weaponMount.transform.rotation)as GameObject
        mesh.transform.parent = weaponMount.transform;
        }
    }
}
```

Finalmente tenemos una función para cambiar el valor de la booleana InCombat, para comprobar si está o no en combate y otra para calcular el aumento de velocidad de ataque bajo los efectos de un buff, pero no está implementada.

2.2.2. Player character

Player carácter es el script que contiene al jugador en sí, lo primero que hace es comprobar si esta instanciado o no el personaje, en caso de no existir crea una instancia de la malla y la carga en la posición deseada, comprueba si tiene el script añadido, y si no lo tiene se lo añade.

Al inicializar comprueba si se ha cargado el personaje y lo carga, cargando los atributos que hemos calculado y tenemos guardados en gameSettings, así como la piel, el pelo y la escala.

Este script también controla los ítems equipados con un get donde comprobamos cuantas cosas tenemos equipadas y un set donde miramos el valor del get, si el mayor que 0 eliminamos le objeto, e instanciamos el nuevo haciendo que los valores del transform coincidan, escalamos la malla, y en el caso del casco eliminamos el pelo dejando únicamente el bello facial.

```
public Item EquipedHats{
get{ return _equipment [(int)EquipmentSlot.offhand];}
set{
    _equipment [(int)EquipmentSlot.offhand] = value;
    // equipedWeapon = value;


    if(helmetMount.transform.childCount>0){
        Destroy(helmetMount.transform.GetChild(0).gameObject);
    }
    //if( equipedWeapon != null){
    if(_equipment [(int)EquipmentSlot.offhand] != null){
    GameObject mesh = Instantiate( Resources.Load(GameSettings2.HAT_MESH_PATH + _equipment[(int)EquipmentSlot.offhand].Name), helmetMou
    mesh.transform.parent = helmetMount.transform;
        //escalar
        mesh.transform.localScale = hairMount.transform.GetChild(0).localScale;
        //colocalr el pelo basico(vello facial)
        hairMount.transform.GetChild(0).gameObject.active = false;

    }
}
```

2.2.3. Character costumization

Es una recopilación de los scripts que intervienen en el nivel de creación de personaje para elegir es aspecto que tendrá el avatar.

2.2.3.1. Personaje básico

Está controlado por el script PlayerModelCostumization, al iniciarse instancia la malla de un modelo por defecto, este script también nos permite rotar la malla para poder ver al personaje desde cualquier angulo.

Es el encargado de controlar la malla que se muestra, existiendo la opción de cambiar la malla, para hacer esto se mueve por un array donde están los nombres de las diversas mallas de personajes, y va actualizando la instancia de la malla, estas mallas tiene unos empty objects donde irán los props.

2.2.3.2. Sexo

Este script sirve para elegir el sexo, es decir que mallas tenemos disponibles, al iniciar, está en un gameObject, al colocar el ratón encima de él lo ilumina añadiéndole un color a su textura, y al salir lo quita, al clicar lo que hace es cambiar el símbolo que aparece en el gameObject que es una textura2D y asignarla al gameObject, y enviar un mensaje en Broadcast avisando del cambio.

Este mensaje es escuchado por el script PlayerModelCostumization, que llama a la función OnToggleGender, que varía el valor de una booleana y vuelve a llamar a Instanciar el personaje. Donde mediante un bloque if else comprobando el valor de la boleana se instancia la primera de las mallas masculinas o la primera femenina.

2.2.3.3. Pelo y color de pelo

El script del pelo crea una serie de botones, dos para cambiar la malla, y 5 para cambiar el color mostrando una textura en el botón, mediante una función OnGUI, donde crea un grupo que los contiene a todos.

Al cambiar la malla del pelo aumentamos el valor de una variable, según este valor al instanciar seleccionaremos una u otra malla, hasta llegar al último valor, donde regresaríamos a la primera.

Para el color de pelo, al clicar sobre el botón guardamos la posición del botón, que coincide con la posición del nombre en la array, al instanciarlo cambiamos la main texture por la seleccionada.

Para instanciar la malla comprobamos si existe una, en caso de existir eliminamos el gameObject actual, calculamos usando el valor de la variable selectedhairMesh su valor para el set i el índex de pelo, y finalmente instanciamos este gameObject, asignándole como padre el empty object creado para el pelo, y sus valores para el trasform, entonces aplicamos un offset s y la rescalamos, si la malla lo necesita.

2.2.3.4. Altura y anchura

Este script nos permite modificar la escala del personaje para hacerlo más alto o bajo o modificar su ancho. En este script creamos dos sliders, uno horizontal y otro vertical, asignándoles un valor máximo y uno mínimo, estos valores los pasamos en el update como un Vector2 a player model costumization, donde en el update recogemos estos valores y modificamos el localScale de la malla con ellos.

2.2.3.5. Tono de piel y cara

Para la selección de tono de piel creamos 3 objetos a los que aplicamos una textura del mismo color que la piel, y les añadimos un script para que cuando el mousse este sobre ellos se iluminen, cambiando el color a amarillo, y al clicar sobre ella pasamos al script playerModelCostumization una variable del tipo int, conteniendo el valor de tono de piel, donde guardamos el valor y actualizamos la textura de la malla en la cabeza cambiando a la textura deseada.

Para cambiar la cara usamos el script HeadChanger cuando el mousse este sobre el gameObject este cambia el color a amarillo, y al clicar sobre el aumentamos el valor de la variable de cabezas, comprobamos que no supere el máximo y si no supera el máximo lo enviamos al script playerModelCostumization, donde actualizamos la textura de la cabeza usando el índice que hemos recibido.

2.2.4. Stats

Recopilación de los scripts empleados en la creación de personaje y su evolución que determinan sus Stats, como pueden ser vida, energía, mana, fuerza, habilidades u otros Stats, empleados para calcular habilidades del personaje, elementos de vida.

En este motor gráfico son un elemento muy importante ya que el jugador no subirá de nivel, si no mejorara Stats, permitiendo una mayor personalización del personaje y desarrollo a través de una profesión. Esta última parte no ha sido implementada aún, pero el objetivo es que al derrotar enemigos consigas puntos de habilidad, que puedas emplear en mejorar una serie de características de tu personaje, por cada nivel que subas de cada habilidad se recalculara la experiencia necesaria para subir ese nivel y será le total de puntos necesarios para subir esa habilidad la siguiente nivel, según los niveles de cada habilidad se asignara una profesión al jugador, de forma que el jugador tendrá la profesión que más le convenga según sus rasgos o habilidades.

2.2.4.1. Base stats

Esta script es empleado como clase base para todos los stats del juego, en este script se define la experiencia base de cada atributo, el valor base de cada atributo, el valor de buff que se suma a la base, este valor será creado por los buffs que se aplique le jugador, la experiencia necesaria para subir de nivel y la cantidad que varía la experiencia al subir de nivel. Este script sobre escribe le método baseStats, asignando valores iniciales a la instancia del método, también incluye una región de setters y getters donde asignamos y obtenemos le valor de las variables, también incluye una función para recalcular el valor de la experiencia cada vez que subimos de nivel una habilidad, donde multiplicamos la

experiencia necesaria para el nivel actual por el modificador de experiencia predeterminado, de forma que crezca el valor necesario para subir el nivel de experiencia.

También incluye una función que es llamada al subir el nivel, llamando a calcular la nueva experiencia para subir de nivel, y aumentando el valor base de ese Stat.

Y una última función encargada de calcular el aumento del valor de un Stat que es modificado por un buff.

2.2.4.2. Modiefied stats

Esta clase es la clase hereda de baseStats, y sirve da base a todos los Stats que será modificados por los atributos del personaje. Esta script crea una lista de atributos que modifican un determinado Stat, tiene una función que sirve para añadir stats a la lista, y otra que los borra, otra que resetea modValue a 0 y comprobamos que exista mínimo un valor en la lista de mods, si existe iteramos en la lista y añadimos el ajuste multiplicado por el ratio a el modvalue, empleando un *foreach* , otra función que calcula el AdjustedBaseValue sumando baseValue + BuffValue + modValue, sobre escribiendo AdjutesdBaseValue de la clase BaseStat. También contiene una estructura que sirve para guardar el atributo y el ratio de modificación del atributo, esta estructura es la que usamos en el script para calcular las variaciones aplicarlas.

2.2.4.3. Vitals

Vital es el Script encargado de controlar los atributos relacionados con la vida, magia y energía del personaje, incluye la definición de la variables de nivel actual, experiencia encesaria apra subir de nivel y el modificador de experiencia de nivel, sobreescribeindo las que hereda de baseStats, esta clase contiene un setter y un getter para asignar el valor y su modificación al valor actual. Originalmente también contenía una enumeración que contenía todos los atributos vitales del personaje, esta enumeración fue movida aun script externo para facilitar el uso del motor.

2.2.4.4. Skills

Esta clase contiene todas las funciones extras para un skill, incluye una variable del tipo boleana para indicar si conoce o no la habilidad, sobre escribimos los valores de experiencia necesaria para subir de nivel, y el valor del ratio para modificar la experiencia la subir de nivel. Este script también incluía originalmente una enumeración que ha sido movida a un script externo, esta enumeración contiene todas las habilidades del juego.

2.2.4.5. Attributes

Esta clase contiene los atributos del personaje, da valor a la experiencia inicial de los atributos, y el ratio en el que aumenta el coste de experiencia la subir de nivel. Este script contenía una enum listando todos los atributos del personaje, pero fue movido a un script externo.

2.3. Scripts de movimiento

Recopilación de los scripts empleados en el proceso de captación de los inputs del usuario, y responsables de modificar elementos en respuesta al input, y activar o desactivar las animaciones correspondientes al input.

Para la creación de las teclas de input utilizamos la propia herramienta de los inputs de Unity, facilitándonos así el trabajo a la hora de calcular desplazamientos o la dirección del mismo, mediante tecla positiva y negativa, y el tiempo que esta pulsada, así como permitiéndonos dejar al usuario que modifique las teclas del input a su gusto.

2.3.1. Advanced movement Script

Este sript es el encargado de definir las variables de movimiento, como la altua de los saltos, la velocidad de caída, de andar o correr entre otros, también añade unas enum, mediante importar el paquete de Collections, en las que definimos posibles estados para el jugador, esperar, inicializar, setup y run, en este último estado de la FSM, donde se decide que acción se realizara dependiendo del input que ha recibido. También se definen que direcciones son positivas y cuales negativas.

El iniciar el juego este script se encarga de guardar el transform del gameobject en el que esta, el CharacterController, así como otro script que es baseCharacter, una vez realizado esto coloca la máquina de estados en el estado de inicializar.

La máquina de estados está en el Start, es un bucle infinito, while (true), en el que revisa en qué estado esta, cada estado tiene una acción determinada.

- **Init**: llama a una función Init, esta función se encarga de comprobar que tiene los controladores necesarios asignados, si lo están pasamos al estado de setup.
- **Setup:** llama a la función setup, que es responsable de inicializar todas las variables necesarias para el funcionamiento del script, al acabar llama al estado run.
- **Run:** este estado llama a la función ActionPicker, esta función decide según los inputs recibidos que acción realiza, comprueba qué valor tiene asignado una variable y algunos boléanos, y lo envía a al unció que activa su animación correspondiente, esta función también se encarga de mover el gameobject

```
private void ActionPicker(){
        //girar jugador
        _myTransform.Rotate(0,(int)_turn*Time.deltaTime*rotateSpeed,0);

    if(_myControl.isGrounded || _isSwimming){
        airTime=0;
        _moveDirection = new Vector3((int)_strife,0,(int)_forward);

        //otra opoión es:
        //_moveDirection = Vector3.forward * Input.GetAxis("Avanzar");

        _moveDirection = _myTransform.TransformDirection(_moveDirection).normalized;
        _moveDirection*= walkSpeed;

        if(!_isAtacking){
            if(_forward != Forward.none){

                if(_isSwimming){
                    Swim();
                }else{

                if(_run){
                    _moveDirection*=runMultiplier;
                    Run();
                }else{
                Walk();
                }
            }
            }
            else if(_strife != Turn.none){
                Strafe();
            }
            else{
                if(_isSwimming){
                    IdleWater();
                }else{
                Idle();
                }
            }
            if(_jump){
                if(airTime<jumpTime){
                    _moveDirection.y+=jumpHeight;
                    Jump();
                    _jump=false;
                }
            }

        else{

            if((_collisionFlags & CollisionFlags.CollidedBelow) == 0){
                airTime+=Time.deltaTime;
                if(airTime>fallTime){
                    Fall();
                }
            }
        }
        }else{
            PlayMeleeAttack();
        }
    }

    if(!_isSwimming)
    _moveDirection.y-= gravity*Time.deltaTime;

    _collisionFlags =   _myControl.Move(_moveDirection * Time.deltaTime);
}
```

Después incluye las funciones para cambiar al animación, mediante el comando crossFade, de forma que el cambio no sea brusco.

2.3.2. Player Input

Este script es el encargado de recibir e interpretar los inputs del usuario, en la función de update comprobamos el input del usuario si coincide con uno definido, mediante el Messenger incluido en unity enviamos un mensaje pasando el valor a la función de advancedMovement correspondiente, y allí decidirá la acción a realizar, también comprobamos con que objeto colisiona, y modificamos la variable de nadar o salto según el objeto con el que colisiona.

```
//mover el jugador hacia adelante y atras
if(Input.GetButtonDown("Avanzar")){
    if(Input.GetAxis("Avanzar") >0){
        SendMessage("moveMeForward",AdvancedMovement.Forward.forward);//funciona como el broadcast, pero solo envia le mensaje a lso monodevelop que estan asociados a un mismo objeto

    }else{
        SendMessage("moveMeForward",AdvancedMovement.Forward.back);

    }

}
if(Input.GetButtonUp("Avanzar")){
    SendMessage("moveMeForward",AdvancedMovement.Forward.none);

}
```

2.4. Enemigos

Recopilación de los scripts que definen las características y comportamientos de los enemigos, incluyendo en que zonas aparecen, para evitar que todos los enemigos sean iguales sus características como fuerza o puntos de vida son generadas aleatoriamente dentro de un rango, así como el arma que llevan equipada algunos monstruos.

En estos scripts falta acabar de detallar el combate de los enemigos, de momento selecciona le ataque según distancia con el objetivo y se muestra la animación, pero no tiene efecto sobre el protagonista, otro elemento que no está desarrollado aún es el sistema de "*dropeo*" de los enemigos, también dejarán caer un objeto aleatorio de una lista para cada tipo de enemigo. Así mismo también faltaría desarrollar un scrip concreto para cada tipo de enemigo detallando más su comportamiento o características específicas, este script heredaría todas las características del script Mob, y reescribiría algunas partes para adaptarlo al comportamiento específico requerido.

2.4.1. Mob

Este script nos define las características básicas de los enemigos, hereda de baseCharacter, así que básicamente sobrescribe las características de todo personaje, y añade algunas concretas de los enemigos, al Iniciarse llama a la función de spawn, donde llamamos a las funciones encargadas de crearle los stats y el equipo que lleva, esto se realiza de forma aleatoria dentro de unos límites.

```
private void SetupStats(){
    for( int cnt = 0; cnt < Enum.GetValues(typeof(atributeNames)).Length; cnt++){
            GetPrimaryAttribute(cnt).BaseValue = UnityEngine.Random.Range(50,126);
        }
```

También se encarga de dar nombre al enemigo, y que este esté siempre orientado al jugador, todos los enemigos tienen un hijo que es un texto 3d, el cual en ser el enemigo marcado como objetivo, pasa a mostrarse y contiene el nombre del enemigo.

Para el equipo creamos un arma aleatoria, mediante el script ItemGenerator.

Un elemento interesante de este script es el uso de un DEBUGGER, preprocesor directives, es una parte de código que se emplea para hacer pruebas y eliminar errores, si está definido se compila, en caso den o estarlo ese código no se compila, simplemente se ignora.

Definición:

```
#define DEBUGGER        //definimos debbuguer(preprocesor Directives)
```

Usamos una variable para ponerlo o no desde el inspector

```
#if DEBUGGER
    public bool debugguer = true;
#else
    Debug.LogWarning("No esta definido el DEBUGGER"");
#endif
```

Parte del código que si no está definido debugger no se compilara, nos muestra por pantalla los stats de un enemigo

```
#if DEBUGGER        //la ventaja de este sistema es que si no esta definido nada de esto es compilado
    //muestra los stats si lo tenemos seleccionado y esta definido DEBUGGER
    void OnGUI(){
        if(debugguer){
            int LH = 25;
            for( int cnt = 0; cnt < Enum.GetValues(typeof(atributeNames)).Length; cnt++){
                GUI.Label(new Rect ( 10,10 +(cnt *LH + 5),300,LH),((atributeNames)cnt).ToString() + " : " + GetPrimaryAttribute(cnt).BaseValue);
            }
            for( int cnt = 0; cnt < Enum.GetValues(typeof(VitalName)).Length; cnt++)
                GUI.Label(new Rect ( 10,10 +(cnt *LH + 5)+(Enum.GetValues(typeof(atributeNames)).Length*LH),300,LH),((VitalName)cnt).ToString() + " : " + Get

            for( int cnt = 0; cnt < Enum.GetValues(typeof(SkillNames)).Length; cnt++)
                GUI.Label(new Rect ( 140,10 +(cnt *LH + 5),300,LH),((SkillNames)cnt).ToString() + " : " + GetSkill(cnt).AdjustedBaseValue);
        }


    }
#endif
```

2.4.2. Mob generator

Este script contiene una FSM, se encarga de crear e inicializar a los enemigos, contiene las array de los puntos de aparición y las mallas que pueden ir en esos puntos.

Al iniciar el script iniciamos la fsm, usando un bucle infinito, pero para que no itere todo el rato utilizamos un yield return 0; para que solo itere una vez cada frame.

```
while(true){
switch(_state){
    case State.Inicializado:
        Initialize();
        break;
        case State.Preparado:
        Setup();
        break;

    case State.SpawnMob:
        SpawnMob();
        break;

    }
    yield return 0; //le dice
}
```

Al inicializar comprobamos que existan puntos y mallas, en caso de no existir, o estar todos los puntos ocupados no generaremos más enemigos.

El setup se encarga de preparar las características del enemigo, y el spawnMob, coloca el enemigo en un punto libre, y lo orienta en el mundo.

```
private void SpawnMob(){
    //Debug.Log(" Dentro de Spawn");

    GameObject[] gos = AvalibleSpawnPoints();

    for ( int cnt =0; cnt< gos.Length; cnt++){
        GameObject go = Instantiate(mobPrefabs[Random.Range(0, mobPrefabs.Length)],
                                    gos[cnt].transform.position,
                                    Quaternion.identity) as GameObject;
        //go.transform.parent = gos.transform;
        go.transform.parent = gos[cnt].transform;

    }
```

Finalmente existe otro estado que es libre, que se utilizará para comprobar cuales están vacíos y volver a generar enemigos en esos puntos.

2.4.3. IA enemigos

Este script es el encargado de definir el comportamiento de los enemigos, Funciona mediante a activación de una corutina de una fsm, en su primer estado comprueba que tenga un colider de tipo esfera, guarda las coordenadas de su punto de origen, en su segundo estado, coloca el colider en el centro y le asigna el radio correspondiente, y lo hace trigger, el tercer estado es en el que se queda de forma habitual un enemigo, que es buscando, espera a que un gameobject con el tag personaje entre en su colider, en caso de entrar pasa al cuarto estado, un estado en el que decide si perseguirlo o no, y que acción realizar si lo persigue, como puede ser atacar de lejos, usar magia, o si está lo suficientemente cerca atacar físicamente, en caso de que el objetivo salga de su campo de acción, el trigger, regresa a su punto de origen.

2.4.4. Spawn points

Este script es un script pensado para el desarrollo, nos permite mostrar por pantalla donde están los puntos en los que se generaran enemigos, en los que dibuja un gizmo para destacarlos, es útil para que aquellos que trabajan creando el mapa puedan colocar rápidamente los puntos y comprobar sus posiciones.

```
public void OnDrawGizmos(){
    Gizmos.color = Color.red;
    Gizmos.DrawWireCube(transform.position, new Vector3(3,3,3));
```

2.5. NPC

Estos scripts son los encargados de definir las características y comportamientos de los personajes no jugables, tanto comerciantes como secundarios, estos scripts no se han desarrollado aún, así que explicare que debería contener y como estaría estructurados.

Algunos NPC tendría un script propio, como por ejemplo aquellos que no tiene un dialogo genérico, o han de realizar peticiones de misiones.

2.5.1. NPC

Este script nos dará las características básicas de los personajes no jugables, este script heredada de baseCharacter, de forma que sobrescribirá ese script para adaptarlo a las necesidades de los NPC, este script también determinara en qué punto del mundo aparecen, y posibles props o ítems que tengan.

Algunos NPC concretos, como puedan ser soldados, comerciantes y personajes relevantes en la historia tendrá un script propio con sus características, que a diferencia de los enemigos serán estándar, no creadas de forma aleatoria.

2.5.2. AI NPCs

El script referente a la inteligencia artificial determinara las zonas por las que se mueven, como interactuaran con el personaje u otras acciones necesarias, esto para los personajes secundarios, se crearía uno específico que heredara de este, para aquellos personajes que necesiten un comportamiento especifico, como comerciantes que en la interacción no solo tendrán que decir una frase, sino además crear un menú de venta y un sistema de comercio, o aquellos personajes que presenten misiones o sean parte de ellas, o los soldados que en determinadas ocasiones puedan entrar en modo combate.

2.6. Items

Esta es la colección de scripts que determinan las características de todos los objetos del juego, incluyendo la duración del objeto, su nombre, el valor del objeto y la rareza del mismo, después en los scripts que heredan se sobrescriben las características y se añaden las específicas para aquellos ítems.

También hay un script para generar objetos para los cofres, o al eliminar un enemigo.

2.6.1. Item

Este script es la base de todos los objetos, aquí se definen las características comunes a todos los objetos del juego, definimos el nombre del objeto, su valor, una enumeración de rareza del objeto, su durabilidad actual y total y el icono del objeto.

En este script se inicializan todos valores, y tenemos todos los setters y getters para las variables.

```
public Item( string name, int values, RarityTypes rare, int maxDur,int curDur){

_name = name;
    _value = values;
    _rarity = rare;
    _maxDur= maxDur;
    _curDur= curDur;

}



public string Name{
    get{return _name;}
    set{_name = value;}
}

public int Value{
    get{return _value;}
    set{_value = value;}
}

public RarityTypes rarity{
    get{return _rarity;}
    set{_rarity = value;}
}

public int MaxDurability{
    get{return _maxDur;}
    set{_maxDur = value;}
}

public int CurDurability{
        get{return _curDur;}
        set{_curDur = value;}

    }

// asignamos el icono

public Texture2D Icon{
    set{ _icon = value; }
    get{ return _icon;}
}
```

Esta clase también crea el toolTip, es decir la información del objeto, utiliza virtual, de forma que puede ser modificado por la clase que hereda, y devuelve la información del item.

```
public  virtual string ToolTip(){
    return Name + "\n" +
            "Valor "+ Value + "\n" +
            "Durabilidad "+ CurDurability + "/" + MaxDurability + "\n";


}
```

2.6.2. Item generator

Este script es el encargado de crear los ítems, tanto para cofres, como para equipar a enemigos o recompensas al jugador.

Para ello tiene una lista de tipos de arma, y de forma aleatoria elegimos un tipo, y con un switch entramos en el caso para este tipo de objeto, creamos un nuevo objeto y llamaos a la función que crea las características del objeto, después del switch asignamos al ítem los valores comunes a todos los objetos, y lo devuelve.

```
public static Item CreateItem( ItemType tipe ){
    Item item = new Item();

        switch( tipe){
    case ItemType.MeleeWeapons:
         item = CreateMeeleWeapon(); //hereda todos los que provenga de los padres weapon , buffitem item...
        break;
    case ItemType.RangedWeapons:
         item = CreateRangedWeapon(); //hereda todos los que provenga de los padres weapon , buffitem item...
        break;
    case ItemType.MobWeapons:
         item = CreateMobWeapon(); //hereda todos los que provenga de los padres weapon , buffitem item...
        break;
    case ItemType.Armor:
        item = CreateArmor(); //hereda todos los que provenga de los padres weapon , buffitem item...
        break;
    }


    // Propiedades comunes a todos los Items...
    item.Value = Random.Range(1,100);
            //Rareza del objeto
    item.rarity = RarityTypes.Comun;
            //Durabilidad maxima
    item.MaxDurability = Random.Range(50,61);
            //durabilidad actual
    item.CurDurability = item.MaxDurability;
            //

    return item;
}
```

En las funciones específicas para cada objeto añadimos las características para ese tipo de objeto, como el tipo de daño que realiza, el daño, el nombre, la distancia de ataque y el icono.

```
private static Weapon CreateMeeleWeapon(){
    Weapon meeleWeapon = new Weapon();

    string[] weaponNames = new string[]{
    "Espada",
     "Hacha",
     "Baston",
     "Martillo",
     "Espada_De_Fuego",
     "Vara_De_Hielo"
    };
    /* string[] weaponNames = new string[6];
        //rellenamos los valores del objeto

            //nombre
    meeleWeapon.Name = weaponNames[Random.Range(0,weaponNames.Length)];
            //daño maximo
    meeleWeapon.MaxDamage = Random.Range(5 ,11);
            //Varacion en % del daño maximo
    meeleWeapon.VarDmg = Random.Range(.2f,.76f);
            //Tipo de daño
    meeleWeapon.TypeOfDamage = DamageType.Cortante;
            //rango o alcance del arma
    meeleWeapon.MaxRange = BASE_MEELE_RANGE;

            //asignar el icono al arma

    meeleWeapon.Icon = Resources.Load(GameSettings2.MEELEE_WEAPONS_ICON_PATH + meeleWeapon.Name) as Texture2D;

    return meeleWeapon;

}
```

2.6.3. BuffItem

Este script hereda de Item, es el encargado de buscar a que item va el buff, añadirlo y eliminarlo, para guardar los buffs activos uso una hash Table, que nos asocia una clave y un valor, guardando el buff y su valor especifico, uso un try / catch para añadir los buffs a esta lista, evitando posibles excepciones y controlando las evitando que el programa falle.

```
public void AddBuff( baseStat stat, int mod){
    try {
    buffs.Add(stat.Name,mod);
    }
    catch(Exception e){
        Debug.LogWarning(e.ToString());

    }
}
```

2.6.4. Clothing

Este script hereda de BuffItem, sirve para asignar a que slot, parte del cuerpo, va cada pieza de ropa.

2.6.5. Armor

Este script hereda de clothing, y añade el nivel necesario para llevar esa armadura.

2.6.6. Jewlery

Este script hereda de Buffitem, sirve para asignar al slot que va ese ítem.

Estos últimos 3 script no están terminados, faltaría también sobrescribir los valores de buffitem, para tener en cuenta las modificaciones que nos da cada ítem al equiparlo.

2.6.7. Consumable

Este script hereda de buffitem, e el declaramos la variables de que vitales curara, catidad que recuperara, y duración del buff si es necesario.

Este script se encarga de buscar la característica vital donde se aplicara el buff, y aplicarle la cantidad de cura que nos da ese ítem, una vez pasado el tiempo designado elimina el buff.

2.6.8. Weapon

Este script hereda de buffitem, nos sirve para añadir las características del arma a las del personaje.

También sobrescribe el tooltip, añadiendo las características del arma, usando el override.

```
public override string ToolTip ()
{
    return base.ToolTip ()+ "\n" +
        MaxDamage * VarDmg + " - " + MaxDamage + "\n" +
            MaxRange;
}
```

2.7. Scripts de cámara

Para este proyecto decidí creer un script de control de cámara propio, así adaptarlo a las necesidades del proyecto y tener un mejor control del sistema de cámara, se ha tomado como base los scripts de cámara que viene un unity y se ha rescrito para crear un script propio mezclando distintos elementos de ambos scripts e incluyendo alguno nuevo.

2.7.1. Camera control

Es el script que controla la cámara, en este script definimos las variable de a que objeto apunta la cámara, la distancia mínima y máxima a la que puede estar la cámara, la velocidad vertical y horizontal de movimiento.

Al iniciarse el script, guardamos el transform de la cámara, para controlar su posición, y en caso de tener un objetivo, pasamos a la función setUp, donde colocamos a la cámara en la posición que le per toca y la orientamos hacia el target, el personaje, en el método Update comprobamos si se está presionado algún botón que suponga una acción para la cámara, o si se han soltado para resetear los valores de la cámara. Después del método Update, usamos un llamado LateUpdate, es un método que se llama siempre después de Update, y nos permite hacer el movimiento de cámara más controlado y fluido, ya que si está el movimiento en el propio update queda un movimiento muy errático, en este método calculamos el desplazamiento de la cámara y su velocidad, y llamamos a la función encargada de girar la cámara, que rotara alrededor del personaje.

```
// Otro emtodo update, este se lalma despues del update original cada frame, así nos aseguramos que la camara sigue bien al personaje, ya que este se mueve en update.
void LateUpdate(){

if(target != null){
        if(_rotateCameraKeyPressed){
            _x += Input.GetAxis("Boton rotar horisontal") * xSpeed * 0.02f;
            _y -= Input.GetAxis("Boton rotar vertical") * ySpeed * 0.02f;

               rotateCamera();
        }

        else if(_camButtonDown == true){

            _x += Input.GetAxis("Mouse X") * xSpeed * 0.02f;
            _y -= Input.GetAxis("Mouse Y") * ySpeed * 0.02f;

            //y = ClampAngle(y, yMinLimit, yMaxLimit);
               rotateCamera();

        }
        else{
            //_myTransform.position= new Vector3(target.position.x,target.position.y + height,target.position.z - minDist);
            //_myTransform.LookAt(target);



            // Calculate the current rotation angles
            float wantedRotationAngle = target.eulerAngles.y;
            float wantedHeight = target.position.y + height;

            float currentRotationAngle = _myTransform.eulerAngles.y;
            float currentHeight = _myTransform.position.y;

            // Damp the rotation around the y-axis
            currentRotationAngle = Mathf.LerpAngle (currentRotationAngle, wantedRotationAngle, rotationDamping * Time.deltaTime);

            // Damp the height
            currentHeight = Mathf.Lerp (currentHeight, wantedHeight, heightDamping * Time.deltaTime);

            // Convert the angle into a rotation
            Quaternion currentRotation = Quaternion.Euler (0, currentRotationAngle, 0);

            // Set the position of the camera on the x-z plane to:
            // distance meters behind the target
            _myTransform.position = target.position;
            _myTransform.position -= currentRotation *  Vector3.forward * minDist;

            // Set the height of the camera
            _myTransform.position = new Vector3(_myTransform.position.x, currentHeight, _myTransform.position.z);

            // Always look at the target
            _myTransform.LookAt (target);


        }


    }
    else{//si no existe el objetivo de la camara lo buscamos y añadimos
        GameObject go = GameObject.FindGameObjectWithTag(publicPlayerName);

        if(go==null){
            return;
        }
        target = go.transform;
    }
}
```

2.8. Creación del ciclo día noche

Estos script son los encargados del control del tiempo en el juego, para darle una sensación más realista y de un mundo vivo, estos scripts se encargan de mover las luces del juego para dar la sensación de paso del tiempo, y el movimiento de las sombras, modificar la iluminación global e intensidad de las luces, activar o desactivar una serie de luces y cambiar el skybox.

El script permite introducir la duración deseada para un día y las horas de salida y puesta del sol, y calcula las duraciones del día y la velocidad a la que se tiene que mover los elementos.

2.8.1. Game Time

Este script es el encargado de calcular la duración del día convirtiendo el input en segundos, y calculando cuantos grados se tiene que mover para emular el movimiento del sol.

Iniciamos el día a media noche, colocando el skybox correspondiente y sin mezclarlo, a medida que avance el día iremos mezclándolo con el otro skybox, para hacer esto usamos un shader especial.

En el método start terminamos de calcular el momento en que sucede cada evento, y creamos un array con todos los soles y lunas que se moverán por el cielo, y llamamos al setupLighting, donde ajustamos la intensidad de la luz para que coincida con la de medianoche.

```
void Start () {
    _tod = timeOfDay.Idle;
    _DayCicleInSeconds = dayCyckeInMinutes * MINUTE;

    RenderSettings.skybox.SetFloat("_Blend",0);
    _sunScript = new Sun[sun.Length];
    for(int cnt=0; cnt < sun.Length; cnt++){
        Sun temp = sun[cnt].GetComponent<Sun>();

        if (temp== null){
        Debug.LogWarning("No se encontro el script Sun");
            sun[cnt].gameObject.AddComponent<Sun>();

            temp = sun[cnt].GetComponent<Sun>();
        }

        _sunScript[cnt] = temp;
    }

    _timeOfDay = 0;

     degreeRotation = DEGREES PER SECOND * DAY /  DayCicleInSeconds ;//(dayCyckeInMinutes * MINUTE)

    //convertimos a segundos las entradas
    sunRise*=_DayCicleInSeconds;
    sunSets*=_DayCicleInSeconds;
    _noonTime = _DayCicleInSeconds/2;
    _morningLength = _noonTime-sunRise;
    _eveningLength= sunSets-_noonTime;
    morningLight*=_DayCicleInSeconds;
    nightLight*=_DayCicleInSeconds;

    //al iniciar ponemos las luces a la minima intensidad xq empezamos a media noche
    SetupLighting();

}
```

En el método update vamos incrementando el tiempo del día, usando time.deltaTime, de forma que no avance dependiendo del frame rate, si no del tiempo real, si supera la duración del día reseteamos el día, en caso de ser noche

enviamos un mensaje en broadcast para activar las luces, calculamos el ángulo del sol, y lo vamos girando segundo a segundo, también llamamos a blendSkybox, donde comprobamos en qué grado tenemos que mezclar los skybox para obtener el cielo de esa hora y le pasamos una variable del tipo boleana a la función encargada de controlar el brillo, donde variamos el ambientLight, y la intensidad de las luces según en qué momento del día estén.

```
private void AdjustLighting (bool brightness){
    float pos = 0 ;
    if(brightness){
         pos = ( timeOfDay - sunRise)/ morningLength;   //posicion del sol en el cielo durante la mñna

    }
    else{
         pos = (sunSets -  timeOfDay)/ eveningLength;   //posicion del sol en el cielo durante la tarde

    }

    RenderSettings.ambientLight= new Color(ambLigMin.r + ambLigMax.r * pos,
                                           ambLigMin.g + ambLigMax.g * pos,
                                           ambLigMin.b + ambLigMax.b * pos);
    for(int cnt=0; cnt < _sunScript.Length; cnt++){
            if(_sunScript[cnt].giveLight){
                //Debug.Log(pos);
                _sunScript[cnt].GetComponent<Light>().intensity = _sunScript[cnt].maxLightBrightness*pos;

            if(_sunScript[cnt].GetComponent<Light>().intensity <  _sunScript[cnt].minLightBrightness){
                _sunScript[cnt].GetComponent<Light>().intensity=_sunScript[cnt].minLightBrightness;
            }

            }
        }
}
```

2.8.2. Sun

Este script está unido a todos los gameobjects que actuaran como sol, es el que se utiliza para identificarlos y añadirlos a la lista de soles, únicamente contiene las variables de cada sol, su brillo máximo y mínimo, el flare, o si da luz o no.

2.8.3. Time Lighting

Este script se une a las luces que únicamente se encienden por la noche, al pasar el atardecer el script GameTime emite un mensaje en broadcast que recibe este script, y activa las luces, al amanecer el script GameTime emite otro mensaje en broadcast que sirve para apagar las luces.

Un script muy parecido a este se puede usar para controlar puertas de comercios, abiertas durante le día y cerradas en la noche.

```
public class TimeLighting : MonoBehaviour {

public void OnEnable(){
    Messenger<bool>.AddListener("Morning Light Time", OnToggleLight);
    }
public void OnDisable(){
    Messenger<bool>.RemoveListener("Morning Light Time", OnToggleLight);
    }
private void OnToggleLight(bool b){
        if(b){
        GetComponent<Light>().enabled=false;
        }
        else{
        GetComponent<Light>().enabled=true;
        }
}
}
```

2.9. GUI

Este script es el responsable de la creación de todos los elementos de la interfaz gráfica, desde el inventario, a la ventana del personaje, también contiene los estilos usados.

Este script recibe por broadcast la activación o desactivación de las ventanas que crea.

```
private void OnEnable(){
    Messenger.AddListener("DisplayLoot",DisplayLoot);
    Messenger.AddListener("closeChest", ClearWindow);
    Messenger.AddListener("toggleInventory", ToggleInventoryWindow);
    Messenger.AddListener("toggleCharacterWindow", ToggleCharacterWindow);
}
private void OnDisable(){
    Messenger.RemoveListener("DisplayLoot",DisplayLoot);
    Messenger.RemoveListener("closeChest", ClearWindow);
    Messenger.RemoveListener("toggleInventory", ToggleInventoryWindow);
    Messenger.RemoveListener("toggleCharacterWindow", ToggleCharacterWindow);
}
```

En el método OnGUI pintamos las ventanas si han sido llamadas, y les damos el estilo requerido.

```
void OnGUI(){

    GUI.skin = mySkin;

    if(_displayCharacterWindow)
     characterWindowRect = GUI.Window( CHARACTER WINDOW ID ,  characterWindowRect , CharacterWindow , "Personaje" ); //, "Inventario obj comun"


    if(_displayInventoryWindow)
    _inventoryWindowRect = GUI.Window( INVENTORY_WINDOW_ID , _inventoryWindowRect , InventoryWindow , "Inventario" , inventario);

    if(_displayLootWindow)
        _lootWindowRect = GUI.Window(LOOT_WINDOW_ID, new Rect(_offset, Screen.height -(_offset+LootWindowHeigth), Screen.width-(_offset*2),LootWindowHeigth), LootWindow,"Cofre",cofre);

    DisplayToolTip();
}
```

Para la ventana de loot al abrir los cofres creamos un scrollview, el cual llenamos con botones, uno para cada elemento, que al apretarlo guarda el ítem en el inventario, y lo elimina de la ventana de loot.

```
for( int cnt = 0; cnt < _chest.loot.Count; cnt++){
   if(GUI.Button(new Rect(5+(buttonWidth *cnt),_offset, buttonWidth,buttonHeight),new GUIContent(_chest.loot[cnt].Icon , _chest.loot[cnt].ToolTip()), nombreInventarioComun)){
      //añadimos el objeto al invetario
      PlayerCharacter2.Instance.Inventory.Add(_chest.loot[cnt]);
      //eliminamos el item
      _chest.loot.RemoveAt(cnt);
   }
```

Para el inventario, creamos una matriz de botones, a los que poblamos con el contenido de nuestro inventario, usamos un contador de tiempo para comprobar si se efectúa o no un doble clic, en caso de ser un doble clic, comprobamos el tipo de elemento mediante un typeof, y según el tipo de elemento decidimos que acción realizamos, en el caso de la armadura usamos un switch para comprobar a que slot pertenece y si está ocupado o no.

```
//typeof nos sirve para ver o coparar un typo de elemento
if( typeof(Weapon)== PlayerCharacter2.Instance.Inventory[cnt].GetType()){
    if(PlayerCharacter2.Instance.EquipedWeapon == null){
        PlayerCharacter2.Instance.EquipedWeapon = PlayerCharacter2.Instance.Inventory[cnt];
        PlayerCharacter2.Instance.Inventory.RemoveAt(cnt);
    }
    else{
        Item temp = PlayerCharacter2.Instance.EquipedWeapon;
        PlayerCharacter2.Instance.EquipedWeapon = PlayerCharacter2.Instance.Inventory[cnt];
        PlayerCharacter2.Instance.Inventory[cnt] = temp;
    }
}
```

Para la ventana de personaje creamos un toolbar, y según en qué opción cliquemos, entraremos en una u otra opción de un switch, que definirá a que función llamamos, y esta será la encargada de crear el contenido, para el equipo se crean botones, una para cada slot, mostrando el objeto equipado, en los atributos se muestran los tributos mediante labels, de la misma forma que los skills son mostrados.

```
public void CharacterWindow (int id){

     characterPanel= GUI.Toolbar(new Rect(5,25,  characterWindowRect.width-10, 50), characterPanel, panelNames); //,estilo!!

    switch(_characterPanel){
    case 0:
        DisplayEquipment();
        break;
    case 1:
        DisplayAttributes();
        break;
    case 2:
        DisplaySkills();
        break;
    }

    GUI.DragWindow();
}

public void ToggleCharacterWindow(){
    _displayCharacterWindow = !_displayCharacterWindow;
}

private void DisplayEquipment(){
    //Debug.Log("Display equipment");
    if(PlayerCharacter2.Instance.EquipedWeapon == null){
        GUI.Label(new Rect ( 15, 120,40,40),"Arma","Vacio");
    }
    else{
        if(GUI.Button(new Rect ( 15, 120,40,40),new GUIContent( PlayerCharacter2.Instance.EquipedWeapon.Icon, PlayerCharacter2.Instance.EquipedWeapon.ToolTip()))){
            PlayerCharacter2.Instance.Inventory.Add(PlayerCharacter2.Instance.EquipedWeapon);
            PlayerCharacter2.Instance.EquipedWeapon = null;
        }
    }
    SetToolTip();
}
```

Este script también es el encargado de crear el tooltip, un recuadro que muestra la información de un objeto, en los scrips de ítems hemos visto como obtenía la información, aquí vemos como se dibuja.

```
private void SetToolTip(){
    if(Event.current.type == EventType.Repaint && GUI.tooltip != _toolTip){
        if(_toolTip != ""){
            _toolTip = "";
        }
        if(GUI.tooltip != ""){
            _toolTip = GUI.tooltip;
        }
    }

}

private void DisplayToolTip(){
    if(_toolTip !=""){
        GUI.Box ( new Rect (Screen.width/2 -100, 10, 200,100), _toolTip);
    }
}
```

2.10. Spell system scripts

Son el conjunto de scripts que definen el comportamiento de los hechizos, en este juego existen tres tipos básicos de hechizo, aquellos que aportan algún buff al jugador, los hechizo de un único objetivo y los hechizo de área.

Estos scripts están todavía en desarrollo y no están implementados en el juego aún, de momento existen los siguientes scripts:

2.10.1. Ispell

Este script contiene una interfaz donde definimos los setters y getters básicos para el sistema de hechizos, como son un string que define el nombre, el gameObject del efecto, a rareza del hechizo, una booleana que define si vemos o no al enemigo, la descripción, o el tiempo entre lanzamientos.

2.10.2. Spell

Este script hereda de Ispell, implementa los setter y getters para definir el hechizo concreto, creando unos datos base y luego modificándolos:

```
public Spell(){
    //falta GameObject Effect


     Name = "Sin Nombre";
     Rarity = RarityTypes.Comun;
     LineOfSight = true;
     Description ="Sin descripcion";

     BaseCoolDownTime = 2.0f;
     CoolDownVariance = 0.2f;
     CoolDownTimmer  = 0;
     Ready = true;

}
```

Al no estar acabado lo que hago es lanzar una excepción diciendo que no está implementado:

```
#region Ispell implementation
public void Cast ()
{
    throw new System.NotImplementedException ();
}
```

2.10.3. SpellGenerator

Esta es el script encargado de crear el hechizo en sí, crea un objeto del tipo Spell, y mira a que categoría pertenece, entonces le añadirá las características que corresponda y lo devolverá.

```
public Spell CreateSpell(){

    if(spell is Buff){
        Debug.Log("Buff");
    }
    //antes AoE xq si no entraria bolt  y no seguiria comprobando ya que AoE implemnte bolt
    else if(spell is AoE){
        Debug.Log("AoE");
    }
    else if(spell is Bolt){
        Debug.Log("Bolt");
    }


    return spell;
}
```

2.10.4. Buff

Hereda de Spell y de Ibuff, en estas clases definimos los comportamientos específicos de los hechizos que nos aportan algún buff, como su valor o su duración.

2.10.5. Bolt

Hereda de Spell y de IBolt, estas clases sirven para añadir los comportamientos específicos a los hechizo con un único objetivo, indican su daño, la variación de este y el rango de alcance que tienen.

2.10.6. AoE

Hereda de Bolt y de IAoE, estas clases sirven para añadir los comportamientos específicos a los hechizo con un efecto de área, indican su rango de área, el daño, la variación y el número máximo de objetivos.

2.11. Enums

Enum es una palabra clave que nos permite declarar una enumeración, es decir un grupo de nombres como valores, facilitándonos la programación y haciéndolo más comprensible, en este motor se utiliza para indicar habilidades, atributos, slots de equipo, rareza para ítems, u otras enumeraciones.

El motivo de su uso es que permite modificar estos valores adaptándolos al juego que se quiera crear haciendo que no se tenga que modificar demasiado código, por ejemplo si se quiere añadir una nueva habilidad basta con escribirla en su enum, ya que el resto de scripts estás pensados para ser escalables y no les afectaría, o añadir un nuevo color de pelo.

Estos Enum originalmente en la etapa de debuging estaban dentro de otros scripts, pero finalmente decidí moverlos crear los enums por separado para facilitar el uso del motor, y evitar que se tengan que abrir los demás scripts.

2.12. Otros scripts

2.12.1. Targeting

Este es el script encargado de permitir al usuario apuntar a un enemigo, contiene una lista con todos los enemigos ordenados por distancia, y el transform del seleccionado al iniciar creamos una lista con los transforms de los enemigos, creamos un array conteneido todos lso gameObject que tengan el tag “enemigo” y lo añadimos a la lista, después ordenamos según la distancia los enemigos.

```
private void enemigoDistancia(){
    targets.Sort(delegate(Transform t1, Transform t2){
        return Vector3.Distance(t1.position, myTransform.position).CompareTo(Vector3.Distance(t2.position,myTransform.position)) ;
    });
```

Al apretar la tecla tabulador, comprobamos si la lista de enemigos contiene información o no, si no la tiene volvemos a rellenarla, si la tiene comprueba las distancias y la reordena, cada vez que apretamos tabulador avanzamos una posición en la lista, al seleccionar un enemigo, buscamos la propiedad del nombre, y activamos el renderizado del texto 3d que tiene el personaje añadiendo su nombre, también mostraremos su salud, y se nos permite atacarlo, y al deseleccionarlo ocultamos el nombre, y le retiramos la selección.

```
private void selected(){
    Transform name =selectedTarget.FindChild("Name");
    if(name == null){
        Debug.LogError("no se pudo encontrar el nombre de"+ selectedTarget.name);
        return;
    }

    name.GetComponent<MeshRenderer>().enabled =true;
    name.GetComponent<TextMesh>().text = selectedTarget.GetComponent<Mob>().name;
    selectedTarget.GetComponent<Mob>().DisplayHealth();
    selectedTarget.GetComponent<Mob>().isSelected=true;

    //Messenger<bool>.Broadcast("show mob vital bar",true );

}
```

2.12.2. Portals

Este script se emplea en portales, son objetos que sirven para transportarse a otro punto del mapa, cada portal tien esta script, donde se le indica mediante un empty object el destino, en el start comprobamos si existe la destinación, en caso de no existir busca un gameobject default, que es el drop zone de la ciudad, en punto donde inicia el jugador, y se le enviaría allí.

Los portales cuentan con un trigger, al entrar comprobamos que sea el gameObject con el tag “Player”, entonces mediante las coordenadas del transform del jugador lo llevamos a la coordenadas del empty object.

```
public void OnTriggerEnter( Collider other){
    if(other.transform.CompareTag ("Player")){
        Debug.Log("player enter");
        other.transform.position = destnation.transform.position;
    }
```

2.12.3. Main menú

Este script es el encargado de crear el menú principal del juego, contiene las variables con los nombres de los niveles, una booleana para comprobar si existe personaje, y el tanto por ciento cargado del nivel.

Al iniciarse comprueba si existen elementos guardados, si existen comprueba que la versión guardada sea la misma que tiene el juego, en caso de fallar alguna comprobación entra directamente a la creación de personaje, si no falla ninguna comprueba que exista un nombre, si todo existe procede a la creación del menú principal.

Para crear el menú usamos le método OnGUI, donde creamos dos botones, y al clicarlos nos dirigen al nivel correspondiente, y creamos una barra que nos indica el % de carga de ese nivel, indicándolo mediante una label y la barra que aumenta de tamaño en aumentar el % de carga.

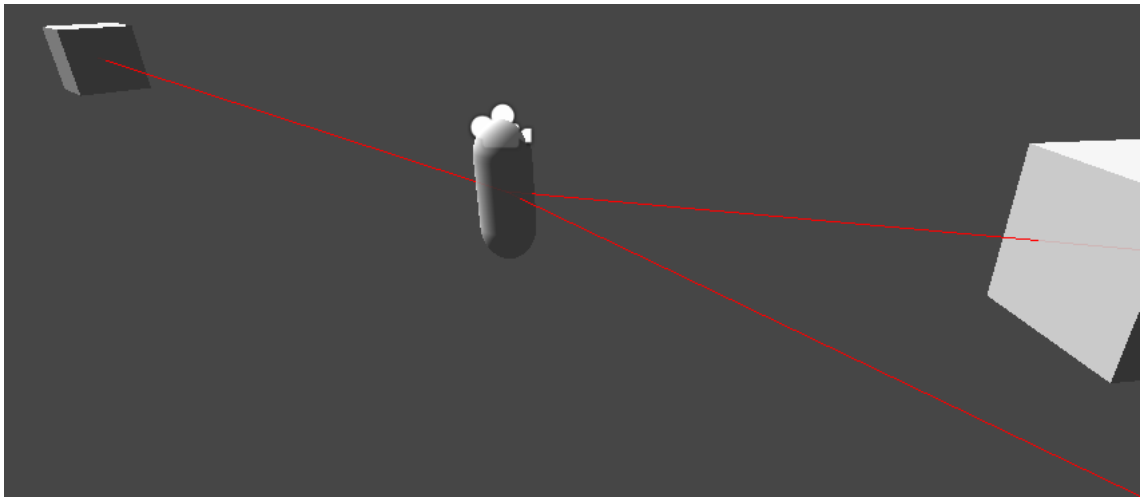
```
GUI.Label(new Rect(Screen.width/2 -25, Screen.height -55, 100, 20), (_PercentLoaded * 100) +" %");
GUI.Box(new Rect(0, Screen.height -20, Screen.width * _PercentLoaded, 20),"");
```

En el update comprobamos si hay un nivel en carga, si no lo hay hacemos un return, pero si lo hay comprobamos el nivel de carga, y una vez completamente cargado vamos a ese nivel.

```
void Update () {
    if(_levelToLoad=="")
        return;

    if(Application.GetStreamProgressForLevel( levelToLoad) == 1){//% de carga
        //Debug.Log("nivel cargado!");
        _PercentLoaded = 1;

        if(Application.CanStreamedLevelBeLoaded(_levelToLoad)){
            Application.LoadLevel(_levelToLoad);
        }

    }
    else{
        _PercentLoaded = Application.GetStreamProgressForLevel(_levelToLoad);
    }

}
```

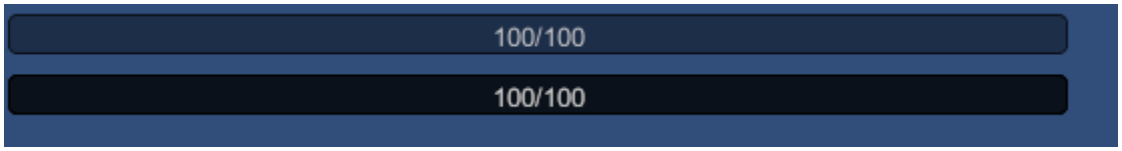
2.12.4. Scripts antiguos

Durante el proceso de desarrollo del motor muchos de los scripts han sido modificados a substituidos por otros nuevos, ya que no eran muy eficientes, o se podían unir a otros scripts, o por la falta de conocimiento no acababan de funcionar de una forma correcta, un ejemplo de esto es el script de apuntar, en una de las primeras fases de experimentación desarrolle un script que creaba una línea entre el enemigo y el personaje y el enemigo le iba siguiendo, pero ya incluía la aparición del nombre del enemigo, este script fue creado para aprender, finalmente fue substituido por un script más complejo que encaja mejor con su propósito en el juego.



Captura del juego en sus primeras versiones donde vemos como se apuntaba.

A medida que el juego iba evolucionando aparecían nuevas necesidades, y había que rehacer algunos scripts para adaptarse a la nuevas necesidades, o para intentar optimizar el juego al máximo. Algunos de estos scripts han dejado de ser validos por cambios en otros scripts, y aún no han sido adaptados al nuevo juego, un ejemplo de esto es el script encargado de mostrar la vida.



Captura del antiguo script de vida.

2.13. Utiles

En el desarrollo del juego se han utilizado una serie de scripts descargados de UnityWiki, de uso libre, como utilidades o herramientas para mejorar el juego.

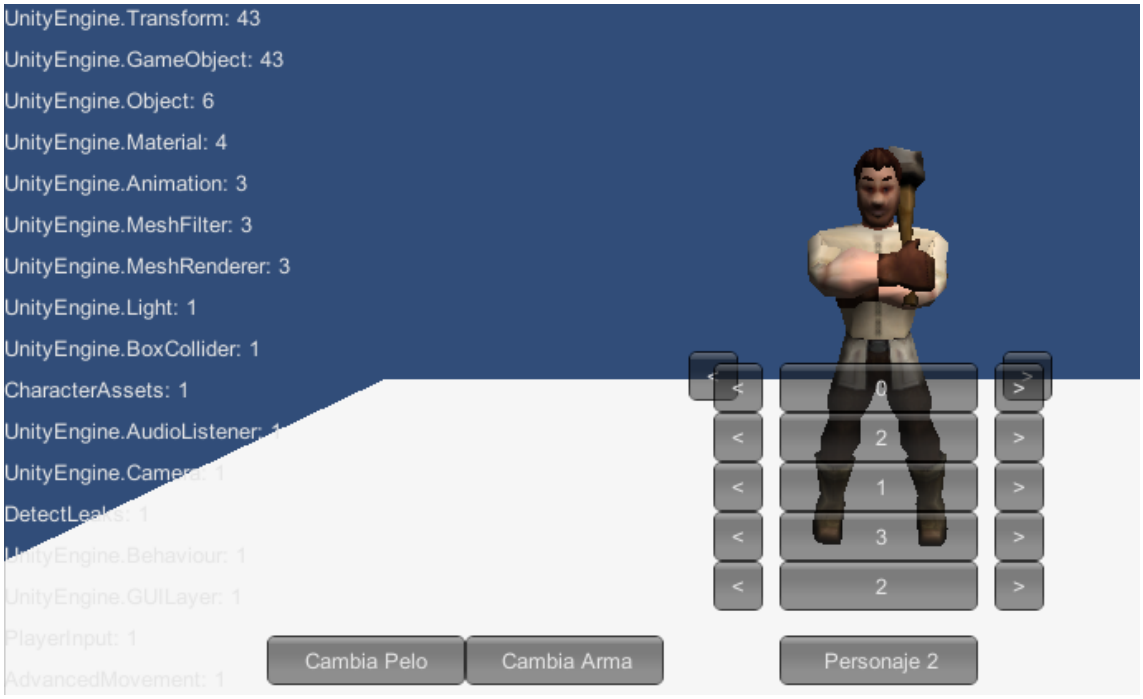
2.13.1. CS Messenger Extended

Este script con autoría de Magnus Wolffelt nos permite solucionar un problema existente en el Messenger incluido en Unity, el Messenger interno únicamente permite comunicar scripts unidos a un mismo gameObject, este script nos permite emitir mensajes en broadcast, y cualquier gameObject puede ser el receptor de dicho mensaje.

Este script ha resultado muy útil y me ha ahorrado mucho tiempo y facilitado mucho algunas tareas, se emplea en muchos scripts del juego.

2.13.2. DetectLeacks

Este script, de autoría de Joachim Ante y Berenger, Juha, nos muestra por pantalla el número de objetos de cada tipo, mallas, texturas, gameObjects, clips de audio… ha sido un script muy útil a la hora de hacer debug, y sobre todo para darme cuenta de cómo trabaja el programa y empezar a optimizar, evitando que los recursos estén todo el rato cargados, o asegurándome de eliminarlos.

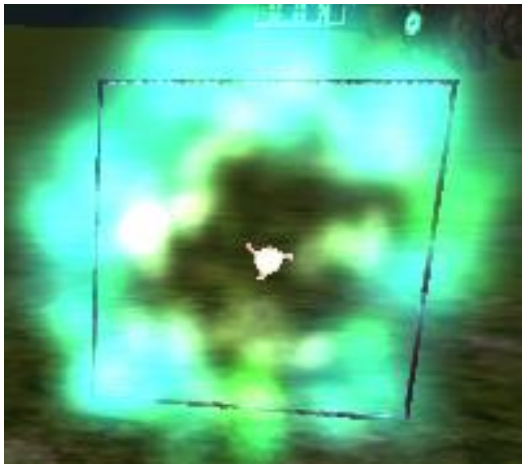


Captura del antiguo editor de personajes, en el lateral podemos ver el script DetectLeacks activo

2.13.3. Animate Tiled texture

Este script de autoría de Joachim Ante, sirve para animar spritesheets, nos permite decir le número de imágenes por segundo, el número de filas y columnas y se encarga de animar la textura.

En este proyecto lo he utilizado para animar los portales.



Captura del portal que contiene la animación de un sprite sheet

2.14. Shaders

Los shaders se encargan de definir los comportamientos de las texturas, realizan los efectos de las texturas como pueda ser transparencia, reflexión o refracción o rugosidad.

2.14.1. SkyBoxBlended

Este shader de autoría de Aras Pranckevicius, permite mezclar dos skybox diferentes mediante un slider que marca la transición, lo utilizo para mejorar el efecto del ciclo día y noche, a la vez que se mueven las luces y varía a la intensidad variar el cielo.

Imagen cielo noche

Imagen amanecer

Imagen medio día

Imagen anochecer.

2.15. Assets

Para el desarrollo de este proyecto he descargado una serie de assets de la tienda de unity, o utilizo algunos ya preincorporados.

2.15.1. Terreno y terraintoolkit

Este es un asset externo que nos mejora la herramienta de generación de terrenos de Unity, aunque no es compatible con las últimas versiones de unity, nos permite generar terrenos según el tipo que queramos, montañoso, zonas planas, costa, etc.

Indicándole el tamaño del área, y las texturas que utilizara según la altura, además incluye nuevas herramientas para el terreno.

2.15.2. Flares

Este asset incluye nuevos tipos de flare extendiendo a los originales de unity, estos flares son los que se han utilizado para recrear los soles y lunas.

2.15.3. Particles

Este asset es una extensión de las partículas incluidas en unity, que se incorporan a shuriken, el creador de partículas de unity.

2.15.4. Tree creator

Este es un asset pre incorporado en unity, nos permite dibujar árboles que unity crea en 3D, eligiendo la forma, el tipo de ramas y el tipo de hoja.

Arboles creados mediante el asset tree creator.

2.16. Sonido

Este script es el encargado de controlar la música del juego, para facilitar su uso, las canciones se cargan desde el inspector, de forma que sea más sencillo cambiarlas, para controlar que clip de audio utiliza en cada momento me ayudo de unas booleanas, al iniciar el juego hago que se empiece a reproducir la música de la zona de exploración, para cambiar la música he usado dos métodos distintos, el primero de ellos es el de combate, utilizo el extended Messenger, en el script de AI en el momento en que el enemigo decide atacar mediante broadcast envió un mensaje, que recibe el script que controla el audio, donde llama a una función que con el valor modificado de la booleana, cambia el clip que reproduce.

```
    _
    AC._battleMusic = false;
    Messenger.Broadcast("MusicChange");
}


     void OnEnable()
 {
     Messenger.AddListener("MusicChange", MusicBox);
 }

 void OnDisable()
 {
     Messenger.RemoveListener("MusicChange", MusicBox);
 }

void MusicBox(){

    if(_battleMusic){

        audio.clip = fight;
        audio.Play();
    }else if(_TownMusic){
        audio.clip = town;
        audio.Play();
    }
    else{
        audio.clip = field;
        audio.Play();
    }
}
```

Código empleado para la música de batalla.

En cambio para la ciudad empleo un trigger, al entrar en la ciudad, llama a una función mediante una instancia del script, donde se cambia el valor de una booleana y llama a la función encargada de controlar el audio.

```
void OnTriggerEnter(Collider other) {
    if(other.tag == "Player"){
        AC.Town();
    }
}

public void Town(){

    _TownMusic= !_TownMusic;
    MusicBox();
    Debug.Log(_TownMusic + " musica ciudad ");
}
```

Código empleado para la música de la ciudad.

2.17. Sistema de diálogos

Este script es el que se encarga de los diálogos, cada NPC tendrá uno propio, para el script creamos dos arrays de strings, una para el dialogo del jugador y otra para el del NPC, y una serie de booleanas para comprobar si se tiene o no que mostrar.

Para crear los diálogos utilizamos el método OnGUI, creamos un área de forma que se nos cree todo dentro del área, para el dialogo del NPC usamos labels, y para las respuestas del usuario buttons, según la respuesta modificamos el valor de unas booleanas y así cambiamos le dialogo o lo terminamos.

```
void OnGUI(){
    GUILayout.BeginArea( new Rect(Screen.width/3,(Screen.height - Screen.height/3),400,400));
    if(DisplayDialog && !ActivateQuests){



        GUILayout.Label(questions[0]);
        GUILayout.Label(questions[1]);
        if(GUILayout.Button(answerButtons[0])){
            ActivateQuests=true;
            HasCompletedQuest=false;
            DisplayDialog = false;
        }

        if(GUILayout.Button(answerButtons[1])){
            DisplayDialog = false;
        }
    }

    if(DisplayDialog && ActivateQuests && HasCompletedQuest){
        GUILayout.Label(questions[2]);
        if(GUILayout.Button(answerButtons[2])){
            DisplayDialog=false;


        }
    }

    GUILayout.EndArea();
    /*
    if(ActivateQuests == true){
        //GUI.DrawTexture(new Rect (10,10,200,150),texture1);
    }*/
}
```

Código para crear la interfaz del dialogo

Para activar e iniciar el dialogo uso un trigger, colocado frente al personaje con el que dialogaras, de forma que solo se produce la conversación estando de cara al personaje.

2.18. Sistema de Quest

Estos scripts será la colección de misiones del juego, cada misión tendrá su propio script donde se especificara el objetivo y npcs involucrados, así como la recompensa, este script se comunica con el de dialogo de NPC, de forma que al cumplir la misión el NPC cambie su dialogo y nos de la recompensa.

Una vez completada la misión destruiremos el gameObject que tenía la misión, de forma que evitemos un posible bug de misión con repeticiones infinitas.

2.19. Video

Este script es el encargado de controlar las escenas cinemáticas del juego, para ello creo una interface grafica de usuario con el método OnGUI, donde dibujo una textura, y le hago iniciar la reproducción, el start inicio una corrutina, que una vez pasado el tiempo de duración del vídeo, llama a una función que elimina la textura creada y cambia la escena.

```
[RequireComponent(typeof(AudioSource))]
public class Video : MonoBehaviour {

    public MovieTexture movText;

    // Use this for initialization
    void Start () {
        StartCoroutine(Wait(10));
        //Debug.Log("duracion: "+ movText.duration );
    }

    // Update is called once per frame
    void Update () {

    }

    void OnGUI(){
        GUI.DrawTexture(new Rect(0,0,Screen.width,Screen.height),movText);
        movText.Play();
        audio.Play();

    }

    private IEnumerator Wait(float duration)
    {
        yield return new WaitForSeconds(duration);
        Debug.Log("Fin");
        Application.LoadLevel("Level1");
    }
}
```

Script de control del video inicial

3. Conclusiones y agradecimientos

3.1. Conclusiones

Al finalizar este trabajo llego a las siguientes conclusiones:

- He descubierto las fases del desarrollo de un videojuego, visto las complicaciones y buscado soluciones para cada una de ellas, siendo ahora capaz de plantearme un proyecto real con unas bases mínimas para saber el tiempo que se necesita para cada fase, que recursos hay que dedicar, y que fases es recomendable haber finalizado antes de empezar otra.
- Actualmente estoy finalizando el desarrollo de la conceptualización del juego, y he aprendido la importancia de la conceptualización antes de iniciar el desarrollo del motor, planear que script son necesarios, que partes tiene que ser comunes, que tiene que contener y como funcionaran.
- Inicialmente plantee un calendario que una vez empezado el desarrollo y visto el auténtico volumen de trabajo he visto que es imposible de cumplir, hoy en día soy capaz de plantear un calendario más realista gracias al conocimiento que he obtenido en este proyecto, a la vez que me permite reducir algunos tiempos.
- He aprendido a presupuestar de forma aproximada un proyecto y tener en mente los costes que puede generar el desarrollo de un juego.
- He aprendido un nuevo lenguaje de programación y obtenido una soltura en él.
- He aumentado mis conocimientos en Unity y aprendido a buscar recursos y soluciones, así como a usar assets externos en proyectos.
- He comprendido mejor el flujo de trabajo necesario en el desarrollo de videojuegos.
- Como organizar de forma óptima los recursos de un proyecto, y como organizarlos en unity para optimizar su funcionamiento.
- Mejorado la comprensión del mercado y la industria de los videojuegos.

3.2. Agradecimientos

Para finalizar quiero agradecer a mi tutor por su tiempo y dedicación a este proyecto, así como a las personas que me han ayudado de forma directa, a través de las respuestas de Unity, como a los creadores de los assets o scripts externos empelados en el desarrollo de este motor.

4. Bibliografía y web grafía

Bibliografía

Ante, J. (s.f.). *wiki unity3D - Animating Tiled texture*. Obtenido de http://wiki.unity3d.com/index.php?title=Animating_Tiled_texture

C#-Wikipedia. (s.f.). Obtenido de http://es.wikipedia.org/wiki/C_Sharp

crigger, L. (s.f.). *1up.com - Chasing D&D: A History of RPGs*. Obtenido de http://www.1up.com/features/chasing-history-rpgs

educación, M. d. (s.f.). *Beneficios de lso juegos de rol en la educación*. Obtenido de http://ntic.educacion.es/w3/recursos2/estudiantes/jovenes/op_11.htm#03

English, S. (7 de Jan de 2008). *nzgamer.com - Japanese and Western RPGs - The Differences* . Obtenido de http://nzgamer.com/features/552/japanese-and-western-rpgs-the-differences.html

Joachim Ante, B. J. (s.f.). *Wiki Unity3D - DetectLeaks*. Obtenido de http://wiki.unity3d.com/index.php?title=DetectLeaks

Kaiser, R. (1 de March de 2012). *Gamasutra - How Mass Effect challenged my definition of 'RPG'*. Obtenido de http://www.gamasutra.com/view/news/129583/Opinion_How_Mass_Effect_challenged_my_definition_of_RPG.php

Microsoft. (2013). *Guía de programación de C#*. Obtenido de http://msdn.microsoft.com/es-es/library/67ef8sbd.aspx

Microsoft. (s.f.). *c# tutorials*. Obtenido de http://msdn.microsoft.com/en-us/library/aa288436(v=vs.71).aspx

Pablo Giménez, m. d. (s.f.). *Los Juegos de Rol: Hacia una propuesta pedagógica*. Obtenido de http://dreamers.com/defensadelrol/articulos/propuesta.htm

point, T. (s.f.). *c# programing*. Obtenido de http://www.tutorialspoint.com/csharp/csharp_overview.htm

Pranckevicius, A. (s.f.). *wiki Unity 3D - SkyboxBlended*. Obtenido de http://wiki.unity3d.com/index.php?title=SkyboxBlended

rol, W. p. (s.f.). *Los 7 mitos sobre los jeugos de rol*. Obtenido de http://dreamers.com/defensadelrol/articulos/7mitos.htm

rol, W. p. (s.f.). *Los juegos de rol -que son?* Obtenido de http://dreamers.com/defensadelrol/articulos/cnice.htm

Ultima III: Exodus. (s.f.). Obtenido de http://es.wikipedia.org/wiki/Ultima_III:_Exodus

Unity 3D answers. (s.f.). Obtenido de http://answers.unity3d.com/questions/61926/what-is-an-enum-.html

Unity 3D- forums. (s.f.). Obtenido de http://forum.unity3d.com/forum.php?s=6f6e6e38055eef3ca11b8d4711f309ab

Unity 3D-Spain. (s.f.). Obtenido de http://spanish.unity3d.com/

Wiki - Anexo:Cronología de los juegos de rol. (s.f.). Obtenido de http://es.wikipedia.org/wiki/Anexo:Cronolog%C3%ADa_de_los_juegos_de_rol

Wikipedia - Anexo Diferencias culturales juegos de rol. (s.f.). Obtenido de http://es.wikipedia.org/wiki/Anexo:Diferencias_culturales_en_videojuegos_de_rol

Wikipedia - Dungeons & Dragons. (s.f.). Obtenido de http://es.wikipedia.org/wiki/Dungeons_%26_Dragons

Wikipedia - Juegos de rol. (s.f.). Obtenido de http://es.wikipedia.org/wiki/Videojuego_de_rol

Wikipedia - William A. Gamson. (s.f.). Obtenido de http://en.wikipedia.org/wiki/William_A._Gamson

Wikipedia- Juegos de rol. (s.f.). Obtenido de http://es.wikipedia.org/wiki/Juego_de_rol

Wiki-Unity. (s.f.). Obtenido de http://wiki.unity3d.com/index.php/Main_Page

Wilson, J. (s.f.). *1up.com - Chasing D&D: A History of RPGs*. Obtenido de http://www.1up.com/features/future-single-player-rpgs

Wipipedia - Anexo:videojeugos de rol. (s.f.). Obtenido de http://es.wikipedia.org/wiki/Anexo:Videojuegos_de_rol

Wolffelt, M. (s.f.). *Wiki unity 3D -CSharpMessenger Extended*. Obtenido de http://wiki.unity3d.com/index.php?title=CSharpMessenger_Extended

5. Anexos

5.1. Análisis completos de referentes

Ragnarok Odyssey

- Plataforma: Psvita
- Género: ARPG - MMORPG
- Fecha de lanzamiento: 20/02/2013(EU)
- Desarrolladores : Game Arts
- Publishers: GungHo Online Entertainment

Sinopsis

El juego nos traslada a un reino asediado por una serie de criaturas mitológicas, en este mundo han surgido diferentes guilds para luchar contra esta amenaza, el protagonista se alistara en una de ella, y se dedicara a cumplir encargos.

MDA

Mecánicas: Avanzar, retroceder, moverse hacia los lados, girar la cámara, atacar, uso de pociones, comprar, aceptar misiones, entrar a modo online, cambiar la clase, cambiar el aspecto físico, recolectar ítems, mejorar el equipo, crear nuevo equipo, hablar con NPC.

Dinámicas: Esquivar, combos de ataques, cazar monstruos para obtener recursos, estrategia ante monstruos.

Estética: motivaciones, el jugador es motivado a mejorar sus habilidades mediante una serie de retos, con una dificultad graduada, recompensas, cada enemigo derrotado libera una serie de ítems, algunos de ellos son materiales para crear mejores equipos, al completar cada capítulo se nos recompensa subiendo un nivel. También propone retos como intentar superar todos los jefes del juego tu solo.

Flujo de partida

El flujo de partida la puede elegir el usuario, ya que las misiones están ordenadas pro dificultad, al iniciar un capítulo tenemos unas misiones de menor dificultad que las ultimas del capítulo, tiene un código de colores para diferenciarlas, también puede elegir jugar on-line y recibir la ayuda de un grupo de jugadores. La dificultad incremente de forma gradual, según el capítulo en el que nos encontremos, para evitar la ansiedad disponemos de misiones secundarias o podemos repetir misiones anteriores con lo que mejoramos nuestras habilidades y equipo, ayudando a balancear la dificultad y nuestras habilidades, haciendo que el jugador no se quede estancado si no puede superar una misión.

Los objetivos están muy definidos, al inicio de la misión se explica el objetivo de la misma, durante la misión el objetivo aparece en la parte superior de la pantalla.

Target

El target de este juego son aquellos jugadores a los que les gusta la acción, actuar y poder superar retos claros y simples, utilizando la clasificación de Bartle vemos que estos adjetivos son los que corresponden a los Killers.

Es un juego para aquellos que les gustan los desafíos y la amenazas, personas persistentes en cumplir sus objetivos y ansiosas, esto nos da dos rasgos en el *Big Five Personality Traits*, eficiente y neurótico.

Esto sirve para explicar por qué el juego ha triunfado tanto entre adolescentes, ya que están en una edad que buscan emociones fuertes, destacar sobre lo demás y superarse.

Críticas al juego

Vandal.net

El juego es divertido y adictivo, falla el comportamiento de la cámara en zonas cerradas, y peca de ser "machaca botones". Destacan la posibilidad de jugar online con hasta 4 jugadores, la opción de poder cambiar la clase y la personalización en cualquier momento, abriendo nuevas formas de juegos o variar según el tipo de misión y nuestra forma de jugar. Valoran positivamente la inclusión del uso de la pantalla táctil "con sentido y nada forzada". Añaden que la inteligencia artificial de los enemigos es muy pobre y que gráficamente no está pulido, hay escenarios repetitivos y las texturas están poco trabajadas.

Hobbyconsolas.com

Titulo muy similar a Monster hunter de Capcom, historia poco elaborada, y el jugador puede no seguirla, ya que "solo tenemos que ir haciendo misiones". Destacan la falta de algunas opciones en la personalización, como el cambio de ropa. Les gusta la opción de jugar On-line y destacan mucho la larga duración que tiene el juego, el sistema de batalla es muy ágil y permisivo, pero el juego cae rápidamente en la repetición. Los escenarios y los enemigos se repiten, y la novedades son introducidas muy poco a poco, y algunas en un orden no adecuado. Todas las misiones tiene como objetivo cazar algún monstruo, no existen otro tipo de misiones como conseguir recursos, o zonas de plataformas como en otro juegos similares, esto potencia la sensación de monotonía, el sistema de mejoras es confuso en general. Destacan positivamente la idea de incluir un sistema de mejoras mediante cartas, y el control de la cámara mediante el stick derecho.

Que podemos extrapolar a nuestro juego

Para los RPG la historia es un elemento muy importante, no se puede dejar de lado, el jugador tiene que evolucionar con ella, el sistema de evolución de un personaje tiene que ser constante y claro, ser fácil entender que es lo que ofrece cada avance. El sistema de combate tiene que ser ágil, sencillo de dominar, pero difícil a la hora de lograr combos muy efectivos, cada enemigo tiene que tener sus propios puntos débiles, hay que conseguir que la inteligencia artificial presente un reto al jugador, no que todos los enemigos tengan unos comportamientos predecibles. Esto se puede ir variando para conseguir controlar el flujo de dificultad según los enemigos.

La calidad gráfica es algo a tener en cuenta, tiene que estar acorde con el target al que nos dirigimos

Tales Of Symphonia

- Plataforma: Nintendo Game Cube/ PlayStation 2
- Género: RPG
- Fecha de lanzamiento: 19/11/2004(EU)
- Desarrolladores : Namco Tales Studio
- Publishers: Namco

Sinopsis

Existen dos mundo superpuestos, ambos mundos están consumiendo el mana del otro mundo y está muriendo, así que cada mundo designa un elegido para salvar a su mundo. El elegido debe viajar por el mundo rompiendo los sellos despertando a los espíritus supremos para así convertirse en ángel, y así regenerar el mana, salvando su mundo.

MDA

Mecánicas: Avanzar en un espacio 3d, avanzar en espacio 2D (combate), atacar, modificar comportamientos de la AI, customizar menús y fuentes, hablar con NPC, un sistema de compra/venta, guardar, cargar, magias.

Dinámicas: Esquivar enemigos, protegerse, ataques combinados,

Estética: motivación las ganas de completar la historia, mejorar el personaje, recompensas

Flujo de partida

El jugador puede elegir el nivel de dificultad al iniciar al juego, al superarlo por primera vez se desbloquea una nueva dificultad. También dependerá de la forma en que le jugador quiera pasarse el juego. A medida que el juego avanza va incrementando la dificultad de los puzles que se presentan y los enemigos que aparecen.

Target

Es un juego pensado para aquellos que les gustan los mundos abiertos, explorar nuevas zonas y conseguir nuevos elementos y títulos , y poder personalizarse, es un juego destinado a exploradores principalmente según el Bartle test, conteniendo elementos para hacerlo atractivo a achivers y Explorers, con la opción de conseguir pasarse el juego completando todos los secretos.

Es un juego para gente con imaginación, que les guste la abstracción, la aventura, y para aquellos que les gustan los desafíos, y son persistentes hasta lógralo, según *Big Five Personality Traits* estas características son de personas eficientes y abiertas.

Críticas al juego

Vandal.net

Destacan la profundidad de la historia, y el desarrollo de cada personaje de forma individual, también el hecho que los combates no sean basado en encuentros aleatorios, si no que puedas evitar un encuentro esquivando a un enemigo.

También destacan su sistema de combate, ya que no usan batallas por turnos, si no a tiempo real, el jugador controla un personaje y los demás son controlados por una inteligencia artificial, permitiendo también que algún otro jugador pueda controlar otro personaje durante el combate.

El juego está diseñado a base de mazmorras con puzles, y el uso de ítems. También seremos capaces de hablar con multitud de NPC permitiendo que nos adentremos más en la historia o conseguir ítems especiales.

Gráficamente utiliza personajes poligonales aplicándoles el efecto Cell shading para dar una estética de comic.

meristation.com

Un juego ágil entretenido y de un gran duración, una historia muy interesante, ofreciéndonos historias secundarias que ayudan a meterse más en la historia, un muy buen nivel gráfico, un sistema de combate distinto que ayuda mucho a mejorar el juego, El apartado musical hace desfallecer el juego, ya que no siempre la música encaja con lo que está sucediendo, en algún momento puede hacerse monótono, ya que siempre se basa en lo mismo, llegar a un continente solucionar algunos problemas de sus habitantes, ir al templo y derrotar a un jefe.

Que podemos extrapolar a nuestro juego

La importancia de una buena jugabilidad, incluyendo elementos para diferenciarse, los jugadores prefieren un sistema de batalla a tiempo real, permitir una personalización, pero sin excederse, ya que puede causar una sensación de no saber qué hace al usuario, o desorientarlo después. La importancia de cuidar los detalles como ofrecer historias secundarias o una riqueza de personajes para evitar caer en la monotonía. La importancia de una buena banda sonora acorde con lo que ocurre para poder mejorar la experiencia del usuario.

5.2. Análisis de referentes gráficos (Juan Mesa)

Diablo II

- Plataforma: PC

- Género: Rol – Acción RPG

- Fecha de lanzamiento: 29 de junio de 2000

- Desarrollador: Blizzard

- Distribuidor: VU Games

Diablo II tiene una gran variedad de objetivos estéticos cuando se analiza en el marco de la MDA. En primer lugar, el juego ofrece la fantasía como elemento principal. El aspecto narrativo, así como su naturaleza desafiante es una fuente de entretenimiento. Por último, al ser un juego de rol, Diablo II da un gran peso a la expresión de cada jugador al desarrollar de su personaje. Desde la perspectiva de la mecánica, Diablo II utiliza sistemas básicos de rol, y se expande en ellos, como point and click RPG.

El juego ofrece a los jugadores la oportunidad de escapar a un mundo alternativo a los demonios de batalla y salvar al mundo, guiados por un gran terreno que se extiende desde el juego anterior de la serie, y continúa a la siguiente. Al igual que con todos los juegos de rol,

Diablo II tiene un creciente sentido de reto en cuánto el juego progresa, lo que garantiza que el juego nunca se ralentiza o se convierte en simple rutina, los enemigos se vuelven cada vez más poderosos. Esta continua sensación de desafío involucra al jugador en la mejora de las habilidades de su personaje, las estadísticas y el equipo para llegar a la conclusión de la trama.

El nivel de personalización y el sistema de botín del juego, así como la compra de artículos y los sistemas de modificación, da a los jugadores la oportunidad de adaptar la experiencia a sus propios deseos. La gran cantidad de opciones que se le presentan al jugador en términos de artículos y equipo, toma mucha importancia en el juego, por ejemplo, existe una sola arma en múltiples variantes con diferentes bonos y valores para las estadísticas, etc. El juego también tiene una gran variedad de árboles de habilidades para cada tipo de personaje, así como una amplia colección de tipos de enemigos con el fin de mantener el interés en el juego durante períodos prolongados de tiempo.

En general, en la aplicación del marco MDA para Diablo II, se puede ver como la estética, la dinámica y la mecánica trabajan bien para crear una experiencia única que uno pueda definir el juego simplemente con "diversión" . En Diablo II, la mecánica y la dinámica van de la mano con el fin de lograr los objetivos estéticos.

¿Qué puedo extraer de Diablo II para utilizarlo en nuestro videojuego?

Primero de todo, la estética del HUB, todo el tema de la visualización del nivel de vida y de maná es muy compatible y adaptable, el sistema de personalización de equipo e inventario, es muy sencillo y a la vez efectivo, por lo que es un buen aliciente a incluir o plasmar en nuestro videojuego.

Nosotros pensamos darle mucha importancia al árbol de habilidades de nuestro personaje, ya que te lo puedes personalizar tal y cómo desees, y en Diablo II se refleja muy bien la misma idea.

Extraigo que una buena combinación de MDA, garantiza el éxito en el videojuego y que la gente no se pueda llegar a aburrir con él.

Sacred 2: Fallen Angel

- Plataforma: PC, Xbox 360 y PS3

- Género: Rol – Acción RPG

- Fecha de lanzamiento: 31 de octubre de 2008

- Desarrollador: Ascaron

- Distribuidor: Atari

Principales características:

Ofrece un inmenso mundo perfectamente construido y vivo, que permite cientos de horas de exploración, ya que existen miles de misiones secundarias.

Seis personajes totalmente diferentes, cada uno con un aspecto distinto, capacidades e innovaciones.

Artes de combate modificables: puedes subir y cambiar el nivel de tu alter ego en el combate y adáptalo a tu estilo de juego.

Las deidades ofrecen disciplinas de combate únicas y la posibilidad de desbloquear desafíos adicionales.

Se utilizan efectos de física y rag doll.

Simulación del mundo real: el mundo simulado de Sacred 2 está controlado por una macro

IA (inteligencia artificial), que analiza permanentemente la situación actual del juego para controlar el comportamiento de la misma.

Los personajes creados se pueden utilizar en todos los modos de juego. Por ejemplo, un jugador puede iniciar una campaña de la historia, luego unirse a una sesión multijugador libre y después volver a continuar con la historia.

Ajuste inteligente de la orientación de los héroes durante el combate.

¿Qué puedo extraer de Sacred 2: Fallen Angel para utilizarlo en nuestro videojuego?

La gráfica, la idea del 3d de Sacred nos viene muy bien para nuestro juego, si podemos conseguir llegar al nivel de esa gráfica o acercarnos mucho, nos aseguraremos un éxito.

La personalización, en cuanto inventario y árbol de habilidades, muy parecida también a la idea que tenemos para nuestro videojuego.

Las misiones secundarias tienen un lugar muy importante en Sacred, creo que un buen juego a parte de su trama principal, tiene que ofrecer una buena variedad de misiones secundarias, para que el usuario no se aburra y disfrute al máximo del videojuego.

5.3. Informe del focus group

Resumen

Este es un informe analizando el procedimiento y los resultados obtenidos del focus group realizado en el Proyecto Torchsnow con el objetivo de conocer mejor las opiniones de nuestro posible target sobre algunos aspectos del guion. Los participantes fueron seleccionados de entre posibles jugadores, incluyendo una persona que se aleja más del target establecido.

Las principales conclusiones que hemos extraído han sido que les gusta poder personalizar y una historia profunda, pero no quieren que la personalización complique mucho o no les permita entender cómo funciona rápidamente.

Procedimiento

Para la realización de este focus group iniciamos planteándonos los reactivos que utilizaríamos a medida que se desarrollaba el guion, incluyendo aquellos aspectos que podían causar alguna incomodidad, o algunos aspectos diferenciales, también incluyendo algún aspecto sobre el diseño y la durabilidad del juego.

Una vez establecidos los reactivos, iniciamos una búsqueda de los participantes, inicialmente buscamos 4 participantes, pero acabamos seleccionando 5, en el apartado Participantes, se especifica el proceso de selección.

Una vez seleccionados los participantes los citamos a un entorno preparado, donde les presentamos y explicamos en qué consistía el proyecto y que era un focus group, una vez explicado, les sentamos en una mesa y procedimos a presentar el primer reactivo y ceder la palabra a un participante, mientras ellos expresaban sus opiniones e impresiones, anotábamos un resumen de estas ideas, al final la sesión les leímos el resumen.

La duración fue de 1 hora aproximadamente.

Finalmente redactamos este informe después de extraer las conclusiones, y utilizamos estas conclusiones como guía al escribir el guion.

Participantes

Para la selección de los participantes nos basamos en el Target de nuestro juego, personas jóvenes con tiempo libre para dedicar a jugar, que les guste explorar un mundo abierto, y obtener nuevos elementos, personas imaginativas, que les gusten los desafíos y pretendan evadirse del mundo real.

Inicialmente decidimos buscar 4 participantes, que cumplieran con las características del target y fueran jugadores habituales de videojuegos, pero finalmente encontramos un quinto miembro, era un jugador habitual de videojuegos, pero no cumplía la mayoría de las características de nuestro target, nos pareció interesante incluir a esta persona como participante para tener un punto de vista externo y más o menos cercano, incluyendo así la opinión de otro posible tipo de usuario de nuestro producto.

Intentamos tener la misma representación femenina que masculina, aunque finalmente Participaron 3 hombres y 2 mujeres.

Localización

El focus group fue llevado a cabo en el Laboratorio de usabilidad del CITM, aprovechando la mesa redonda que tiene, colocamos a los usuarios y al moderador en ella, elegimos

usar una mesa redonda ya que es una forma sencilla de que todo el mundo escuche a todo el mundo y se puedan ver fácilmente.

Objetivo

El objetivo principal de este focus group era ver las opiniones de nuestro target sobre diversos temas que aparecían en nuestro proyecto, conocer sus opiniones y sensaciones, y así ser capaces de mejorar el proyecto e incluir nuevos elementos haciendo que este sea más cercano a lo que ellos desean.

Reactivos

1. Me considero una persona tranquila.
2. Me gusta poder hacerme un personaje único.
3. Muchas veces no entiendo tantas opciones que tengo a la hora de crear un personaje, ni como me afectan durante el juego.
4. Me gusta compartir lo que hago en el juego en las redes sociales.
5. Creo que una mujer puede desempeñar un papel protagonista.
6. Me gusta conocer el pasado de los personajes para entender mejor sus acciones y sus motivaciones.
7. No me molesta que los enemigos sean animales.
8. Creo que añadir efectos de salpicadura de sangre a los ataques es buscar un realismo excesivo.
9. No existe ni el bien ni el mal.
10. El héroe tiene que salvarlos a todos.
11. El héroe tiene que perder algo importante.
12. El viaje del héroe tiene que ir paso a paso y mostrando como lo que está aprendiendo cambia su perspectiva del mundo.
13. El héroe necesita compañeros.
14. Me gusta que varíe mi profesión según como yo distribuyo mis puntos de habilidad.
15. En un menú quiero ver todas las opciones posibles, no me gusta tener submenús.
16. No me importa la calidad gráfica si el juego es divertido y tiene una buena historia.
17. Quiero aprender cosas mientras juego.
18. Juego para evadirme de la vida real y olvidarme de mis problemas cuotidianos.
19. La banda sonora tiene que contar pate de la historia, ayudar a entender las situaciones y colaborar en la inmersión.
20. Me gusta la temática nórdica.

Conclusiones

Después de analizar las respuestas e impresiones dadas por los usuarios llegamos a las siguientes conclusiones:

A los usuarios les gusta poder personalizar el aspecto de su personaje, les hace identificarse más con él, pero no quieren tener muchas opciones de estadísticas del personaje ya que son confusas, quieren conocer para que sirve cada uno y como les afecta, ya que muchas veces esto produce confusión.

No creen que sea algo imprescindible poder compartir en las redes sociales sus progresos o momentos en el juego.

En general creen que una mujer puede ser protagonista, pero no les llama demasiado la atención esto, prefieren poder elegir el sexo del protagonista.

Conocer la historia de los personajes y así entender mejor sus motivaciones les parece algo básico en estos juegos, les gusta que se desvele a medida que avanza la historia y a ser posible ir dejando pista sobre ellos antes de desvelarlo, consideran que tiene que haber alguna historia cruzada y que alguien acompañe y apoye al protagonista.

Los usuarios no han mostrado demasiado aprecio por enemigos animales, prefieren que sean animales antropomórficos o no existentes, las mujeres se han mostrado poco interesadas en el uso de sangre, a los hombre no les ha parecido un elemento imprescindible aunque les hace gracia.

Todos consideran que las acciones no son buenas ni malas, depende de quien las percibe, alguien puede ser malo y al final ser bueno y a la inversa, El héroe no tiene que ser el bien absoluto, les gusta poder decidir si es bueno o no tan bueno, y a quien salvar y a quién no, no creen que sea necesario que el héroe pierda algo durante su viaje, pero lo consideran interesante, les gusta la idea de un mundo que va cambiando a medida que avanzas y descubres nuevas cosas, y ver el mundo de forma distinta según la historia y lo que aprende el protagonista.

Les gusta la idea de que puedan aplicar puntos y variar sus estadísticas y eso les otorgue una profesión, pero quieren pocas opciones ya que si no lo ven muy complicado.

Prefieren tener submenús con una navegación sencilla.

No consideran la calidad gráfica como algo primordial, pero reconocen que hace el juego más atractivo.

Les parece interesante la idea de aprender cosas nuevas mientras juegan.

No consideran el juego como una forma de evasión, si no como algo para distraerse y desestresarse, un pasa tiempo en sus momentos de ocio.

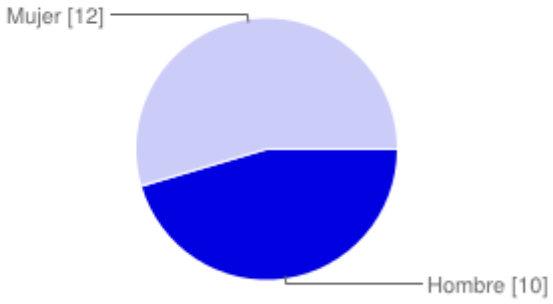
Creen que una buena banda sonora es algo que ayuda mucho a la calidad final de un juego.

No les parece una mala temática, no es su favorita, ni lo que más les guste, pero les parece muy interesante.

5.4. Datos obtenidos de la encuesta sobre videojuegos y su percepción social

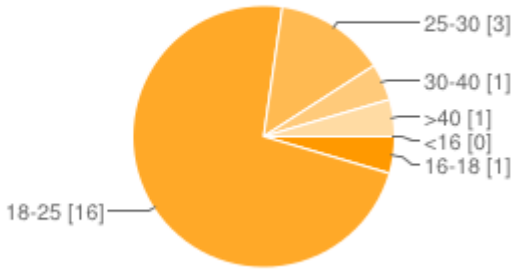
Los resultados de la encuesta están adjuntos en versión pdf y versión Excel para su consulta más detallada.

¿Cuál es tu sexo?



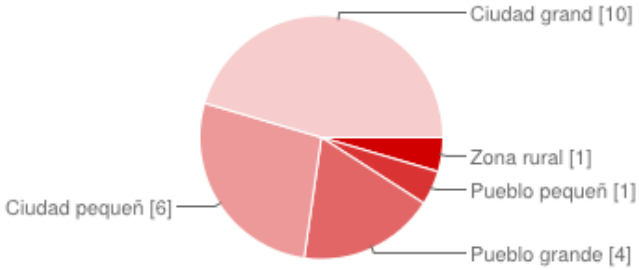
Hombre	**10**	45%
Mujer	**12**	55%

¿Cuál es tu edad?



16-18	**1**	5%
18-25	**16**	73%
25-30	**3**	14%
30-40	**1**	5%
>40	**1**	5%
<16	**0**	0%

¿En qué tipo de población vives?

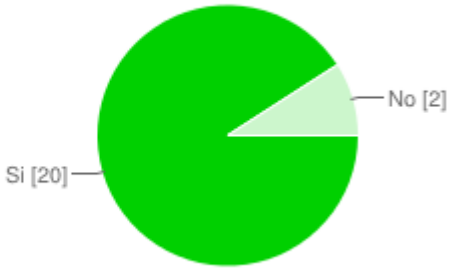


Zona rural	**1**	5%
Pueblo pequeño	**1**	5%
Pueblo grande	**4**	18%
Ciudad pequeña	**6**	27%
Ciudad grande	**10**	45%

¿Has jugado alguna vez a algún videojuego?

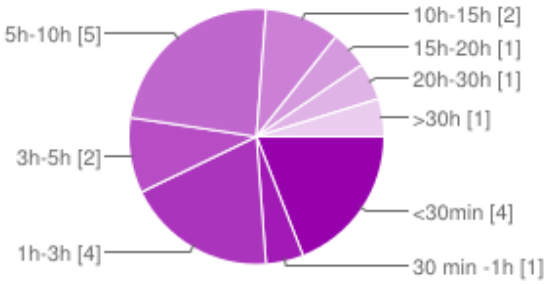
Si	22	100%
No	0	0%

¿Te gustan los videojuegos?



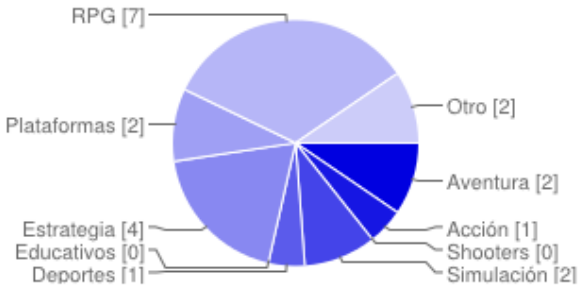
Si	20	91%
No	2	9%

En caso afirmativo ¿Cuántas horas a la semana juegas?



<30min	4	19%
30 min -1h	1	5%
1h-3h	4	19%
3h-5h	2	10%
5h-10h	5	24%
10h-15h	2	10%
15h-20h	1	5%
20h-30h	1	5%
>30h	1	5%

¿Cuál es el género al que más juegas?



Aventura	2	10%
Acción	1	5%
Shooters	0	0%
Simulación	2	10%
Deportes	1	5%
Educativos	0	0%
Estrategia	4	19%
Plataformas	2	10%
RPG	7	33%
Otro	2	10%

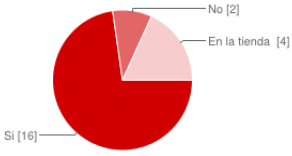
¿Crees que los videojuegos son una nueva forma de arte?

Si	22	100%
No	0	0%

¿Por qué?

Porque en un videojuego hacen falta personas que se encarguen del guión, artistas en animación, en modelado, texturizado o en cualquier variedad de arte digital. También es necesario crear un game design con la historia, los niveles, el género, etc. Un videojuego, en definitiva, tiene mucho trabajo y muchos artistas involucrados. Es una nueva forma de expresar distintos mundos y comportamientos, acontecimientos, etc. de forma virtual. Los videojuegos tienen una finalidad comunicativa(aparte de otras), donde se expresan ideas, emociones, sentimientos situaciones e incluso una visión concreta del mundo(inclusive de otros mundos inventados). Y todo ello con un medio artístico y estético. Eso lo hace un arte, aunque como tal habrá buenas obras y obras mediocres. La creación de un videojuego implica la creatividad de las personas, aplicando esta creatividad e imaginación se desarrollan los videojuegos. Por lo tanto creo que los videojuegos son una obra de arte, como lo son los cuadros, libros, etc. Són como un cuadro, una película... no todos lo són pero pueden llegar a ser-lo Son una manera diferente de explicar una historia como sería en el cine PORQUÉ SON HERMOSOS ;_; no ahora enserio, todo lo que es inventarte una historia original, hacer los gráficos - esta parte me encanta - y que te haga adentrarte en el mundo, es mejor que el cine y el cine es un arte, así que los videojuegos también. Porque es una manera de expresar a través de historias. Un conjunto de personas busca expresar emociones, sentimientos o simplemente mostrar mundos imaginarios a través de personajes, épocas, historias, etc. Es otro tipo de expresión artística, y combina distintos tipos de arte. Per què es tracta de crear una realitat Porque permiten expresar y hacer llegar emociones de una forma equivalente a otros medios considerados arte (véase cine, literatura, etc), y por lo tanto no deberían ser considerados menos. Porque expresa un mensaje, en un sistema audiovisual, contiene todo tipo de elementos considerados arte (texto, imágenes, vídeo, etc) Porque requieren estar bien pensados y realizados como.podria ser una pelicula para hacerte realmente sentirte dentro de él. Puedes encontrar juegos que tienen gráficas espectaculares. Por lo que ha intervenido un diseñador. Por lo tanto se puede considerar arte. Independientemente de que el estilo sea de tu agrado. Incluso alguno de ellos llegan a tener una poética visual. por que requieren imaginación por parte de los desarrolladores! :D Los gráficos de algunos son impresionantes y no se puede considerar el cine una forma de arte y no hacerlo de un videojuego, que exige también de un buen diseño audiovisual (y encima interactivo, lo que lo hace más complejo). Hay que sí, y hay que no... Los que considero que sí, son los que trabajan mucho con los gráficos y con las historias, que hasta pueden llegar a conmover. Hay otros que son sólo para matar el aburrimiento, y no los considero arte.

¿Crees que los padres o tutores deben informarse del tipo de juego que compran a sus hijos siendo estos menores?

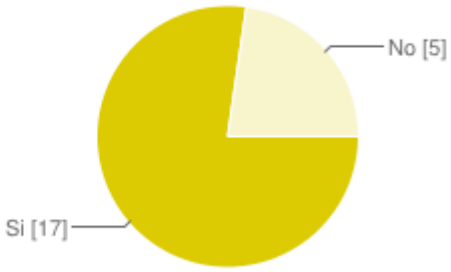


Si	16	73%
No	2	9%
En la tienda deberían de advertirles si ven que el juego no es apropiado para la edad de sus hijos	4	18%

¿Por qué crees que muchos no lo hacen?

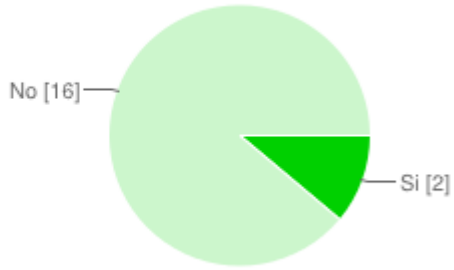
en el caso de la tienda, para vender mas en el caso de los padres, para que el niño esté quietecito X) Per tenir al crio content i entretingut En general la gente es apática. Por vagancia (?) O bien se criaron en una época en la que los juegos eran algo para niños y por eso no se los toman en serio o bien les da igual. Supongo que algunos les dejan más libertad para escoger los juegos a los que quieren jugar, y otros quieren controlarlo más. Creo que hasta cierto punto, es comprensible. Vagancia Porque en las tiendas solo quieren vender y los padres contentar a sus hijos, algo que no esta bien, las tiendas deberían de informar y los padres informarse. Para vender el producto Porque no prestan la suficiente atención. Únicamente buscan que sus hijos estén entretenidos durante un tiempo. Porque a los padres de hoy en día lo que les importa es tener entretenidos a los hijos y que a ellos no les molesten. Por despreocupación y ignorancia. Falta de conocimiento o por no querer hacer enfadar a sus hijos. Por pereza o pasotismo. Por que para ellos son un simple juego que les da bastante igual y prefieren esperar que su hijo haga o diga algo que no tendría que hacer o decir y quejarse del juego que anticiparse y comprobar que ese juego es apropiado para su edad. Porque sólo compran para saciar las ansias de los niños. Muchos padres solo quieren que sus hijos estén entretenidos, no molesten y no se quejen, si para ello deben pagar por un juego no recomendable para sus hijos, lo hacen. Por desconocimiento de los contenidos del juego. Por la presión de los menores.

¿Sabes que es PEGI?



Si	17	77%
No	5	23%

En caso afirmativo ¿Crees que la gente sigue sus recomendaciones?



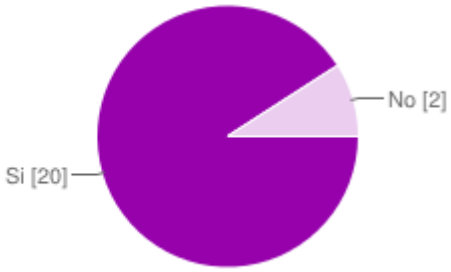
Si	2	11%
No	16	89%

¿Por qué?

Es igual que los términos y condiciones de uso. La gente que compra sus propios videojuegos(sean mayores de edad o no), compran basándose en otros aspectos normalmente. A muchos tutores de los menores de edad que compran el juego(entendiendo que aun siendo menor el hijo es suficientemente pequeño como para no ir a comprarlo el solo) solo les importa que el hijo deje de pedirles continuamente X juego, muchos ni se informan de las recomendaciones, así que mucho menos van a seguirlas. (Luego hay otros padres que no compran un juego solo porque lo pida el hijo, sino que comprobarán como es este, si incluye violencia, ... y en caso que no se recomiende, no lo comprarán) Porque prefieren evitar discusiones Vagancia again Por desconocimiento. por que si les gusta no van a privarse de ello por que una pegatinita lo diga Porque lo toman como algo muy relativo. Porque a los niños les gusta los videojuegos de violencia A muchos les da igual, y aceptan recibir ese contenido. La gente no le preocupa lo que contenga un juego, para la mayor parte de la gente todos los juegos son iguales. Por que ahora los niños van a comprar juegos ellos solos sin sus padres y muchas veces los padres no se dan cuenta. Porque el videojuego tiene una imagen atractiva para el jugador, aunque la edad no sea la apropiada, igual que en las películas. No conozco a nadie que lo haga. En general creo que no se suele tener en cuenta, sobretodo cuando ya eres mayor y escoges tu mismo los videojuegos. Siguen sus recomendaciones, aunque no de forma inteligente. Es decir, no valoran a su hijo de forma particular y lo adaptan al contenido del juego, sino que utilizan un simple criterio de edad, bastante objetivo y sin valor real. Potser els tutors legals, però sinó encara és més emocionant

Símbolos PEGI

¿Reconoces estos símbolos?

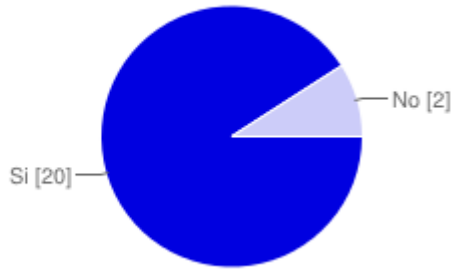


Si	20	91%
No	2	9%

¿Crees que es necesario incluir alguno más? ¿Cuál? ¿Por qué?

Creo que tendrían que tener algo mas de seguridad y que al menos los de 18 estuvieran mas restringidos , como que en la tienda te pidieran el DNI o algo o acompañado de un tutor. Que son adictivos (o es el símbolo de los dados?) Que pueden causar aislamiento social. no Puede. ni fava No me parece necesario incluir más. Algún símbolo relacionado con Dinero? Quizás algún videojuego incita a tener tendencias de despilfarro de dinero y en la vida real los niños o jugadores que no son conscientes, siguen con la misma actitud. Si, para saber que edad necesitas para comprar el juego No. Ahora mismo no se me ocurren otros para incluirlos. Yo creo que no! Los que están ya especifican de que tratan los juegos. No. Creo que con esos se pueden representar bien todos los contenidos. No, porque los existentes ya nos proporcionan información sobre el contenido del videojuego. No No, creo que ya está todo. ¿no? NOSE No. Ya tienen todo lo que considero necesario. Nope No, creo que todos dan buenas explicaciones No. Ya estan bien.

¿Crees que los padres o tutores tendrían que jugar con sus hijos?



Si	20	91%
No	2	9%

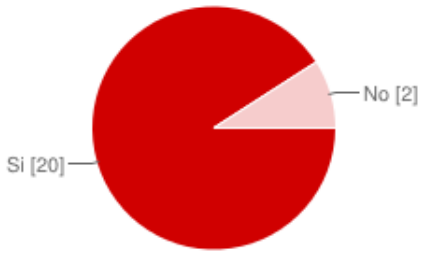
¿Crees que los videojuegos son una herramienta positiva para la educación?

Si	22	100%
No	0	0%

En tu opinión, ¿mejoraría la introducción de videojuegos el sistema educativo actual?

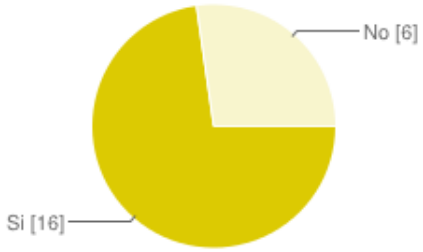
no Podría ser útil para aspectos memorísticos, pero habría que hacerlo bien. Depende de cada juego, naturalmente. Ya se sabe que sí y además resulta mucho más interesante convertir el aprendizaje en un juego. Si, muchísimo. Si, si se enseña de forma adecuada. Actualmente ya se usan algunos(pocos) videojuegos en el sistema educativo(trabajo en una escuela primaria pública). Y la mejora, los niños prefieren jugar al ordenador en algún videojuego educativo que hacer los ejercicios tal cual en un papel o libreta. (aunque opino que ambas cosas son necesarias, por más que sean más receptivos con los videojuegos, no se debe prescindir de lo demás) No lo sé. Si no han de ser un trunyo avorrit, sí Si. Hay muchos juegos educativos y a la vez no son educativos pero te enseñan un montón de cosas. Se han hecho experiencias yo por lo que se no han dado el fruto que se esperaba. (o por limitación de la programación del juego, no hay final, o por desconocimiento del profesor de las posibilidades) Enseñar jugando es mucho mejor que memorizar y leer libros aburridos. Si. si Sep. Pero no todos los géneros. Hay muchos videojuegos que requieren la resolución de problemas complejos, fomentando la capacidad de atención, de resolver problemas y la creatividad del chaval. Además aportan valores como la amistad, el sacrficio, la generosidad, el trabajo en equipo, etc. Si Siii Siempre y cuando estén adaptados a la edad y madurez de los niños. Igualmente, cabe decir que la mejor escuela para los niños es la vida real y las relaciones reales. Si, de hecho ya hay. Muy simples e infantiles pero si.

¿Conoce los juegos de rol? Tanto en su formato de juego de mesa o videojuego



Si	20	91%
No	2	9%

¿Has jugado alguna vez a alguno?

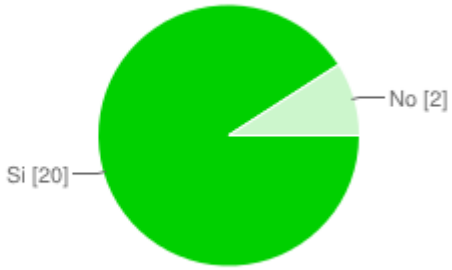


Si	16	73%
No	6	27%

¿Podrías explicar brevemente tu opinión sobre este tipo de juegos?

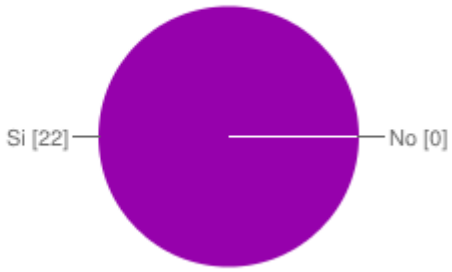
Creo que son buenos para el desarrollo de la inteligencia lógico matematica http://blogs.elpais.com/laboratorio-de-felicidad/2014/03/8-formas-de-ser-inteligentes.html Son entretenidos y, como se señala más adelante, creo que pueden servir para mejorar las habilidades sociales y la empatía. En realidad creo que se utiliza en psicología para "ensayar" como enfrentarse a determinadas situaciones. Muy divertidos, con posibilidades infinitas, y representan un buen desafío, además de aportar una gran sensación de realismo. Creo que son muy divertidos, entretenidos y además permiten desarrollar tu imaginación, conocerte más a ti y también al grupo de amigos con los que juegas. Son juegos muy complejos que se tienen que jugar con mucha cabeza No se me dan bien, así que no suelo jugar. Pero están bien para quien los disfrutan. Son juegos interesantes, que nos ayudan a ejercitar la mente y a desarrollar capacidades. Me encantan. Han sido mi infancia. Una gran formad e escapar de la realidad y de entretenimiento. Que son geniales! Te permiten usar tu imaginación al máximo, son juegos en los que el limite se lo pone uno. A demás ayudan a mejorar la expresión de la gente y la interpretación. creativos Para mi gusto, exigen mucho al jugador y se gasta mucho tiempo. Son un tipo de juego que considero que puede dar a lugar juegos bastante interesantes con bastante rejugabilidad. Es uno de mis generos favoritos, tanto los de mesa como en videojuego. Jugar en un juego de rol de mesa con los amigos ayuda a crear situaciones inesperadas dentro del juego, ver como se desenvuelve cada persona. Se dice que nos definen nuestras acciones, y aunque sea en ficción en este caso, puede ayudar mucho a ver como es uno mismo(y los demás). En cuanto a los videojuegos de rol, saber que tus acciones acarrearán unas consecuencias o otras(dentro del juego) es muy interesante y estimulante. Para mi son de los mejores. Hace que uses muchísimo la imaginación y creo que es una de las cosas que faltan en esta sociedad ES MUY DIVERTIDO, siempre que el máster sea enrollado. sirven para pensar y desarrollar el cerebro

¿Crees que pueden mejorar la empatía?



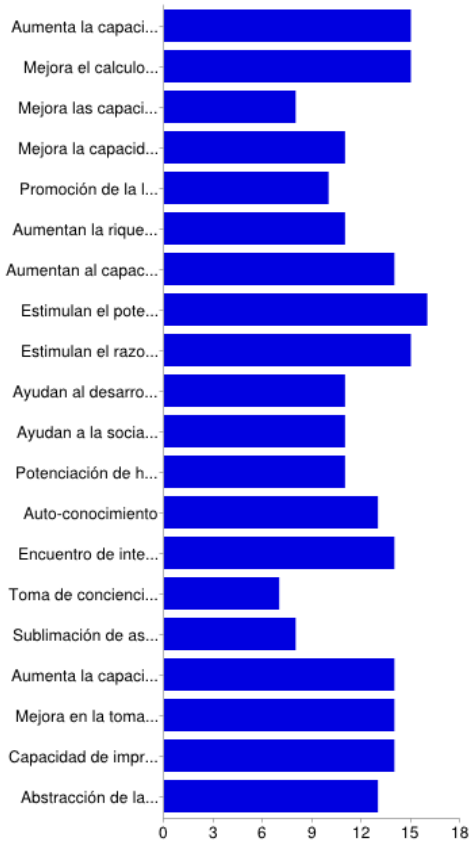
Si	20	91%
No	2	9%

¿Crees que pueden usar en la enseñanza?



Si	22	100%
No	0	0%

¿Cuáles de los siguientes beneficios cree que aportan los juegos de rol?



Aumenta la capacidad de Enseñanza-Aprendizaje	15	6%
Mejora el calculo mental	15	6%
Mejora las capacidades mnemotécnicas	8	3%
Mejora la capacidad de esquematizar y la toma de notas	11	4%
Promoción de la lectura como medio lúdico y recreativo	10	4%
Aumentan la riqueza expresiva	11	4%
Aumentan al capacidad de comprensión lectora	14	6%
Estimulan el potencial creativo e imaginativo de la persona	16	7%
Estimulan el razonamiento y la lógica	15	6%
Ayudan al desarrollo de la empatía y la tolerancia	11	4%
Ayudan a la socialización	11	4%
Potenciación de habilidades no explotadas	11	4%
Auto-conocimiento	13	5%
Encuentro de intereses y gustos personales	14	6%
Toma de conciencia y responsabilidad	7	3%
Sublimación de aspectos reprimidos de la personalidad	8	3%
Aumenta la capacidad de trabajo en grupo	14	6%
Mejora en la toma de decisiones	14	6%
Capacidad de improvisación y resolución de problemas	14	6%
Abstracción de la realidad	13	5%

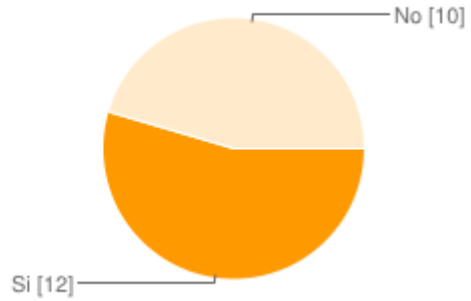
¿Opinas que es merecida la fama de juegos violentos que tienen los juegos de rol?

ni idea no juego | no | No, todo es sensacionalismo, por desgracia. | Me la pela, me gustan y punto. Al que no, que no mire. | por supuesto que no | No para nada. | Algunas veces sí, ya que se trata de violencia gratuita. | No, que yo sepa todo viene del llamado "asesino del rol", pero la motivación de su crimen poco o nada tiene que ver con el juego. | No es para nada merecida. Dependerá de cada título, aun así la mayoría no lo son. Batallas y estrategia: si, violencia:no. | Hay muchos aspectos colindantes que se tendrían que analizar respecto a la mala fama de los juegos violentos de rol. Los casos que se han dado con finales catastróficos, se tendría que analizar o conocer mas sobre el estado emocional del jugador, su entorno familiar, su sociabilización, su capacidad de discernir la realidad del juego, entre otros factores. Pero creo que todo influye. | No | No era consciente de que se consideraban violentos | Que es injusta! Creo que por culpa de varios casos fortuitos se le tildo como juegos violentos. | No, una cosa es el videojuego y otra muy distinta como esté educado el niño y su mentalidad. | Por mi parte nunca he tenido la sensación de que tuvieran esa fama. | Ciertamente, algunos lo son, pero no todos. | Si | No. Hay muchos juegos de Rol que no implican violencia o al menos no tanta, y si al implica esta claro que en la televisión hay el doble de violencia y esta al alcance de todos los niños. | No, no todos los juegos de rol son violentos

¿Que significa RPG?

No lo sé | No sé | Role Playing Game | Role Playing Game | Role playing game | juego de rol | role playing games (juegos de rol) | Role Playing Game | Rolling play game | Role-playing game | roling player game

¿Conoces los JRPG?

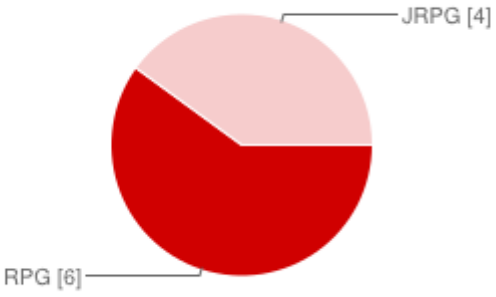


Si	12	55%
No	10	45%

¿Que significa JRPG?

No lo sé | Japanese RGP | Japan RPG | Japanese Role Playing Game | Los juegos de rol de origen japones | juego de rol japones | Japanese role-playing game

Prefieres los RPG o los JRPG

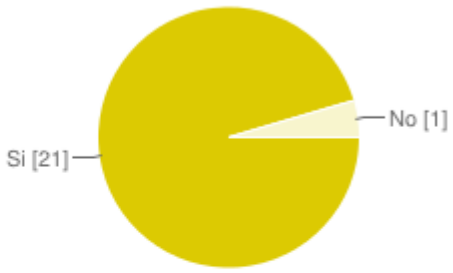


RPG	6	60%
JRPG	4	40%

¿En caso de haber jugado, a que videojuegos de rol has jugado?

Final Fantasy y algun otro que ahora no recuerdo el nombre. | Muchos lol | No suelo jugar a juegos de rol, por lo que no sabría decir. | Final fantasy (III, IV, V, VI, VII, VIII, IX, X, X-2) Diablo (I, II y III), Secret of mana, Sword of mana, Chrono trigger, Neverwinter Nights, Mass effect, The Elder Scrolls (III(Morrowind), IV(Oblivion) y V(Skyrim)), ... | Sobretodo a Guild wars. Otros he probado pero ninguno me ha durado tanto. | Final Fantasy, Dragon Quest, World of Warcraft, Guild Wars (1 y 2), Breath of Fire, etc. | final fantasy | Ranma 1/2 SNES RPG Dragon Ball RPG Super Mario RPG Sailor Moon SNES RPG Final Fantasy Star Ocean The Last Remnant etc. | Fable 1 y 3 Sacred Torchlight 1-2 Overlord 1-2 Diablo 2-3 Elder Scrolls Morrowind, Oblivion y Skyrim Terraria Starbound etc... | League of legends, final fantasy | The legend of Zelda , Final Fantasy ,World of Warcraft , The elder Scroll entre otros. | wow, lineage, tibia, sacred, diablo, final fantasy, baldurs gate, dragon quest, dungeon master, everquest, fallout, kingdom hearts, neverwinter nights, phantasy star, pokemon, Star Wars: Knights of the Old Republic, the elder scrolls, ultima online, vagrant story, mapple story, | Diablo, wow, Dragon Age y muchos mas.

¿Dejas/Dejaras a tus hijos jugar a videojuegos?

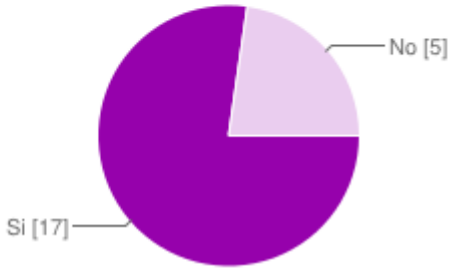


Si	21	95%
No	1	5%

¿Y a videojuegos de rol?

Si	21	95%
No	1	5%

¿Participas/Participaras en la selección de videojuegos de tu hijo?

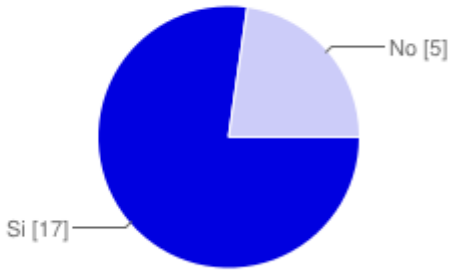


Si	17	77%
No	5	23%

¿Por qué?

Podría hacer recomendaciones, pero no prohibir u obligar a nada. Por que puede ser una parte del proceso de desarrollo del hijo. Por su contenido y por como puede afectarle. Para que su contenido sea el adecuado para su edad. por que no quiero que compre mierda. Que el genero sea de calidad :D Porque soy game designer, le haré yo los juegos ;). XD Porque el podría escoger lo que quisera no querria prohibirle las elecciones de sus cosas Mientras sea pequeño sí. En parte porque quiero utilizar los videojuegos para reforzar su educación y en parte porque yo no dejaría a un niño pequeño jugar a GTA o ver Sálvame. Debo asegurarme que hacen una buena elección y que aquello a lo que jugarán no será negativo para ellos. Depende de la edad que tenga mi hijo no le permitiré jugar a según que juegos. Tal y como estan las nuevas generaciones, hace falta mas control sobre ellos. no tengo Para conectar y pasar un rato con ellos Mi hijo deberá elegir sus videojuegos como elegirá a sus amigos. Porqué si se va a gastar el dinero, que almenos se lo gaste en algo bueno. Los niños son muy suggestionables, y si un videojuego no esta pensado para mi(futuro/supuesto)hijo pues no veo porque debería jugar. Si el elige bien, obviamente decide él, es quien va a jugar, pero siempre acorde con el contenido. Porque me aseguraré que escoge los más acordes a su edad. N'hi ha que no son adequats per certes edats Quiero que elija las opciones más adecuadas, a parte de poder recomendarle algunos títulos.

¿Estas/Estarás junto a tu hijo mientras juegue?



Si	17	77%
No	5	23%

¿Por qué?

Para que no tenga actitudes negativas y enseñarle a canalizar las emociones si en algún momento se enfada. Porque a mi tambien me gustan Depende pero si, porque es un buen momento para compartir con él. Depende del juego no tengo Per què s'ha de saber desenvolupar sol en el seu personatge o rol dins el videojoc. És una oportunitat per a què la criatura experimenti què significa ser qui vols ser i n'aprengui de llibertats. Ara bé, s'ha de tenir un control horari... Porque se la ilusión que me hacía a mí que mi padre jugara conmigo. Para ganar confianza en la relacion padre-hijo Por que no sera necesario. A lo mejor un día me pongo a jugar con el pero no lo veo necesario. por que querré jugar yo también! :P C'mon, daddy's the fucking boss at that shit. El niño debe aprender. Hasta saber el desarrollo del juego, y como se implica en el. Para saber en qué le puede beneficiar o afectar. No todo el tiempo(jugar solo también es divertido y ayuda a conocerse uno mismo), pero pienso que es muy enriquecedor para un niño que su madre o su padre jueguen con él también. Ademas de reforzar los vínculos efectivos entre ambos. Cuándo él quiera, haré experimentos con él. Si es muy pequeño o necesitara supervisión sí pero también es una buena oportunidad de que aprenda a ser más autónomo. Porque es mi hijo. Para ayudarle.