mã code đã học tìm hiểu trong Unity

1.Public , private <kiểu dữ liệu > <tên biến > ;

khai báo biến và hiển thị trên inspector

ví dụ :

```
public int score;
public float speed;
private string playerName;
```

2.[Range(xf,yf)]

hiển thị phạm vi từ x đến y đơn vị trên inspector;

3.Public , private <component> <biến component> ;

khai báo biến component

khởi tạo

tên biến = GetComponent<tên của component > (); .

bạn có thể đặt khối GetComponent trong hàm Awake() nếu bạn cần truy cập biến component từ đầu quá trình và muốn đảm bảo rằng nó đã được khởi tạo. Hoặc bạn có thể đặt nó trong hàm Start() nếu bạn chỉ cần truy cập biến component trong các phương thức sau này và không cần sử dụng nó trong Awake().

ví dụ

```
public Rigidbody rigidbody;
public Collider collider;
public AudioSource audioSource;
private void Start()
{
   rigidbody = GetComponent<Rigidbody>();
   collider = GetComponent<Collider>();
   audioSource = GetComponent<AudioSource>();
}
```

4. khai báo biến tham chiếu đến class script khác

public <tên lớp> <tên biến>;

từ biến đó mã đang viết có thể tham chiếu đến phương thức đó

```
private GameController m_gc;

private void Start()
{
   m_gc = GetComponent<GameController>();
   m_gc.StartGame(); // Gọi phương thức StartGame() từ lớp GameController
   int current Score = m_gc.score; // Truy cập thuộc tính score từ lớp GameController
}
```

5. lấy đầu vào

tạo 1 biến và áp dụng đoạn code

tên biến = Input.GetAxis/GetAxisRaw (‘Horizontal”/”Vertical”);

GetAxis : lấy giá trị đầu vào từ bàn phím -1<=x<=1

GetAxisRaw : lấy giá trị đầu vào từ bàn phím -1,0,1;

Horizontal đại diện cho trục ngang hàm sẽ trả về giá trị -1, 0 hoặc 1, tương ứng với hướng di chuyển ngang sang trái, không di chuyển hoặc di chuyển ngang sang phải.
Vertical đại diện cho trục dọc hàm sẽ trả về giá trị -1, 0 hoặc 1, tương ứng với hướng di chuyển dọc lên, không di chuyển hoặc di chuyển dọc xuống.

6.Time.deltatime

Time.deltaTime là một giá trị thời gian được sử dụng để đồng bộ hóa các hành động trong trò chơi với tốc độ khung hình (frame rate). Nó đại diện cho thời gian đã trôi qua kể từ khung hình trước đó.

7. 2 cách truy xuất vị trí/góc quay của vật

1. khi ko có biến rigidbody2D

dùng transform.(position/rotation). (x,y,z) = … f.

ví dụ Vector3 objectPosition = transform.position.(x or y or z);
Quaternion objectRotation = transform.rotation(x,y,z);

2. khi có biến rigidbody2d

dùng tên biến . (position/rotation) (x,y,z) = … f.

ví dụ : Vector2 Object Position = rb..position.(x or y or z);
float objectRotation = rigidbody2D.rotation();

8. 3 cách di chuyển

a/ dùng transform.position= biến có độ chênh lệch
ví dụ : transform.position += new Vector3(moveSpeed * Time.deltaTime, 0f, 0f);

b/ Sử dụng transform.Translate(): Bạn có thể sử dụng phương thức Translate() của thành phần Transform để dịch chuyển đối tượng theo một vector di chuyển. Ví dụ:
transform.Translate(Vector3.forward * moveSpeed * Time.deltaTime);

c/ Sử dụng rigidbody.velocity: Khi sử dụng thành phần Rigidbody, bạn có thể sử dụng thuộc tính velocity để thiết lập vector vận tốc của đối tượng. Ví dụ:
rigidbody.velocity = new Vector3(moveSpeed, 0f, 0f);
rb.velocity = new Vector2(movestep, 0);

10.tương tác trên pc để thực hiện lệnh

bàn phím

```
void Update()
{
   if (Input.GetKeyDown(KeyCode.Space))
   {
      // Code để xử lý khi phím Space được nhấn xuống
      // Ví dụ: nhảy, bắn đạn, chuyển đổi trạng thái, vv.
   }
}
```

Input.GetKeyDown(KeyCode.Space) là một hàm dùng để kiểm tra xem phím Space (phím cách) có được nhấn xuống (pressed down) trong khung cảnh hiện tại hay không. Hàm này trả về giá trị true nếu phím Space được nhấn xuống và false nếu không.
tương tự
Input.GetKey: Phương thức này kiểm tra xem một phím cụ thể được giữ (được nhấn và giữ) trong một khung hình cụ thể hay không. Nó trả về true trong trường hợp phím đang được giữ và false nếu không.

Input.GetKeyUp: Phương thức này kiểm tra xem một phím cụ thể đã được nhả ra (chỉ nhả ra trong một khung hình duy nhất) trong khung hình hiện tại hay không. Nó trả về true trong trường hợp phím đã được nhả ra và false nếu không.

```
if (Input.GetKeyDown(KeyCode.W))
{
   // Code để xử lý khi phím W được nhấn xuống
}

if (Input.GetKeyDown(KeyCode.Mouse0))
{
   // Code để xử lý khi nút chuột trái được nhấn xuống chuột trái mouse0 phải mouse1
}

if (Input.GetKeyDown(KeyCode.Escape))
{
   // Code để xử lý khi phím Escape được nhấn xuống
}
if (Input.GetKeyDown(KeyCode.Alpha1))
{
   // Code để xử lý khi số 1 được nhấn xuống
}
```

chuột

```
void Update()
{
   if (Input.GetMouseButtonDown(0))
   {
      // Code để xử lý khi nút chuột trái được nhấn xuống
      // Ví dụ: bắn đạn, tương tác với đối tượng, vv.
   }
}
void Update()
{
   if (Input.GetMouseButtonDown(1))
   {
      // Code để xử lý khi nút chuột phải được nhấn xuống
      // Ví dụ: hiển thị menu ngữ cảnh, thực hiện hành động đặc biệt, vv.
   }}
```

11.[SerializeField]

là một thuộc tính (attribute) trong Unity được sử dụng để xác định rằng một trường (field) trong một lớp (class) sẽ được lưu trữ và hiển thị trong Inspector của Unity.
[SerializeField] trong Unity cho phép bạn tham chiếu và kéo thả các đối tượng cùng loại đã khai báo trong Inspector.
Khi một trường được đánh dấu bằng [SerializeField], nó sẽ được hiển thị trong Inspector của Unity dù trường đó là private, protected hay public

12. Destroy

Destroy là một phương thức được sử dụng để hủy bỏ, xóa bỏ một đối tượng trong scene. Khi một đối tượng được hủy bỏ bằng cách sử dụng phương thức Destroy, nó sẽ bị loại bỏ khỏi scene và các tài nguyên được sử dụng bởi đối tượng đó sẽ được giải phóng. Điều này cho phép giải phóng bộ nhớ và các tài nguyên không cần thiết.
Destroy(object/biến.gameobject,thời gian trễ);
ví dụ :
Destroy(gameObject, 3.0f);
Destroy(component, 5.0f);

13. Invoke
gọi phương thức sau x giây;
Invoke (“tên phương thức “ , xf giây);

14.tag

sử dụng để xác định va chạm . đặt tên khác cho tag khi va chạm tag đó sẽ hiện lệnh

15.audio source - âm thanh cơ bản của trò chơi

[SerializeField] audio Source tên biến
tạo component chọn bài hát và kéo thả audio source vào c#script
khi cần âm thanh tại 1 đoạn mã logic nào đó dùng tên biến . play();

16.Debug.Log(“lời nhắn”);

Như console.WriteLine c# bth .

17 . return ;

kết thúc câu lệnh và ko làm gì kể cả void

18. nếu biến ko phải kiểu bool thì điều kiện chỉ cần tên để xác định có null hay ko

public GameObject dan;
public Transform shootingPoint;
if (dan && shootingPoint)

19.tạo mới đối tg

Instantiate() là một phương thức trong Unity được sử dụng để tạo ra một thể hiện mới của một đối tượng (Prefab).
GameObject newObject = Instantiate(prefabObject, position, rotation);
Trong đó:

prefab Object là một biến kiểu GameObject đại diện cho đối tượng Prefab mà bạn muốn tạo thể hiện mới từ nó.

position là một biến kiểu Vector3 hoặc Vector2 đại diện cho vị trí trong không gian 3D hoặc 2D mà bạn muốn đặt thể hiện mới tại đó.
rotation là một biến kiểu Quaternion đại diện cho hướng quay (rotation) của thể hiện mới. Bạn có thể sử dụng Quaternion.identity nếu bạn muốn hướng quay mặc định (không xoay).
ví dụ :GameObject newDiabay = Instantiate(diabay, spawnPos, Quaternion.identity);
ngoài ra nếu đổi hướng quay thì Quaternion rotation = Quaternion.Euler(0, 0, 270);
GameObject newshoot = Instantiate(dan, shootingPoint.position, rotation);

20.random
Vector2 randomVector = new Vector2(Random.Range(minX, maxX), Random.Range(minY, maxY));
Trong đó:

Random.Range(minX, maxX) tạo một giá trị ngẫu nhiên nằm trong khoảng từ min X đến max x. Đây là một hàm trong Unity để tạo giá trị ngẫu nhiên.
Random.Range(minY, maxY) tạo một giá trị ngẫu nhiên nằm trong khoảng từ min Y đến max Y. Đây cũng là một hàm trong Unity để tạo giá trị ngẫu nhiên.

21. hàm va chạm // va chạm ko xuyên và xuyên
Phương thức OnCollisionEnter được gọi khi hai Collider bắt đầu va chạm với nhau. Khi hai Collider chạm nhau, sự kiện OnCollisionEnter sẽ được gọi một lần duy nhất và chỉ trong frame đầu tiên của va chạm. Phương thức này thường được sử dụng để xử lý các tác động ban đầu khi va chạm xảy ra, ví dụ như âm thanh, hiệu ứng hình ảnh, hoặc gọi các hàm để thay đổi trạng thái của các đối tượng.

Ví dụ về sử dụng phương thức OnCollisionEnter2D trong Unity:

```
void OnCollisionEnter2D(Collision2D collision)
{
    if (collision.gameObject.tag == "Player")
    {
        // Xử lý va chạm với đối tượng "Player"
    }
}
```
Phương thức OnCollisionExit2D được gọi khi hai Collider2D không còn tiếp xúc với nhau sau một lần va chạm ban đầu. Khi hai Collider không còn tiếp xúc, sự kiện OnCollisionExit sẽ được gọi một lần duy nhất. Phương thức này thường được sử dụng để xử lý các tác động khi va chạm kết thúc, ví dụ như dừng âm thanh, hiệu ứng, hoặc gọi các hàm để thiết lập lại trạng thái của đối tượng.

Ví dụ về sử dụng phương thức OnCollisionExit2D trong Unity:

```
void OnCollisionExit2D(Collision2D collision)
{
    if (collision.gameObject.tag == "Player")
    {
        // Xử lý khi đối tượng "Player" không còn tiếp xúc
    }
```

}
Phương thức OnTriggerEnter2D được gọi khi một Collider2D đi qua hoặc bắt đầu tiếp xúc với một Trigger Collider2D. Trigger Collider 2D là một loại Collider2D không tạo ra va chạm vật lý, mà chỉ tạo ra sự kiện va chạm để xử lý logic. Phương thức OnTriggerEnter2D sẽ được gọi khi một Collider2D va chạm với Trigger Collider 2D và chỉ được gọi một lần duy nhất trong frame đầu tiên của va chạm.

Ví dụ về sử dụng phương thức OnTriggerEnter2D trong Unity:

```
void OnTriggerEnter2D(Collider2D other)
{
   if (other.CompareTag("Player"))
   {
      // Xử lý khi va chạm với đối tượng "Player"
   }
}
```

Phương thức OnTriggerExit2D được gọi khi một Collider2D không còn tiếp xúc với Trigger Collider 2D sau một lần va chạm ban đầu. Phương thức này sẽ được gọi một lần duy nhất khi Collider2D không còn tiếp xúc với Trigger Collider2D.

Ví dụ về sử dụng phương thức OnTriggerExit2D trong Unity:

```
void OnTriggerExit2D(Collider2D other)
{
   if (other.CompareTag("Player"))
   {
      // Xử lý khi đối tượng "Player" không còn tiếp xúc
   }
}
```

21.
tên biến = FindObjectOfType<T>(); T là kiểu hoặc lớp của đối tượng bạn muốn tìm kiếm. nếu thấy thì player = đối tg đó
22.
gọi trong 1 phương thứ tự đặt để thoát game Application.Quit();
23.
gọi trong 1 phương thức tự đặt để khi gọi phương thức thì sẽ chuyển đến scene có số thứ tự trong project Build
SceneManager.LoadScene(x);

24.Flip X: Gọi spriteRenderer.Flip X = true/false để lật sprite theo trục X.
Flip Y: Gọi spriteRenderer.Flip Y = true/false để lật sprite theo trục Y.

25. truyền text

tham chiếu text script ;
gọi chuỗi biến.text = “chuỗi” .Tostring();