(Event Functions)
Các hàm sự kiện (Event Functions) trong Unity không chạy liên tục như một chương trình truyền thống, mà Unity gọi các hàm cụ thể đã được khai báo trong script tại các điểm thời gian khác nhau. Khi một hàm hoàn thành việc thực thi, điều khiển được chuyển trở lại cho Unity. Các hàm này được gọi là hàm sự kiện vì chúng được kích hoạt bởi Unity để phản ứng với các sự kiện xảy ra trong quá trình chơi game. Unity sử dụng một quy tắc đặt tên để xác định hàm nào được gọi cho một sự kiện cụ thể. Dưới đây là một số sự kiện phổ biến và quan trọng nhất.

Các sự kiện cập nhật thường xuyên:
Một trò chơi tương tự như một hoạt hình, trong đó các khung hình hoạt hình được tạo ra khi chạy. Một khái niệm quan trọng trong lập trình game là thay đổi vị trí, trạng thái và hành vi của các đối tượng trong game ngay trước khi mỗi khung hình được hiển thị. Hàm Update() là nơi chính để viết code như vậy trong Unity. Update() được gọi trước khi khung hình được hiển thị và trước khi các hoạt ảnh được tính toán.

```
void Update()
{
   float distance = speed * Time.deltaTime * Input.GetAxis("Horizontal");
   transform.Translate(Vector3.right * distance);
}
```

Bộ máy vật lý cũng được cập nhật theo từng bước thời gian rời rạc tương tự như khung hình được hiển thị. Một hàm sự kiện riêng gọi là FixedUpdate() được gọi trước mỗi cập nhật vật lý. Vì các cập nhật vật lý và cập nhật khung hình không xảy ra với cùng tần suất, kết quả chính xác hơn từ mã vật lý nếu bạn đặt nó trong hàm FixedUpdate() thay vì Update().

```
void FixedUpdate()
{
   Vector3 force = transform.forward * driveForce * Input.GetAxis("Vertical");
   rigidbody.AddForce(force);
}
```

Đôi khi cũng hữu ích khi bạn muốn thực hiện các thay đổi bổ sung sau khi các hàm Update() và FixedUpdate() đã được gọi cho tất cả các đối tượng trong cảnh và sau khi tất cả các hoạt ảnh đã được tính toán. Một ví dụ là khi một camera nên tiếp tục nhìn vào một đối tượng mục tiêu; việc điều chỉnh hướng của camera phải được thực hiện sau khi đối tượng mục tiêu đã di chuyển. Một ví dụ khác là khi mã script cần ghi đè lên hiệu ứng của một hoạt ảnh (ví dụ: làm cho đầu nhân vật nhìn vào một đối tượng mục tiêu trong cảnh). Hàm LateUpdate() có thể được sử dụng cho những tình huống như thế này.

```
void LateUpdate()
{
   Camera.main.transform.LookAt(target.transform);
}
```

Sự kiện khởi tạo:
Thường thì cần gọi mã khởi tạo trước bất kỳ cập nhật nào xảy ra trong quá trình chơi game. Hàm Start() được gọi trước khung hình hoặc cập nhật vật lý đầu tiên trên một đối tượng.

Hàm Awake() được gọi cho mỗi đối tượng trong cảnh khi cảnh được tải. Lưu ý rằng mặc dù các hàm Start() và Awake() của các đối tượng khác nhau được gọi theo thứ tự tùy ý, tất cả các hàm Awake() sẽ đã hoàn thành trước khi Start() đầu tiên được gọi. Điều này có nghĩa là mã trong hàm Start() có thể sử dụng các khởi tạo khác đã được thực hiện trước đó trong giai đoạn Awake().

Sự kiện GUI:
Unity có một hệ thống để vẽ các điều khiển GUI lên màn hình chính trong cảnh và phản ứng với sự nhấp chuột vào các điều khiển này. Mã này được xử lý một cách khác biệt so với cập nhật khung hình thông thường và vì vậy nên đặt nó trong hàm OnGUI(), sẽ được gọi định kỳ.

```
void OnGUI()
{
   GUI.Label(labelRect, "Game Over");
}
```

Bạn cũng có thể phát hiện các sự kiện chuột xảy ra trên một GameObject khi nó xuất hiện trong cảnh. Điều này có thể được sử dụng để nhắm mục tiêu vũ khí hoặc hiển thị thông tin về nhân vật hiện đang được trỏ chuột. Một tập hợp các hàm sự kiện OnMouseXXX (ví dụ: OnMouseOver, OnMouseDown) có sẵn để cho phép một script phản ứng với hành động của người dùng với chuột. Ví dụ, nếu nút chuột được nhấn trong khi con trỏ đang ở trên một đối tượng cụ thể, thì một hàm OnMouseDown trong script của đối tượng đó sẽ được gọi nếu nó tồn tại.

Sự kiện vật lý:
Bộ máy vật lý sẽ báo cáo các va chạm với một đối tượng bằng cách gọi các hàm sự kiện trên script của đối tượng đó. Các hàm OnCollisionEnter, OnCollisionStay và OnCollisionExit sẽ được gọi khi có tiếp xúc, tiếp tục tiếp xúc và kết thúc tiếp xúc. Tương ứng, các hàm OnTriggerEnter, OnTriggerStay và OnTriggerExit sẽ được gọi khi collider của đối tượng được cấu hình như một Trigger (tức là một collider chỉ phát hiện khi có gì đó đi vào mà không tác động vật lý). Các hàm này có thể được gọi nhiều lần liên tiếp nếu có nhiều va chạm được phát hiện trong quá trình cập nhật vật lý và vì vậy một tham số được truyền vào hàm để cung cấp thông tin về va chạm (vị trí, danh tính của đối tượng đến, v.v.).

```
void OnCollisionEnter(Collision otherObj)
{
   if (otherObj.tag == "Arrow")
   {
      ApplyDamage(10);
   }
}
```

Hi vọng rằng thông tin trên đã giúp bạn hiểu về các hàm sự kiện trong Unity.