Sistema de Movimiento de Personaje

-Ciro Sebastián Hernández López

Contenidos

Este proyecto consiste en una escena reminiscente a los juegos clásicos de plataformas, en ella se encuentran el objeto jugador, una meta y varios obstáculos. El jugador cuenta con una serie de opciones de movimiento para recorrer la escena, cada una de estas estas controlada por un Script de C#.

Funcionamiento

El script más importante es el llamado “PlayerMovement”, este permite mover al jugador usando las teclas del teclado.

Este funciona primero aplicando una velocidad a un RigidBody2D aplicado al objeto jugador, la dirección de esta depende del Input Axis “Horizontal” asignado a las teclas “A” y “D”.

```
// Encargada de movimientos de izquierda a derecha
Move = Input.GetAxis("Horizontal");

rb.velocity = new Vector2(speed * Move, rb.velocity.y);
```

Para permitir saltar al jugador se aplica una fuerza al mismo RigidBody2D cuando se presiona la tecla “W”, haciendo que se eleve.

```
// Encargada del Salto
if (Input.GetKeyDown(KeyCode.W) && isJumping == false && JumpCount > 1)
{
    rb.AddForce(new Vector2(rb.velocity.x, jump));
    isJumping = true;
    JumpCount--;
}
```

También se realiza un chequeo cada que toca el suelo usando una variable llamada “JumpCount”, solo permite saltar cuando sea mayor que 1, al saltar esta se reduce por uno, al tocar el suelo se suma 1. Esto evita que el jugador pueda saltar varias veces antes de tocar el suelo.

```
//Determina si el Jugador esta saltando o no
Mensaje de Unity | 0 referencias
private void OnCollisionEnter2D(Collision2D other)
{
    if(other.gameObject.CompareTag("Floor"))
    {
        isJumping = false;
        JumpCount++;
    }
}
```

Para agacharse primero se revisa si el jugador está saltando, en caso de que no se reduce a la mitad la escala del objeto jugador en caso de que la tecla “S” está siendo presionada, y se regresa a su escala normal cuando esta se suelta.

```
//Controla la funcion de "Agacharse"
if (Input.GetKeyDown(KeyCode.S) && isJumping == false)
{
    transform.localScale = new Vector2(1f, .5f);
}

if (Input.GetKeyUp(KeyCode.S) && isJumping == false)
{
    transform.localScale = new Vector2(1f, 1f);
}
```

En caso de que el jugador este en el aire cuando presiona la tecla “S” este realizara una caída rápida, este utiliza la misma funciona de manera similar que el salto. Aplica una fuerza al RigidBody2D, pero esta vez con un valor negativo mandándolo hacia abajo.

```
//Permite realizar una "Caida Rapida"
if (Input.GetKeyDown(KeyCode.S) && isJumping == true)
{
    rb.AddForce(new Vector2(rb.velocity.x, fall));
}
```

Otro Script es encargado de las funciones de muerte y reaparición. Este funciona reiniciando la posición del jugador cada que reaparece.

```
//Metodo encargado de la reaparición del jugador cuando muere o se reinicia el nivel
3 referencias
private void Respawn()
{
    transform.position = new Vector2(0, 0);
}
```

Para activar el método “Respawn” se tiene que llegar a la meta, tocar algo que mate al jugador o presionando el botón de reinicio.

```
// Update is called once per frame
Mensaje de Unity | 0 referencias
void Update()
{
    //Permite reinicar el nivel en cualquier momento
    if (Input.GetKeyDown(KeyCode.R))
    {
        Respawn();
    }
}
```

Para saber si se entra en contacto con la meta o un objeto dañino se usa una función OnTriggerEnter que verifica el Tag del objeto.

```
//Determina si el jugador entra en contacto con algo dañino o la meta
Mensaje de Unity | 0 referencias
private void OnTriggerEnter2D(Collider2D other)
{
    if (other.gameObject.CompareTag("Goal"))
    {
        Respawn();
        WinScreen();
    }
    if (other.gameObject.CompareTag("KillBox"))
    {
        Respawn();
        DeathScreen();
    }
}
```

Para mostrar pantallas de muerte o de victoria de utiliza una serie de métodos e “Invokes” para activar y desactivar los GameObjects correspondientes.

```
//Metodos encargados de mostrar y dejar de mostrar la pantalla de Muerte
1 referencia
private void DeathScreen()
{
    Death.SetActive(true);
    Invoke("StopDeathScreen", 2f);
}

1 referencia
private void StopDeathScreen()
{
    Death.SetActive(false);
}

//Metodos encargados de mostrar y dejar de mostrar la pantalla de Victoria
1 referencia
private void WinScreen()
{
    Win.SetActive(true);
    Invoke("StopWinScreen", 2f);
}

1 referencia
private void StopWinScreen()
{
    Win.SetActive(false);
}
```

El ultimo Script se dedica a que la cámara siga al jugador con un Offset en el eje Z para poder mostrar la escena conforme el jugador se mueve.

```
public class CameraMovement : MonoBehaviour
{
    public Transform player;
    public float speed=2f;
    public Vector3 Offset;
    Mensaje de Unity | 0 referencias
    void Start()
    {

    }

    // Update is called once per frame
    Mensaje de Unity | 0 referencias
    void Update()
    {
        transform.position = Vector3.Lerp(transform.position, player.position + Offset, speed);
    }
}
```

Usos

El sistema de movimiento en el Script “PlayerMovement” es fácilmente transferible a otros programas con movimiento lateral en 2 dimensiones.