## Tạo project

* Vào ***Open 🡪 New project...***
* Lựa chọn game 3D hoặc 2D
* Đặt tên project (viết liền để thao tác trên github) và chọn thư mục chứa

Lưu ý: Có thể cài đặt nền tảng build game để xác định kích thước màn hình hiển thị của game từ trước: Open 🡪 Building Setting

## Cài đặt scence

* Tạo scence mới
  + Cách 1: ***ctrl + N***
  + Cách 2: ***File 🡪 New scence***
  + Cách 3: Trên thẻ ***Project***  chọn  ***Create 🡪 Scence***
* Các đối tượng trong scene được liệt kê trong thẻ ***Hierarchy***
* Tạo các đối tượng trong scene:
  + Cách 1: Vào ***GameOject***
  + Cách 2: Ở thẻ ***Hierarchy*** chọn ***Create***
  + Cách 3: Kéo thả các object được tạo sẵn từ các thư mục (được download hoặc thiết kế trước) ở thẻ ***Project***  vào thẻ ***Hyerarchy***
* Thông tin chi tiết của các đối tượng bao gồm tên, tên thẻ (tag) và tên lớp (Layer) các component (Chi tiết xem ở phần Component) và script (có thể có hoặc không) của chúng.

Lưu ý: Đối với mỗi đối tượng cần cài đặt, từ giờ chúng ta sẽ xét trên 2 phương diện là các components và scipt. Nên cài đặt Visual Studio vì nó hỗ trợ cài đặt sript rất tốt.

## Các component và tác dụng

Các components của một đối tượng được hiển thị ở cửa sổ **Inspector**. Để tạo môt component:

* Cách 1: Trên thanh ***Menu***  chọn ***Component***
* Cách 2: Trên cửa sổ ***Inspector*** chọn ***Add Component***
* Hàm gọi 1 Component:

**GetComponent<[Tên\_Component]>();**

* Hàm **Compare.Tag**, thường được sử dụng trong việc tương tác giữa đối tượng này với đối tượng khác:

using UnityEngine;

using System.Collections;  
  
public class ExampleClass : [MonoBehaviour](https://docs.unity3d.com/ScriptReference/MonoBehaviour.html) {

void OnTriggerEnter([Collider](https://docs.unity3d.com/ScriptReference/Collider.html) other) {

if (other.CompareTag("Player"))

Destroy(other.gameObject);

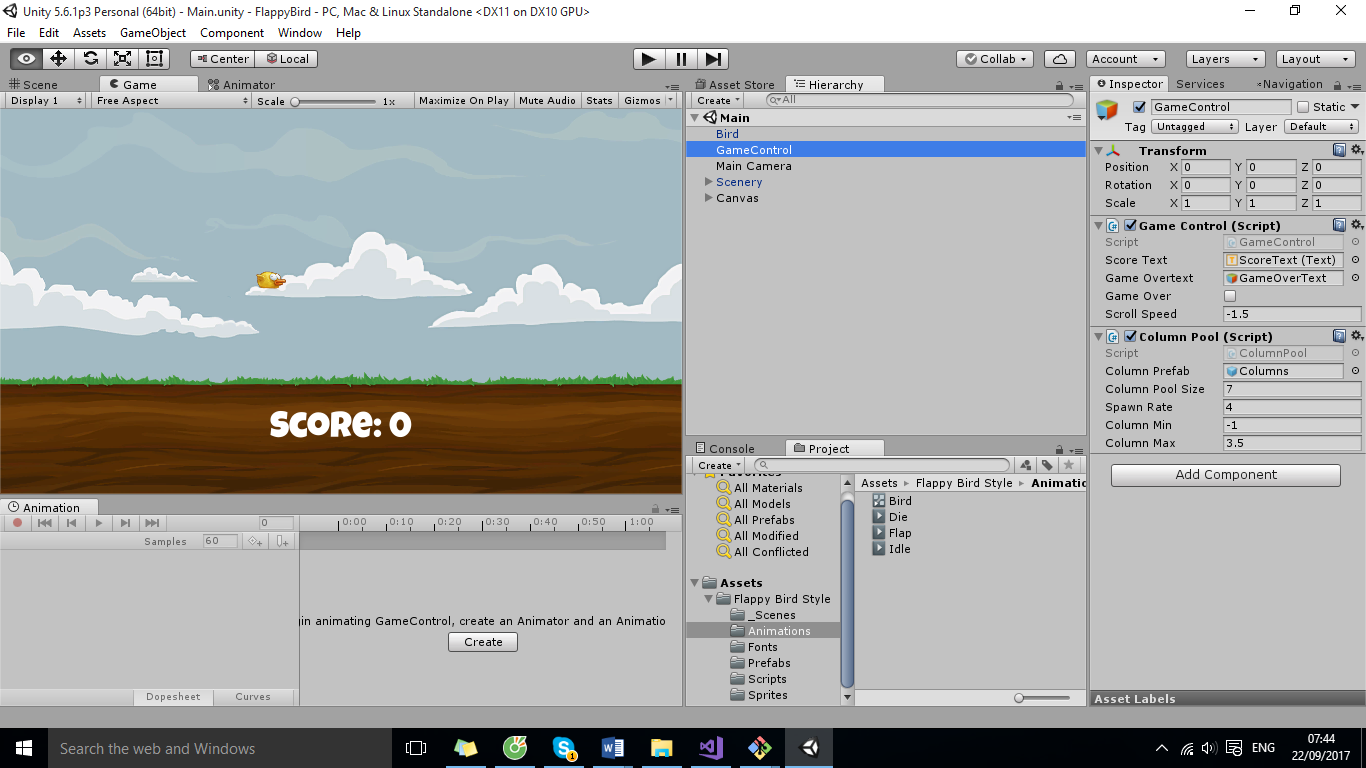
}

}

* Hàm trả về giá trị lớp mặt nạ layer của người dùng
* [LayerMask.GetMask](https://docs.unity3d.com/ScriptReference/LayerMask.GetMask.html) ("UserLayerA", "UserLayerB")

Dưới đây là một số component thường xuyên sử dụng

1. Transform

 Cài đặt thông tin về tọa độ (position), góc quay quay trục (rotation), kích thước co dãn (Scale) trong không gian 3 chiều X, Y, Z. Đại diện của porsition là Vector3(x,y,z), của rotation là Quanternion(x,y,z). Một số biến tương tác với Transform Component thường xuyên sử dụng.

transform.position = new Vector3(0,0,0) // Gán tọa độ của đối tượng

transform.position.x //Gọi tọa độ x của đối tượng

//Chuyển đổi vị trí của đối tượng từ không gian này sang không gian khác

Transform.TransformPoint([Vector3](https://docs.unity3d.com/ScriptReference/Vector3.html) position);

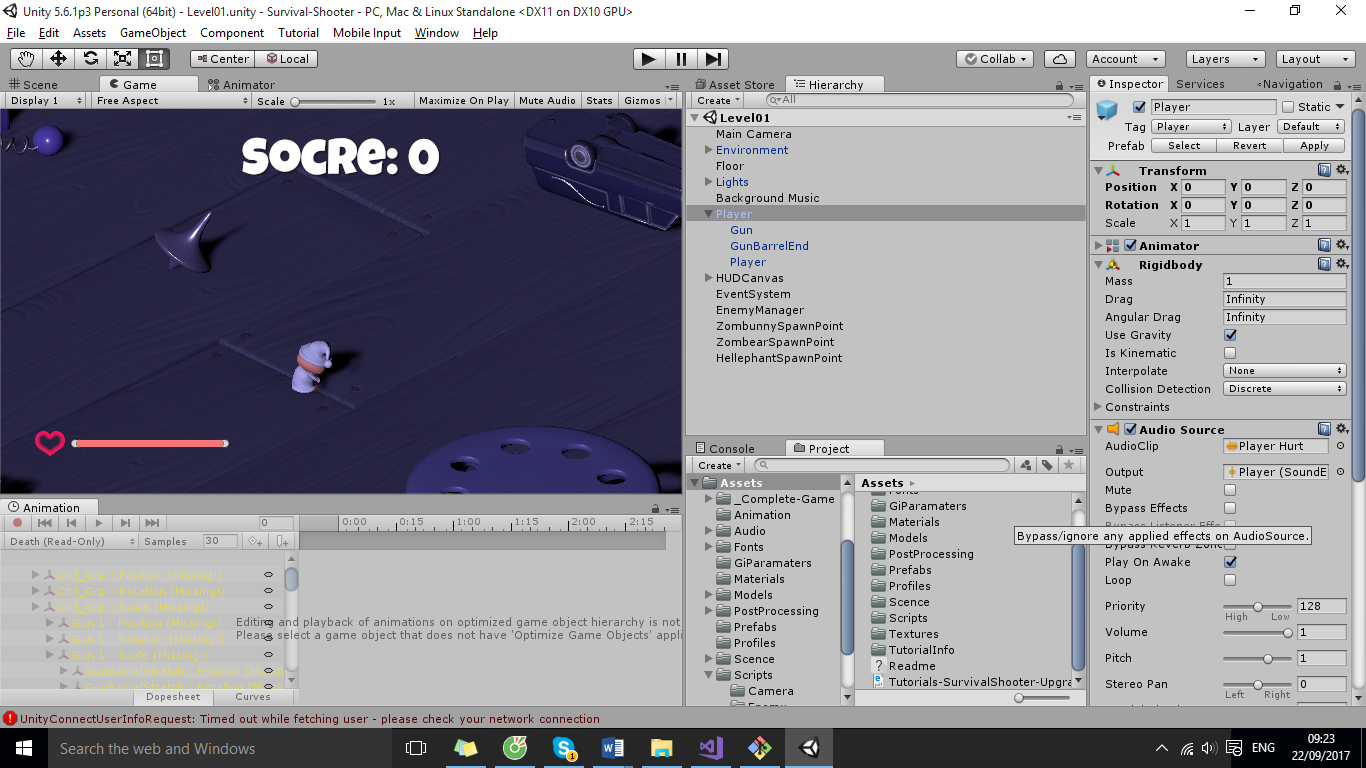
[Transform.TransformDirection](https://docs.unity3d.com/ScriptReference/Transform.TransformDirection.html) //Chuyển đổi vị trị theo hướng của một vector có sẵn

[Transform.InverseTransformPoint](https://docs.unity3d.com/ScriptReference/Transform.InverseTransformPoint.html) //Quay lại vị trí ban đầu

Tham khảo thêm một số hàm tương tác với transform component.

<https://docs.unity3d.com/ScriptReference/Transform.html>

1. Các component đặc tả đặc tính vật lý bên trong đối tượng
2. **Rigidbody**

**Khi thêm component này vào đối tượng, đối tượng sẽ có các đặc tính của một vật rắn với các tương tác như chuyển động, va chạm, chịu ảnh hưởng của trọng lực,...**

**Khai báo và kích hoạt:**

public [Rigidbody](https://docs.unity3d.com/ScriptReference/Rigidbody.html) rb;

rb = GetComponent<[Rigidbody](https://docs.unity3d.com/ScriptReference/Rigidbody.html)>();

**Một số biến thường dùng:**

* **Velocity: vận tốc vật rắn**

public [Rigidbody](https://docs.unity3d.com/ScriptReference/Rigidbody.html) rb;

void Start() {

rb = GetComponent<[Rigidbody](https://docs.unity3d.com/ScriptReference/Rigidbody.html)>();

}

void FixedUpdate() {

if ([Input.GetButtonDown](https://docs.unity3d.com/ScriptReference/Input.GetButtonDown.html)("Jump"))

rb.velocity = new [Vector3](https://docs.unity3d.com/ScriptReference/Vector3.html)(0, 10, 0);

}

* **ArgularVelocity: tốc độ quay của vật rắn được tính bằng radian**
* **useGravity: là một biến bool xem vật rắn có chịu tác động của trọng lực không.**
* **Mass: khối lượng vật rắn**
* **Drag: làm chậm lại vật rắn**

**Một số hàm thường dùng:**

* **Cài đặt lực tác dụng vào vật rắn**

public void AddForce([Vector3](https://docs.unity3d.com/ScriptReference/Vector3.html) force, [ForceMode](https://docs.unity3d.com/ScriptReference/ForceMode.html) mode = ForceMode.Force);

* **Di chuyển vật rắn tới 1 vị trí**

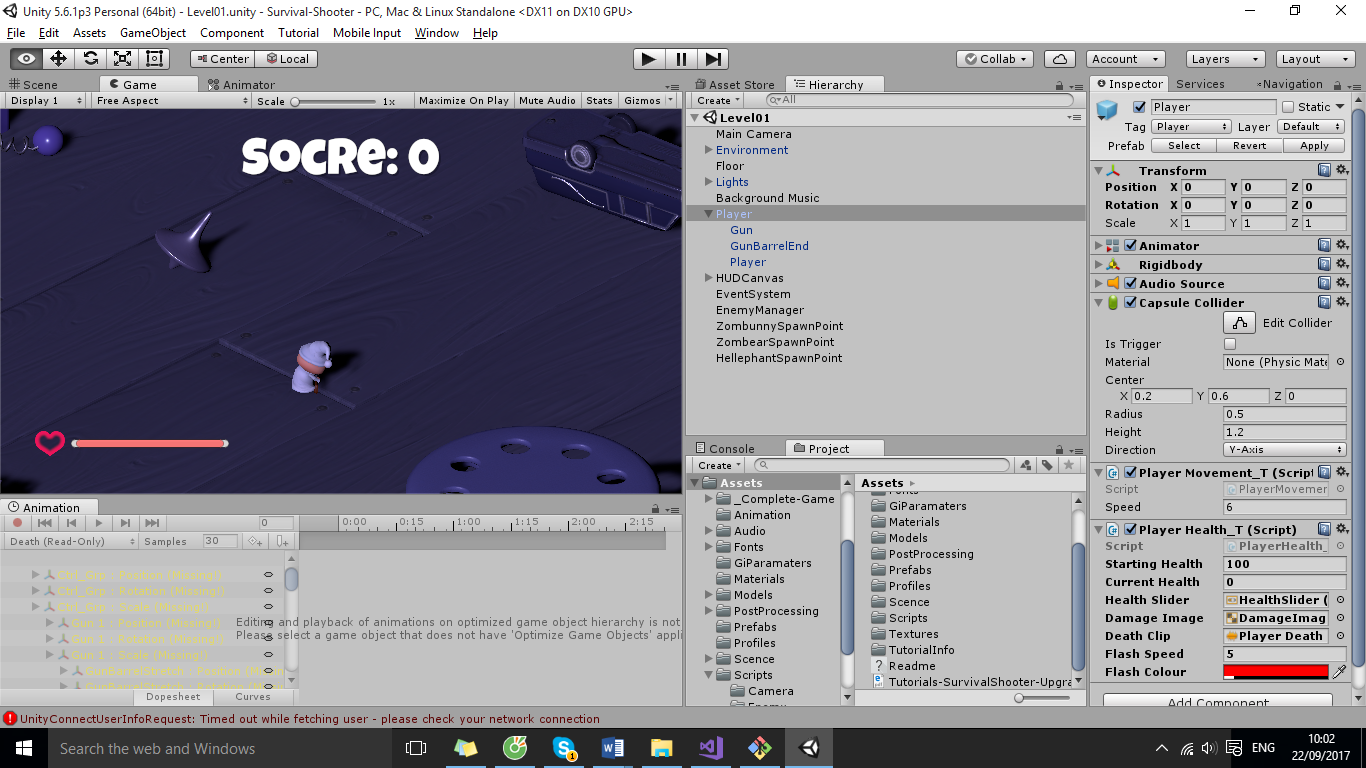
public void **MovePosition**([Vector3](https://docs.unity3d.com/ScriptReference/Vector3.html) **position**);

* **Xoay vật rắn theo góc xoay**

public void **MoveRotation**([Quaternion](https://docs.unity3d.com/ScriptReference/Quaternion.html) **rot**);

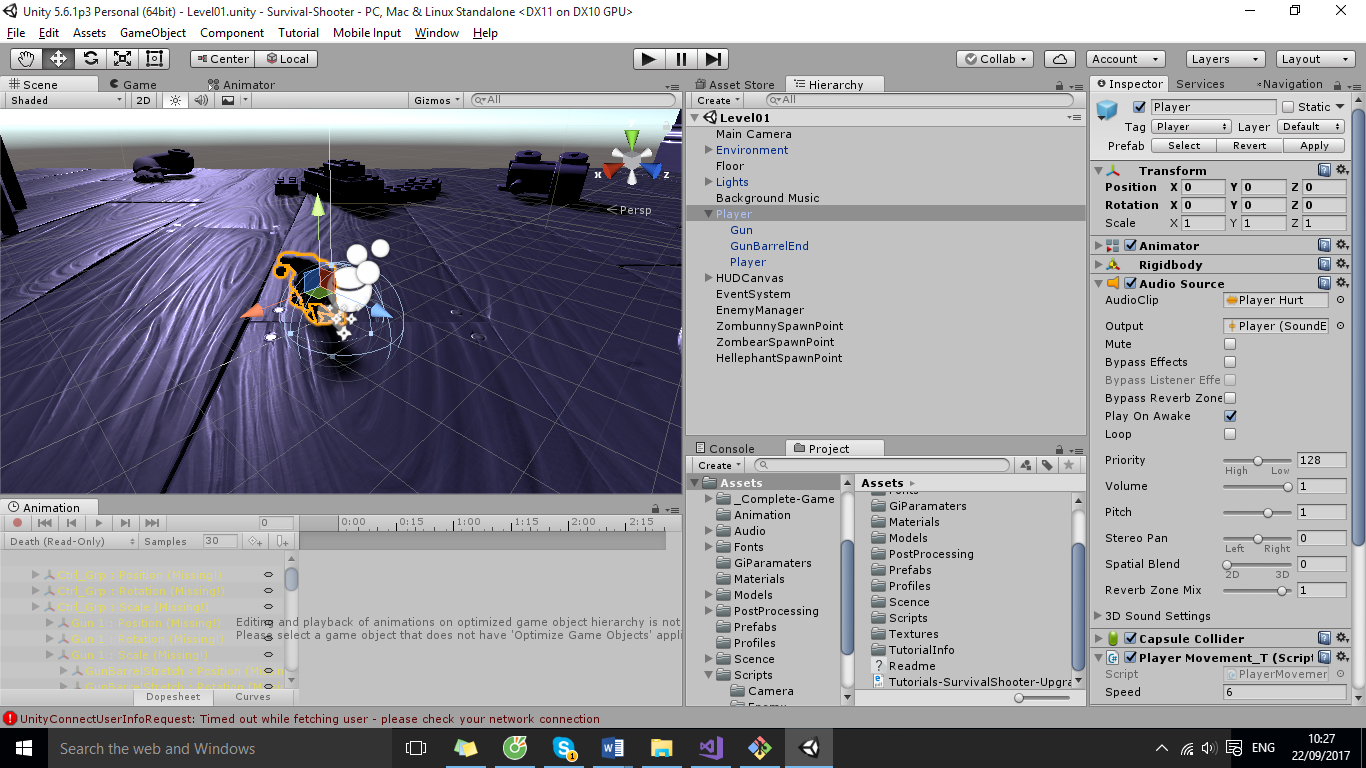
**Xem thêm các hàm, biến:** <https://docs.unity3d.com/ScriptReference/Rigidbody.html>

1. **CapsuleCollider**

Thường được sử dụng trong việc điều khiển nhân vật hoặc các hoạt động của các vật thể có hình dạng bất thường.

Bán kính và chiều cao của vật có thể thay đổi một cách độc lập

1. Audio



Cài đặt âm thanh cho game, âm thanh của các event xuất hiện trong game.

Hàm thường dùng:

* Kích hoạt âm thanh:

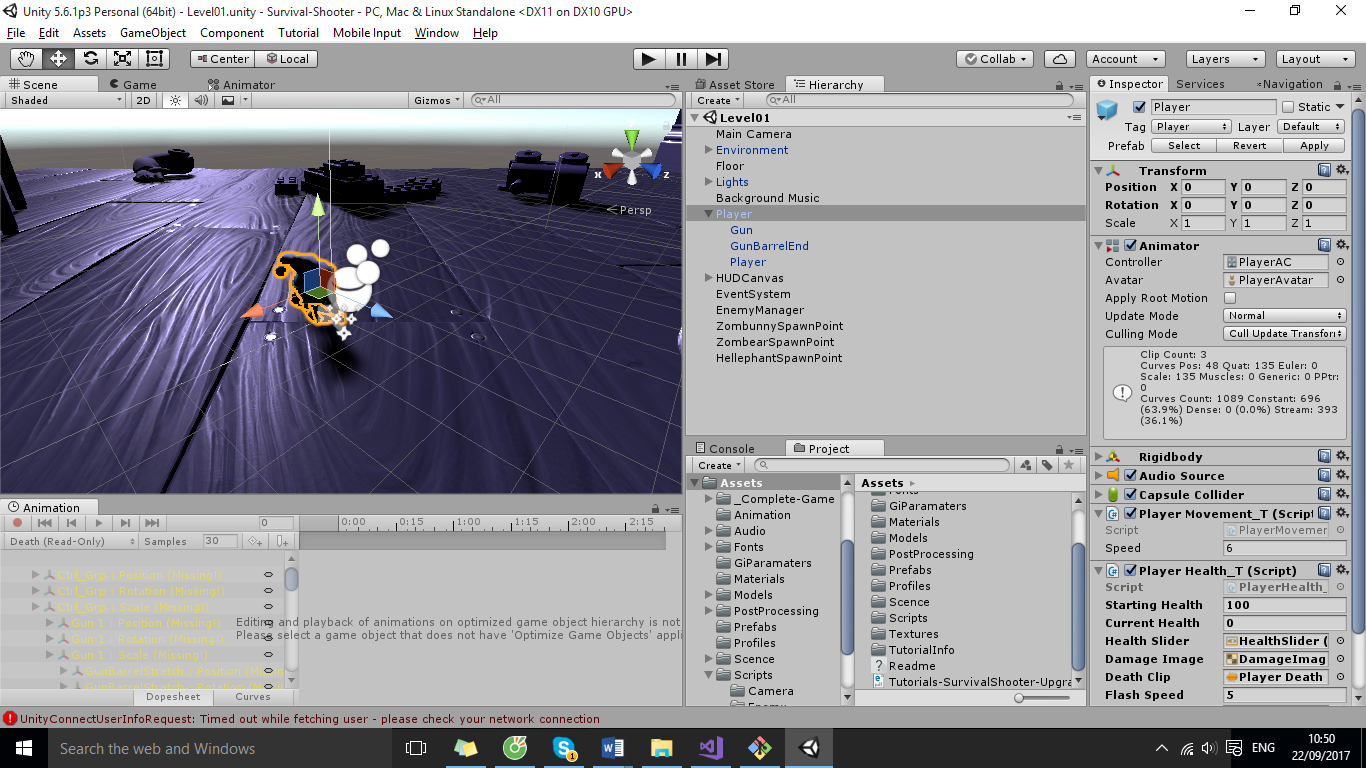
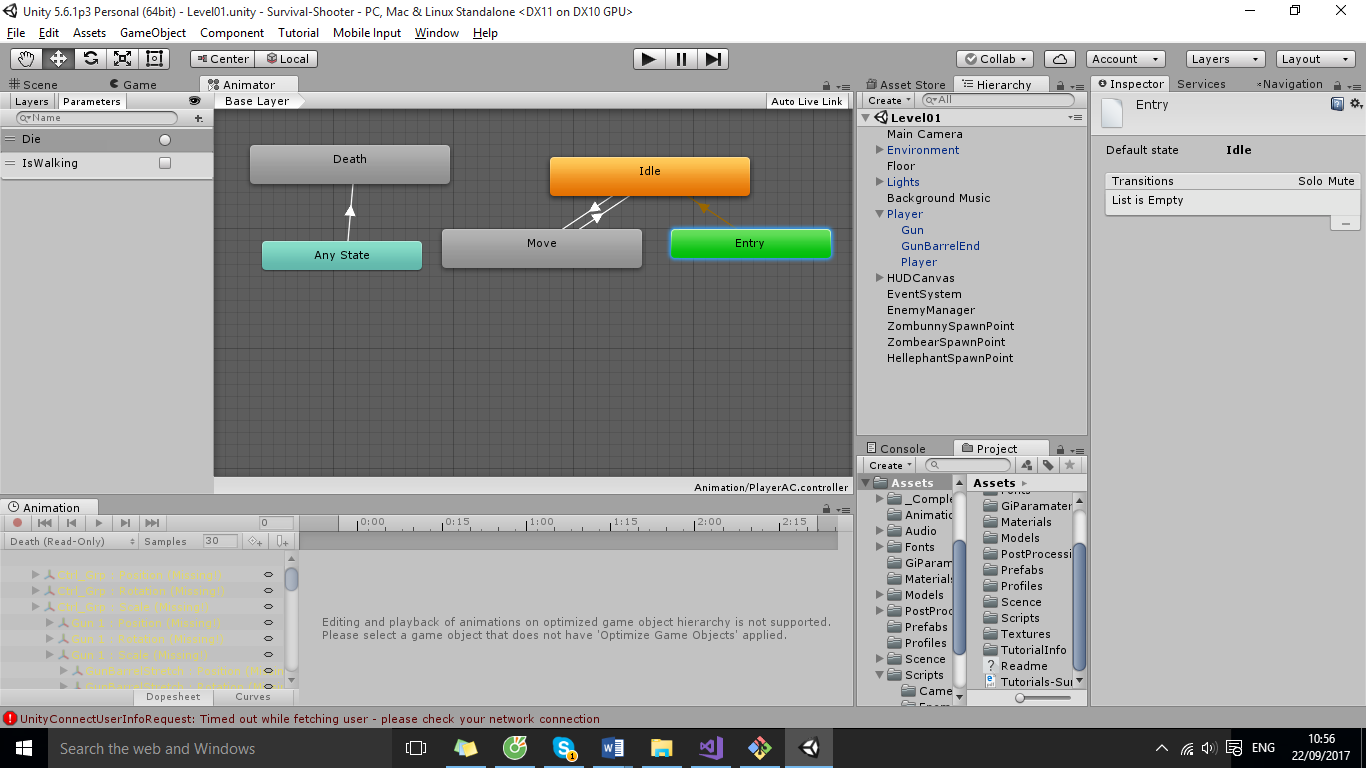
audio.Play();

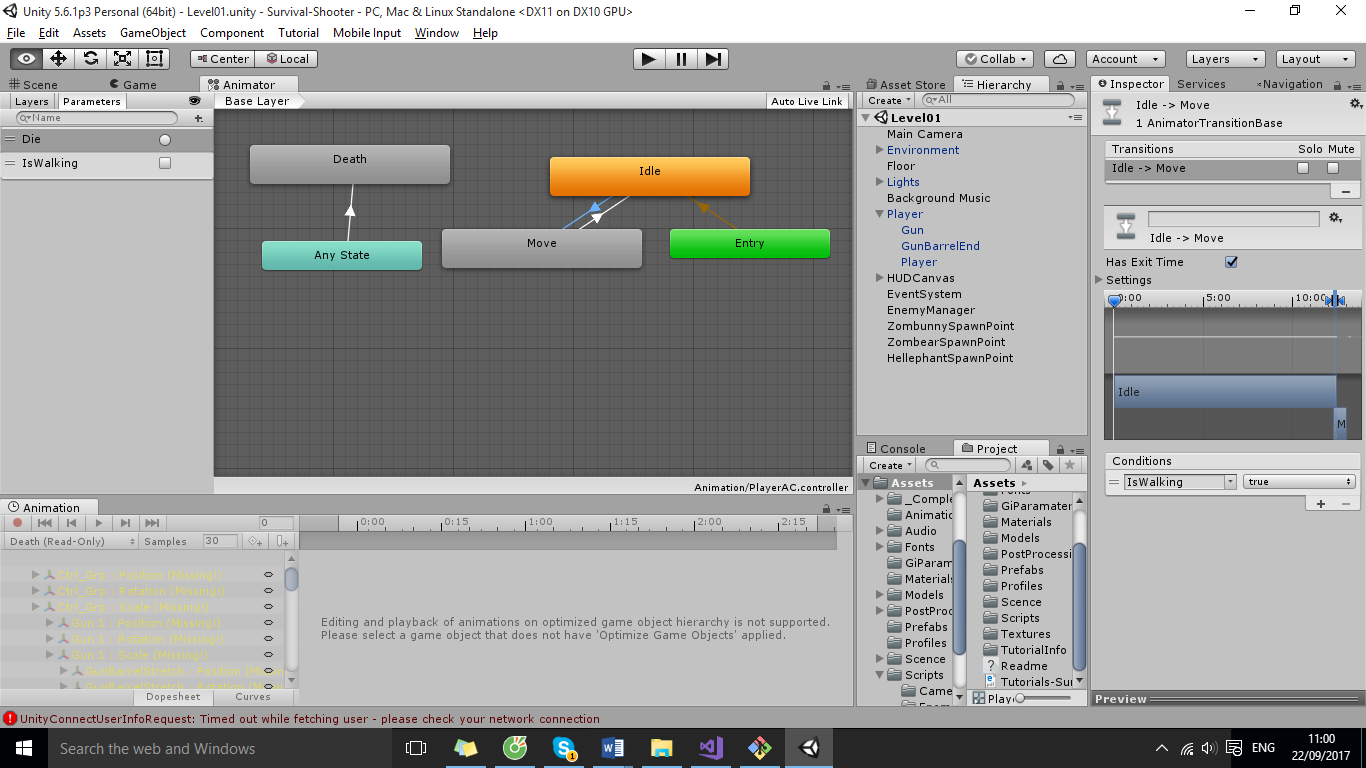
* Âm thanh của đối tượng

var other : [GameObject](https://docs.unity3d.com/ScriptReference/GameObject.html);

other.GetComponent.<[AudioSource](https://docs.unity3d.com/ScriptReference/AudioSource.html)>().Play();

1. Animator

Dùng để thiết lập các trạng thái chuyển động của vật thể. Hình bên là hình ảnh của Animator component.

Để thiết lập thứ tự, thời gian của các trạng thái chuyển động, vào ***Windows 🡪 Animator*** để mở cửa sổ Animator. Tạo các layer (là các trạng thái). Tạo mũi tên biểu thị sự chuyển động giữa trạng thái này với trạng thái kia. Click vào mũi tên để điều chỉnh condition và thời gian chuyển giao.