**Documento de Programa**

**General**

A

**Tools**

* Unity (ver. 2021.3.30f)
* Visual Studio 2022 (C#)

# **Cámara**

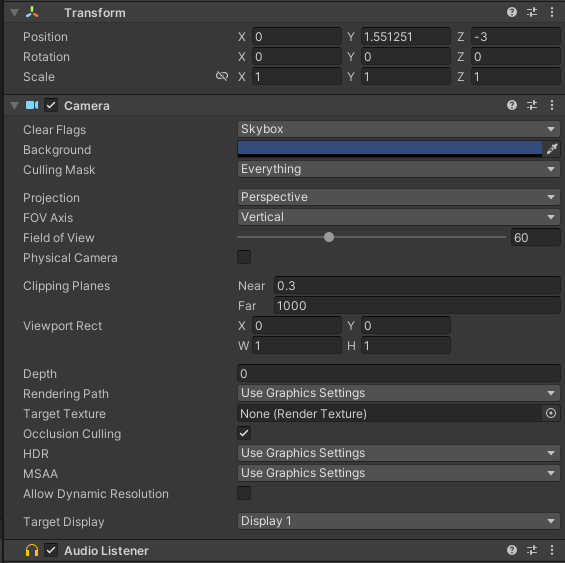
La camara utiliza el componente Cinemachine.

https://docs.unity3d.com/Packages/com.unity.cinemachine@2.3/manual/index.html

## Objeto: Camara

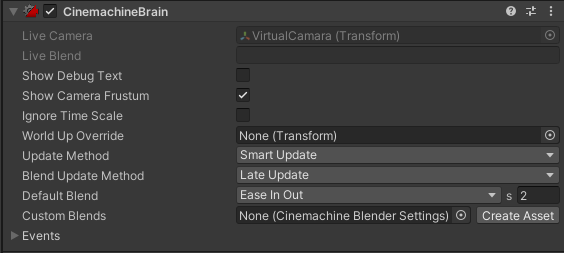
Este objeto funciona como la base de la camara.

### Componentes basicos



### Cinemachine Brain

Cinemachine Brain supervisa todas las cámaras virtuales activas en la escena.

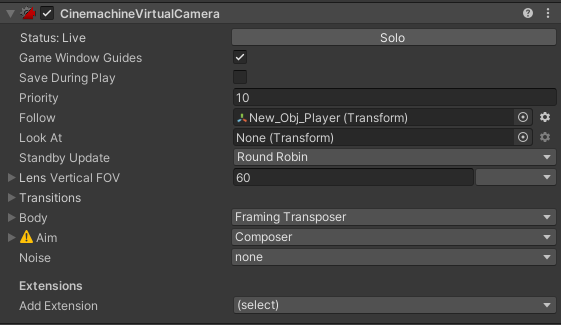


## Objeto: VirtualCamara

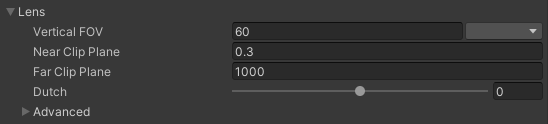
Este objeto funciona como la camara principal que persigue al jugador.

### Componente VirtualCamera

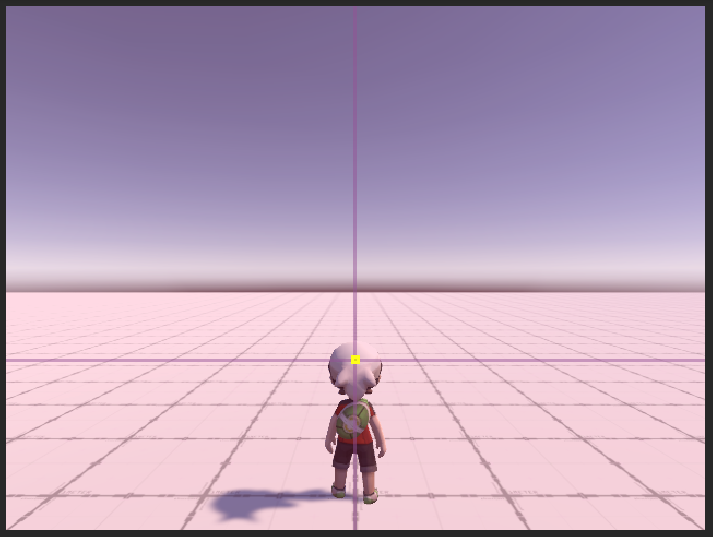
Este componente es la camara en si, aca se encuentran los valores con los que se maneja.

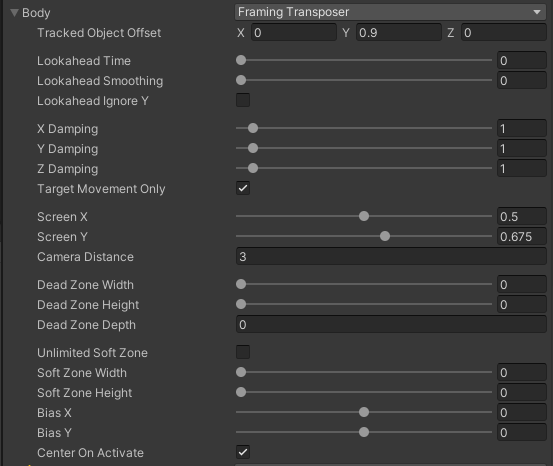


* Follow – Se le asigna el objeto que tiene que seguir la camara (En estecaso el Transform del jugador)



* **Vertical FOV** – La vista de la cámara en grados verticales > 60
* **Near Clip Plane** – Lo cerca que tiene que estar un objeto de la camara para que se vea > 0.3
* **Far Clip Plane** - Lo lejos que tiene que estar un objeto de la camara para que se vea > 1000

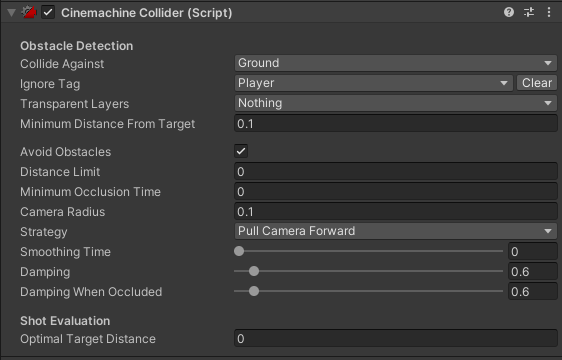




* **Body** – Farming Transposer – La camara se mueve en una relación de espacio de pantalla fija con el objetivo de seguir.
* **Track Object Offset X/Y/Z**– Es el “posicionamiento” de la camara > Y 0.9
* **Screen X/Y** - Es el “posicionamiento” de la camara > Y 0.675
* **Camera Distance** - La distancia la cámara del objetivo a seguir.
* **Soft Zone W/H** – Los rangos en donde la camara no se tiene que mover. W > 0, H > 0

### Cinemachine Collider

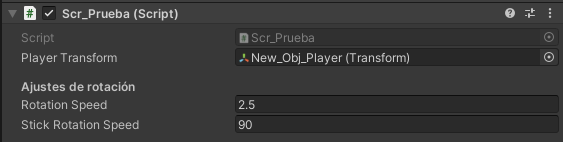
Este componente detecta colisiones para que la camara evite ciertos objetos.



* **Collide Against** – El Layer que tienen que tener los objetos para que la camara los evite. Tag - Ground
* **Ignore Tag** – El Layer que tiene que tener los objetos para que la camar los ignore. Tag - Player
* **Strategy** – La maiobra que va realizar la camara para evitar los objetos. Pull Camera Foward – Empuja la camar enfrente del objeto.

## Script: Scr\_PlayerCamaraControl

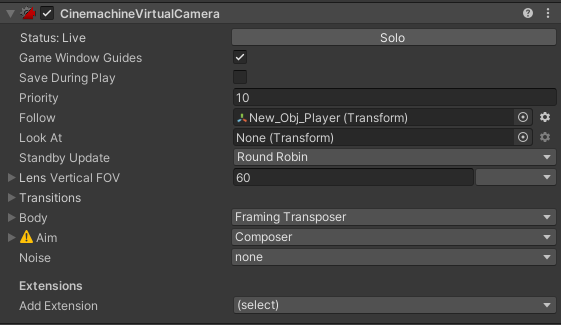
Este Script maneja la posicion de la camara según los inputs del jugador.



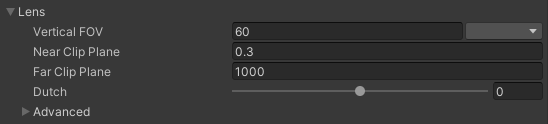
* **Player Transform** – Referencia al componente Transform del jugador.
* **Rotation Speed** – La velocidad de rotacion general.
* **Stick Rotation Speed** – La velocidad de rotacion en base al input de la intensidad del stick.

### Componente VirtualCamera

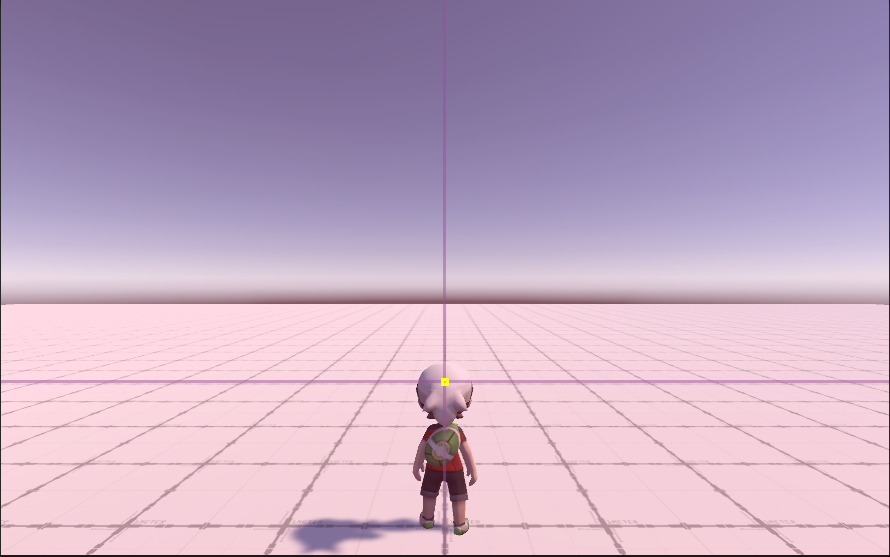
Este componente es la camara en si, aca se encuentran los valores con los que se maneja.

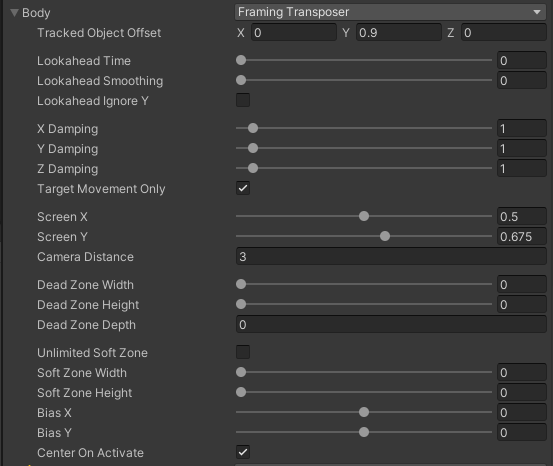


* Follow – Se le asigna el objeto que tiene que seguir la camara (En estecaso el Transform del jugador)



* **Vertical FOV** – La vista de la cámara en grados verticales > 60
* **Near Clip Plane** – Lo cerca que tiene que estar un objeto de la camara para que se vea > 0.3
* **Far Clip Plane** - Lo lejos que tiene que estar un objeto de la camara para que se vea > 1000





* **Body** – Farming Transposer – La camara se mueve en una relación de espacio de pantalla fija con el objetivo de seguir.
* **Track Object Offset X/Y/Z**– Es el “posicionamiento” de la camara > Y 0.9
* **Screen X/Y** - Es el “posicionamiento” de la camara > Y 0.675
* **Camera Distance** - La distancia la cámara del objetivo a seguir.
* **Soft Zone W/H** – Los rangos en donde la camara no se tiene que mover. W – 0, H – 0

# **Player**

El objeto del jugador usa como base el *Demo Character* de la *3D Scene* de el componente *Character Controller Pro*.

https://lightbug14.gitbook.io/ccp

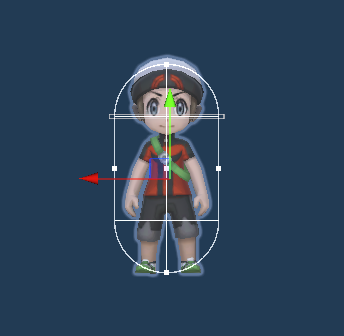
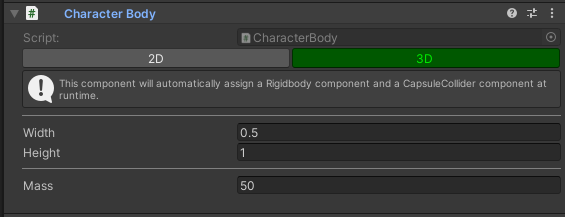
## Objeto Padre: Obj\_Player

Este objeto funciona como la base de la camara.

### Character Body

Este componente representa el cuerpo del personaje.

Se encarga del componente Collider y Rigidbody.



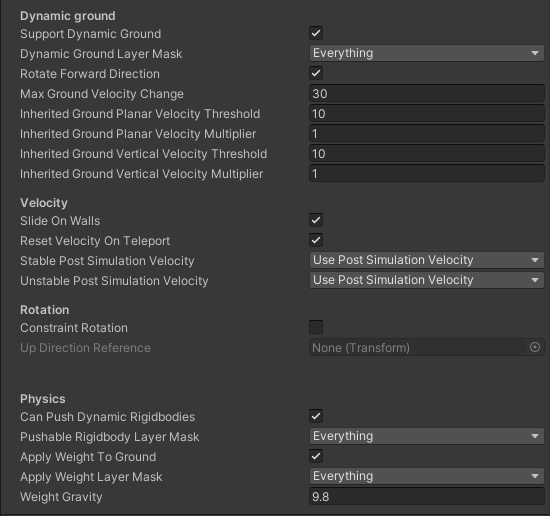
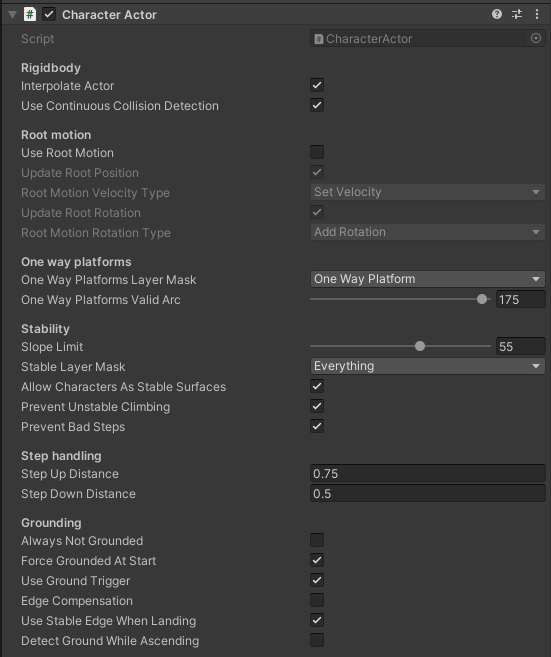
* **Width / Height** – El ancho y largo del collider del jugador.
* **Mass** – El masa del jugador, cuanto pesa.

Bugs:

* Al cambiar Width a 0.5 el perosnaje se agachaba constantemente, lo solucione cambiando en el Blend Tree del Animator los parametros Pos Y de CrouchIdle y CrouchMoving a 2.

### Character Actor

Este componente se encarga de todas las interacciones del personaje con el entorno.

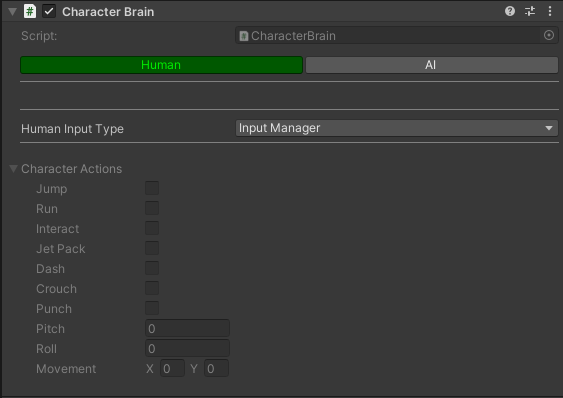


No cambie nada de este componente.

## Objeto Hijo 1: Actions

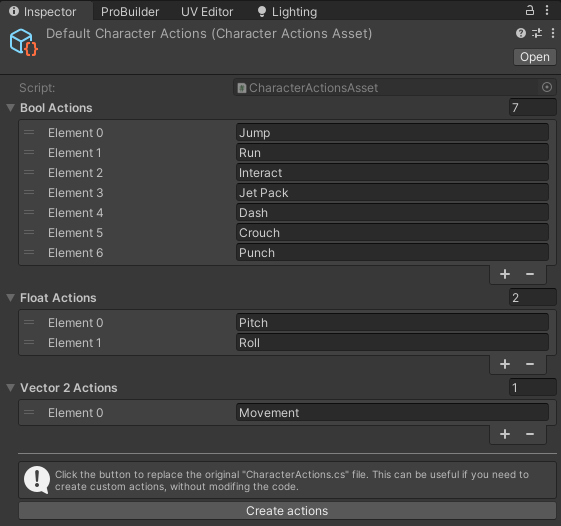
### Character Brain

Este componente responsable de manejar todas las acciones del personaje, independientemente de si provienen del jugador o no.

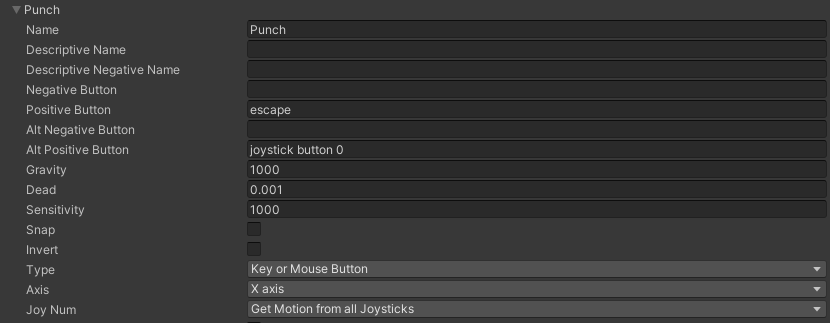


Character Actions

Este componenete contiene todas las acciones que puede realizar el personaje. Dependiendo de la accion el tipo que tiene (Bool, Float, Vector2).



Para que terminen de funcionar, tenemos que setear el input de la accion.



En Project Settings -> Input Manager, creamos un nuevo Input y hacemos que coincida con el nombre de la accion, el componente lo detectara automaticmente.

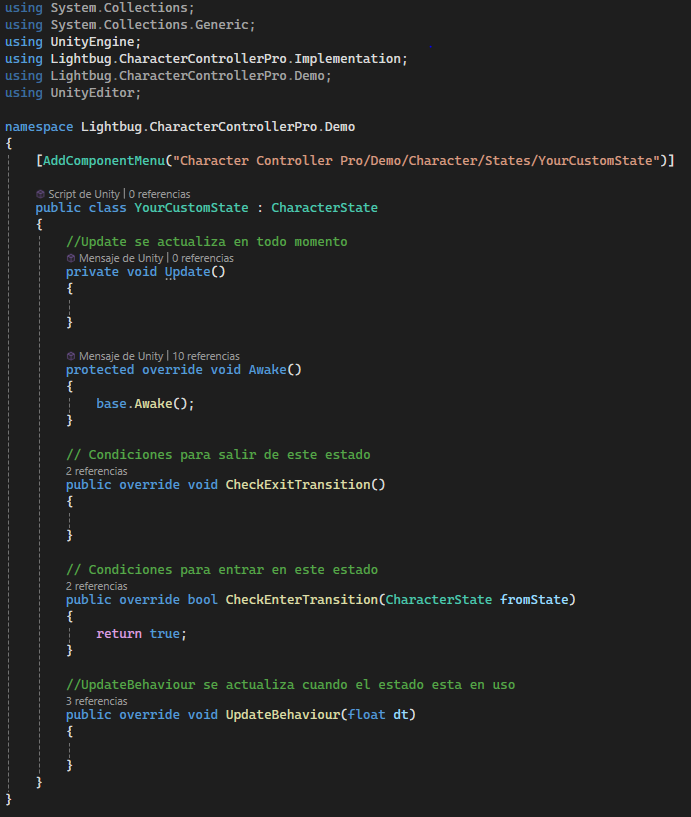
## Objeto Hijo 2: States

Los estados se puede explicar como el conjunto de codigos que forman las acciones.

### Como crear un Estado

Para crear un estado nuevo hay que copiar el Script YourCustomState, cambiarle de nombre (Ej: Scr\_Attack) al archivo y a /States/Attack, public class Scr\_Attack.

**IMPORTANTE:** El Script debe estar localizado en 09-Packages/Character ControllerPro/Demo/Script/State, esto para que se pueda referenciar en otros scripts.

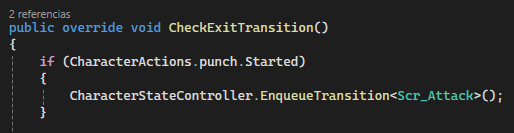


* **Update** – Se actualiza mientras el objeto este en escena
* **UpdateBehaviour** – Se actualiza mientras el estado este en ejecucion.
* **CheckExitTransition** – Esta parte de codigo se evalua primero, dentro estan las condiciones para salir de este estado.
* **CheckEnterTransition** - Esta parte de codigo se evalua segunda, dentro estan las condiciones para entrar a este estado.

Hay que tener en cuenta que el proceso sera el siguiente. Primero hay que programa una condicion de salida desde otro estado para entrar a este, luego una condicion de entrada para el estado actual y por ultimo la condicion de salida.

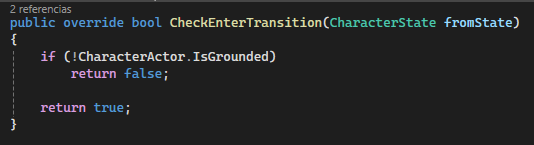
1 – Se sale del estado NormalMovement y se va a Attack.

En este caso si se presiona un Input (esto hace uso de las acciones en Default Character Actions)



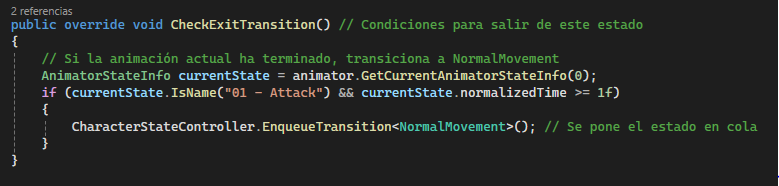
2 – Se entra a el estado Attack

En este caso solo si el personaje esta en tierra. (IsGrounded)

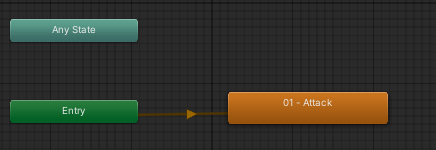


3 - Condicion para salir del estado Attack

En este caso al terminar la animacion.

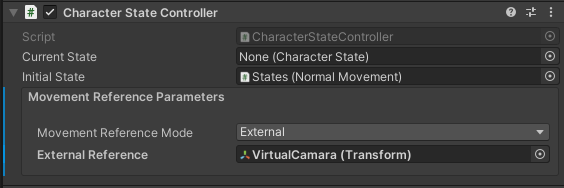


Luego hay que crear un AnimatorController correspondiente a las acciones del estado.



### Character State Controller

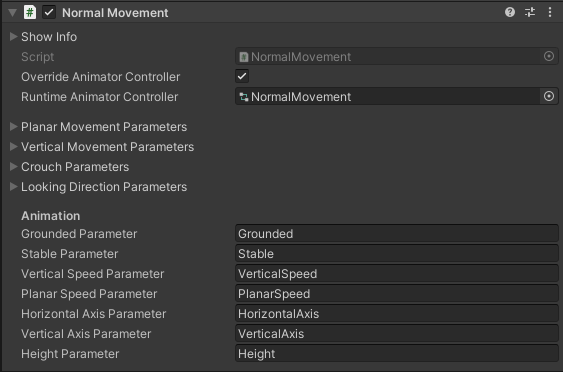
Este componente controla el estado actual del personaje.



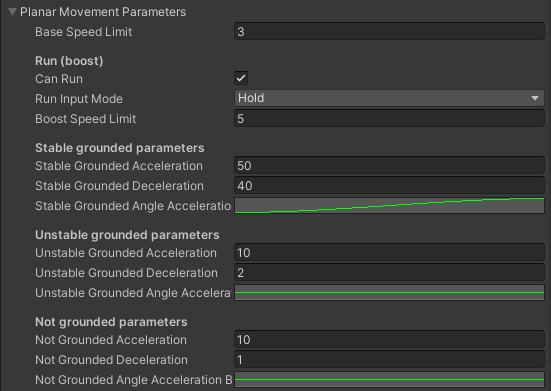
* **Current State** – El estado actual en el que se encuetnra el Jugador.
* **Initial State** – El estado inicial en el que se ve a encotrar el Jugador.
* **Movment Reference Mode** – En referencia a que se ve a mover el jugador. – External (En este caso en rerencia a donde mira la VirtualCamara).

### Normal Movment

Este script maneja el movimiento basico del personaje. Movimiento en superficie, dos velocidades (correr y caminar), agacharse, salto y movimiento en el aire.

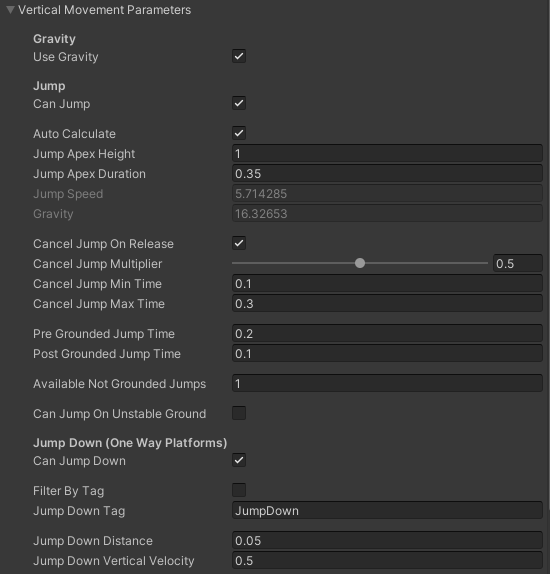


Planar Movment Parameters



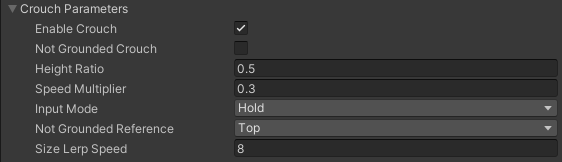
* **Base Speed Limit** - La velocidad del personaje al caminar > 3
* **Boost Speed Limit** - La velocidad del personaje al correr > 5

Vertical Movment Parameters



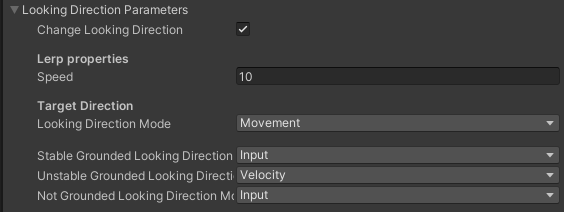
* **Jump Apex Height** - Altura del Salto > 1
* **Jump Apex Duration** - La velocidad del Salto > 0.35
* **Avaliable not Grounde Jump** - Los saltos extra > 1

Crouch Parameters



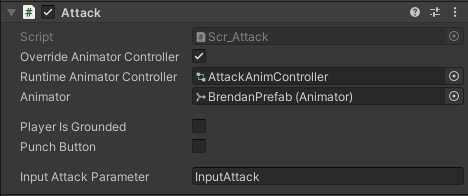
* **Hight Ratio** - La altura del collider al agacharse en realcion a la actual > 0.5
* **Speed Multiplier** - La velocidad del personaje al agacharse > 1

Looking Direction Parameters

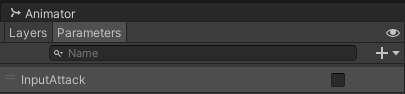


### Attack

Este script maneja los ataques del jugador.



* **Player Is Grounded** – Verifica si el jugador esta en una superficie.
* **Punch Button** – Verifica si el Input eesta presionado.



* **Input Attack Parameter** – Hacer referencia al parametro bool InputAttack del Animator. Detecta si en input esta siendo presionado.