Bài 2: Design pattern trong Game 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. Hệ thống UI
8. Sử dụng Particle System
9. Chuyển đổi màn chơi
10. Sound
11. **Design Pattern trong Game**

Design Pattern trong Game

- Design Pattern hay mẫu thiết kế, là cách xây dựng một lớp dựa trên một thiết kế nhất định nào đó. Một Pattern được sử dụng nhiều trong game đó là Singleton.
- Vậy khi nào thì sử dụng Singleton ? - Singleton sử dụng khi trong chương trình chỉ tồn tại duy nhất một instance (thể hiện or đối tượng) của một lớp nào đó.
- Ví dụ: Ta có lớp SoundManager để quản lý tất cả sound cho game, thì suốt chương trình chỉ cần duy nhất 1 thể hiện của lớp SoundManager để quản lý âm thanh trong game.
- Như vậy ta sẽ cài đặt lớp SoundManager theo mẫu Singleton.

Design Pattern trong Game

- Về cơ bản mẫu Singleton được thiết kế như sau:
- Hàm dựng (constructor) để là private, để đảm bảo không tạo được đối tượng từ bên ngoài lớp, chỉ cho phép tạo đối tượng từ hàm GetInstance thôi.
- Câu lệnh instance = this; để đảm bảo biến instance được trỏ đến đối tượng duy nhất vừa mới tạo ra.
- Như vậy ở bất cứ đâu trong dự án, muốn điều khiển Sound, bật tắt bài nào đó ta chỉ việc gọi SoundManagerBehaviour.GetInstances().Method() là được.

Design Pattern trong Game

Cách truyền một giá trị từ script này sang script khác.

- Nhu cầu truyền một giá trị từ script này sang script khác rất thường xuyên. Ở Unity một script được coi là một component của một GameObject, vì vậy để làm việc này ta chỉ cần xác định được đối tượng GameObject chứa script cần truyền giá trị, sau đó từ đối tượng này ta sẽ truy xuất giá trị cần thiết thông qua component script.
- Ví dụ: ở Script SoundManagerBehaviour ta muốn truy xuất đến các giá trị ở Script MainCharacterBehaviour ta làm như sau:
- Đầu tiên ta sẽ đặt cho đối tượng MainCharacter tag là: Player (Xem lại phần 2 để biết cách đặt tag).

Design Pattern trong Game

Cách truyền một giá trị từ script này sang script khác.

Chú ý:

- Lệnh **mainCharacter = GameObject.FindGameObjectWith Tag("Player");** cần phải đặt ở hàm Start (hàm này chỉ gọi một lần khi khởi tạo xong đối tượng),
- Nếu lệnh này đặt ở ***Update/FixedUpdate/Render***... thì sẽ làm chậm chương trình và không tối ưu. Vì quá trình tìm kiếm một đối tượng dựa vào tag hay vào một tiêu chí nào đó sẽ mất rất nhiều thời gian.

Design Pattern trong Game

Cách truyền một giá trị từ script này sang script khác.

- Sau khi tìm thấy đối tượng chúng ta có thể truy xuất, thay đổi các thành phần public của script đó bằng cách:

```
if (mainCharacter != null)

{

MainCharacterBehaviour script =
mainCharacter.GetComponent< MainCharacterBehaviour>();

script.speed = new Vector3(1, 0, 0); // access and change
speed va lue of main character

}
```

- Hoặc bạn có thể cài đặt đối tượng MainCharacter theo Singleton như hướng dẫn ở trên, vì thông thường trong một game thì chỉ có 1 MainCharacter.

Camera

- Camera trong Unity cũng được dùng để hiển thị game trên thế giới cho người chơi. Nó cũng được coi là một Game Object trong Unity.
- Có thể xoay, di chuyển.... tùy chỉnh nó theo ý tưởng.
- Camera được sử dụng để hiển thị cảnh trong game, chúng ta có thể làm cho game của mình trở nên độc đáo hơn nhờ tùy chỉnh Camera.
- Trong một cảnh, chúng ta có thể có một hoặc rất nhiều Camera.

Camera

Các thuộc tính Camera:

- ***Clear Flags:*** Xác định các bộ phận mà màn hình sẽ bị xóa. Thuận tiện khi sử dụng nhiều máy ảnh và để vẽ nhiều đối tượng khác nhau. Không xóa nó sẽ hiển thị màu đen xì.
- ***Background:*** Màu nền cho phần màn hình còn lại.

Camera

Các thuộc tính Camera:

- ***Culling Mask:*** Chỉ định các lớp đối tượng của bạn trong Inspector. Cho phép hoặc bỏ qua các đối tượng được hiển thị trong Camera.
- **Projection:**
  - ***Perspective***: Camera hiển thị các đối tượng theo phối cảnh trọn vẹn.
  - ***Orthographic:*** Hiển thị các đối tượng như một thể thống nhất, không có theo nghĩa của phối cảnh(Perspective).
- ***Size***: Kích thước quan sát của Camera khi chọn phép chiếu là: Orthographic.
- ***Field of view***: Chiều rộng của góc nhìn Camera. Được đo bằng độ dọc theo trục Local Y.

Camera

Các thuộc tính Camera:

- **Clipping plane:**
  - ***Near***: Khoảng cách gần nhất hiển thị trong Camera.
  - ***Far***: Khoảng cách xa nhất hiển thị được trong tầm nhìn của Camera.
- **Normalized View Port Rect:**
  - ***X:*** Bắt đầu từ vị trí ngang mà Camera hiển thị.
  - ***Y***: Bắt đầu từ vị trí thẳng đứng mà Camera hiển thị.
- ***Depth:*** Vị trí. Camera có một giá trị lớn hơn sẽ được hiển thị lên đầu. Tức là cái nào có giá trị lớn hơn thì nó sẽ được ưu tiên được hiển thị lên màn hình.

Camera

Các thuộc tính Camera:

- ***Rendering path:*** Tùy chọn cho việc xác định phương pháp vẽ những gì sẽ được hiển thị lên Camera.
  - ***Use Player Setting:*** Sử dụng cài đặt Player.
  - ***Vertex Lit:*** tất cả các đối tượng được đưa ra bởi máy ảnh sẽ được trả lại như đối tượng Vertex-Lit.
  - ***Forward:*** tất cả các đối tượng sẽ được trả về như một tài liệu. Giống như tiêu chuẩn Unity 2.x.
  - ***Deferred Lighting*** (Unity Pro only): Mất phí, không chơi. Tất cả các đối tượng sẽ được rút ra mà không có ánh sáng, sau đó tất cả ánh sáng được trả lại trong hàng đợi Render.
  - ***Target Texture:*** Phiên bản free 4.0 đã có. Tham chiếu đến một texture. Cho phép dựng hình High Dynamic Range cho máy ảnh này.


THANKS FOR
WATCHING!