

Bài 8: Tổng quan về lập trình trong Unity

Giảng viên:

MỤC TIÊU

- Giới thiệu chung
- Ngôn ngữ trong Unity
- Lớp components
- Truy xuất thuộc tính của script
- Các phương thức cơ bản
- Debugging
- Tối ưu hóa mã nguồn

Nội dung

- Ngôn ngữ trong Unity
- JavaScript
 - Syntax của JavaScript trong game
 - Điều khiển GameObject với JavaScript
- C#
 - Syntax của C# trong game
 - Điều khiển GameObject với C#
- So sánh C# và JavaScript
- Lớp components
- Truy xuất thuộc tính của script
- Các phương thức cơ bản
- Debugging
- Tối ưu hóa mã nguồn

Ngôn ngữ

– GameEngine là gì?

- Là phần mềm để xây dựng, lập trình lên game
- Có thể giao tiếp với các phần mềm khác.
- Giúp cho việc tạo game một cách nhanh chóng



Ngôn ngữ

- Unity chạy trên Mono framework
- Unity hỗ trợ 3 ngôn ngữ:
 - JavaScript
 - C#
 - Boo
- Một vài phần mềm viết cho Unity.
 - Notepad, Notepad++ , MS Word
 - MS .Net
 - Công cụ mặc định của Unity
 - UnityScript Editor

Ngôn ngữ - Biên dịch Script

- Các scripts trong "Standard Assets", "Pro Standard Assets" hoặc "Plugins" được biên dịch trước tiên.
- Đến các scripts trong "Standard Assets/Editor", "Pro Standard Assets/Editor" hoặc "Plugins/Editor" được biên dịch tiếp sau đó.
- Đến các scripts trong "Editor" được biên dịch tiếp.
- Các scripts còn lại sẽ được biên dịch

Ngôn ngữ – Biến variable

- Member Variables
 - Cách khai báo biến:

//JavaScript

var memberVariable = 0.0;

//C#

```
public class example : MonoBehaviour {  
    public float memberVariable = 0.0F;  
}
```

//boo

```
class example(MonoBehaviour):
```

```
    public memberVariable as single = 0.0F
```

Ngôn ngữ – Biến variable

- Các biến Private
 - Các biến được khai báo với từ khóa Private

//JavaScript

private var memberVariable = 0.0;

//C#

```
public class example : MonoBehaviour {  
    private float memberVariable = 0.0F;  
}
```

//boo

```
class example(MonoBehaviour):
```

```
    private memberVariable as single = 0.0F
```


Ngôn ngữ – Biến variable

- Biến Global
 - Tạo một biến global bằng cách sử dụng từ khóa **static**.
 - *//JavaScript*

static var globalVariable = 0.0;

//C#

public class example : MonoBehaviour
{
 static float globalVariable
}

//boo

class example(MonoBehaviour)
{
 static globalVariable as float = 0.0f
}

Truy cập từ một class khác trong project:

//javascript
NameFile.globalVariable

//C#
NameClass.globalVariable

//Boo
NameClass.globalVariable

Ngôn ngữ - Event

- Ví dụ: Khi click chuột trái, thì sự kiện "*OnMouseDown*" được gọi.
- Bây giờ ta viết đoạn code:
Debug.Log("I click mouse-left");//Javascript
- Khi đó, sự kiện được một hành động trong Unity gọi khi được xử lý.
- Các sự kiện khác: *OnMouseUp, OnMouseOver, Awake, Start, Update, LastUpdate,...*
- Có rất nhiều các sự kiện sẽ được giới thiệu ở phần tiếp theo.

Ngôn ngữ- JavaScript

- **Phần 1: JavaScript**

- **Biến và các loại biến (variable)**
- Hàm (Function)
- Sự kiện (Event)
- Lớp (Class)
- Điều khiển một GameObject bằng JavaScript

Ngôn ngữ - JavaScript

- Khai báo biến:

var <name>[: Type] [= <value>];

Ví dụ:

var n1;

var n: int = 0;

var value1 = 0.0;

var value1: Number = 0.0;

var name = "String variable";

var name: String = "String variable";

Ngôn ngữ - JavaScript

- Các loại biến:
 - Undefined , Ví dụ: *var n1;*
 - Null , Ví dụ : *var n1 = null;*
 - int , Ví dụ : *var n1: int= 1;*
 - Number , Ví dụ : *var n1: Number = 0.1;*
 - String , Ví dụ : *var n1 = "test String";*
 - Boolean , Ví dụ : *var n1:Boolean =false;//true*
 - Array , Ví dụ : *var myArray = [];*
 - Date , Ví dụ : *var d = new Date(2010,2,1);*

Ngôn ngữ - JavaScript

Math

Ví dụ	Giá trị tra lại rounded to 5 digits	Miêu tả
Math.abs(-2.3)	2.3	Giá trị tuyệt đối : $(x < 0) ? -x : x$
Math.acos(Math.SQRT1_2)	0.78540 rad. = 45°	Arccosine
Math.ceil(1.1)	2	Ceiling: round up to smallest integer = argument
Math.cos(Math.PI/4)	0.70711	Cos
Math.exp(1)	2.7183	Hàm mũ
Math.floor(1.9)	1	Floor: round down to largest integer = argument
Math.max(1, -2)	1	Lớn nhất : $(x > y) ? x : y$
Math.min(1, -2)	-2	Nhỏ nhất : $(x < y) ? x : y$
Math.pow(-3, 2)	9	Lũy thừa (raised to the power of): Math.pow(x, y) gives xy
Math.random()	0.17068	Lấy số ngẫu nhiên giữa 0 (inclusive) và 1 (exclusive)
Math.round(1.5)	2	Làm tròn (e.g. 1.5 rounds to 2)
Math.sin(Math.PI/4)	0.70711	Sine
Math.sqrt(49)	7	Square root
Math.tan(Math.PI/4)	1	Tangent

Ngôn ngữ - JavaScript

- Hàm trong JavaScript

- Khai báo hàm:

```
[private/public/static] function <name_func> ([par1,par2])  
[:<type_resunt>]
```

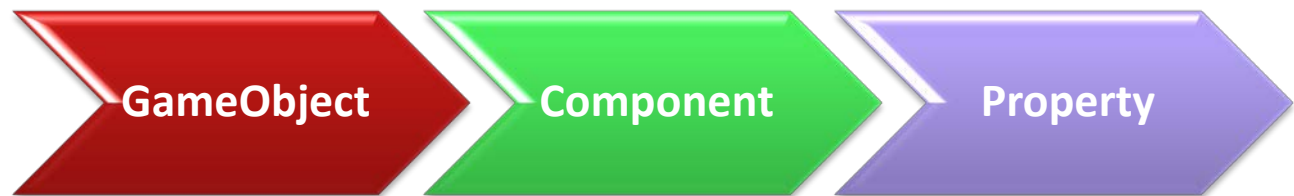
```
{ // bắt đầu vào hàm  
  // Viết code và câu lệnh ở đây  
} // đóng hàm
```

- Ví dụ về hàm:

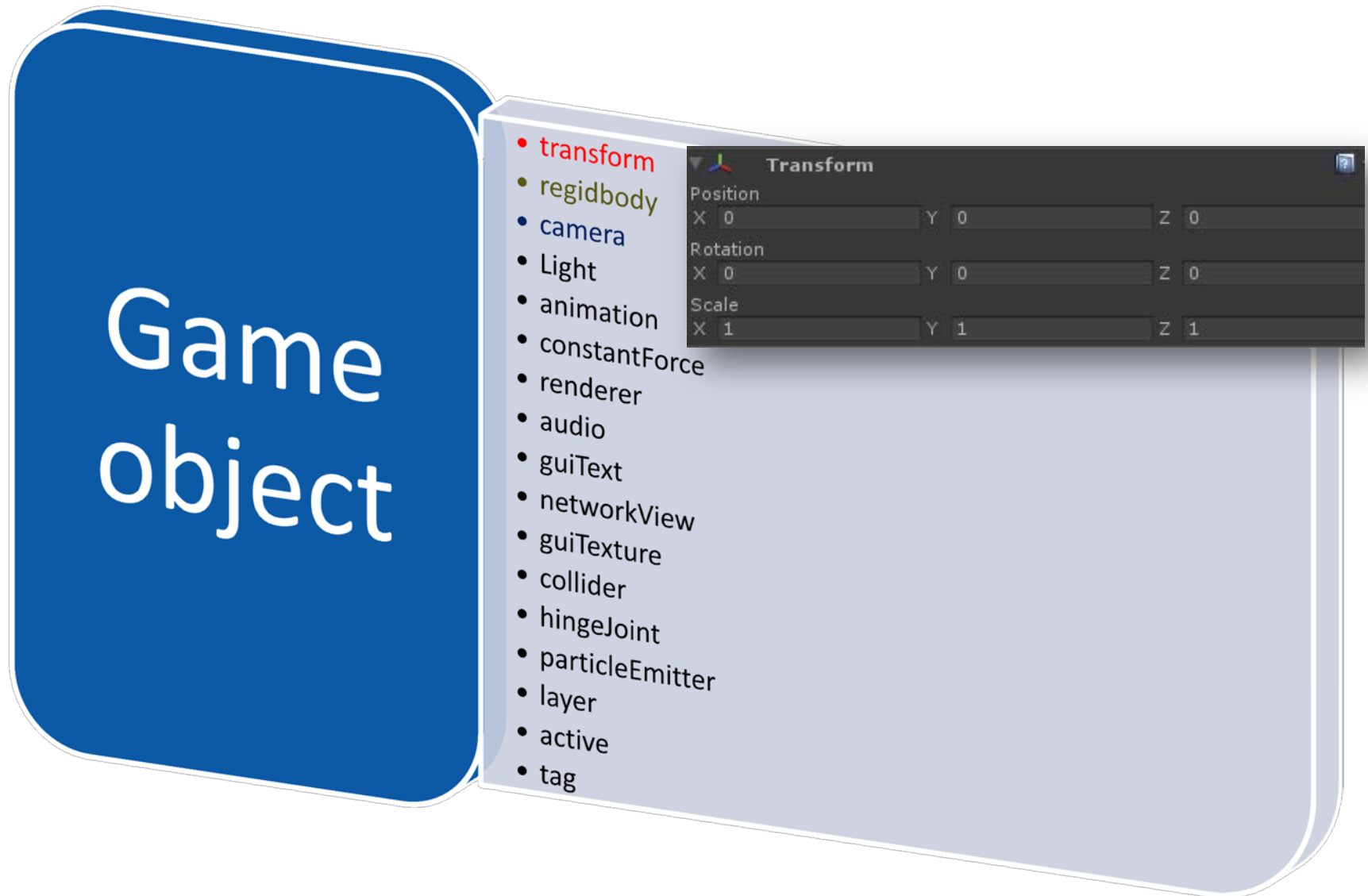
```
private function HelloWorld()  
{  
    Debug.Log("I can create easy a function");  
}
```

Ngôn ngữ - JavaScript

- Viết JavaScript trong Unity
 - Unity có chứa các GameObjects, mỗi GameObject có nhiều thành phần (components), mỗi component có nhiều thuộc tính (properties).
 - Chúng ta có thể viết code để thêm/xóa các component
 - Chúng ta có thể viết code để thay đổi giá trị properties của của các component
 - Logic:



Ngôn ngữ - JavaScript



Ngôn ngữ - JavaScript

- Bây giờ, ta viết một đoạn code để điều khiển gameobject di chuyển trên trục x

```
function Update()
```

```
{
```

```
    gameObject.transform.position.x += 1;
```

```
}
```

- Hướng dẫn: Menu >> Help >> Scripting Reference >> Nhập "gameobject"
- Chọn transform >> Sau đó chọn Transform class. Bây giờ nhìn giá trị position property.

Ngôn ngữ - JavaScript

- Tương tự áp dụng với các thành phần khác như: camera, light, rigidbody, sound, animation, guiText,...
- Ví dụ:

var other : GameObject;

other.camera.fieldOfView = 45;

Ngôn ngữ - JavaScript

- Ví dụ code: "SetRotateObj.js"

function Update()

{

//set gameobject move to (0,0,0)

transform.position = Vector3(0, 0, 0);

// Set's the rigidbody velocity x

*rigidbody.velocity = transform.forward * 10;*

// Y axis at the same speed.

*transform.Rotate(Vector3.up * Time.deltaTime, Space.World);*

}

Ngôn ngữ - JavaScript

- Class trong JavaScript
 - Khai báo bằng cách sử dụng từ khóa "class"
 - Kế thừa: sử dụng từ khóa "extends"
 - Nên kế thừa từ "MonoBehavior"

Class MyJavaScriptClass extends MonoBehaviour

{

// member variable


// method

}



DEMO

Javascript

A white hand cursor icon, resembling a mouse pointer, is pointing upwards at the letter 'O' in the word 'DEMO'. The hand is stylized with a rounded palm and four fingers.

Ngôn ngữ– C#

- **C#**
 - Cú pháp trong C#
 - Class
 - Variable
 - Method
 - Điều khiển GameObject với C#

Ngôn ngữ– C#

- Có thể dùng MS.Net C# để viết mọi thứ trong Unity
- Phải có kinh nghiệm lập trình hướng đối tượng OOP
- C# được biên dịch nhanh hơn JavaScript.
- C# không giống JavaScript về cách biên dịch, JavaScript thực hiện từng bước một còn C# thì không
- C# sử dụng các lớp phức tạp hơn.

Ngôn ngữ– C#

- Class: Khai báo
 - Sử dụng từ khóa “class”
 - Tiếp theo: Tên class, phải đặt tên của class giống tên file .cs

Ví dụ: tên file “*MyClass.cs*”

Tên class phải khai báo như sau:

```
public class MyClass: MonoBehaviour
```

```
{
```

```
}
```

Ngôn ngữ– C#

- Class: Import
 - Sử dụng từ khóa *“using”*
 - *Thư viện sử dụng phải có trên Mono hoặc thư viện .Net*
 - Ví dụ:

using UnityEngine;

using System.Collections;

public class MyClass: MonoBehaviour

{

}

Ngôn ngữ– C#

- Class: biến variable

- Cấu trúc sử dụng:

`private/public/static <type> <name> [= <new type()>];`

- *Khi khai báo biến, phải khai báo loại biến*

- Ví dụ:

using UnityEngine;

using System.Collections;

public class MyClass: MonoBehaviour

{

public int count = 0;

private bool flagShow = false;

static string myName = "Langvv";

}

Ngôn ngữ– C#

- Class: method
 - Sử dụng cấu trúc: `private/public/static <type_return> <name_func> ([par1,par2]) { ... }`

- Ví dụ:

```
using UnityEngine;  
using System.Collections;  
public class MyClass: MonoBehaviour{  
    private int count = 0;  
    private void Start(){ // Event  
        count = 100; }  
    public int GetCount(){ //method  
        return count; }  
}
```

Ngôn ngữ– C#

- Điều khiển GameObject của C# giống như JavaScript.



- Sự kiện events trong C# giống sự kiện trong JavaScript

Ngôn ngữ– C#


- Điều khiển transform của gameobject

```
using UnityEngine;  
using System.Collections;  
public class MyClass: MonoBehaviour{  
    private int count = 0;  
    void Update(){ //event  
        gameObject.transform.position.x += 1;  
    }  
    void OnMouseDown(){  
        count +=1; Debug.Log("Get count = "+count);  
    }  
}
```



DEMO

C#

A white hand cursor icon, resembling a mouse pointer, is pointing upwards towards the 'C#' text. The hand is stylized with a white palm and fingers.

So sánh C# - Javascript

- Về Unity Script Directives

// Javascript example

@script AddComponentMenu ("Transform/Follow Transform")

*class FollowTransform extends MonoBehaviour {
}*

// C# example:

*[AddComponentMenu("Transform/Follow Transform")]
class FollowTransform : MonoBehaviour {
}*

So sánh C# - Javascript

- Các loại biến variable

C#: bool, Javascript: boolean

C#: string, Javascript: String

- Khai báo biến

// C#: the type is always stated when declaring a variable:

public int myPublicInt = 1; // a public var

*int myPrivateInt = 2; // **private** access is default, if access is unspecified*

public GameObject myObj; // a type is specified, but no value assigned

// Javascript - type specification is not necessary:

*var myPublicInt = 1; // **public** access is default, if unspecified*

private var myPrivateInt = 2; // a private var

var myObj : GameObject; // a type is specified, but no value assigned

So sánh C# - Javascript

- Variables with dynamic type resolution

Chỉ trong Javascript, các biến mới có kiểu unspecified. Chỉ xảy ra nếu bạn không gán giá trị trong khi khai báo biến.

Ví dụ:

// Javascript:

var numBullets : int; // statically typed (because type specified)

var numLivesLeft = 3; // statically typed (because type is inferred from value assigned)

var player; // dynamically typed (because neither a type or value is specified)

So sánh C# - Javascript

- Khai báo mảng nhiều chiều

// C#:

int[,] = new int[16,16]; // 16x16 2d int array

// Javascript:

var a = new int[16,16];

So sánh C# - Javascript

- **Khai báo class:**

// Javascript example

```
class MyClass extends MonoBehaviour {  
    var myVar = 1;  
    function Start() {  
        Debug.Log("hello world!");  
    }  
}
```

// C# example:

```
class MyClass : MonoBehaviour {  
    public int myVar = 1;  
    void Start() {  
        Debug.Log("hello world!");  
    }  
}
```

Lớp Components

- Là lớp cơ bản nhất được tích hợp vào các GameObject.
- Lớp Component bao gồm các thuộc tính lưu trữ những thông tin cơ bản và quan trọng nhất đối với một GameObject như vị trí của đối tượng trong không gian 3D, độ xoay của đối tượng, tên, tag ...v...v

Thuộc tính	Chú thích
transform	Thông tin về vị trí, độ xoay
rigidbody	Giả lập khối lượng và trọng lượng cho GameObject.
camera	Truy xuất đến các thuộc tính của class Camera nếu GameObject được tích hợp một camer
light	Truy xuất đến các thuộc tính của class Light nếu GameObject được tích hợp một Light.
animation	Truy xuất đến các thuộc tính của class Animation nếu GameObject bao gồm các animation.
constantForce	Truy xuất đến các thuộc tính của class ConstantForce nếu GameObject được tích hợp ConstantForce.

Lớp Components

Thuộc tính	Chú thích
renderer	Truy xuất đến các thuộc tính của class Renderer nếu GameObject được tích hợp Renderer.
audio	Truy xuất đến các thuộc tính của class Audio nếu GameObject được tích hợp Audio.
guiText	Truy xuất đến các thuộc tính của class GUIText nếu GameObject được tích hợp GUIText
networkView	Truy xuất đến các thuộc tính của class NetworkView nếu GameObject được tích hợp NetworkView.
guiTexture	Truy xuất đến các thuộc tính của class GUITexture nếu GameObject được tích hợp GUITexture.
collider	Truy xuất đến các thuộc tính của class Collider nếu GameObject được tích hợp Collider (được dùng để kiểm tra các va chạm).
hingeJoint	Truy xuất đến các thuộc tính của class HingeJoint nếu GameObject được tích hợp HingeJoint.
particleEmitter	Truy xuất đến các thuộc tính của class ParticleEmitter nếu GameObject được tích hợp ParticleEmitter.
particleSystem	Truy xuất đến các thuộc tính của class ParticleSystem nếu GameObject được tích hợp ParticleSystem.

Lớp Components

Thuộc tính	Chú thích
gameObject	Truy xuất đến đối tượng mà nó tích hợp đến.
tag	Tag của component.

- Chú ý: Mặc định khi khởi tạo một đối tượng, đối tượng đó sẽ mang tên "GameObject" và được tích hợp sẵn thành phần Transform.

Lớp Components

Phương thức	Chú thích
GetComponent	Trả về kiểu Component được tích hợp vào đối tượng.
GetComponentInChildren	Trả về kiểu Component được tích hợp vào đối tượng hoặc con của đối tượng.
GetComponentsInChildren	Giống GetComponentInChildren nhưng trả về nhiều
Component	GetComponents Giống GetComponent nhưng trả về nhiều Component
CompareTag	Đối tượng có thuộc tag nào không?
SendMessageUpwards	Gửi yêu cầu thực hiện một phương thức đến các thành phần được tích hợp trong cùng một đối tượng.
SendMessage	Gửi yêu cầu thực hiện một phương thức đến các thành phần được tích hợp trong cùng một đối tượng.
BroadcastMessage	Gửi yêu cầu thực hiện một phương thức đến các đối tượng con.

Truy xuất đến thuộc tính của script

- Bắt đầu bằng một ví dụ cơ bản, đoạn code sau sẽ di chuyển một đối tượng thông qua input và một biến hiệu chỉnh tốc độ.

```
using UnityEngine;
```

```
using System.Collections;
```

```
public class Move : MonoBehaviour {
```

```
    public float speed = 5.0F;//tốc độ của đối tượng
```

```
void Start () {
```

```
}
```

```
// Update is called once per frame
```

```
void Update () {
```

```
    float x = Input.GetAxis("Horizontal") * Time.deltaTime * speed; // tốc độ theo cấu trúc
```

```
    ngang
```

```
    float z = Input.GetAxis("Vertical") * Time.deltaTime * speed; // tốc độ theo cấu trúc ngang
```

```
    transform.Translate(x,0,z);// di chuyển theo vector (x,0,z)
```

```
}
```

```
}
```

Truy xuất đến thuộc tính của script

- **Chú ý:** Sau khi tích hợp script trên vào một đối tượng, thuộc tính speed của class Move được public ra ngoài để ta có thể hiệu chỉnh trực tiếp bằng cách click chuột và nhập thông số, bản thân các script nên được xây dựng như một tool để tiện lợi cho việc hiệu chỉnh và tránh can thiệp vào mã nguồn, ngoài ra còn đơn giản hóa việc sử dụng script.

Truy xuất đến thuộc tính của script

- Trong quá trình phát triển chúng ta sẽ có nhu cầu về việc hiệu chỉnh tốc độ thông qua code, sử dụng phương thức GetComponent để truy xuất đến thuộc tính của một thành phần bất kỳ được tích hợp trong đối tượng.

```
using UnityEngine;
using System.Collections;
public class SetSpeed : MonoBehaviour {
    // Use this for initialization
    void Start () {
        // tăng tốc độ mỗi khi player nhấn phím U
    void Update () {
        if (Input.GetKey(KeyCode.U))
        {
            Move moveScript = gameObject.GetComponent<Move>; // truy xuất đến
            // lớp Move
            moveScript.speed += 0.5F; // tăng tốc độ
        }
    }
}
```

Truy xuất đến thuộc tính của script

- **Giải thích:** Sau khi tích hợp script SetSpeed vào cùng một đối tượng với script Move, thuộc tính gameObject sẽ truy xuất đến đối tượng chứa 2 script này và thông qua phương thức GetComponent để truy xuất đến một thành phần bất kỳ được tích hợp vào đối tượng, cụ thể ở đây là script Move.
- **Chú ý:** Tại script C# không thể truy xuất đến script Javascript thông qua phương thức GetComponent.

Các phương thức cơ bản

- Khi khởi tạo một script, mặc định nó đã chứa các phương thức `Start()`, `Update()`, đây là hai phương thức rất hữu dụng, ngoài ra còn một số phương thức khác được liệt kê dưới đây:
- **Update():** Những đoạn code thuộc phương thức này sẽ được gọi lại mỗi frame (khung hình).
- **FixedUpdate():** Giống với `Update` nhưng ta có thể hiệu chỉnh được số khung hình ban đầu, và không bị ảnh hưởng khi tần số frame không ổn định.
- **Awake():** Những đoạn code thuộc phương thức này được gọi khi script ở giai đoạn khởi tạo, thường được dùng để thiết lập hoặc tải dữ liệu ban đầu cho các Component.

Các phương thức cơ bản

- **Start():** Được gọi thực hiện trước phương thức Update() nhưng lại sau Awake(), Khác nhau cơ bản giữa Start() và Awake() là phương thức Start() chỉ được gọi khi script không bị đình chỉ hoạt động (enabled).
- **OnCollisionEnter():** Những đoạn code thuộc phương thức này sẽ được thực thi khi đối tượng chứa nó bao gồm một Collider và va chạm với một Collider hoặc Rigidbody của một đối tượng khác.
- **OnMouseDown():** Những đoạn code thuộc phương thức này sẽ được thực thi khi chuột của người chơi click vào một đối tượng có tích hợp thành phần GUIElement hoặc Collider.



Debugging

- Unity cung cấp lớp Debug để hỗ trợ lập trình viên theo dõi và kiểm soát các lỗi, ở đây chúng ta quan tâm đến phương thức Debug.Log().
- Phương thức Log() cho phép người dùng gửi một thông tin đến Unity Console nhằm mục đích:
 - Chứng minh rằng đoạn mã này đang được thực hiện.
 - Báo cáo tình trạng hiện tại của biến.
- Trở lại ví dụ về lớp SetSpeed, kiểm tra tốc độ của đối tượng mỗi khi người chơi tăng tốc.

Debugging

- Trở lại ví dụ về lớp SetSpeed, kiểm tra tốc độ của đối tượng mỗi khi người chơi tăng tốc.

```
using UnityEngine;
using System.Collections;
public class SetSpeed : MonoBehaviour {
    // Use this for initialization
    void Start () {
    }
    // Update is called once per frame
    void Update () {
        if (Input.GetKey(KeyCode.U))
        {
            Move moveScript = gameObject.GetComponent<Move>;// truy xuất đến lớp Move
            moveScript.speed += 0.5F;// tăng tốc độ
            Debug.Log(moveScript.speed);
        }
    }
}
```

Tối ưu hóa mã nguồn:

Sử dụng các thành phần của GameObject thông qua biến tĩnh:

- Như đã tìm hiểu ở các mục trên, bản thân một GameObject luôn được tích hợp sẵn các thành phần để lưu trữ thông tin của chúng trong không gian 3D, ở đây chúng ta sẽ lấy ví dụ cụ thể với thành phần transform.

```
using UnityEngine;
using System.Collections;

public class Example : MonoBehaviour {

    void Update () {
        transform.Translate(0,0,0);
    }
}
```

Tối ưu hóa mã nguồn:

Sử dụng các thành phần của GameObject thông qua biến tĩnh:

- Đoạn mã trên được sửa lại như sau:

```
using UnityEngine;
using System.Collections;
public class Example : MonoBehaviour {
    private Transform myTransform;
    void Awake() {
        myTransform = transform;
    }
    void Update () {
        myTransform.Translate(0,0,0);
    }
}
```

Tối ưu hóa mã nguồn:

Sử dụng các thành phần của GameObject thông qua biến tĩnh:

- Mỗi khi chúng ta gọi một thành phần bất kỳ của GameObject, hệ thống Unity sẽ tốn thời gian để duyệt qua các thành phần được tích hợp trong GameObject để cuối cùng trả về kết quả thích hợp,
- Để tiết kiệm khoản thời gian đó nên sử dụng thành phần của GameObject thông qua biến tĩnh.

Tối ưu hóa mã nguồn:

Sử dụng mảng tĩnh

- Các lớp ArrayList hay Array thì rất dễ sử dụng, chúng ta có thể dễ dàng thêm một phần tử vào mảng và sử dụng các phương thức, nhưng chi phí phải trả để hệ thống thực hiện điều đó là rất cao.
- Thay vì vậy, việc sử dụng các mảng được cấu trúc sẵn sẽ giảm bớt công việc cho hệ thống, vì các phần tử trong mảng có cùng kiểu và độ dài của mảng đã được xác định từ trước, chúng ta chỉ mất thời gian để xác định độ dài mảng cần thiết, bù lại chúng ta tiết kiệm được một khoản chi phí khá lớn.

Tối ưu hóa mã nguồn:

Sử dụng mảng tĩnh

- Chú ý: Hạn chế sử dụng for, foreach, nên sử dụng while để đạt được tốc độ tối ưu.

```
using UnityEngine;
using System.Collections;
public class Example : MonoBehaviour {
    private Vector3[] positions;
    void Awake() {
        positions = new Vector3[100];
        int i = 0;
        while (i < 100) {
            positions[i] = Vector3.zero;
            i++;
        }
    }
}
```

Tối ưu hóa mã nguồn:

Hiệu chỉnh tần số sử dụng phương thức

- **Đặt vấn đề:** Khi người chơi ở quá xa, kẻ địch sẽ ở trạng thái ngủ tạm thời.
- **Giải thích:** Đoạn mã trên sẽ xét khoảng cách giữa người chơi và kẻ địch tại mỗi frame, nhưng điều đó là không cần thiết, trung bình một giây hệ thống chạy từ 30 đến 40 khung hình, chi phí bỏ ra quá lớn nhưng hiệu quả lại không cao.

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

```
public class Example : MonoBehaviour {
```

```
    public Transform target;
```

```
    void Update () {  
        if (Vector3.Distance(transform.position, target.position) > 100)  
            return;  
    }  
}
```

Tối ưu hóa mã nguồn:

Yield và Coroutine

- Các Coroutine cho phép chúng ta đình trệ hay làm trễ việc thực thi một đoạn mã hoặc một phương thức, áp dụng để xử lý vấn đề nêu trên.

```
using UnityEngine;
using System.Collections;
public class Example : MonoBehaviour {
    public Transform target;
    void Start()
    {
        StartCoroutine("TestDistance");
    }
    IEnumerator TestDistance()
    {
        while (true)
        {
            if (Vector3.Distance(transform.position, target.position) > 100)
            {
                Debug.Log("Khong lam gi");
            }
            yield return new WaitForSeconds(2);
        }
    }
}
```


Tối ưu hóa mã nguồn:

Yield và Coroutine

- **Giải thích:** Việc thực thi vòng lặp while sẽ bị chậm lại 2 giây sau mỗi lần lặp do tác động của yield, việc xét khoảng cách giữa người chơi và kẻ địch sẽ được thực thi 2 giây 1 lần, tiết kiệm được nhiều chi phí và giảm nhẹ công việc cho hệ thống.



Giao diện Scene,
Inspector, menu

Kết luận

- Ngôn ngữ trong Unity
- JavaScript
 - Syntax của JavaScript trong game
 - Điều khiển GameObject với JavaScript
- C#
 - Syntax của C# trong game
 - Điều khiển GameObject với C#
- So sánh C# và JavaScript
- Lớp components
- Truy xuất thuộc tính của script
- Các phương thức cơ bản
- Debugging
- Tối ưu hóa mã nguồn



FPT POLYTECHNIC

THANK YOU!

www.poly.edu.vn