Dan Roeniger
Gerard Maynegre
Sergi Morral

Memoria escrita: Proyecto Unreal

Controles:

Mover: awsd
Saltar/volar: espacio
Parar/reanudar tiempo: left alt
correr más (con power up) left shift
Disparar balas de fuego: R

Dan Roeniger

-Programación condiciones de victoria y derrota a través de una game instance y level blueprint.
-Todos los blueprints, behavour tree i blackboard de los enemigos
-Programación de todas las mecánicas y power ups (bajar velocidad tiempo con barra de energia, power up de boost de velocidad extremo, jetpack)
-HUD básico del juego (barras de vida del player, enemigo y barra de paro del tiempo)
-Inclusión de animación de disparo al del thirdpersoncharacter que a través de un event hace que el jugador no se pueda mover cuando dispara.
-Bala del player hecha con particulas de fuego que explota al chocar y mata/quita vida a los enemigos.
-Bala del cañón que también hace daño a los enemigos humanos si les da y botiquines que curan al player
-Un postproceso que hace un efecto de velocidad alrededor de la pantalla según la velocidad que lleva el player.
-Blueprint mejorado de las plataformas que ya hice para la práctica 1
-Blueprint mejorado del cañon hecho en clase para que solo te dispare si te está mirando y si estas suficientemente cerca.
-Cooperación en el balanceo
-Ayuda general a los miembros del grupo
-Guardar y cargar los enemigos
-Efecto muerte al quedarse sin vida-

-Daño por proporcional por caída (al principio era por impacto, por lo que si corrías mucho y chocabas con una pared también morías pero a causa de que con los baches también perdías un poco de vida lo limite a solo daño por movimiento vertical.

-Finalmente el útimo dia tuve que encargarme de arreglar el guardado (tanto como pude), cargado y los menús porque estaban fatal cuando supuestamente iban bien según se me había comunicado.
En general he tenido muchos problemas a causa de crashs de unreal con alguna que otra pérdida de información y a la hora de realizar los migrates porque de golpe cosas que funcionaban dejaban de funcionar, o no se enviaban bien, por ejemplo el postproceso de velocidad dejaba de funcionar bien y necesitaba de golpe una velocidad enorme para que se vea, o cuando Gerard cambio las meshs, el forward de los objetos cambió y empezaron a rotar mal y los tuve que arreglar manualmente.
Hemos planteado la practica de manera iterativa, mientras yo hacia todo el sistema de juego, Gerard se encargaba de hacer el mapa y buscar los modelos, entonces lo combinamos (con muchos problemas de unreal) y se lo pasamos a Sergi para que aplicara todo lo que él había hecho por su cuenta (mientras yo mejoraba algunos blueprints) entonces me lo volvió a enviar apliqué mis mejoras de los blueprints y mejoramos el nivel entre todos en términos de diseño entre todos.

Gerard Maynegre

En este proyecto mis partes han sido las siguientes:

-Un solo nivel

-Mundo abierto o mezcla de mundo abierto y cerrado (uso de Landscape obligatorio con material de al menos dos capas).

-Uso de la herramienta de vegetación.

-Al menos una instancia dinámica de material con un parámetro que se modifica en tiempo real.

-Balancing del juego, junto con las instrucciones.

-Búsqueda y aplicación de modelos 3D.

Antes de empezar, estuvimos hablando sobre el diseño del juego y cual seria su objetivo, y una vez decidimos entre todos el objetivo, me puse a diseñar como sería el nivel. El objetivo del juego, el cual es recoger las 3 banderas repartidas por el mapa. Para ello, he creado un level en donde hay 3 zonas bien diferenciadas: una montaña muy alta, un lago con una isla en el medio y finalmente un laberinto, y al final de cada una (en la salida del laberinto, en la isla central del lago y en lo alto de la montaña) es donde se encuentran dichas banderas, además de un power up en 2 de ellas. Los bordes del mapa están rodeados de montañas para así evitar sensación de final del mapa, además el landscape tiene un material con 5 capas:

hierba, roca, tierra, arena y nieve, para así darle más color al nivel, sobretodo a las montañas.

Una vez hecho esto, me puse con la herramienta de vegetación. Me costó mucho encontrar un modelo de árbol funcional, y cuando encontré uno, nos causaba una bajada muy considerable de los fps a todos los miembros del grupo, y por eso decidimos eliminar la mayoría de ellos y dejar unos pocos. Decidí poner hierba, y como los modelos que encontré tampoco eran gran cosa, decidí usar el que usamos en clase ya que así me aseguré de que no nos diera problemas con los fps.

Seguidamente, me puse con la agua del lago, donde cogí el material que hicimos en clase ya que funcionaba de maravilla, y lo apliqué con el blueprint que te permite establecer la superficie deseada.

Una vez tuve todo esto, empezé a poner las plataformas que te permiten llegar a la isla central del lago y que el player tiene que superar si quiere la bandera del lago. Llegados a este punto, empezamos a fusionar los proyectos con los otros miembros del grupo, y con eso empezé a distribuir los enemigos y botiquines por el mapa, además de aplicar todos los modelos 3D para los enemigos, power ups, etc para que no sean simples cubos grises.

Sergi Morral

Las partes cubiertas dentro del proyecto han sido estas:

- Al menos un blueprint de animación con su máquina de estados (retoques del original)
- Al menos un blendspace (Retoques del original)
- Al menos un UMG con al menos dos variables a representar en tiempo real
- Un menú principal básico y posibilidad de guardar y recuperar la partida (total o parcialmente)
- Guardar y Cargar el juego.
- Sonidos al juego

Una de las partes que más trabajo ha supuesto es la implementación de sonido en ciertos objetos, personajes y escenarios, ya que durante el curso no nos han enseñado la implementación de sonidos, así que he tenido que buscar por mi mismo cómo hacer uso de sonidos, especialmente la reproducción de pisadas de manera aleatoria al correr.
Mi idea era hacer un sistema dinámico de sonido de pasos dependiendo de qué terreno se pise. En el caso del juego he puesto la condición de si se pisa el suelo o se pisa por agua.
DIcho eso, para poder poner estos sonidos he tenido que hacer un fichero de sonido

donde he mezclado de manera aleatoria los sonidos de pisadas en el suelo en un pack y los sonidos de pisadas de agua en otro pack. Luego, usando las variables de Suelo y Agua añadidos en el Physical Surface (dentro de los ajustes de proyecto, en motores), he creado dos physical materials i los he asignado a los MAT que están aplicados dentro del suelo y del agua (landscapeMAT y waterMAT respectivamente)

He tenido que crear eventos dentro de la animación de correr, para que se active un custom event que reproduzca los sonidos dentro del archivo de sonido con los sonidos ya mezclados aleatorios.

Para los UMG, he creado un menú principal, un menú de pausa, un menú de guardar y cargar, un menú de opciones y un menú game over.

El menú principal esta dentro de otro nivel en negro, donde solo hay el UMG del del menú principal. En este menú hay 3 botones: Empezar el juego, las opciones y salir del juego, cada uno haciendo las acciones correspondientes: Empezar el juego carga el nivel del juego, el de opciones carga el menú de opciones dentro del nivel del menú principal y el botón de salir termina el juego.

El menú de pausa tiene 3 botones: un botón que reanuda el juego, un botón que abre el menú de salvar y guardar y el menú de terminar el juego. El de reanudar simplemente sale del menú y continua el juego, el de terminar el juego termina el juego y el de salvar y guardar abre otro menú que es el de salvar y guardar el juego.

El menú de salvar y guardar el juego tiene 3 botones: el botón de salvar el juego, el de guardar y el de ir al menú anterior (el de pausa).

Para hacer el menú de game over he tenido que modificar parte del blueprint hecho de Dan, solo añadiendo un retraso después de la muerte del player i añadiendo también que aparezca el game over. En este menú hay 2 botones: Restart y Exit, donde el restart empieza el nvel de nuevo (y si esto se hace un load y se vuelve a la última posición de guardado) y el Exit donde se sale del juego.

Para hacer los save/load games, he tenido que usar el SInventory que hemos usado en clase donde hemos guardado todas las variables necesarias: En este caso, al pulsar el botón guardar guardamos en una clase nueva las variables de: vida, vida total, player pos, stoptime, número de banderas, un bool de jetpack y un bool de boostvel (los bools son como powerups).
La cosa funciona de la siguiente manera: Obtengo el SInventory y le hago un break, de todas las variables dentro del SInventory,las guardo como variables dentro de otra clase donde tiene todas las variables del SInventory. Estas variables de la classe les hago un set y los guardo dentro de una ranura de juego.

Para hacer el load parecido: Obtenemos la clase donde hemos guardado las variables y hacemos un get de todas las variables que tiene, i le hacemos un set en el SInventory.
La pos player y el stoptime funcionan de diferente manera ya que vienen del player en vez de el SInventory, asi que en vez de llamar el SInventory, tambien llamo al player, cojo estas 2 variables y las pongo dentro de la clase.