



Desarrollo de Videojuegos

Jugabilidad: Arquitectura y
mecánicas principales
Avatar y movimiento

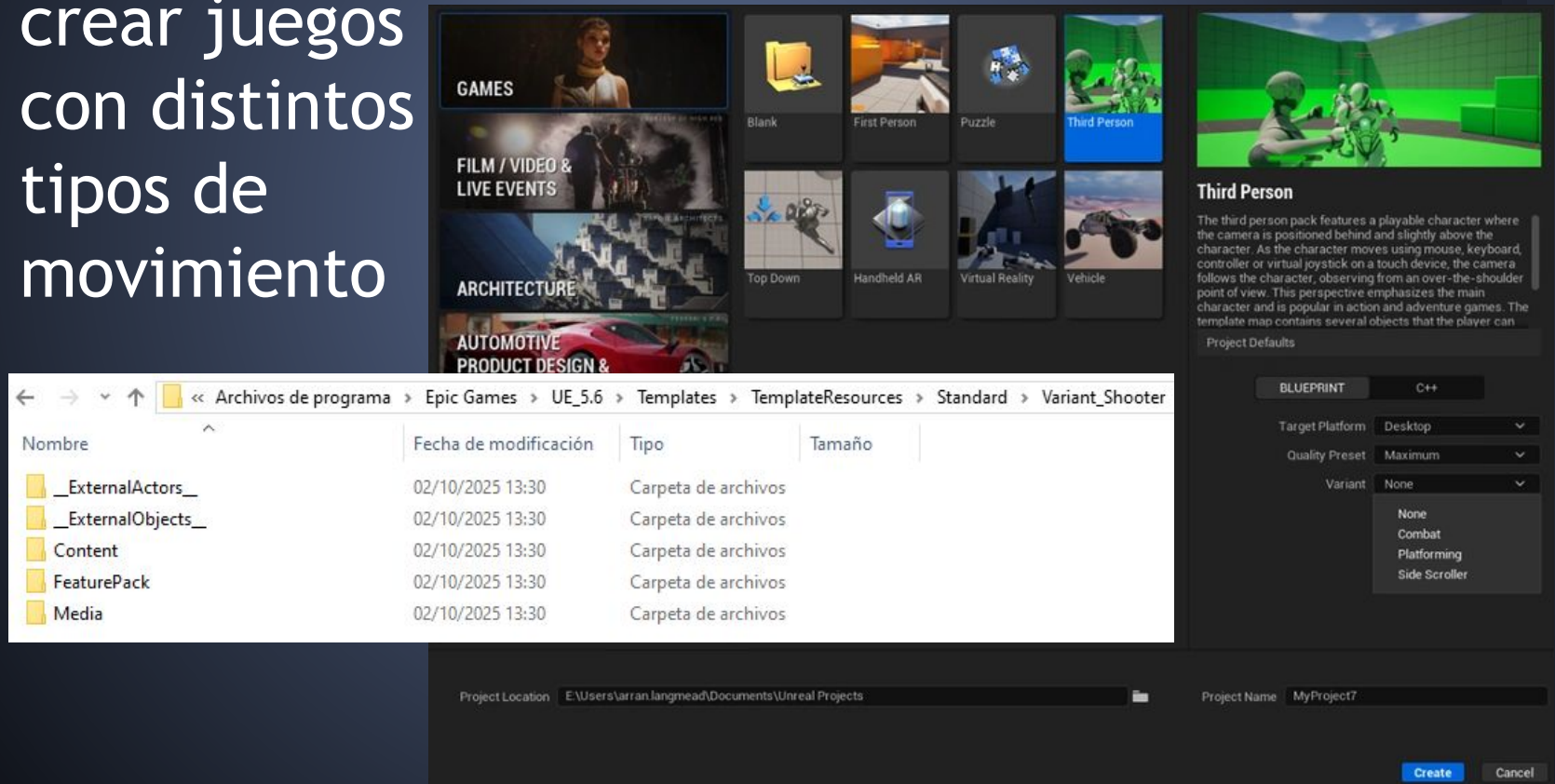
Motivación

- El videojuego es multimedia *interactivo*...
 - ... los avatares se controlan mediante **entrada/salida**



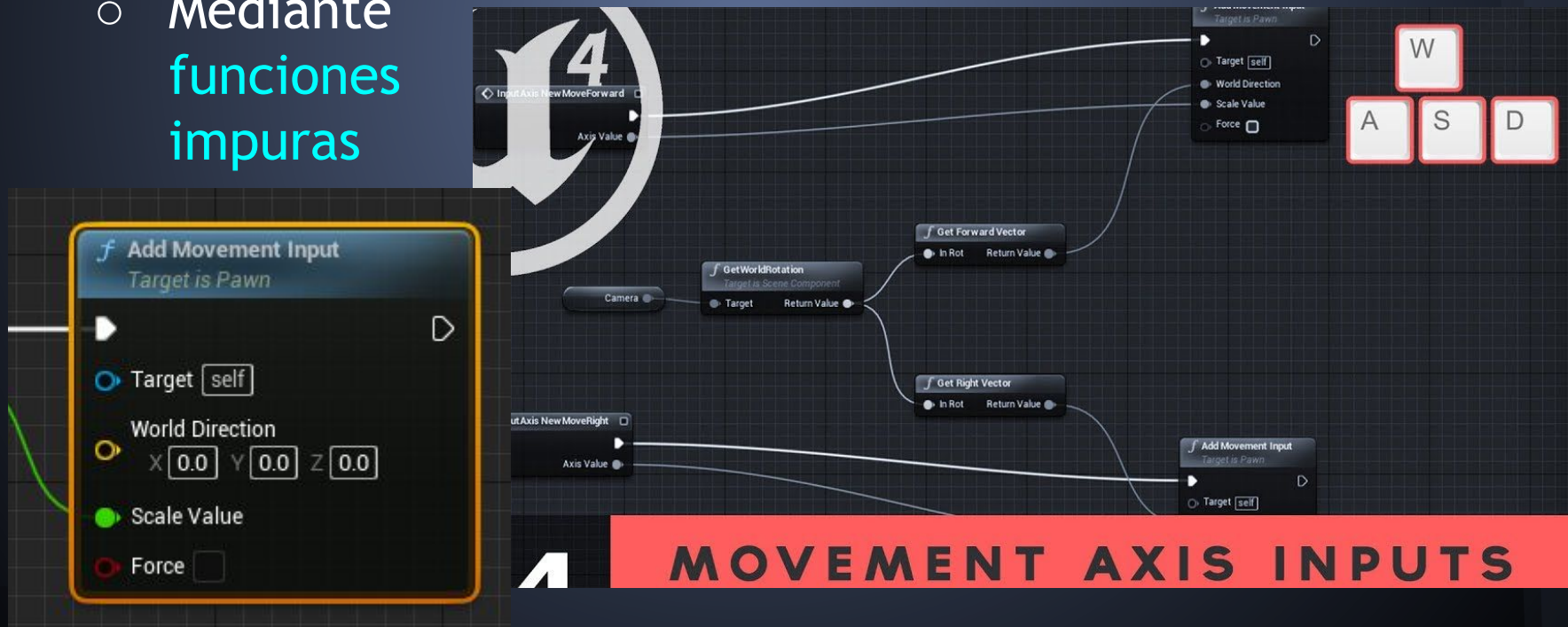
Motivación

- Hay **plantillas con variantes** para empezar a crear juegos con distintos tipos de movimiento



Percepción

- El jugador percibe el estado del mundo desde su **vista**, introduce su **entrada** y el mundo cambia: su avatar **se mueve**
 - Mediante **funciones impuras**



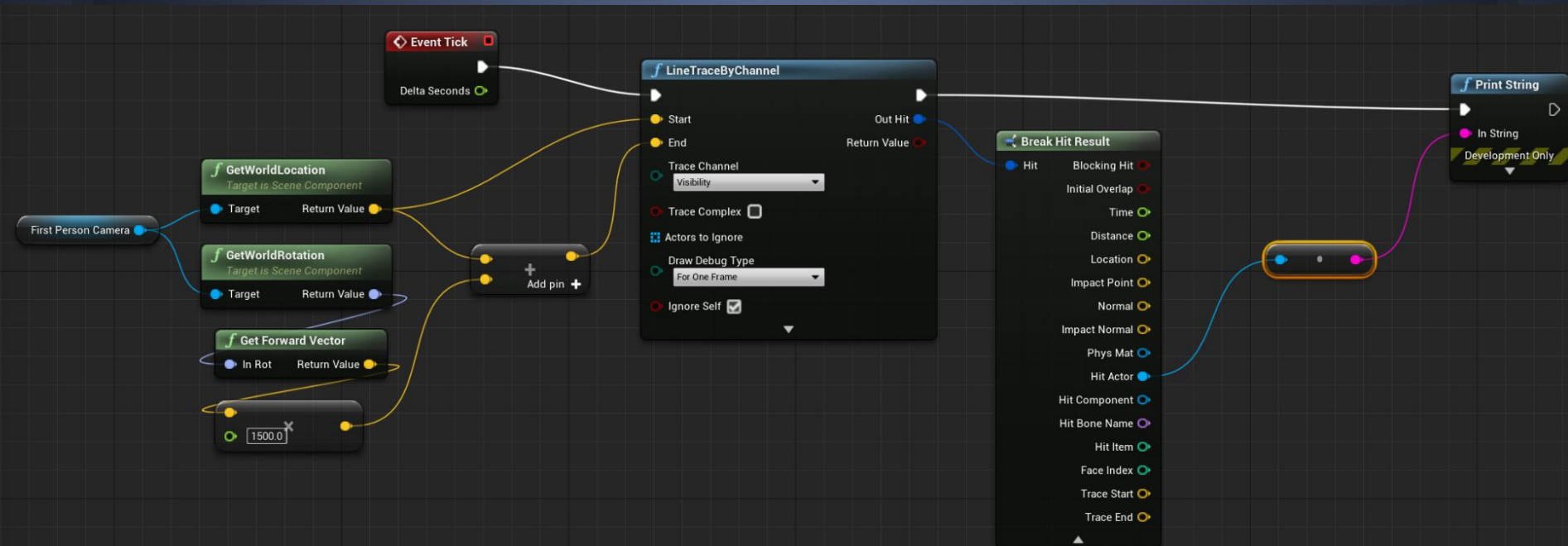
Percepción

- Habitualmente se usan **trazas** (proyección de **rayos**) para “detectar” suelo, paredes, etc.

TRACES

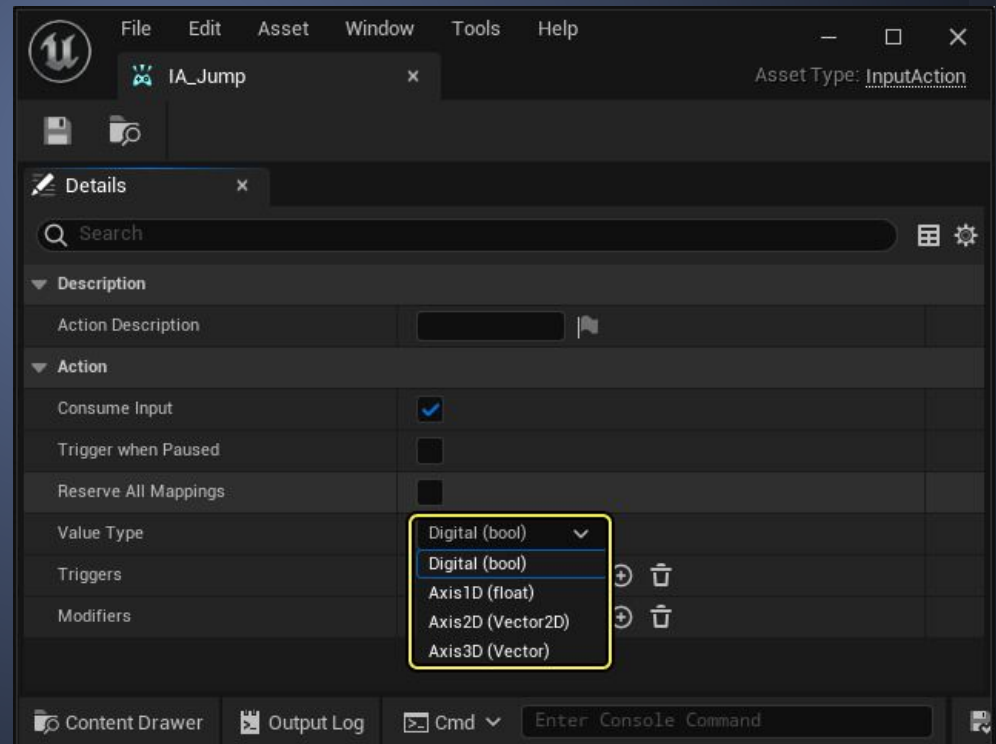
RAYCASTING

(proyección de rayos)



Configuración de la entrada

- Dentro de la configuración del proyecto, consulta la sección **Engine / Enhanced Input**
 - **Input Actions**, acciones lógicas (digitales, analógicas o vectoriales)
 - **Input Modifiers**, preprocesadores para modificar las señales de entrada
 - Ej. cambiar ejes, manejar zonas muertas o convertir del espacio de la entrada al del mundo



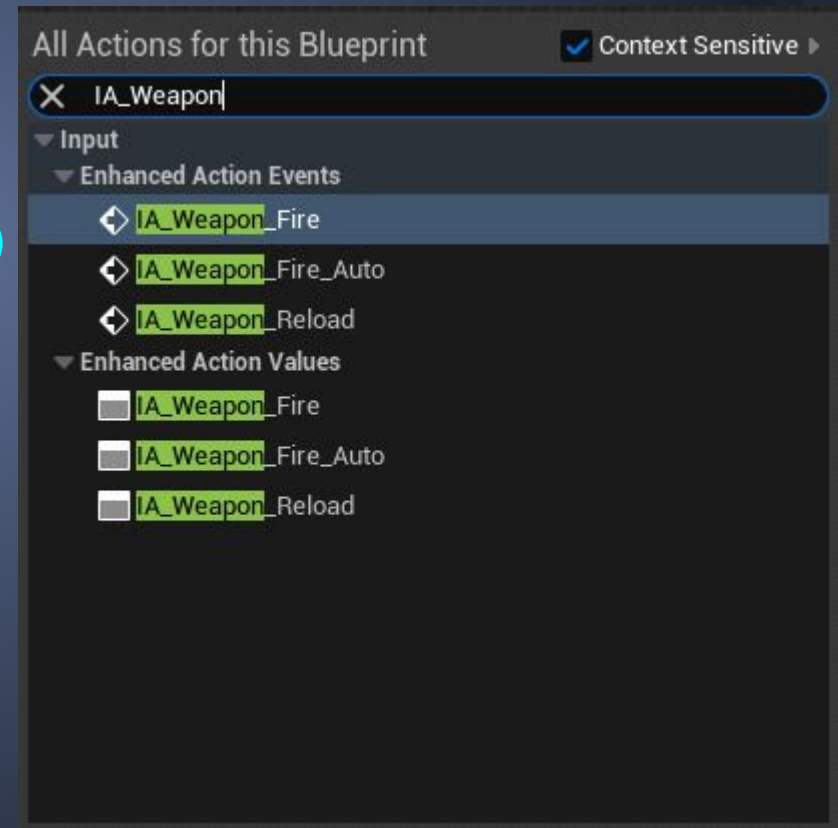
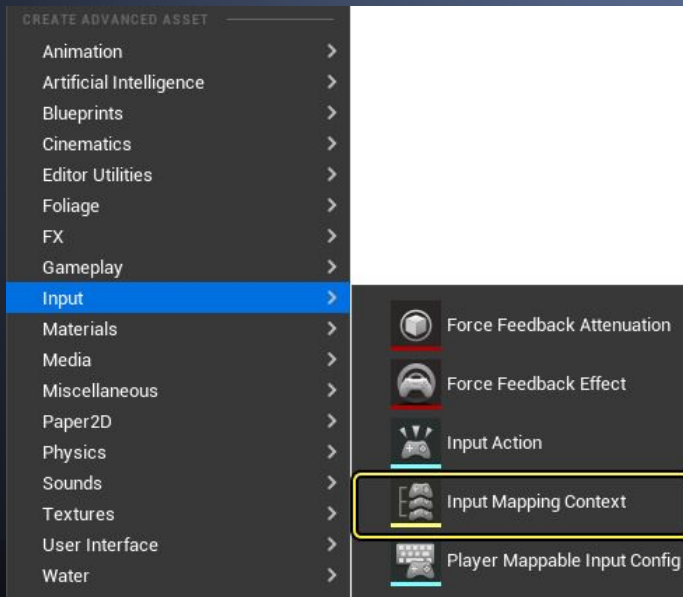
Configuración de la entrada

- **Input Triggers**, filtros (**instantáneos**, **mantenibles** o **condicionales**) para decidir si una señal (tal vez modificada) *activa o no* un *Input Action*
- Finalmente también hay **Trigger States**, los estados de las entradas
 - **Started** (ej. pulso el botón para empezar a cargar un rayo)
 - **Ongoing** (ej. cargando el rayo...)
 - **Triggered** (ej. se lanza el rayo tras cargar 2 segundos)
 - **Completed** (ej. suelto el botón tras lanzar el rayo)
 - **Canceled** (ej. suelto el botón antes de lanzarlo)

<https://docs.unrealengine.com/en-US/enhanced-input-in-unreal-engine/>

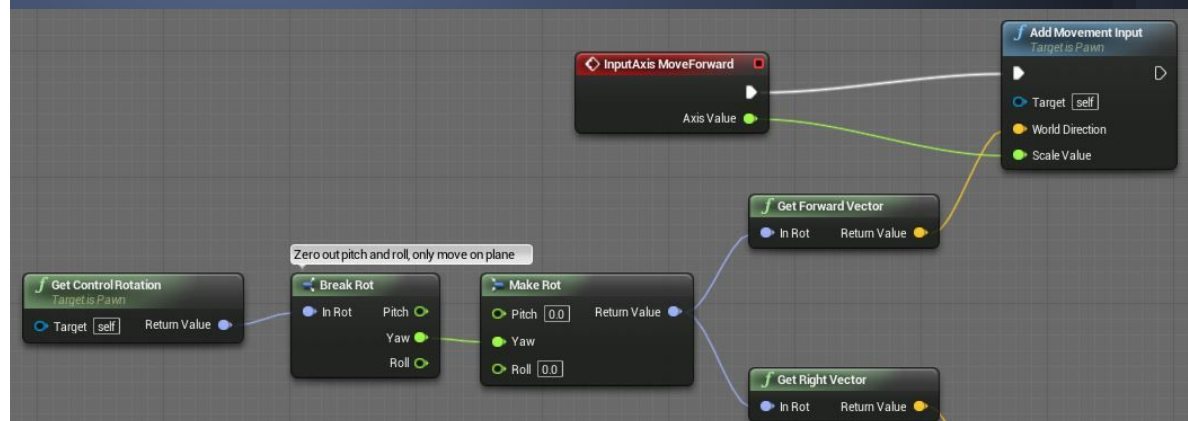
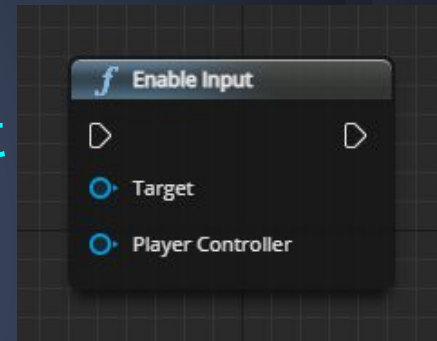
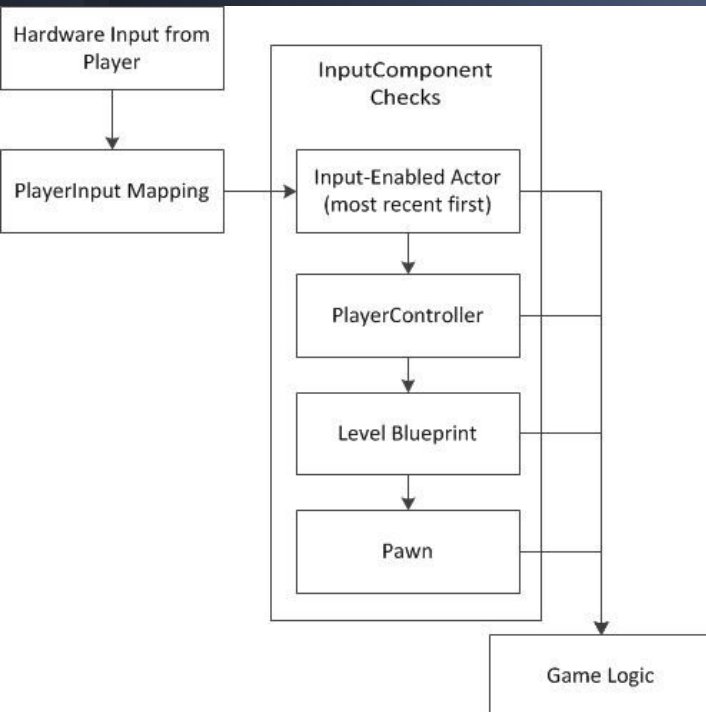
Configuración de la entrada

- En el código se ponen **Input Listeners**
- Es posible cambiar dinámicamente de **Input Mapping Context** (conectar señales con Input Actions)



Configuración de la entrada

- Los eventos de entrada de un jugador los gestiona su controlador
 - Salvo que haya actores con **Enable Input** activo para gestionar dichas entradas

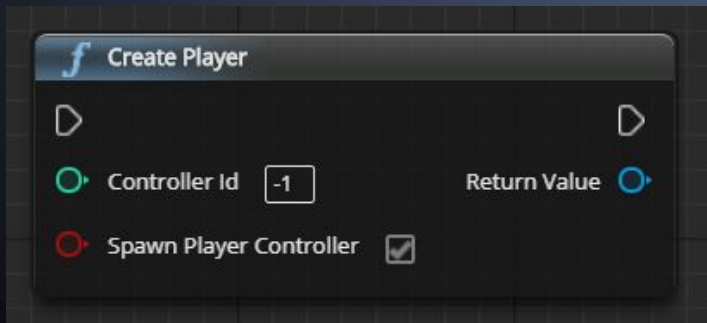
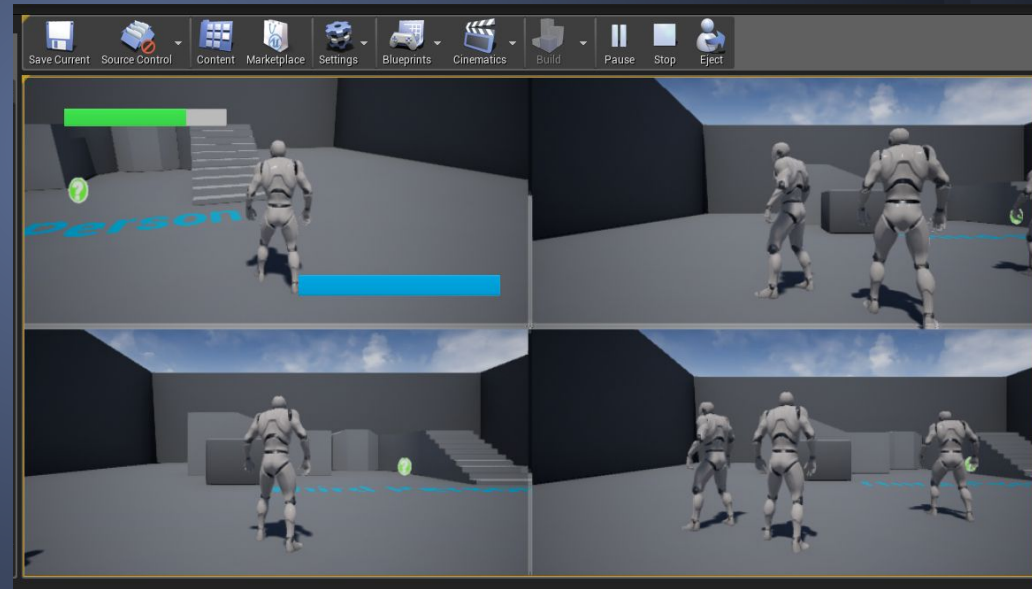
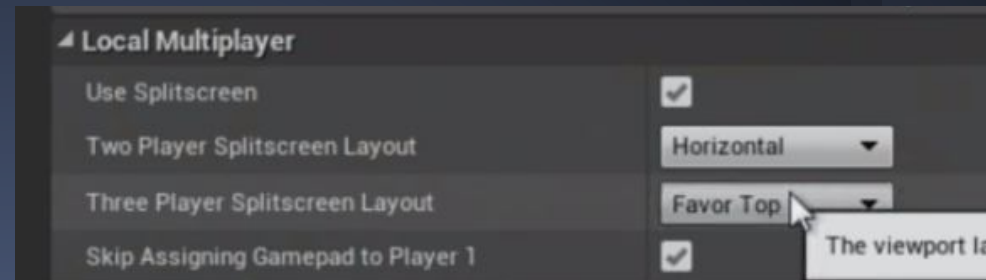


Ejemplo: Ratón

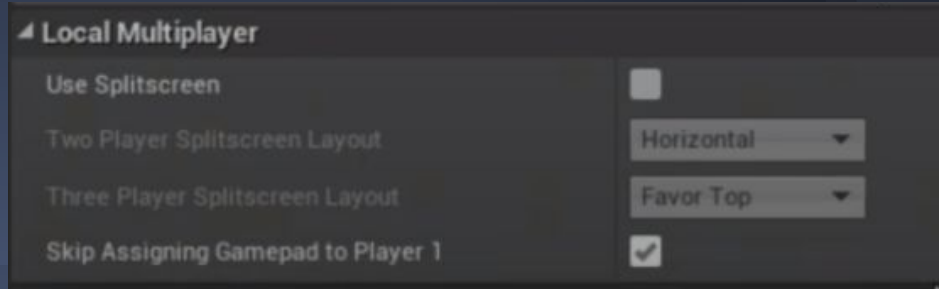
- Cada fotograma envía **dos números reales** con valores en un rango distinto según el movimiento
 - Muy rápido (-50 y 50... *depende del HW*)
 - Normal (-10 y 10)
 - Suave (-1 y 1)
- **MouseX** positivo significa *a la derecha*
- **MouseY** positivo significa *hacia abajo*
 - Es habitual invertirlo con un **Input Modifier** de -1 que se pone en el **Input Mapping Context**

Multijugador local

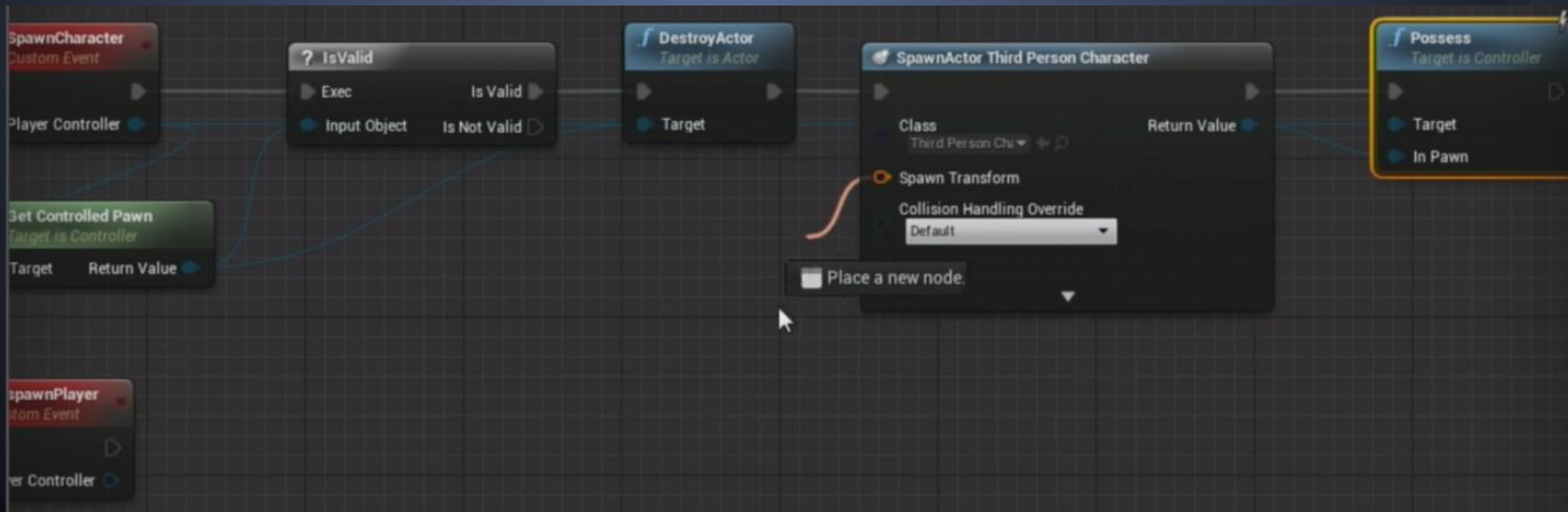
- Podemos tener varios jugadores, cada uno con su controlador y su **gestión de entrada**
 - Puedes **crear nuevos jugadores**, indicando siempre su **ID** (-1 significa “el siguiente que esté libre”)



Multijugador local

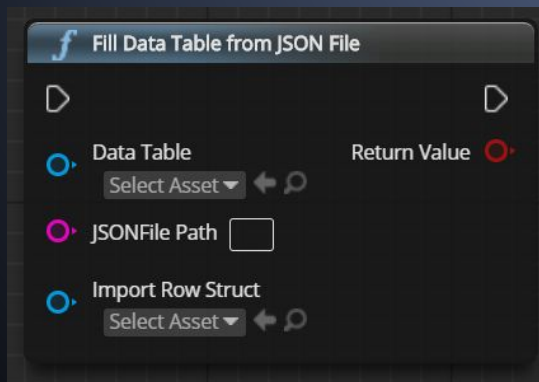


- Según el juego se podrá configurar cómo asignar los *gamepads* a los jugadores, o si hay una sola *cámara* que se comparte entre todos los controladores



Tablas de datos y curvas

- También son “entrada” los datos, que podemos meter en **DataTables** y **CurveTables**
 - Hojas de cálculo que cargar desde fichero **CSV** o **JSON**



Name	XPtoLvl	XP	Asset
1	0	0	/Game/UI/HUD/Actions/Barrel
2	1000	1000	/Game/UI/HUD/Actions/Barrel
3	1000	2000	/Game/UI/HUD/Actions/Barrel
4	1500	3500	/Game/UI/HUD/Actions/Barrel
5	2000	5500	/Game/UI/HUD/Actions/Barrel
6	2500	8000	/Game/UI/HUD/Actions/Barrel
7	3000	11000	/Game/UI/HUD/Actions/Barrel
8	3500	14500	/Game/UI/HUD/Actions/Barrel
9	4000	18500	/Game/UI/HUD/Actions/Barrel
10	4500	23000	/Game/UI/HUD/Actions/Barrel
11	5000	28000	/Game/UI/HUD/Actions/Barrel
12	5500	33500	/Game/UI/HUD/Actions/Barrel
13	6000	39500	/Game/UI/HUD/Actions/Barrel
14	6500	46000	/Game/UI/HUD/Actions/Barrel
15	7000	53000	/Game/UI/HUD/Actions/Barrel
16	7500	60500	/Game/UI/HUD/Actions/Barrel
17	8000	68500	/Game/UI/HUD/Actions/Barrel
18	8500	77000	/Game/UI/HUD/Actions/Barrel
19	9000	86000	/Game/UI/HUD/Actions/Barrel
20	9500	95500	/Game/UI/HUD/Actions/Barrel

<https://docs.unrealengine.com/en-US/Gameplay/DataDriven/index.html>

Tablas de datos y curvas

TRUE	42	80,085	4,815,162,342	3.14
enemyShip7	53maleParis	Start Game	(7.4, 64.8, 12.3)	(90.0°, 270.0°, 180.0°)
(34.8, 9.5, 823.2) (90.0°, 270.0°, 180.0°) (134.3, 75.7, 93.17)	3.14 enemyShip7 (7.4, 64.8, 12.3)	{Larry, Curly, Moe}		
	{1,2,3,3,4}	{2,1,4,3}	{weapon, 3 armor, 7 magic, 5}	

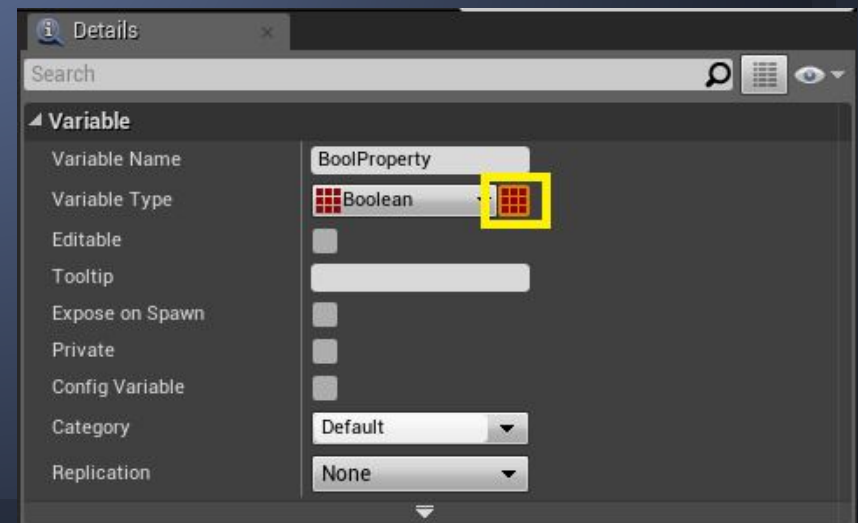
- Recuerda que una variable tendrá tipo **básico**, **referencia** o **estructura** anidable

STRUCTURE / STRUCT



* **Vector, Rotator y Transform**
son estructuras tan comunes que
aparecen en la lista inicial

- Puedes crear tus propias estructuras y **tipos contenedores**
 - Arrays, mapas y conjuntos**



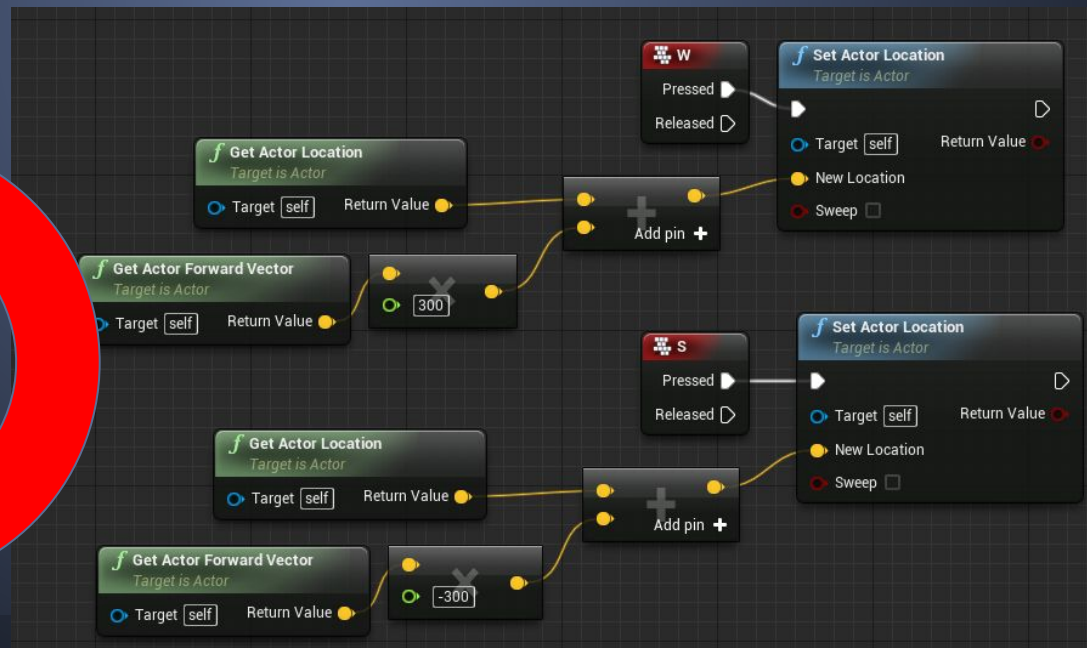
Movimiento

- Para moverse en el juego los actores deben tener activa su **movilidad**
 - Propiedad **Mobility** con valor **Moveable**
 - Por defecto son **estáticos** (inmóviles en el juego), lo ideal para calcular la iluminación
- Luego hay que saber **dónde está el actor**, la **dirección de avance** y al menos la **velocidad**
 - Transform > **Location** (vector 3 dimensiones)
 - **Dirección** (vector 3 dimensiones)
 - **Velocidad** (escalar)
 - ...

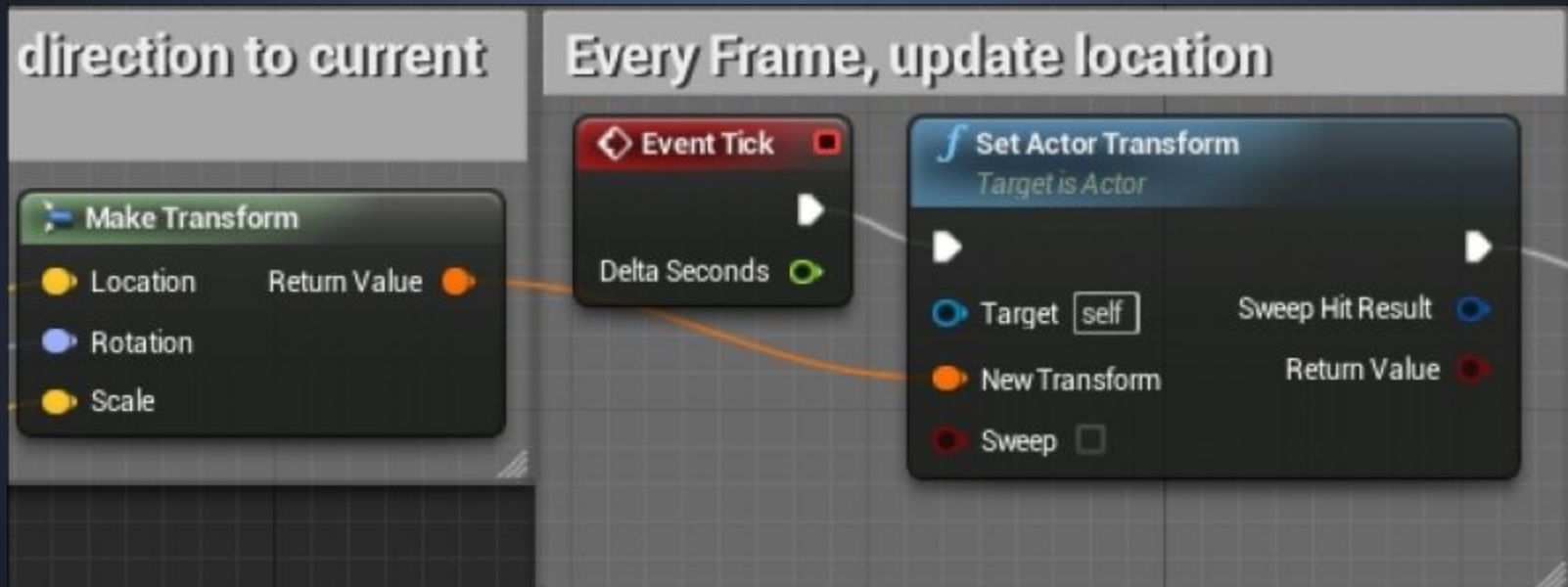


Movimiento

- En el **movimiento cinemático** no se trabaja con *aceleración* (como pasa en el dinámico)
 - En cualquier caso el código **debe depender del tiempo**, no del *render* o de la *entrada del jugador*

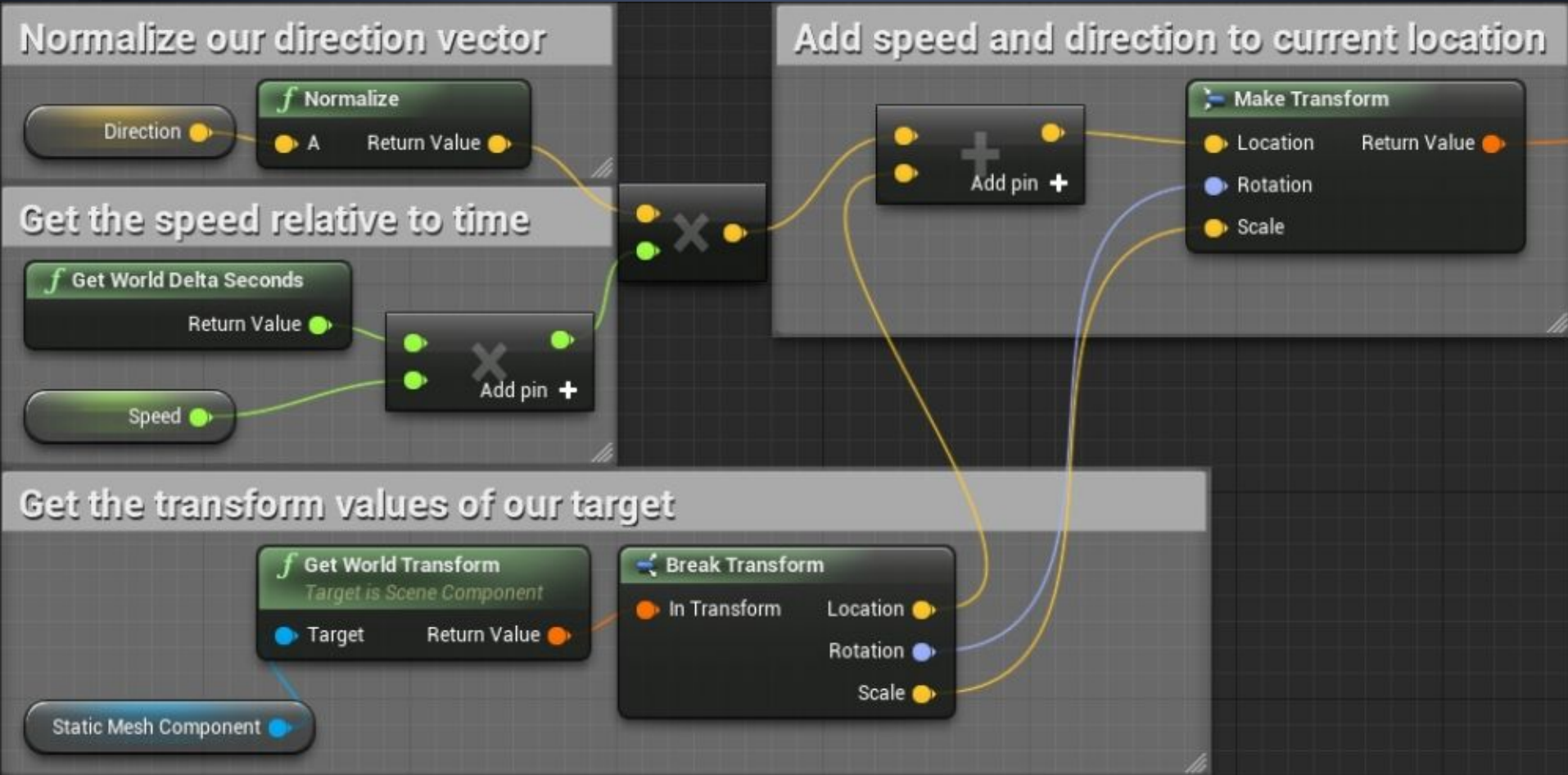


Movimiento



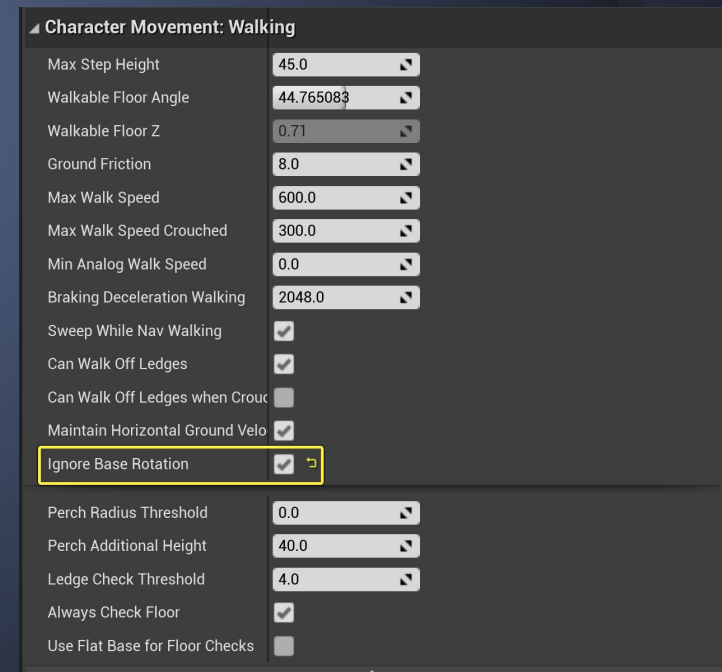
Add Actor World Offset / Add Actor Local Offset

Movimiento



Movimiento

- Ya sabemos que la “mente” de un personaje se debe programar en su *controlador*
 - Sea **PlayerController** o **AIController**
 - Desde allí se dan *órdenes* de movimiento, que ejecuta el “cuerpo”, típicamente un **componente de movimiento** del propio *peón*
 - **CharacterMovement**
 - **ProjectileMovement**
 - **RotatingMovementComponent**



Movimiento

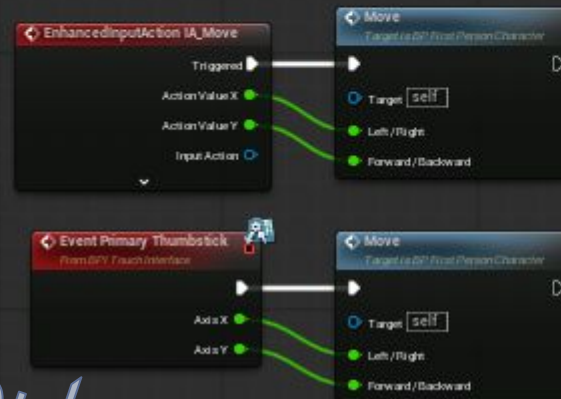
- First Person

Add Movement Input

Camera Input



Movement Input



Jump Input - Jump can be configured in the CharacterMovementComponent



Movimiento

- Third Person

Orbiting Camera Input. The CameraBoom component adopts the character's Control Rotation

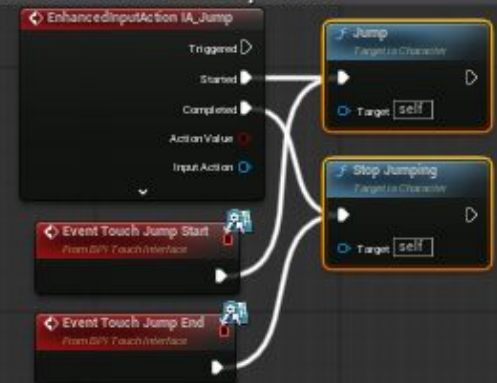


A basic Character with a controllable, orbiting Third Person camera.

Movement Input



Jump Input - Jump properties such as height, press-and-hold and multi-jump can be customized on the Character Movement Component.



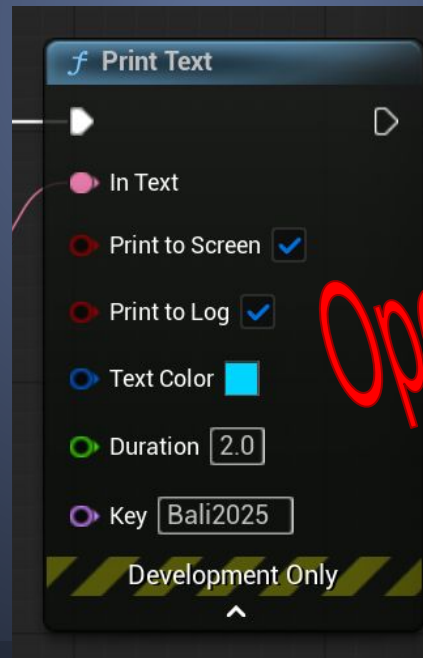
Participación

- ¿Cuál es la diferencia entre **Input Action** *digital o analógica*?
 - A. La primera es para botones y la otra para ejes
 - B. La primera ejecutan órdenes que la otra da
 - C. La primera es para acciones rápidas del ratón
 - D. Son sólo categorías para organizar nuestro código
 - E. La segunda es para acciones rápidas del ratón



Salida no diegética

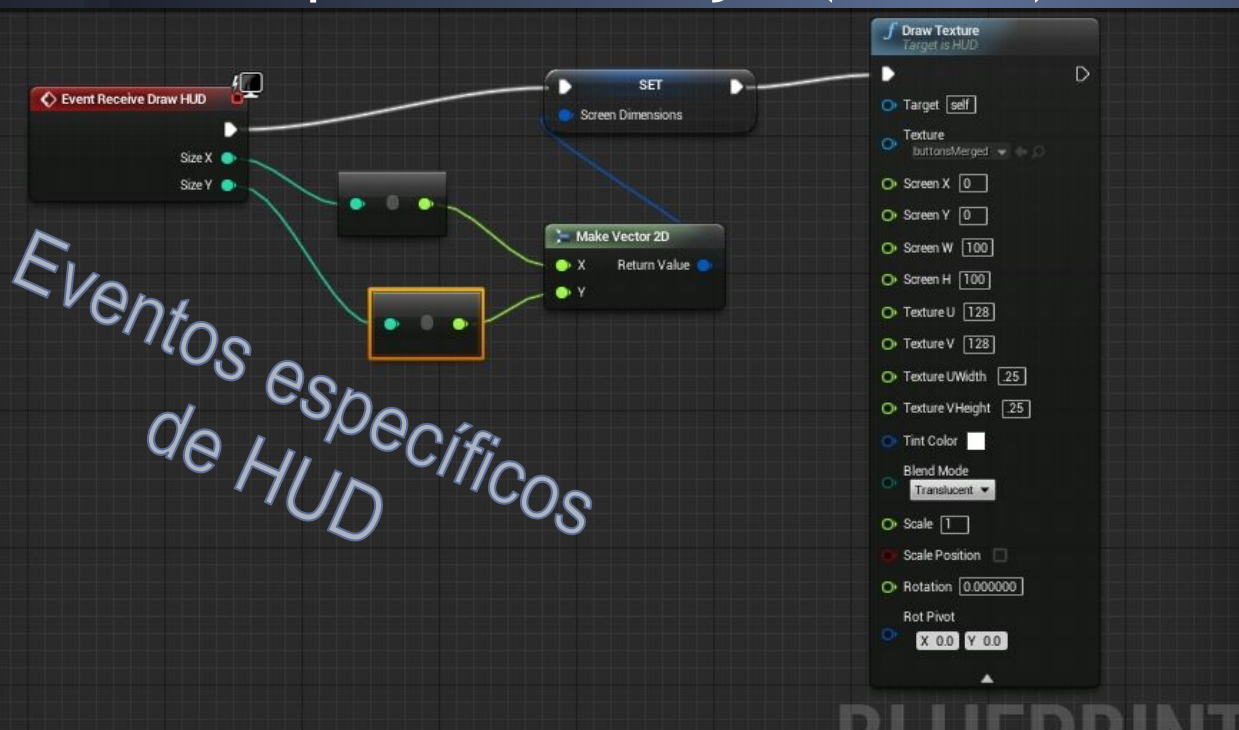
- El mundo virtual es la **salida diegética**, pero si queremos información adicional...
 - Se puede imprimir en el **registro (log)** o la **pantalla**



Opción "rápida y barata"
en depuración

Salida no diegética

- Slate es un **armazón** para interfaces de **usuario** que usa *el propio editor* de Unreal
 - Desde Blueprints tenemos funciones que nos permiten dibujar (**Draw...**) sobre el **HUD**



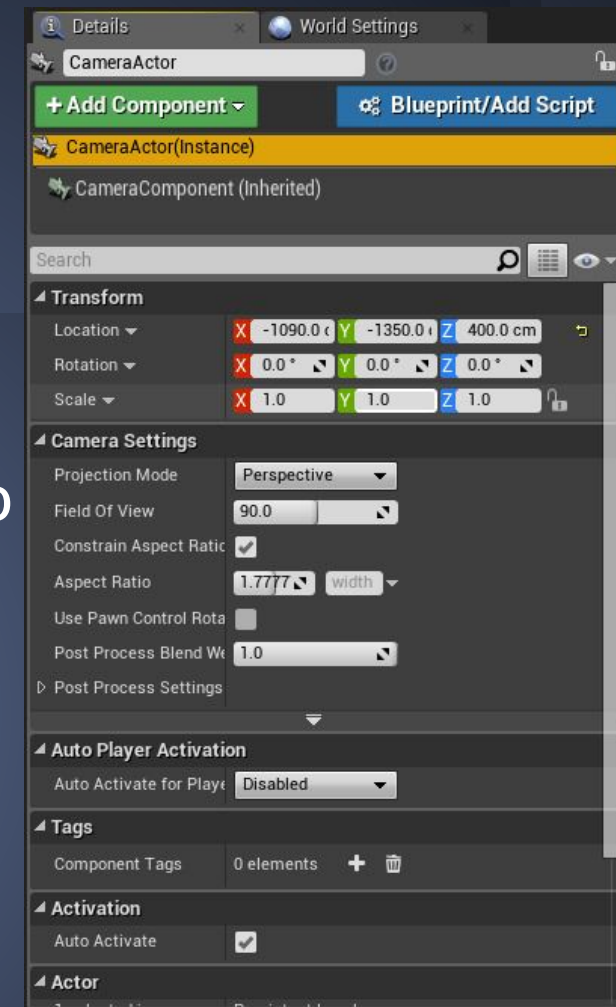
Cámaras

- La salida diegética viene de las **cámaras**, que podemos colocar como *actores* o como *componentes* dentro de otro actor
 - Ej. Cámara que graba al avatar en tercera persona
 - Al seleccionarlás, aparece su vista, que podemos **dejar fija** mientras trabajamos con otros actores



Cámaras

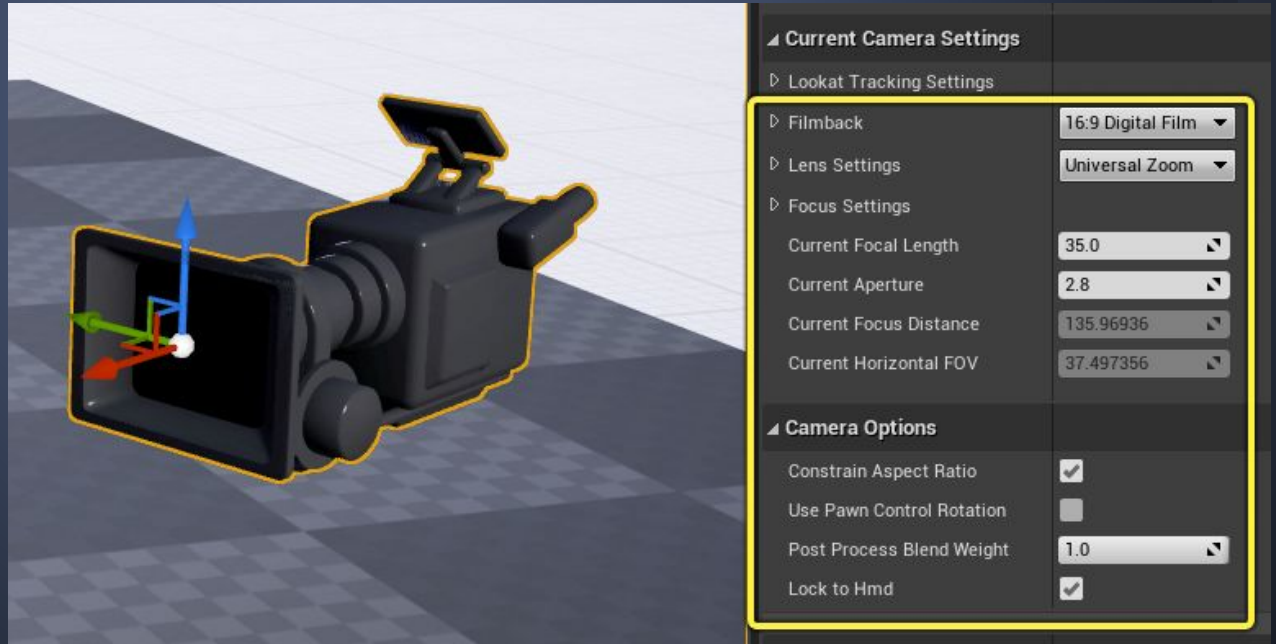
- Tienen muchos parámetros
 - Configuración (proyección, campo de visión, relación de aspecto...)
 - “Película” (tinta, saturación...)
 - configuración (tipo de proyección)
 - Resplandor BLOOM
 - Autoexposición AUTO EXPOSURE
 - Destellos de lente LENS FLARES
 - Profundidad de campo DEPTH OF FIELD
 - Desenfoque de movimiento MOTION BLUR
 - ...
- Hay componentes “grua”, posprocesados...



Cámara de cine

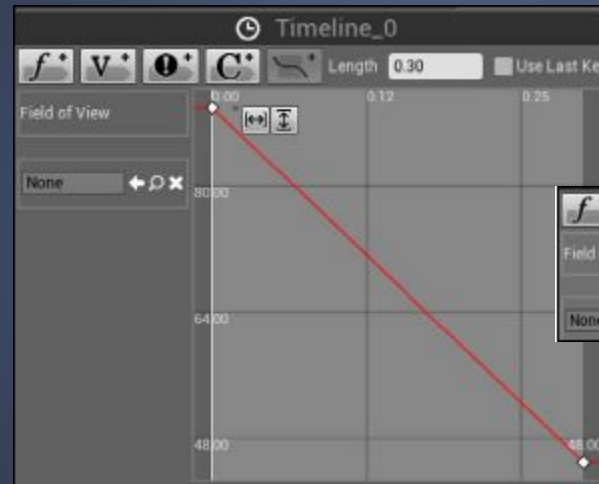


- Incluso hay cámaras que, con mayor coste, simulan **parámetros físicos** de las cámaras reales
 - Distancia focal
 - Apertura
 - Tamaño de sensor
 - ...



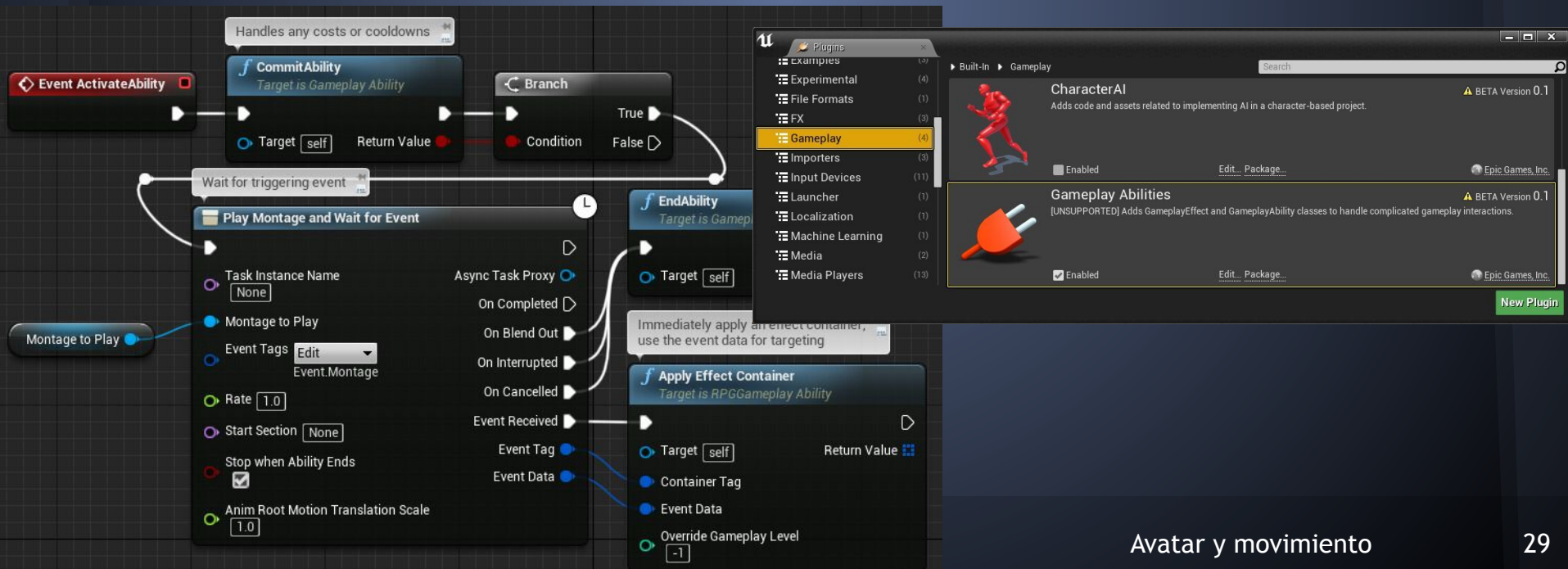
Cámaras en movimiento

- Ejemplo: Zoom con la cámara para apuntar



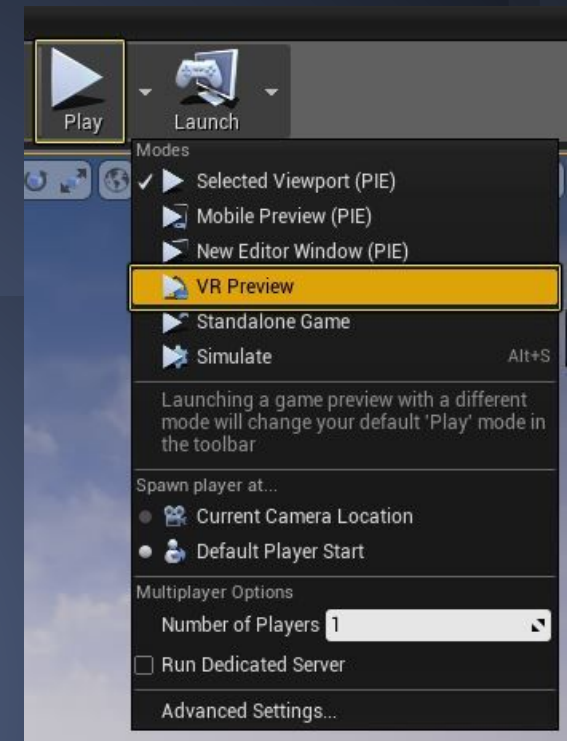
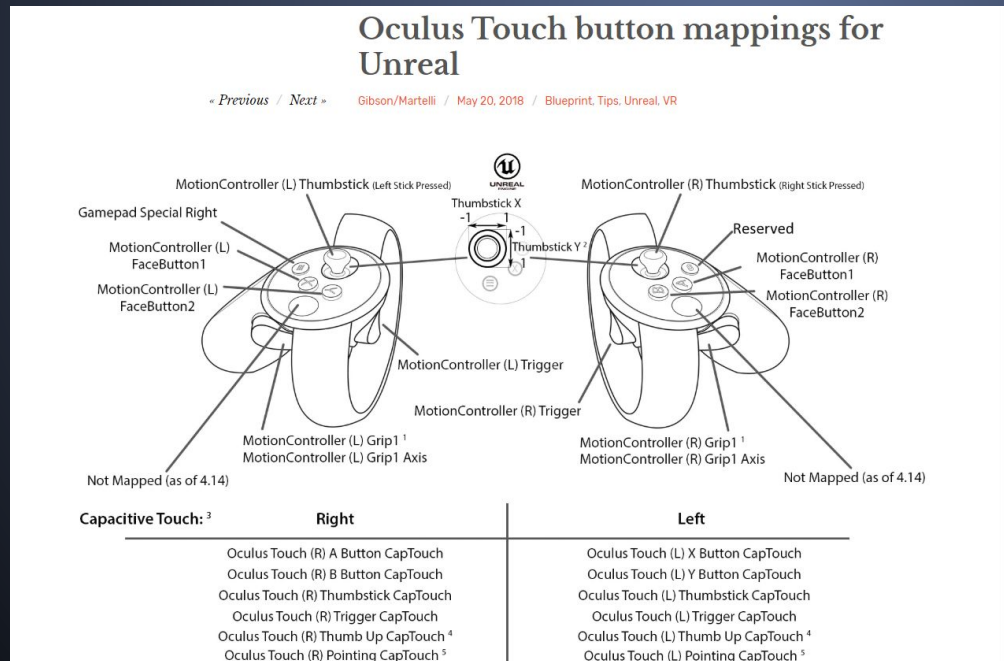
Ej. Armazón de habilidades

- ¡Unreal Engine ofrece mucho más!
- Por ejemplo, un **armazón de habilidades** sobre el que montar tu propia jugabilidad



Ej. Realidad virtual

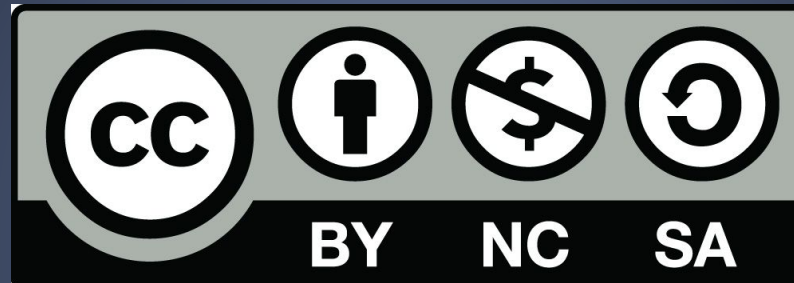
- También soporte para **Meta Quest 3**, **Steam VR**, **Playstation VR2**... y plantillas que funcionan con todo



Editor en VR

https://wiki.unrealengine.com/VR_Template

Críticas, dudas, sugerencias...



** Licencia sólo aplicable al texto original de estas diapositivas*

Federico Peinado (2019-2026)

www.federicopeinado.es