



# Desarrollo de Videojuegos

Pedro Yuri Marquez Solis

```

1  using UnityEngine;
2  using System.Collections;
3
4  public class Ejemplo : MonoBehaviour
5  {
6      public GameObject[] jugadores;
7      public int numeroDeJugadores;
8
9      public void AgregarJugadores ()
10     {
11         jugadores = new GameObject[2];
12         jugadores[0] = new GameObject("Seneca1");
13         jugadores[1] = new GameObject("Seneca2");
14     }
15
16     protected void Start ()
17     {
18         AgregarJugadores ();
19         numeroDeJugadores = jugadores.Length;
20         for(int i = 0; i < numeroDeJugadores; i++)
21         {
22             Debug.Log("El jugador " + i + " se llama " + players[i].name);
23         }
24     }
25 }

```

Palabras  
reservadas

Estructuras  
de datos

Estructuras  
de  
Control

Variables

Pedro Yuri Marquez Solis

# Orden de ejecución de eventos en Unity

La clase MonoBehaviour tiene varios eventos propios que se lanzan de forma automática. El orden de ejecución de los eventos es el siguiente:

1. Awake()
2. OnEnable()
3. Start()
4. Update()
5. LateUpdate()
6. OnGUI()
7. OnApplicationQuit()
8. OnDisable()
9. OnDestroy()

**Pedro Yuri Marquez Solis**

# Diferencias entre Update y FixedUpdate

**Update :** depende de los FPS a los que pueda rendir la computadora, no se tiene control.

- ejecuciones por segundo de Update es variable
- Para corregir : multiplicar por `timeDeltaTime()`
- En sobrecargas el update se llama menos veces.
- Update se utiliza para la parte lógica

**FixedUpdate:** Se puede ejecutar mas de una vez en cada frame, No depende de los FPS

- ejecuciones por segundo de FixedUpdate es fija
- Podemos modificar el Time Step en Project Settings > Time, por defecto es 0.02 segundos (20 milisegundos).
- Se utiliza para la parte física, movimientos y animaciones, es decir acciones que deben cambiar de manera regular en el tiempo. Para Rigidbody

Pedro Yuri Marquez Solis

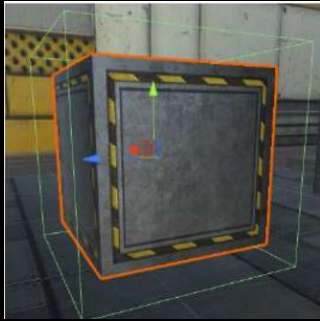
# RigidBody —

- Le permite al GameObject estar afectado por variables físicas:
  - Mover los objetos.
  - Aplicar fuerzas
  - Detectar colisiones.
  - Provee de las funciones para modificar la velocidad, la masa, la gravedad, torque, fricción, colisión.
  - Es aconsejable agregar consecuentemente un objeto **colider**.

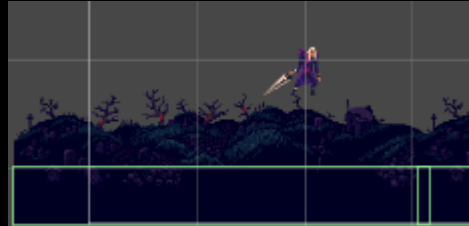
**Pedro Yuri Marquez Solis**

# Collider -- BoxCollider – SphereCollider-capsule-mesh

- Los colliders nos permiten que el objeto cree una colisión con otros objetos cuando estos trabajan con físicas.



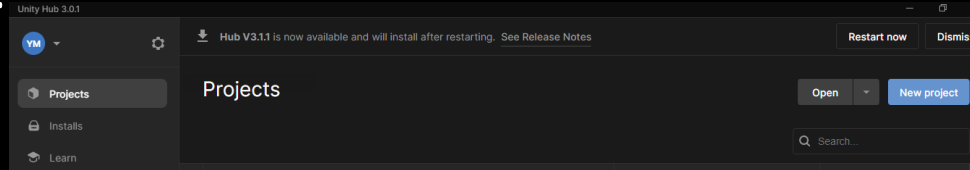
representada en color verde, en la imagen se ha exagerado el tamaño para que se diferencie bien del modelo. Normalmente cuando creas un componente collider este se adapta automáticamente al modelo.



**Pedro Yuri Marquez Solis**

# GhoticVania – Juego 2d Plataformas

- Inicia Unity y crea un proyecto en 2d, que tenga por nombre tu apellido.



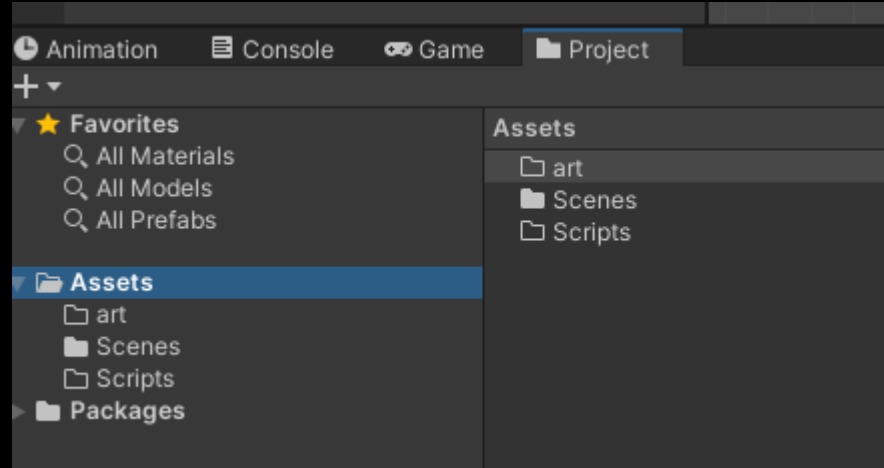
- Selecciona el tipo 2D, esta



Pedro Yuri Marquez Solis

# Estructurar básicamente el proyecto

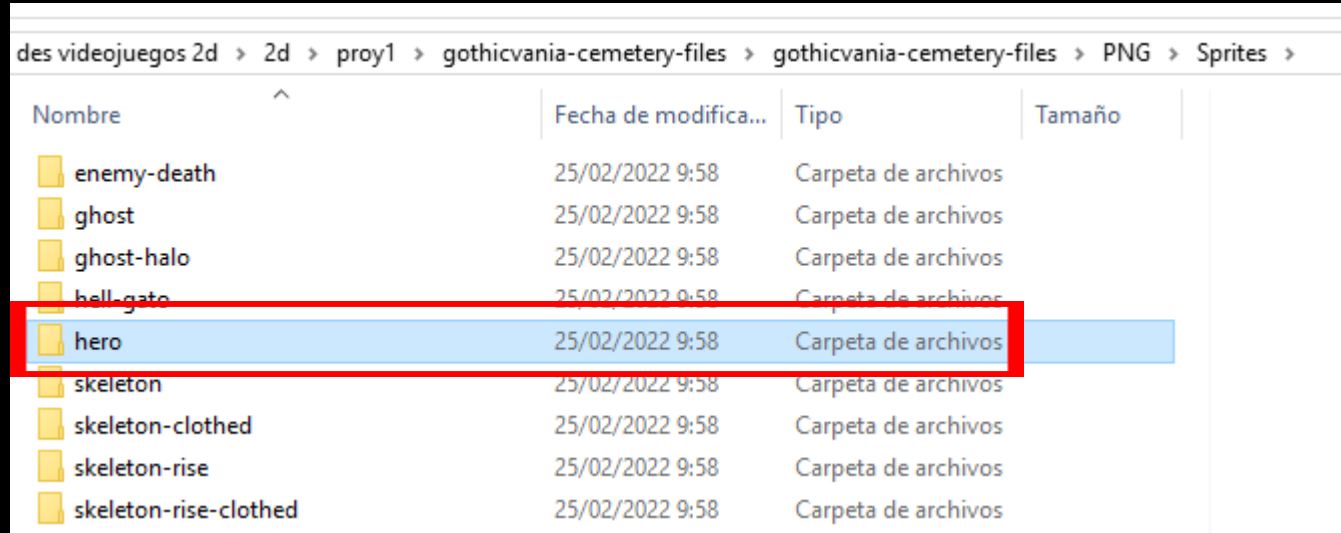
- Agregar las carpetas
  - Art
  - Scripts



**Pedro Yuri Marquez Solis**



# Copiar los archivos de Hero a Arts

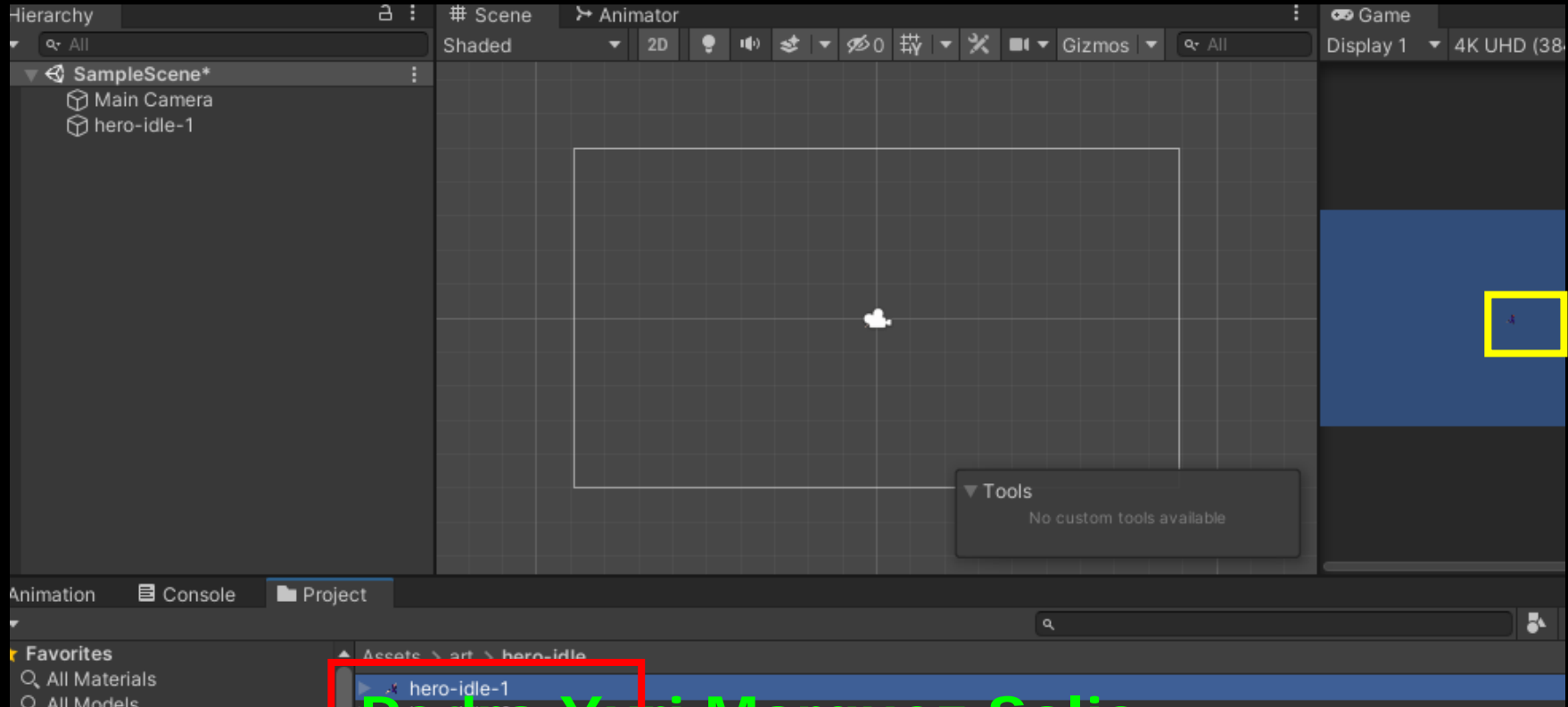


des videojuegos 2d > 2d > proy1 > gothicvania-cemetery-files > gothicvania-cemetery-files > PNG > Sprites >

Nombre	Fecha de modifica...	Tipo	Tamaño
enemy-death	25/02/2022 9:58	Carpeta de archivos	
ghost	25/02/2022 9:58	Carpeta de archivos	
ghost-halo	25/02/2022 9:58	Carpeta de archivos	
hell-gato	25/02/2022 9:58	Carpeta de archivos	
hero	25/02/2022 9:58	Carpeta de archivos	
skeleton	25/02/2022 9:58	Carpeta de archivos	
skeleton-clothed	25/02/2022 9:58	Carpeta de archivos	
skeleton-rise	25/02/2022 9:58	Carpeta de archivos	
skeleton-rise-clothed	25/02/2022 9:58	Carpeta de archivos	

**Pedro Yuri Marquez Solis**

# Arrastrar el personaje



Pedro Yuri Marquez Solis

# Configurar la cámara

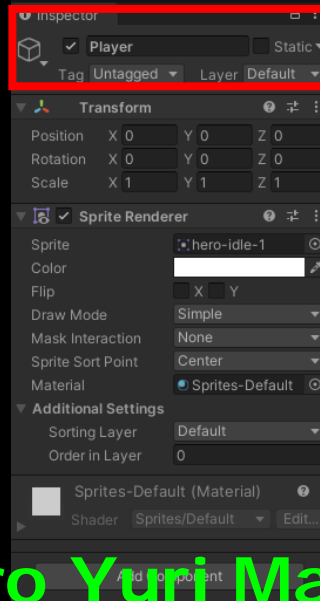
- 1: Ortográfica
- Acercar con Size en : 1.5



Pedro Yuri Marquez Solis

# Renombrar el game Object

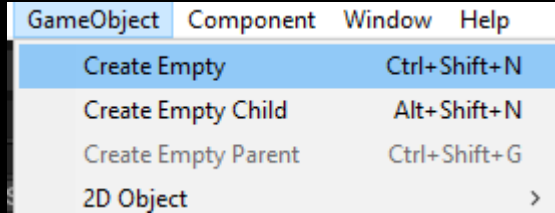
- Proporcionar el name: Player
- Cambiar el background de la camara



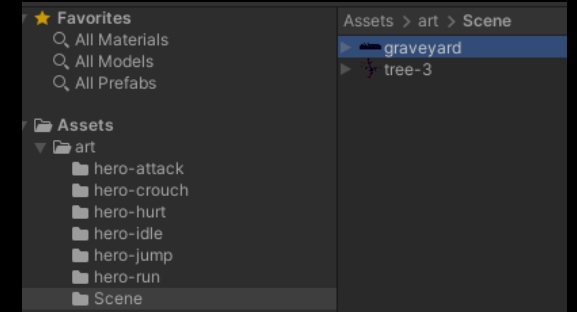
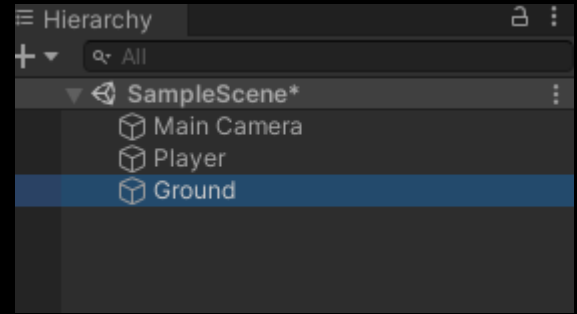
Pedro Yuri Marquez Solis

# Crear el piso

- Emplear un game Object Empty y renombrarlo.



- Crear un folder Scene y copiar hacia allí :  
*tree-3 y graveyard.*

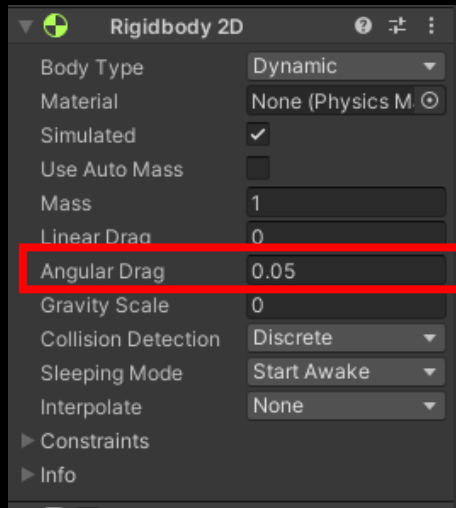


- Asociar a ground, arrastrándolo sobre el game Object



**Pedro Yuri Marquez Solis**

# Agregar un Rigidbody2d a Player



Luego de implementar el script se volverá a  
Poner la gravedad en 1

**Pedro Yuri Marquez Solis**

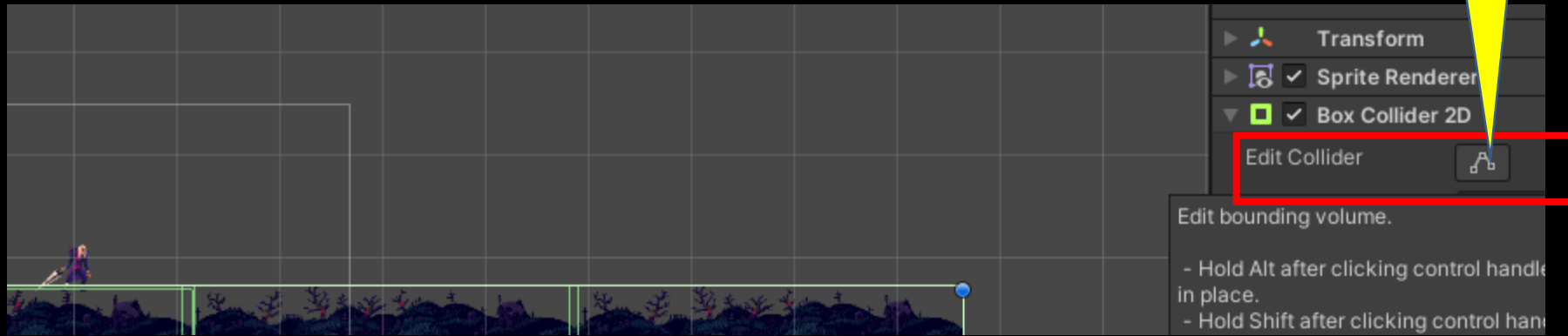
# Darle movimiento a Player

```
public class PlayerController : MonoBehaviour
{
    // Start is called before the first frame update
    public float speed;
    float velx, vely;
    Rigidbody2D rb;
    void Start()
    {
        rb = gameObject.GetComponent<Rigidbody2D>();
    }
    // Update is called once per frame
    void Update()
    {
        velx = Input.GetAxisRaw("Horizontal");
        vely = rb.velocity.y;
        rb.velocity = new Vector2(velx * speed, vely);
    }
}
```

**Pedro Yuri Marquez Solis**

# Agregar un BoxCollider2D al Piso

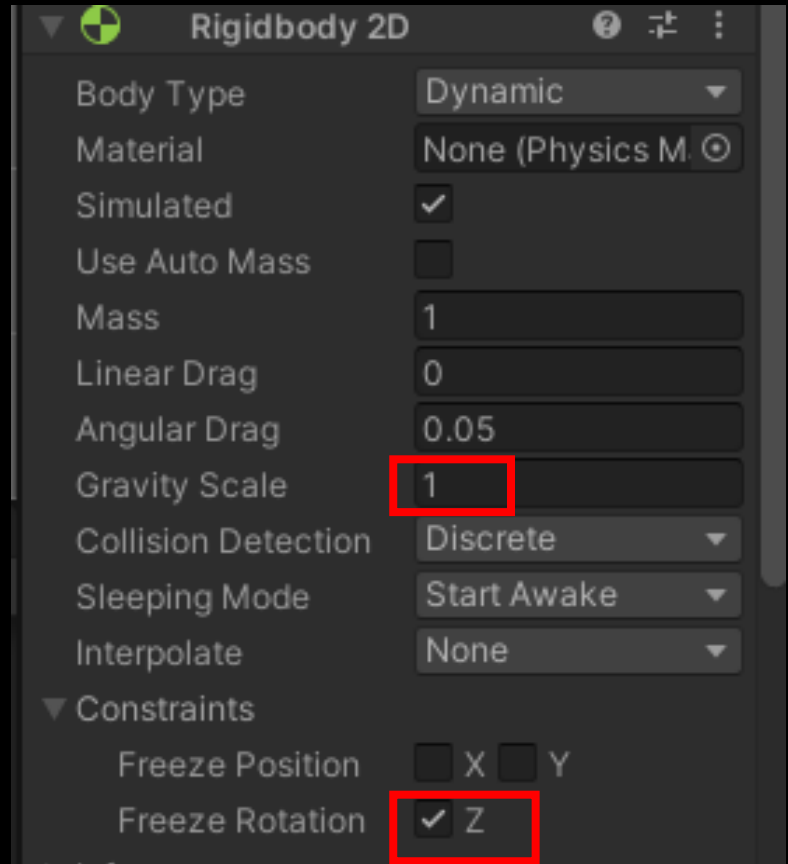
- Seleccionar los 3 pisos y agregarles el BoxCollider2d.
- Modificar los puntos de contacto



**Pedro Yuri Marquez Solis**



# Modificar el Rigidbody2d



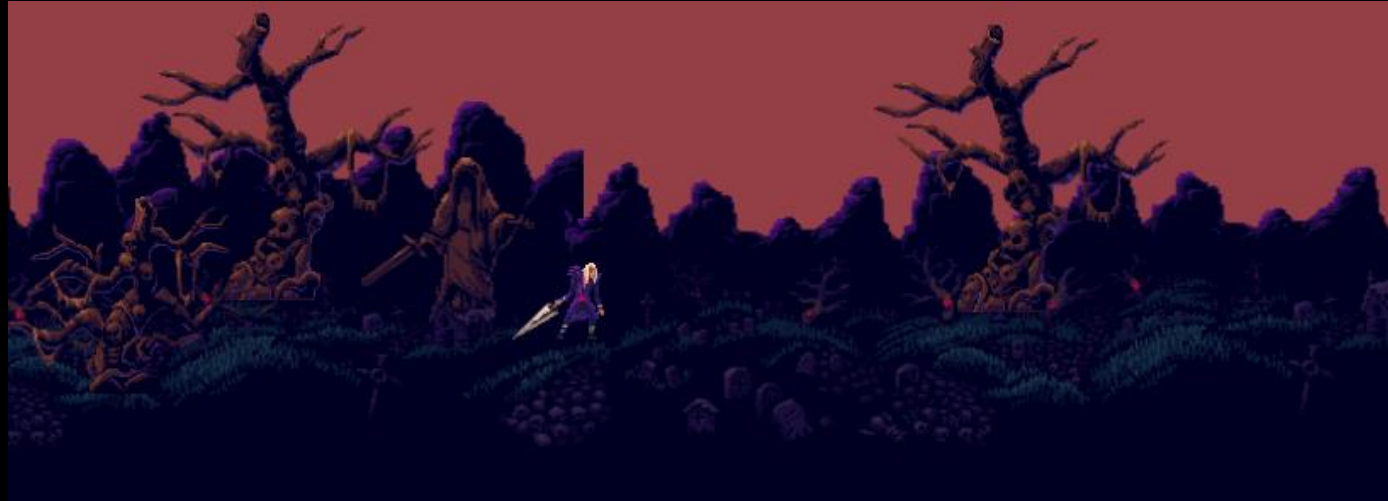
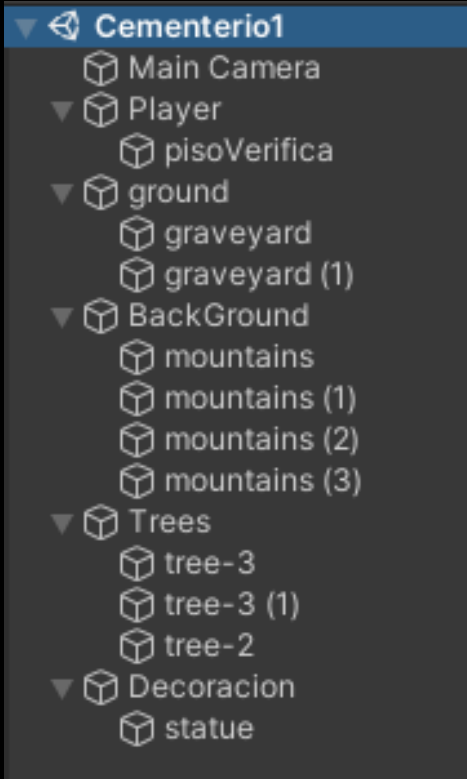
**Pedro Yuri Marquez Solis**

# Modificar el Script para que gire en la dirección del movimiento

```
public class PlayerController1 : MonoBehaviour
{
    public float speed;
    float velx, vely;
    Rigidbody2D rb;
    void Start()
    {
        rb = gameObject.GetComponent<Rigidbody2D>();
    }
    // Update is called once per frame
    void Update()
    {
        velx = Input.GetAxisRaw("Horizontal");
        vely = rb.velocity.y;
        rb.velocity = new Vector2(velx * speed, vely);
        if (rb.velocity.x > 0)
            transform.localScale = new Vector3(1, 1, 1);
        else if (rb.velocity.x < 0){
            transform.localScale = new Vector3(-1, 1, 1);
        }
    }
}
```

Pedro Yuri Marquez Solis

En el proyecto definir la siguiente estructura y agregar los siguientes GameObjects

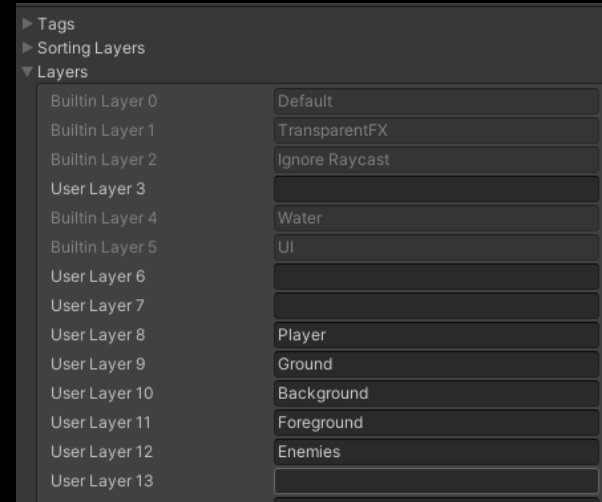


**Pedro Yuri Marquez Solis**

# Definir las siguientes capas – Layers

Los Layers son comúnmente utilizados por Cameras para renderizar solo una parte de la escena, y por las Lights para iluminar solo partes de la escena. Pero también pueden ser utilizados por raycasting para selectivamente ignorar los colliders o crear *collisions*.

0: Default  
1: TransparentFX  
2: Ignore Raycast  
4: Water  
5: UI  
8: Player  
9: Ground  
10: Background  
11: Foreground  
12: Enemies



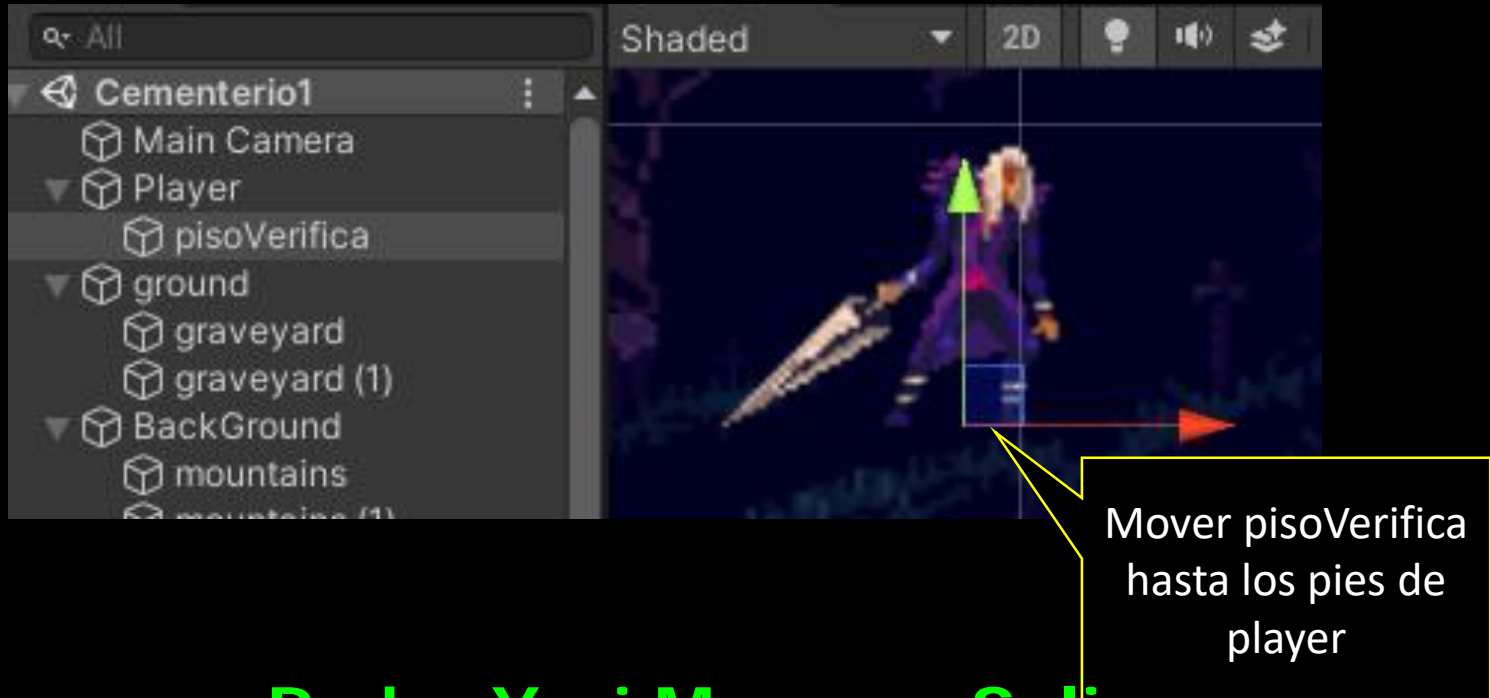
Pedro Yuri Marquez Solis

# Asignación de GameObjects a los Layers

- Graveyard → ground
- Mountain → Background
- Tree-3 → Background
- Statua → Background
- Tree-1 → Foreground
- Player a la capa Player

**Pedro Yuri Marquez Solis**

# Controlar los saltos de Player



**Pedro Yuri Marquez Solis**

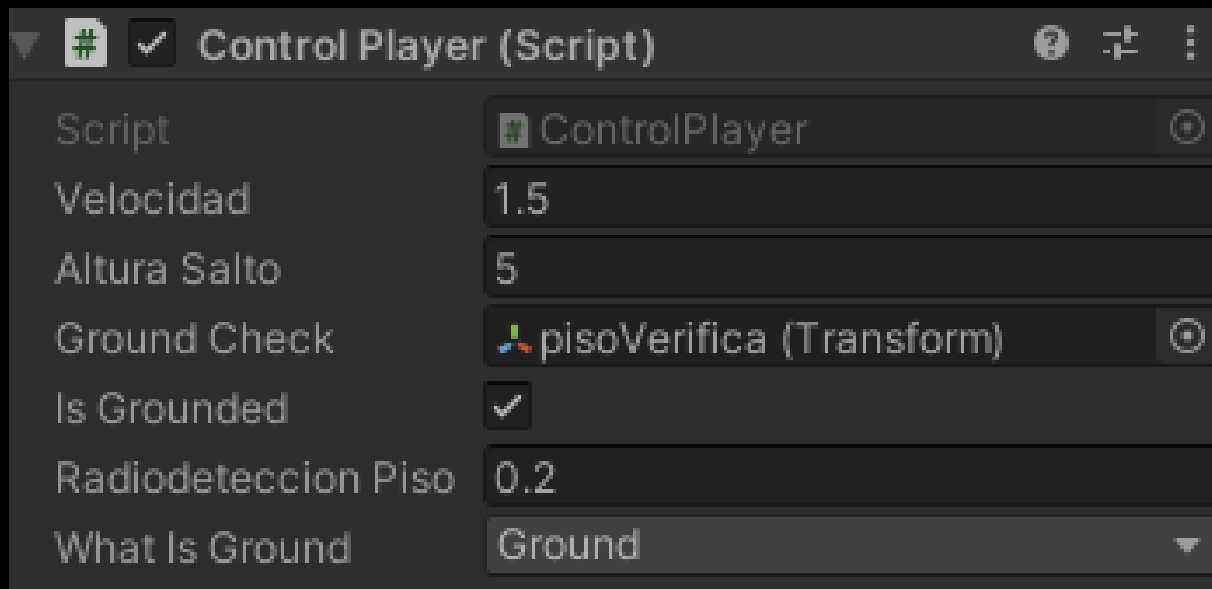
Agregar las siguientes propiedades a  
ControlPlayer:

```
public class ControlPlayer : MonoBehaviour
{
    Rigidbody2D rb;
    float velx=0, vely=0;
    public float velocidad;
    public float alturaSalto;

    public Transform groundCheck;
    public bool isGrounded;
    public float radiodeteccionPiso;
    public LayerMask whatisGround; // cual de los layers es el piso
    void Start()
```

**Pedro Yuri Marquez Solis**

# En las propiedades publicas de Player



**Pedro Yuri Marquez Solis**

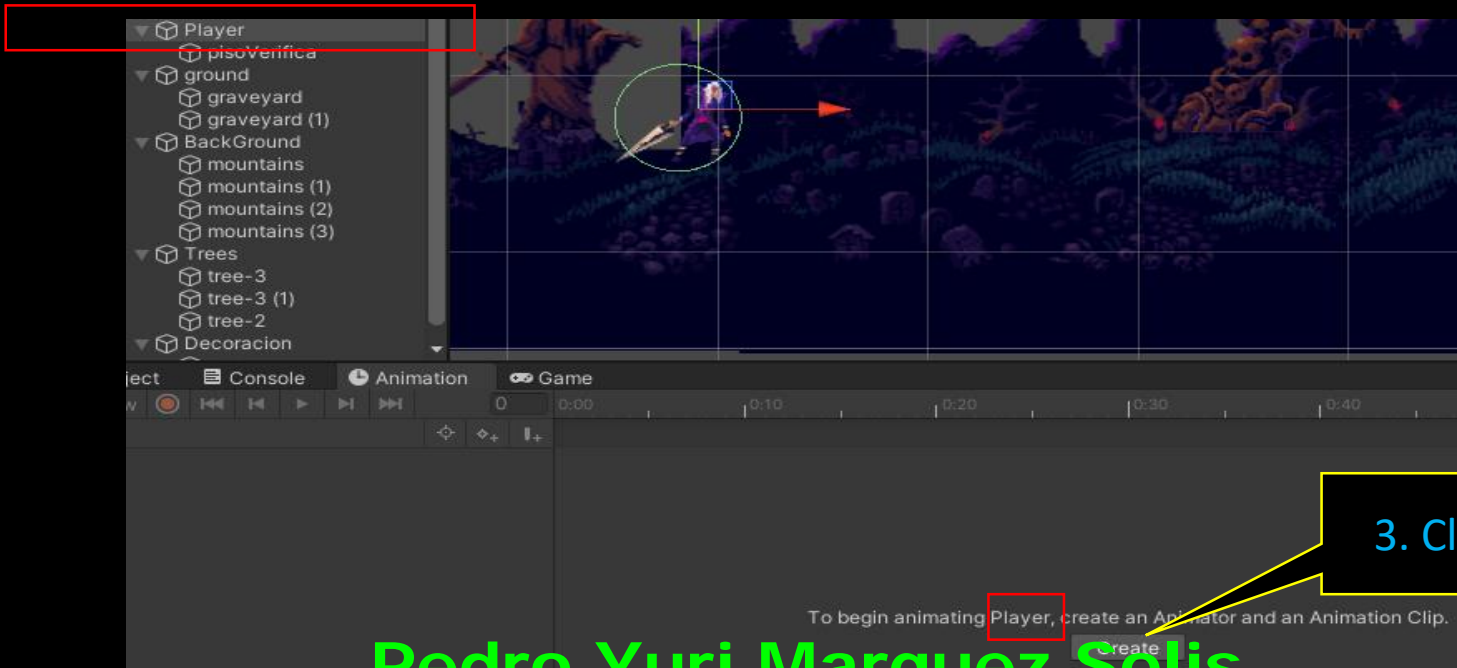


# Animations – sprites

1. Activar la herramienta de Animation

menú Window → Animation → Animation

2. Seleccionar el GameObject al que se agregará la animación.

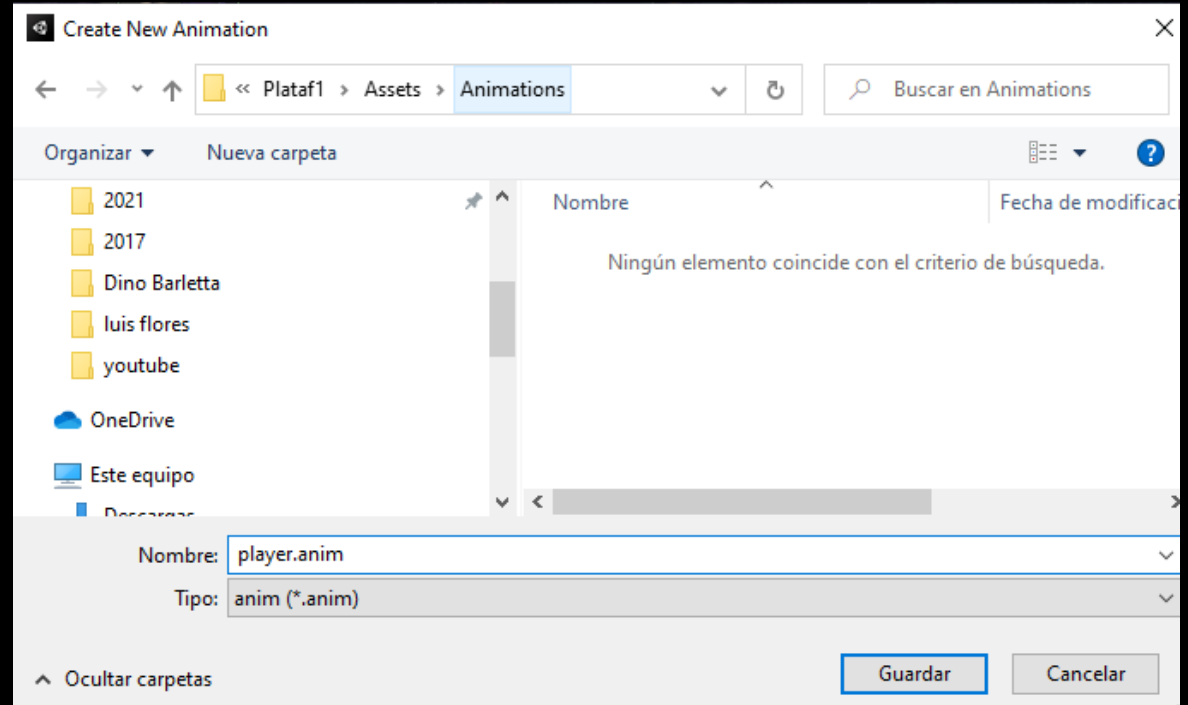


3. Click a create

Pedro Yuri Marquez Solis

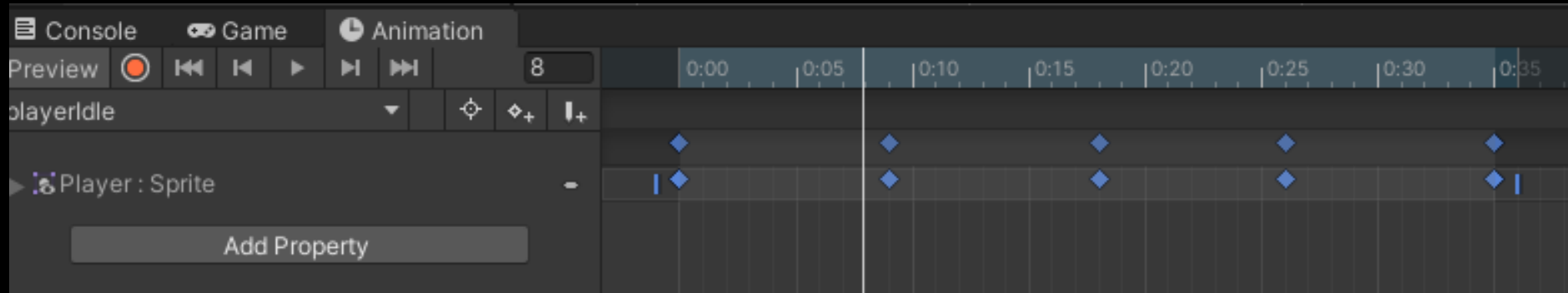
## 4. Indicar la ruta y el nombre de la animación

- PlayerIdle.anim



**Pedro Yuri Marquez Solis**

# Cambiar el tiempo de animación a 0.30

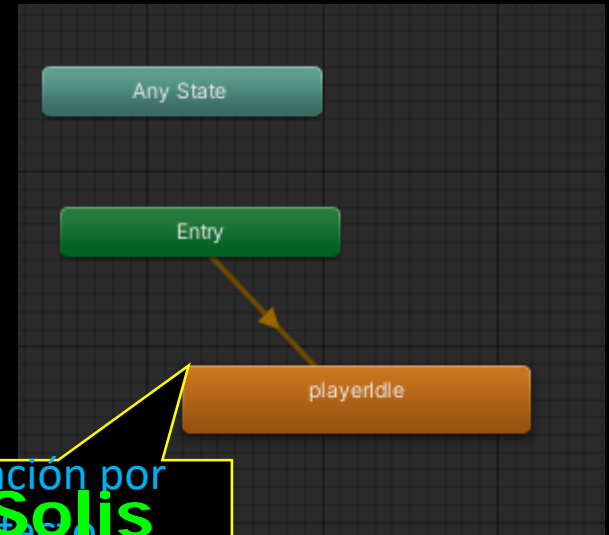
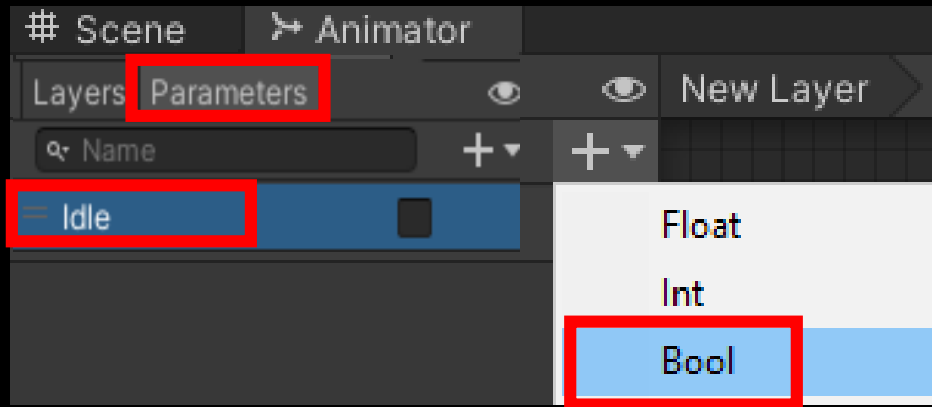


**Pedro Yuri Marquez Solis**

# Activar la animación



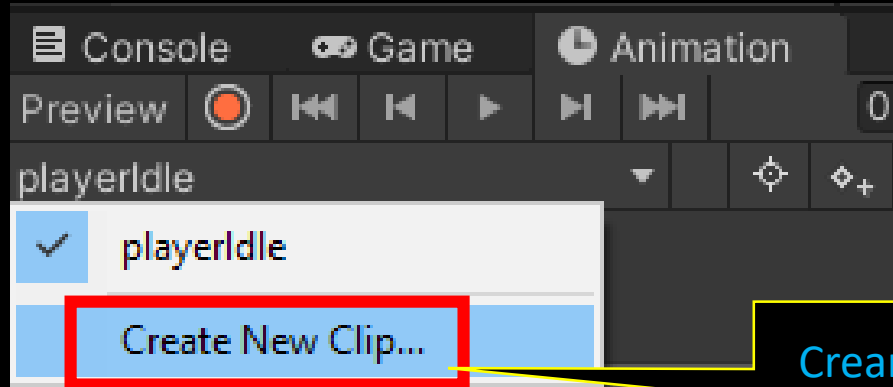
Crear un nuevo parámetro de tipo boolean que se llamará Idle



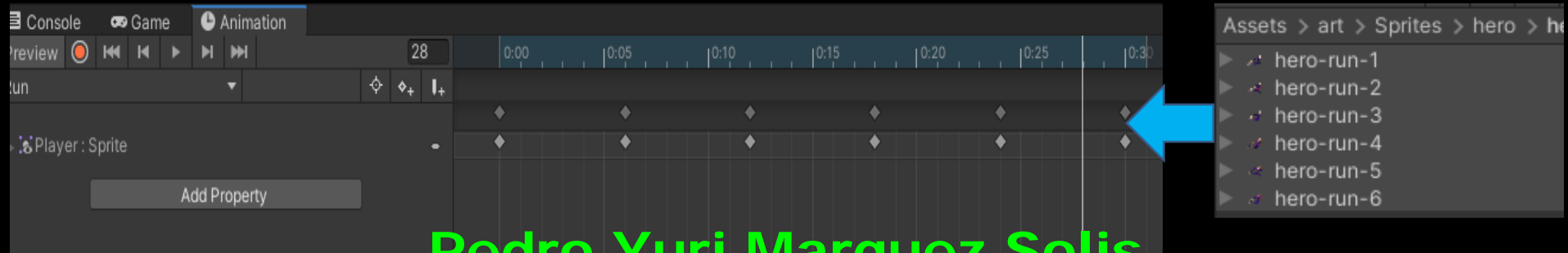
Animación por defecto

**Pedro Yuri Marquez Solis**

# Agregar una nueva animación

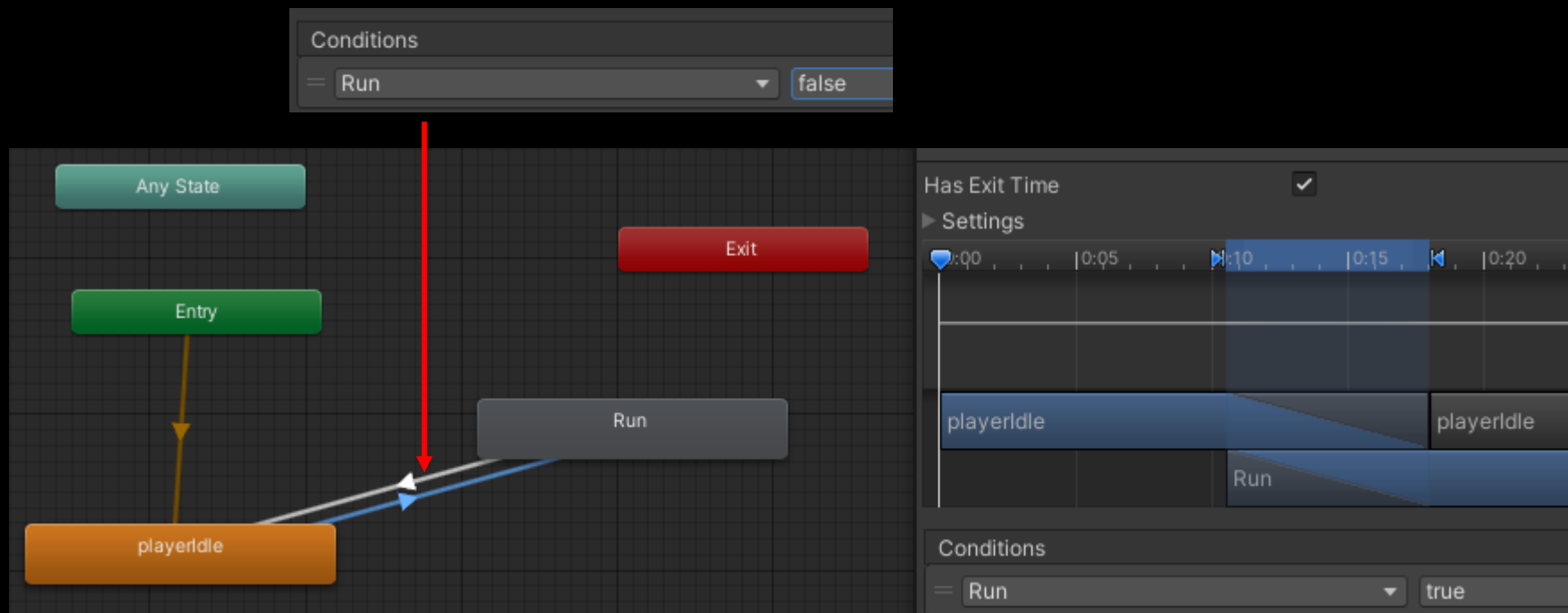


Crear la Animación **Run**



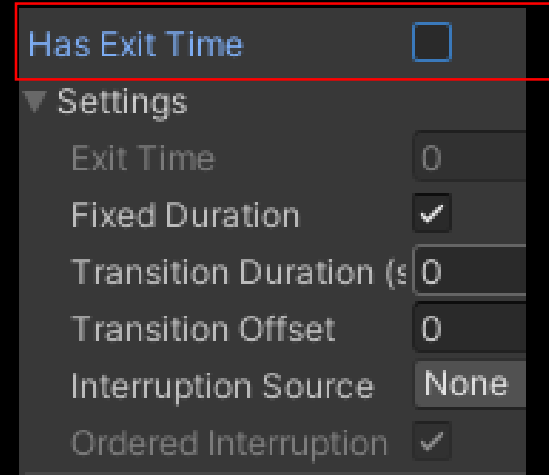
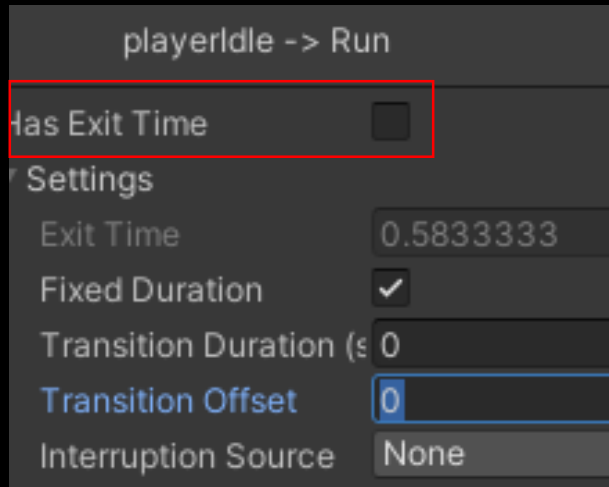
**Pedro Yuri Marquez Solis**

# Lógica de las animaciones



**Pedro Yuri Marquez Solis**

Para quitar el delay de pasar de una animación a otra



**Pedro Yuri Marquez Solis**

Prueba creando la animación de saltar

Una vez concluido comparte tu pantalla con el docente para que tu esfuerzo sea contabilizado con una nota

**Pedro Yuri Marquez Solis**



# Configurar los estados del animator

- Idle → jump

= Jump true

- Jump → idle

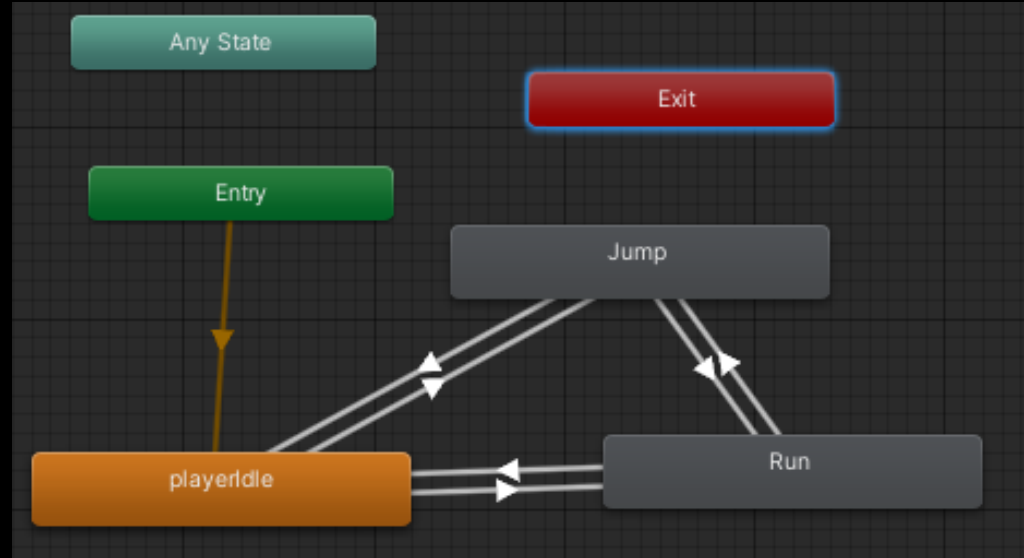
= Jump false

- Jump → run

= Jump false

- Run → Jump

= Jump true



**Pedro Yuri Marquez Solis**

# Actualizar el código para admitir el Jump

```
void Update()
{
    isGrounded = Physics2D.OverlapCircle(groundCheck.position, radiodeteccionPiso, WhatIsGround);
    rotar();

    if (isGrounded)
        anim.SetBool("Jump", false);
    else
        anim.SetBool("Jump", true);
}
```

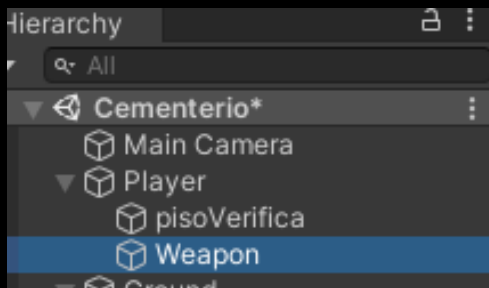
**Pedro Yuri Marquez Solis**

# Animación ataque

- Estado actual player:

- Collider sencillo Capsula, no tiene Trigger, ni efectos
- Tiene un GameObject transform: isGrounded.

- Ahora se le debe agregar un GameObject para el arma:

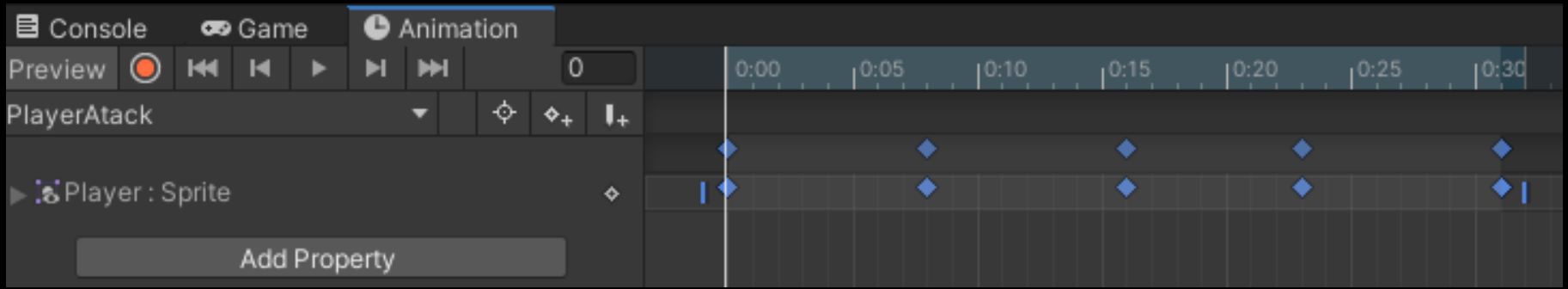


- Luego agregamos un Tag arma y lo asignamos al game Object Weapon



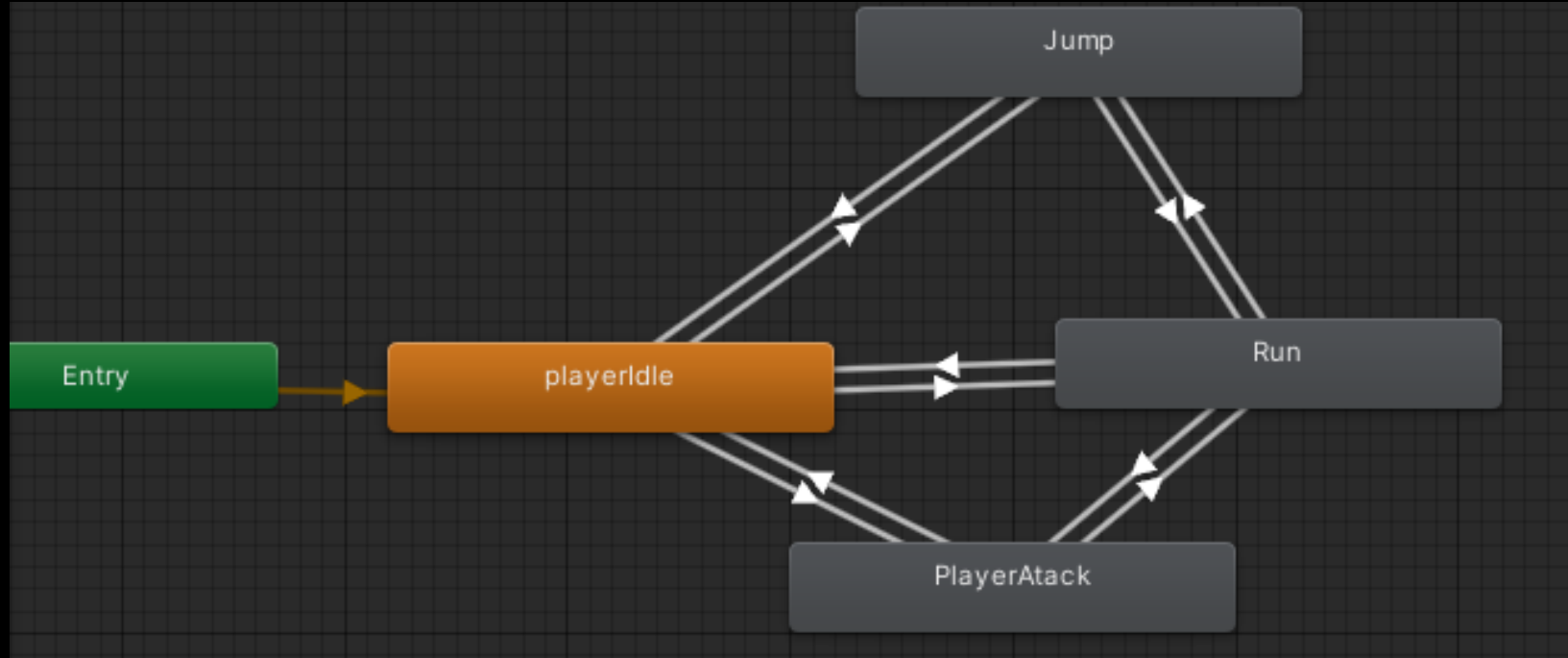
**Pedro Yuri Marquez Solis**

# Animación de ataque



**Pedro Yuri Marquez Solis**

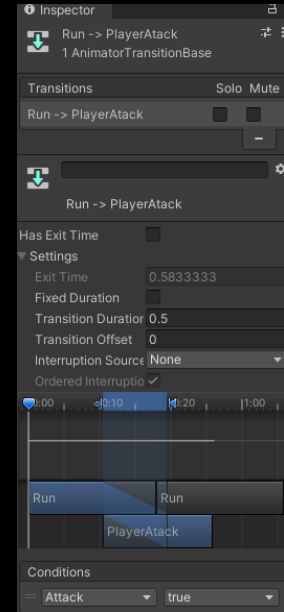
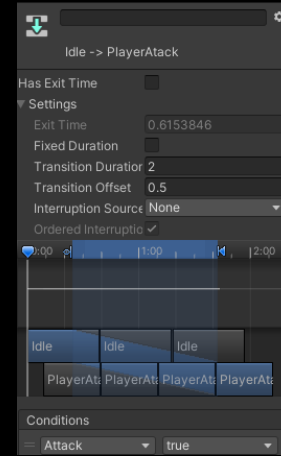
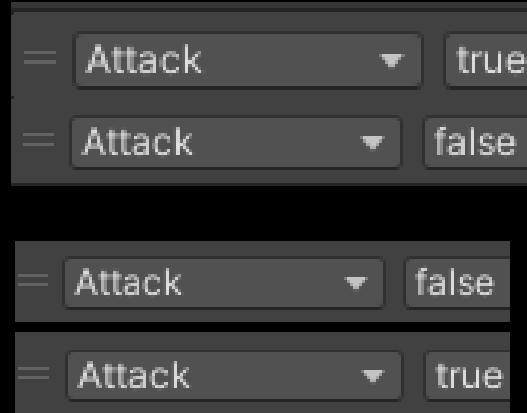
# Animator con Attack incluido



**Pedro Yuri Marquez Solis**

# Animator – condicionales de Attack

- Idle > Attack
- Attack → idle
- Attack → Run
- Run → Attack



- No olvidar establecer los **has exit time**



**Pedro Yuri Marquez Solis**

# Agregando el script de Attack

```
void Update()
{
    //se esta tocando o no el piso
    isGrounded = Physics2D.OverlapCircle(groundCheck.position, radioDeteccionPiso, WhatIsGround);
    rotar();
    if (isGrounded) anim.SetBool("Jump", false);
    else anim.SetBool("Jump", true);
    attack();
}

private void FixedUpdate()...
private void salto()...
private void Movimiento()...
private void rotar()...
public void attack() {
    if (Input.GetButtonDown("Fire1")) anim.SetBool("Attack", true);
    else anim.SetBool("Attack", false);
}
```

**Pedro Yuri Marquez Solis**

# Agregar weapon - collider

- Agregar un boxCollider2d y ubicarlo un tanto desplazado

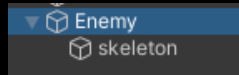


Pedro Yuri Marquez Solis

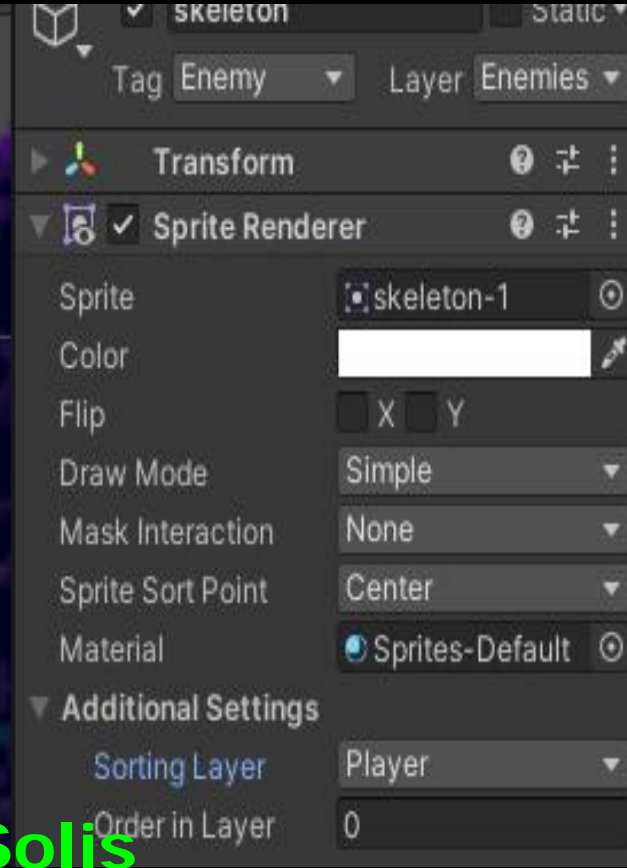


# Agregar Enemigo

En un game Object



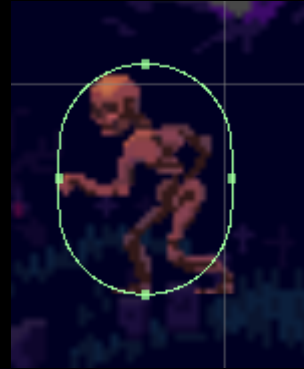
- Rigidbody2d Freeze **Z**



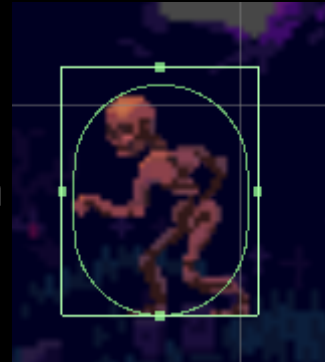
**Pedro Yuri Marquez Solis**

# Agregar Colliders

- Capsule Collider:  
caminar, contacto con paredes, piso

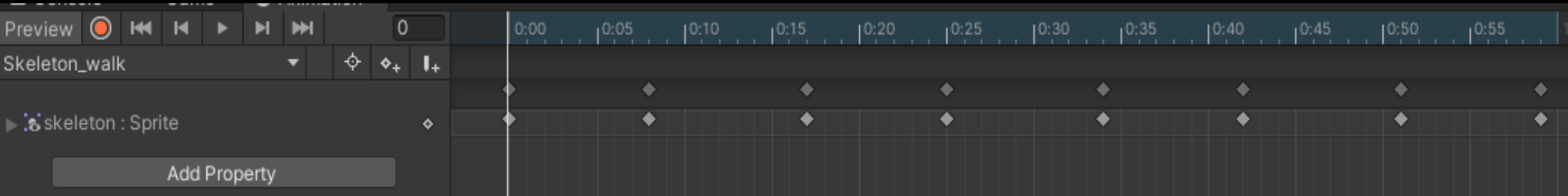


- Agregar otro BoxCollider2d :  
para detectar el contacto con la espada



**Pedro Yuri Marquez Solis**

# Animación Enemigo:



**Pedro Yuri Marquez Solis**

# Muestra la animación del enemigo

Una vez concluido comparte tu pantalla con el docente para que tu esfuerzo sea contabilizado con una nota

**Pedro Yuri Marquez Solis**