**Player script**

**Player movement.cs**

using System.Collections;

using System.Collections.Generic;

using System;

using UnityEngine;

using UnityEngineInternal;

public class Playermovement : MonoBehaviour

{

[SerializeField] float speed = 1f;

[SerializeField] float jump\_heigh = 1f;

[SerializeField] float Attack2Cooldown = 0;

private Rigidbody2D player;

private Animator player\_animation;

bool Player\_isGround;

bool Fall\_detect = false;

bool Double\_jump = false;

private float speedIn = 0f;

private float attack2CooldownTimer = Mathf.Infinity;

private void Awake()

{

player = GetComponent<Rigidbody2D>();

player\_animation = GetComponent<Animator>();

}

// Update is called once per frame

void Update()

{

if (Input.GetKey(KeyCode.LeftShift))

{

speedIn = speed \* 2f;

}

else speedIn = speed;

float HorizontalInput = Input.GetAxis("Horizontal");

player.velocity = new Vector2(HorizontalInput \* speedIn, player.velocity.y);

if (HorizontalInput > 0f ) transform.localScale = new Vector3(3.5f, 3.5f, 3.5f);

if (HorizontalInput < 0f ) transform.localScale = new Vector3(-3.5f, 3.5f, 3.5f);

if (Input.GetKeyDown(KeyCode.Space) && (Player\_isGround ^ Double\_jump))

{

player.velocity = new Vector2(player.velocity.x, jump\_heigh);

player\_animation.SetTrigger("Jump");

Player\_isGround = false;

Double\_jump = !Double\_jump;

}

if (player.velocity.y < -2f)

{

Fall\_detect = true;

Player\_isGround = false;

}

else Fall\_detect = false;

player\_animation.SetBool("Player\_run\_check", HorizontalInput != 0f);

player\_animation.SetBool("Player\_isGrounded", Player\_isGround);

player\_animation.SetBool("Fall", Fall\_detect);

//attack movement

attack2CooldownTimer += Time.deltaTime;

if (Input.GetKeyDown(KeyCode.Mouse0))

{

player\_animation.SetBool("Attack1", true);

}

else player\_animation.SetBool("Attack1", false);

if (Input.GetKeyDown(KeyCode.Mouse1))

{

if (attack2CooldownTimer >= Attack2Cooldown)

{

player\_animation.SetBool("Attack2", true);

attack2CooldownTimer = 0;

}

}

else player\_animation.SetBool("Attack2", false);

}

private void OnCollisionEnter2D(Collision2D collision)

{

if (collision.gameObject.tag == "Ground\_map")

{

Player\_isGround = true;

Double\_jump = false;

}

}

}

Đoạn mã này xử lý chuyển động, nhảy và hoạt ảnh tấn công của người chơi trong một trò chơi 2D Unity. Người chơi có thể di chuyển sang trái hoặc phải, nhảy (với tính năng nhảy kép), và thực hiện hai loại tấn công.

Biến thành viên:

* float speed: Tốc độ di chuyển của người chơi.
* float jump\_heigh: Chiều cao của cú nhảy.
* float Attack2Cooldown: Thời gian hồi chiêu giữa các lần thực hiện Attack2.
* Rigidbody2D player: Thành phần Rigidbody2D gắn với người chơi, được sử dụng cho các tính toán vật lý.
* Animator player\_animation: Thành phần Animator dùng để điều khiển hoạt ảnh của người chơi.
* bool Player\_isGround: Biến xác định liệu người chơi có đang đứng trên mặt đất hay không.
* bool Fall\_detect: Biến xác định liệu người chơi có đang rơi hay không.
* bool Double\_jump: Biến xác định liệu người chơi có thể thực hiện nhảy kép hay không.
* float speedIn: Tốc độ hiện tại của người chơi, có thể thay đổi khi nhấn phím Shift.
* float attack2CooldownTimer: Bộ đếm thời gian cho lần tấn công thứ hai, khởi đầu bằng giá trị vô cực.

**Awake():** Phương thức này được gọi khi đối tượng được khởi tạo. Nó lấy các thành phần Rigidbody2D và Animator từ đối tượng người chơi.

**Update():** Phương thức này được gọi mỗi khung hình. Nó xử lý các hành động và trạng thái của người chơi như di chuyển, nhảy và tấn công.

**OnCollisionEnter2D(Collision2D collision):** Phương thức này được gọi khi người chơi va chạm với đối tượng khác. Nó xác định liệu người chơi có đang trên mặt đất hay không dựa trên thẻ của đối tượng va chạm. Phần này đảm bảo rằng người chơi có thể nhảy lại khi tiếp đất và đặt lại biến **Double\_jump** để cho phép nhảy kép trong lần tiếp theo.

**Player\_attack.cs**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class Player\_attack : MonoBehaviour

{

// Start is called before the first frame update

[SerializeField] private int Damage1;

[SerializeField] private int Damage2;

[SerializeField] private Collider2D SwordCollider;

private int DamageToEnemy;

void Start()

{

}

// Update is called once per frame

void Update()

{

if (Input.GetKeyDown(KeyCode.Mouse0)) DamageToEnemy = Damage1;

if (Input.GetKeyDown(KeyCode.Mouse1)) DamageToEnemy = Damage2;

}

private void OnTriggerEnter2D(Collider2D collision)

{

if (collision.tag == "Enemy")

{

Debug.Log(collision.tag + " " + collision.GetComponent<Enemy\_Health>().currentHealth);

collision.GetComponent<Enemy\_Health>().TakeDamage(DamageToEnemy);

//if (Input.GetKeyDown(KeyCode.Mouse1)) collision.GetComponent<Enemy\_Health>().TakeDamage(Damage2);

}

}

}

Đoạn mã này xử lý các hành động tấn công của người chơi trong một trò chơi 2D Unity. Người chơi có thể tấn công bằng hai loại đòn khác nhau, gây sát thương lên kẻ địch khi va chạm với chúng.

Biến thành viên:

* int Damage1: Sát thương của đòn tấn công thứ nhất (nhấn chuột trái).
* int Damage2: Sát thương của đòn tấn công thứ hai (nhấn chuột phải).
* Collider2D SwordCollider: Bộ va chạm của vũ khí, được sử dụng để phát hiện va chạm với kẻ địch.
* int DamageToEnemy: Sát thương hiện tại sẽ gây ra cho kẻ địch, thay đổi dựa trên đòn tấn công được sử dụng.

**Start():** Phương thức này được gọi khi đối tượng được khởi tạo. Trong đoạn mã này, phương thức Start không thực hiện bất kỳ hành động nào.

**Update():** Phương thức này được gọi mỗi khung hình. Nó kiểm tra các phím bấm để xác định loại đòn tấn công và cập nhật giá trị sát thương tương ứng.

* Nếu nhấn phím Mouse0 (chuột trái), cập nhật sát thương hiện tại là Damage1.
* Nếu nhấn phím Mouse1 (chuột phải), cập nhật sát thương hiện tại là Damage2.

**OnTriggerEnter2D(Collider2D collision):** Phương thức này được gọi khi bộ va chạm của vũ khí va chạm với một đối tượng khác. Nó kiểm tra nếu đối tượng va chạm có thẻ "Enemy" và gây sát thương cho kẻ địch.Nếu đối tượng va chạm có thẻ là "Enemy", thực hiện các hành động sau:

* In ra thẻ của đối tượng và máu hiện tại của kẻ địch.
* Gọi phương thức TakeDamage của đối tượng Enemy\_Health để gây sát thương cho kẻ địch với giá trị DamageToEnemy.

Phần này đảm bảo rằng khi vũ khí của người chơi va chạm với kẻ địch, kẻ địch sẽ nhận sát thương tương ứng với loại đòn tấn công được sử dụng.

**Health.cs**

using System;

using UnityEngine;

public class Health : MonoBehaviour

{

[SerializeField] public float startingHealth;

public bool dead = false;

public float currentHealth { get; private set; }

private Animator anim;

private void Awake()

{

currentHealth = startingHealth;

anim = GetComponent<Animator>();

}

public void TakeDamage(int damage)

{

currentHealth -= damage;

if (currentHealth > startingHealth) currentHealth = startingHealth;

if (currentHealth < 0) currentHealth = 0;

if (currentHealth > 0)

{

// take hurt

anim.SetTrigger("Hurt");

}

else

{

// go dead

anim.SetTrigger("Die");

GetComponent<Playermovement>().enabled = false;

dead = true;

}

}

public void AddHealth(int health)

{

currentHealth += health;

if (currentHealth > startingHealth) currentHealth = startingHealth;

if (currentHealth < 0) currentHealth = 0;

}

private void Update()

{

if (Input.GetKeyDown(KeyCode.H)) { currentHealth = startingHealth; }

}

}

Đoạn mã này xử lý máu của một đối tượng trong trò chơi Unity. Nó kiểm tra máu, cho phép đối tượng nhận sát thương, hồi phục máu và xử lý các trạng thái như "hurt" và "die".

* float startingHealth: Máu ban đầu của đối tượng.
* bool dead: Biến xác định liệu đối tượng đã chết hay chưa.
* float currentHealth: Máu hiện tại của đối tượng, chỉ có thể được truy cập từ bên ngoài.
* Animator anim: Thành phần Animator dùng để điều khiển hoạt ảnh của đối tượng.

**Awake():**Phương thức này được gọi khi đối tượng được khởi tạo. Nó thiết lập máu hiện tại bằng máu ban đầu và lấy thành phần Animator từ đối tượng.

**TakeDamage(int damage):** Phương thức này được gọi khi đối tượng nhận sát thương. Nó giảm máu hiện tại theo giá trị sát thương, sau đó cập nhật hoạt ảnh và trạng thái của đối tượng.

**AddHealth(int health):** Phương thức này được gọi khi đối tượng nhận thêm máu. Nó tăng máu hiện tại theo giá trị máu được thêm vào.

Phần này đảm bảo rằng đối tượng có thể nhận sát thương, hồi phục máu, và kích hoạt các hoạt ảnh và trạng thái tương ứng dựa trên máu hiện tại.