Universitat Oberta de Catalunya
uoc.edu

Videojuego: Lands of the Gondria.

Alumnos:
Julen Gallego Sevil
Jordi Puertas Prieto

Máster en Diseño y desarrollo de videojuegos
Trabajo Final de Máster

Profesores:
Javier Luis Cánovas Izquierdo
Jordi Ducha Gavaldà
Helio Tejedor Navarro

Fecha Entrega:
30-06-2019

Esta obra está sujeta a una licencia de Reconocimiento-NoComercial-SinObraDerivada 3.0 España de Creative Commons

©
Reservados todos los derechos. Está prohibido la reproducción total o parcial de esta obra por cualquier medio o procedimiento, comprendidos la impresión, la reprografía, el microfilme, el tratamiento informático o cualquier otro sistema, así como la distribución de ejemplares mediante alquiler y préstamo, sin la autorización escrita del autor o de los límites que autorice la Ley de Propiedad Intelectual.

FICHA DEL TRABAJO FINAL

Título del trabajo:	*The Lands of Gondria*
Nombre del autor:	*Julen Gallego Sevil, Jordi Puertas Prieto*
Nombre del consultor/a:	Javier Luis Cánovas Izquierdo
Nombre del PRA:	*Jordi Duch Gavaldà, Helio Tejedor Navarro*
Fecha de entrega:	06/2019
Titulación::	*Máster en Diseño y desarrollo de videojuegos*
Área del Trabajo Final:	*Trabajo Final de Máster*
Idioma del trabajo:	*Español*
Palabras clave	*Videojuego, Master*

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

El trabajo realizado y expuesto a lo largo de la memoria trata de un videojuego estilo Acción-Rpg, en el que se deberá salvar el reino de Gondria de bandidos y maleantes, poniéndote en la piel de uno jóven guerrero.

El motor gráfico que se ha utilizado para la realización del mismo es Unity y el lenguaje de programación C#.

Para los Assets de modelos 3D, se han buscado gratuitamente o comprado varios de ellos, ya que no se ha destinado tiempo alguno a la creación propia de dichos elementos, centrando el tiempo en el modelado del mapa utilizando dichos Assets y la programación misma del videojuego, así como la realización de la historia.

El reparto de tareas ha sido equitativo en ambos miembros del grupo, ayudándonos en lo necesario para intentar minimizar las pérdidas de tiempo y ser eficaces.

Si bien, se propusieron una serie de Hitos a realizar, no la totalidad de ellos han acabado integrándose en el videojuego o simplemente se han reducido las características de los mismos haciéndolos más simples y asequibles.

Los resultados han sido mayormente satisfactorios con lo obtenido en el tiempo dado, pudiendo haber abarcado más temas o extendiendo los implementados de no ser por temas expuestos a lo largo de la memoria. Pese a ello, se pretende continuar con el videojuego una vez realizada la entrega.

Abstract: This game was developed by Julen Gallego and Jordi Puertas, Uoc's students. This videogame is the result of two years where we have obtained the knowledge, habilities and experience necessary to do this game posible.

We used 3d model (Assets) to do this game.

The tasks were been equitatives for each of us, help to each other if necessary, but not all tasks have been acomplished, unafortunely.

The results have been satisfactoriy even with the time against.

Índice

Capítulo 1: Introducción 1
1.1 Contexto y justificación del Trabajo 1
1.2 Objetivos del Trabajo 1
1.3 Enfoque y método seguido 2
1.4 Planificación del Trabajo 2
1.5 Breve sumario de productos obtenidos 3
1.6 Breve descripción de los otros capítulos de la memoria 4
Capítulo 2: Estado del arte 5
Capítulo 3: Conceptualización 10
4. Diseño técnico 14
4.1 Entorno elegido, requerimientos técnicos y herramientas empleadas 14
4.2 Assets 19
4.3 Esquematización del videojuego 26
4.3.1 Estructura carpetas 26
4.3.2 Distribución de Escenas 28
4.3.3 Estructura de Clases C# 29
5. Funcionalidades Implementadas en el videojuego 31
Sistema Online 31
UI del videojuego 32
Sistema de Combate 34
Sistema de Movimiento 36
Sistema Inventario 36
Sistema de Diálogos 36
Sistema de Misiones 37
Sistema de portales y puertas 40
Sistema de Enemigos e IA ENEMIGA 41
Sistema de NPC’s vivos 45
Post-procesing 45
Otras implementaciones 46
6. Diseño del nivel 48
7. Conclusiones 55
8. Glosario 57
9. Bibliografía 58
10. Anexos 59

Capítulo 1: Introducción

1.1 Contexto y justificación del Trabajo

El Trabajo realizado trata de un videojuego que es la suma de aplicar parcial o totalmente, lo aprendido a lo largo del máster.

En él se han volcado las ideas y pensamientos que teníamos sobre un tipo específico de videojuego y han sido plasmadas en el mismo.
Se ha decidido hacerlo entre dos personas para poder abarcar más campos y darle varios matices diferentes. Puesto que cada uno tiene una visión distinta de un mismo tema, creemos que ha sido un aspecto positivo para el conjunto del videojuego sopesando pros y contras hasta obtener el resultado final deseado.

Si bien se dejan algunas cosas sin implementar o habiendo sido simplificadas por un tema de recursos (tiempo), se está contento con el resultado del trabajo realizado en estas semanas que ha durado la realización.

Como tal, el videojuego pretende seguir y ser acabado después de la entrega, ya que la idea principal era utilizar este trabajo como punto de partida inicial creando un videojuego 'semi-cerrado', es decir, que se pueda seguir extendiendo y añadiendo material que no se ha podido implementar, al tiempo que mejorando ciertos aspectos etc.

1.2 Objetivos del Trabajo

En la siguiente lista aparece la totalidad de las implementaciones y objetivos pensados, incluidos y propuestos en las entregas anteriores de las PEC.

-Mecánica Online
-Historia del videojuego
-Personajes Modelos 3D
-Animaciones
-Entorno y Mapeado
-Inventario
-Sistema de objetos
-Jerarquía de Clases
-Sistema de combate
-IA de enemigos y NPCS (No Player Controllable)
-Sistema de Dialogo con NPCS y Misiones
-Sonidos / FX
-Variedad de mecánicas *in-game*.

1.3 Enfoque y método seguido

La estrategia para abarcar el trabajo ha sido tener una clara planificación de qué hace cada persona del grupo.
Pese a ello, hay ciertos aspectos que al final terminan lastrando y ocupando más tiempo del que se pensó inicialmente.

Otro método seguido ha sido el de evitar que uno de los dos se estuviera más de uno o dos días con un problema bloqueado. En ese caso se decidió que se abordaría entre ambos, para poder continuar con la correcta implementación del resto de puntos.

La valoración referente a la estrategia para abordar el estancamiento o bloqueo en ciertas tareas, sirve para evitar frustración o desgana.

En este caso, al entrar el otro miembro del grupo en el problema o implementación y ser partícipe de ello activamente, se analiza y se le da un enfoque distinto pudiéndose lograr una solución no vista inicialmente. Con esto se busca lograr que el trabajo como tal nunca se quede en una pausa prolongada por, ya sea, desconocimiento o mal enfoque inicial.
Si se diera el caso de no poder realizar dicha tarea, se simplificaría o sería retirada provisionalmente.

El producto no busca ser algo novedoso, si bien la historia como tal sí que lo es, en lo que se refiere a que está pensada desde cero, el resto de sistemas ya han sido vistos en un videojuego tipo Acción-Rpg (diálogos, misiones, combate etc.) ya que se nutre mucho de estos.

1.4 Planificación del Trabajo

Para la planificación del trabajo se ha empleado un Excel con las tareas a realizar, el tiempo disponible para cada una de ellas, y quién se encargaría de cada hito del proyecto.

Se han utilizado diversos colores para una fácil visualización, y toda tarea separada por las semanas desde que empezó la primera PEC referente al TFM.

Si bien el Excel que se entregó en un principio, hay puntos que por tema de recursos no se han podido implementar, el resto se han ido siguiendo bastante a la par con lo que se propuso inicialmente.

Tiempo Meses	Marzo	Abril					Mayo					Junio				
Objetivos	25 - 31	1 - 7	8 - 14	15 - 21	22 - 28	29 - 30	1 - 5	6 - 12	13 - 19	20 - 26	27 - 31	1 - 2	3 - 9	10 - 16	17 - 23	24 - 30
Menú del juego / pantallas de selección / Loading...	X														X	
Juego multijugador y offline.	X	X	X	X												
Guardar juego / progreso															X	X
UI juego	X															X
Mapa (Niebla de guerra)							X	X								
Gestion de Misiones										X	X	X	X	X		X
Sistema de lucha y casteo de magias.								X	X	X	X	X				X
IA de enemigos.								X	X	X	X	X				
IA de NPC							X	X	X							
Creación mapa		X	X	X	X	X	X	X	X	X	X	X	X	X		
FX / Musica	X													X	X	
Animaciones (De Mixamo) y creadas en Unity	X						X	X	X	X	X					
	Julen	Jordi														
	Julen y Jordi															

Aproximación de tiempos de trabajo

A su vez, se han ido haciendo reuniones tanto físicamente como por Skype o (programas de comunicación) para ir planificando qué temas se tocarían cada semana, como llevábamos los que se estaban implementando en los días, cambios que se hacían en base a mejor o ideas nuevas que iban pudiendo surgir etc.

1.5 Breve sumario de productos obtenidos

El producto obtenido en base a estas semanas de trabajo ha sido un videojuego que tiene muchos aspectos cumplidos en el mismo.
Con esto se quiere decir, que se han implementado muchas de las mecánicas que se querían implementar.

-Inteligencia Artificial de los enemigos.
-Sistema de diálogos.
-Sistema de misiones.
-Sistema de clases.
-Creación de mundo virtual.
-Sistema Online (Incompleto).
-Sistema de combate.
-Animaciones y música.
-Mecánicas variadas de entorno/habilidades.
-Sistema básico de inventario.
-Scripts y mecánicas in-game.

Si bien queda un recorrido muy largo y extenso, hasta poder obtener un videojuego completo, vemos esta entrega como una pequeña demo que muestra una parte del mismo realizada.
Si bien en el juego como tal se ha acortado camino para que sea jugable y auto explicativo.

El detalle de cada uno de los capítulos referentes al mismo está explicado en las siguientes páginas.

1.6 Breve descripción de los otros capítulos de la memoria

Los siguientes capítulos de la memoria, junto a una pequeña descripción de lo incluido en los mismos son los siguientes:

Capítulo 2: Arte

- Este capítulo trata del género que tiene el videojuego y la tecnología que hay detrás, tanto el motor gráfico, como una comparativa entre las disponibles que se suelen utilizar etc.

Capítulo 3: Definición del juego

- Aquí se tratará la conceptualización del videojuego en sí, con que personajes cuenta, niveles disponibles, sistemas de combate etc.

Capítulo 4: Diseño técnico

- En este capítulo se definirán varios aspectos:
    - Justificación del motor gráfico elegido.
    - Requerimientos necesarios para el entorno del desarrollo del juego.
    - Herramientas empleadas para la creación del mismo, tanto a nivel de programación, como a nivel de organización de tareas.

Capítulo 5: Funcionalidades Implementadas

- Assets utilizados y recursos en el propio juego. Tanto hechos o creados por nosotros como obtenidos de terceros.
- Arquitectura del juego y sus componentes.
- IA de enemigos, junto al sistema de Misiones y Diálogos entre otras implementaciones.

Capítulo 6: Diseño de niveles

- Esta sección trata del entorno virtual creado para el videojuego. Por qué el mapa está hecho de tal manera. Explicación del criterio seguido y guía para pasarlo.

Capítulo 7: Conclusiones

- Conclusiones varias sobre todo el proceso de creación del videojuego.

Capítulo 8: Glosario

- Palabras varias y sus definiciones utilizadas en esta memoria.

Capítulo 9: Bibliografía

- Enlaces a webs o páginas de documentos utilizadas para esta memoria.

Capítulo 10: Anexos

Capítulo 2: Estado del arte

El género de un videojuego es la clasificación del mismo dentro de un grupo según su mecánica. Si bien hay videojuegos que únicamente están dentro de un género, ya sea de Terror, Aventuras, Conducción, normalmente la relación no siempre es uno a uno y más en los tiempos de hoy, que se intenta innovar cruzando géneros en un mismo videojuego o crear de nuevos en base a la mezcla de varios.
Un ejemplo podría ser el videojuego *Rocket League*, que mezcla la conducción con el deporte del fútbol.

(Imagen del videojuego Rocket League desarrollado por Psyonix)

El juego del que habla está memoria, Lands Of The Gondria, se podría clasificar como un Aventuras-Acción con un toque RPG.
Una breve descripción sobre el mismo podría ser que el videojuego, si más no, se comprende y caracteriza por la exploración del mundo virtual creado, resolución de pequeños puzles por parte del jugador, misiones que hacer por el mundo ya sea de recolección, acabar con malvados maleantes o criaturas etc. Y en definitiva desafía los reflejos del jugador, y la capacidad de resolver problemas en situaciones ya sean violentas o no.

Como detalle, el primer videojuego categorizado como “de aventura”, fue Adventure en 1979 para la consola Atari 2600.

Para la realización de este tipo de videojuegos, desde el primer juego hecho a código puro, hasta la utilización de potentes motores gráficos a día de hoy, los más comunes, o de los que más se habla en estos tiempos para pequeños desarrolladores, o estudios *Indie* son los siguientes: *CryEngine*, *Unreal Engine* 4 y *Unity*
Si bien, ahora entraré en un detalle de los 3, y más detallado será Unity, que es el software que se ha usado para el desarrollo de este videojuego, hay muchos

más motores gráficos que son posible utilizar, (y más simples y sencillos que alguno de los aquí mencionados) pero que si es cierto, tienen menos herramientas, son menos potentes o simplemente no han tenido un efecto-llamada en la sociedad de desarrolladores tan alta.

CryEngine

El motor gráfico CryEngine, ha sido desarrollado por la empresa de Software CryTek. Tiene varias versiones que van, desde el original, hasta el 5.
Permite el uso de gráficos foto realistas, pero a su vez es bastante difícil de utilizar sin conocimiento previo, haciéndolo una mala decisión de uso si el estudio recién se ha creado, o no se tiene un equipo grande detrás o medios.
Algunas de las características que incluye el motor son las siguientes:

-Soporte gráficos ultra realistas.
-Soporte realidad virtual.
-Sistema de cambio de día dinámico. (Noche, día, nieve…)
-Mapeado de tono HDR, definición de luz, tonos oscuros, etc.
-Teselado y refinamiento de mallas.
-Niebla volumétrica y partículas en sombra.
-*Render* de agua y océanos.
-Reflexión en tiempo real.
-Gratuito.

Estas son algunas de las características que presenta el motor.

Unreal Engine

El motor gráfico Unreal Engine se lanzó inicialmente en 1998, desarrollado por la desarrolladora Epic Games. A día de hoy, van por la versión 4 del motor, y como tal es multiplataforma, soportando la creación para varios dispositivos ya sea en entornos de PC, consolas o dispositivos móviles.
Algunas de las características que tiene UE4 son las siguientes:

-Hiperrealismo en los gráficos.
-Edición en tiempo real para juegos de Realidad Virtual.
-Acceso abierto al código fuente, permitiendo adaptarlo a las necesidades.
-Facilidades de programación con Visual Scripting, una herramienta propia.
- Teselado y refinamiento de mallas.
-Niebla volumétrica y partículas en sombra.
-Sistema de día/noche/efectos climáticos.

A diferencia de CryEngine, tiene una curva de aprendizaje más sencilla. Es gratuita su descarga y uso, aunque cobran un % según contrato y licencia.

Unity

El último motor gráfico a presentar es Unity, creado por Unity Technologies y lanzado en 2005. Siendo la última versión del mismo el Unity 5.
Algunas de las características de Unity son las siguientes:

-Curva de aprendizaje sencilla.
-UI Amigable, intuitiva y fácil de usar.
-Gran comunidad detrás de desarrolladores.
-Poca exigencia en el PC de desarrollo.
-Multiplataforma.
-Facilidades de programación.
-Sistema día/noche, y efectos.
-Tienda virtual

No da soporte a gráficos tan realistas como puede darlos los motores mencionados anteriormente.
En definitiva, pese a que da menos calidad gráfica, para los primeros pasos dentro del mundo, y si no buscamos foto realismo y hacer cosas muy complejas es la mejor opción en el mercado actual.

Como tal, es gratuito aunque también tiene sus 'peros', si se supera cierta remuneración (100k) en base a la creación de un juego.

Entornos para la modelación

Para el tema de creación de modelos 3D, hay multitud de herramientas en el mercado, desde gratuitas, hasta de pago. Algunas de las más utilizadas son las siguientes:

ZBrush

La herramienta ZBrush, es un software de modelado 3D desarrollado por Pixologic en 1999. Apto para Windows y Mac, permite la creación de modelos para los motores gráficos, películas, entre otras funcionalidades.
ZBrush no es una herramienta gratuita.

Blender:

Este software gratuito de código abierto multiplataforma, se dedica al modelaje, *renderizado*, iluminación entre otros de gráficos tridimensionales (3D). A día de hoy, cualquiera puede realizar mejoras o implementaciones. Se lanzó inicialmente en 1998 (Aunque sin el código fuente, este se liberó posteriormente.)

3ds Max

Anteriormente denominado 3D Studio Max, es un programa con el cual se puede crear gráficos y animación 3D. Desarrollado por Autodesk en 1990.
Es uno de los más utilizados para la creación de videojuegos, aunque también se le da uso para películas, televisión etc. El software es de pago aunque tiene versión gratuita para estudiantes.

Maya

Maya es un programa informático que se usa para el desarrollo de gráficos 3D, animación, efectos especiales entre otros. Se desarrolló en conjunto con Alias Systems y Autodesk y fue lanzado en 1998.El software es de los más utilizados junto a 3DS Max para la realización de videojuegos, como tal este también es de pago aunque tiene una versión gratuita para estudiantes.

Para el caso de Autodesk Maya/3DS Max, uno es más usado para modelaje (3DS Max) y Maya se suele utilizar más comúnmente para animaciones.
Aunque ambos sirven, si más no, para ambas cosas.

Para la realización de las texturas el programa o paquete más usado comúnmente es el de Adobe (Photoshop, ilustrator etc)

Capítulo 3: Conceptualización

Lands of the Gondria es un videojuego para PC con unos gráficos tipo Low Poly en 3D. Básicamente esto quiere decir que la malla de los modelos que usa tanto para personajes como entorno (salvo el terreno) tiene un número pequeño de polígonos.
La decisión de utilizar este tipo de gráficos, es para poder crear nosotros en caso de necesidad, las cosas a través de un software de creación de modelos 3D. Además, en temas de optimización y *renderizado*, reduce la necesidad de potencia bruta para poder mover el juego.

(Concept Art del Mundo virtual con gráficos Low Poly.)

El juego, al tener un toque RPG, busca tener una historia propia, si bien para la demo únicamente se contempla hasta cierto punto, el resumen de la historia del videojuego es la siguiente.

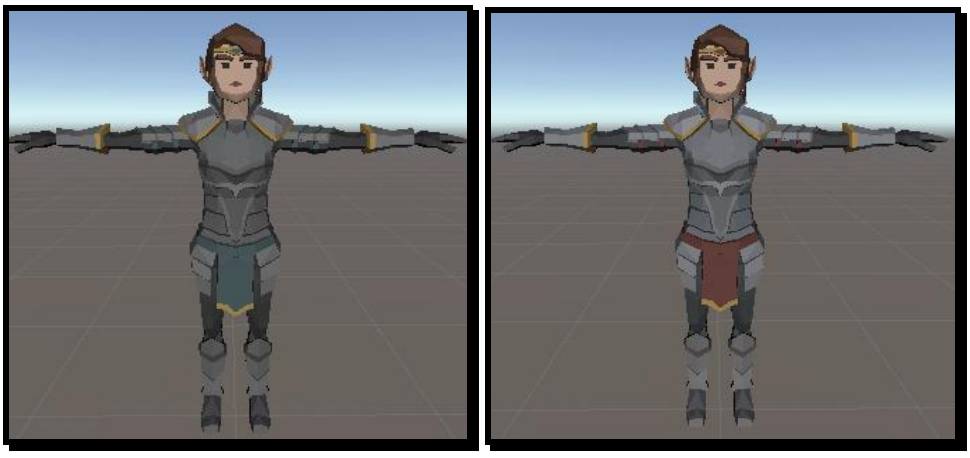
En el mundo mágico llamado Gondria, habitan un total de 3 razas, humanos, elfos y orcos. Estos estuvieron en guerra hasta hace 2088 años. Pasó que los reyes de cada raza pactaron un tratado de paz para acabar con el derramamiento de sangre.
Una de las varias condiciones que habían en este tratado era que no se podía tener descendientes con otras razas, y en caso de tenerlos toda la familia sería desterrada al sur de desierto de Shoku. Todo el mundo estaba obligado a respetar el tratado, nadie será excluido del castigo, ni siquiera los propios reyes de los reinos.
Un día, el rey de los humanos se enamoró de una maga elfa. Con ella tuvo dos hijos mellizos, de los cuales se enteró de su existencia al poco de nacer.
El rey tuvo que tomar la difícil decisión de capturar a la madre y a sus hijos para matarlos, para que nadie se enterase nunca de lo sucedido y así que no lo desterraran.
Para esta misión envió a dos de sus leales caballeros y cuando encontraron a la madre y a los niños la madre luchó contra uno de los caballeros, mientras que el otro fue directo a por los mellizos para darles el golpe de gracia. El caballero no pudo matar a dos pequeños así que decidió abandonar la misión encomendada por el rey para escaparse con los niños, y ponerlos a salvo.

Por otro lado, la madre fue capturada y llevada ante el rey.
El rey, incapaz de matar a su amada la encerró en una mazmorra de por vida, ya que si la liberase podría poner en serio peligro su reinado, y envió una orden de busca y captura con una gran recompensa a quien encontrase y entregase al desertor, inventándose una falsa historia de traiciones para desacreditarlo y salvarse.
Nexiant, el caballero que desertó y escapó con los mellizos se quedó en una granja en la otra punta del reino de los humanos porque sabía que el rey le intentaría dar caza. Los años pasaban y los pequeños crecían, aprendiendo el arte de la lucha física y mágica, al tiempo que llevando una vida normal y tranquila en la granja junto a Nexiant, al que consideraban su padre, pues nunca les contó la verdad, puesto que no les veía preparados todavía para afrontarla.
Un día, mientras Nexiant estaba en la granja y los mellizos estaban haciendo recados en el pueblo que había cerca de la granja, Mocholo, varios soldados del rey atacaron la granja, cuando los mellizos llegaron a la misma se encontraron a Nexiant a punto de fallecer, ya que uno de los leales, quien condujo un pequeño pelotón, le asestó un golpe mortal al no revelar la ubicación de los chicos.
Justo antes de morir, Nexiant les contó la verdadera historia de quienes son realmente. Pese a escuchar gran parte del relato, Nexiant murió antes de poder decirles que el rey era su verdadero padre, así que los mellizos decidieron ir al castillo a rescatar a su madre.

Para la entrega se obvian varios lugares y únicamente se es jugable con el personaje Ryn, puesto únicamente se desarrolla el primer *Stage* del juego, así que los personajes y lugares que se contemplan son los siguientes:

Definición de personajes:
Ryn y Ren: Mellizos hijos del rey Daener y la elfa Nora.
Nexiant: Caballero desertor que cuida a los mellizos
Daener: rey del reino humano.
Nora: Madre de los mellizos encarcelada por el rey.

Para los personajes principales, de Ryn (el que es jugable) y Ren, el modelo es el mismo, únicamente cambia el color de la armadura y el pelo.



El resto de personajes y Npc's son del estilo *low poly*, igual que los personajes principales.

Lugares disponibles del Stage I:
Granja Nexiant: lugar donde crecen los mellizos (Punto inicial del juego).
Aldea de las aguas: Pequeña aldea cerca de la granja.
Mocholo: Pueblo comercial que hay cerca de la granja.
Tierras fangosas: Tierras de nadie, donde van los desterrados y conviven bandidos con orcos

El mapa creado en formato boceto, es el siguiente:

Aunque el mapa mostrado en la ilustración superior, muestra todo el Reino de *Gondria*, el primer *Stage* es el que está modelado y creado como se comentó en páginas anteriores. (Es el recuadro marcado con un cuadrado rojo)

Si bien el *Stage I* ha sufrido alguna variación de lo que se propuso inicialmente, el mismo se tratará en los próximos capítulos de la memoria.
El resto del mapa representa los Stages II y III, que no estarán incluidos en el juego a presentar por un tema de recursos, pero como tal se diseñó el mapa entero.

En la creación del mapa se busca el tener varias áreas, ya sean bosque, desierto, montañas/ montañas nevadas para dar variedad y color.

4. Diseño técnico

Antes de ponerse manos a la obra y crear un videojuego o cualquier proyecto sea en el ámbito que sea, hay que tener presente varios aspectos para tener claro cómo se va a organizar el trabajo, qué entornos de desarrollo y herramientas se van a utilizar, esquematizar las cosas y qué funcionalidades va a tener.
Si bien en puntos anteriores se han tratado algunos de los aspectos de los que ahora se tratará, pero en esta sección se profundizará sobre los temas.

4.1 Entorno elegido, requerimientos técnicos y herramientas empleadas

Para la realización del videojuego Lands Of The Gondria se decidió utilizar el software de desarrollo Unity 5, en concreto la versión 2018.3.10f1.

Para tener un control de versión y poder trabajar las dos personas del equipo, se ha decidido utilizar Unity Collaborate.
También se ha subido el proyecto usando Github para tener una copia de seguridad y que pueda acceder el profesorado.

Las justificaciones utilizadas para la elección de este y no otro motor gráfico para el desarrollo son las siguientes:

-A lo largo de este curso es el motor gráfico que se ha utilizado para la realización de las PECS y por lo tanto, ya se tiene una idea y concepto de cómo usarlo y sacarle partido.

-Unity se adquiere de forma gratuita y no hace falta desembolsar dinero inicial para empezar a crear un videojuego. Si bien cuenta con una versión de pago y profesional, que incluye alguna herramienta más que la que trae la versión básica (Por ejemplo información en tiempo real técnica como consumos de CPU, GPU, excepciones etc) además de soporte oficial, sino se superan los 100.000$ en ingresos no se está obligado a obtenerla.

-Cuenta con una comunidad detrás de usuarios para las dudas que pueden surgir durante el desarrollo del proyecto.

-Dispone de una tienda denominada *Asset Store* desde la cual se obtienen *Assets* (Recursos) tanto de pago como gratuitos para poder desarrollar nuestro videojuego si no se tienen ciertos conocimientos en otros ámbitos como puedan ser el modelaje de personajes, etc.

-Permite el uso de lenguaje C#. Este punto es bastante crucial, pues ambos conocemos el lenguaje.

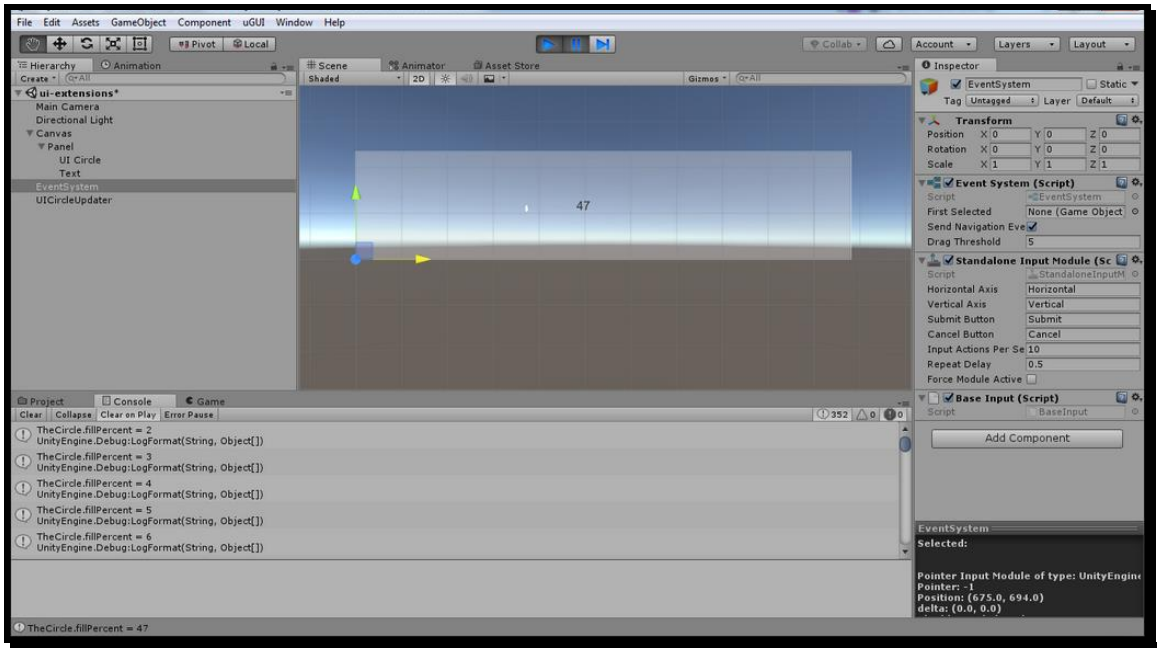
-Dispone de un editor visual muy útil, con una interfaz amigable, que se sabe utilizar y se conoce su funcionamiento.

-Gracias al Unity Collab, Unity va detectando los cambios realizados en ficheros o escenas y quedan marcados como modificados o añadidos. En la interfaz UI hay una herramienta para descargar cambios o subir los propios sin necesidad de programas de terceros.

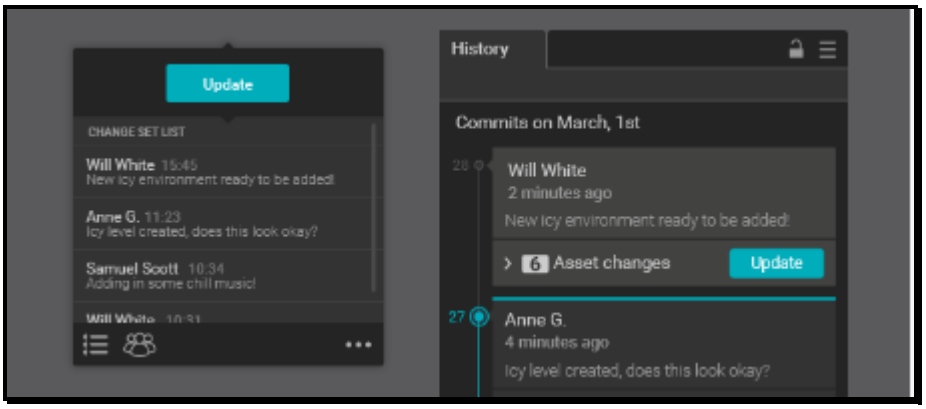
-El punto que mayor y más peso tiene para la elección es básicamente la experiencia que se tiene ya desarrollando con Unity.

Al hacer el videojuego para la plataforma de PC y con gráficos 3D, se requiere de un motor gráfico acorde.

La interfaz de Unity es la que se muestra en la ilustración inferior.



Para acceder a las opciones de Unity collab hay que ir a la parte derecha de la Interfaz.



Desde ahí se puede comparar ficheros, saber qué ficheros se han modificado o añadido y subir/descargar los cambios pertinentes.

Esta es una herramienta imprescindible para poder trabajar sin pisarse entre los miembros del grupo, puesto que te avisa si en caso de descargar algo te haría perder los cambios realizados.

Para la necesidad de crear texturas o modificas las ya existentes se ha utilizado el programa Adobe Photoshop, ya que dispone de multitud de herramientas para la modificación de colores para los modelos en el juego.

Otra herramienta utilizada ha sido la plataforma de Github, donde periódicamente se han ido subiendo los cambios en el proyecto.

Gracias a Github se tiene una copia del proyecto dónde los profesores pueden acceder. A su vez, también nos sirve como resguardo de seguridad del proyecto.

Con el programa de ofimática de Excel se creó una tabla indicando los tiempos de duración y realización de los hitos a realizar por parte de cada uno de los integrantes del grupo, esto es especialmente útil para tener un seguimiento de qué hay que hacer y cuánto tiempo se dispone para ello.

Para tratar sobre el tema del proyecto, se usaron dos software de comunicación de voz. Discord y Skype, ambas herramientas gratuitas que permiten la comunicación y permiten también el poder compartir la pantalla del

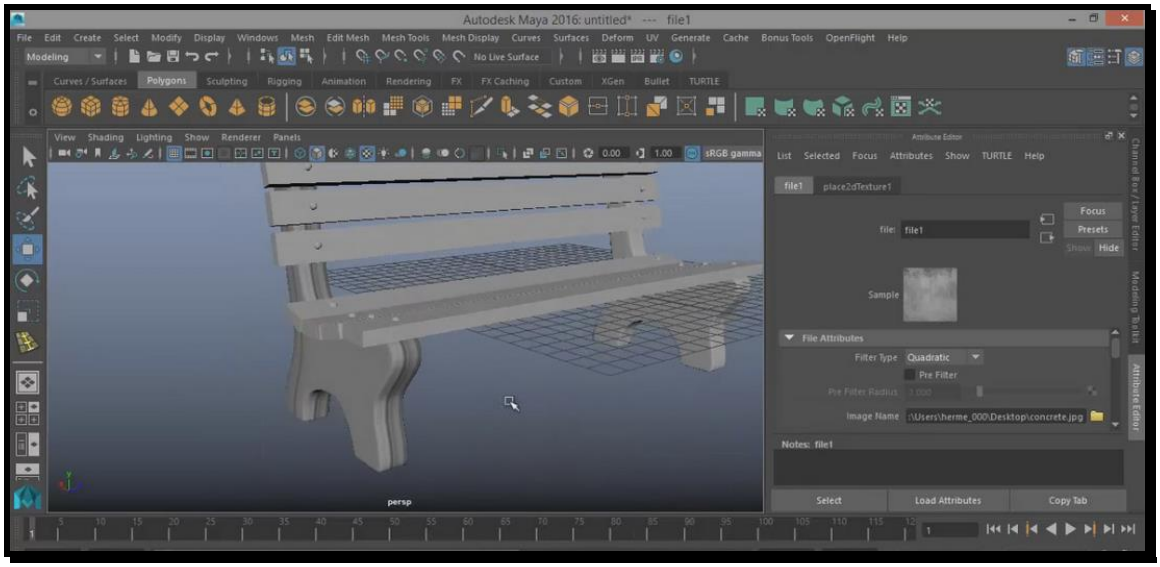
ordenador para, en caso de necesitar, enseñar cosas del proyecto al otro miembro del grupo etc.

(A la izquierda el logo de Discord, a la derecha el de Skype.)

El tema de sonidos en el videojuego, o música se han sacado los sonidos de bibliotecas web. A su vez, con el programa Cubase 8 se han hecho modificaciones pertinentes según necesidad tal como recortar audios, modificar velocidad o pitch etc.

Para las animaciones de los modelos 3D, se ha utilizado en mayor medida la herramienta de Mixamo y la que trae por defecto Unity. Esta herramienta cuenta con animaciones ya realizadas a modelo siendo posible su exportación.

Para modificar los modelos 3D, y ver la estructura que tenian, se ha utilizado el software Maya. Si bien únicamente se ha utilizado para algun momento puntual, como por ejemplo separar los modelos y personajes que venían en un pack, o hacer pequeñas modificaciones.



4.2 Assets

-Assets Modelos 3D y Entorno.

Los assets que se han elegido para el desarrollo del videojuego son modelos tipo Low Poly, es decir, que tienen un número reducido de polígonos. La principal razón de utilizar este tipo de gráficos se basa en dos puntos.

-Optimización: Al tener pocos polígonos se permite tener un mayor número de objetos o modelos en pantalla, estos no sobrecargan tanto el procesador gráfico permitiendo conseguir que los FPS (Fotogramas por segundo) se mantengan en valores óptimos. Además, permite la ejecución del videojuego en ordenadores que no dispongan de un hardware potente.

-Facilidad de expansión: En caso de necesidad, al no ser modelos demasiado complejos de realizar con un software de modelado, si se requiriese en un futuro se podrían crear nuevos modelos o editar los existentes.
Si bien, para el modelo del personaje, si se han hecho algunas pequeñas modificaciones tanto a nivel de modelo, como de texturas a usar, dando la posibilidad de cambiar el color de cierta ropa.

*(Los Assets de modelos 3D, tanto personajes como entorno han sido obtenidos a través de un amigo en común de los integrantes del grupo. Los mismos se pueden adquirir desde la tienda de Unity).

Polygon Nature

Estos Assets nos permiten crear desde cero un modelado para el terreno dotándolo de todo lo necesario para el videojuego y, en concreto, del primer Stage del mismo.

Así pues se ha utilizado desde crear lo más simple, como puede ser darle texturas varias al suelo, ya sea hierba o arena o como la creación de distintos tipos de árboles, arbustos, diversas plantas, montañas, rocas, agua para un rio etc.

Si bien no son “low Poly” del todo, puesto que el Asset de “Polygon Adventure” tiene arboles más en formato “low poly”, nos pareció correcta la implementación, dándole al mapeado un toque especial juntando elementos.

Como extra, el mismo paquete de modelos también incluyen algunas estructuras como puentes, estructuras de roca con formas rectangulares a modo de vigas o pilares, muros de piedra etc.

También incluye algún FX como son la simulación de espuma en una cascada, o las ondas del agua del rio.

Dentro del juego con los assets ya mencionados se ha creado dentro del primer Stage lo siguiente:

(Ilustración directa del videojuego desde Unity)

-Árboles independientes decorativos, bosques, texturas (suelo, paredes…)
-Montañas y rocas decorativas.
-Ríos, charcas y cascadas.
-Plantas varias por el entorno
-Texturas del suelo y paredes
-Objetos varios tipo cofres, vallas, campamentos base, antorchas…

(Ilustración del Asset de Polygon Nature la Asset Store)

Polygon Adventure

Estos Assets han sido utilizados para la gran mayoría de Npc's del juego, así como mobiliario tipo casas simples, algunas armas simples, u objetos decorativos para una ciudad (barriles, bancos, mesas…) y en general, dotar al mapa de detalles haciendo más realista los lugares.

Los Assets de Polygon Adventure se han utilizado para lo siguiente:

(Ilustración directa del videojuego desde Unity)

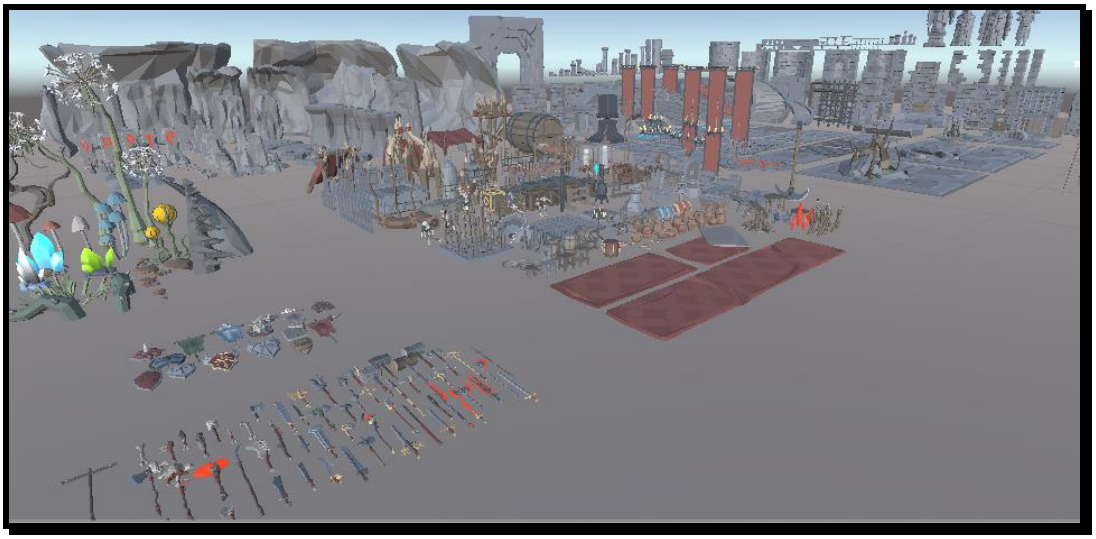
-Npc's varios (Bandidos, Caballeros, Campesinos…)
-Estructuras, casas y tenderetes
-Mobiliario vario, bidones, sartenes, mesas, cajas, leña, cortinas, etc.
-Armas simples como espadas, dagas…

(Ilustración del Asset de Polygon Adventure la Asset Store)

Polygon Dungeon

Estos Assets han sido utilizados para la zona de los enemigos, armas más poderosas en el juego, creación de diversas estructuras (casas, cuevas etc.) objetos creados con madera, como ruedas o viguetas simples. Además, trae varios efectos FX utilizados para las magias del jugador.

Con los Assets de Polygon Dungeon se ha creado lo siguiente:



(Ilustración directa desde Unity, Inventory Dungeon tile)

-Armas varias especiales (Espadas, escudos, lanzas…)
-Casas, edificio en ruinas
-Estructuras varias (casa derruida, bases enemigas)
-Material vario (Antorchas, plantas especiales, estructuras de madera)
-Modelo 3D del personaje principal
-Modelos 3D de NPc's varios (Guerrero escudo, muertos, orcos)

(Ilustración del Asset de Polygon Dungeon de la Asset Store)

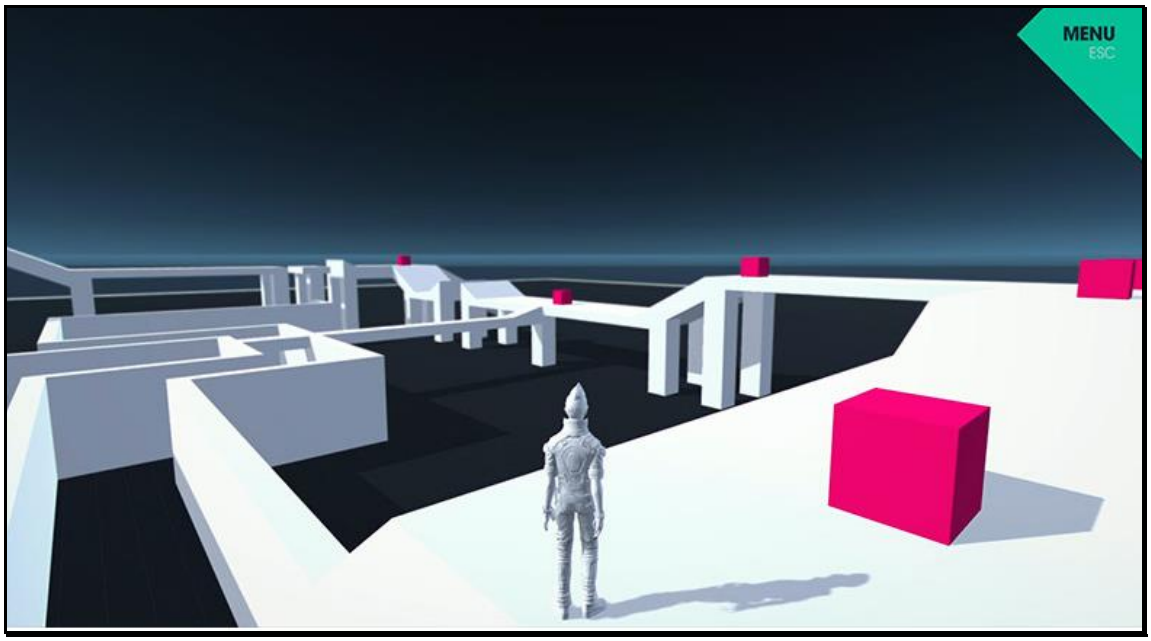
-Assets Control Personaje

Para el control del personaje principal se utiliza un asset externo denominado *Third Person Controller,* que viene incluido en el paquete de Standard Assets. Las funcionalidades que nos ofrece dicho asset son las expuestas a continuación:

-Mover al personaje en la dirección que mande el jugador a través del teclado.

- Posibilidad de saltar, y agacharse.
- Posibilidad de alternar entre correr y caminar.

Este Asset está disponible gratuitamente y se puede obtener en la Asset Store de Unity.



PostProcessing Assets

El *post-processing* se refiere al segundo render o retoque de imagen que se realiza después del primer render que se graba en un buffer de la memoria. El primer render trabaja la imagen total generando una textura, que después se carga y se vuelve a renderizar con efectos como luz, sombreado o desenfoque.

Algunas de las opciones que nos brinda el Asset son las siguientes:

-Fog: Posibilidad de niebla en la escena.

-Anti-aliasing: Eliminación o supresión de *dientes de sierra.*

-Efecto Motion Blur: (Desenfoque al mover la cámara).

-Bloom: Incremento de brillo en los pixeles cuyo valor sea superior al de un valor preestablecido.

-Viggnete: Oscurece bordes de la imagen.

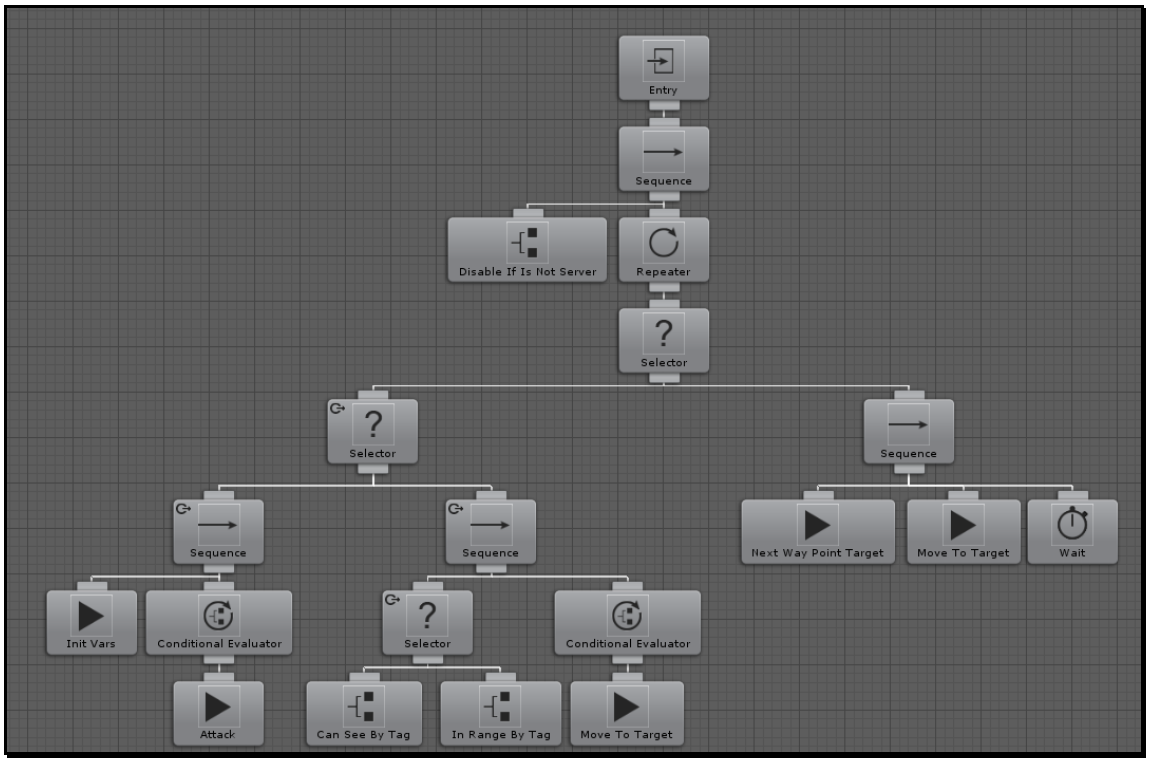
(Imagen sin post-processing de la pantalla inicial del videojuego).

(Imagen con post-processing de la pantalla inicial del videojuego).

En las imágenes superiores se aprecia la diferencia de la pantalla de inicio del videojuego con el Asset de Post-processing off/on.
Este asset está disponible gratuitamente en la Asset Store de Unity.

Behaviour Bricks

Este Asset se ha utilizado para la configuración y realización de la Inteligencia Artificial que usan los enemigos, y, en menor medida, los NPC que se van moviendo por el mapa.
Si bien en páginas posteriores se trata el tema de la IA, este apartado está enfocado a qué ofrece dicha herramienta, más que los usos dados en el juego.
A continuación adjunto una imagen de un árbol utilizado para la IA enemiga.



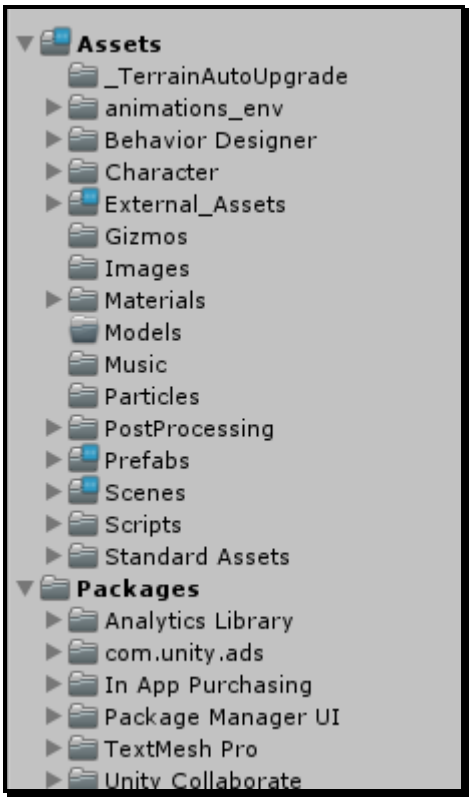
Como se puede observar, el mismo es secuencial y toma las ordenes de los eventos de arriba hacia abajo y de izquierda a derecha.

Cuando está en marcha se vuelve de color verde indicando por dónde va en tiempo real.
Este Asset ha sido comprado en la Asset Store de Unity.

4.3 Esquematización del videojuego

4.3.1 Estructura carpetas

La estructura de carpetas que se ha utilizado en Unity para distribuir todo lo necesario para la realización del videojuego es la siguiente.



Dentro del videojuego hay dos carpetas principales, la de Packages que son librerías propias de Unity y no se tratarán en esta memoria y los Assets, que hace referencia a la totalidad de utilidades que se han utilizado.

TerrainAutoUpgrade: Archivos varios de Unity.

Animations_env: Dentro de esta carpeta se encuentran la totalidad de las animaciones que utilizan las puertas en el juego, es decir, animaciones de entorno.

Behaviour Designer: Paquete del Asset que da nombre a la carpeta, en él se encuentran varios archivos y los diseños de la IA, Npc's y todo lo necesario para generar movimiento, ataques, etc. Estos archivos están distribuidos en varias carpetas.

Character: Carpeta con el personaje de prueba que se utilizó hasta la obtención de los definitivos. Se utiliza para pruebas al tener un controlador simple.

External_Assets: Dentro de esta carpeta están los Assets externos para temas de Modelado de escenario.
La misma carpeta tiene sub-carpetas para separar y diferenciar los Assets incluidos.

Esas carpetas son Nature_pack, nature pack(utilizada para guardar cierttos modelos), PolygonAdventure y PolygonDungeon, los ya mencionados en anterioridad.
Gizmos: Carpeta para guardar pruebas de BBricks.

Images: Utilizada para guardar las imágenes que se utilizan en el juego como pueden ser la de mirilla in-game, título del juego etc.

Materials: Materiales varios creados para dar color a ciertos GO simples creados por nosotros.

Models: Por si se importaban modelos 3D de otros Assets (No contiene nada).

Music: Sonidos y música del juego.

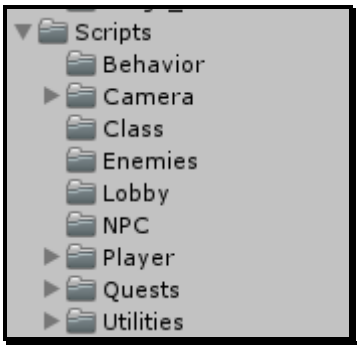
Particles: Aquí están las partículas que se han modificado de los Assets Originales.

PostProcessing: Asset de Post-procesado, incluye ficheros varios necesarios para su correcto funcionamiento.

Prefabs: Todos y cada uno de los prefabs que se han creado a través de otros prefabs de los Assets, o el conjunto de varios, como puede ser una puerta con un marco alrededor, el jugador, enemigos etc…

Scenes: Carpetas con las escenas que hay en el juego, tanto para escenas de prueba como para los escenarios.

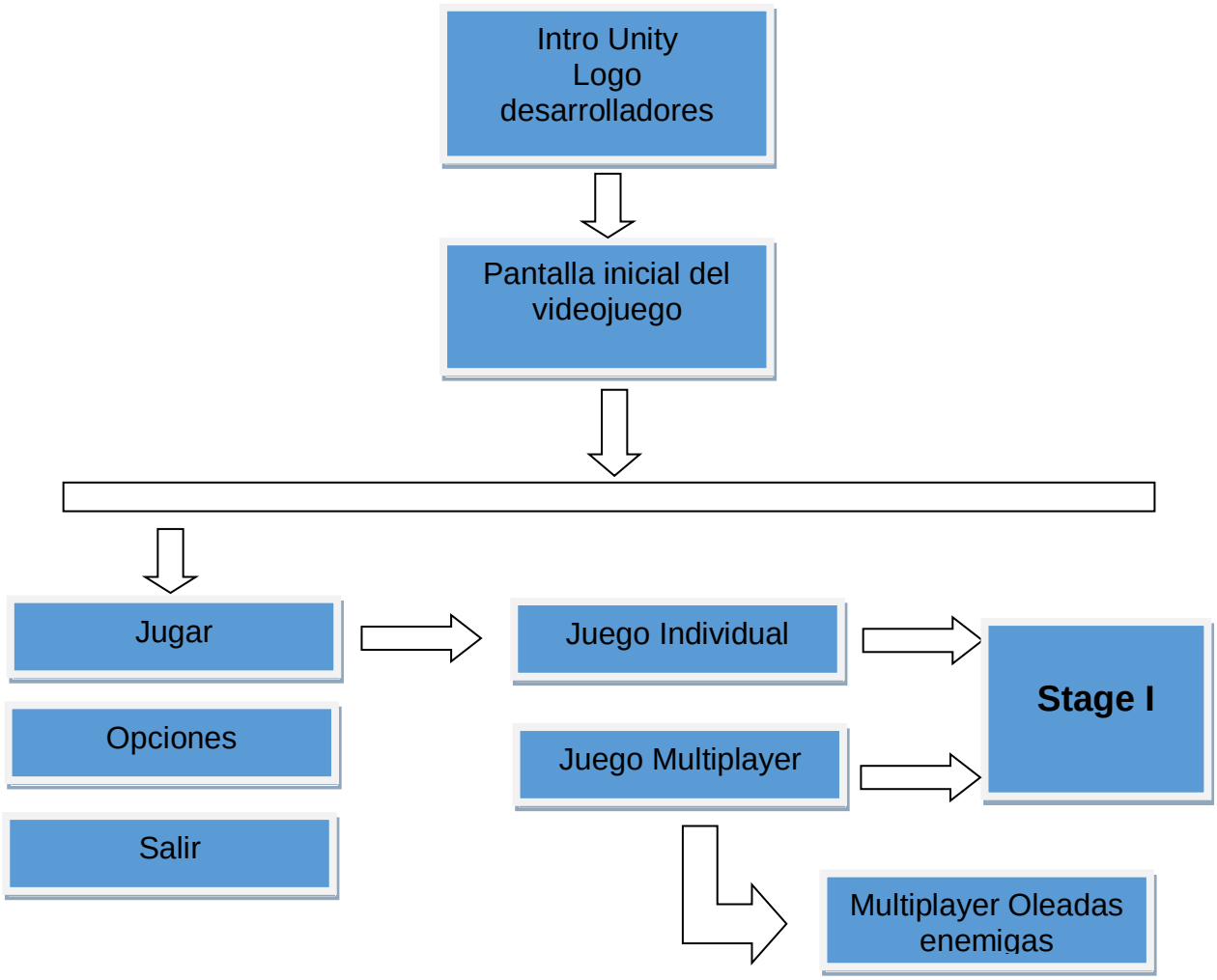
Scripts: Esta se puede considerar la carpeta más importante del proyecto, la que contiene todos los archivos que hacen que el juego tenga vida y sea usable. Dentro de la misma, hay varias sub-carpetas dentro.



Como se observa en la imagen, por tal de no tener los scripts juntos se han dividido según el uso o la función, ya sea para los Enemigos, Classes, Lobby, Npc's, jugador etc.
La carpeta de Utilities sirve para los que no se pueden clasificar o se ha utilizado directamente la carpeta raíz para la ubicación de esos mismos archivos.

Standard Assets: Paquete de Assets que viene por defecto en Unity, con un controlador básico de Movimiento para la 3º persona que ha sido editado por los desarrolladores de este proyecto para su adaptación al videojuego.

4.3.2 Distribución de Escenas



4.3.3 Estructura de Clases C#

Para tener un control y evitar repetir código para varios de los aspectos del videojuego, se ha creado un sistema de clases en el mismo.
Este sistema básicamente se basa en la herencia y la extensión de sub-clases.

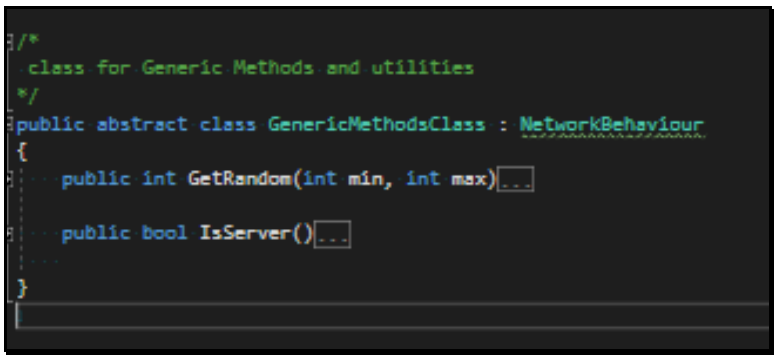
Como el videojuego se planteó de forma Online, se creó una Clase Interfaz que tenía los métodos para saber si el usuario se conectaba como cliente o servidor. Esta clase se denominó "*GenericMethodClass*" la cual extendía de *Network Behaviour*.
A su vez, esta clase tenía varios métodos para poder o no, utilizar posteriormente como devolver una bool indicando si la conexión es de servidor/usuario, obtener numero aleatorio etc.
Si bien, muchos otros GO del videojuego tienen clases independientes que no tienen herencia salvo del Behaviour de Unity propio.

Desde esta clase, se extienden dos más, que son las siguientes:
-MainEnemyClass
-MainPlayerClass

Ambas son una clase abstracta que hereda de *GenericMethodClass*, en la que se definen varias funciones necesarias para todo lo que envuelve tanto al jugador como a los enemigos.
Además de funciones, también tienen parámetros como la vida, el maná (Para el caso del jugador) o declaradas varias constantes.

Para el caso de los enemigos, la clase que hereda de *MainEnemyClass* es la denominada *EnemyManager* que se encarga de varios aspectos relacionados con el daño que recibe el enemigo por parte del jugador, el rango que tiene desde su punto inicial de aparición, si tiene a rango al jugador para poder realizar un ataque, quitar vida, etc.

Para el caso del jugador, éste tiene muchas más clases que si no necesitaban herencia se hicieron independientes, pero las que sí, heredaban de MainPlayerClass.
Adjunto imágenes de ejemplo para el jugador.



(Clase Genérica que hereda de *NetworkBehaviour*)

```
6
7   public abstract class MainPlayerClass : GenericMethodsClass
8   {
9       //HUD CONSTANTS
10
11      public const int LIFE_HUD = 1;
12      public const int MANA_HUD = 2;
13      public const int EXPERIENCE_HUD = 3;
14      public const int LEVEL_HUD = 4;
15      public const int LIFE_SELECTION_HUD = 5;
16
17
18      public const string GAMECONTROLLER_TAG = "GameController";
19      public const string ENEMY_TAG = "Enemy";
20      public const string PLAYER_TAG = "Player";
21      public const string NPC_TAG = "NPC_TAG";
22
23      protected bool selection = false;
24      protected GameObject goSelected = null;
25      public bool HasSelection()...
29
30      public void SetSelectionToTrue(GameObject go)...
35      public void SetSelectionToFalse()...
40
41
42  }
```

(Clase *MainPlayer* que hereda de la clase Genérica)

```
5
6   public class PlayerDialogues : MainPlayerClass
7   {
8
9       //public Canvas dialogue;
10      public Text main_text;
11      public Text next_text;//aux
12
13      private string textToShow = "";
14      private string textAux = "";//aux
15      private List<string> dialogAux = new List<string>();
16      public GameObject dialoguesManager;
17
18      private int cnt = 0;
19
20      private string goNpc_name;
21
22      private bool npc_selected;
23
24      public GameObject chld;
25      public GameObject ply;
26      private GameObject auxTarget;
27
28      public GameObject faux;
29
30      // Start is called before the first frame update
31      void Start()...
37
38      // Update is called once per frame
39      void Update()
40      {
41          if (Input.GetKeyDown(KeyCode.E) && npc_selected)...
45          DoAction();
46          RayCastAction();
47
48      }
49
50      void Fill_text() ...
56
57      void Next_text() ...
71
72      public void OnOffDialogue(bool k)...
76      public bool DialogueState()...
80
81      public void StartDialog(string name)...
91
92      public void RayCastAction()...
136
137     public void DoAction()...
164 }
165
```

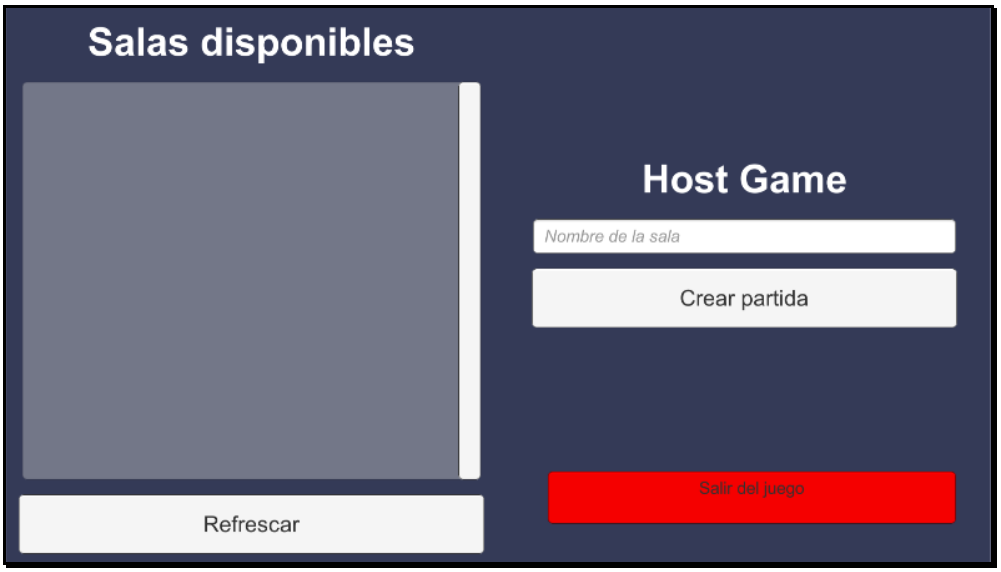
(Clase Específica para los Diálogos del jugador, que hereda de MainPlayer)

5. Funcionalidades Implementadas en el videojuego

El videojuego cuenta con una gran variedad de implementaciones que hacen posible la diversidad de aspectos y tareas que podrá realizar el jugador. Si bien en muchos aspectos únicamente se ha implementado un tipo de hito para cada uno de estos aspectos con únicamente el fin de realizar la integración de dicho objetivo.

Sistema Online

El videojuego cuenta con un sistema online que, muy a nuestro pesar no ha sido posible la correcta implementación en el videojuego dado muchas complicaciones en lo que a sincronización cliente-servidor se refiere. De todas formas, creemos oportuno incluir las partes en la memoria puesto que ha sido un trabajo realizado de igual manera.
El Host de la partida, puede crear una sala y un cliente puede unirse, a través de una amigable interfaz fácil e intuitiva de utilizar.



Como se aprecia en la fotografía el sistema es simple. un jugador crea una partida asignándole un nombre y el cliente al abrir el juego le aparecerán las partidas disponibles, sólo debe buscar la que le interesa para poder unirse.

Si las salas están llenas, estas no aparecerán. El número máximo de jugadores por sala son dos. En caso de que no aparezca la sala, al clicar en el botón *Refrescar* se realizaría otra petición al servidor para cargar las salas disponibles.
Una vez tanto el Host como el Cliente están en partida, ambos tienen un Identificador de Red. Para nuestro caso, en los scripts que controlan el tema del *multijugador* siempre utilizamos las referencias de si se es cliente o se es host, ya que nunca puede haber más de un cliente por partida.

Como tal el videojuego implementa dos modalidades en línea. La primera modalidad es ser acompañante del personaje principal, es decir, un cliente puede unirse y tendrá un personaje idéntico al del host. Su función será la de ayudarle a completar las misiones acabando con los enemigos, puesto este cliente no podrá realizar misiones. (No se implementa funcionando al 100%)

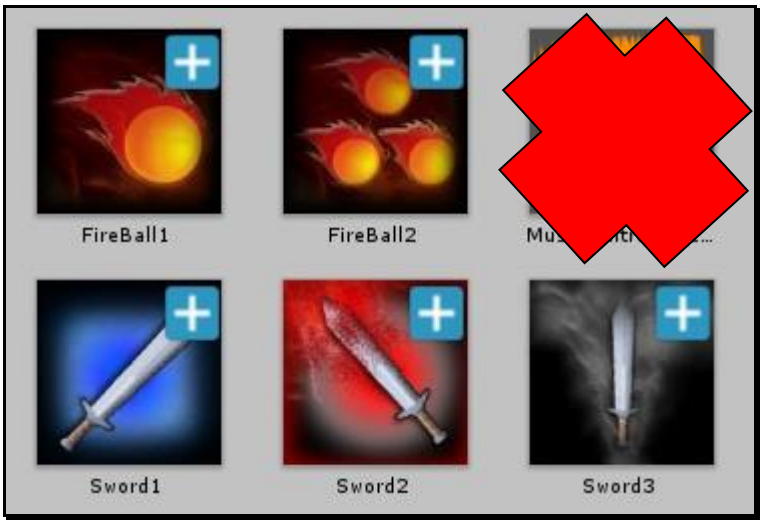
UI del videojuego

La interfaz que plantea el videojuego es bastante simple para no molestar al usuario con demasiados detalles, apareciendo únicamente por pantalla lo necesario en cada momento.
Para la parte de la vida del jugador, aparecen dos barras en la zona superior izquierda de la pantalla. La verde se corresponde a la vida, y la azul, se corresponde a la energía o maná del jugador.
Estos valores decrecen si nos atacan enemigos o si utilizamos habilidades especiales. A su vez, tienen una regeneración aproximada de +-1p cada segundo.
Cuando se tiene un enemigo en visión aparece en la zona superior-centro, la vida del enemigo.

(Ui del juego donde se aprecian las habilidades y la vida/energía)

Otro de los elementos que serán visibles a partir de cierto punto, son las habilidades especiales. El juego cuenta con un total de cuatro, siendo dos de estas ataques físicos y dos mágicos.
Las habilidades, tal como muestra la fotografía suprior se encuentran en la zona de debajo de la pantalla, en el centro. La que está marcada de color azul representa que es la que se tiene elegida y en caso de hacer *click* derecho con el ratón la que será efectuada por el jugador.

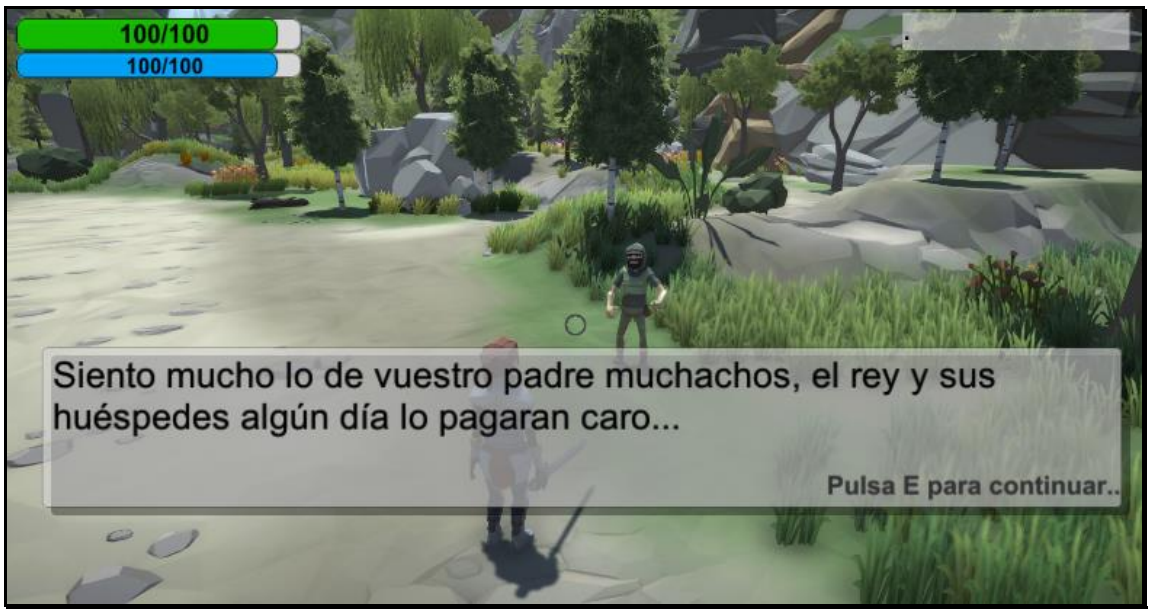
Se diseñaron 4 iconos para estas.



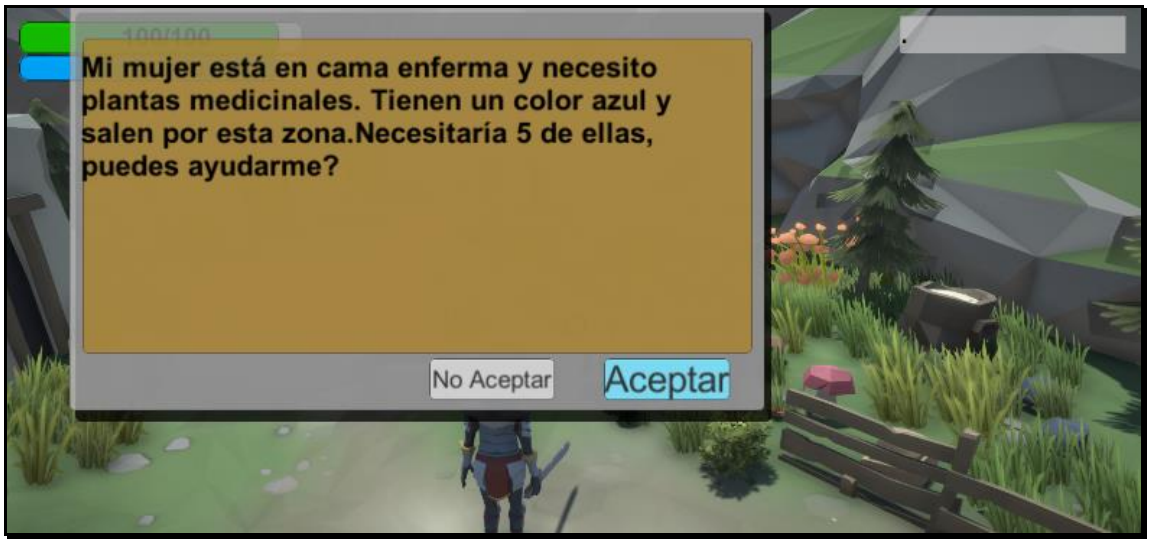
(Miniaturas de los ataques disponibles)

Con la creación de los iconos se busca que el jugador asemeje el tipo de ataque a efectuar con el icono. Por ejemplo, las bolas de fuego lanza una o tres, siendo este el número de bolas que hay en la imagen, la espada con una especie de tornado detrás el golpe de giro, y la roja el golpe potente.

Para el tema de los diálogos del videojuego con los diferentes NPC del juego, aparece un pequeño recuadro indicando al usuario que puede interactuar con el NPC. Una vez se establece conversación, aparece un diálogo con el personaje y más indicaciones para el usuario, para continuar leyendo o finalizar la conversación una vez terminada.



Sobre la UI referente a las misiones, sigue un poco la misma estela que los diálogos en el primer punto de interacción.
Al entablar conversación con los NPC que dan misiones, aparecerá un diálogo con la tarea a realizar, y dos botones para aceptar o no aceptar dicha misión.



Arriba a la derecha, hay un pequeño diálogo que sirve de “ayuda” para saber por ejemplo cuantas plantas de las 5 necesarias llevas en cada momento.

Si un enemigo acaba con la vida del jugador, aparecerá un mensaje de muerte en pantalla y un botón para reiniciar, igual que si pulsamos la tecla Escape del teclado aparecerá un botón que nos redirigirá a la pantalla de inicio del juego.

Sistema de Combate

El sistema de combate del videojuego se conforma tanto de ataques físicos, como mágicos. Si bien no se han implementado una gran variedad de ellos, si que son suficientes como para darle al jugador varias opciones de acabar con los enemigos que se van presentando a lo largo de la aventura.

Para golpear con la espada y efectuar un golpe simple, basta con hacer *click* izquierdo con el ratón (Siempre y cuando no estés hablando con algún NPC o con algún diálogo abierto).
Como tal, el ataque básico tiene una potencia de golpe bastante baja, pero a cambio no consume nada de la barra de mana/energía.
Los otros ataques se efectúan con el *click* derecho, siendo estos más dañinos pero a cambio, consumen el recurso de energía/maná. Esta barra es

compartida, de ahí que haga referencia a ambos aspectos, en el caso de maná sería cuando se habla de un ataque mágico y de energía cuando se hace sobre uno físico.

En el videojuego hay disponibles dos ataques físicos, que son los siguientes:

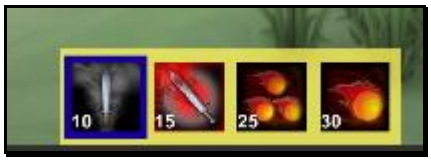
Doble espadazo: Realiza dos cortes con la espada, tiene una potencia base de 15 puntos de daño real al enemigo, el icono del ataque es el siguiente.
Su coste de energia es 15.

Espadazo tornado: Realiza un giro sobre sí mismo y blande la espada en semi-circulo. Este ataque es ideal para cuando se tiene más de un enemigo en frente. Tiene un daño real al enemigo de 12 puntos y su coste de energía es de 10 puntos.

Bola de fuego: Lanza una potente bola de fuego hacia delante, infligiendo al enemigo 25 puntos de daño. El coste de maná de esta habilidad son 30 puntos.

Triple descarga de fuego: Lanza tres bolas de fuego hacia delante o la dirección en la que apunte el personaje. El daño real al enemigo son 25 puntos y el coste en maná 25.

La habilidad que se tiene seleccionada en el menú de habilidades aparecerá rodeada de un marco azul, dandole información al usuario de cual tiene en selección, tal y como se puede apreciar en la siguiente fotografía.



El númerito en color blanco es el coste de maná/energía de la habilidad.

La forma en la que se efectuan los ataques funcionan mediante una animación de mixmamo, para el caso de los físicos. Entonces, cuando efectua el jugador

el ataque y los colisionadores detectan al enemigo, el script que controla el ataque accede al script que quita vida al enemigo, restandole el valor pertinente.
Para los mágicos, estos instancian la bola de fuego y le dan una fuerza hacia delante, y si detectan enemigo le hieren. Estas bolas tienen una duración de vida de 2 segundos, una vez se instancian se llama a la función Destroy y se le da un tiempo de vida (*t)*.

Sistema de Movimiento

El sistema de movimiento del videojuego es simple, con las teclas predefinidas "WASD" mueves al personaje, con la Espacio saltas, y C para agacharse.

Al haber usado un asset externo, los controles ya vienen predefinidos, como tal se han efectuado algunos cambios en los scripts para, por ejemplo, hacer que al girar el personaje vaya un poco más despacio, no poder subir por las paredes etc.

Sistema Inventario

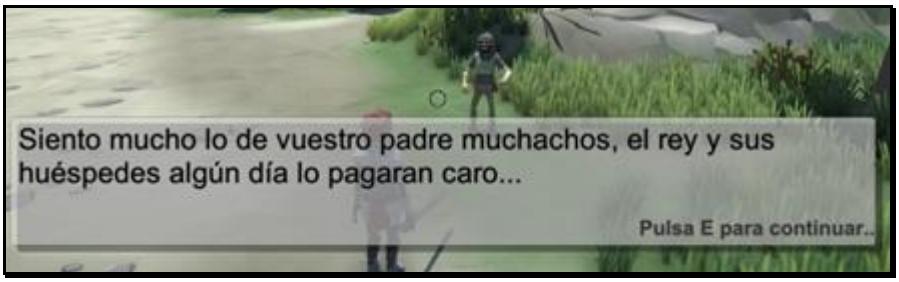
El jugador tiene un inventario para poder almacenar los objetos o llaves que va consiguiendo a lo largo de la aventura, pero no es visible para el jugador. Se trata de dos arrays, uno para objetos que posteriormente serán destruidos (como por ejemplo plantas recogidas para una misión) y otro para objetos que son persistentes ya que se pueden seguir utilizando (por ejemplo, una llave).

Sistema de Diálogos

El sistema de diálogos permite que el jugador pueda interactuar con varios (que no todos) de los NPC que están distribuidos por el mapa. Estos NPC's pueden dar información útil al jugador, como dónde está algún pueblo o ciudad, o el porqué de algo. Se utilizan para irle contando al jugador un poco la historia y darle el toque de RPG al videojuego.

El funcionamiento es bien simple, cuando tienes al NPC en el punto de mira se activa el *canvas* que controla la ayuda al usuario, mostrando un mensaje para poder hablar con él a través de la pulsación de una tecla, en el caso de este juego se utiliza la *F* para entablar conversación.
El jugador NO puede realizar preguntas al NPC o responder nada, únicamente es él el que te habla.

El Dialogo se muestra con un panel y el texto de lo que dice ese NPC, aparece también la ayuda para pasar entre textos o, si no tiene nada más que contar, cerrar la conversación.



La manera que se ha utilizado para distinguir los personajes, es a través del TAG. Se ha creado uno específico para los que tienen diálogos llamado "*npc_dialog*".

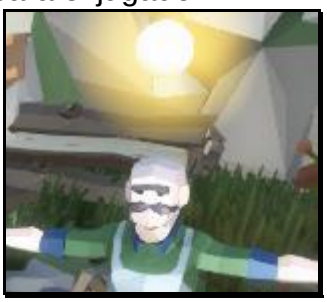
Este sistema está creado en base a un diccionario en un Script que almacena un clave-valor, es decir, cada personaje tiene un nombre (por ejemplo "NPC_0015") que se utiliza como clave y luego una lista de *strings* con los textos a decir. El personaje tiene un script que, al ser cargado accede a este diccionario y a través del nombre almacena los textos que le tocan y luego, muestra al usuario cuando este entabla conversación.

Sistema de Misiones

El sistema de misiones del videojuego permite al jugador tener un hito a lograr/ cumplir. Es lo básico de un videojuego de esta índole, ya que si no, no tendría nada que hacer durante la aventura.

Como se vio anteriormente en la sección de UI, al hablar con un NPC que ofrece misión aparece el texto que indica la tarea a realizar y dos botones interactivos. El sistema que se usa para almacenar los textos y mostrarlos es idéntico al de los diálogos descrito anteriormente .Únicamente cambia la manera de mostrar el texto. La manera de saber que el NPC ofrece misión y para que el jugador pueda distinguirlos rápidamente de los demás, es a través de una bola de varios colores que éstos tienen encima de la cabeza.

Color amarillo: Si la bola aparece de color amarillo, quiere decir que ese NPC tiene disponible una misión para el jugador.

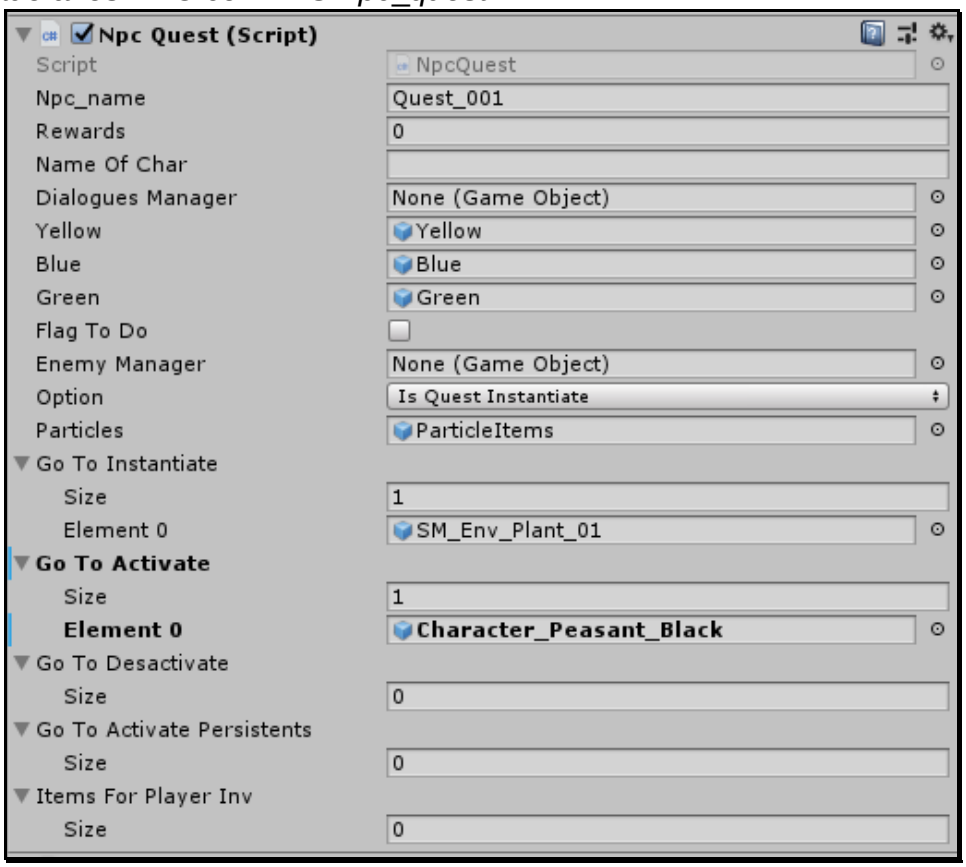


Color azul: Si la bola aparece de color azul, esto quiere decir que ese NPC está a la espera que termines la misión. Es decir, estás en el progreso de la misma.

Color Verde: Quiere decir que ya has completado la misión de ese personaje y que, por lo tanto, no tiene más misiones para el jugador.

Si tuviese más misiones, simplemente aparecería otra vez en color amarillo. Esta implementación se buscaba que fuese de tipo diegética.

Los NPC tienen un Script que controla todo el tema de qué tipo de misión es, si debe o no instanciar algún objeto, en caso de que el jugador la complete hay que activar algún GO o desactivarlo etc. Todo ello se controla desde el script asociado a los NPC con TAG *npc_quest*.



Se explica a continuación parte de lo que contiene el script.

Como tal, hay 4 tipos distintos de misiones que el jugador puede encontrarse y se seleccionan desde el selector de *Option* en el script superior.

- **Misión de recoger objetos:**

Estas misiones se basan en que el NPC que te la ofrece tiene que recoger ciertas cosas por el mundo, por ejemplo 5 plantas medicinales o setas.
Básicamente cuando se activa esta misión, se instancian los ítems que se encuentran en la lista de "*GO To Instantiate".* Las coordenadas de estos objetos están también guardadas en un fichero y se localizan a través de un diccionario utilizando el nombre del npc y la misión que ofrece como clave.
Una vez se instancian, a cada ítem se le asigna un *colisionador* con *trigger* y un script, que cuando el jugador pasa por encima se le suma en su inventario de tipo misión. Cuando tienes todos los objetos en tu inventario y vuelves a hablar con el NPC, este comprueba tu inventario y mira si el número total de instancias de ese objeto coincide con el total de objetos de ese tipo del *array* del jugador de inventario de misiones.

(Imagen de ejemplo de objeto instanciado para ser recogido)

Tal y como se aprecia en la imagen superior, todos los objetos instanciados tienen un sistema de particulas añadido para que sean fácilmente reconocibles por el usuario.

- Misión de Buscar Objetos:

Este tipo de misión, se basa en el mismo sistema que las de recoger objetos, de hecho se podría decir que es lo mismo con un pequeño cambio. En estas, se plantea que el usuario únicamente busque un objeto determinado y no un conjunto, además suelen incluir el tener que acabar con varios enemigos ya que se instancian en zonas donde hay bandidos.
Otro aspecto que tiene, es que están pensadas para que al jugador le den una recompensa ya sea una llave, magia nueva, armas etc. o que tal vez, por ejemplo, sea de buscar una rueda para un carro que si no lo reparas no se puede continuar a la siguiente zona, o buscar una llave para abrir la puerta y liberar a alguien a que te ayude a progresar. Es decir, "son misiones que se necesitan realizar para pasar de zona.".

- **Misión de Matar enemigos:**

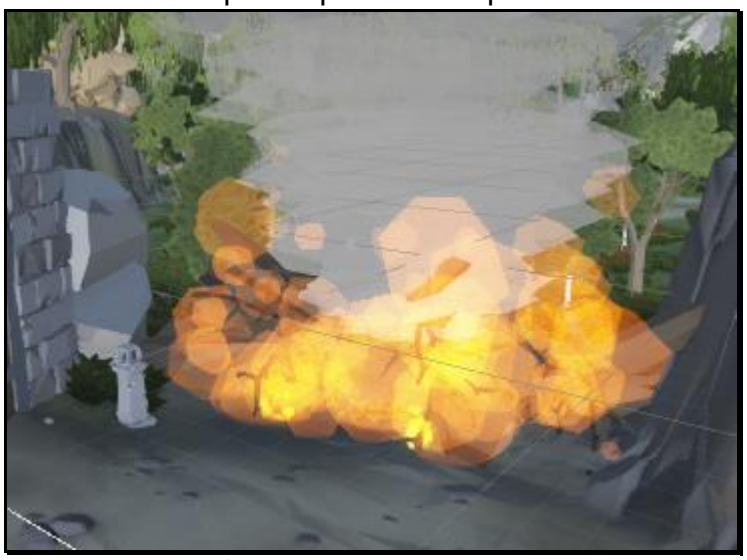
Tipo de misión por excelencia en todos y cada uno de los videojuegos de hoy en día, tienes la misión de terminar con ciertos enemigos de una zona, simple y llanamente. Estos enemigos se instancian igual que las de tipo recoger objetos. Y pueden o no dar acceso a otros GO, dar ítems etc.

- **Misión de hablar con NPCS:**

Estas misiones son simples, se trata de que al activar la misión se activa el GO (se instancia o activa, depende) con el que se debe hablar. No hay que hacer nada más, simplemente buscar al NPC y hablar con él. Un ejemplo sería el de tener que llevar una cantimplora a un NPC o buscar a un niño perdido en un bosque.

- **Misión Especial:**

Este tipo de misión son las que incluyen algún efecto FX por el camino, como por ejemplo por la mitad del juego cuando se debe buscar a una persona para que queme unos matorrales que no permiten el paso.



(Misión especial con efectos FX.)

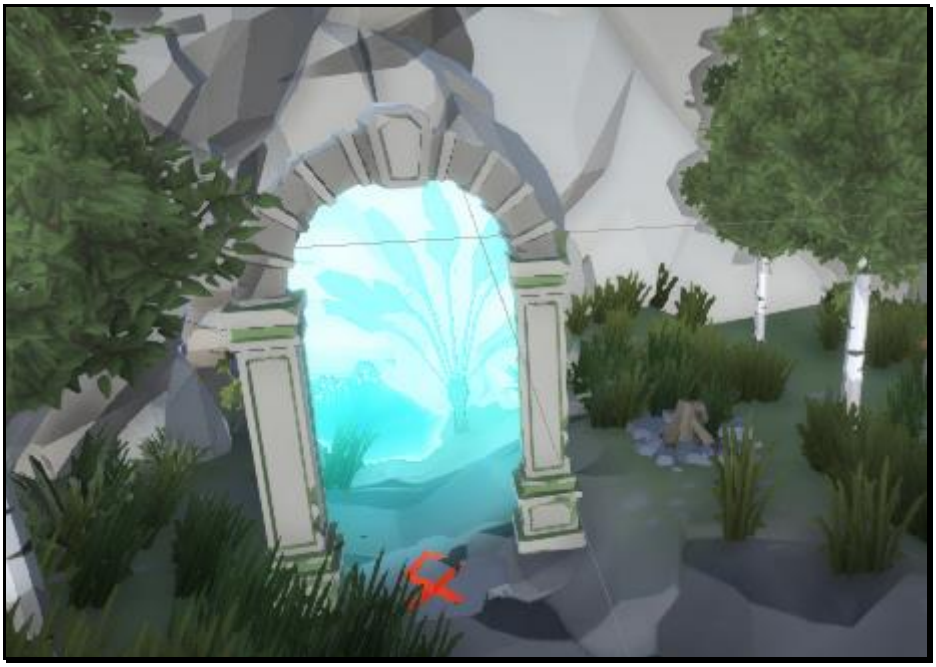
Como se mencionó anteriormente, algunas misiones pueden dar ítems especiales como llaves o piedras mágicas para poder atravesar puertas o portales.
Los demás campos del script, como los GO persistentes son GO que al realizar cierta misión deben desaparecer para siempre o moverse de lado, como por ejemplo la imagen superior, donde una vez quemada la maleza desparecería para siempre o un carro que se mueve y pasa de estar varado bloqueando el paso en un puente, a apartado permitiendo que el jugador avance.

Sistema de portales y puertas

El sistema de portales y puertas, es la manera de pasar por distintas zonas o *tele transportarse* de un lugar a otro. Para el caso de los portales, el juego cuenta con uno simple que no requiere tener ningún ítem especial, y se encuentra en la primera fase del juego. Este está para pasar de un lugar a otro de la granja, simplemente al estar cerca de la puerta, esta te detecta con un *colisionador* y te *tele posicionará* a la otra posición.

Para los portales mágicos que requieren llave, aunque el sistema es el mismo tienen un aura azul indicando que por ahí se pasa a otro lugar. Si requiere llave el script adjunto al GO que te detecta, hace un chequeo de tu inventario de

llaves, y si comprueba que el nombre de la llave dado en ese GO, está en poder del jugador, permite el *teleport*.



(Efecto de degradado azul en un portal, forma visual de indicar al jugador que se puede atravesar)

Para las puertas el sistema es parecido en lo que a llaves se refiere, pero no hace falta que el jugador tenga que realizar acción alguna para abrirlas. Cuando el GO de la puerta lo detecta, se abre automáticamente a través de una animación, y se cierra automáticamente cuando salta la función de *OnTriggerExit* siendo *player* el que sale.

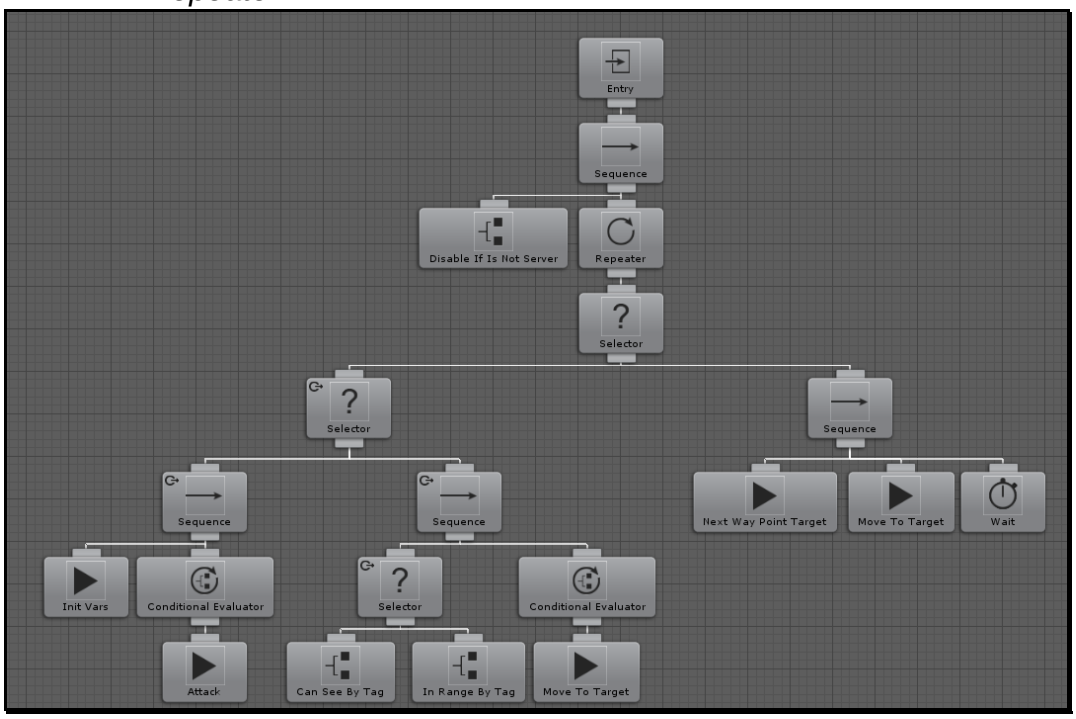
Sistema de Enemigos e IA ENEMIGA

Los enemigos que hay en el juego se van moviendo por varias posiciones, a su vez hay algunos que siempre están en un lugar y, si los matas, reaparecen al tiempo y otros que únicamente se instancian por misión. Todo esto lo maneja un *EnemyManager.* El sistema de Inteligencia Artificial que utilizan los enemigos se ha efectuado mediante el Asset de Behaviour Bricks. Su funcionamiento y el de los enemigos es el explicado a continuación.

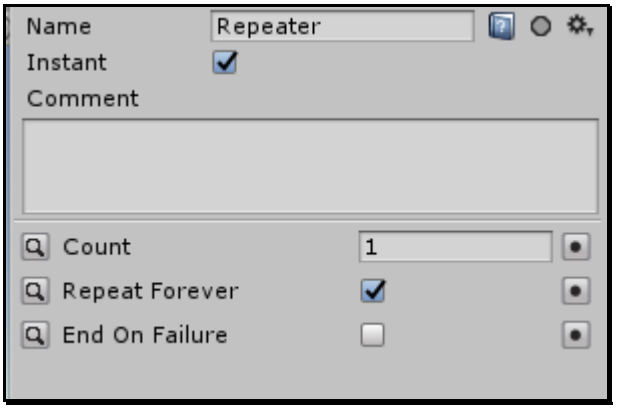
Como todo *Behaviour Tree*, se configura la IA a base de nodos, en este caso el curso de los nodos es de arriba hacia abajo y de izquierda a derecha.

Como podemos ver en la siguiente imagen, lo primero el primer nodo que tenemos cuando se activa la IA es un "*sequence*" que tiene como hijo dos nodos, "*Disable if is not server*" y "*repeater*", lo que hará esto es que si la IA no se está ejecutando en el servidor el primer nodo lo detectara y devolverá un fallo a su nodo padre para que no continúe ejecutando la IA. Si en el caso contrario esta IA se está ejecutando en el servidor, el primer nodo devolverá un

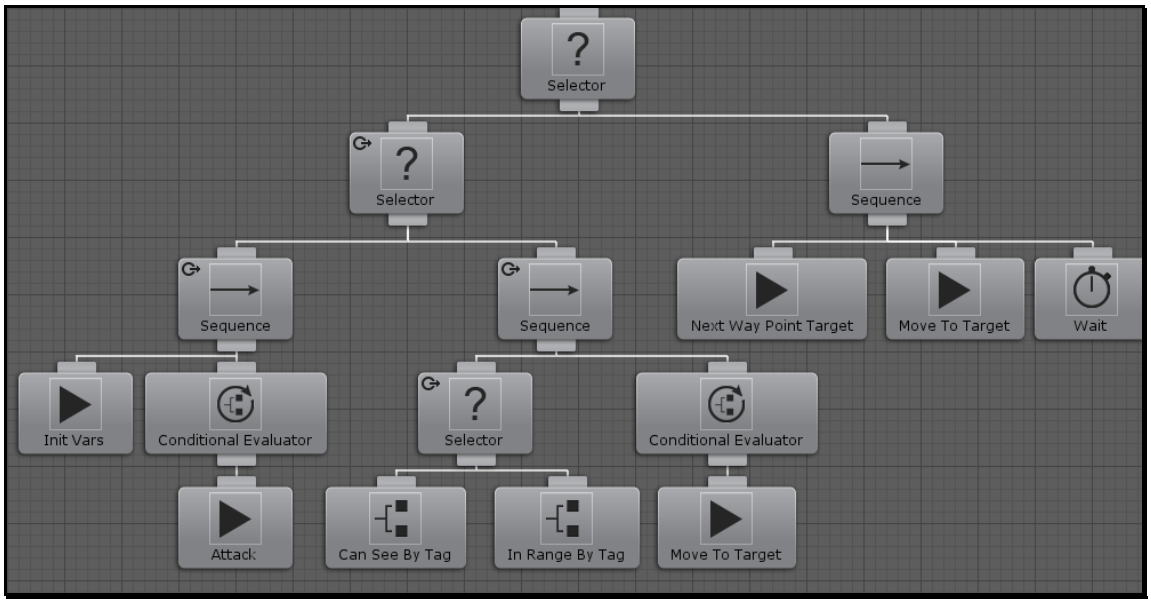
"*Success*" lo que hará que el nodo padre siga la ejecución de los nodos, en este caso el "*repeater"*.



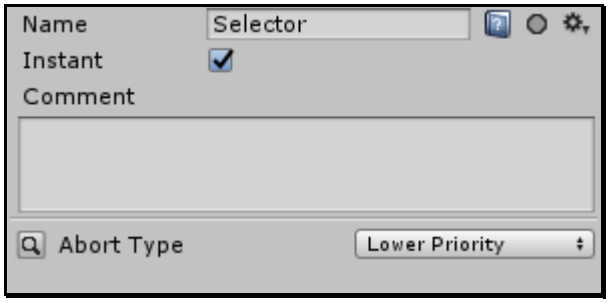
Este "*repeater*" tiene la configuración de "*repeat forever*" a true, tal y como el nombre indica siempre va a ejecutar a sus nodos hijos.



Debajo del "*repeater*" tenemos un nodo "selector", que vendría a ser la condición OR en un IF, esto va a hacer que si el primer nodo que tiene como hijo da un fallo ejecutará su segundo hijo para ver si ese también da fallo o no.

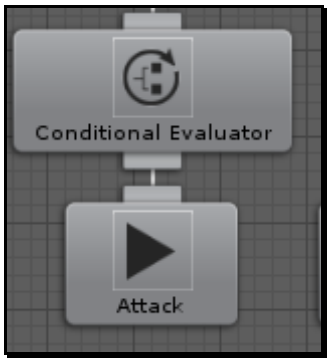


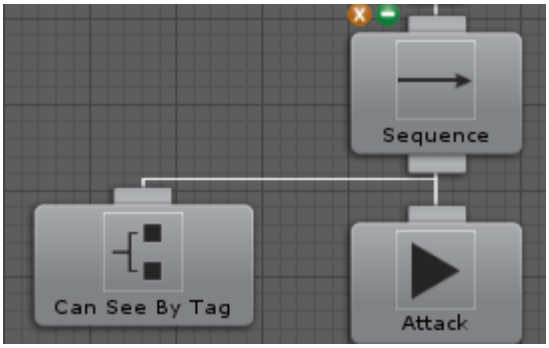
Como podemos ver todos los nodos principales que tenemos en la parte izquierda tienen un símbolo de un circulo con una flecha, esto está indicando que son nodos "*Low priority*", esto va a hacer que aunque devuelva un fallo en el siguiente *frame* volverá a ejecutarlo para comprobar si el resultado es correcto o es fallido. Como ya hemos explicado antes al tener como padre un "*Selector*" si el primer nodo da fallo hará el siguiente, pero al tener el primero en "*low priority*" si en algún momento durante la ejecución del segundo nodo el primero no devuelve fallo dejara de hacer el segundo nodo y se pondrá a ejecutar el primero sin importar el estado del segundo. Como podemos ver nuestro primer nodo también tiene dos nodos más como hijos también marcados como "*low priority*".



(Aplica low priority desde BT)

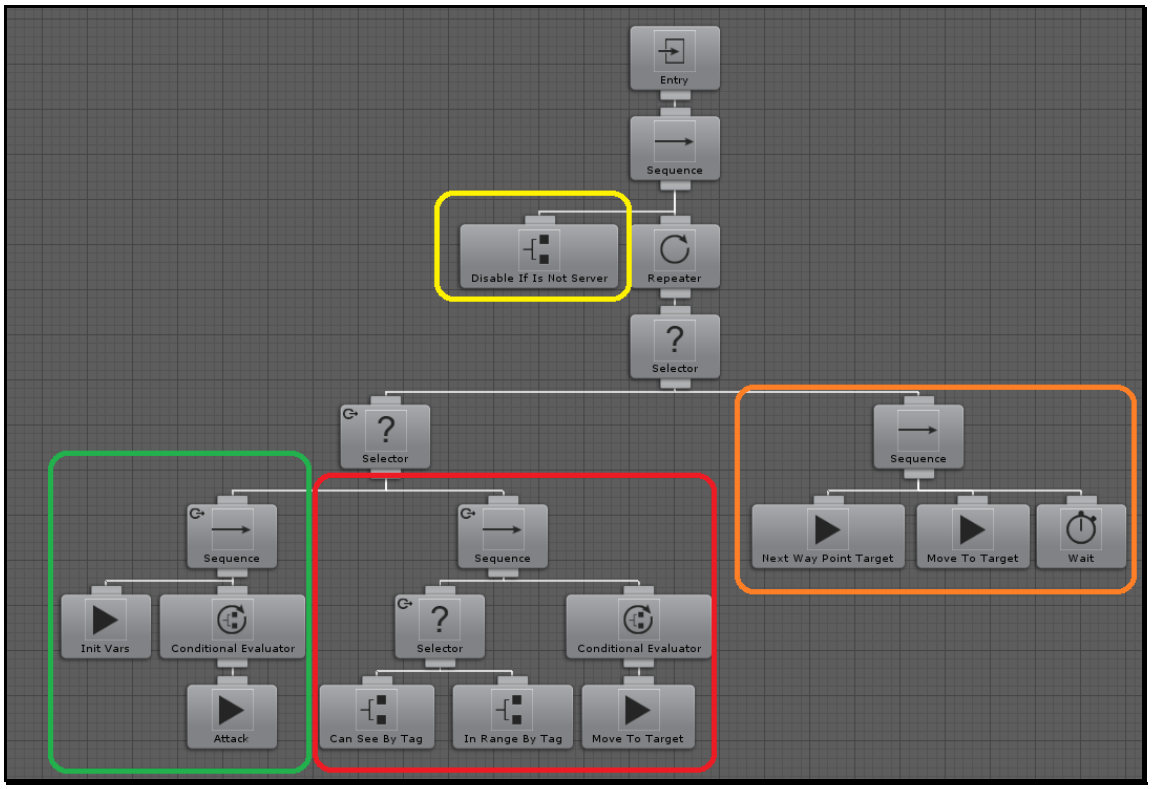
También tenemos unos que se llaman "*Conditional Evaluator*", estos nodos hacer la misma funcionalidad que el nodo "*Sequence*" pero sirven para tener los nodos más ordenados, por ejemplo: Podemos tener un nodo "*Sequence*" como padre y un condicional como hijo, en este caso el condicional "*Can See By Tag*", y a la derecha el nodo de atacar o bien poner un evaluador con este mismo condicional y como hijo el nodo de ataque.





En ambos casos el funcionamiento es el mismo, pero como podemos ver en las imágenes que tenemos arriba la primera opción ocupa más espacio y si queremos tenerlo más ordenado la segunda opción nos ayudará más.

Con la configuración que tenemos actualmente de la IA, el funcionamiento de la misma seria el siguiente:



(Nodo amarillo)
Lo primero a comprobar seria si la IA se está ejecutando en el servidor o no. En caso de que estemos en el servidor seguiremos ejecutando la IA, y sino devolveremos un fallo y acabaremos aquí. Este nodo al estar fuera del *repeater* solo comprobara una vez, el resto de nodos estarán en constante funcionamiento.

(Nodo verde)
En este nodo inicializamos algunas variables y comprobamos si tenemos delante al *player* para poder atacarle, el caso que no este lo suficientemente cerca pasaremos al siguiente nodo, pero si algún momento de la ejecución de los siguientes nodos se cumple esta condición.

(Nodo rojo)
Este nodo se encarga de buscar al *player* e ir hacia el si lo tenemos en nuestro rango de visión o si está cerca nuestro. Este nodo puede ser interrumpido por el nodo verde. En caso de no tener al *player* cerca o no tenerlo en nuestra línea de visión pasaremos al siguiente nodo.

(Nodo naranja)
En este grupo de nodos lo que se hace es buscar un *waypoint* que se lo dará un script externo y nos moveremos hacia él, una vez llegamos al *waypoint* nos pararemos durante unos segundos y el *repeater* volverá a ejecutar los nodos desde el principio. Este nodo puede ser interrumpido por el nodo verde o el rojo.

Sistema de NPC's vivos

Igual que hay NPC's que dan misiones o hablan, hay un tercer tipo que no se comunicará con el jugador, pero dará la sensación de que las cosas están vivas, ya que irá moviéndose. Estos están en la ciudad de *Mocholo*, la idea principal era dotar de vida a la ciudad así que, utilizando el Behaviour, se les asignan *Waypoints* y simplemente se van moviendo dando vueltas.

Post-procesing

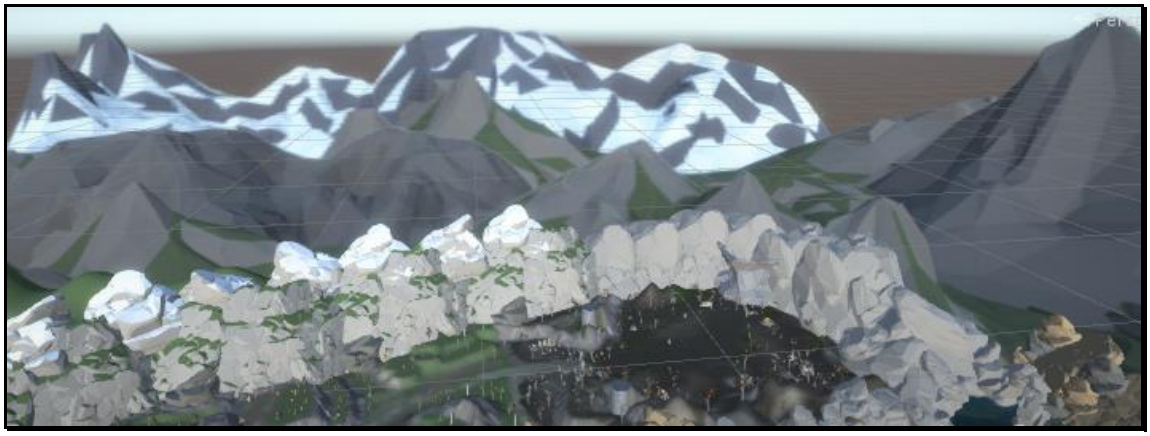
El post-processing se utiliza para dotar al juego de efectos a nivel gráfico dándole un aspecto más bonito y agradable a la vista, aplicándole efectos como Bloom, Anti-Aliasing, reflejos, Bluur etc.

Otro ejemplo, además del mencionado anteriormente en la descripción de Assets y más detallado.

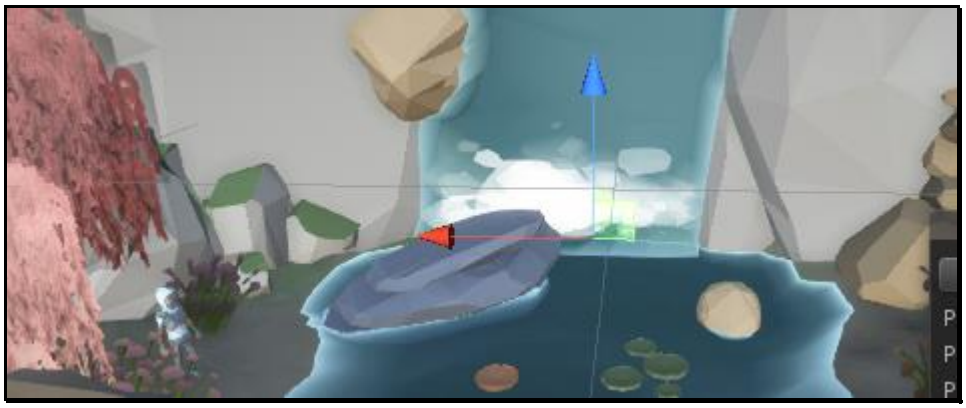
La imagen de la derecha se ve con menos bordes, potenciando los colores vivos y dando potencia a las sombras, entre otros. Varios efectos son perceptibles al mover al jugador.

Otras implementaciones

El videojuego cuenta con imágenes utilizadas a modo de simulación de montañas para dotar al nivel de realismo, igual que un rio con movimiento, cascadas y efectos FX varios. Se exponen varios ejemplos a continuación.



(Sistema de montañas como fondo)



(Sistema de cascada)

(Pequeña hoguera)

También se dispone de la funcionalidad de visualización del mapa. Esto se ha realizado con coordenadas en el juego y cámaras posicionadas desde arriba apuntando hacia abajo. Es decir, cuando el jugador está en una zona y pulsa la tecla de mapa (La M en el teclado) se abre una vista de la zona donde se encuentra. En total el mapa está dividido en 5 trozos, y para cada trozo hay una cámara que, si el jugador está entre los valores de la misma, se activa esta.

(Vista del mapa desde Camera 1)

Al entrar en ciertos lugares (Base enemiga) aparece niebla de forma progresiva y hay cambios de luz en el escenario. Además, no se permite la visualización del mapa en estos lugares.

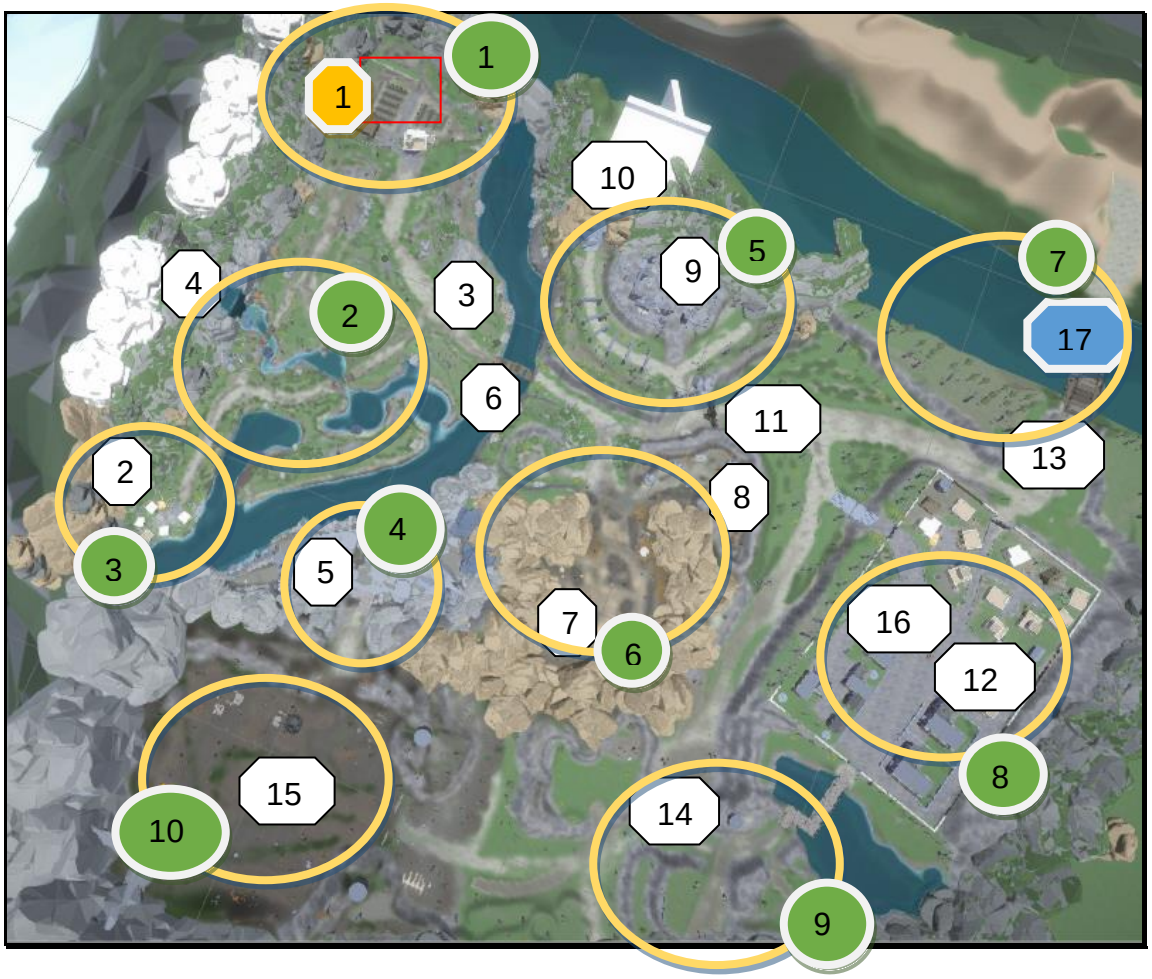
Hay un sistema de sonido para acciones como caminar, efectos de agua, o para las varias habilidades

6. Diseño del nivel

El diseño empleado para la creación del videojuego se pensó para que el mismo tuviese una duración entre 30 minutos y 1 hora de juego aproximadamente.
Esos valores de tiempo serían dados para un jugador que es la primera vez que juega y no es habitual en estos juegos, si bien un jugador más experimentado podría terminarlo antes no dejan de ser valores aproximados.

El nivel está diseñado de una forma lineal en lo que a historia se refiere, es decir, debes ir completando todas las misiones que el juego va ofreciendo hasta llegar al final, pero tienes la libertad de moverte por todo el mapa según vas avanzando y obteniendo llaves, eliminando obstáculos en el camino etc.

En la ilustración inferior adjunto mapeado completo del *Stage_I* y paso a detallar la ruta a seguir por el usuario y el comentario de detalles y otros para darle variedad al mismo.



A modo de leyenda:
El rectangulo rojo es la zona de inicio donde se empieza la aventura.

Las zonas con círculo amarillo son los lugares, y los círculos verdes con número representan los números expuestos aquí abajo.
1 Granja Nexiant
2 Las aguas
3 Aldea del agua
4 Base enemigo 1
5 Castillo antiguo
6 Bosque perdido
7 Puente de Shoku
8 Mocholo
9 Zona entrenamiento Mocholo
10 Tierras Devastadas, Base enemigo 2

Los números con fondo blanco son los puntos en guía por donde se va pasando que se explicarán a continuación por orden. El 1 marcado de amarillo es el inicio, y el azul el final.
Hay más npcs con los que poder hablar pero no saldrán en la siguiente explicación del Stage I.

1- Antes de comenzar la aventura, hay un texto breve que explica la misma en formato de *StoryTelling*, la primera misión es salir de la granja ya sea a través del portal que conecta las puertas o simplemente, saltando la valla. Justo al salir hay un NPC que nos dirá si le hablamos que en el Pueblo de Agua nos pueden ayudar.

Granja Nexiant

2- Una vez en el pueblo, tendremos acceso a nuestra primera misión que trata de recoger ciertas hierbas para curar a la esposa de un cocinero. Simplemente se deben buscar las plantas esparcidas por la zona del circulo verde (nº 2). Estas brillan para que sean fácilmente identificables. Una vez completada la misión el cocinero nos dirá que el chico de los víveres debería haber llegado ya y que vayamos a buscarle.

Aldea del agua

3- Si hubiésemos llegado aquí antes de hacer la primera misión, simplemente un carro nos bloquearía el paso, ya que el chico de la segunda misión, que está en el punto 3 del mapa, se supone que va tras los ladrones que le roban y por ello, no está.
Al terminar la primera misión, el chico del carro nos ofrecerá la segunda misión, que es obtener la rueda del carro robada, ya que su carro de víveres nos bloqueará el camino al no poder moverlo. También nos dará una especie de piedra mágica que se le cayó a uno de esos ladrones. Esta piedra mágica es la que tiene la capacidad de activar el portal de la cascada que nos llevará a la Base enemigo 1.
El chico nos advierte que debemos ir armados, ya que los bandidos nos intentarán matar si nos detectan en su base.

Camino bloqueado

4- En el punto 4 es donde se encuentra el portal, basta con cruzarlo y aparecer en el punto 5.
5- Aquí es donde tenemos los primeros encuentros con bandidos, se trata de encontrar la rueda del carro robada por la base, que la misma se encuentra en el círculo amarillo (nº4(color verde)). Estos bandidos no hace falta matarlos, pero no nos dejarán tranquilos si nos detectan. Una vez recuperada la rueda se vuelve a hablar con el chico atravesando los portales a la inversa y ya podremos continuar.

Asentamiento bandido 1

6- Una vez la misión se completa, la Zona 1 del juego está acabada y pasamos a la zona 2.
7- Aquí nos ofrecerán la 3º misión del juego, que es llevar una cantimplora al hermano de un chico que se encuentra en el siguiente punto. Por la zona, hay bandidos rondando que intentarán acabar con nosotros.

Bosque Perdido

8- Una vez se encuentra al chico, nos dirá que robó una llave a unos bandidos que abre la puerta de acceso al punto 9 y que él no ha sido capaz de encontrar la forma de quemar los troncos que bloquean el paso (el 11). Así que nos ofrece la 4º misión que es entrar en la mini base enemiga (circulo amarillo punto nº5) para ver si damos con la solución.
9- Aquí se trata de buscar una llave que abre la puerta de la zona 10. La misma está encima de una mesa.

Casa derruida

10- Una vez abierta la puerta, un caballero de *Mocholo* nos da las gracias por liberarle y nos ofrece su ayuda a cambio. Su manera de agradecernos es quemando esos troncos que bloquean el paso.
11- Aparecerá el mismo caballero del punto 10 y habrá unos pequeños efectos de fuego que quemarán los troncos y plantas que bloquean el paso.
12- Una vez en *Mocholo*, un guardia nos dirá que si queremos ir a Shoku deberemos ir al puente.

Mocholo

13- En el puente, un guarda nos dirá que los bandidos talaron las maderas y tardarán en reconstruirlo, que deberíamos ir a hablar con el caballero *Sparrow* , en la zona de entrenamiento, para dar fin a los bandidos de las Tierras Devastadas

Puente hacia la Zona Entrenamiento desde Mocholo

14- En este punto hablamos con *Sparrow* y nos dará la última misión, que será ir al punto 15 y acabar con los 4 jefes de los bandidos (Se diferencian de los normales al tener la ropa de color rojo).

Sparrow y soldados

15- Se acaba con los bandidos y se vuelve a hablar con *Sparrow* que nos dirá que hablemos con el Alcalde de *Mocholo* en el punto 16.

Tierras Devastadas

16- El Alcalde nos dará las gracias, y dirá que ya está el puente arreglado. (Básicamente aparece el puente ya construido de nuevo) con una bola color lila, que cuando se recoja terminará el Stage I.

17-Fin del juego

El mapa tiene colisionadores para que el jugador no pueda atravesar ciertas áreas, o salirse del mapa.

7. Conclusiones

Al ser dos personas las que realizan este TFM, se coloca el nombre de cada uno seguido de las conclusiones.

Julen Gallego:

Gracias a la realización del trabajo durante estas semanas, se ha seguido ampliando el conocimiento adquirido a lo largo del curso.
Desde un inicio se propuso que el videojuego sería completamente On-Line, pero el ir pasando de los días y ver, que tanto Unity dejaría de dar soporte a las librerías utilizadas y la complejidad de las mismas, se fue apartando la idea, centrándose en una historia para un jugador, con el online como complemento y no como parte del todo, por lo que pese a tener ciertos aspectos implementados, se dejó de lado, ya que se prefirió invertir más tiempo en otras implementaciones.

Se podría decir que no se han podido cumplir todos los objetivos. Si bien hacer dos mapas más para terminar en si la historia, habiendo llevado la creación del Stage I casi dos meses (y a mi juicio, está sin terminar) se vio inviable, así que únicamente se dejó el primero para hacer.

Al final se aprende que la creación de un nivel, es tarea no difícil, pero si larga y que requiere de mucho tiempo para tener un nivel detallado.

La planificación que se ha ido siguiendo a lo largo de la realización, se intentó ceñir bastante al plan original, dando lugar siempre a pequeñas improvisaciones o implementaciones que no se habían propuesto en un inicio o cosas que por una cuestión de puro tiempo, se dejaron aparte.

Yo en lo personal, creo que sí que ha sido una buena planificación y metodología, pues si más no se iba siguiendo un ritmo constante, tanto para el videojuego, como para la realización de la memoria. También pienso que los cambios que se han introducido y que estaban dudosos al inicio o descartados, se tuvieron que hacer efectivos para la realización del videojuego (Por ejemplo, el ya mencionado tema on-line).

Se quedan muchas cosas en el tintero a implementar, más magias, misiones, los dos mapas restantes, dar más detalle a ciudades, habilidades propias para poder elegir, sistema de nivel para jugador, magias enemigas y un largo etc.

Al final se hace lo que se puede con el tiempo que se tiene, y yo sinceramente estoy bastante satisfecho con el trabajo desarrollado.

Julen Gallego Sevil

Jordi Puertas:

¿Qué lecciones se han aprendido del trabajo?
Se ha aprendido a usar un *behavior tree* para configurar la IA de los enemigos haciendo scripts lo más genéricos posible, para que de esta manera los nodos se pudieran usar no solo en la IA enemiga sino que también son usados en algunos NPC.

¿Hemos logrado todos los objetivos? ¿Porque?
No, no se han logrado todos los objetivos. Al asignar objetivos no se tuvieron en cuentas varias casuísticas que nos han ido ocurriendo a lo largo del trabajo y que nos ha hecho ir más lentos en algunos aspectos.

¿Se ha seguido la planificación? ¿La metodología prevista ha sido la adecuada?
Se ha intentado seguir la planificación lo máximo posible, pero según se iba avanzando en el trabajo nos dábamos cuenta de que había objetivos planificados que se tardaban menos en hacer y otros que se tardaban más descuadrando el mapa planificado.

¿Ha habido que incluir cambios para garantizar el éxito del trabajo? ¿Porque?
Sí, siempre que se podía y era necesario se modificaba la planificación, los objetivos o se buscaban alternativas para poder avanzar más rápido. En ningún momento hemos querido ceñirnos a la planificación inicial ya que íbamos aprendiendo de nuestros errores según íbamos haciendo.

Las líneas de trabajo futuro que no se han podido explorar en este trabajo y han quedado pendientes.
El juego *multiplayer* se ha quedado a medias a causa de todos los problemas que da y la poca información que hay en la API de Unity.

8. Glosario

Player: Persona física que utiliza el jugador en el mundo virtual. Es decir, se hace referencia al Personaje controlable.

NPC: Persona virtual que no maneja un jugador.

Colisionador: o Colliders en Unity: definen la forma de un objeto para los propósitos de colisiones físicas.

Stage I: Primera zona de juego implementada en el mismo.

Waypoints : Zonas predeterminadas donde se puede mover un GO.

GO: GameObject en Unity.

IA: Inteligencia Artificial

Teleport: Tele-transportador de unidades GO.

Quest: Misiones.

Trigger: Disparadores (en el código) o eventos

Array: vectores (programación).

StoryTelling: Contar una historia.

9. Bibliografía

Referencias

01-05-2019

https://www.3ciencias.com/wp-content/uploads/2012/09/graficos-max-potencia.pdf 01-05-2019

https://es.wikipedia.org/wiki/CryEngine 02-05-2019

https://www.cryengine.com/ 02-05-2019

https://www.fayerwayer.com/2013/08/detalles-de-lo-que-nos-trae-la-tecnologia-cryengine/ 02-05-2019/

http://www.baboonlab.com/blog/noticias-de-marketing-inmobiliario-y-tecnologia-1/post/unreal-engine-4-el-motor-grafico-que-ofrece-realismo-al-maximo-23 02-05-2019

Assets Unity Store

Nature Pack
https://assetstore.unity.com/packages/3d/vegetation/trees/polygon-nature-pack-120152

Adventure Pack
https://assetstore.unity.com/packages/3d/environments/fantasy/polygon-adventure-pack-80585

Dungeon Pack
https://assetstore.unity.com/packages/3d/environments/dungeons/polygon-dungeons-pack-102677

Post-Processing
https://assetstore.unity.com/packages/essentials/post-processing-stack-83912

Behaviour-Tree
https://assetstore.unity.com/packages/tools/visual-scripting/behavior-designer-behavior-trees-for-everyone-15277

10. Anexos

Manual básico de controles

W: Tecla movimiento hacia delante.
S: Tecla movimiento hacia atrás.
A: Tecla movimiento hacia izquierda
D: Tecla movimiento hacia derecha

Barra Espaciadora: Saltar
C: Agacharse

1: Habilidad especial doble espadazo
2: Habilidad espacial espada tornado
3: Habilidad espacial triple descarga de fuego
4: Habilidad especial gran bola de fuego

Click izquierdo: Ataque básico con la espada
Click derecho: Ataques especiales

F: Interacción con los NPCS
E: Pasar textos en el juego