DOCUMENTO JUSTIFICATIVO

Óscar Gómez Monedero

# Justificación del documento

Tal y como se informa en las instrucciones de la práctica, el alumno puede entregar documentación para la explicación del proyecto.

Este documento servirá para explicar detalles de la parte y del *bloque 2: mecanim:generic* y de la *parte 2 del bloque 2: mecanim:humanoide*.

# Justificación de la existencia de dos proyectos

Se ha decidido crear dos proyectos para poderlos diferenciar y estructurar mejor, al no tener elementos en común más allá de pertenecer a la misma práctica. La parte uno corresponde con la primera parte del bloque 2, y esta relacionado con el modelo y las animaciones del bloque uno.

# Estructuración de los proyectos

Los proyectos se encuentran estructuraos en diferentes carpetas. En la raíz podemos encontrar las carpetas *Materials, Models*, *Scenes* y *Scripts.*

En la carpeta *Materials* guardaremos los materiales usados en el proyecto.

En la carpeta *Scenes* guardaremos las diferentes escenas.

En la carpeta *Scripts* guardaremos todos los scripts usados. Los más relevantes son *PlayerController* e *InputController* (este último generado con el plugin *InputSystem*).

Y, por último, la carpeta *Models* almacenará todo lo relacionado con los modelos. Esto significa que tendremos el modelo, su controlador y una carpeta para las animaciones y otra carpeta para las texturas.

# Animación

La práctica se basa en usar el componente animator que viene en unity y animar a los diferentes modelos. Por lo tanto, este es el que más trabajo ha llevado realizar y donde más contenido podemos encontrar.

En la primera parte, se ha usado un diagrama de estados, ya que disponemos de pocos y cada uno tiene una función específica y sencilla de conectar.

En cambio, en la segunda parte se han usado *Blend Trees*, debido a que disponemos de más elementos con funciones más difusas (por ejemplo, correr hacia adelante, o hacerlo a la vez girando hacia los lados). Gracias a esta función se ha podido facilitar el trabajo de la transición entre las diferentes animaciones.

Dentro del *Blend Tree* nos encontramos otro *Blend Tree* de una dimensión para el tratamiento de los diferentes *Idles* que el personaje usará. La razón de usar este modo de implementar el estado ‘desocupado’ del modelo es para relacionarlo de manera sencilla con el resto de las animaciones, ya que los *idles* tienen la misma posición XY en el árbol e implementarlos los 3 a la vez podría llevar a más complicaciones.

Además de los 3 *idles* que pide la práctica, hay que implementar un 4 que se lance cada 30 segundos que el usuario esté inactivo. Este se último se ha implementado fuera del árbol de mezcla, para poder el estado ‘*Any state’* que ofrece Unity.

El manejo de todas las animaciones *idle* se realiza mediante un script que contiene el árbol padre. El uso es sencillo: cada vez que entremos en el *blend tree* después de un salto, caída o animación especial, aleatorizaremos el idle por defecto. Esto se hace para evitar que el usuario sienta que no hay cohexión entre las acciones (por ejemplo, si el personaje pasa de repente de un *idle* “feliz” a uno “triste”). Este *script* también es el responsable de calcular el tiempo inactivo y activar el especial.

En la capa “inyured” no se han implementado diferentes animaciones para la inactividad, debido a que se ha pedido usar un paquete de animaciones de máximo, el cual solo incluye un idle. En caso de implementarlo sería parecido al anterior.

# Caída Libre

Por último, el cálculo de si el usuario se encuentra en caída libre se encuentra en el *script* de movimiento del usuario. Debido a que en la memoria no se especifica cuanta altura ha de caer el usuario para empezar la animación, se ha declarado la variable pública *fallingThreshold* para que el profesor sea el que determine este límite. Durante la animación, el resto de las animaciones quedan bloqueadas.

Esta animación finaliza cuando el usuario toca una superficie. Esto es detectado mediante un ‘*capsule collider’* que el sujeto tiene en los pies.