Introducción a Unreal Engine 4 - Implantar un sistema de animaciones a un personaje en primera persona.

Autor: Juan José Gómez Simón

En este artículo se hará un explicación introductoria y básica a cómo implantar un sistema de animaciones a un modelado 3D importado mediante el uso de "*BluePrints*".

Para la realización de esta explicación, he tomado como proyecto base el creado en el artículo anterior (https://ujiplayers.github.io/numero1/articulo1/n1a1.html) y los "*assets*" usados han sido sacados de https://www.mixamo.com/. Sin embargo, lo que vamos a necesitar va a ser un modelado 3D "*riggeado*" y cinco animaciones; parado, andar hacia adelante, andar hacia atrás, andar lateralmente hacia la izquierda y andar lateralmente hacia la derecha.

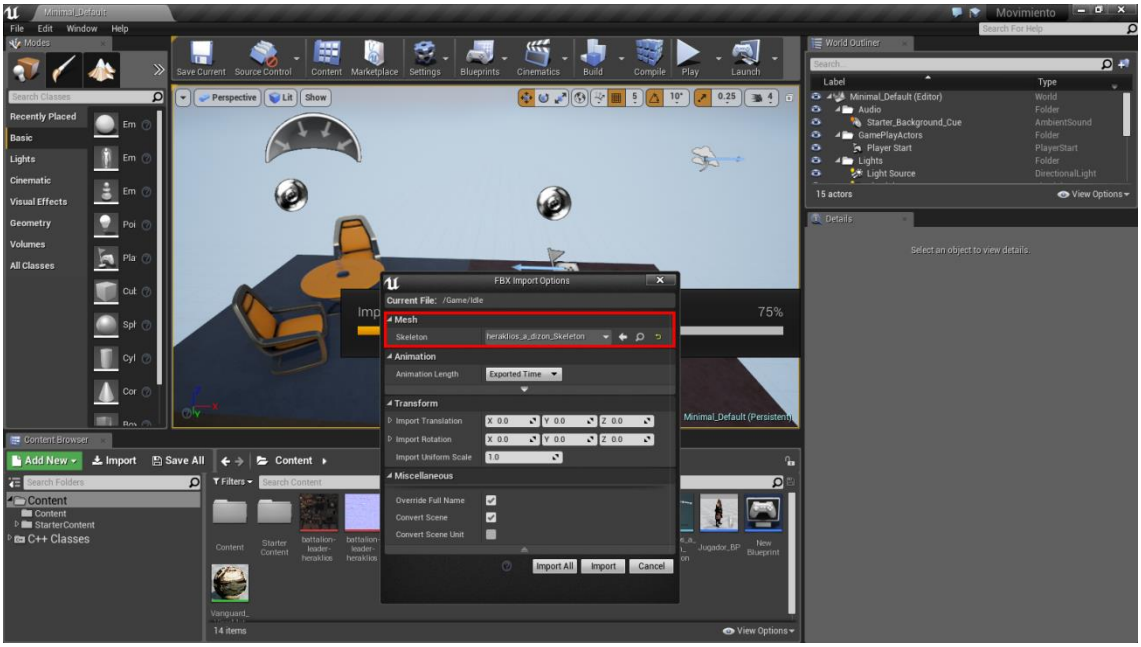
Previamente se necesita conocer dos conceptos relacionados principalmente con el mundo de la animación:

- Esqueleto: https://es.wikipedia.org/wiki/Skeletal_animation
- Rigging: https://www.notodoanimacion.es/que-es-y-como-hacer-un-buen-rigging/

Preparando el proyecto

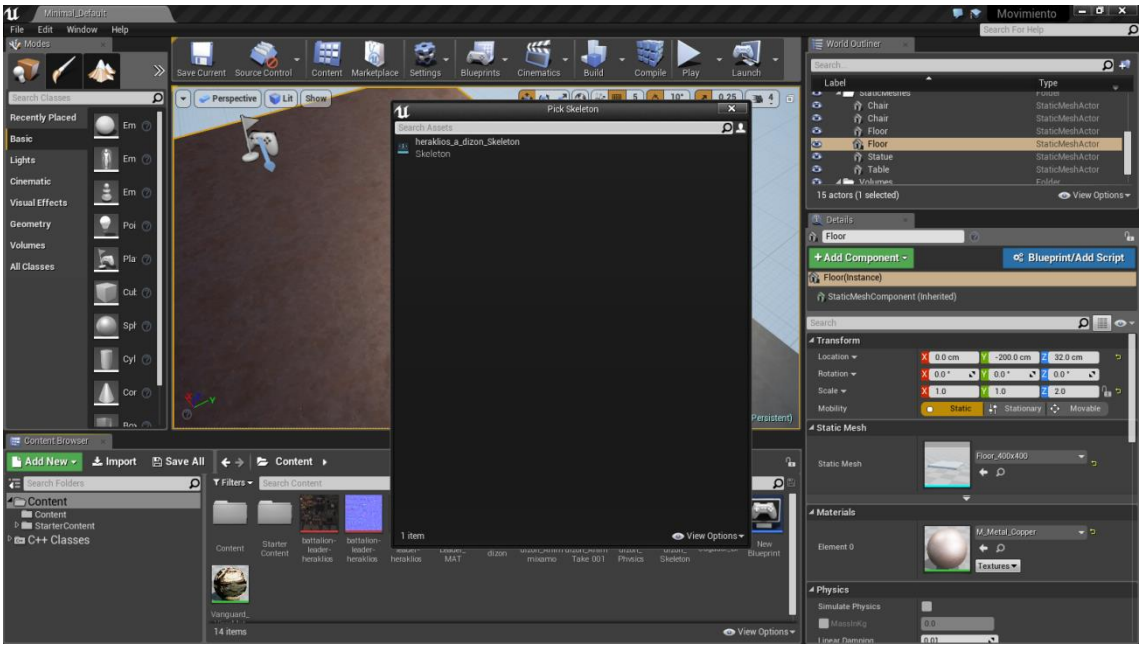
Antes de empezar, hay que importar al proyecto los modelados y animaciones que previamente hayamos descargado. Para ello, es tan simple como coger los "*assets*" y arrastrarlos al navegador de contenido o pulsando en el botón "*Import*" y seleccionando desde ahí lo que queramos.

Cabe decir que, cuando importemos tanto un modelado como una animación, saldrá una ventana que nos mostrará varias opciones de cómo queremos hacerlo. En nuestro caso, no hace falta tocar nada a excepción de cuando importemos las animaciones elijamos el esqueleto del personaje que vayamos a usar.

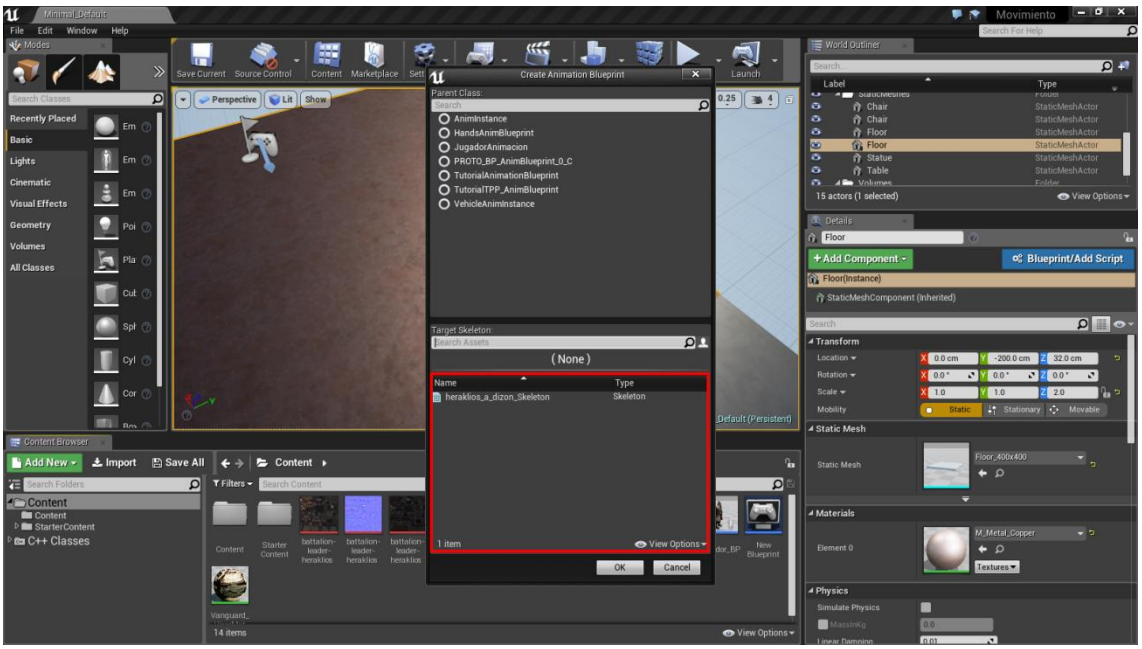


Después, crearemos dos elementos fundamentales que compondrá nuestro sistema de animaciones. Uno de ellos es el "*BlendSpace*" y otro es el "*Animation BluePrint*". Ambos se explicarán más adelante.

Para crear el "*BlendSpace*[1] ", haremos *click* derecho en el navegador de contenido, iremos a la categoría "*Animation*" y seleccionaremos "*BlendSpace* ", el cual nos pedirá que elijamos el esqueleto que vayamos a usar en este sistema de animaciones, en nuestro caso el del jugador.



Luego para crear el "*Animation BluePrint*"[2], haremos el mismo proceso, *click* derecho en el navegador de contenido->"*Animation*"->"*Animation BluePrint*", y nuevamente nos pedirá el esqueleto que queremos usar para el sistema que será el mismo que elegimos para el "*BlendSpace*".

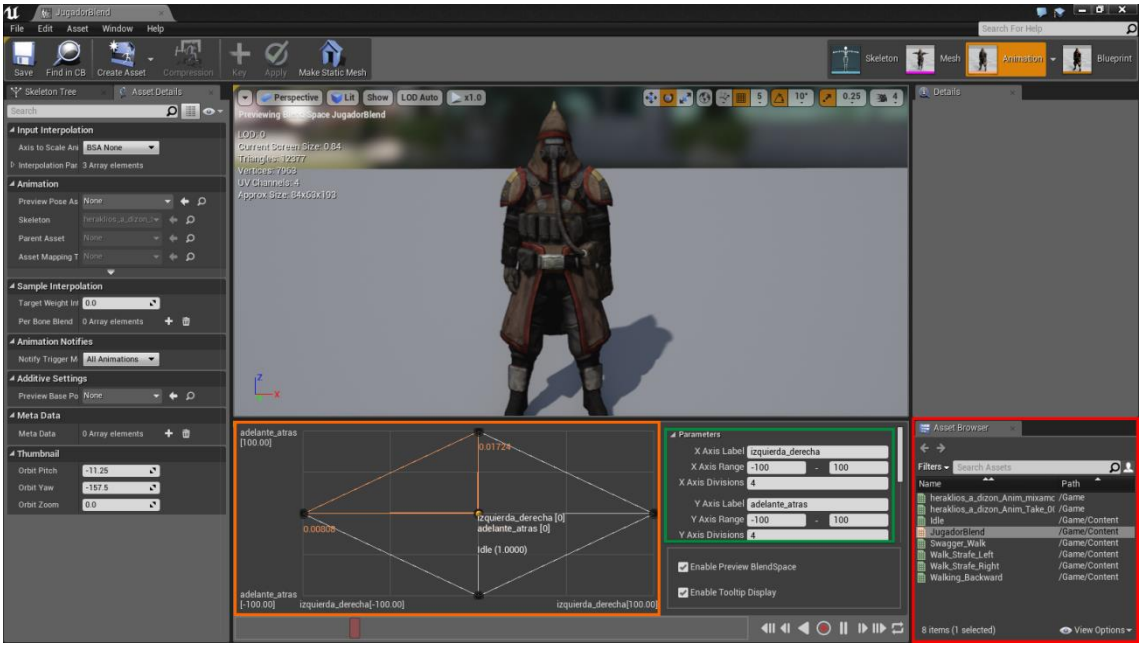


[1] Al cual lo he nombrado como *JugadorBlend*.
[2] El cual lo he denominado *JugadorAnimacion*.

BlendSpace

El "*BlendSpace*" es el encargado de ejecutar las animaciones respecto a dos parámetros dados y dependiendo del valor que tengan se ejecutarán unas animaciones u otras. ¿Cuál es la ventaja de usar esto en vez de ejecutar las animaciones directamente? Principalmente una de las ventajas y la que se cubrirá en este artículo es que se hace la transición automáticamente.

Por ejemplo, supongamos que tenemos una animación de parado y otra de correr, entonces si ejecutamos las animaciones directamente solo estarían esos dos estados. Sin embargo, mediante este método podemos estar parado, corriendo y andando a diferentes velocidades según el valor del parámetro introducido.



Una vez abramos el "*BlendSpace*" veremos varias ventanas. Sin embargo, solamente le prestaremos atención a tres que están marcadas con diferentes colores.

La ventana con el marco rojo es el navegador de contenido del sistema de animaciones, aquí aparecerá tanto los "*BlendSpace*" y las animaciones que ejecutaremos. Pero antes de eso debemos configurar los parámetros que vamos a usar y para ello enfocaremos nuestra atención en el recuadro verde.

En este recuadro, hay dos parámetros como ya mencionamos previamente (los cuales podemos nombrar como queramos), el primero representa el eje X y el otro el eje Y, (más adelante se explicará por qué, lo menciono para que esté presente). Cada uno tiene tres propiedades, el nombre[3], los valores mínimos y máximos que puede tener [4] y en cuantas partes queremos que se divida el gráfico del recuadro naranja (cuatro está bien).

[3] Al eje X lo he llamado *izquierda_derecha* y a la Y *adelante_atras*.
[4] en este caso lo he puesto a 100 y -100 cada uno porque es la velocidad máxima que puede alcanzar este personaje en las cuatro direcciones ya que solo puede andar.

Una vez configurado los parámetros, pasaremos finalmente al recuadro naranja que es donde se colocarán las animaciones. Como se ha mencionado previamente, este gráfico está compuesto por los ejes X e Y representados por un parámetro distinto, entonces, dependiendo del valor que tenga cada uno la animación irá cambiando de un estado a otro respecto a los valores de X e Y.

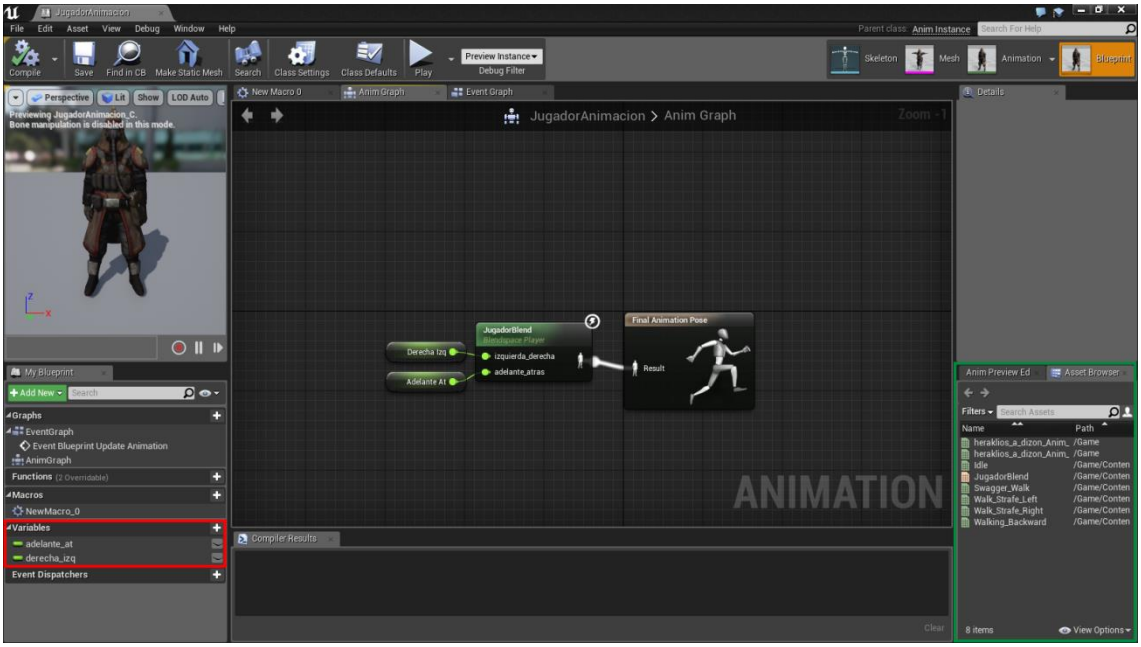
Tomando como ejemplo la imagen, hemos situado la animación de andar hacia adelante cuando *adelante_atras* (es decir en el eje Y) valga 100 y la animación de andar hacia atrás cuando valga -100, lo mismo hemos hecho con *izquierda_derecha* (es decir en el eje X), cuando valga 100 se ejecutará completamente la animación de andar lateralmente hacia la derecha y cuando valga -100 hacia la izquierda.

Sin embargo, supongamos que *adelante_atras* e *izquierda_derecha* valen 50, ¿qué animación ejecutaría? pues aquí es donde radica el potencial de los "*BlendSpace*", lo que ejecutaría sería una mezcla entre un andar lento y a la par un desplazamiento diagonal hacia la derecha quedando así una transición fluida y sin necesidad de tener una animación específica para ello.

Animation BluePrint

Una vez creado y configurado el "*BlendSpace*" pasaremos al "*Animation BluePrint*" el cual se encarga de determinar cuándo y cómo se ejecutan las animaciones mediante el uso de los "*BluePrints*" que es un método de scripting visual en el cual tenemos nodos que podemos unir para hacer todo tipo de operaciones(sin embargo no se recomiendan usar mucho ya que son lentos en tiempo de ejecución).

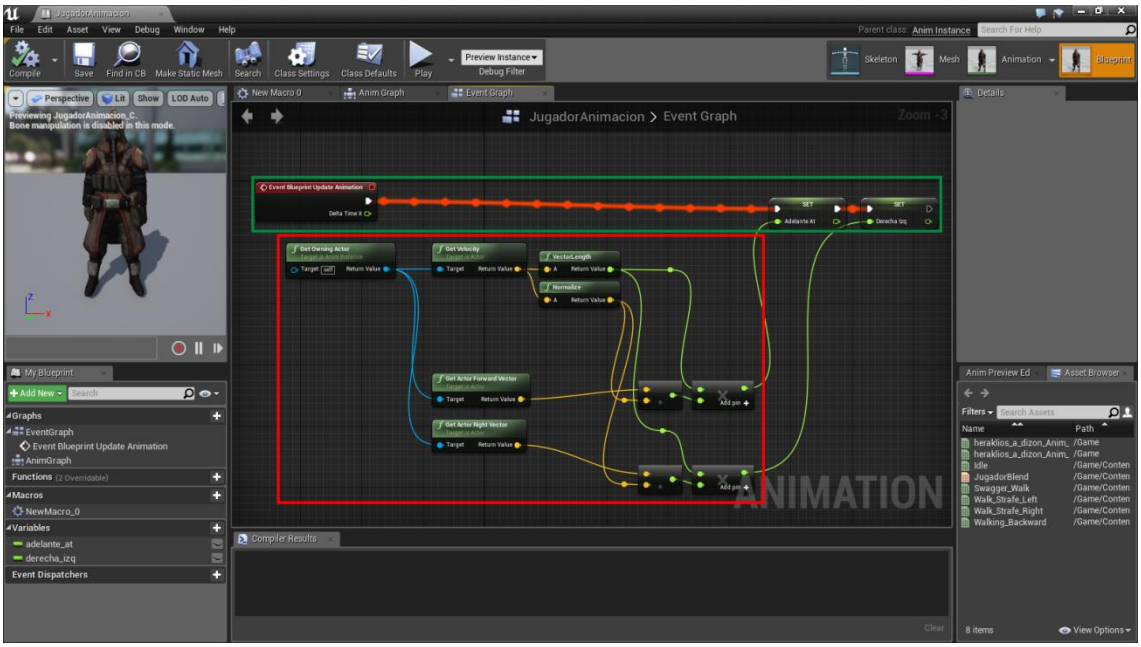
El "*Animation BluePrint*" tiene dos pestañas, una en donde programaremos las animaciones via "*BluePrint*" y la otra en donde colocaremos el "BlendSpace" para que finalmente las animaciones pasen al esqueleto del jugador.



Primero crearemos dos variables (en mi caso las he llamado *adelante_at* y *derecha_izq*) que serán las que cambien el valor de los parámetros *adelante_atras* e *izquierda_derecha respectivamente*, después desde el navegador de contenido cogeremos y arrastraremos el "*SpaceBlend*"(llamado en mi caso *JugadorBlend*) a la ventana central.

Finalmente, hacemos *click* derecho en la ventana central y escribimos en el cuadro de búsqueda que habrá aparecido "*get* [nombre de la variable que has creado]" haciendo aparecer así un nodo el cual obtendrá el valor de la variable escrita, es decir un "*getter*", hacemos esto dos veces para obtener el valor de ambas variables creadas.

Una vez tengamos estos dos nodos los unimos respectivamente a sus parámetros *adelante_atras* e *izquierda_derecha* y después la salida del "*SpaceBlend*" lo unimos a la entrada del "*Final Animation Pose*" que será el que transmita la animación al esqueleto del personaje.



Ahora, lo que haremos será asignar el valor de la velocidad del jugador a sus respectivas variables para que así se ejecute la animación adecuada o su correspondiente transición.

Primero, nos fijaremos en el recuadro rojo, en donde vemos los nodos encargados de hacer las operaciones para calcular la velocidad y dirección del jugador, para ello primero se crea un nodo "*Get Owing Actor*" el cual obtiene una referencia del objeto que está ejecutando este sistema de animaciones (en este caso el jugador), desde su salida se crean tres más; "*GetVelocity*"[5], "*GetActorForwardVector*" y "*GetActorRighVector*"[6].

Seguidamente desde la salida del nodo "*GetVelocity*" crearemos dos más, "*VectoLength*"(para convertir el vector de velocidad en un valor flotante siempre positivo obteniendo así la velocidad máxima del jugador) y "*Normalize*"(retorna una versión unitaria del vector).

Después crearemos dos nodos "*DotProduct*"(que devuelven el producto del punto entre dos vectores) de los cuales a la entrada del primer nodo le introduciremos la salida del nodo "*Normalize*" y del "*GetActorForwardVector*", respecto al segundo "*DotProduct*" le introduciremos también el nodo "*Normalize*" y el nodo "*GetActorRightVector*".

Posteriormente, crearemos dos nodos de multiplicación entre valores flotantes, al primer nodo le introduciremos la salida del "*VectorLenght*" y del "*DotProduct*" perteneciente a "*GetActorForwardVector*", respecto al segundo nodo introduciremos también la entrada del "*VectorLenght*" y la entrada del "*DotProduct*" correspondiente al "*GetActorRightVector*".

Con esto, lo que hemos hecho ha sido obtener el vector global de velocidad del jugador tanto en el eje X como Y y transformarlo a nivel local haciendo así que si nos desplazamos hacia adelante siempre nos dará un valor positivo sin importar la dirección a la que apuntemos y si vamos hacia atrás siempre nos dará un valor negativo(lo mismo para la derecha e izquierda).

---

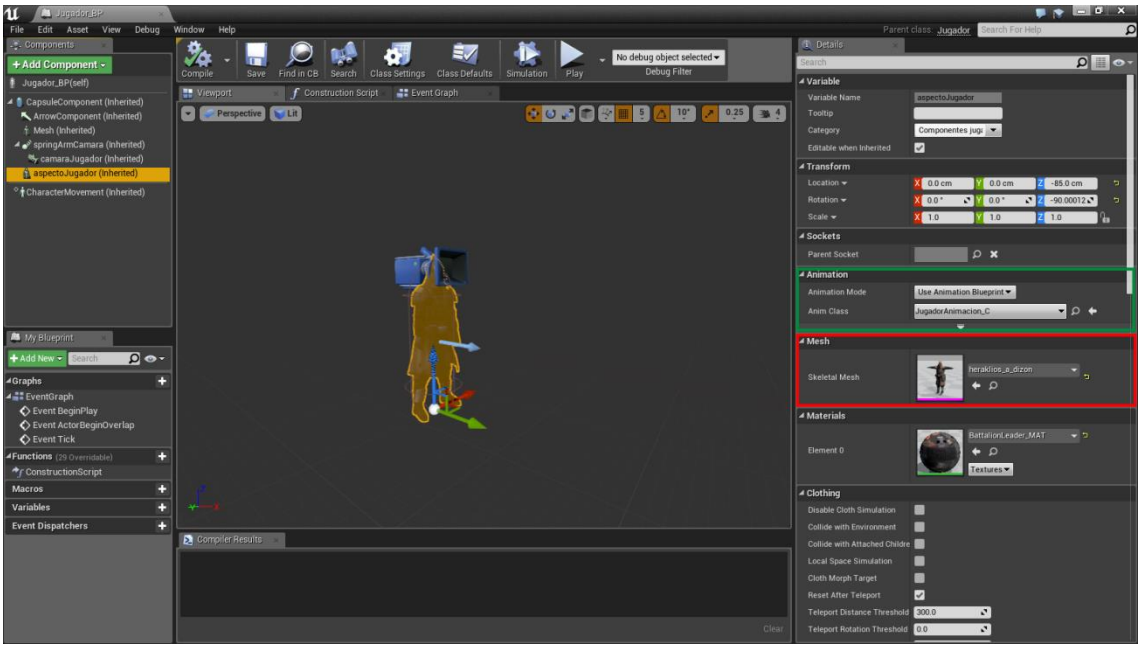
[5] Para obtener un vector de la velocidad del jugador en unas coordenadas globales.
[6] (para obtener un vector de la dirección hacia la que está apuntando en los ejes X e Y respectivamente)

Finalmente, como se puede observar en el recuadro verde de la imagen, hay un nodo evento llamado "*Blueprint Update Animation*" el cual se ejecuta cada vez que se actualiza una animación. Desde aquí crearemos dos nodos "s*et*[7] [nombre de tu variable[8]]" que serán las variables previamente creadas y en sus entradas introduciremos la salida de las respectivas multiplicaciones (en mi caso la multiplicación del "GetActorForwardVector" iría con *adelante_at* y la del "GetActorRightVector" iría con *derecha_izq*).

Con esto por fin hemos enlazado el movimiento del jugador con los parámetros del "*SpaceBlend*" consiguiendo así que se ejecute las animaciones y todas sus posibles combinaciones acordes a la velocidad y dirección del jugador.

[7] Se encarga de establecer un nuevo valor a la variable.
[8] En mi caso *adelante_at* y *derecha_izq.*

Configurando el personaje



Finalmente abrimos el objeto del jugador, seleccionamos el "*Skeletal Mesh*" y en el apartado "*Skeletal Mesh*" seleccionamos nuestro personaje y en "*Animation*", más concretamente en "*Anim Class*" seleccionamos nuestro "*Animation BluePrint*" creado y ya estaría listo para moverse, aquí dejo una demostración del resultado final https://youtu.be/e9sB4Z833QA.

Como última anotación, para que se vea el cuerpo simplemente he puesto la cámara a la altura de los ojos del "*Skeletal Mesh*", sin embargo si se hace esto lo recomendable es cortar la cabeza del modelado para así evitar efecto de "*clipping*".

Y con esto se acabó, espero que te haya servido y muchas gracias por leerme.

Si quieres preguntar algo puedes hacerlo vía Twitter(@GJuanjo1999), e-mail o por mi página web(http://juanjosegomez.x10host.com/).