

Bài 4: Prefab, Script và một số xử lý cơ bản

Giảng viên:

MỤC TIÊU

- Cấu trúc Project
- A. Khởi tạo và cấu hình dự án Game 2D
- B. Tạo các đối tượng cơ bản
 - 1. *Game Object*
 - 2. *Sprite*
 - 3. *Animation và điều khiển hành động nhân vật*
 - **4. *Prefab***
 - **5. *Script và điều khiển máy trạng thái***
 - 6. *Thành phần vật lý và xử lý va chạm*
 - 7. *Sử dụng Text,*
 - 8. *Sử dụng Particle System*
 - 9. *Chuyển đổi màn chơi*
 - 10. *Sound*
 - 11. *Design Pattern trong Game*

Nội dung

- Cấu trúc Project
- A. Khởi tạo và cấu hình dự án Game 2D
- B. Tạo các đối tượng cơ bản
 - 1. *Game Object*
 - 2. *Sprite*
 - 3. *Animation và điều khiển hành động nhân vật*
 - **4. *Prefab***
 - **5. *Script và điều khiển máy trạng thái***
 - 6. *Thành phần vật lý và xử lý va chạm*
 - 7. *Sử dụng Text,*
 - 8. *Sử dụng Particle System*
 - 9. *Chuyển đổi màn chơi*
 - 10. *Sound*
 - 11. *Design Pattern trong Game*

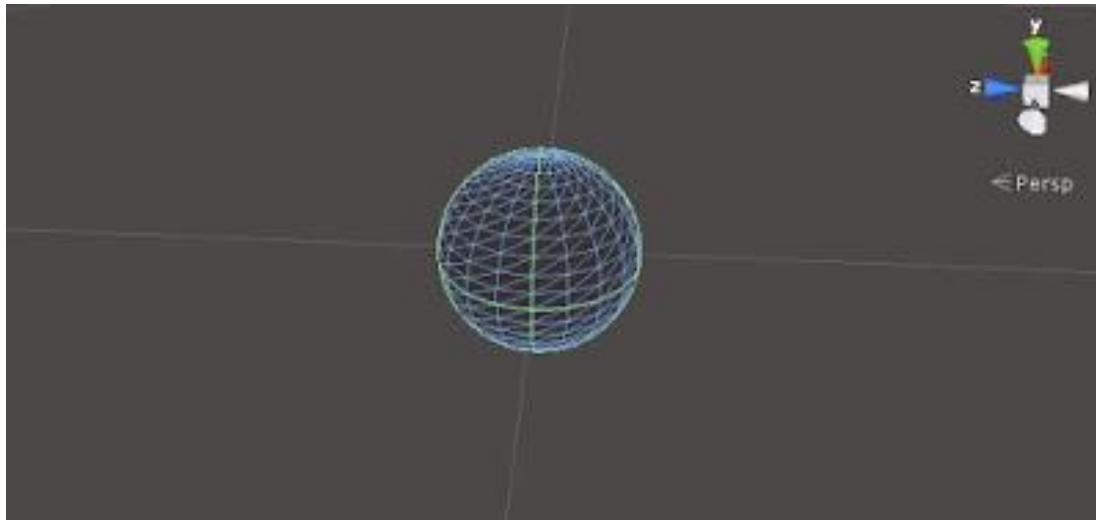
Prefab

- Prefab cho ta tạo ra các bản sao nhanh của một đối tượng mà không cần thiết lập lại các giá trị khởi tạo của một đối tượng nào đó ngoài trừ các giá trị transform (vị trí, tỉ lệ, quay).
- Để tạo một Prefab cho một đối tượng nào đó, ta chỉ cần kéo thả đối tượng đó ở cửa sổ Hierarchy xuống thư mục Prefab trong cửa sổ Project.
- Sau này muốn sử dụng ta chỉ việc kéo các Prefab này trở lại cửa sổ Scene.

Prefab

Cách để làm...

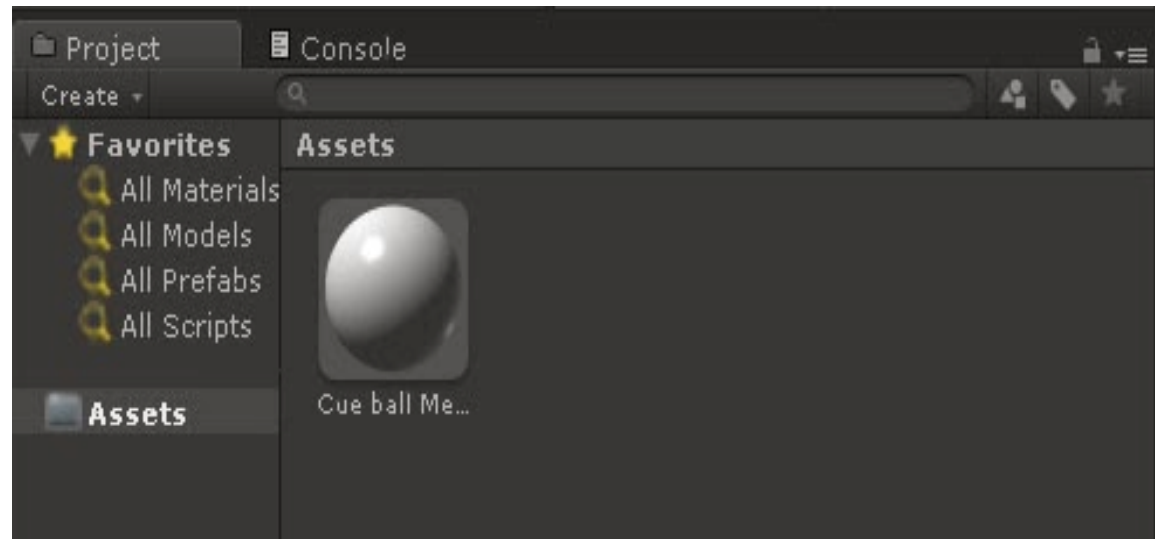
- Để tạo một Prefab, chúng ta sẽ làm như sau:
- 1. Trong Unity editor, vào GameObject | 3D Object| Sphere.



Prefab

Cách để làm...

- 2. Trong thẻ Hierarchy, nhấp chuột phải vào Sphere và chọn Rename rồi đổi tên thành Cue Ball.
- 3. Trong thẻ Project, nhấp phải vào khung Asset và chọn Create | Material. Sau đó đổi tên vật liệu mới tạo lại thành Cue Ball Material.



Prefab

Cách để làm...

- 4. Nhấp chuột chọn Cue Ball Material trong thẻ Project. Sau đó qua thẻ Inspector và đổi giá trị Shader thành Specular.
- 5. Điều chỉnh Specular Color thành màu trắng và kéo thanh trượt Shininess về phía bên phải để thiết lập giá trị cao nhất.
- 6. Từ thẻ Project, kéo Cue Ball Material trong thẻ Project vào Cue Ball trong thẻ Hierarchy.

Prefab

Cách để làm...

- 7. Nhấp chọn Cue Ball trong thẻ Hierarchy. Vào Component | Physics | Rigidbody để gắn Rigidbody (giả lập khối lượng và trọng lượng cho sự vật).
- 8. Nhấp chuột phải vào khoảng trống trong thẻ Project và chọn Create | Prefab. Sau đó đổi tên lại thành Cue Ball Prefab.
- 9. Kéo Cue ball trong thẻ Hierarchy vào Cue Ball Prefab vừa tạo. Và chúng ta có thể sử dụng lại gói tài nguyên này vào các lần sau.

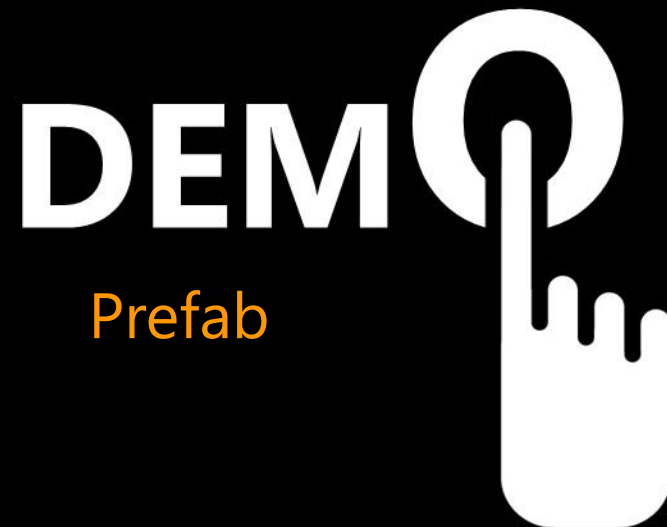
Prefab

Công dụng...

- Trong Unity, game object (đối tượng được sử dụng trong game) có thể được chứa vào một cái kho gọi là Prefab.
- Điều này rất hữu dụng cho các trường hợp muốn sử dụng lại một game object trong nhiều tầng khác nhau hay dùng các đối tượng này vào để code.
- Tương tự như chức năng MovieClips trong Adobe Flash.

Mang những Prefab vào project khác

- Bạn cũng có thể sử dụng lại Prefab của bạn vào các project khác bằng cách Export (xuất tài nguyên) ra thành một Custom package và Import (nhập tài nguyên) vào project bạn cần dùng.

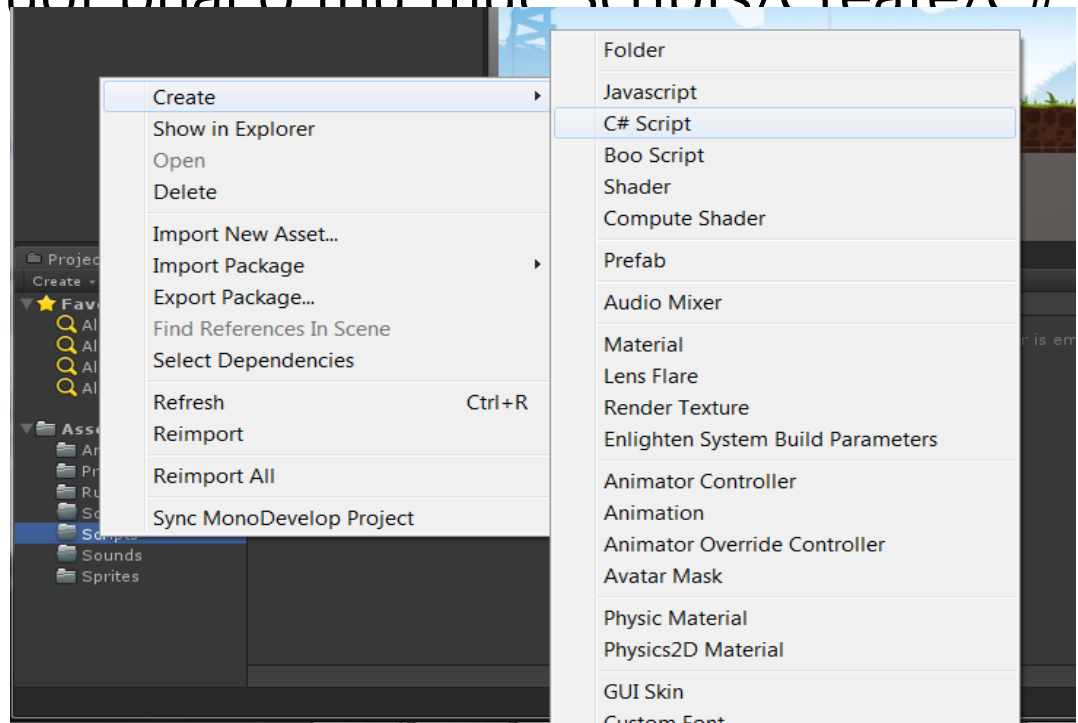


Script và một số xử lý cơ bản

Script

- Script là một tập tin (cũng là Game Component) chứa các mã điều khiển cho một đối tượng nào đó trong game, được viết bằng C# hay Javascript hoặc BOO.
- Để tạo script ta click chuột phải ở thư mục Scripts/Create/C# Script

Chú ý: Đổi tên file Script và tên class thành MainCharacterBehavior.



Script và một số xử lý cơ bản

Script

- **Tip:** Chúng ta chú ý đến việc đặt tên các hàm, biến, đối tượng rõ ràng, phù hợp với chức năng của chúng, điều này sẽ giúp chúng ta kiểm soát code dễ dàng, đọc code dễ hiểu, dễ debug, dễ dàng phán đoán ra lỗi, và đặt biệt là khi làm nhóm.
- Tránh đặt tên viết tắt, sai lệch với chức năng, phải viết comment dài dòng gây rối code và khó hiểu cho chúng ta...

Script và một số xử lý cơ bản

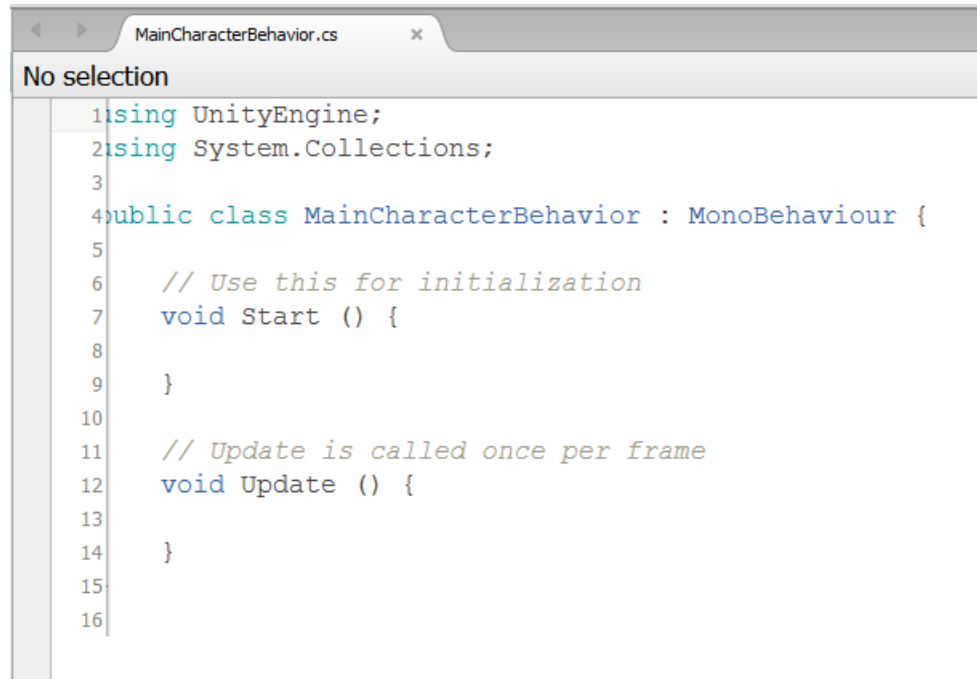
Script

- Với mỗi script tạo ra, ta cần chỉ định script này thuộc về đối tượng nào. Ta thực hiện bằng cách:
- Chọn đối tượng MainCharacter ở cửa sổ Hierarchy, ở cửa sổ Inspector chọn **Add Component/Scripts/Main Character Behaviour.cs** ở danh sách.

Script và một số xử lý cơ bản

Script

- Kích đúp để mở file script ta sẽ thấy như sau.



```
1 using UnityEngine;
2 using System.Collections;
3
4 public class MainCharacterBehavior : MonoBehaviour {
5
6     // Use this for initialization
7     void Start () {
8
9     }
10
11    // Update is called once per frame
12    void Update () {
13
14    }
15
16 }
```

Cấu trúc script mặc định



DEMO

Tạo một
script

A white hand cursor icon, resembling a computer mouse pointer, is pointing at the letter 'O' in the word 'DEMO'. The hand is stylized with a white outline and a white fill.

Script và một số xử lý cơ bản

Script

■ Khi khởi tạo Scene:

- **Awake:** Hàm này luôn chạy khi một scene bắt đầu, chú ý đối với các đối tượng được tạo ra trong lúc đã load scene thì ko nên dùng hàm này
- **OnEnable:** Cũng mặc định này như hàm Awake , nhưng sẽ không chạy nếu bạn ẩn đối tượng này ngay lúc đầu, Hàm này chạy sau hàm Awake

■ Trước khi hàm update đầu tiên được chạy:

- **Start:** không cần phải nói nhiều, muốn khởi tạo gì thì cứ thỏa mái ở đây

■ Giữa các frame:

- **OnapplicationPause:** Trong trường hợp cờ pause game được bật lên thì nên sử dụng hàm này thay cho Update

Script và một số xử lý cơ bản

Script

■ Update:

- **FixedUpdate:** Thường được gọi nhiều lần hơn Update, có thể được gọi nhiều lần trong 1 frame, điều đặc biệt là FixedUpdate này sẽ chạy theo một định thời cố định, nếu muốn game chạy chính xác thì bạn có thể thay thế hoàn toàn hàm Update bằng FixedUpdate, còn nếu yêu cầu tốc độ thì nên làm ngược lại
- **Update:** Chạy mỗi frame 1 lần
- **LateUpdate:** Chạy mỗi frame 1 lần, luôn luôn chạy sau hàm Update, Hàm này rất thích hợp cho trường hợp xung đột bộ nhớ do các hàm Update của các đối tượng khác nhau dùng chung 1 bộ nhớ
- **OnDestroy:** Được gọi khi một đối tượng bị bạn Destroy, thông thường sẽ dc gọi sau hàm LateUpdate, kể từ lúc bạn Destroy
- **OnApplicationQuit:** Được gọi khi Bạn quit game

Script và một số xử lý cơ bản

Script

- Vòng đời của một script hay vòng đời của một game object các bạn có thể tham khảo chi tiết tại:
<https://docs.unity3d.com/Documentation/Manual/ExecutionOrder.html>.
- Mới bắt đầu chúng ta cần quan tâm đến một số hàm cơ bản sau:
 - **Start()**: Được gọi 1 lần đầu tiên sau khi khởi tạo đối tượng, trước khi vào Update.
 - **Update()**: Được gọi liên tục sau mỗi frame, sau Start.
 - **OnWillRenderObject()**: Được gọi liên tục sau một frame sau Update. Thường thì ta sẽ ít đụng đến hàm này.
 - **OnDestroy()**: Được gọi khi đối tượng bị huỷ.

Script và một số xử lý cơ bản

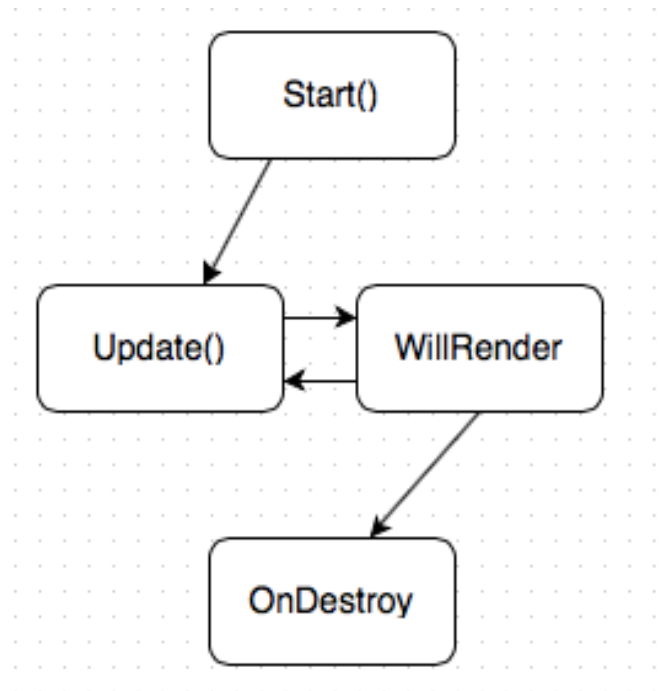
Script

- Vòng đời của một script hay vòng đời của một game object các bạn có thể tham khảo chi tiết tại:
<https://docs.unity3d.com/Documentation/Manual/ExecutionOrder.html>.
- Mới bắt đầu chúng ta cần quan tâm đến một số hàm cơ bản sau:
 - **Start()**: Được gọi 1 lần đầu tiên sau khi khởi tạo đối tượng, trước khi vào Update.
 - **Update()**: Được gọi liên tục sau mỗi frame, sau Start.
 - **OnWillRenderObject()**: Được gọi liên tục sau một frame sau Update. Thường thì ta sẽ ít đụng đến hàm này.
 - **OnDestroy()**: Được gọi khi đối tượng bị huỷ.

Script và một số xử lý cơ bản

Script

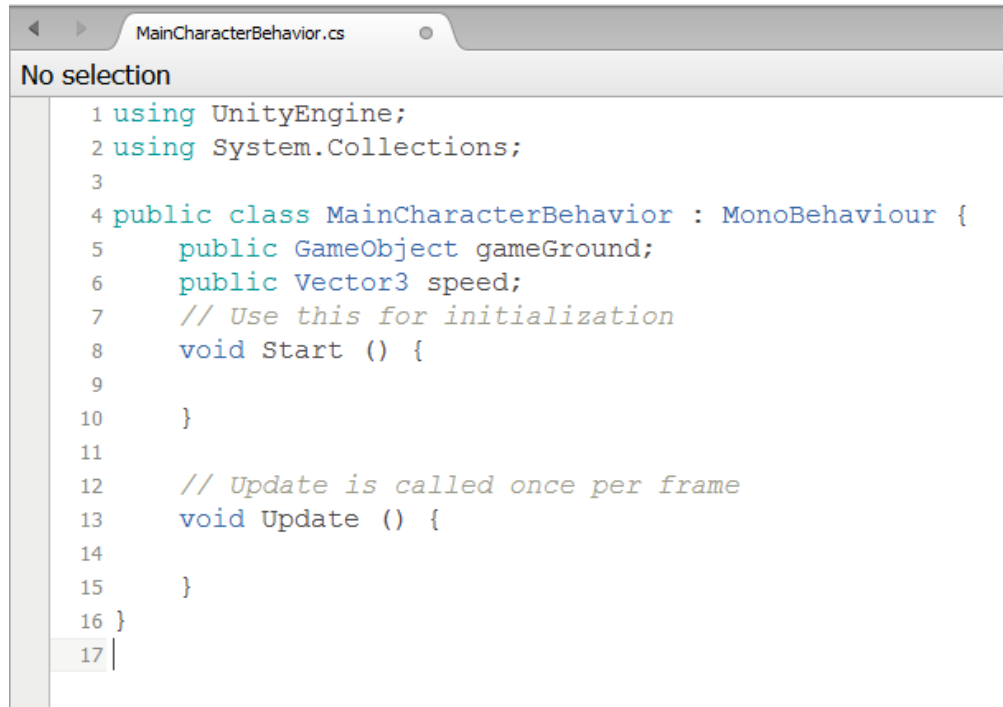
Biểu đồ:



Script và một số xử lý cơ bản

Script

- Ta có thể khai báo các thuộc tính trong file script, các thuộc tính này sẽ được hiển thị ở cửa sổ Inspector Ở MainCharacterBehaviour:

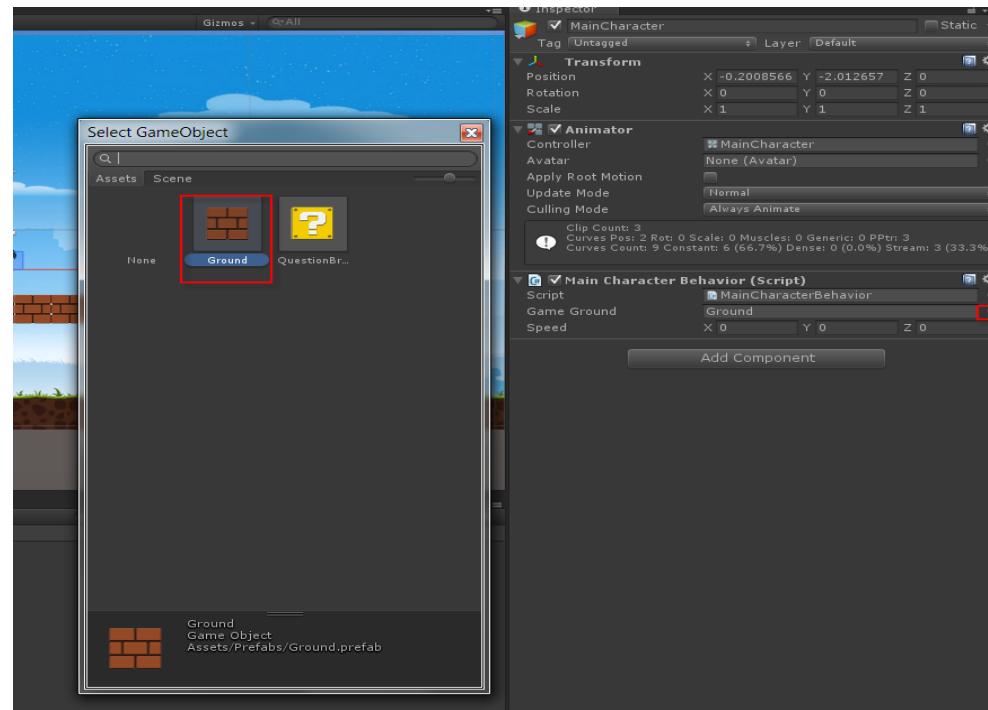
A screenshot of a code editor window titled 'MainCharacterBehavior.cs'. The editor shows a C# script for a Unity game. The script includes using statements for 'UnityEngine' and 'System.Collections'. It defines a 'MainCharacterBehavior' class that inherits from 'MonoBehaviour'. The class has two public fields: 'gameObject' of type 'GameObject' and 'speed' of type 'Vector3'. There are two methods: 'Start()' and 'Update()'. The 'Start()' method is currently empty. The 'Update()' method has a comment indicating it is called once per frame. The code is numbered from 1 to 17.

```
1 using UnityEngine;
2 using System.Collections;
3
4 public class MainCharacterBehavior : MonoBehaviour {
5     public GameObject gameObject;
6     public Vector3 speed;
7     // Use this for initialization
8     void Start () {
9
10    }
11
12    // Update is called once per frame
13    void Update () {
14
15    }
16 }
17
```

Script và một số xử lý cơ bản

Script

- Ở Inspector, đối với thuộc tính là GameObject chúng ta có thể nhấn vào nút đỏ, sau đó chọn prefabs cho đối tượng ở bảng mới hiện ra.



Script và một số xử lý cơ bản

Một số xử lý cơ bản

- ***this.gameObject***: truy cập vào đối tượng game hiện tại thông qua Thay đổi vị trí, tỉ lệ, quay đối tượng thông qua `gameObject.transform(.position, .scale, .rotate)`
- ***Destroy(GameObject)*** : để huỷ một đối tượng game (*)
- ***Instantiate(gameObject, Vector3, Quaternion)*** (**) : Để tạo một prefab trong quá trình thực thi game: .
- ***Input.GetKeyDown(keyCode), GetKey(keyCode), GetKeyUp(keyCode)***: kiểm tra xem một key được bấm, được giữ, được thả ra hay không ?
- **Chú ý**: (*), (**) thường được sử dụng trong trường hợp như một nhân vật bắn súng.

Script và một số xử lý cơ bản

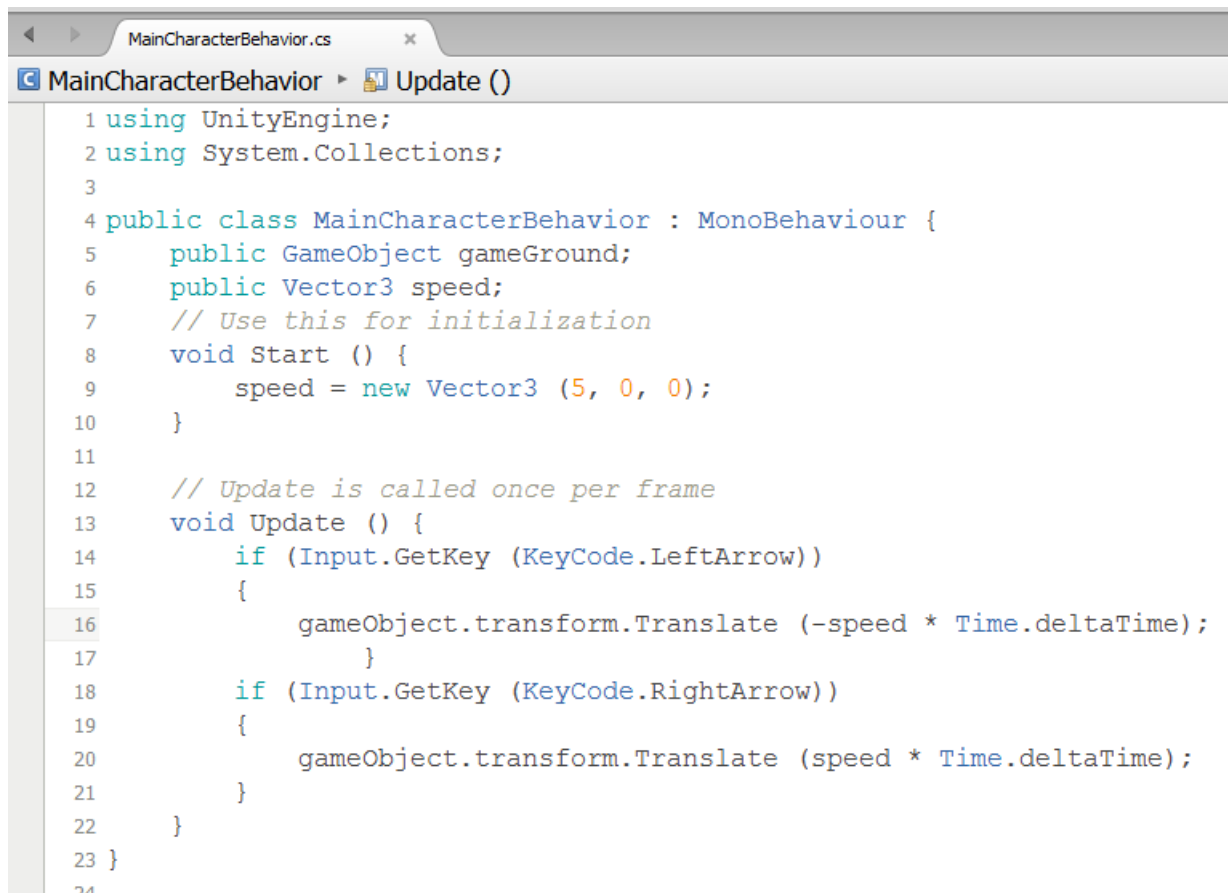
Một số xử lý cơ bản

- ***Input.GetAxis ("Horizontal")*** trả về giá trị số thực trong khoảng -1..1 nếu có sự kiện các key right hoặc left được bấm (key ngang).
- ***Input.GetAxis ("Vertical")*** trả về giá trị số thực trong khoảng -1..1 nếu có sự kiện các key up hoặc down được bấm (key dọc).
- ***OnMouseDown, OnMouseUp, OnMouseDown*** các hàm xử lý chuột, kiểm tra xem chuột được bấm, thả hoặc di chuyển
- ***gameObject.GetComponent<ComponentName>()*** get Game Component được đính kèm trong gameObject hiện tại có thể là: Animator, Transform ...

Script và một số xử lý cơ bản

Một số xử lý cơ bản

- Ví dụ dưới đây là một xử lý đơn giản khi bấm nút left, right là đối tượng sẽ đi qua đi về bằng cách thay đổi vị trí của đối tượng:



```
1 using UnityEngine;
2 using System.Collections;
3
4 public class MainCharacterBehavior : MonoBehaviour {
5     public GameObject gameGround;
6     public Vector3 speed;
7     // Use this for initialization
8     void Start () {
9         speed = new Vector3 (5, 0, 0);
10    }
11
12    // Update is called once per frame
13    void Update () {
14        if (Input.GetKey (KeyCode.LeftArrow))
15        {
16            gameObject.transform.Translate (-speed * Time.deltaTime);
17        }
18        if (Input.GetKey (KeyCode.RightArrow))
19        {
20            gameObject.transform.Translate (speed * Time.deltaTime);
21        }
22    }
23 }
24
```

Script và một số xử lý cơ bản

Điều khiển chuyển đổi trạng thái

- Ta sẽ set trạng thái mặc định của Animation là Idle. Các thiết lập trạng thái mặc định đã đề cập trong phần trước.
- Tiếp theo, ta sẽ khai báo một animator để tham chiếu đến Animator Component của đối tượng MainCharacter.

```
public GameObject gameGround;  
public Vector3 speed;  
protected Animator animator;  
// Use this for initialization  
void Start () {  
    speed = new Vector3 (5, 0, 0);  
    animator = gameObject.GetComponent<Animator> ();  
}
```

- Ở hàm update ta sẽ xử lý để chuyển đổi state như sau:

```
//Switch states  
if (Input.GetKey (KeyCode.LeftArrow) || Input.GetKey (KeyCode.RightArrow)) {  
    Debug.Log ("Running...");  
    animator.SetBool ("isIdle",false);  
    animator.SetBool ("isRunning",true);  
}  
  
else {  
    Debug.Log ("Idle...");  
    animator.SetBool ("isRunning",false);  
    animator.SetBool ("isIdle",true);  
}
```

Script và một số xử lý cơ bản

Điều khiển chuyển đổi trạng thái

- Tiếp theo ta sẽ xử lý thêm phần xoay đối tượng theo chiều x, khi nhấn nút Left/Right như sau:

```
Vector3 localScale = transform.localScale;
//Translations
if (Input.GetKey (KeyCode.LeftArrow))
{
    gameObject.transform.Translate (-speed * Time.deltaTime);
    if (localScale.x>0)
    {
        localScale.x *=-1.0f;
    }
}
if (Input.GetKey (KeyCode.RightArrow))
{
    gameObject.transform.Translate (speed * Time.deltaTime);
    if (localScale.x<0)
    {
        localScale.x *=-1.0f;
    }
}
```

Nhấn nút play và nhấn thử nút Left, Right ta sẽ thấy kết quả.



DEMO

Thực hiện
script



Kết luận

- Cấu trúc Project
- A. Khởi tạo và cấu hình dự án Game 2D
- B. Tạo các đối tượng cơ bản
 - 1. *Game Object*
 - 2. *Sprite*
 - 3. *Animation và điều khiển hành động nhân vật*
 - **4. *Prefab***
 - **5. *Script và điều khiển máy trạng thái***
 - 6. *Thành phần vật lý và xử lý va chạm*
 - 7. *Sử dụng Text,*
 - 8. *Sử dụng Particle System*
 - 9. *Chuyển đổi màn chơi*
 - 10. *Sound*
 - 11. *Design Pattern trong Game*

Chuẩn bị bài sau

- Cấu trúc Project
- A. Khởi tạo và cấu hình dự án Game 2D
- B. Tạo các đối tượng cơ bản
 - 1. *Game Object*
 - 2. *Sprite*
 - 3. *Animation và điều khiển hành động nhân vật*
 - 4. *Prefab*
 - 5. *Script và điều khiển máy trạng thái*
 - **6. Thành phần vật lý và xử lý va chạm**
 - 7. *Sử dụng Text,*
 - 8. *Sử dụng Particle System*
 - 9. *Chuyển đổi màn chơi*
 - 10. *Sound*
 - 11. *Design Pattern trong Game*



FPT POLYTECHNIC

THANK YOU!

www.poly.edu.vn