


Bài 2: Thành phần xử lý va chạm trong Unity



MỤC TIÊU

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 một số xử lý cơ bản
 - 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
- 

Thành phần xử lý va chạm

Va chạm trong Unity

- Va chạm và xử lý va chạm là những thành phần không thể thiếu khi lập trình game. Va chạm trong game là xảy ra khi có 2 object đi vào không gian của nhau.
- Ví dụ như : trúng đạn, trúng bom, chạm phải quái vật, xuất phát, tới đích, trúng mũi tên đều là các sự kiện va chạm và khi lập trình game chúng ta cần phải xử lý các va chạm đó.
- Chúng ta sẽ có 2 quá trình xử lý với va chạm như sau :
 - **Quá trình 1 : phát hiện và thông báo sự kiện va chạm**
 - **Quá trình 2 : xử lý sự kiện va chạm**

Thành phần xử lý va chạm

Va chạm trong Unity

- Trong Unity có 2 loại va chạm đó là :
- **Collision** : là loại va chạm mà 2 đối tượng sẽ không đi xuyên qua nhau, khi đối tượng này gặp đối tượng kia thì sẽ bị cản lại, bật lại tùy theo tính chất vật lý mà chúng ta xét cho đối tượng. Ví dụ : quả bóng rơi từ trên cao rơi xuống sân cỏ sẽ bật lên ...
- **Trigger** : là loại va chạm mà các đối tượng này có thể đi xuyên qua đối tượng kia. Ví dụ: chúng ta sẽ sử dụng trigger trong các hoạt cảnh như làm cho tiếng nhạc bật lên khi đối tượng đi qua loa hay làm cho cây đổ khi người chơi đi tới, hay lửa, hoặc quả bóng bay ngang qua ngọn lửa.

Thành phần xử lý va chạm

Va chạm trong Unity

Collision:

- Collision nghĩa là sự va chạm – khái niệm này gắn liền với Component Collider trong Unity. Unity cung cấp cho chúng ta một Class Collider trong đó có 3 Messages Sent (một kiểu event – sự kiện) để kiểm soát trạng thái của Collision như sau:

```
void OnCollisionEnter()
```

```
{Debug.Log("Phat hien va cham voi: " + info.gameObject.name);}
```

```
void OnCollisionStay()
```

```
{Debug.Log("O lai voi: " + info.gameObject.name);}
```

```
void OnCollisionExit()
```

```
{Debug.Log("Da thoat ra khoi: " + info.gameObject.name);}
```

Lưu ý: Thật sự Class này cung cấp 6 Messages Sent về Collision, tuy nhiên chúng ta khảo sát trước về 3 Messages Sent trên, 3 Messages Sent còn lại sẽ được nêu ngay sau đây (ở phần nói về Trigger).


Thành phần xử lý va chạm

Trigger:

- Trigger theo Google Translate nghĩa là “**Kích hoạt**” hay “Cò súng”.
- Trong Unity, Trigger là một loại Collider có thể xuyên qua, nếu Collider thông thường dùng để phát hiện va chạm và ngăn chặn mọi tác động xuyên qua Object của một Collider khác thì Trigger chỉ đóng vai trò phát hiện ra sự va chạm nhưng vẫn cho Collider tạo nên sự va chạm đó đi xuyên qua Object.
- Thao tác tạo Trigger cũng giống như tạo Collider, tuy nhiên sau khi tạo ra Collider, chúng ta **check** chọn giá trị “**Is Trigger**” trong **Component Collider** đó.

Thành phần xử lý va chạm

Trigger

- Dưới đây là 2 Message Sent khác của Trigger, 2 Message Sent này hợp cùng 3 Message Sent chúng ta đã khảo sát ở phần 1 tạo nên 5 Message Sent gắn liền với Class Collider, chuyên đảm nhận các vấn đề xoay quanh Collision Của Collider.
 - **OnTriggerExit**
 - **OnTriggerStay**
 - **Lưu ý:** Để xây dựng Trigger bắt sự kiện ở một vị trí nào đó như trên, chúng ta có thể tạo đơn giản bằng một **cube**, **check "Is Trigger"** sau đó bỏ check **Mesh Renderer** của Cube.
- 


Thành phần xử lý va chạm

Collider trong môi trường 3D:

- Vào Menu Component > Physics > chọn 1 kiểu Collider trong số các kiểu sau:
 - **Box Collider:** không gian va chạm bao quanh object là 1 hình khối vuông.
 - **Sphere:** ... hình khối cầu.
 - **Capsule:** ... hình viên nang.
 - **Mesh:** ... hình dạng tương ứng với hình dạng của vật thể (do vậy sẽ làm nặng game nhất).
 - **Wheel:** ... hình đĩa tròn.
 - **Terrain:** ... hình bề mặt, thường dùng cho mặt đất

Thành phần xử lý va chạm

Collider trong môi trường 3D:

- Các thông số trong Component Collider:
 - **Is Trigger:** xác định là loại va chạm nào, giá trị True là va chạm trigger, False là va chạm collision.
 - **Material:** bề mặt va chạm để tạo hiệu ứng vật lý, ví dụ như hiệu ứng đàn hồi cần bề mặt đàn hồi,
 - **Center:** vị trí tương đối của Collider so với object.
 - Ngoài ra, mỗi loại Collider sẽ có vài thông số riêng như: Box có Size, Sphere có Radius (bán kính), Capsule có Height và Direction, ...
- 


Thành phần xử lý va chạm

Collider trong môi trường 2D:

- 2D khác 3D ở chỗ không có chiều z trong không gian. Với Collider thì sự va chạm giữa 2 collider sẽ chỉ xét tới vị trí theo chiều x và y, không xét tới chiều z khi va chạm.
- Vào Component > Physics 2D > Chọn 1 trong 4 loại Collider 2D
 - **Circle collider:** collider dạng hình tròn.
 - **Box collider:** ...dạng hình vuông.
 - **Edge collider:**... dạng đoạn thẳng
 - **Polygon collider:**... dạng đa giác tùy biến

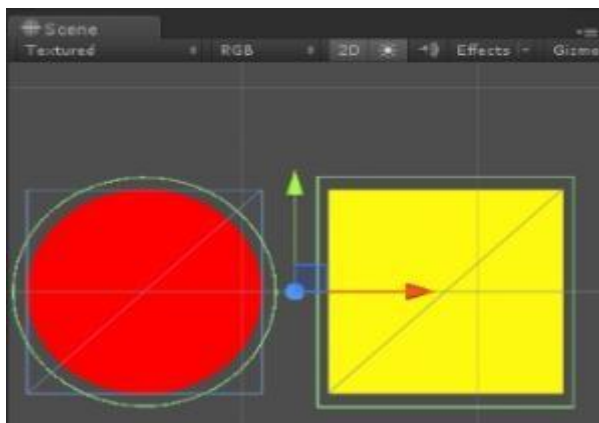
Thành phần xử lý va chạm

Collider trong môi trường 2D:

- Các thông số trong Component Collider:
 - **Is Trigger:** xác định là loại va chạm nào, giá trị True là va chạm trigger, False là va chạm collision.
 - **Material:** bề mặt va chạm để tạo hiệu ứng vật lý, ví dụ như hiệu ứng đàn hồi cần bề mặt đàn hồi,
 - **Used By Effector:**
 - **Offset:**
 - **Radius:**
- 

Thành phần xử lý va chạm

- Để hiểu rõ hơn ta đi vào ví dụ va chạm nhỏ trong Unity 2D.

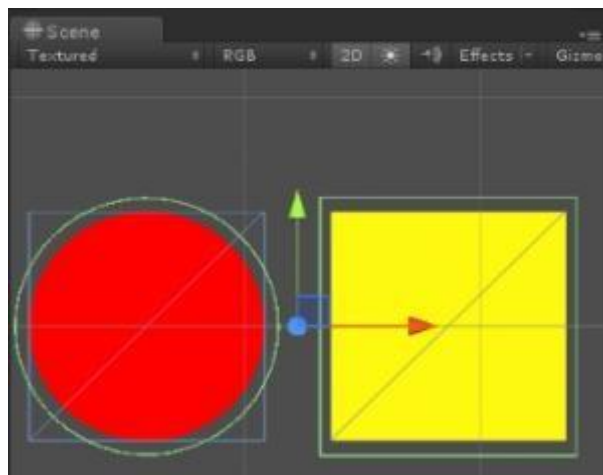


- Ví dụ, có 2 object là Hình Cầu A và Hình Khối B, để A và B có thể xảy ra va chạm, thì hai object này phải được “bao” lại bằng 1 không gian va chạm Collider, khi hai không gian Collider này “đụng” nhau, thì xảy ra va chạm
- Với ví dụ này thì chúng ta sẽ không xét tới chiều không gian z, vì đây là môi trường 2D.

Thành phần xử lý va chạm

Để thực hiện chúng ta sẽ gồm có 3 bước :

- **Bước 1** : Tạo ra không gian va chạm bao quanh object bằng cách thêm một component Collider cho object đó
- **Bước 2** : Viết code xử lý va chạm.
- **Bước 3**: Chạy thử demo.



Thành phần xử lý va chạm

Bước 1 : Tạo không gian Collider

- Ta có 2 đối tượng là hình tròn A và hình vuông B. Ta sẽ thêm collider 2D cho 2 đối tượng này bằng cách :
 - Chọn đối tượng (hình tròn và hình vuông)
 - Nhấn vào Component > Collider 2D
- Sau đó chọn 1 trong các loại collider 2d dưới đây :
 - Circle collider 2D: collider dạng hình tròn
 - Box collider 2D: collider dạng hình vuông
 - Edge collider 2D: collider dạng đoạn thẳng
 - Polygon collider 2D: collider dạng đa giác tùy biến
- Khi chọn xong thì không gian xung quanh của đối tượng sẽ được bao quanh bởi một khung màu xanh lá, là không gian Collider.

Thành phần xử lý va chạm

Bước 1 : Tạo không gian Collider

▪ **Chú ý :**

- Một đối tượng có thể có nhiều Collider 2D hoặc 3D nhưng không được phép đồng thời có cả Collider 2D lẫn Collider 3D.
- Để thay đổi kích thước nhanh chóng cho Collider, bạn có thể nhấn Shift và kéo thả các nút trên khung xanh lá
- Trên thanh Inspector của đối tượng lúc này sẽ xuất hiện thêm component collider 2D mà bạn vừa chọn.
- Ngoài ra, để có thể nhận sự kiện va chạm, bạn cũng cần add component Rigidbody cho từng object. Chú ý tới Gravity Scale bởi vì nó sẽ thiết lập lực hút khiến cho đối tượng tự động rơi xuống.

Thành phần xử lý va chạm

Bước 2 : Viết code xử lý va chạm.

- Hàm sự kiện va chạm phải được đặt trong đoạn script là component của object có Collider (xem hình).
- Ở ví dụ này, ta tạo mới script *Xu_Ly_Va_Cham_Hinh_Cau_A* cho hình cầu A và *Xu_Ly_Va_Cham_Hinh_Khoi_B* cho hình khối B
- Ở mỗi script của A và B, ta viết thêm các hàm sự kiện sau:

Thành phần xử lý va chạm

Bước 2 : Viết code xử lí va chạm.

```
using UnityEngine;
```

```
using System.Collections;
```

```
public class Collision_Box_B :  
    MonoBehaviour {
```

```
    // Use this for initialization
```

```
    void Start () { }
```

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

```
    void Update () { }
```

```
    void OnCollisionEnter2D() {
```

```
        print(gameObject.name + " OnCollisionEnter2D voi " +  
            col1.gameObject.name);}
```

```
    void OnTriggerEnter2D() {
```

```
        print(gameObject.name + " OnTriggerEnter2D voi " +  
            col2.gameObject.name);}
```

```
    }
```

Thành phần xử lý va chạm

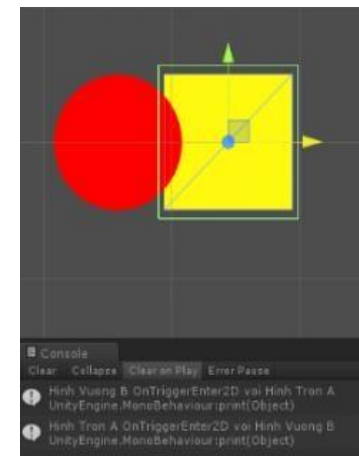
Bước 2 : Viết code xử lý va chạm.

Giải thích:

- **OnCollisionEnter:** Hàm sự kiện ứng với va chạm loại collision, khi hai Collider đều thuộc loại collision bắt đầu va chạm, hay nói cách khác là collider của object tiếp xúc với collider của object kia
- **OnTriggerEnter:** Hàm sự kiện ứng với va chạm loại trigger, khi hai Collider thuộc loại trigger bắt đầu va chạm, hay nói cách khác là collider của object này đi vào collider của object kia
- **OnTriggerExit:** Hàm sự kiện ứng với va chạm loại trigger, khi hai Collider thuộc loại trigger thoát khỏi va chạm, hay nói cách khác là collider của object này thoát khỏi collider của object kia.
- Đây là ba hàm thường dùng nhất, ngoài bạn có thể tham khảo thêm các hàm khác:
<http://docs.unity3d.com/ScriptReference/Collider2D.html>,
phần Messages.

Thành phần xử lý va chạm

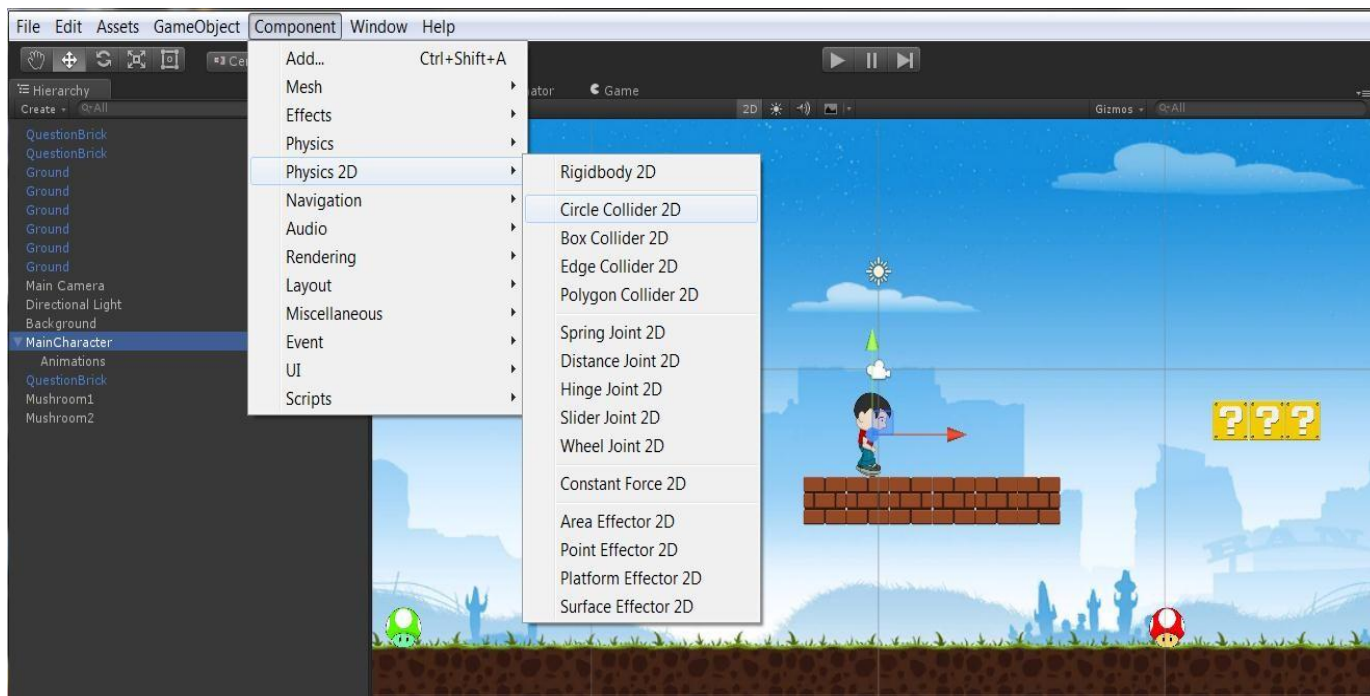
- **Bước 3: Chạy thử demo.**
- Để có ColliderEnter ở Object A: Object A:
 - Có Collider Component
 - Is Trigger = False
 - Ít nhất 1 object có Rigidbody component
 - Is Kinematic không được = True ở cả hai objectObject B:
 - Có Collider Component
 - Is Trigger = False
- Để có TriggerEnter hoặc TriggerExit ở Object A:
 - Cả 2 object có Collider Component
 - Ít nhất 1 object có Is Trigger = True
 - Ít nhất 1 object có Rigidbody component



Thành phần xử lý va chạm

Thành phần xử lý va chạm

- Ở Hierarchy, chọn đối tượng MainCharacter (đối tượng cần thêm) / **Menu / Component / Physics 2D / Circle Collider 2D**

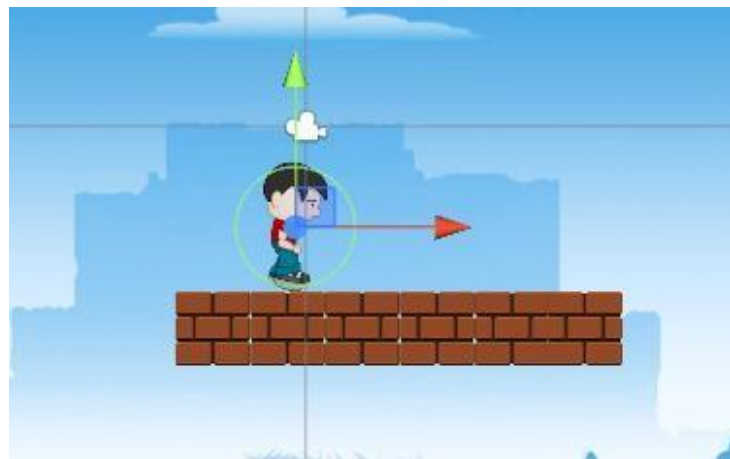


Thành phần xử lý va chạm

Thành phần xử lý va chạm

- Đối tượng sẽ có thêm thuộc tính Collider:
- Ta sẽ chọn tâm và bán kính để xác định vùng xử lý va chạm.
- Thuộc tính: Is Trigger: nếu chọn thì đối tượng chỉ dùng để xác định va chạm mà không ảnh hưởng bởi tác động vật lý.

Kết quả ta sẽ thấy như sau:



Thành phần xử lý va chạm

Tiếp theo ta tiến hành thêm thành phần Collider cho đối tượng Ground, lần này ta sẽ chọn Box Collider 2D thay vì Circle Collider 2D. Các thuộc tính cũng tương tự, ta có thể chỉnh sửa hình chữ nhật để xác định vùng va chạm.

Chú ý: ta chọn Circle Collider 2D cho đối tượng MainCharacter để tránh trường hợp nền (cái đối tượng Ground, có độ cao không đều) nhấp nhô dẫn đến các không di chuyển được nhân vật MC hay còn gọi là bị Stuck.

Thành phần xử lý va chạm

Bây giờ ta nhấn nút play để test thì sẽ thấy đối tượng rơi xuống, gặp cái dây đối tượng Ground thì đứng lại, và đối tượng bị nghiêng (Ta tưởng tượng có một chiếc bánh xe hình tròn rơi xuống mặt đất, nó sẽ lăn :D). Để tránh đối tượng nghiêng này chúng ta sẽ tick vào thuộc tính Fixed Angle của nhân vật MainCharacter.



Thành phần xử lý va chạm

Ta sẽ tìm hiểu thêm phương thức:

```
void FixedUpdate()  
{  
  
}
```

--> Các tính toán, tương tác vật lý, chúng ta sẽ đặt trong hàm này, ví dụ như AddForce, etc (Chi tiết

<http://docs.unity3d.com/Documentation/ScriptReference/MonoBehaviour.FixedUpdate.html>)

Thành phần xử lý va chạm

```
void OnCollisionEnter2D()  
{  
  
}
```

--> Hàm này được gọi khi có hai đối tượng va chạm nhau.

```
void OnTriggerEnter2D()  
{  
  
}
```

--> Hàm này được gọi khi có hai đối tượng va chạm nhau, trong đó có 1 hoặc cả hai đối tượng là Trigger. Các bạn có thể tham khảo thêm chi tiết nhiều hàm khác ở đây: <http://docs.unity3d.com/Documentation/ScriptReference/MonoBehaviour.html>

Thành phần xử lý va chạm

Sau khi đã thêm phần tính toán va chạm và vật lý, ta sẽ cập nhật lại các điều khiển đối tượng bằng cách sử dụng tác dụng vật lý để làm đối tượng di chuyển, và nhảy. Khai báo thêm hai thuộc tính `movingForce` và `jumpForce` trong `MainCharacterBehavior` để lưu trữ giá trị lực khi nhảy và khi di chuyển.



THANKS FOR
WATCHING!

