# Làm Game 2D bằng Unity - Phần 6 - Text, Particle System và chuyển đổi màn chơi

22 February 2014 at 00:10

Chào các bạn,

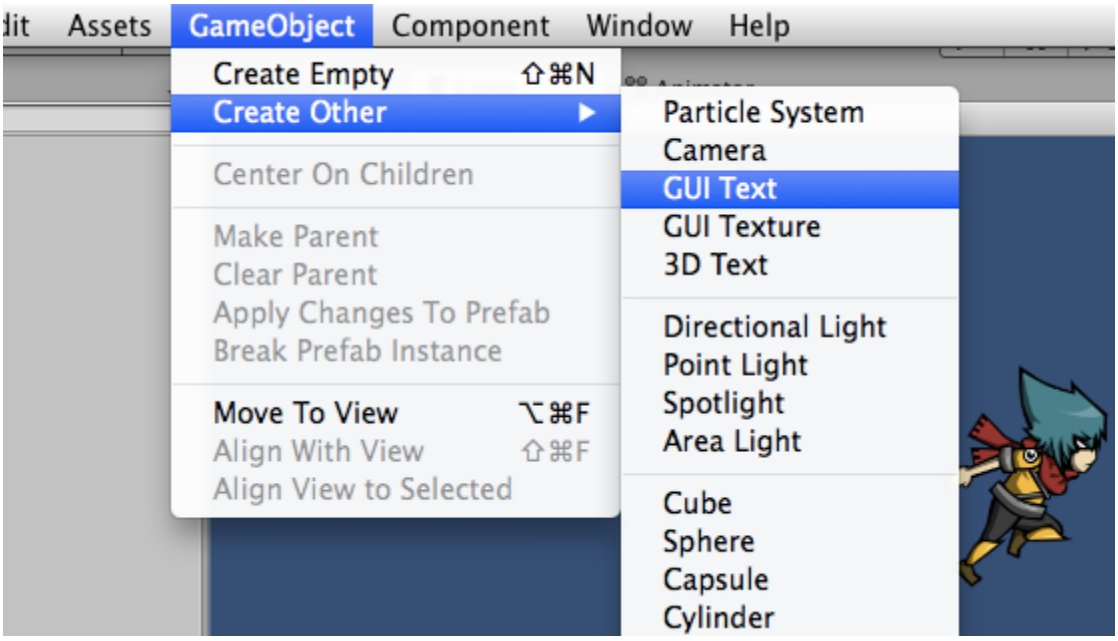
Phần trước mình đã tìm hiểu cách thêm thành phần vật lý và xử lý va chạm rồi, phần này mình sẽ trình bày các đưa text để hiển thị thông tin, sử dụng các particle system và cách chuyển đổi màn chơi trong Unity.

Trước khi đi tiếp, nếu bạn nào chưa đọc các phần trước, ta có thể tham khảo lại ở đây
**Link phần 5:** https://www.facebook.com/notes/hội-lập-trình-viên-game-đà-nẵng/làm-game-2d-bằng-unity-phần-5-thành-phần-vật-lý-và-xử-lý-va-chạm/246753422163594

**Link dự án mẫu phần 6:** https://www.dropbox.com/sh/71xkikzqw1zr3iu/Kk23e2qzU1

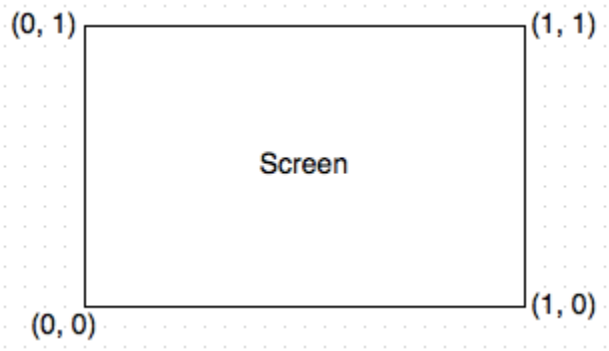
**III. Tạo các đối tượng cơ bản**
**1. Game Object**
**2. Sprite**
**3. Animation**
**4. Prefab**
**5. Script và điều khiển máy trạng thái**
**6. Thành phần vật lý và xử lý va chạm**
**7. Sử dụng text**

Từ chọn **Menu/Game Object/Create Other/Gui Text**



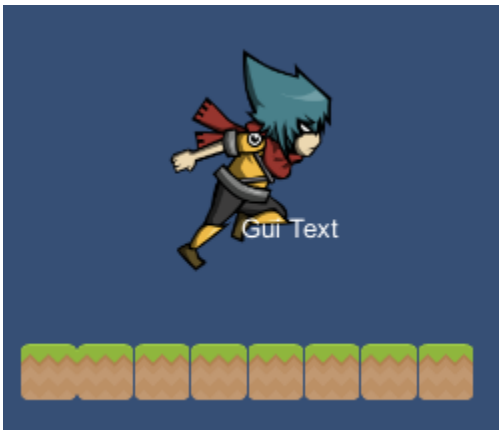

Ta sẽ thấy ở cửa sổ Hierarchy thêm một đối tượng tên là GUI Text, chọn đối tượng này ta sẽ thấy ở cửa sổ Inspector có các thuộc tính như sau:


Nhìn vào các thuộc tính ta có thể dễ dàng đoán được chức năng của từng thuộc tính, một lưu ý quan trọng là giá trị position của GUI Text chỉ nhìn thấy nếu nằm trong đoạn [0, 1]. Với quy ước góc như hình vẽ:




Có nghĩa rằng nếu bạn đặt text ở vị trí (0.5, 0.5) thì lúc chạy game lên, đoạn text này lúc nào cũng nằm ở giữa màn hình.

Để xem hiển thị trực quan chúng ta chuyển qua tab Game, sẽ thấy text được hiển thị đầy đủ và đúng như mô tả ở hình trên.

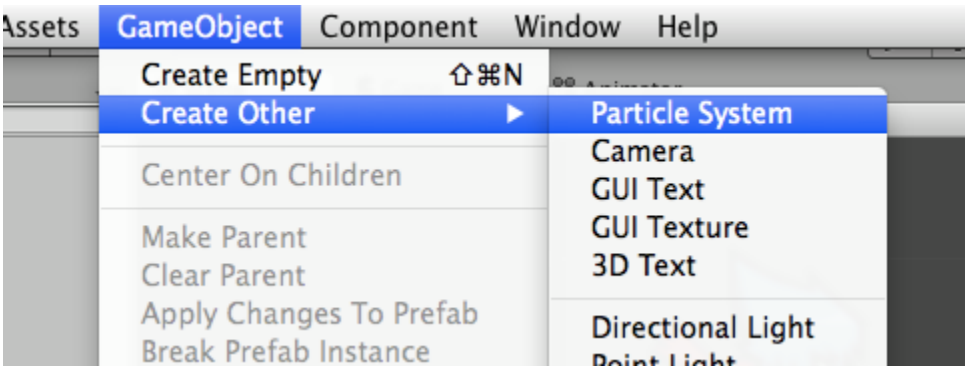



Vậy bây giờ bạn đã biết cách sử dụng GUI Text để hiển thị thông tin lên màn hình.

**8. Sử dụng Particle System**

Particle System là một trong những kỹ thuật tạo ra các hiệu ứng cháy nổ hay sương, khói... được sử dụng rất thường xuyên trong game. Các bạn có thể google để tìm hiểu thêm về Particle system và cách tạo Particle System. Ở đây mình chỉ hướng dẫn sử dụng các Particle System có sẵn.

Các bạn chọn Menu/Game Object/Create Other/Particle System như hình sau:



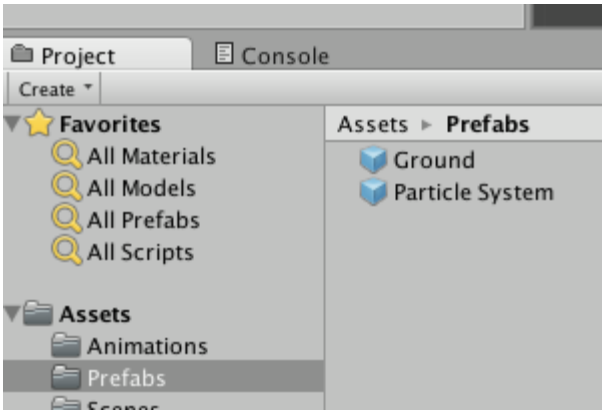

Một đối tượng Particle System sẽ được xuất hiện ở cửa sổ Hierarchy


Tiếp theo các bạn thêm một Script cho đối tượng Particle System này, và đặt tên là ParticleSystemBehaviour.cs

Và edit nội dung của nó thành như sau:

```
3
4 public class ParticleSystemBehaviour : MonoBehaviour {
5
6     // Use this for initialization
7     void Start () {
8         Destroy (gameObject, 3);
9     }
0
1     // Update is called once per frame
2     void Update () {
3
4     }
5 }
6
```

Destroy(gameObject, 3) --> có nghĩa rằng đối tượng này sau khi xuất hiện 3s sẽ tự động huỷ.
Chúng ta có thể nhấn nút Play rồi đợi 3s để xem kết quả.
Tiếp theo ta kéo đối tượng Particle System vào thư mục Prefabs để toạ prefab cho đối tượng này.




Sau đó ta có thể xoá đối tượng Particle System này ở cửa sổ Hierarchy đi, lúc nào có va chạm ta mới sinh ra một đối tượng Particle System này.

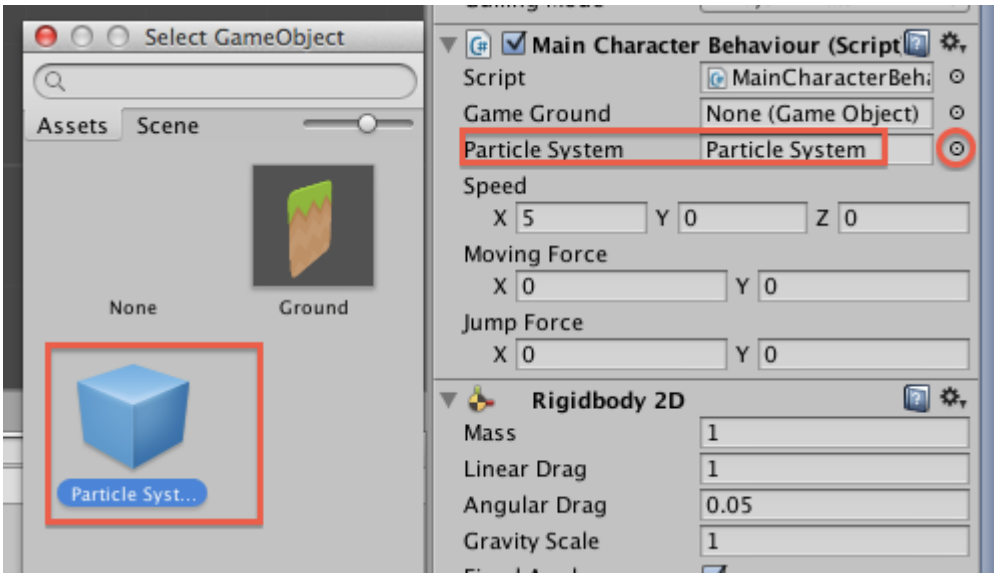
Tiếp theo ở MainCharacterBehaviour.cs ta khai báo thêm một thuộc tính là particleSystem như sau:

```
public GameObject gameGround;
public GameObject particleSystem;
public Vector3 speed;

public Vector2 movingForce;
public Vector2 jumpForce;
```

Chọn đối tượng MainCharacter, ở cửa sổ Inspector, thành phần Script ta sẽ thấy thêm một thuộc

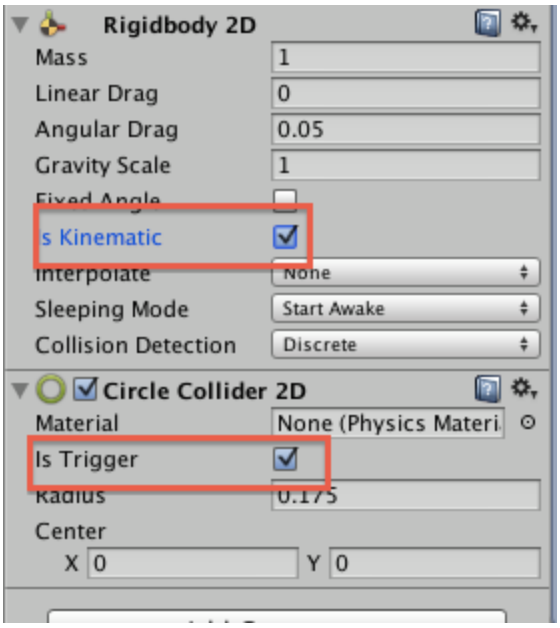
tính đó là Particle System như hình:



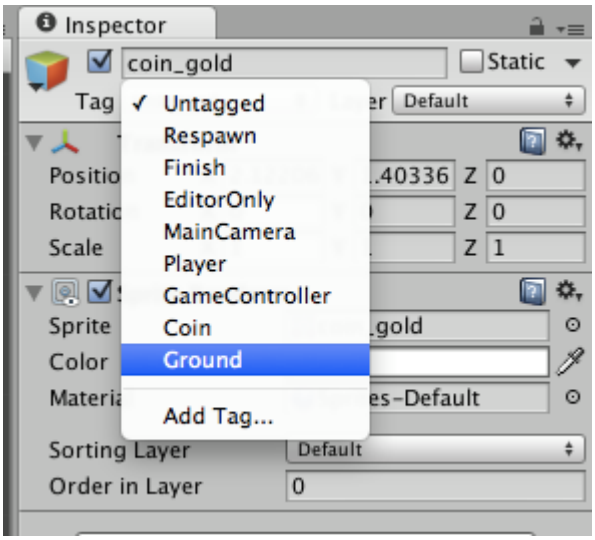

Ta chọn vào nút khoan tròn màu đỏ, sau đó một cửa sổ các Prefabs sẽ hiện ra, ta chọn cho nó là đối tượng Particle System .

Tiếp theo, ta sẽ thêm Sprite đồng tiền, để khi đối tượng chạm vào đồng tiền (như các trò chơi thường thấy ta sẽ thêm điểm cho người chơi.

Ta phải thêm đầy đủ các thành phần vật lý, xử lý va chạm và một điều quan trọng nữa là thêm tag  cho đồng tiền.

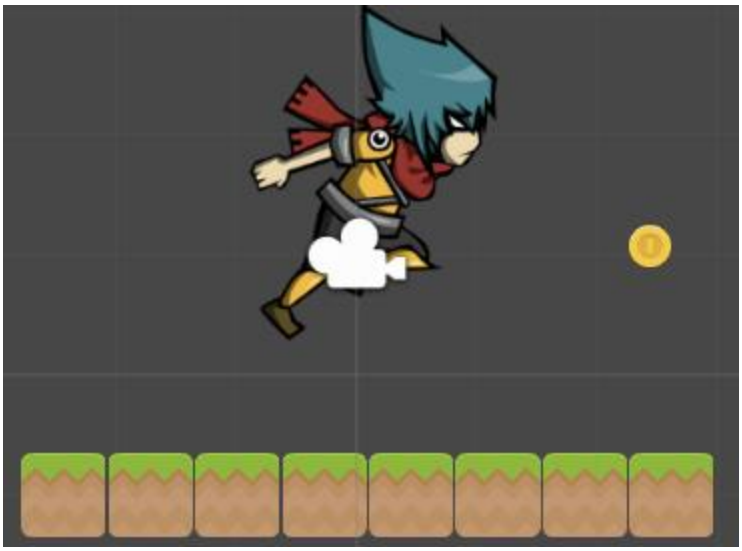


**Chú ý:** là ta sẽ chọn: Is Kinematic và Is Trigger cho đối tượng đồng tiền.




**Chú ý:** Nếu chưa có tag "Coin" ta có thể chọn Add Tag rồi thêm.

Ta đặt đồng tiền ở một vị trị sao cho khi MC chạy tới thì có thể va chạm với đồng tiền.



Tiếp theo, ở MainCharacterBehaviour.cs ta sẽ thêm đoạn xử lý này:

```
void OnCollisionEnter2D(Collision2D other)
{
    //other.gameObject
    //Debug.Log ("OnCollisionEnter2D with object has tag = " + other.gameObject.tag);
}
void OnTriggerEnter2D(Collider2D other)
{
    //other.gameObject
    Debug.Log ("OnTriggerEnter2D with object has tag = " + other.gameObject.tag);
    if (other.gameObject.tag == "Coin")
    {
        Destroy(other.gameObject);
        Instantiate(particleSystem,
                    other.gameObject.transform.position,
                    other.gameObject.transform.localRotation);
    }
}

void OnMouseDown()
{
```

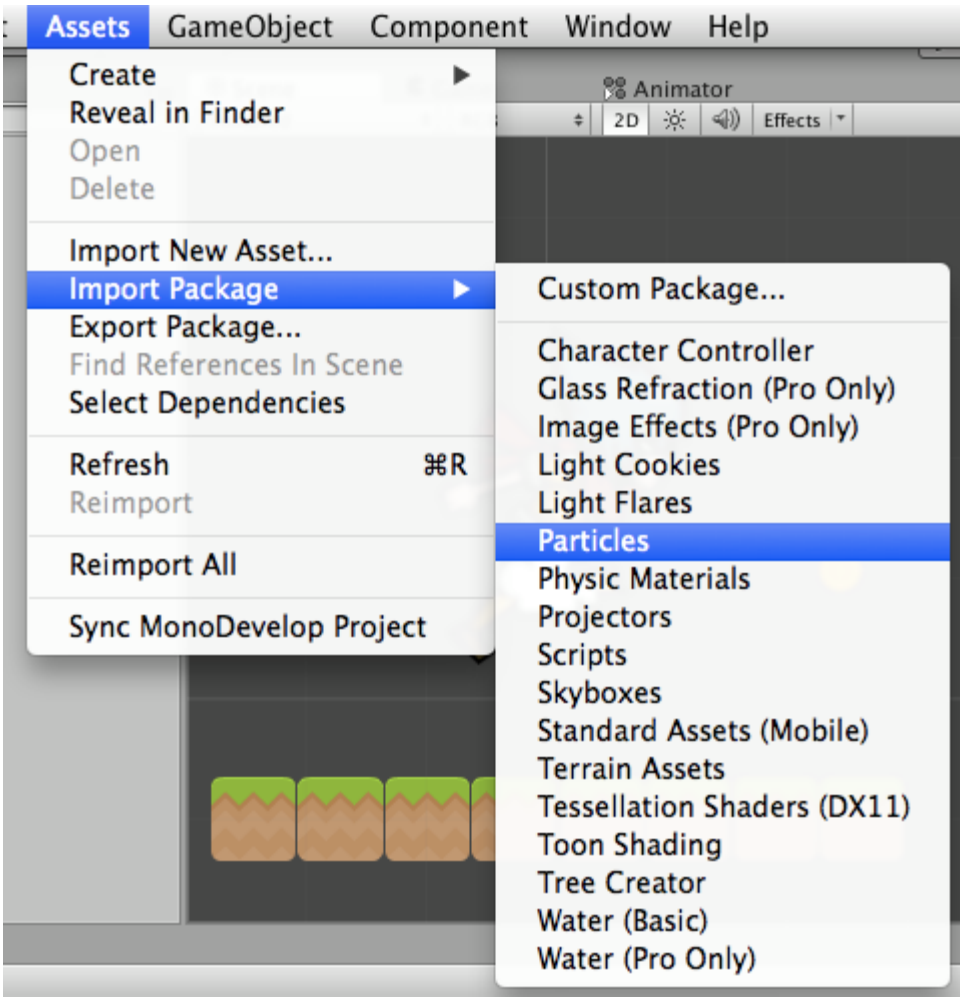
Như vậy mỗi khi có va chạm giữa MainCharacter với một trigger nào đó, hàm này sẽ được gọi và chúng ta sẽ kiểm tra nếu đối tượng va chạm có Tag là "Coin" ta sẽ xoá đối tượng coin_gold đi bằng lệnh Destroy(other.gameObject) và đặt vào tại đó một đối tượng Particle System bằng lệnh **Instantiate** (Đối tượng Particle System này sau 3s sẽ biến mất theo như đã thiết lập ở trên).

Ta có thể nhấn nút Play để kiểm tra lại kết quả.

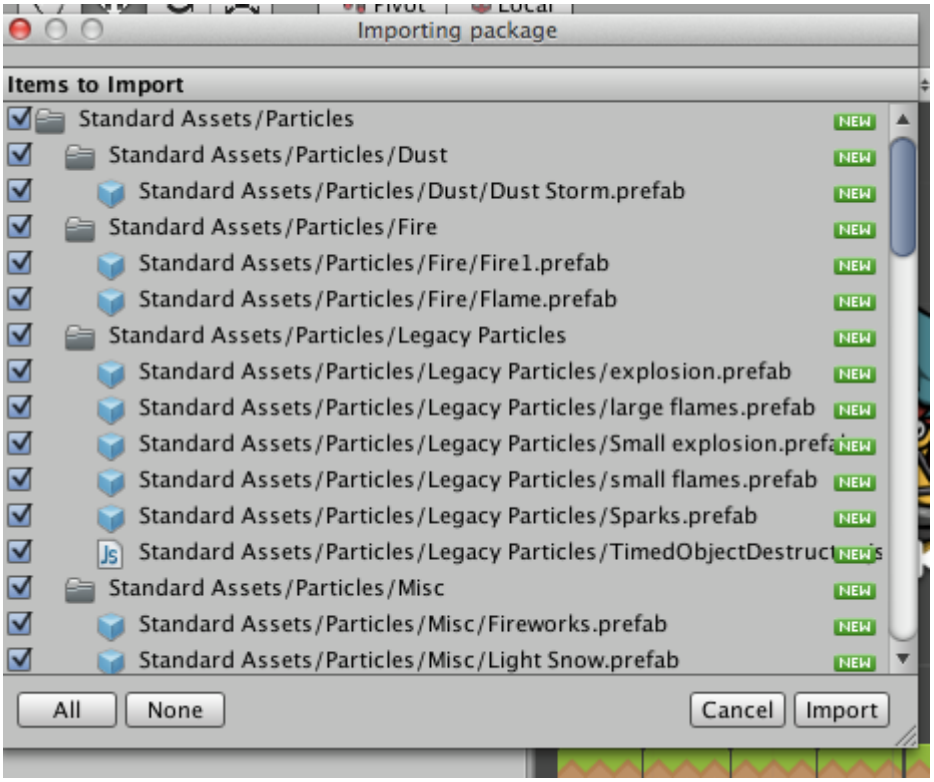
Ở trên ta chỉ sử dụng đối tượng Particle có sẵn, để đảm bảo rằng bạn nào cũng có thể làm được, để cho đẹp hơn chúng ta có thể mua thêm các Particle System khác hoặc sử dụng các Particle System free trên Internet hoặc cộng đồng Unity chia sẻ.

Ta import các gói assets free của Unity như sau:

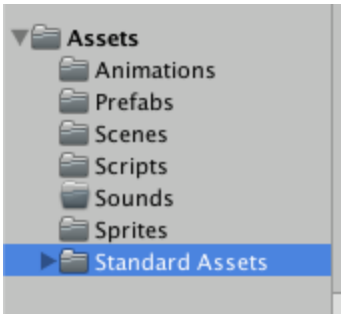
Bước 1:




Bước 2: chọn các asset cần, và nhấn nút Import




Bước 3: các assets sẽ được import vào thư mục như sau:

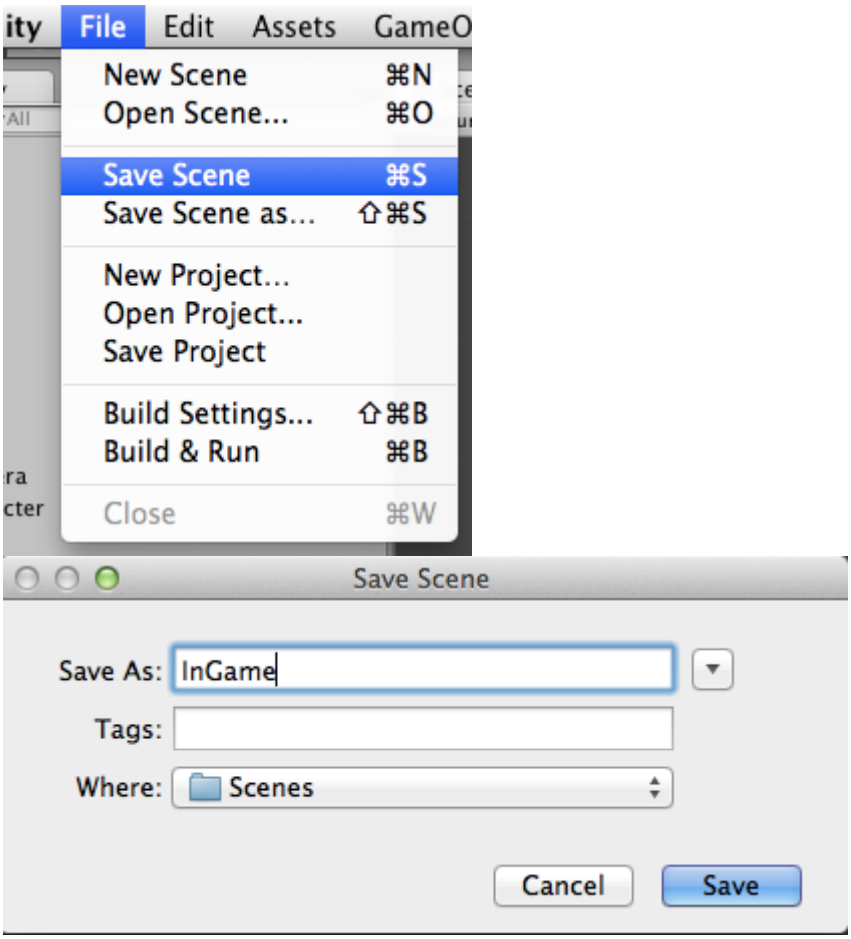



Sau đó bạn chỉ việc kéo thả các Asset này thay vì sử dụng đối tượng Particle System có sẵn.

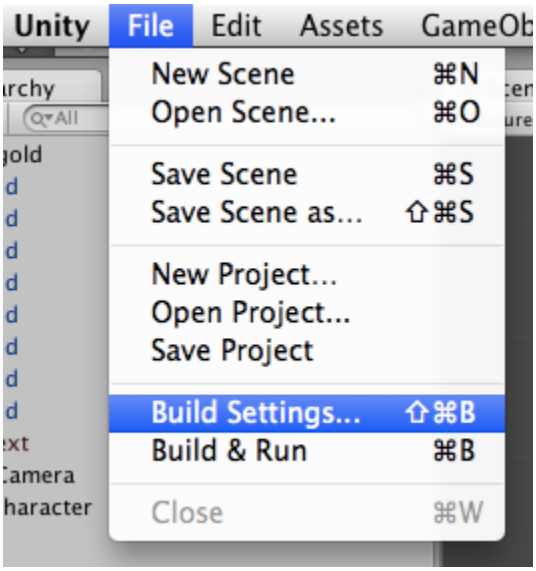
### 9. Chuyển đổi màn chơi

Trong một game sẽ có nhiều màn chơi, hoặc nhiều cảnh game, lấy một ví dụ đơn giản khi đối tượng rơi xuống (hoặc hết máu hay gì đó) thì game sẽ kết thúc và hiện ra màn hình thông báo là Game Over chẳng hạn.

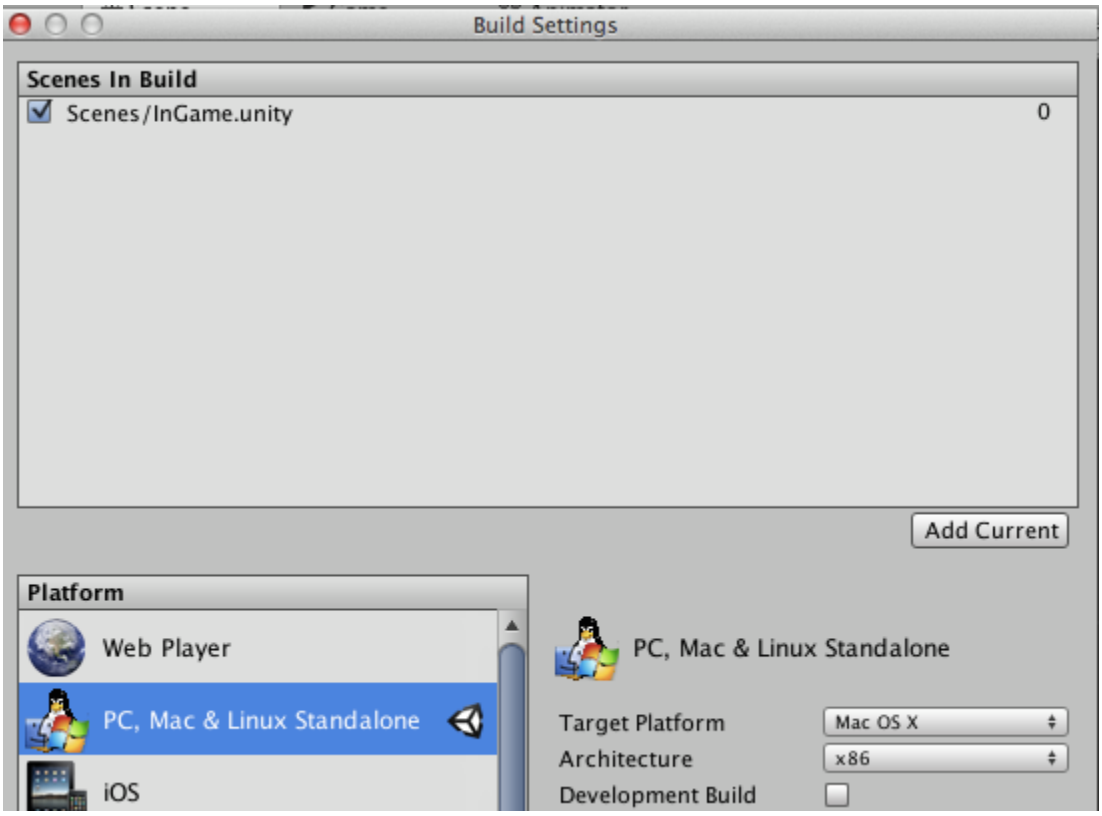
Đầu tiên, ta save Scenes hiện thời lại và đặt tên là InGame.



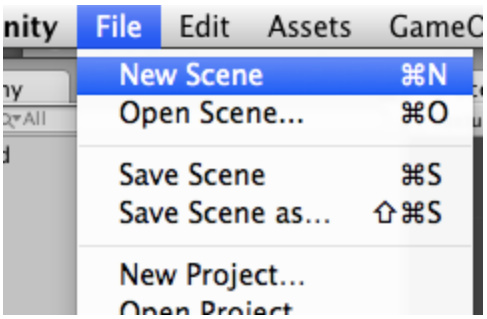

Để dễ quản lý ta sẽ lưu trong thư mục Scenes của thư mục Assets.
Tiếp theo ta vào Build Setting:

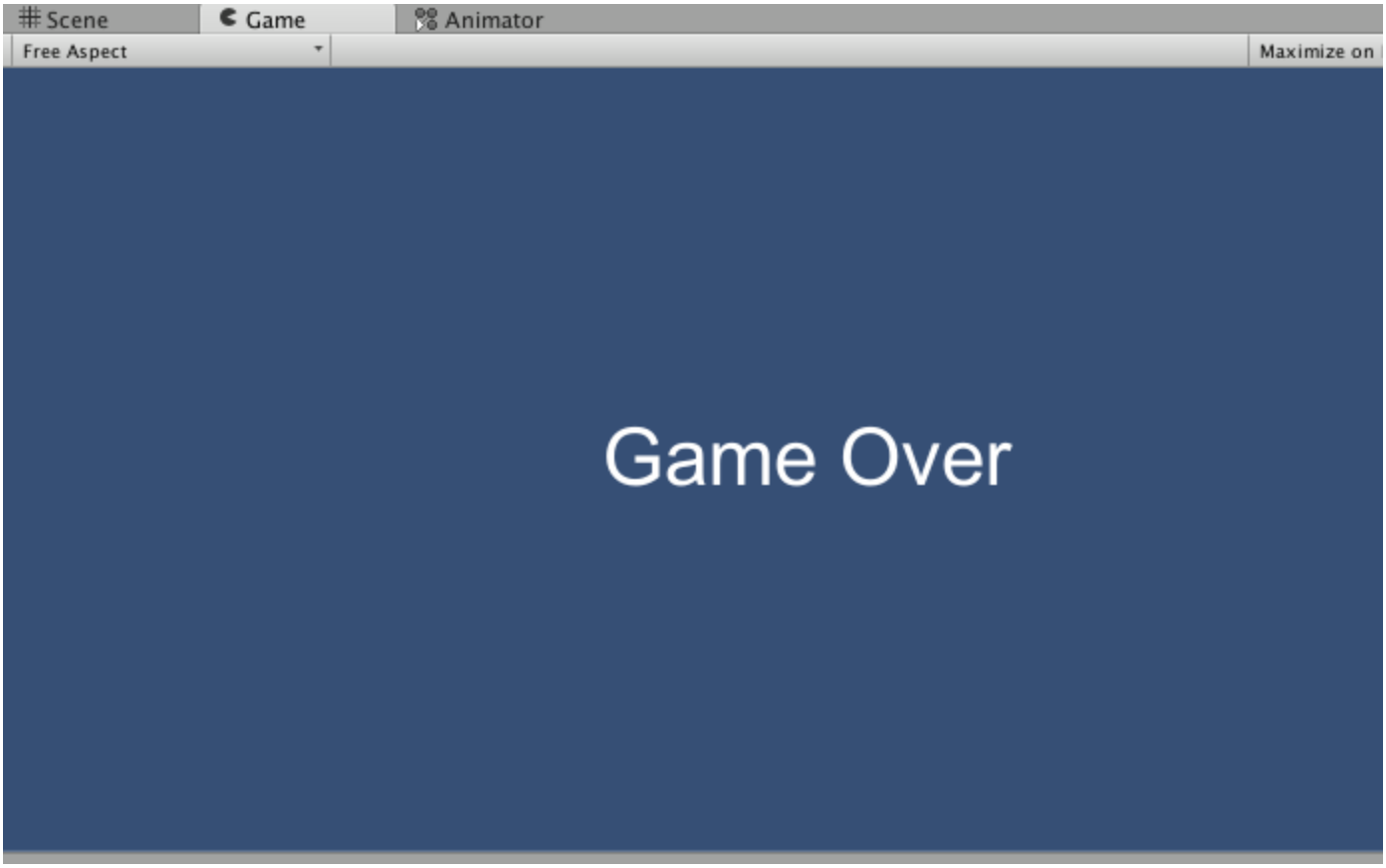


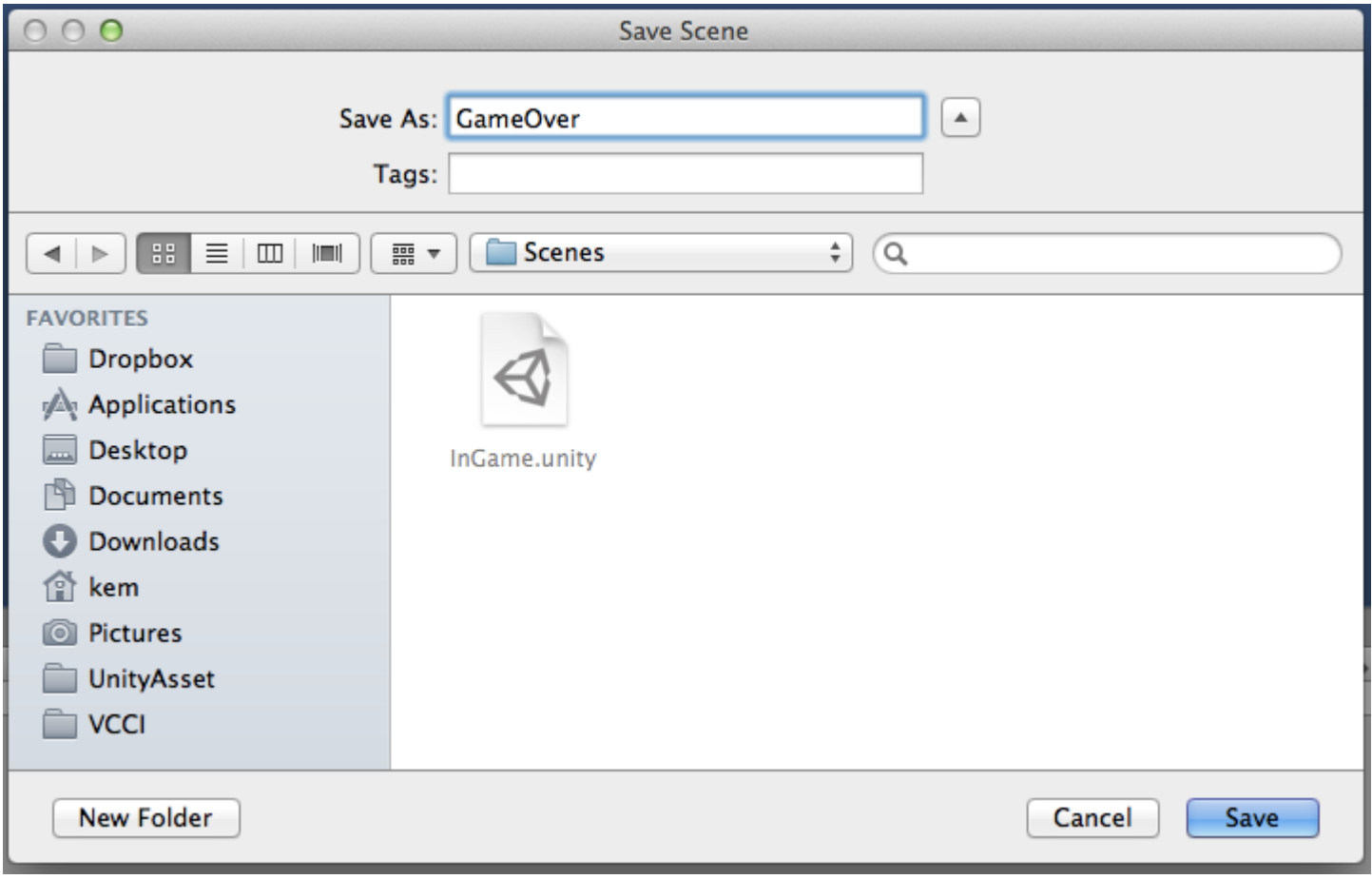
Một cửa sổ mới hiện ra, ta chọn Add Current.




Tắt cửa sổ mới hiện ra, tiếp theo các bạn tạo mới một Scenes, lưu lại với tên là GameOver

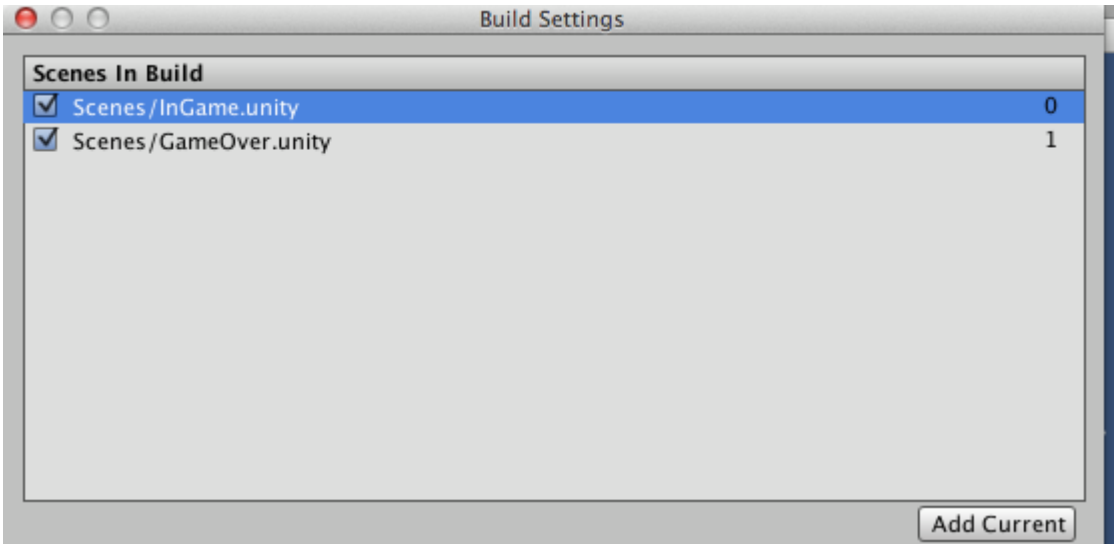



Scene
Game
Animator
Free Aspect
Maximize on
Game Over

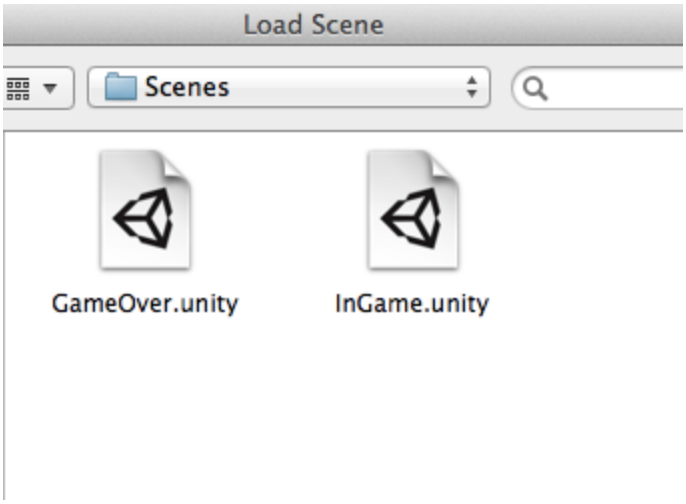



Tiến hành vào Build Setting, thêm Scenes GameOver vào.

**Chú ý:** Scenes nào cần hiển thị đầu tiên thì ta sẽ thêm vào Build Setting đầu tiên, hoặc chúng ta có thể kéo thả ngay tại cửa sổ Build Settings.



Bây giờ ta có hai Scenes là InGame và GameOver. Tiến hành Save lại, sau đó Open Scenes InGame lại.




Xử lý chuyển đổi màn chơi:

Ở MainCharacter, ta sẽ thêm đoạn lệnh này ở hàm Update:

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

    if (gameObject.transform.position.y < -3.0f)
    {
        Application.LoadLevel("GameOver");
    }

}
```

Ta sẽ điều khiển nhân vật ra ngoài nền, để đối tượng rơi xuống khi giá trị y của position < -5 game sẽ tự động chuyển qua màn hình GameOver.

Như vậy cơ bản, các bạn đã tìm hiểu và biết cách sử dụng các đối tượng cơ bản trong Unity để tự tạo cho mình một game 2D đơn giản hay phức tạp. Phần tiếp theo mình sẽ hướng dẫn sử dụng âm thanh, các cách điều khiển camera và "Design Pattern" trong game hay các mẫu thiết kế lớp để sử dụng trong quá trình làm game.