Memòria de proyecto de Unreal

Roger Llovera, Dani Quesada

Para poder jugar al juego creado, tenemos que ir a la carpeta maps y seleccionar JuegoUnreal.

Un menú principal básico y posibilidad de guardar y recuperar la partida

Primero hemos creado widget de user interface que nos permita poner botones y hemos colocado los tres botones dentro de un vertical box que nos permite distribuir los botones y poner margenes entre ellos. A continuación pasamos a darle funcionalidad enlazando su click con una funcion. El botón de new game lo que hace es colocar y crear (En caso de que sea necesario como por ejemplo que has matado al enemigo) todos los objetos de la escena que pueden tener un estado modificado al inicial. El boton de cargar hacer la misma funcionalidad que el botón de new game pero los valores a cargar vienen dados por el slot de guardado que tenemos creado (En el caso de que hayamos guardado alguna partida anteriormente). El botón de quit te permite salir del juego.

Para hacer el menú más bonito hemos puesto una camara que enfoca todo el nivel y al entrar al menú principal hacemos que esta cámara se la que tiene mas prioridad y la que enfoque el juego. Creamos el widget de menú principal y lo añadimos al viewport.

Dentro del juego también tienes un menú de pausa, dicho menú se abre con la tecla R, este menú congelará el tiempo de partida y hará aparecer tres botones: Volver al juego (Lo que hace este botón es descongelar el tiempo y borrar el menú de pausa del viewport), guardar partida (Este botón guarda la class SaveGame donde tenemos los datos de posición de los personajes, si hemos recogido la llave y si hemos matado al enemigo), y volver al menú principal (Este botón colocará todo como al principio a la espera de que pidas una nueva acción, ya sea salir del juego, empezar de nuevo o cargar el nivel guardado).

Blueprint de animación con su máquina de estados

Para crear el blueprint de animaciones del personaje principal con su máquina de estados hemos creado un blueprint de tipo animación, en este blueprint tenemos puesto distintas variables que nos permiten pasar de un estado a otro cambiando así la animación que se ejecuta en ese momento, alguna de las variables son: velocidad (float) para poder saber si el personaje se está moviendo o no, isInAir (bool) para saber si el personaje está en saltando, o cayendo, o bien está tocando el suelo, etc.

La misma máquina de estados sirve para la IA pero para el ataque, como intentamos acceder a una variable del blueprint específico del player, probamos de hacer un cast al blueprint del player y si no es posible tal cast, intentamos hacer un cast al blueprint de la IA.

Después montamos la lógica de animación, ponemos el blendspace del idle_walk_run como primera animación a ejecutar. A continuación ponemos transiciones a la maquina de estados de saltar que transiciona desde empezar el salto, se mantiene en la posición de salto hasta que detecta que ha tocado suelo y hace la animación de caer. También tiene una transición a la animación de nadar en caso de que el personaje esté nadando y otra a la animación de atacar en caso que la variable de atacar del propietario del set de animaciones esté atacando.

Blendspace

Para el blendspace hemos creado un blueprint de tipo blendspace que nos permite añadir distintas animaciones y se interpolen de forma 2D (Solo modificadas por un valor variable) o bien se interpolan de forma 3D (por dos variables). Para el blendspace de idle_walk_run, hemos creado un blendspace 2D, dentro hemos puesto la animación de idle a la izquierda como valor 0, a la derecha hemos puesto la animación de correr con un valor máximo (valor de 420). Este valor numérico que irá modificándose a lo largo del tiempo será igual a la velocidad que tenga el personaje en cada momento. Para poder hacer que el blendspace funcione tenemos que añadirlo al blueprint de animaciones y en el valor interno del blandspace que permite modificar el valor de idle_walk_run poner la velocidad del carácter que está usando la animación.

IA con Behaviour Tree en al menos un tipo de NPC

Hemos creado un AI_Controller que nos permite decir que behaviour tree ha de ejecutar la IA y que blackboard ha de usar. Una vez ejecutado el código que nos informa que behaviour tree tenemos que usar, vamos al blueprint del AI_Character y le indicamos que AI_Controller debe usar. En el behaviour tree tenemos un selector que tiene un servicio que está todo el rato comprobando si tenemos dentro del rango de visión al player (Este rango de visión es un pawnSensor que tienes el AI_Character y se encarga de modificar la variable booleana de la blackboard que nos indica si estamos viendo el player). En caso de que veamos el player iremos hacia él y una vez realizado con éxito el acercarnos al player, el enemigo atacará. Si pierde de vista al player hará una patrulla aleatoria cada 2 segundos alrededor del punto inicial.

Objetivo, Condición de victoria y Condición de fracaso

El objetivo principal del nivel es matar al Dark Link que está patrullando en la subida de las escaleras y conseguir la llave que suelta en el punto central de patrulla. Una vez hayas conseguido matarlo (condición de victoria) la llave aparecerá, la recogerás y podrías ir a abrir el muro que está subiendo las escaleras. Dentro encontrarás una mazmorra con un cofre y se habrá acabado el juego, podrás seguir jugando para explorar un poco el mapa, pero no habrá nada más.

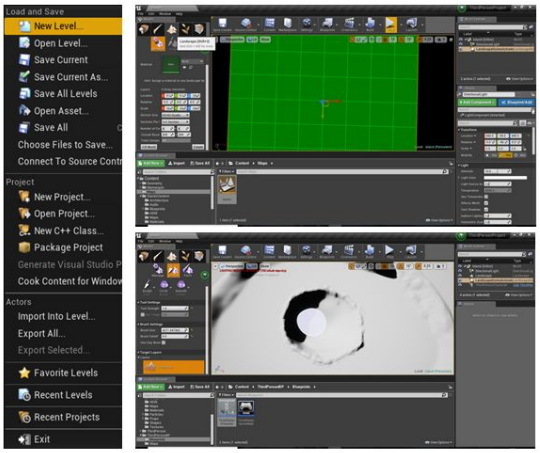
En caso de que mueras luchando contra el dark link (condición de derrota), volveras al menu de inicio donde podrás empezar de nuevo o cargar la partida que tengas guardada.

UMG con al menos dos variables a representar en tiempo real

En el UMG del juego tenemos dos barras que representan la barra de vida del player y la barra de stamina del player, esta segunda barra se gasta cada vez que el personaje ataca y se regenera de forma continua mientras no estas atacando.

Para crear estas dos barras del hud hemos creado un nuevo widget de tipo user interface y hemos puesto dos barras de progreso que su valor de porcentaje se modifica a tiempo real accediendo a las variables de vida y stamina del player que se van modificando cada vez que te atacan o tu atacas.
Este nuevo widget se añade cada vez que entres a jugar en el juego y no tengas ese hud activado.

LandScape

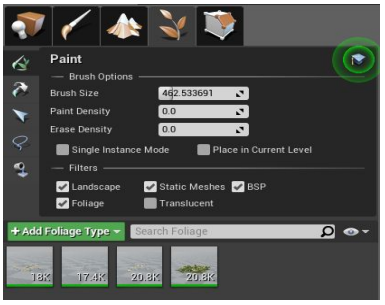


Para crear el nivel hemos usado la herramienta de Landscape y lo hemos retocado con las herramientas de pinceladas que nos aportaba Unreal. El nivel debía ser un mundo abierto y con mezcla de mundo cerrado, así que decidimos hacer como una especie de cráter rodeado de agua y que hayan unas ruinas en el centro donde tendrá que ir el player. Así que creamos un nivel, añadimos las dimensiones, tampoco muy grande porque la mayoría sería océano a la lo lejos y algunas montañas. finalmente una vez tenemos ya el plano empezamos a esculpir.

La mayoría de Assets usados en la práctica han sido descargados de la Store de Unreal. Solo para ponerlo más bonito.

Herramienta de Vegetación

Por ejemplo, en la herramienta de pincelado de vegetación le añadimos algunas flores y modelo 3d de árboles y vegetación que venían del pack de la store. Tampoco tenía mucho misterio, añadimos los modelos que queremos que se vayan a colocar en el mapa con el pincel en el botón de Add Foliage Type, y una vez elegidas vamos con el pincel y las vamos poniendo.

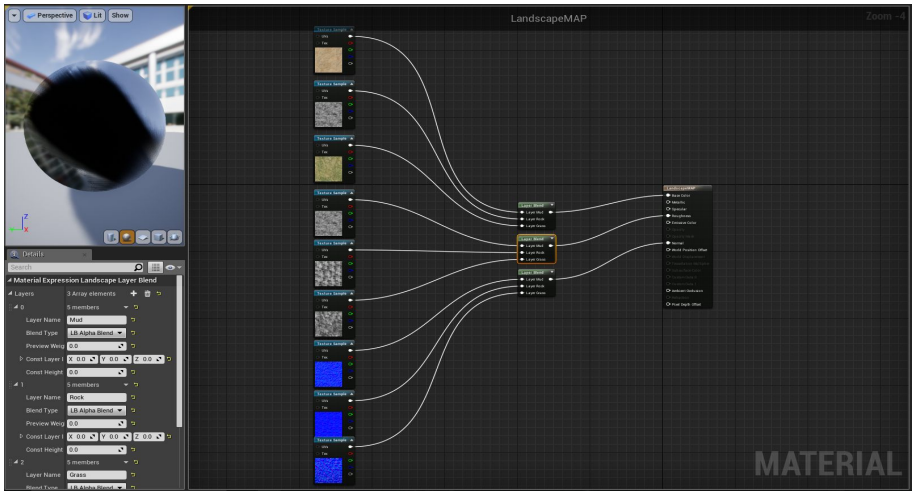


Después pintamos el landscape on algunos materiales que proporcionaba el pack de assets, pero nosotros construimos el multimaterial, añadiendo como lo hicimos en clase.

LandScape Material

Ponemos los 3 tipos de materiales dentro de un material: Hierba, roca y barro y sus respectivos albedos, gloss y normales. Pero ahora están divididos individualmente con sus mismos tipos de mapas dentro de un Layer blend para cada tipo de mapas, es decir, todos los mapas de albedos juntos, todos los mapas de normales en otro.

Finalmente añadimos cada parte donde toca. Después, nos tenemos que ir a la herramienta de pintar en landscape y crear una instancia/referencia de esos materiales y ahora ya pintará el material que hayas elegido y su albedo y sus normales.



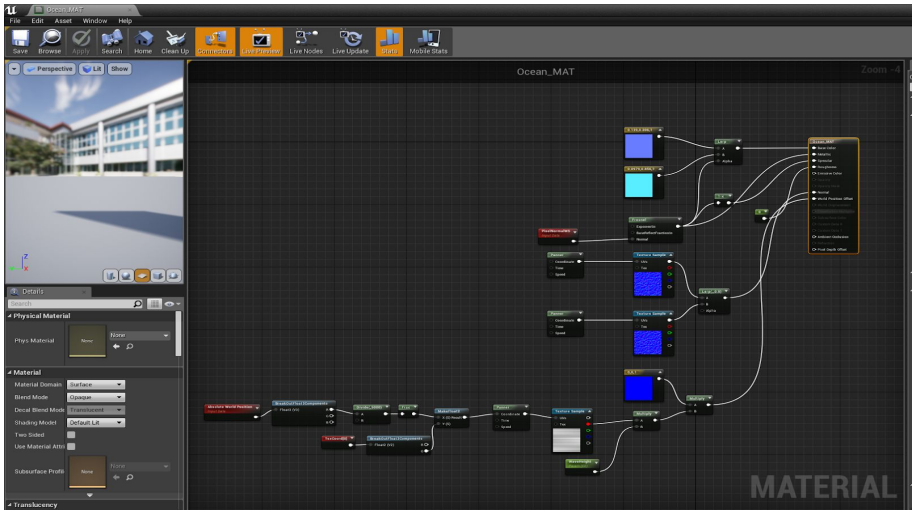
Material del Oceano

Para el agua podría haber puesto la que hicimos en clase, pero como la mayoría de assets estaban muy acordes unos a otros con la misma estética, chocaba bastante, como sabiamos ya previamente como hacer un oceano, nos limitamos a usar el que nos venía por defecto en la asset store de este pack. Para eso usamos:

Efecto Fresnel: aquel que se produce cuando tú estás mirando en la orilla de un lago, que parece que brilla más o menos. Cuando miras el agua que está más lejos, la ves más brillante (que parece que tenga estrellitas). Eso es el contraste de las micro-olas que nos están mirando a nosotros o las olas perpendiculares. Cuando lo tengamos perpendicular, lo veremos brillar más que de frente. Queremos resaltar cada parte del plano brillará de forma distinta. Estaré simulando la forma de las olas. Queremos cambiar el color de la superficie según el ángulo perpendicular a la superficie.

LinearInterpolate: me interpola dos valores teniendo en cuenta el valor Alpha. Si Alpha es 0, me dará A. Si es 1, me dará B. Es un LERP. Esto admite cualquier tipo de valor

Pixel Normal WorldSpace (WS): Esto me da la dirección normal de cada pixel en world Space. Pero vamos a independizar la proyección (resultado del shader) sobre el plano y haciendo que dependa de sus coordenadas de mundo. Haremos que la ola y el material sólo dependan de la posición absoluta del mundo suyas.



y luego hacíamos el frac, que consiste en según la amplitud y la anchura del plano, hacer que encajaran bien entre ellos para luego poder poner la cantidad que queramos de dimensiones de oceano (X · Y) y así crearnos nuestro océano infinito sin tener que ir uno a uno.

Pawn Primera/Tercera persona

Para el pawn en primera persona y el de tercera persona primero debemos crear 2 camar, una muy cerca de la cara del player para que simulara la vista en primera persona y otra con un brazo añadido que siga un radio y rodee al player como si se tratase de una cámara en 3ª persona.

Estas cámaras las activamos y la desactivamos según nos pareciera conveniente con un botón a elegir. Con el nodo Flip-Flop vamos siempre alternando entre un evento u otro. Basicamente desconectabamos la otra camara para activar la nueva y cortabamos las rotaciones según creiamos conveniente.