Event functions
Scripts trong Unity3D của chúng ta được viết theo hướng sự kiện, hầu hết các hàm trong MonoBehaviour đều là event callbacks, và được gọi sau khi có một sự kiện nào đó xảy ra.

Mình sẽ chia các loại sự kiện ra thành một vài loại chính để bạn dễ hình dung: các sự kiện khởi tạo, sự kiện vật lý, sự kiện input – nhận phím từ người dùng chẳng hạn.

Unity sẽ "trao" quyền điều khiển các events này cho chúng ta bằng các hàm đặc biệt gọi là Event Functions. Số lượng Event Functions khá nhiều, vì vậy trong bài viết mình chỉ nêu các hàm cơ bản và phổ biến:

Sự kiện khởi tạo: các hàm mà mình đã đề cập ở bài trước Awake() và Start().
Sự kiện chuyển frame: khi chuyển frame, các hàm Update sẽ được gọi bao gồm Update và LateUpdate, trong đó các sự kiện vật lý sẽ gọi ở FixedUpdate().
Sự kiện input: bạn có thể sử dụng hàm OnMouseDown(), OnMouseUp() hay OnMouseOver() ở các gameObject có Collider để phát hiện khi người dùng click vào vật đó.
Sự kiện vật lý: điển hình là các hàm xử lý va chạm như OnCollisionEnter, OnTriggerEnter,… mình đã đề cập ở bài trước.
Đấy là những hàm "điển hỉnh" và sử dụng khá nhiều mà mình muốn nêu ra, tuy nhiên có rất rất nhiều các hàm mình không liệt kê, để biết thêm các bạn có thể đọc Unity Documentation nhé.

Tạo gameObject bằng Prefab
Mình đã giới thiệu về prefab các bài trước đó, prefab là một gameObject được cache lại và có thể thể tạo ra các bản sao của nó tại thời điểm game đang chạy, ví dụ như gameObject cây cối, nhà cửa,…

Các prefabs được lưu trữ ở trong Assets folder, bạn có thể đặt nó trong folder Resources để không cần phải tham chiếu tới prefab bằng cách kéo thả.

"Folder Resources là gì?"

Resources folder là một loại thư mục đặc biệt trong Assets (Assets/Resources), các files trong thư mục này có thể load lên lúc game chạy.

Tham khảo: Một số folder đặc biệt trong Unity

Ứng dụng
Giả sử bạn có một gameObject tàu chiến (RocketShip) và gameObject này có thể bắn ra các tên lửa (Missile).

Đầu tiên bạn cần có prefab Missile, hãy tạo một gameObject tên là Missile trên tab Hierachy, kéo thả từ gameObejct này từ tab Hierachy xuống tab Project. Như vậy bạn đã có một prefab tên là Missile.

Để hiểu rõ về folder Resources, hãy tạo một folder tên là Resources và kéo prefab Missle vào trong thư mục này.

Khi RocketShip của bạn muốn bắn ra các Missile thì chúng ta cần tham chiếu tới prefab Missile trong folder Resources như sau:

```
public class RocketShipController : MonoBehaviour
{
    public GameObject mPrefab;

    void Start()
    {
        mPrefab = Resources.Load("Missile") as GameObject;
    }
}
```

Như vậy prefab Missile đã được tham chiếu thành công, việc tiếp theo là mỗi khi RocketShip bắn hay gọi hàm Shoot() chúng ta cần tạo ra một bản copy của Missile bằng hàm Instantiate(..):

```
void ShootMissile()
{
    GameObject newMissile = Instantiate(mPrefab, transform.position, transform.rotation);

    // ... Xử lý với newMissile
}
```

Đấy, như vậy là bạn đã có một gameObject được “sinh ra” trong lúc game chạy.

Coroutines
Coroutines giúp bạn thực thi một hàm qua nhiều frames khác nhau, ví dụ mình muốn in ra số 1, sau đó 1 giây sau in ra số 2, tiếp theo 2 giây sau in ra số 3.

Việc implements thuật toán với hàm Update cần phải xử lý với deltaTime, tracking số giây hiện tại,…

Sử dụng Coroutine rất đơn giản:

```
IEnumerator Counting()
{
    Debug.Log("1");
    yield return new WaitForSeconds(1.0f); // chờ 1s

    Debug.Log("2");
    yield return new WaitForSeconds(2.0f); // chờ 2s

    Debug.Log("3");
}
```

Để hiểu rõ hơn về Coroutines, bạn có thể đọc bài viết Coroutine trong Unity là gì? Có liên quan tới Threads hay không?