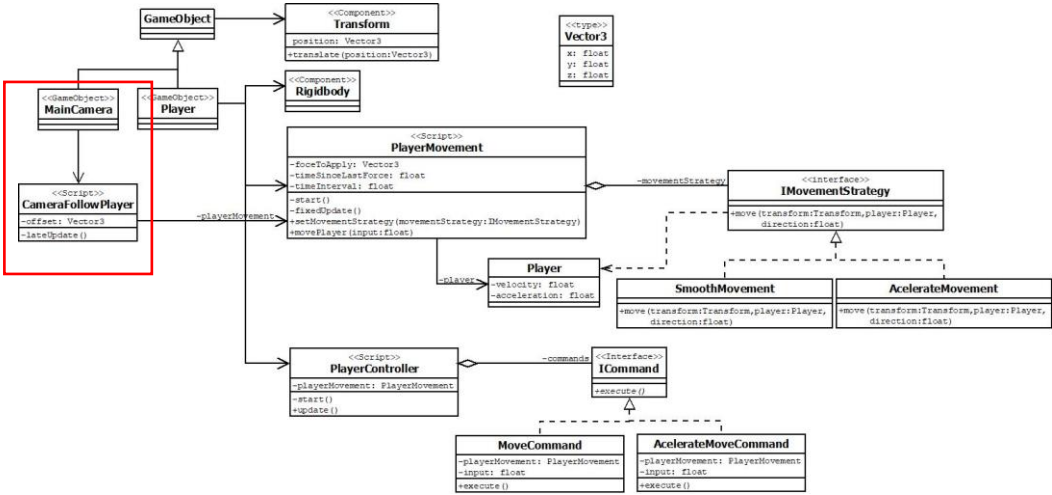


Historia de Usuario: Seguimiento del player con la cámara	
Identificador: UH05	Usuario: Jugador
Prioridad en negocio: -	Riesgo en desarrollo: -
Puntos estimados: -	Iteración asignada: -
Programador Responsable: Ariel Vega	
<p>Descripción</p> <p>Como jugador quiero la cámara siga los movimientos de player desde atrás para visualizar todo lo que tiene en el frente.</p>	
<p>Observaciones</p> <p>Datos de entrada:</p> <ul style="list-style-type: none"> Posición del player Un vector offset que indica la distancia que debe estar la cámara respecto del player <p>Proceso:</p> <p style="text-align: center;">Posición de la cámara = posición del player + offset</p> <p>Salida: offset</p> <p>Diagrama de clases</p>  <pre> classDiagram class GameObject { <<Component>> } class Transform { <<Component>> position: Vector3 +translate(position: Vector3) } class Rigidbody { <<Component>> } class Vector3 { <<type>> x: float y: float z: float } class PlayerMovement { <<Script>> -forceToApply: Vector3 -timeToStartForce: float -timeInterval: float +start() +fixedUpdate() +setMovementStrategy(movementStrategy: IMovementStrategy) +movePlayer(input: float) } class Player { -velocity: float -acceleration: float } class CameraFollowPlayer { <<Script>> -offset: Vector3 +lateUpdate() } class PlayerController { <<Script>> -playerMovement: PlayerMovement +start() +update() } class IMovementStrategy { <<interface>> +move(transform: Transform, player: Player, direction: float) } class SmoothMovement { +move(transform: Transform, player: Player, direction: float) } class AccelerateMovement { +move(transform: Transform, player: Player, direction: float) } class ICommand { <<interface>> +execute() } class MoveCommand { -playerMovement: PlayerMovement -input: float +execute() } class AccelerateMoveCommand { -playerMovement: PlayerMovement -input: float +execute() } GameObject -- > Transform GameObject -- > Rigidbody PlayerMovement -- > Rigidbody PlayerMovement -- > IMovementStrategy PlayerMovement -- > ICommand PlayerMovement -- > SmoothMovement PlayerMovement -- > AccelerateMovement PlayerMovement -- > Player CameraFollowPlayer -- > Transform CameraFollowPlayer -- > Player PlayerController -- > Player PlayerController -- > PlayerMovement PlayerController -- > ICommand PlayerController -- > MoveCommand PlayerController -- > AccelerateMoveCommand IMovementStrategy < .. SmoothMovement IMovementStrategy < .. AccelerateMovement ICommand < .. MoveCommand ICommand < .. AccelerateMoveCommand </pre>	
<p>Criterios de aceptación</p> <ul style="list-style-type: none"> La cámara sigue al Player a una distancia configurada 	

Actividad 01: Crear el Script CameraFollowPlayer e inicializar offset

```
1 using UnityEngine;
2
3 public class CameraFollowPlayer : MonoBehaviour
4 {
5     private Vector3 offSet;
6     void Start()
7     {
8         offSet = new Vector3(0, 1, -5);
9     }
10
11 }
```

Actividad 02: Modificar la posición de la cámara usando el método `LateUpdate()`. Con el objetivo de garantizar una correcta actualización de la cámara y evitar efectos tipo serrucho, es conveniente definir la posición de la cámara en `LateUpdate()`.

```
1 using UnityEngine;
2
3 public class CameraFollowPlayer : MonoBehaviour
4 {
5     private Vector3 offSet;
6     private PlayerMovement playerMovement;
7     void Start()
8     {
9         offSet = new Vector3(0, 1, -5);
10        playerMovement = FindObjectOfType<PlayerMovement>();
11    }
12
13    private void LateUpdate()
14    {
15        gameObject.transform.position = playerMovement.transform.position + offSet;
16    }
17 }
```

Hay varios aspectos a destacar. En primer lugar, se agrega un nuevo atributo que de tipo el Script `PlayerMovement`. Para instanciarlo en el método `Start()` existen diferentes mecanismos. En esta cátedra se incentiva utilizar los mecanismos que usen programación. En este caso mediante el método `FindObjectOfType` podemos realizar la búsqueda de un Componente que esté presente en un `gameObject` dentro de la jerarquía de objetos. Debido a que busca dentro de la jerarquía de `GameObjects`, se debe tener en cuenta que debe existir un único `gameObject` que te tenga este componente, sino se podrían obtener resultados inesperados o errores.

Ahora pruebe el resultado; no olvide previamente asignar este Script al `gameObject` `MainCamera`.

Conclusión

Se recomienda estudiar y aprender los diferentes mecanismos para acceder a componentes de otros `gameObjects` diferentes al que pertenece el Script donde esté parado. Puede resultar muy cómodo y sencillo usar la asignación en tiempo de edición en el Inspector de Unity, pero esto a la larga puede provocar errores de pérdida de enlaces.