**các hàm đã học đc trong game :**

**private void OnTriggerEnter2D(Collider2D collision){**

**if(collision.gameObject.name =="dog"){**

**collision.gameObject.transform.SetParent(transform);**

**}**

**}**

**private void OnTriggerExit2D(Collider2D collision){**

**if(collision.gameObject.name =="dog"){**

**collision.gameObject.transform.SetParent(null);**

**}}**

* private: Từ khóa private chỉ ra rằng phương thức này chỉ có thể truy cập từ bên trong lớp chứa nó.
* void: Kiểu trả về của phương thức là void, có nghĩa là phương thức này không trả về bất kỳ giá trị nào.
* OnTriggerEnter2D: Tên của phương thức, cho biết phương thức này được gọi khi một Collider2D khác va chạm vào Collider2D của đối tượng hiện tại.
* OnTriggerExit2D: Tên của phương thức, cho biết phương thức này được gọi khi một Collider 2D khác rời khỏi Collider2D của đối tượng hiện tại.
* Collider2D đại diện cho một hình dạng không gian hai chiều (ví dụ: hình chữ nhật, hình tròn, v.v.) và được sử dụng để xác định vùng va chạm của đối tượng. Nó có thể được sử dụng để phát hiện va chạm với các Collider2D khác, như Collider2D của đối tượng khác hoặc Collider2D của cùng một đối tượng.
* tên biến bất kì .gameObject: Đây là đối tượng (GameObject) mà va chạm đã xảy ra. Biến “tên bất kì “ là tham chiếu đến Collider2D của đối tượng va chạm, và gameObject là tham chiếu đến GameObject chứa Collider2D này.
* transform: Đây là thành phần Transform của đối tượng hiện tại, tức là Transform của đối tượng mà script này được gắn vào. Transform chứa thông tin về vị trí, quy mô và góc quay của đối tượng.
* SetParent(null): Đây là một phương thức của Transform, được gọi để đặt cha của đối tượng va chạm là null, tức là không có cha (parent). Khi phương thức này được gọi, đối tượng va chạm sẽ không còn là con của bất kỳ đối tượng nào và nó sẽ trở thành một đối tượng gốc (root object) trong cấu trúc hệ thống GameObject.

[SerializeField]private AudioSource finishsound;

void Start()

{

finishlevel =GetComponent<AudioSource>();

}

// Update is called once per frame

private void OnTriggerEnter2D(Collider2D collision)

{

if (collision.gameObject.name == "dog")

{

finishsound.Play();

Invoke("Completelevel",2f);

Completelevel();

}

}

private void Completelevel(){

SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex+1);

}

* [SerializeField]: Đây là một thuộc tính (attribute) đặc biệt trong Unity được sử dụng để hiển thị và tuỳ chỉnh các biến riêng tư trong trình chỉnh sửa Inspector của Unity. Khi một biến được đánh dấu [SerializeField], nó sẽ xuất hiện trong Inspector và cho phép bạn gán giá trị cho biến từ giao diện người dùng của Unity mà không cần phải làm biến này công khai (public).
* AudioSource: Đây là kiểu dữ liệu của biến. Trong Unity, AudioSource là một thành phần cho phép phát âm thanh và điều khiển các thuộc tính liên quan đến âm thanh như âm lượng, bản lặp, và độ trễ.
* Phương thức Start() là một phương thức được gọi tự động khi đối tượng chứa kịch bản (script) bắt đầu chạy trong Unity.
* Biến T.GetComponent<T> t là component của biến . gán t là component đó
* T.Play(); : Phát âm thanh của biến T nếu đã chọn component là Audio Sound
* Invoke("tên phương thức l", x);: Gọi phương thức trong “”l sau x giây.số thực phải thêm f đằng sau
* “tên phương thức “(); gọi phương thức bên ngoài vào trong hàm chứa lệnh này
* SceneManager là một lớp trong Unity cho phép quản lý các cảnh (scenes) trong trò chơi.
* LoadScene là một phương thức của lớp SceneManager dùng để tải một cảnh mới.
* SceneManager.GetActiveScene() là một phương thức khác của SceneManager dùng để lấy thông tin về cảnh hiện tại đang chạy.
* buildIndex là một thuộc tính của đối tượng cảnh (Scene) và trả về chỉ số xây dựng (build index) của cảnh hiện tại.

public class NewBehaviourScript : MonoBehaviour

{

[SerializeField] private LayerMask jumpable;

// Khai báo thuộc tính

public Rigidbody2D rb;

public bool duocphepnhay = false;

public float dirX = 0;

public Animator anim;

public SpriteRenderer sprite;

[SerializeField] public AudioSource jumpeffect;

// Start is called before the first frame update

void Start()

{

// Thay tên thuộc tính

rb = GetComponent<Rigidbody2D>();

anim = GetComponent<Animator>();

sprite = GetComponent<SpriteRenderer>();

}

// Update is called once per frame

void Update()

{

dirX = Input.GetAxis("Horizontal");

rb.velocity = new Vector2(dirX \* 7f, rb.velocity.y);

if (Input.GetButtonDown("Jump") && duocphepnhay)

{

jumpeffect.Play();

rb.velocity = new Vector2(rb.velocity.x, 8);

duocphepnhay = false;

anim.SetTrigger("jump");

}

if (dirX > 0f)

{

anim.SetBool("running", true);

sprite.flipX = false;

}

else if (dirX < 0f)

{

anim.SetBool("running", true);

sprite.flipX = true;

}

else

{

anim.SetBool("running", false);

}

if (!duocphepnhay && rb.velocity.y <= 0)

{

duocphepnhay = true;

}

if(rb.velocity.y > 0 )

{

duocphepnhay =false;

}

}

* Animator (người điều khiển): Là thành phần trong Unity dùng để điều khiển và quản lý trạng thái và chuyển động của đối tượng trong trò chơi.
* Animation (hoạt hình): Là dữ liệu hoặc tệp tin chứa chuỗi các khung hình hoặc dữ liệu hoạt hình để tạo ra sự chuyển động của đối tượng.
* SpriteRenderer là một component trong Unity được sử dụng để hiển thị các hình ảnh (sprite) trên một đối tượng trong trò chơi hoặc ứng dụng. Nó được gắn kết với một đối tượng và quản lý việc hiển thị sprite cho đối tượng đó.
* Input.GetAxis("Horizontal") trong Unity. Đoạn mã này trả về giá trị của trục ngang (horizontal) từ các thiết bị đầu vào như bàn phím hoặc điều khiển. Giá trị này thay đổi từ -1 đến 1, tương ứng với hướng trái và phải.
* biến.velocity là một thuộc tính của thành phần Rigidbody trong Unity, nó đại diện cho vận tốc hiện tại của đối tượng trong không gian 2D hoặc 3D.
* Trong không gian 2D, velocity.x đại diện cho thành phần vận tốc theo trục x (chiều ngang) của đối tượng, còn velocity.y đại diện cho thành phần vận tốc theo trục y (chiều dọc) của đối tượng.
* new Vector2(x,y); nếu di chuyển trái phải thì x là số vận tốc , y ko thay đổi dùng y.velocity
* SetTrigger được sử dụng để kích hoạt một trạng thái hoạt hình đặc biệt, thường được sử dụng cho các trạng thái không liên tục như nhảy, tấn công, hoặc hiệu ứng đặc biệt.Để kích hoạt một trigger, bạn gọi phương thức SetTrigger trên Animator và truyền vào tên của trigger như một cụm ký tự. Ví dụ: anim.SetTrigger("jump");anim phải là 1 component animator
* SetBool được sử dụng để đặt giá trị của một biến bool trong Animator Controller. Điều này thường được sử dụng cho các trạng thái liên tục như chạy, đứng yên, đang tấn công.Để đặt giá trị của một biến bool, bạn gọi phương thức SetBool trên Animator và truyền vào tên của biến bool cùng với giá trị bool tương ứng. Ví dụ: anim.SetBool("running", true);anim phải là 1 component animator
* 1 biến SpriteRenderer muốn lật mặt dùng tên biến .flip(X hoặc y)

private int strawberries = 0;

[SerializeField] private AudioSource tingeffect;

private void OnTriggerEnter2D(Collider2D collision)

{

if (collision.gameObject.CompareTag("straw"))

{

tingeffect.Play();

strawberries++;

Destroy(collision.gameObject);

Debug.Log("straw: " + strawberries);

}

}

* CompareTag("abc") là một phương thức của đối tượng GameObject trong Unity. Nó kiểm tra xem tag của đối tượng GameObject có trùng khớp với chuỗi "abc" hay không. Nếu trùng khớp, phương thức này trả về true, ngược lại, nó trả về false.
* debug.log(“thông điệp) nếu mã nguồn đến mã này ko lỗi . thông điệp sẽ chạy trên console
* Destroy (tên biến .gameObject); nếu có lớp cha là object còn đâu ko cần

public void Quit(){

Application.Quit();

}

* public void Quit(){ Application.Quit(); } hàm thoát game thực

[SerializeField] private GameObject[] waypoints;

private int currentwaypoint=0;

[SerializeField] private float speed =2f ;

void Update()

{

if(Vector2.Distance(waypoints[currentwaypoint].transform.position,transform.position)<.1f){

currentwaypoint ++;

if(currentwaypoint >=waypoints.Length){

currentwaypoint =0;

}

} transform.position = Vector2.MoveTowards(transform.position,waypoints[currentwaypoint]. transform.position,Time.deltaTime\*speed);

}

* [SerializeField] private GameObject[] : gọi trường object
* Distance là một phương thức trong lớp Vector2 trong Unity. Nó được sử dụng để tính toán khoảng cách giữa hai điểm trong không gian hai chiều.
* position là một thuộc tính của thành phần Transform và đại diện cho vị trí không gian hai chiều của đối tượng. suy ra transform.position: Đây là vị trí hiện tại của đối tượng, nơi đối tượng đang đứng.
* Time.deltaTime: Đây là thời gian đã trôi qua kể từ khung hình trước đó. Nó được sử dụng để tính toán di chuyển dựa trên thời gian thực của trò chơi, đảm bảo rằng tốc độ di chuyển của đối tượng không bị ảnh hưởng bởi tốc độ khung hình.