Sistema de Combate

Isaac López Nolasco

Parcial 1

Programación Para Videojuegos II

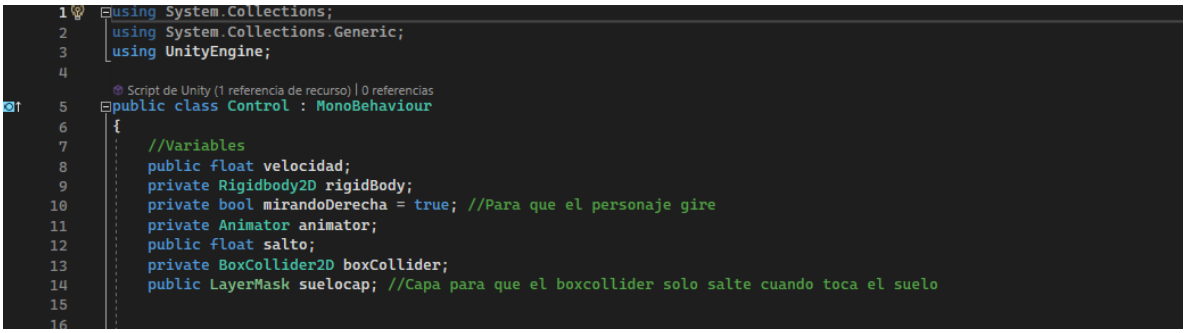
UTECA

08/02/2023

Índice

Contenido

Esta es la parte inicial de mi programa, aquí declaro las variables iniciales



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

Script de Unity (1 referencia de recurso) | 0 referencias
public class Control : MonoBehaviour
{
    //Variables
    public float velocidad;
    private Rigidbody2D rigidBody;
    private bool mirandoDerecha = true; //Para que el personaje gire
    private Animator animator;
    public float salto;
    private BoxCollider2D boxCollider;
    public LayerMask suelocap; //Capa para que el boxcollider solo salte cuando toca el suelo
```

Utilizaremos las variables para crear un controlador del personaje principal, en este caso mi personaje se mueve en un eje x, y, con las flechas de movimiento

```
Mensaje de Unity | 0 referencias
private void Start()
{
    rigidBody = GetComponent<Rigidbody2D>();
    boxCollider = GetComponent<BoxCollider2D>();
    animator = GetComponent<Animator>();
}

Mensaje de Unity | 0 referencias
void Update()
{
    ProcesarMovimiento(); //LLama la funcion
    ProcesarSalto();
}

1 referencia
bool EstaSuelo()
{
    //Bounds es para los limites del colisionador
    //EL raycast es como un laser, que sirve para para tener informacion o saber cuando un objeto colisiona con el, todo esto para saber cuando el personaje toca el suelo y no salte infinitas veces
    RaycastHit2D raycastHit = Physics2D.BoxCast(boxCollider.bounds.center, new Vector2(boxCollider.bounds.size.x, boxCollider.bounds.size.y), 0f, Vector2.down, 0.2f, suelocap );
    return raycastHit.collider != null;
}

1 referencia
void ProcesarSalto()
{
    //Salto con barra de espacio
    if(Input.GetKeyDown(KeyCode.Space) && EstaSuelo())
    {
        rigidBody.AddForce(Vector2.up * salto, ForceMode2D.Impulse);
    }
}
```

Aquí utilizamos el animator dentro del script, para que sepa cuando ejecutarlas y el salto

```
1 referencia
void ProcesarSalto()
{
    //Salto con barra de espacio
    if(Input.GetKeyDown(KeyCode.Space) && EstaSuelo())
    {
        rigidBody.AddForce(Vector2.up * salto, ForceMode2D.Impulse);
    }
}

1 referencia
void ProcesarMovimiento()
{
    //puro movimiento
    float inputMovimiento = Input.GetAxis("Horizontal");


    rigidBody.velocity = new Vector2(inputMovimiento * velocidad, rigidBody.velocity.y);

    GestionarOrientacion(inputMovimiento);

    if(inputMovimiento != 0f) //Para saber cuando el personaje esta quieto o moviendose y ejecutar la animacion
    {
        animator.SetBool("EstaCorriendo", true);
    }
    else
    {
        animator.SetBool("EstaCorriendo", false);
    }
}
```

Esto fue utilizado para transformar la orientación del personaje

```
1 referencia
void GestionarOrientacion(float inputMovimiento)
{
    //El personaje gira segun la direccion que deseamos
    if ((mirandoDerecha == true && inputMovimiento < 0) || (mirandoDerecha == false && inputMovimiento > 0))
    {
        mirandoDerecha = !mirandoDerecha;
        transform.localScale = new Vector2(-transform.localScale.x, transform.localScale.y);
    }
}
}
```

Este es otro script, el ataque del jugador, aquí definimos otra animación y la el input para atacar, ósea el click izquierdo

```
public class Ataque : MonoBehaviour
{
    [SerializeField] private Transform Golpe;
    [SerializeField] private float radio;
    [SerializeField] private float dañoGolpe;
    private Animator animator;

    Mensaje de Unity | 0 referencias
    private void Start()
    {
        animator = GetComponent<Animator>();
    }
    Mensaje de Unity | 0 referencias
    void Update()
    {
        if (Input.GetButtonDown("Fire1"))
        {
            GolpeC();
        }
    }
    1 referencia
    private void GolpeC()
    {
        animator.SetTrigger("Golpe");
        Collider2D[] objetos = Physics2D.OverlapCircleAll(Golpe.position, radio);
    foreach (Collider2D colisionador in objetos)
        {
            if (colisionador.CompareTag("enemigo"))
            {
                colisionador.transform.GetComponent<Enemigo>().TomarDaño(dañoGolpe);
            }
        }
    }
```

Utilice gizmos para definir el radio de golpe

```
    Mensaje de Unity | 0 referencias
    private void OnDrawGizmos()
    {
        Gizmos.color = Color.red;
        Gizmos.DrawSphere(Golpe.position, radio);
    }

    // Start is called before the first frame update



    // Update is called once per frame

}
```

Este es el script de el enemigo y su controlador, así como su muerte y sus animaciones

```
public class Enemigo : MonoBehaviour
{
    [SerializeField] private float vida;

    private Animator animator;
    public Rigidbody2D rb2d;
    public Transform jugador;

    private bool mirandoderecha = true;

    [Header("ataque")]
    [SerializeField] private Transform controladorAtaq;
    [SerializeField] private float radioAt;
    [SerializeField] private float dañoAt;

    // Start is called before the first frame update
    Mensaje de Unity | 0 referencias
    private void Start()
    {
        animator = GetComponent<Animator>();
        rb2d = GetComponent<Rigidbody2D>();
        jugador = GameObject.FindGameObjectWithTag("player").GetComponent<Transform>();
    }

    1 referencia
    public void TomarDaño(float daño)
    {
        vida -= daño;
        if(vida <= 0)
        {
            Muerte();
        }
    }
    1 referencia
    private void Muerte()
    {
        animator.SetTrigger("Muerte");
        Destroy(gameObject);
```

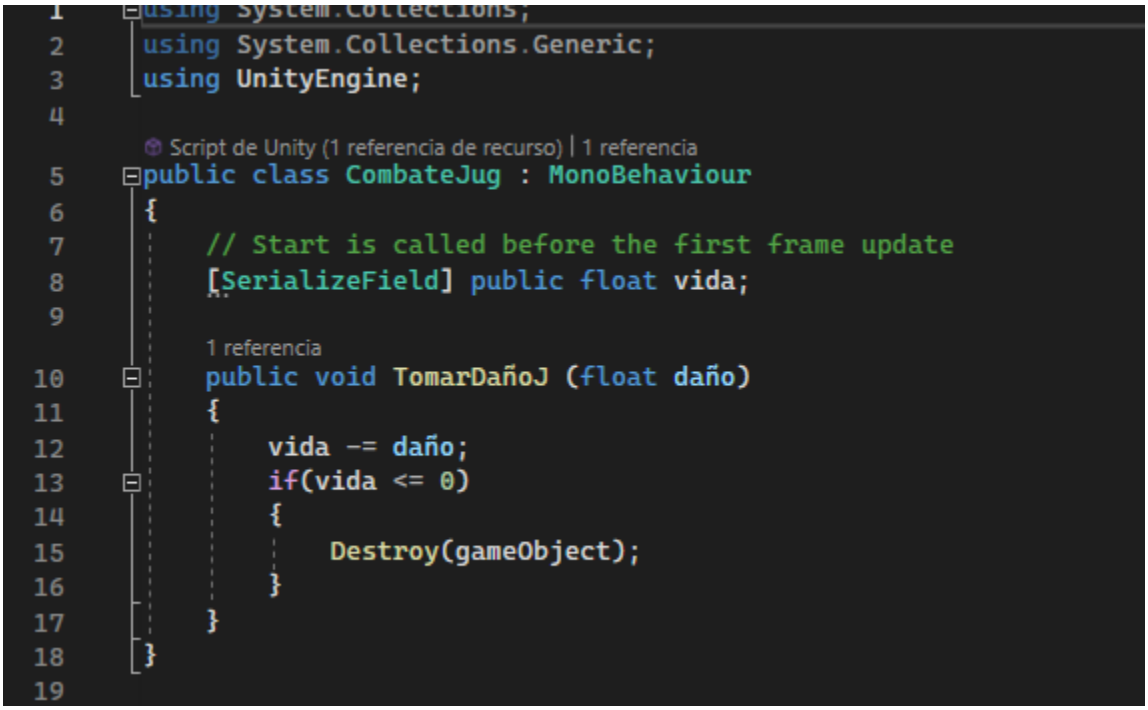
Aquí utilice etiquetas para que el enemigo le pegue al personaje principal, este también tiene un radio de golpe con gizmos

```
1 referencia
public void MirarJugador()
{
    if((jugador.position.x> transform.position.x && !mirandoderecha) || (jugador.position.x < transform.position.x && mirandoderecha))
    {
        mirandoderecha = !mirandoderecha;
        transform.eulerAngles = new Vector3(0, transform.eulerAngles.y + 180, 0);
    }
}

Mensaje de Unity | 0 referencias
private void Update()
{
    float distanciajug = Vector2.Distance(transform.position, jugador.position);
    animator.SetFloat("distanciajug", distanciajug);
}

0 referencias
private void Ataque()
{
    Collider2D[] objetos = Physics2D.OverlapCircleAll(controladorAtaq.position, radioAt);

    foreach(Collider2D collision in objetos)
    {
        if (collision.CompareTag("player"))
        {
            collision.GetComponent<CombateJug>().TomarDañoJ(dañoAt);
        }
    }
}
Mensaje de Unity | 0 referencias
private void OnDrawGizmos()
{
    Gizmos.color = Color.red;
    Gizmos.DrawWireSphere(controladorAtaq.position, radioAt);
}
```

Este script lo hice desde el animator, para animaciones del enemigo, aqui solo quite algunas diagonales y agregue mi código

```
1   using System.Collections;
2   using System.Collections.Generic;
3   using UnityEngine;
4
        Script de Unity | 0 referencias
5   public class EnemigoCaminando : StateMachineBehaviour
6   {
7       private Enemigo enemigo;
8       private Rigidbody2D rb2d;
9
10      [SerializeField] private float VelMov;
11      // OnStateEnter is called when a transition starts and the state machine starts to evaluate this state
            Mensaje de Unity | 0 referencias
12      override public void OnStateEnter(Animator animator, AnimatorStateInfo stateInfo, int layerIndex)
13      {
14          enemigo = animator.GetComponent<Enemigo>();
15          rb2d = enemigo.rb2d;
16
17          enemigo.MirarJugador();
18      }
19
20      // OnStateUpdate is called on each Update frame between OnStateEnter and OnStateExit callbacks
            Mensaje de Unity | 0 referencias
21      override public void OnStateUpdate(Animator animator, AnimatorStateInfo stateInfo, int layerIndex)
22      {
23          rb2d.velocity = new Vector2(VelMov, rb2d.velocity.y) * animator.transform.right;
24      }
25
26      // OnStateExit is called when a transition ends and the state machine finishes evaluating this state
            Mensaje de Unity | 0 referencias
27      override public void OnStateExit(Animator animator, AnimatorStateInfo stateInfo, int layerIndex)
28      {
29          rb2d.velocity = new Vector2(0, rb2d.velocity.y);
30      }
31
32      // OnStateMove is called right after Animator.OnAnimatorMove()
33      //override public void OnStateMove(Animator animator, AnimatorStateInfo stateInfo, int layerIndex)
34      //{
35      //    // Implement code that processes and affects root motion
36      //}
```

Vida de mi personaje principal, simple pero funciona



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

Script de Unity (1 referencia de recurso) | 1 referencia
public class CombateJug : MonoBehaviour
{
    // Start is called before the first frame update
    [SerializeField] public float vida;

    1 referencia
    public void TomarDañoJ (float daño)
    {
        vida -= daño;
        if(vida <= 0)
        {
            Destroy(gameObject);
        }
    }
}
```

Controles

Los controles son las flechas de movimiento, la barra espaciadora para saltar, clic izquierdo para golpear

Es mi primer sistema de combate que hago, espero que les guste :D