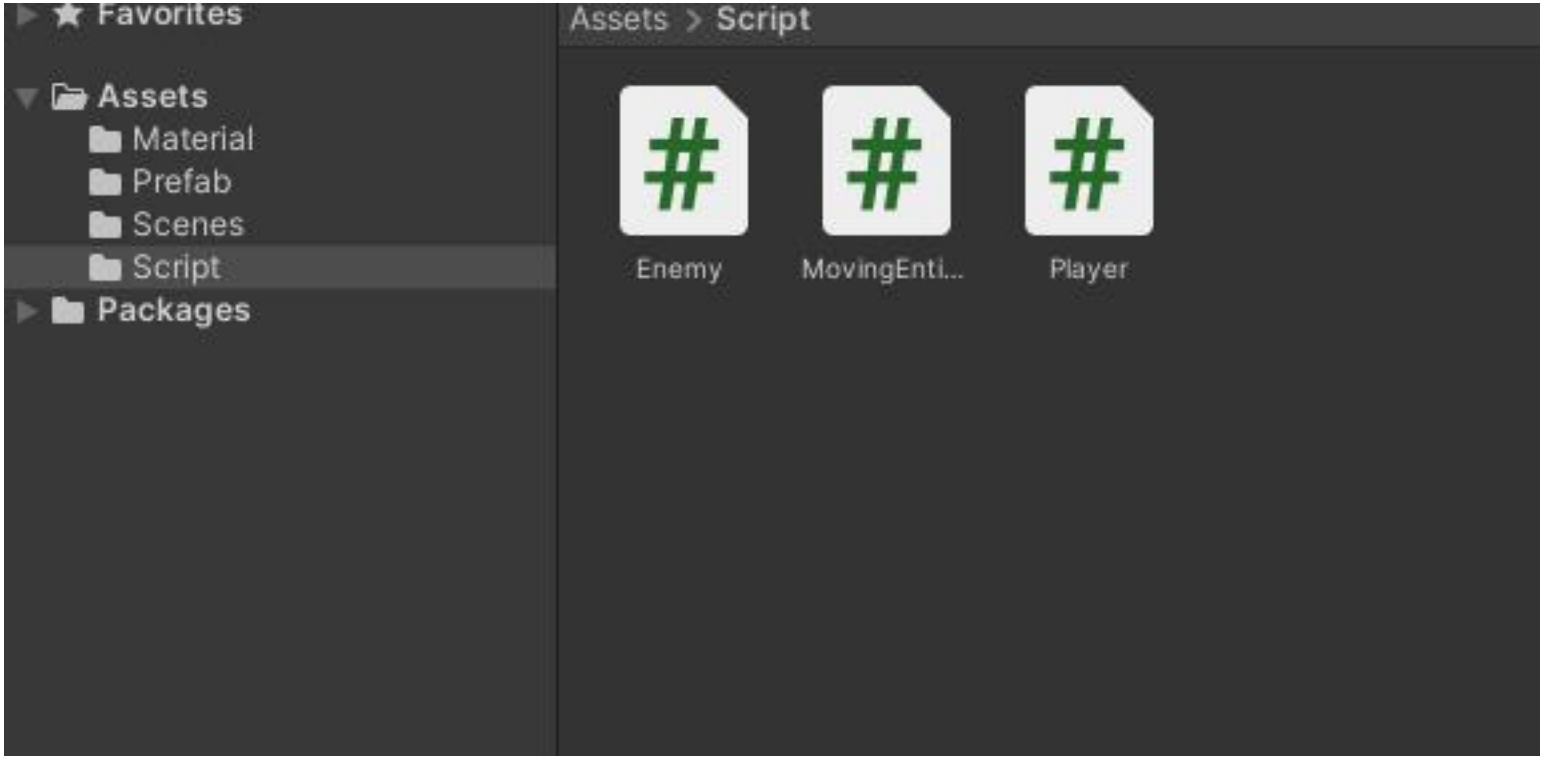
IA_2_1



Creamos tres códigos para separar el enemigo, el Player y los movimientos.

IA_2_1 (MOVINGENTITY)

En *MovingEntity* trasladamos una serie de variables que anteriormente estaban asignadas al código del enemigo.

```
public class MovingEntity : MonoBehaviour
{
    public float movementSpeed = 1f;
    public float rotationSpeed = 1f;

protected void MoveTowards (Vector3 direction)
```

MoveTowards -Orientaba al objeto en las direcciones en las que se va a mover. De esta manera, tanto el Player como el enemigo heredan de esta clase y puedan acceder a esta función

IA_2_1 (*ENEMY)*

En *ENEMY* en lugar de heredar de *MonoBehaviour* lo hace de *MovingEntity*.

```
ublic class Enemy : MovingEntity
{

    Vector3 targetPosition;
    Vector3 towardsTarget;

    float wanderRadius = 5f;

    void RecalculateTargetPosition ()
    {
        targetPosition = transform.position + Random.insideUnitSphere *
wanderRadius;
        targetPosition.y = 0;
```

IA_2_1 (*ENEMY)*

En Update utiliza la función *MoveTowards* del Padre (*MovingEntity*).

```
void Update()
   {
      towardsTarget = targetPosition - transform.position;
      MoveTowards(towardsTarget.normalized);

      if (towardsTarget.magnitude < 0.25f)
         RecalculateTargetPosition();




      Debug.DrawLine (transform.position, targetPosition, Color.green);
```

IA_2_1 (*PLAYER)*

```
public class Player : MovingEntity
{

    // Update is called once per frame
    void Update()
    {
        Vector3 desiredDirection = Vector3.zero;
        if (Input.GetKey(KeyCode.A))
            desiredDirection += Vector3.left;
        if (Input.GetKey(KeyCode.D))
            desiredDirection += Vector3.right;
        if (Input.GetKey(KeyCode.W))
            desiredDirection += Vector3.forward;
        if (Input.GetKey(KeyCode.S))
            desiredDirection += Vector3.back;

        if (desiredDirection != Vector3.zero)
            MoveTowards(desiredDirection.normalized);
```

IA_2_1 (*PLAYER)*

```
public class Player : MovingEntity
{

    // Update is called once per frame
    void Update()
    {
```

Vector3 desiredDirection = Vector3.zero;- Inicia en cero e irá sumando las direcciones en las que el jugador quiere moverse según las teclas que pulse.

```
        if (Input.GetKey(KeyCode.A))
            desiredDirection += Vector3.left;
        if (Input.GetKey(KeyCode.D))
            desiredDirection += Vector3.right;
        if (Input.GetKey(KeyCode.W))
            desiredDirection += Vector3.forward;
        if (Input.GetKey(KeyCode.S))
            desiredDirection += Vector3.back;

        if (desiredDirection != Vector3.zero)
            MoveTowards(desiredDirection.normalized);
```

IA_2_1 (*PLAYER)*

```
if (Input.GetKey(KeyCode.A))
        desiredDirection += Vector3.left;
```

Mediante la librería Input se puede acceder a distintas funciones que nos permiten saber el estado de las teclas, botones o dispositivos.

GetKey(KeyCode.A)) D, W, S se volverán verdadero si se encuentran pulsadas.

Cada tecla sumará a *desiredDirection* la dirección que le responden.
A- Vector a la izquierda.
D-Vector a la derecha.
W- Vector adelante.
S- Vector atrás.

Si la dirección resultante no es *zero*, llamaremos a **MoveTowards** del padre (**MovingEntity)** con la dirección de movimiento normalizada.
Reciclamos el código y el Player se mueve de manera similar al enemigo, solo que sin autonomía y controlado por el teclado.