Universidad Nacional de San Agustin
Facultad de Ingenieria, Produccion y Servicios
Escuela Profesional de Ciencia de la Computacion

Juego 2d usando el Efecto Parallax

Integrantes: Kelvin Paul Pucho Zevallos
Luis Armando Sihuinta Perez
Josnick Chayña Batallanes
Angelo Aldo Perez Rodriguez

Profesor: Diego Alonso Iquira Becerra

Fecha de realización: 31 de mayo de 2022
Fecha de entrega: 1 de junio de 2022

Arequipa - Perú

Índice de Contenidos

Índice de Figuras

Índice de Códigos

1. Creación de Objetos

La escena esta constituida por los siguientes objetos:

- **Grid**: El cual contiene el TileMap en este caso se uso para representar el suelo del juego.
- **MainCamera**: La cámara principal que enfoca los movimientos del personaje junto al cielo.
- **Cal_Idle_0**: Es el Player de nuestro juego, este esta animado mediante sprite.
- **Clouds**: Sprite de Nubes.
- **Mountains**: Sprite de Montañas.
- **Trees**: Sprite de Arboles.

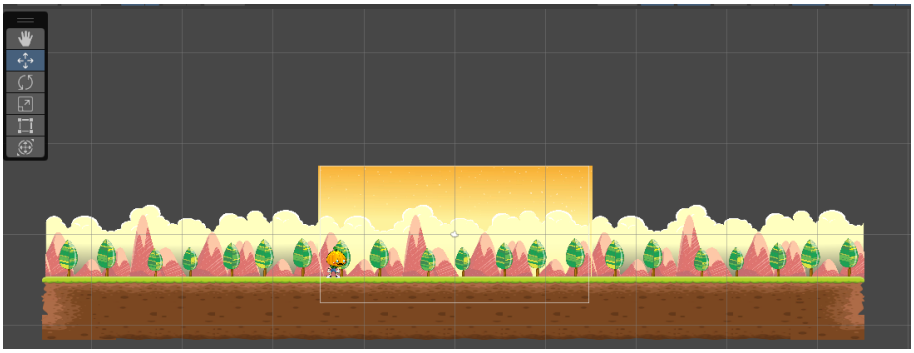

Figura 1: Escenario

1.1. Vista de Juego

La cámara enfoca entorno al jugador.

Figura 2: Vista del Juego

1.2. Hierarchy

Figura 3: Hierarchy

2. Assets

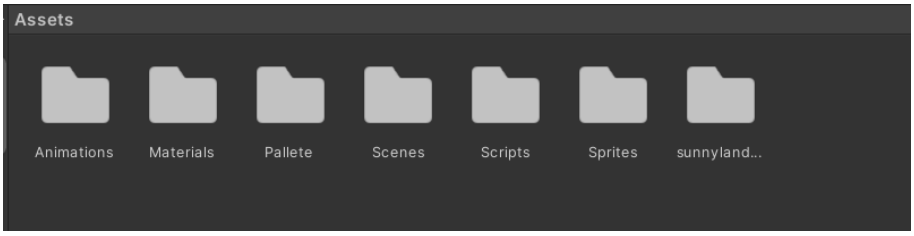

Figura 4: Assets

- El archivo de Animaciones contiene los movimientos animados del Idle,Run y Jump junto con su controlador para pasar de un estado de movimiento a otro.
- El archivo de Pallete contiene los sprite para el tilamap en este caso el suelo.
- El archivo de Sprites y SunnyLand es para usar los sprites de fondo para hacer el efecto parallax.

3. Sprites

Sprites usados de la terceros.

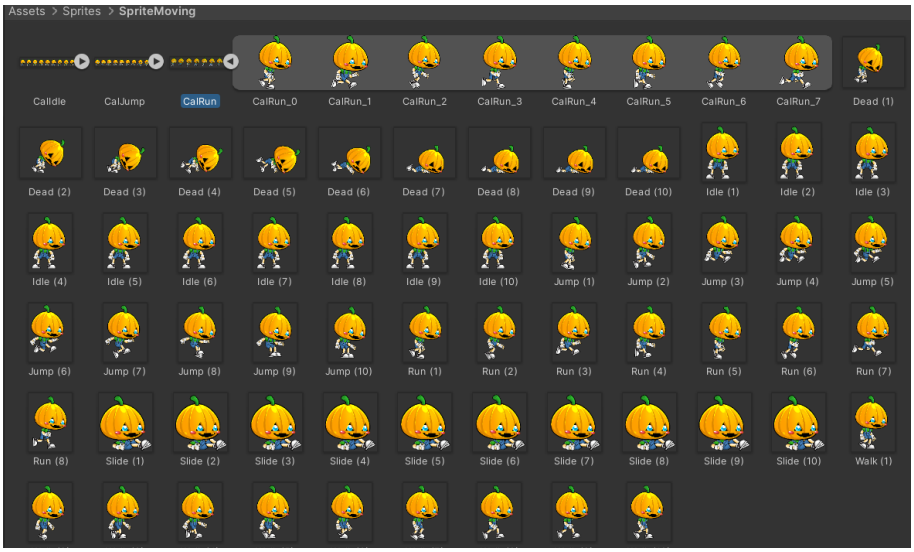

Figura 5: Sprite del personaje

Figura 6: Sprite del fondo

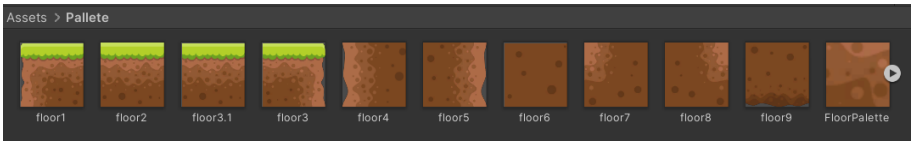

Figura 7: Sprite en el Pallete del suelo

4. Animation Player

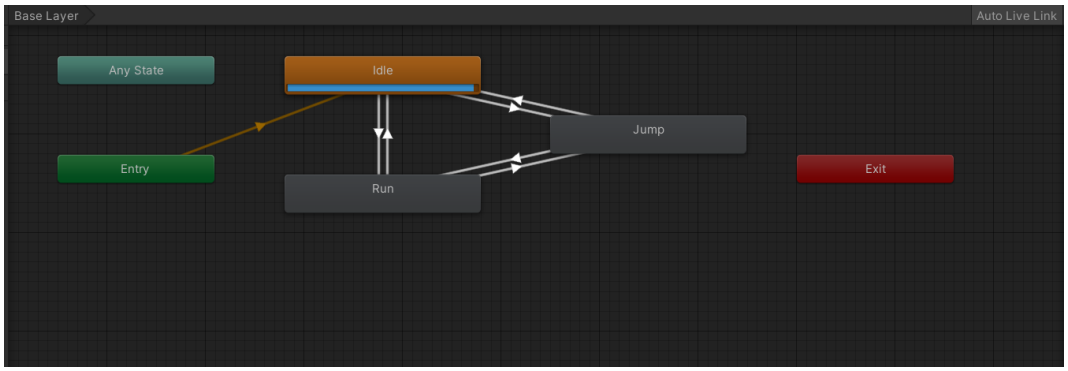

Figura 8: Base Animator

5. Inspector del Objeto Player

- **Inspector del Player**

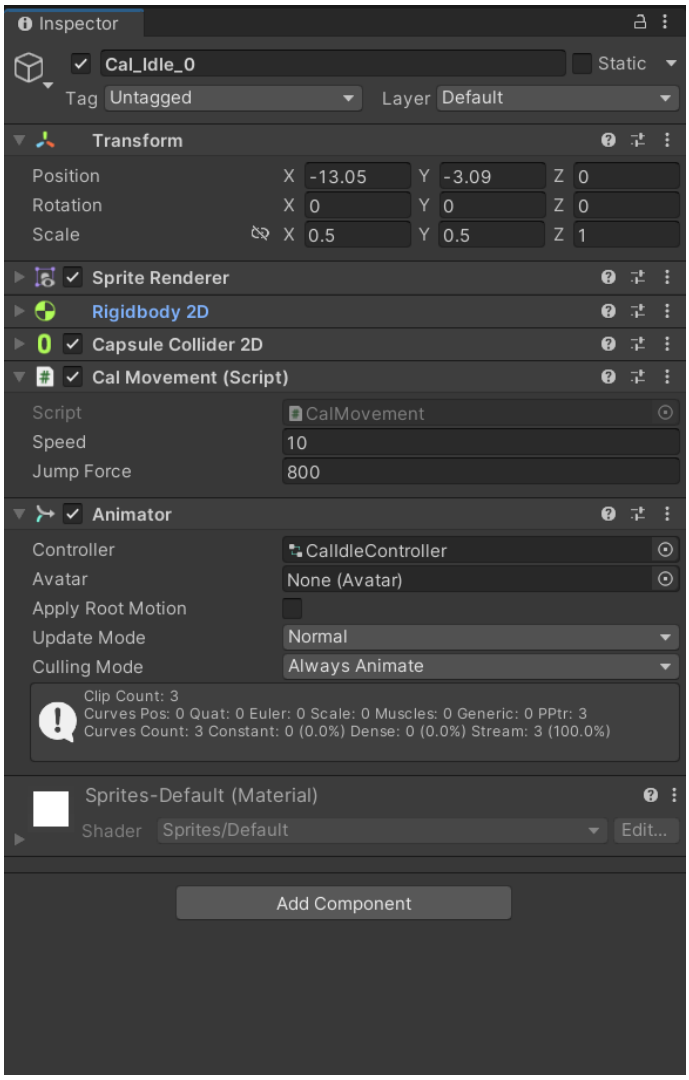

Figura 9: Sprite Renderer, Rigidbody2D, Capsule Collider 2D, Cal Movement (script), Animator

- **Sprite Renderer**: Es el componente que contiene el sprite actual que se ve en la pantalla, ademas de que tiene como capa de ordenamiento el Player.
- **RigidBody 2D**: Es usado para que nuestro personaje tenga fisicas las cuales usaremos para el movimiento y salto.
- **Capsule Collider 2D**: De manera que podemos especificar la cápsula para que el personaje colisione con el suelo.
- **Script**: El script con variables publicas para especificar la velocidad e impulso para el salto del personaje.
- **Animator**: Contiene todo los estados de animación administrador por un controlador del cual se instanciara por medio de un script para cambiar los estados de animación.

• **Inspector del TileMap**

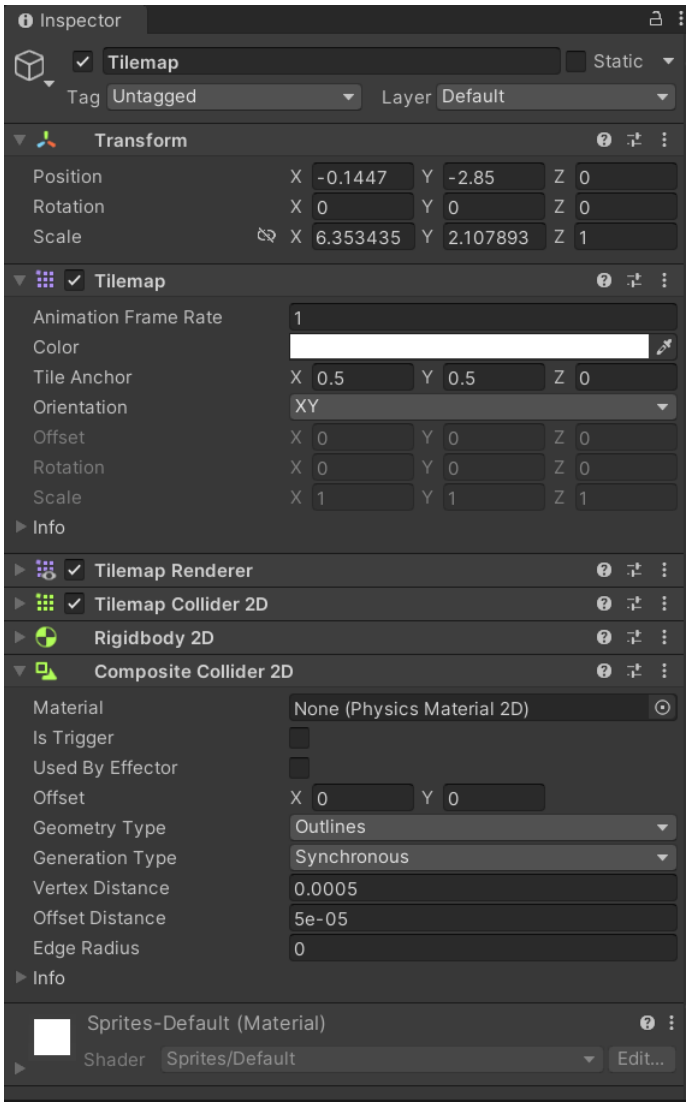

Figura 10: TileMap, TileMap Renderer, TileMap Collider 2D, Rigidbody 2D, Composite Collider 2D

- **Tilemap**: Son las celdas que dispone en todo la escena, se modifico su tamañp para el sprite.
- **Tilemap Renderer**: El cual contiene los tile que en este caso es el suelo.
- **TileMap Collider 2D**: Este componente es importante debido a que se quiere que el jugador colisione con el suelo para evitar la caida.
- **Composite Collider 2D**: Debido a que cada tile se considera como un collider entonces el usuario tendría que verificar la colisión en cada uno por ello se agrupo usando este componente para que solamente tenga un collider que es del suelo y no por celdas.

6. Efecto Parallax

Para este efecto se hizo los siguientes pasos:

- Conseguir los sprite necesarios.

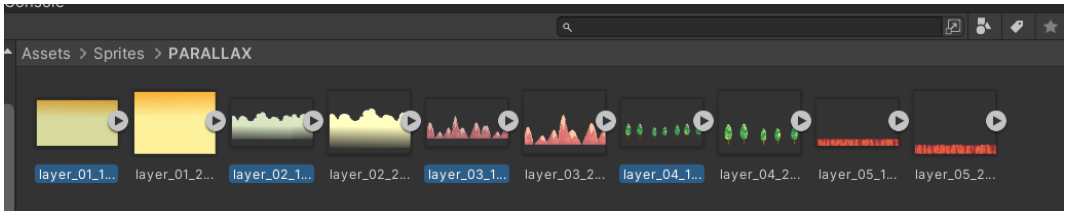

Figura 11: Sky,Mountains y Trees

- Instanciar las imágenes como objetos, se hizo 3 instancias de cada uno para llenar el campo de juego.

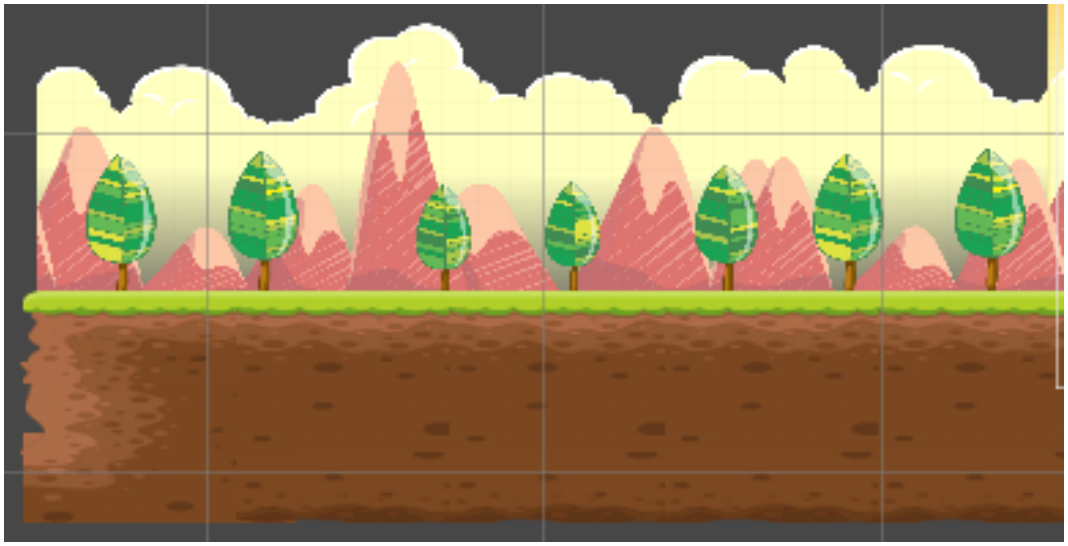

Figura 12: Sky,Mountains y Trees

- Despues para animarlos de tal manera que contega un efecto parallax. Se hizo de forma diferente ya que si se utilizaba un solo bloque de imágenes y hacerlos renderizar de forma repetitiva creaba un efecto diferente y no concuerda. Por ello se hizo 3 instancias par simular el efecto parallax.
- Entonces se creo un script (Parallax Efect) que contiene una variable parallax multiplayer, esta variable especifica la velocidad de movimiento de cada sprite. Como se quiere que las efecto se vea realista se declaro esta variable de forma manual haciendo que las nubes, montañas y los arboles se muevan a diferente velocidad. Por ello para las nubes se coloco una velocidad de 0.9, para las montañas con 0.7 y para los arboles con 0.3. Mientras mas menor sea el valor mas rápido sera la animación ya que los arboles al estar mas cerca se mueven mas rápido mientras que las nubes mas lento.

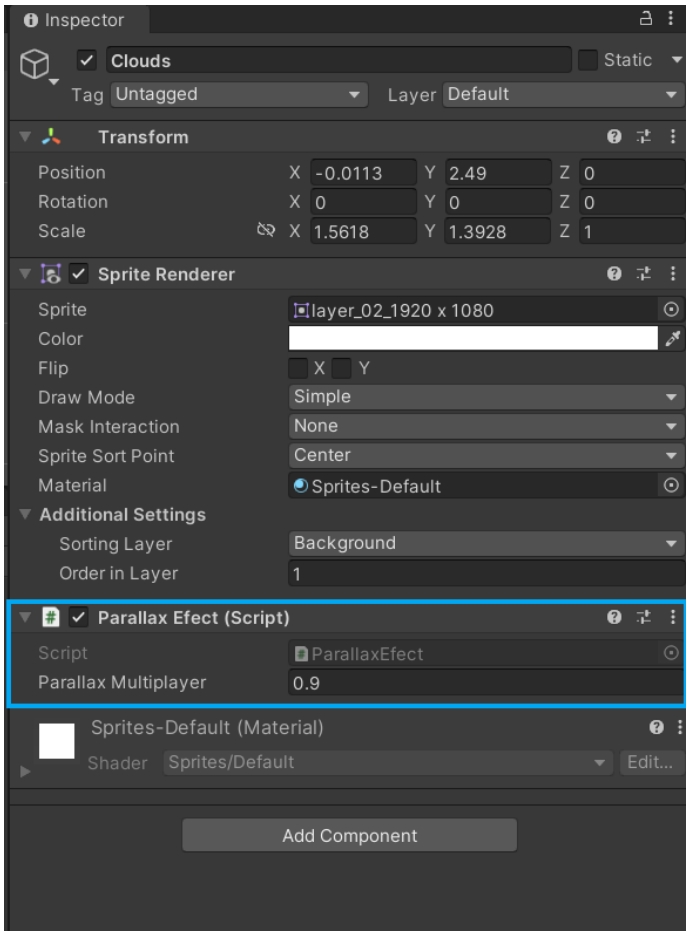

Figura 13: Inspector del objeto Clouds

- Pasos para la implementación del Script.

Código 1: ParallaxEfect.cs

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class ParallaxEfect : MonoBehaviour
6  {
7      [SerializeField] public float parallaxMultiplayer;
8
9      private Transform cameraTransform;
10     private Vector3 previousCameraPosition;
11     private float spriteWidth, startPosition;
12
13     // Start is called before the first frame update
14     void Start()
15     {
16         cameraTransform = Camera.main.transform;
```

```
17        previousCameraPosition = cameraTransform.position;
18        spriteWidth = GetComponent<SpriteRenderer>().bounds.size.x;
19
20        Debug.Log(spriteWidth);
21        startPosition = transform.position.x;
22    }
23
24    // Update is called once per frame
25    void LateUpdate()
26    {
27        float deltaX = (cameraTransform.position.x - previousCameraPosition.x) *
      ↪ parallaxMultiplayer;
28        float moveAmount = cameraTransform.position.x * (1 - parallaxMultiplayer);
29        //Debug.Log(cameraTransform.position);
30        transform.Translate(new Vector3(deltaX, 0, 0));
31        previousCameraPosition = cameraTransform.position;
32
33        if (moveAmount > startPosition + spriteWidth)
34        {
35            transform.Translate(new Vector3(spriteWidth, 0, 0));
36            startPosition += spriteWidth;
37        }
38        else if (moveAmount < startPosition - spriteWidth)
39        {
40            transform.Translate(new Vector3(-spriteWidth, 0, 0));
41            startPosition -= spriteWidth;
42        }
43    }
44 }
```

- Como se muestra en el código 1, la variable ParallaxMultiplayer contiene un valor que equilibra la velocidad de movimiento de los sprites.
- Las dos variables cameraTransform y previuosCameraPosition almacenan la posición actual y previa de la cámara.
- Entonces en la función Start iniciamos las dos variables privadas que contiene la posición de la cámara.
- En la función LateUpdate el cual invocado después de que el personaje se mueva. Se tiene una variable deltaX el cual toma la distancia de la cámara de su posición previa y su posición actual y lo multiplica por la variable parallaxMultiplayer para modificar la distancia en función de cada objeto.
  Después se modifica el movimiento del objeto (sprite) de acuerdo a la distancia entre el movimiento de la cámara para así trasladar suavemente el sprite a la posición que va el personaje. Y finalmente se guarda la posición de la cámara previa.
  Los parent sprite que contienen este script son Clouds, Mountains y Trees que modifican su movimiento de acuerdo a la cámara y con la ayuda de una variable de la parallaxMultiplayer podemos modificar la velocidad de cada sprite. Por ello es que se crea ese efecto parallax.
- Para crear un juego infinito solo posicionamos los sprite con la ayuda de una variable que indica la cantidad de movimiento si este es mayor que la anchura del sprite entonces que

se reubique a la derecha y si este es menor que se posicione a la izquierda.

7. Codigos

7.1. Follow Player

Este script pertenece al objeto de la cámara principal que se reubica en la posición del movimiento del personaje.

En este script en cada frame colocamos una condición el cual interactúa con el tilemap en este caso con el suelo. Cuando el personaje esta cerca del borde izquierda o derecha entonces el suelo (composite tile) se reubica, haciendo que el juego tenga un suelo infinito y evitar que el personaje caiga.

Con la ayuda de la posición previa y la actual de la cámara mas una variable que especifica si el usuario esta cerca del borde es que posicionamos el tile del suelo y posteriormente guardamos la nueva posición de la cámara.

Código 2: FollowPlayer.cs

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class FollowPlayer : MonoBehaviour
6  {
7      [SerializeField] public GameObject target;
8      [SerializeField] public GameObject tileMap;
9      private float previouStep = 25.0f;
10     private float previousPosition;
11     //public GameObject[] floorTiles;
12     private float zCam = -10.0f;
13     private float yPos = 2.4f;
14
15     void Start()
16     {
17         previousPosition = transform.position.x;
18     }
19
20     // Update is called once per frame
21     void Update()
22     {
23
24
25         if (transform.position.x  > previouStep + previousPosition)
26         {
27             tileMap.transform.position = new Vector3(target.transform.position.x, 0, 0);
28             previousPosition = transform.position.x;
29         }
30         else if(transform.position.x  < previousPosition - previouStep)
31         {
```

```
            tileMap.transform.position = new Vector3(target.transform.position.x, 0, 0);
            previousPosition = transform.position.x;
        }

        transform.position = new Vector3(target.transform.position.x, target.transform.position.y +
    ↪ yPos, target.transform.position.z + zCam);

    }
}
```

7.2. Cal Movement

En este script se establece los movimiento del personaje por medio de datos de entrada que es el teclado. También en cada iteración se activa la animación de acuerdo al movimiento del personaje por medio de la variable animator que instancia al controlador de la animación para cambiar de estado Idle a Run o de Idle a Jump o de Run a Jump por medio de condiciones. Se uso Raycast para trazar la distancia entre el personaje y el suelo para así verificar si el personaje esta sobre un elemento. Y para que el movimiento sea mas realista se especifica la velocidad y el impulso.

Código 3: CalMovement.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class CalMovement : MonoBehaviour
{
    public float speed = 10;
    public float jumpForce = 800;
    private float horizontal;
    private bool Grounded;

    private Rigidbody2D rb2D;

    private Animator animator;
    // Start is called before the first frame update
    void Start()
    {
        rb2D = GetComponent<Rigidbody2D>();
        animator = GetComponent<Animator>();
    }

    // Update is called once per frame
    void Update()
    {
        horizontal = Input.GetAxisRaw("Horizontal");

        if (horizontal < 0.0f) transform.localScale = new Vector3(-0.5f,0.5f, 1.0f);
        else if (horizontal > 0.0f) transform.localScale = new Vector3(0.5f, 0.5f, 1.0f);

```

```
30        animator.SetBool("Running", horizontal != 0.0f);
31
32        Debug.DrawRay(transform.position, Vector3.down * 2.0f,Color.red);
33        if (Physics2D.Raycast(transform.position,Vector3.down,2.0f)) // este es la linea trazada para
      ↪ ver el collider
34        {
35            Grounded = true;
36        }
37        else Grounded = false;
38
39        if (Input.GetKeyDown(KeyCode.Space) && Grounded)
40        {
41            animator.SetTrigger("Jumping");
42            Jump();
43
44        }
45    }
46
47    private void Jump()
48    {
49        rb2D.AddForce(Vector2.up*jumpForce);
50    }
51
52    private void FixedUpdate()
53    {
54        rb2D.velocity = new Vector2(horizontal * speed, rb2D.velocity.y);
55    }
56 }
```

Referencias

- **Grabación de Explicación:** https://drive.google.com/file/d/1jF8CGdjg1MnkQiH1LWf78eK7V____zQUo/view?usp=sharing
- Sprite del personaje: https://www.gameart2d.com/jack-o-lantern-free-sprites.html
- Sprite de fondo: https://opengameart.org/content/3-parallax-backgrounds
- https://iainscarr.github.io/parallaxium/
- Guia para el efecto parallax: https://www.youtube.com/watch?v=LDeIGXwv8tc
- https://www.youtube.com/watch?v=_f4nI8FtVqQ
- https://www.youtube.com/watch?v=GbmRt0wydQU
- https://assetstore.unity.com/packages/2d/environments/sunnyland-woods-129708#description
- https://www.youtube.com/watch?v=7bJT6rf-Jvk
- https://learn.unity.com/tutorial/2d-roguelike-setup-and-assets?uv=5.x&projectId=5c514a00edbc2a0020694718#5c7f8528edbc2a002053b6fa