**Documento de Programa**

**General**

A

**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.

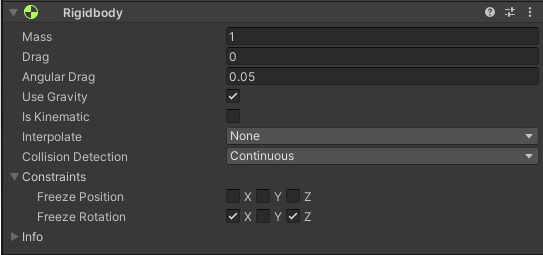
### 1.1 - Transform

Determina la posición, rotación y escala del objeto.



### 1.2 - RigidBody

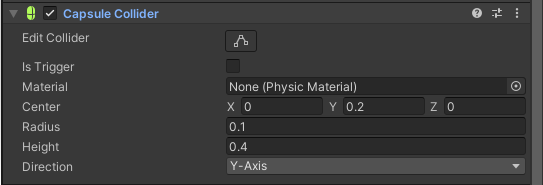
Proporciona la física a un objeto.

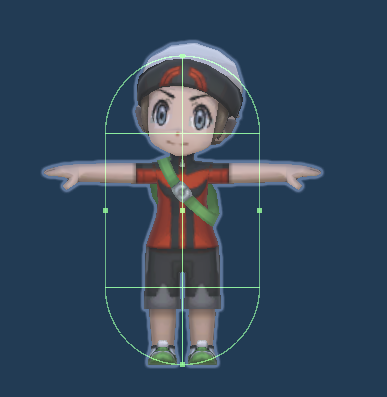


### 1.3 – Capsule Collider

Determina el espacio que ocupa el personaje.

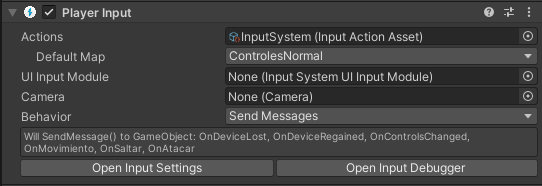
Usamos Radio y Altura para mover los límites.



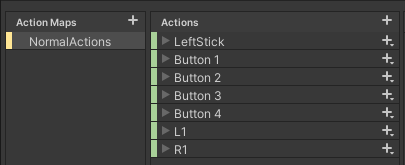


### 1.4 – Player Input

Determina el control y tipo de controles que tiene el personaje.



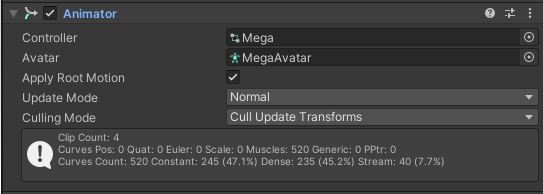
Controles Normal



* Left Stick – Stick Izquierdo
* Button 1 – Botonf Inferior
* Button 2 – Boton Izquierdo
* Button 3 – Boton Derecho
* Button 4 – Boton Superior
* L1 – Gatillo Izquierdo
* R1 – Gatillo Derecho

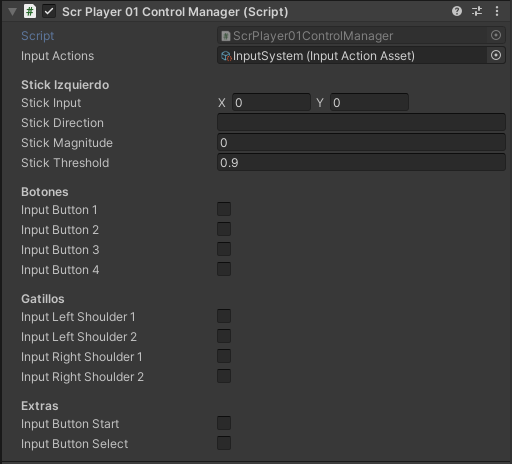
### 1.5 - Animator

Este componente gestiona las animaciones del objeto.

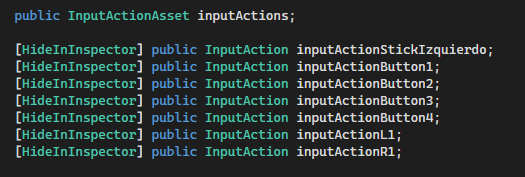


### 1.6 – ScrPlayer01ControlManager

Controla y detecta los Input del jugador.

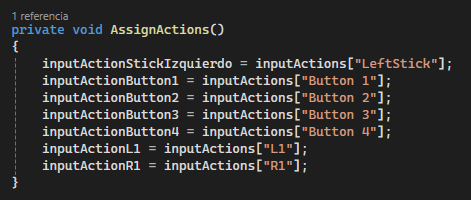
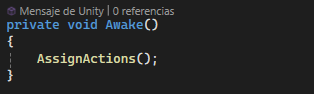


#### Codigo: Input Actions



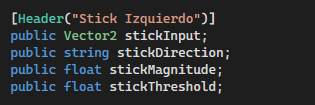
InputActionsAsset hace referencia al componente InputSystem el cual maneja el control del jugador.

Las variables InputAction hace referencia a cada accion que contenga el InputSystem.



Asignamos cada variable InputAction en Awake con la funcion AssignActions().

#### Codigo: Stick

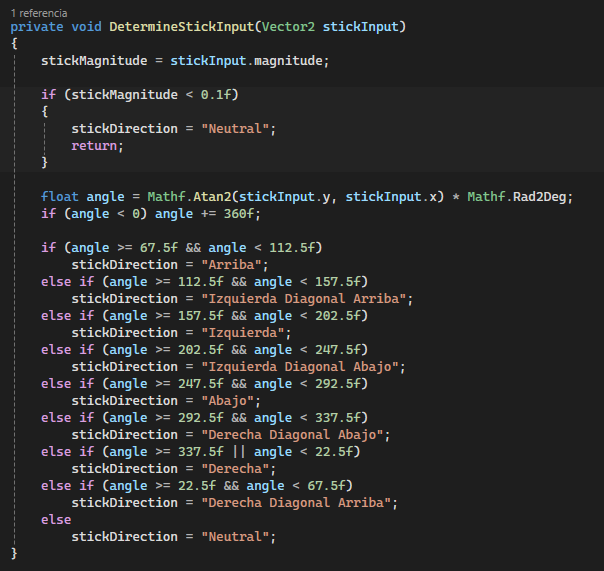
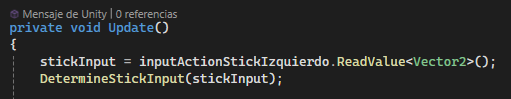


stickInput, un Vector2 que representa la dirección en la que el se mueve el stick izquierdo.

stickDirection es un string que describe la dirección del stick (arriba, abajo, izquierda, etc.)

stickMagnitude representa la fuerza con la que el jugador empuja el stick.

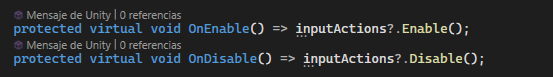
stickThreshold es un umbral para filtrar entradas del Stick.



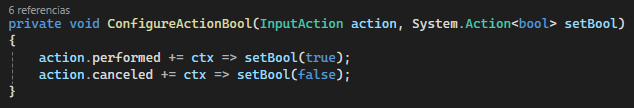
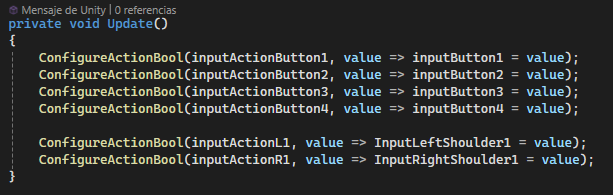
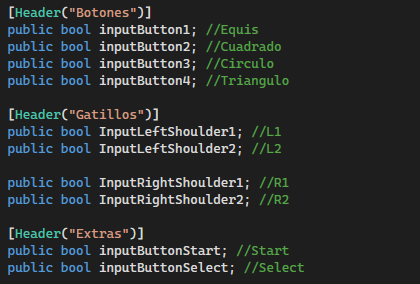
Este es el método encargado de procesar la entrada del stick izquierdo.

Si el stick no se está moviendo, la dirección se establece como "Neutral". Si el stick tiene movimiento, se calcula el ángulo entre las coordenadas X e Y del stick. Dependiendo del ángulo, se asigna una dirección, como "Arriba", "Izquierda", "Derecha", etc.

#### Codigo: Botones



Utilizamos OnEnable y OnDisable permite esuchar las entradas de los inputs.



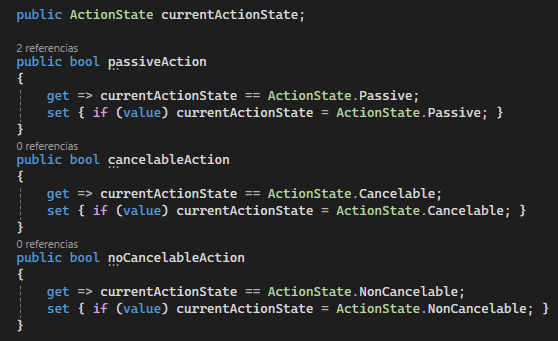
La funcion ConfigureActionBool es la que se encarga de prender la bool corresponidente cuando se presiona un input para poder verlo en el editor.

### 1.7 – ScrPlayer02StateManager

Controla y detecta los estados en el que seencuentra el Jugador.



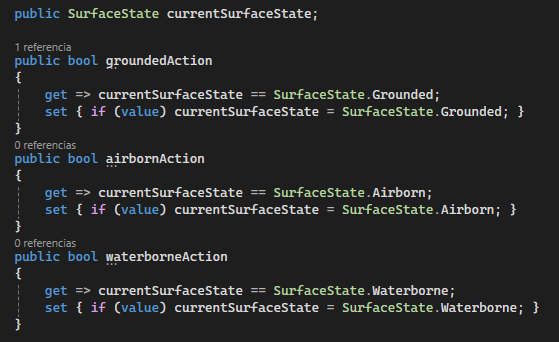
#### Codigo: Current Action State



Hace referencia a la disponibilidad del personaje.

* + passiveAction – Se encuentra en este estado al no presionar ningún input, puede ser cancelado por cualquier acción. (Idle, Fall).
  + cancelableAction – Se encuentra en este estado al realizar acciones que a la vez pueden ser canceladas por el resto. (Walk, Jump, Swim)
  + noCancelableAction – Se encuentra en este estado al realizar acciones que no pueden ser canceladas. (Punch, Spin, Others)

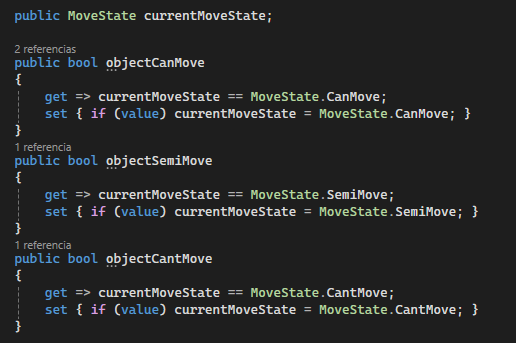
#### Codigo: Current Surface State



Hace referencia a la superficie sobre la que se encuentra el personaje.

* + groundedAction – Si se encuentra sobre una superficie.
  + airbornAction – Si se encuentra en el aire.
  + waterborneAction – Si se encuentra sobre un liquido.

#### Codigo: Current Move State

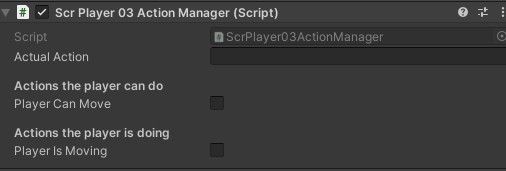


Hace referencia a la capacida de movimiento del personaje.

* + objectCanMove – Si el personaje puede moverse.
  + objectSemiMove – Si el personaje cuenta con una capacidad limitada de movimiento.
  + objectCantMove *-*  Si el personaje no puede moverse.

### 1.8 – ScrPlayer03ActionManager

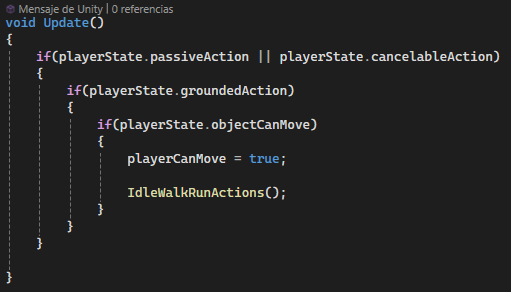
Controla y determina las acciones del jugador.



* Actual Action – Es el string principal, determia la accion actual que se esta realizando.
* Actions the player can do – Son bools que al prenderse determina que acciones puede realizar el jugador.
* Action the player is doing – Son bools que al prenderse determina las acciones que esta haciendo el jugador.

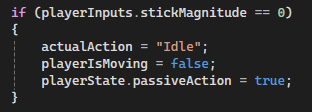
#### Codigo: Action Organization

Las acciones se organizan dependiendo de los tres estados en ScrPlayer02StateManager y los Inputs de ScrPlayer01ControlManager.



Los estados se organizan de la sigueinte manera:

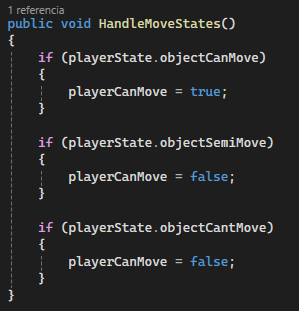
Action State > Surface State > Move State.



Despues, organizada en una funcion, dependiendo del input se determina:

1. actualAction – El nombre de la accion actual que se ejecute.
2. Scrip Bools – Activar o desactivar las bools caracteristicas que definan la accion.
3. playerState – Modificar Action State dependiendo del tipo de accion que es.

#### Codigo: HandleMoveState()



Se encarga de controlar la bool canMove dependiendo del estado de playerState Current Move State.

#### Codigo: IdleWalkRunActions()



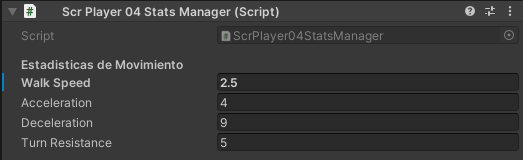
Se encarga de controlar el movimiento terrestre con el stick.

Dependiendo de la magnitud del stick junto a stickThreshold se determina si el personaje esta en Idle, normalWalk o fastWalk.

normalWalk y fastWalk son cancelableActions.

### 1.9 – ScrPlayer04StatsManager

Se encarga de almacenar el valor de las estadisticas del objeto del jugador.



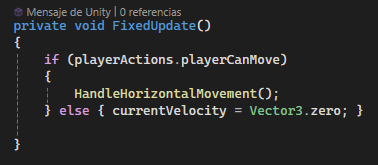
* Walk Speed – Velocidad maxima de movimiento en superifice.
* Acceleration – Velocidad con la que el jugador acelera.
* Dececelaration – Velocidad con la que el jugador desacelera.
* Turn Resistance – La resistencia que genera el personaje al girar.

### 1.10 – ScrPlayer05MovementManager

Controla el movimiento del jugador.

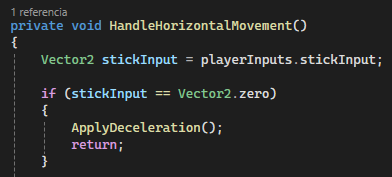


#### Codigo: Acceder a Movimientos



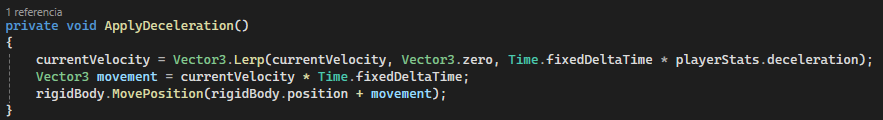
Para acceder a HandleHorizontalMovment(), que almacena el movimiento del stick en tierra, el jugador tiene que estar en playerActions.playerCanMove.

#### Codigo: Stick Movment



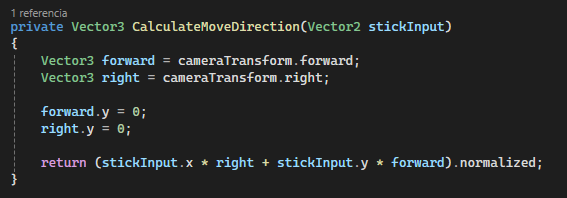
Vector2 stickInput obtiene la dirección en la que el jugador mueve el stick, usando el script ScrPlayer01ControlManager. Si el stick no se mueve se llama a ApplyDeceleration() para que el personaje desacelere.

Acceleracion



Similar al movimiento, cuando no hay input, se interpola la velocidad actual, currentVelocity, hacia Vector3.zero usando Vector3.Lerp. La rapidez de desaceleración depende del valor de playerStats.deceleration.

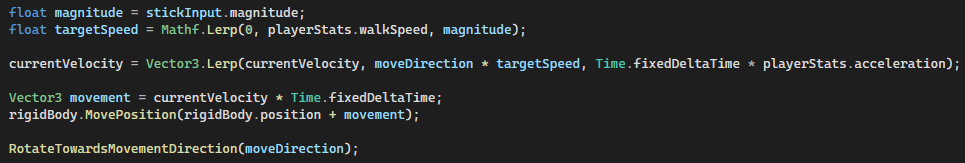
Direccion



moveDirection calcula la dirección hacia la que el personaje debe moverse. Se basa en la cámara y en la entrada del stick.

Esta función está definida en CalculateMoveDirection(). Combina el input del stick con las direcciones de la cámara para obtener la dirección final en la que el personaje se moverá.

Velocidad y Magnitud



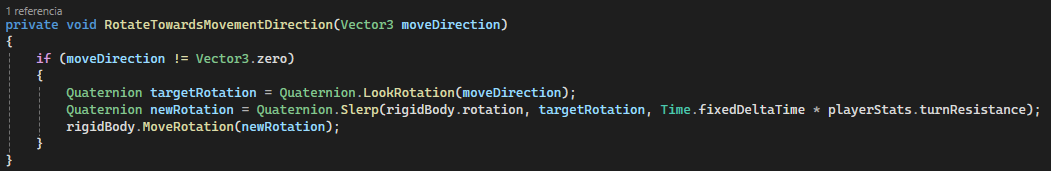
targetSpeed usa Mathf.Lerp y la walkSpeed para que cuanto más fuerte se empuja el stick, mayor será la velocidad objetivo. La magnitude del stickInput determina qué tan fuerte se empuja el sticks.

currentVelocity Vector3.Lerp se usa para suavizar el cambio entre la velocidad actual (currentVelocity) y la velocidad objetivo (moveDirection \* targetSpeed).

playerStats.acceleration controla qué tan rápido el personaje alcanza su velocidad máxima.

Se calcula el desplazamiento del personaje multiplicando la velocidad por el tiempo para asegurar un movimiento suave. Luego, aplica ese desplazamiento a la posición del Rigidbody con rigidBody.MovePosition().

Rotacion



La rotacion se llama después de calcular el movimiento, en RotateTowardsMovementDirection().

Verifica si moveDirection no es Vector3.zero, si no lo es procede a girar el personaje. Si es cero, significa que no hay movimiento y no se debe rotar.

Usa Quaternion.Slerp para suavizar el giro del personaje, interpolando entre la rotación actual del Rigidbody y la rotación targetRotation.

El valor de playerStats.turnResistance controla la velocidad con la que el personaje rota, lo que ayuda a evitar giros bruscos o deslizamientos.

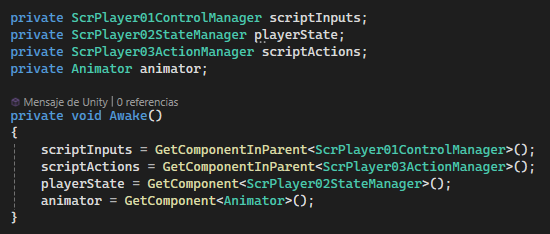
Finalmente, se aplica la nueva rotación al Rigidbody con rigidBody.MoveRotation().

### 1.11 - ScrPlayer06AnimationManager

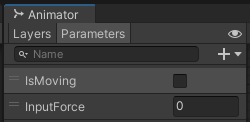
Se usa para manejar las animaciones, la bools y los eventos.

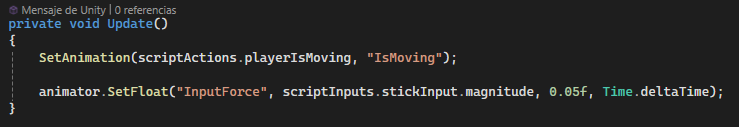


#### Code: Componentes

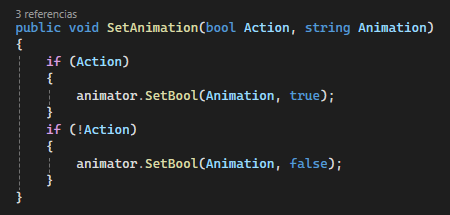


#### Code: Parametros de Animaciones





SetAnimation funciona para asignar parametros que son Bools, mientras que para los float usamos un acercamiento directo de animator.SetFloat.



La funcion SetAnimation tiene dos variables, Action que es la condicion y Animation que es el nombre de la bool de animacion que se prende si la condicion se cumple.

## 2. TargetCamara

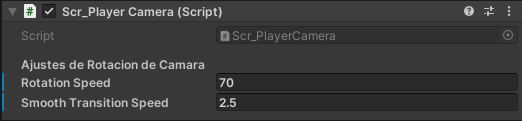
Es el objeto al que enfoca la camara y contiene parte de los componentes para controlarla.

### 2.1 - Transform

Determina la posición, rotación y escala del objeto.



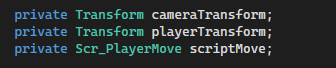
### 2.2 – ScrPlayer04CameraManager



rotationSpeed: Velocidad de la rotación de la cámara.

smoothTransitionSpeed: Velocidad de rotación.

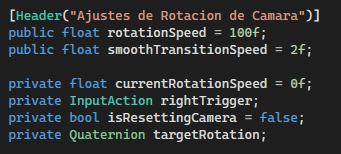
#### Codigo: Componentes



cameraTransform: Referencia al transform de la cámara virtual.

playerTransform: Referencia al transform del jugador.

scriptMove: Referencia al script que controla el movimiento del jugador (Scr\_PlayerMove).



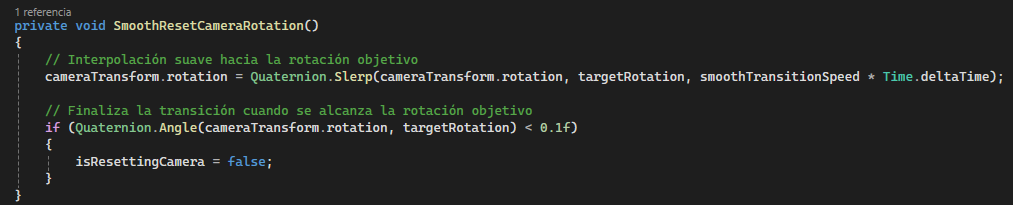
currentRotationSpeed: Velocidad actual de rotación que puede variar suavemente.

rightTrigger: Acción asociada al gatillo derecho para resetear la cámara.

isResettingCamera: Bool que indica si la cámara está en proceso de reseteo.

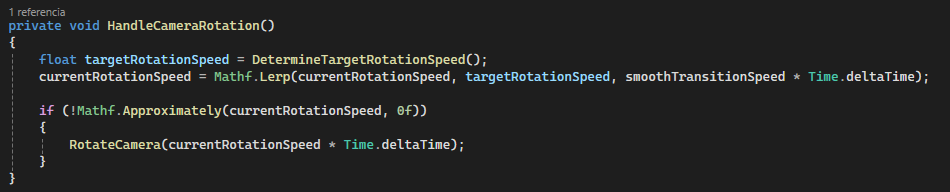
targetRotation: Rotación objetivo de la cámara durante el reseteo.

#### Codigo: Metodos



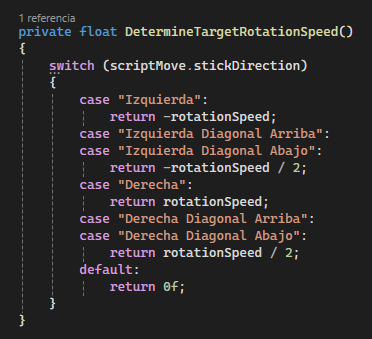
Método SmoothResetCameraRotation().

Realiza una interpolación suave de la rotación actual de la cámara hacia la targetRotation. Cuando la cámara casi alcanza la rotación objetivo se detiene el proceso de reseteo.



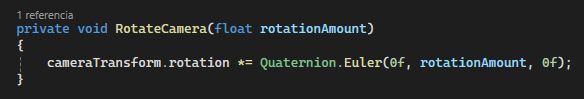
Método HandleCameraRotation().

Maneja la rotación de la cámara basándose en la dirección del stick. Calcula la velocidad de rotación objetivo y realiza una interpolación suave hacia esta velocidad. Si la velocidad de rotación actual no es cero, rota la cámara en consecuencia.



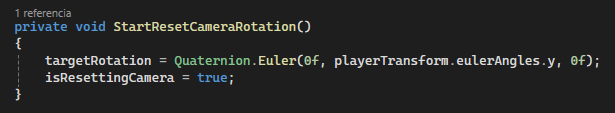
Método DetermineTargetRotationSpeed().

Este método determina la velocidad de rotación de la cámara basándose en la dirección del stick (por ejemplo, rotación completa a la izquierda, rotación diagonal más lenta, etc.).



Método RotateCamera().

Rota la cámara aplicando una rotación en el eje Y, multiplicando el quaternion actual de la cámara por uno nuevo calculado en función de rotationAmount.



Método StartResetCameraRotation().

Se activa cuando el jugador presiona el gatillo derecho. Establece la rotación objetivo de la cámara alineada con la rotación del jugador en el eje Y y activa el proceso de reseteo (isResettingCamera).