**Documento de Programa**

**Tools**

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

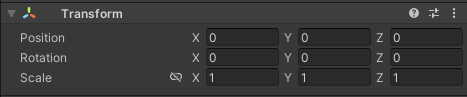
# **Objetos**

## 1. Obj\_Player

Es el objeto principal del objeto, contiene todos los hijos y componentes esenciales para el personaje.

### 1.1 - Transform

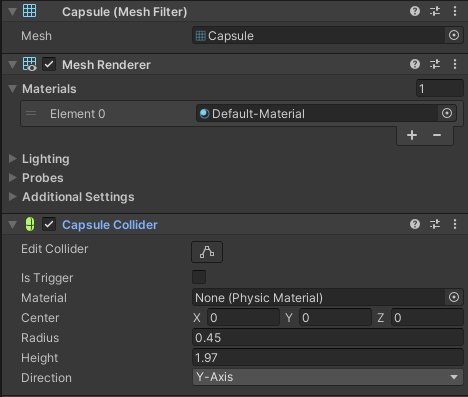
Determina la posición, rotación y escala del objeto.



### 1.2 – Capsule Collider y Renderer

Determina el espacio que ocupa el personaje. Solo aspectos visuales son en 2D, pero la forma del jugador es en 3D.

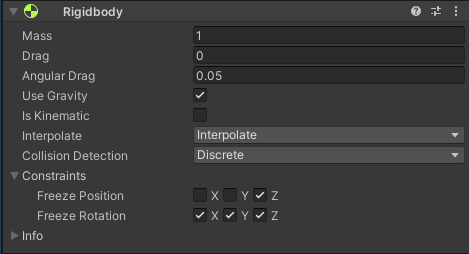
Usamos *Radius* y *Height* para mover los límites.





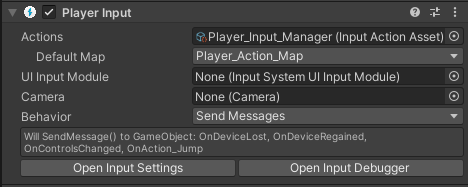
### 1.3 - RigidBody

Proporciona la física a un objeto.

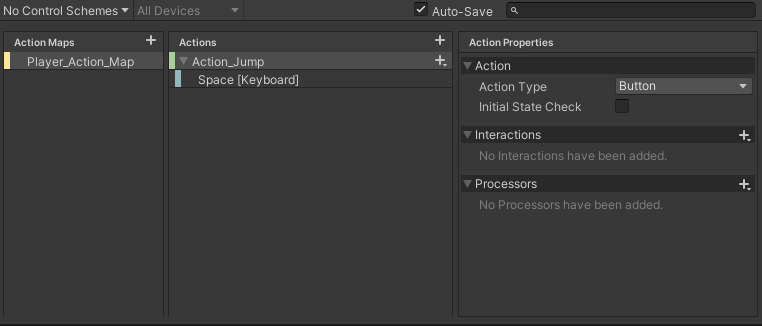


### 1.4 – Player Input

Este componente es el encargado de manejar el Input System de Unity.



*Default Map* se refiere al mapa de acciones al que accedemos por defecto. El mapa de acciones contiene dentro las acciones y inputs que se realizan.



En esta ventana es donde se controlan y asginan los Inputs.

*Action Maps* se refiere al conjunto de accions que tiene un tipo de control, por ejemplo, se puede tener un tipo para las acciones del personaje y otra para cuando se navegan los menus.

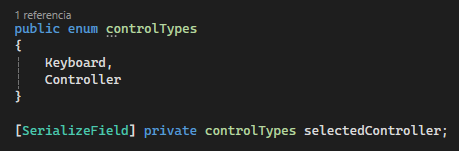
*Actions* es cada accion individual que exista, cada una tiene un *Action Type*:

* *Button* - Para entradas ON/OFF (Ej: saltar con un botón).
* *Value* - Para valores continuos (Ej: dirección de un joystick).
* *Pass-Through* - Para flujos de datos sin filtrado.

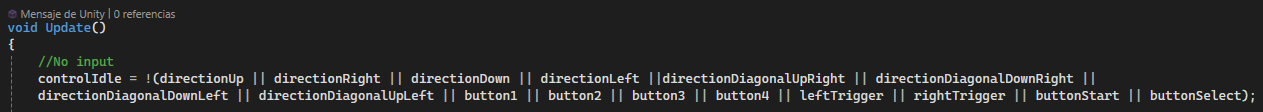
Debajo de las acciones se define el *Binding*, la tecla/boton la cual ejecuta la accion.

### 1.5 – Scr\_Player\_01\_Controls

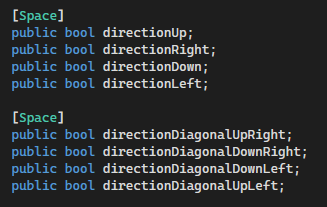
Este script se encarga de manejar el tipo de control que usa el jugador y sus inputs.

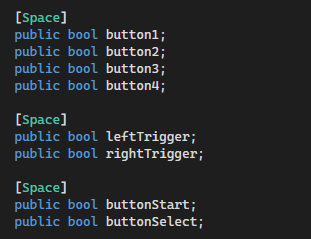
Hay dos tipos de controles, usando el teclado (Keyboard) o usando un cotnrol tradicional (Controller).

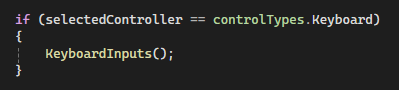


La bool *controlIdle* se activa cuando ningun otro boton esta activado.

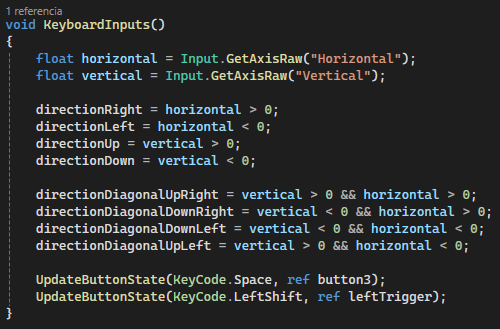


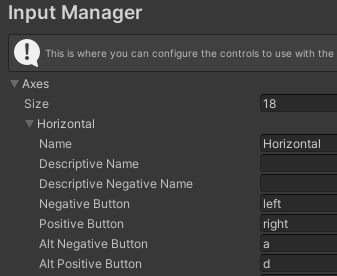
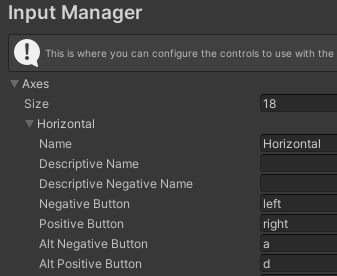
Las direcciones se componen de los cuatro ejes mas cuatro diagonales.





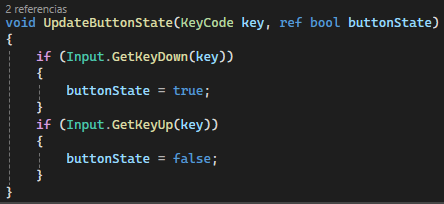
La funcion *KeyboardInputs* almacena los inputs que se pueden generar por teclado.





La manera que tenemos de caluclarla es usando los Axis Horizontales y Verticales. *Input.GetAxisRaw* lee la entrada en el input manager.

En caso Horizontal A es -1 y D es 1. En caso Vertical S es -1 y W es 1.



*UpdateButtonState* se encarga de activar la bool correspondiente dependiendo del boton que le demos.

### 1.6 – Scr\_Player\_02\_States

Este script controla el estado en el que esta el personaje.



* **playerWalkSpeed** – La velocidad al caminar.
* **playerRunSpeed** – La velocidad al correr.
* **playerJumpForce** – La fuerza de salto.

### 1.7 – Scr\_Player\_02\_Stats

Este script contiene en un mismo lugar el valor numerico de las estadisticas del personaje.



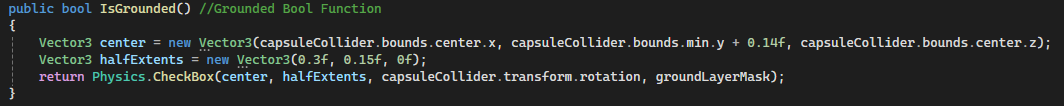
* **playerWalkSpeed** – La velocidad al caminar.
* **playerRunSpeed** – La velocidad al correr.
* **playerJumpForce** – La fuerza de salto.

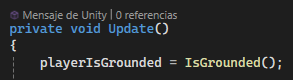
### 1.8 – Scr\_Player\_03\_Movement

Este script maneja el movimiento basico del jugador, movimiento horizontal y salto.

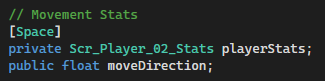
Variables



* **playerCanMove** – Verdadero si el personaje puede moverse.
* **playerIsGrounded** – Verdadero si el personaje esta en una superficie.

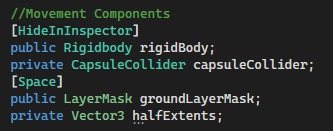


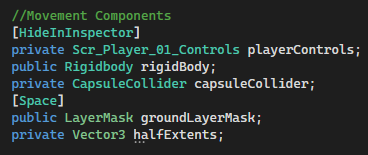
*IsGrounded()* es una funcion que contiene el codigo para saber si el jugador esta en colision con una superficie. Esta funcion regresa verdadero o falso, dependiendo de eso en el metodo *Update()* se prende la bool *playerIsGrounded*.



Las estadisticas de movimiento del jugador.

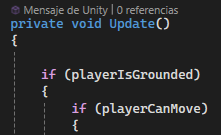
* **playerStast** – El script en donde se almacenan los valores de las estadisticas del jugador.
* **moveDirection** – Le da valor 1 si se presiona la derecha y -1 si se presiona la Izquierda.



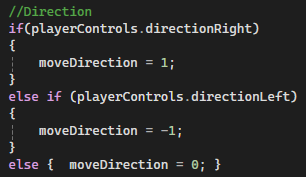


* **playerControls** – El script que contiene los inputs del jugador.
* **rigidBody** – Referencia al componente rigidBody del objeto.
* **capsuleCollider** – Referencia al componente capsuleCollider del objeto.
* **groundLayerMask** – Determina las Layers que son concideradas superficie.
* **halfExtents** – Determina la distanacia para la comprobacion de superficie.

Metodo Update()

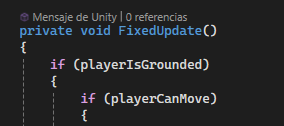


Si el jugador esta en una superifice (*playerIsGrounded*) y puede moverse (*playerCanMove*).

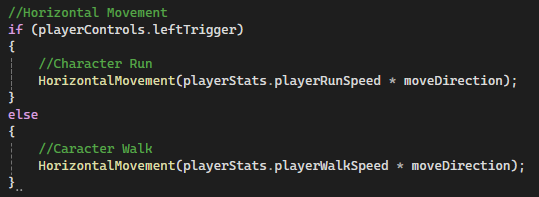


*moveDirection* se calcula usando *playerControls*, su valor depende del input que se ingrese.

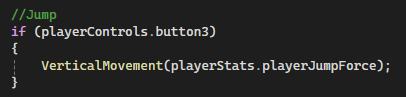
Metodo FixedUpdate()



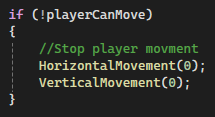
Si el jugador esta en una superifice (*playerIsGrounded*) y puede moverse (*playerCanMove*).



La funcion *HorizontalMovment* se encarga del movimiento horizotnal dependiendo del valor y la direccion que le pasemos, este depende si *leftTrigger* es precionado.

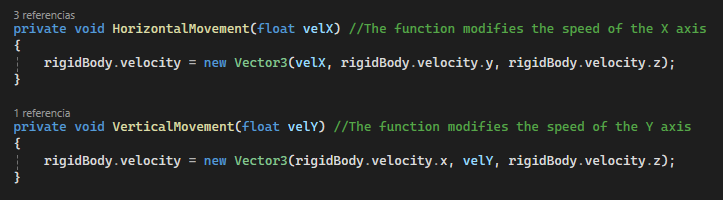


La funcion *VerticalMovment* se encarga del salto dependiendo del valor que le pasemos.



Si el jugador no puede moverse, se les pasa 0 a las funciones de movimiento para que el personaje pare.

Functions



Las funciones *HorizotnalMovement(velX)* y *VerticalMovment(VelY)*, son funciones que pasan un dato al eje X e Y respectvamente, causando que se mueva el personaje.