Trần Thế Thịnh- Report

Table of Contents

[**1.Graphics** 3](#_Toc161159506)

[1.1 Lighting 3](#_Toc161159507)

[1.1.1 Direct Lighting 3](#_Toc161159508)

[1.1.2 Indirect lighting 6](#_Toc161159509)

[1.1.3 Chế độ sáng 7](#_Toc161159510)

[1.1.4 Cookie 8](#_Toc161159511)

[1.1.5 Emissive materials ( vật liệu phát xạ) 9](#_Toc161159512)

[1.1.6 Shadows 9](#_Toc161159513)

[1.2 Cameras 10](#_Toc161159514)

[1.2.1 Tổng quan 10](#_Toc161159515)

[1.2.2 Camera component 11](#_Toc161159516)

[1.3 Materials 12](#_Toc161159517)

[1.3.1 Tổng quan về material 12](#_Toc161159518)

[1.4 Shaders and Textures. 12](#_Toc161159519)

[1.4.1 Tổng quan về shader và cách sử dụng 12](#_Toc161159520)

[1.4.2 Tổng quan về texture và cách sử dụng 14](#_Toc161159521)

[**2. Physics** 16](#_Toc161159522)

[2.1 RigidBody 16](#_Toc161159523)

[2.1.1Tổng quan về Rigidbody 16](#_Toc161159524)

[2.1.2 Các thành phần trong rigidbody: 17](#_Toc161159525)

[2.2 Collider 19](#_Toc161159526)

[2.2.1 Tổng quan về Collider 19](#_Toc161159527)

[2.2.2 Một số điểm quan trọng về Collider 19](#_Toc161159528)

[2.2.3 Collider events và điều kiện đẻ xảy ra 21](#_Toc161159529)

[2.3 Physic materials 21](#_Toc161159530)

[2.3.1 Tổng quan về Physic materials 21](#_Toc161159531)

[2.3.2Các thành phần trong physic materials 22](#_Toc161159532)

[2.4 Physic join 23](#_Toc161159533)

[2.4.1 Character Joint 23](#_Toc161159534)

[2.4.2Configurable Joint 24](#_Toc161159535)

[2.4.3 Fixed Joint 24](#_Toc161159536)

[2.4.4Hinge Joint 25](#_Toc161159537)

[2.4.5 Spring Joint 25](#_Toc161159538)

[**3.Scripting** 26](#_Toc161159539)

[3.1 Tổng quan về script 26](#_Toc161159540)

[3.2 Một số khái niệm về Script 26](#_Toc161159541)

[3.2.1 Biến và Inspector 26](#_Toc161159542)

[3.2.2 Instantiate prefabs trong thời gian chạy 26](#_Toc161159543)

[3.2.3 Thứ tự thực hiện của các event fuctions 27](#_Toc161159544)

[3.2.4 Event Function 28](#_Toc161159545)

[3.2.5 Coroutines 28](#_Toc161159546)

[3.2.6 Null Reference Exceptions 28](#_Toc161159547)

[3.3 Một số class quan trọng 28](#_Toc161159548)

[3.3.1 MonoBehaviour 28](#_Toc161159549)

[3.3.2 GameObject 28](#_Toc161159550)

[3.3.3 Transform 29](#_Toc161159551)

[3.3.4 Vector 29](#_Toc161159552)

[3.3.5 Quaternion 29](#_Toc161159553)

[3.3.6 ScripableObject 29](#_Toc161159554)

[3.3.7 Time 29](#_Toc161159555)

[3.3.8 Mathf 29](#_Toc161159556)

[3.3.9 Random 29](#_Toc161159557)

[3.3.10 Gizmos and Handles 30](#_Toc161159558)

[**4. Audio** 30](#_Toc161159559)

[4.1 Tổng quan về audio 30](#_Toc161159560)

[4.2 AudioSource 30](#_Toc161159561)

[4.3 AudioListener 31](#_Toc161159562)

[4.4 AudioClip 31](#_Toc161159563)

[**5. Animation** 32](#_Toc161159564)

[5.1 Animator 32](#_Toc161159565)

[5.2 Animation 33](#_Toc161159566)

[5.3 AnimationClip (Đoạn hoạt hình) 33](#_Toc161159567)

[5.4 AvatarController 33](#_Toc161159568)

[5.5 AvatarMask 33](#_Toc161159569)

[**6. Navigation** 34](#_Toc161159570)

[6.1 Tổng quan về Navigation 34](#_Toc161159571)

[6.2 NavMesh (viết tắt của Navigation Mesh) 34](#_Toc161159572)

[6.3 NavMesh Agent 34](#_Toc161159573)

[6.4 OffMesh Link 34](#_Toc161159574)

[6.5 NavMesh Obstacle 35](#_Toc161159575)

[**7. Assets** 35](#_Toc161159576)

[7.1 Tổng quan về assets 35](#_Toc161159577)

[7.2 Importing assets 35](#_Toc161159578)

[7.3 Các loại asset thường dùng 35](#_Toc161159579)

# **1.Graphics**

**-**Các tính năng đồ họa trong unity cho phép kiểm soát giao diện ứng dụng của mình và có khả năng tùy biến cao.

## 1.1 Lighting

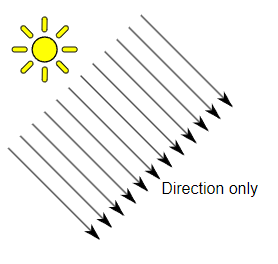
* + - * + Lighting trong Unity hoạt động bằng cách mô phỏng gần đúng cách ánh sáng hoạt động trong thế giới thực. Unity sử dụng các mô hình chi tiết về cách hoạt động của ánh sáng để có kết quả thực tế hơn hoặc các mô hình đơn giản hóa để có kết quả cách điệu hơn.

### 1.1.1 Direct Lighting

* + - * **Định nghĩa**: Chiếu sáng trực tiếp là chiếu sáng trực tiếp từ nguồn sáng tới vật thể mà không phản xạ ra bất kỳ bề mặt nào.
      * **Các loại ánh sáng trực tiếp trong unity**
        + ***Directional Lights***:

+ Đèn định hướng rất hữu ích để tạo các hiệu ứng như ánh sáng mặt trời trong cảnh của bạn.

+Hoạt động giống như mặt trời theo nhiều cách, đèn định hướng có thể được coi là những nguồn sáng ở xa tồn tại ở rất xa.



+Đèn định hướng không có bất kỳ vị trí nguồn có thể nhận dạng nào và do đó đối tượng ánh sáng có thể được đặt ở bất kỳ đâu trong cảnh.

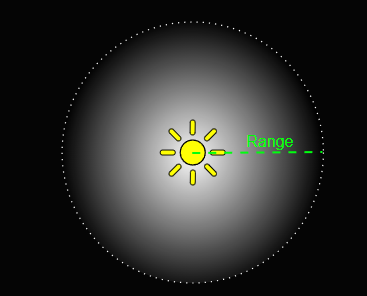
+Mọi vật thể trong khung cảnh đều được chiếu sáng như thể ánh sáng luôn đến từ cùng một hướng.

+Khoảng cách của ánh sáng đến vật thể mục tiêu không được xác định và do đó ánh sáng không giảm đi.

* + - * + ***Point Lights***

+ Đèn điểm được đặt tại một điểm trong không gian và phát ra ánh sáng theo mọi hướng như nhau.

+Hướng của ánh sáng chiếu vào một bề mặt là đường nối từ điểm tiếp xúc trở lại tâm của vật sáng.



+Cường độ giảm dần theo khoảng cách từ ánh sáng, đạt 0 ở một phạm vi xác định.

+Cường độ ánh sáng tỉ lệ nghịch với bình phương khoảng cách tới nguồn.

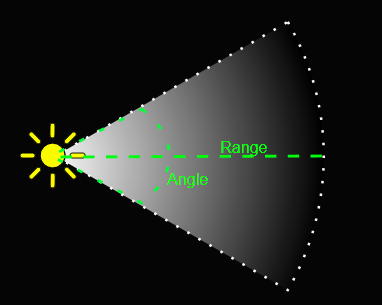
+Điều này được gọi là 'luật bình phương nghịch đảo' và tương tự như cách ánh sáng hoạt động trong thế giới thực.

* + - * + ***Spot Lights***

+ Giống như Đèn điểm, Đèn chiếu sáng có một vị trí và phạm vi xác định mà ánh sáng sẽ tắt

+Spot Light bị hạn chế ở một góc, dẫn đến vùng chiếu sáng hình nón. Tâm của hình nón hướng về phía trước (Z) của vật sáng.

+Ánh sáng cũng giảm dần ở các cạnh hình nón của Spot Light. Việc mở rộng góc sẽ làm tăng chiều rộng của hình nón và cùng với đó là tăng kích thước của phần mờ dần này.

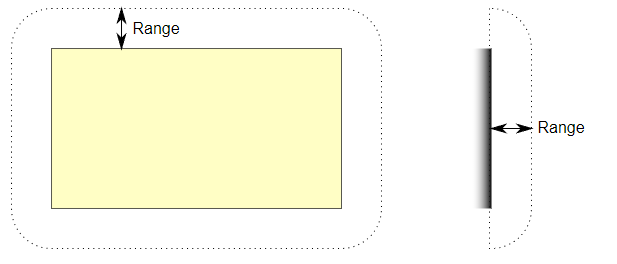


+ Đèn chiếu sáng thường được sử dụng làm nguồn sáng nhân tạo như đèn pin, đèn pha ô tô và đèn pha. Với hướng được điều khiển từ kịch bản hoặc hoạt ảnh, đèn điểm chuyển động sẽ chỉ chiếu sáng một khu vực nhỏ của cảnh và tạo ra hiệu ứng ánh sáng ấn tượng.

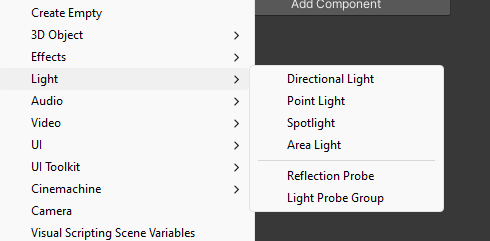
* + - * + ***Area Lights***

+ Đèn khu vực phát ra ánh sáng từ một phía của hình dạng đó.

+Ánh sáng phát ra lan truyền đồng đều theo mọi hướng trên diện tích bề mặt của hình dạng đó.



+Thuộc tính Range xác định kích thước của hình đó. Cường độ chiếu sáng do Đèn khu vực cung cấp giảm dần theo tỷ lệ được xác định bằng nghịch đảo bình phương khoảng cách từ nguồn sáng. Vì tính toán ánh sáng này sử dụng khá nhiều bộ xử lý nên Đèn khu vực không khả dụng trong thời gian chạy và chỉ có thể được đưa vào bản đồ ánh sáng.



### 1.1.2 Indirect lighting

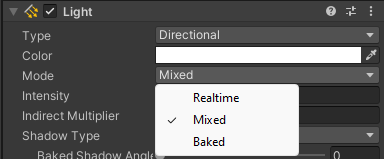
* + - * **Định nghĩa**: Ánh sáng gián tiếp là ánh sáng bật ra khỏi bề mặt và chiếu sáng gián tiếp các vật thể trong khung cảnh. Nó góp phần tạo nên bầu không khí tổng thể và tính chân thực của một cảnh.
      * Trong Unity, để thực hiện ánh sáng gián tiếp, có thể sử dụng các kỹ thuật như Global Illumination (GI) hoặc Enlighten. Global Illumination giúp mô phỏng việc ánh sáng lan tỏa trong môi trường, tạo ra ánh sáng gián tiếp.
      * Bật Global Illumination:

+Window > Rendering > Lighting.

In the Lighting window, Bật Realtime Global Illumination and/or Baked Global Illumination.

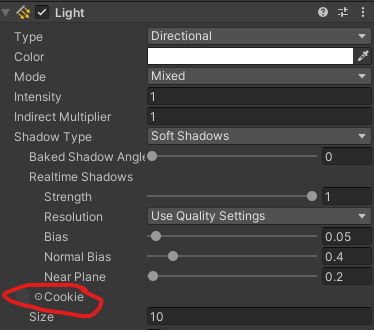
### 1.1.3 Chế độ sáng

Chế độ sáng là phần mode của thành phần light. Thuộc tính mode của đèn xác định mục đích của chúng.

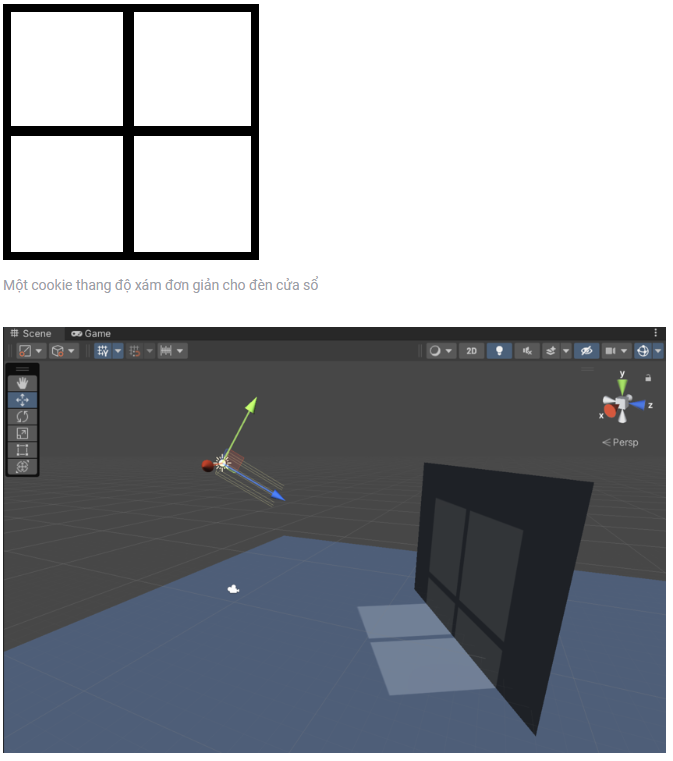


* Realtime
  + Unity tính toán cập nhật ánh sáng của đèn thời gian thực ở mọi khung hình trong thời gian chạy.
  + Thích hợp cho các hiệu ứng ánh sáng động, như đèn chớp, ánh sáng môi trường động, và các tác động ánh sáng khác có thể thay đổi liên tục.
* Mixed
  + Chế độ này là sự kết hợp giữa Realtime và Baked.
  + Đèn trong chế độ này có thể có cả ánh sáng thời gian thực và ánh sáng đã được tính toán trước đó (baked).
  + Thích hợp cho các tình huống khi bạn muốn kết hợp cả ánh sáng động và ánh sáng tĩnh trong môi trường của mình.
* Baked
  + Ánh sáng của đèn đã được tính toán trước và lưu trữ trong bản đồ ánh sáng.
  + Phù hợp cho môi trường cố định mà ánh sáng không thay đổi thường xuyên, giả sử ánh sáng môi trường không thay đổi trong suốt quá trình chạy ứng dụng.
  + Hiệu suất cao hơn so với Realtime, nhưng không thể thay đổi ánh sáng động.

### 1.1.4 Cookie

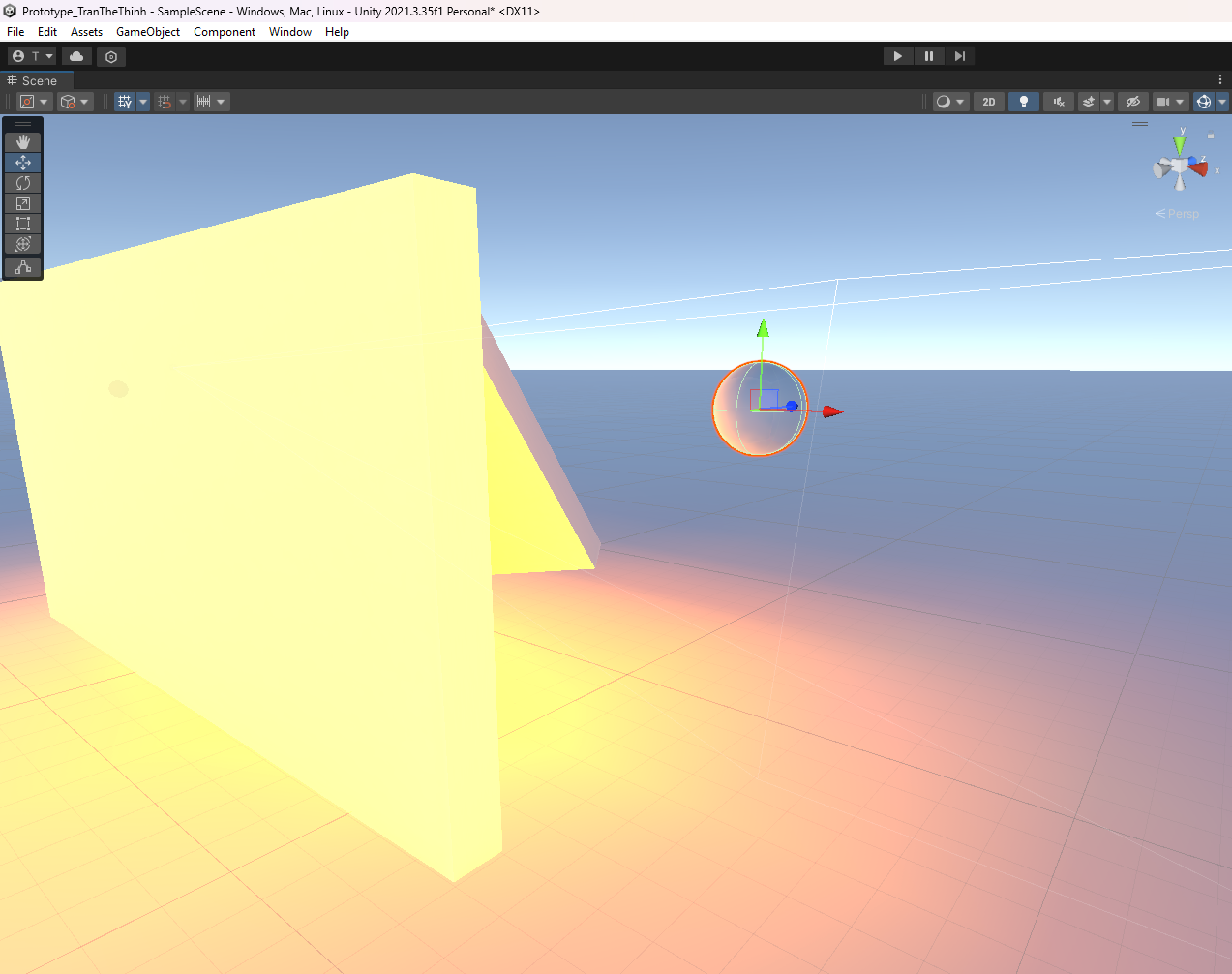


* Cookie là một mặt nạ mà bạn đặt lên Light để tạo bóng có hình cụ thể, làm thay đổi hình thức và cường độ của Ánh sáng.
* Cookie là một cách hiệu quả để mô phỏng các hiệu ứng ánh sáng phức tạp với tác động tối thiểu hoặc không ảnh hưởng đến hiệu suất thời gian chạy.
* Các hiệu ứng có thể mô phỏng bằng cookie bao gồm hiệu ứng ăn da, bóng mờ và hình dạng ánh sáng.



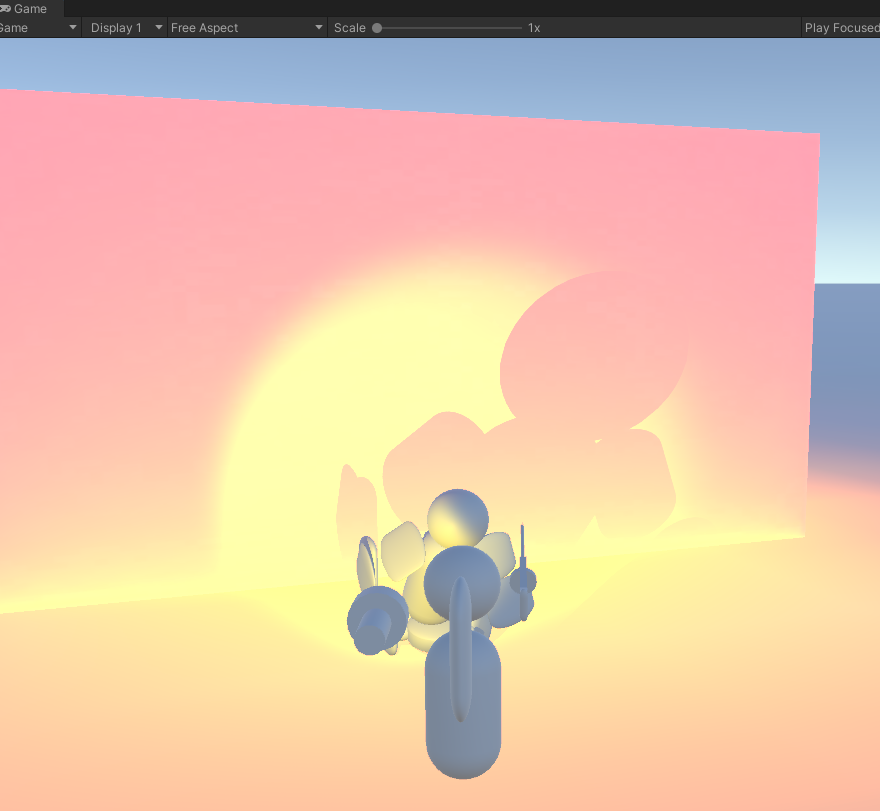
### 1.1.5 Emissive materials ( vật liệu phát xạ)

* Vật liệu phát xạ phát ra ánh sáng trên diện tích bề mặt của chúng. Chúng góp phần tạo ánh sáng phản xạ và các thuộc tính liên quan như màu sắc và cường độ.
* Phát xạ và một thuộc tính của sharder cho phép các static object phát ra ánh sáng.

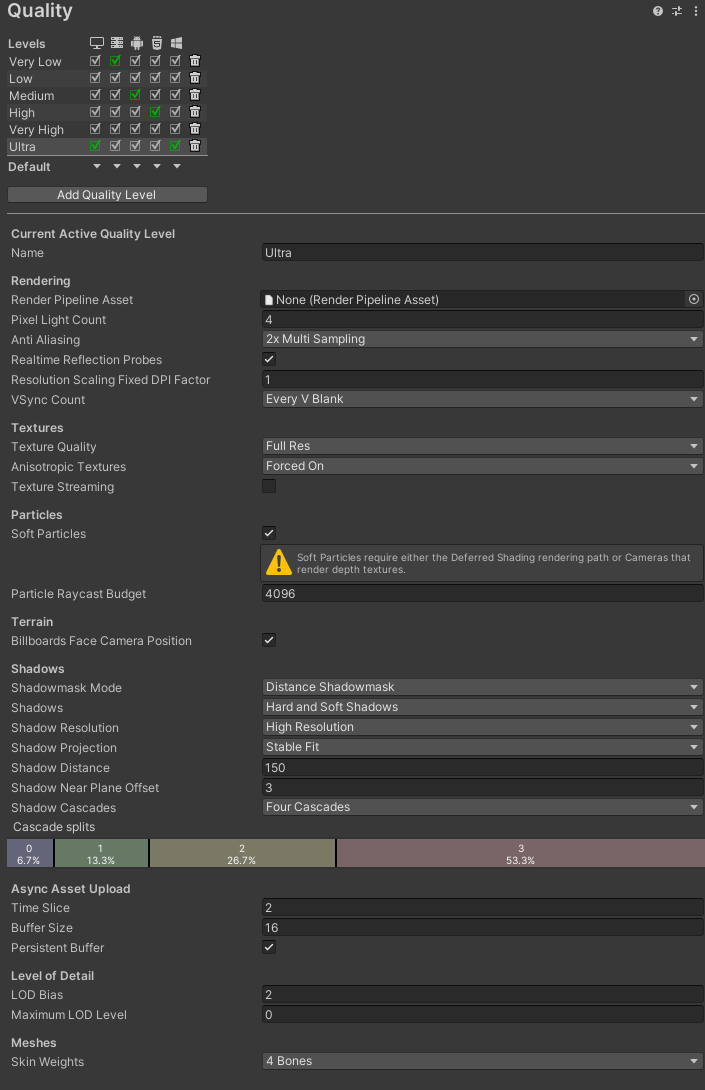


### 1.1.6 Shadows

* Bóng trong unity tạo độ sâu và độ chân thực cho hình ảnh trong scene, chúng làm nổi bật các vật thể trông có vẻ phẳng.



* Trong unity đèn có thể tạo bóng từ một object này nên chính nó hoặc một object khác gần đó.
* Unity sử dụng kết cấu được gọi là bản đồ bóng, tưởng tượng việc đặt một camera ở vị trí của nguồn sáng, các khu vực mà camera ảo này không nhìn thấy được sẽ tương ứng với các vùng bị bóng trong scene.
* Thay đổi độ phân giải của bóng Edit-> ProjectSettings-> quality và thay đổi Shadow resolution.



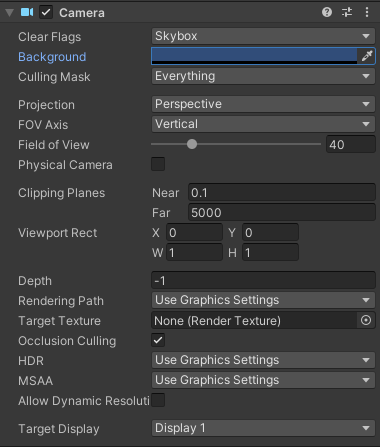
* Thay đổi Shadow distance để thay đổi khoảng cách xuất hiện bóng.

## 1.2 Cameras

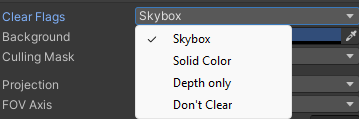
### 1.2.1 Tổng quan

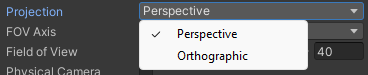
* Trong unity các Gameobject được biểu diễn dưới dạng không gian ba chiều, hình ảnh hiển thì là 2 chiều nên cần đưa nó về 2 chiều thông qua camera.
* Camera trong unity được sử dụng để hiển thị cảnh từ một góc nhìn cụ thể, cho phép người chơi xem và tương tác với thế giới trò chơi.

### 1.2.2 Camera component



* Clear Flags



* Background: Màu được áp dụng cho màn hình còn lại sau khi tất cả các phần tử trong chế độ xem đã được vẽ và không có Shy box.
* Culling mask : Bao gồm hoặc bỏ qua các lớp đối tượng được Máy ảnh hiển thị.
* Projection: Chuyển đổi khả năng mô phỏng phối cảnh của máy ảnh.
* 

+Perspective:Camera sẽ hiển thị các đối tượng với phối cảnh nguyên vẹn.

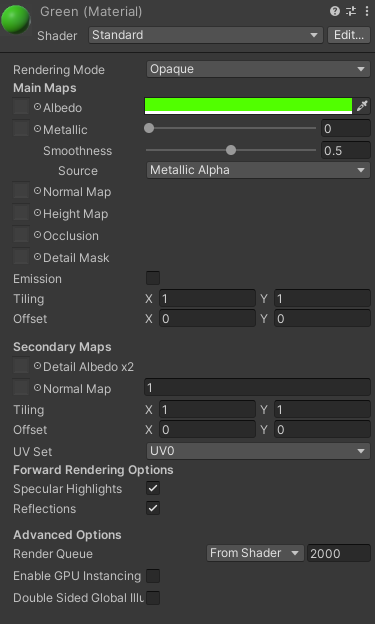
+Orthographic : Camera sẽ hiển thị các đối tượng một cách đồng nhất, không có cảm giác về phối cảnh.

* Pov axis: trục quan sát
* Field of view: Góc nhìn của Máy ảnh, được đo bằng độ dọc theo trục được chỉ định trong Trục FOV .
* Phisical camera: Đánh dấu vào ô này để bật thuộc tính Camera vật lý cho camera này.
* Clipping planes: Khoảng cách từ máy ảnh đến khi bắt đầu và dừng kết xuất.
* Viewport rect: Bốn giá trị cho biết vị trí trên màn hình mà chế độ xem camera này sẽ được vẽ. Được đo bằng Tọa độ khung nhìn (giá trị 0–1).
* Depth: Vị trí của camera theo thứ tự vẽ. Camera có giá trị lớn hơn sẽ được vẽ lên trên camera có giá trị nhỏ hơn.
* Rendering Path: Các tùy chọn để xác định phương pháp hiển thị nào sẽ được máy ảnh sử dụng.
* Target texture: vị trí mà hình ảnh từ camera sẽ hiển thị nếu không có thì sẽ là full màn hình. Thường dùng để tạo bản đồ nhỏ.
* Occlusion culling: bật tính năng này cho phép ẩn các đối tượng đằng sau đối tượng khác sẽ không được hiển thị.
* Hdr
* MSAA: cho phép nhiều mẫu răng cưa cho máy ảnh.
* Allow dynamic resolution
* Target display: xác định thiết bị bên ngoài nào sẽ hiển thị.

## 1.3 Materials

### 1.3.1 Tổng quan về material

* Material là thứ để xác định bề mặt của vật thể xuất hiện như thế nào
* Kiểm soát các thuộc tính của vật thể như màu sắc kết cấu độ bóng đọ trong suốt
* Tạo material mới Assets>create>Material



## 1.4 Shaders and Textures.

### 1.4.1 Tổng quan về shader và cách sử dụng

* Shader là một chương trình chạy trên gpu, điều này có nghĩa là các chương trình đổ bóng là một phần của quy trình đồ họa.
* Object shader: là một cách làm việc dành riêng cho Unity với các chương trình đổ bóng; nó là một trình bao bọc cho các chương trình đổ bóng và các thông tin khác. Nó cho phép bạn xác định nhiều chương trình đổ bóng trong cùng một tệp và cho Unity biết cách sử dụng chúng.
  + Thông tin của một shader:

+Thông tin về chính nó, chẳng hạn như tên của nó

+ Một đối tượng Shader dự phòng tùy chọn mà Unity sẽ sử dụng nếu nó không thể sử dụng đối tượng này

+ Một hoặc nhiều SubShader

**SubShader** cho phép bạn tách đối tượng Shader của mình thành các phần tương thích với phần cứng khác nhau, quy trình kết xuất và cài đặt thời gian chạy.

Một SubShader chứa:

+Thông tin về phần cứng, đường dẫn kết xuất và cài đặt thời gian chạy mà SubShader này tương thích với

+Thẻ SubShader, là cặp khóa-giá trị cung cấp thông tin về SubShader

+Một hoặc passes

Một **Pass** chứa:

Thẻ Pass, là cặp khóa-giá trị cung cấp thông tin về Pass

Hướng dẫn cập nhật trạng thái render trước khi chạy các chương trình đổ bóng của nó

Các chương trình đổ bóng, được tổ chức thành một hoặc nhiều biến thể đổ bóng

* Phân loại Shader trong Unity:

Unity hỗ trợ ba loại shader chính: Vertex/Fragment Shader, Surface Shader và Compute Shader.

+Vertex/Fragment Shader: Cho phép bạn kiểm soát mỗi pixel trên màn hình bằng cách tính toán giá trị màu sắc, độ sáng, vv.

+Surface Shader: Cung cấp một cách dễ dàng để tạo ra các hiệu ứng phức tạp hơn như ánh sáng, bóng đổ và texture mapping.

+Compute Shader: Cho phép bạn thực hiện tính toán song song trên GPU, thường được sử dụng cho các tác vụ không liên quan đến đồ họa như tính toán vật lý hoặc xử lý dữ liệu.

+ShaderLab: Ngôn ngữ của Unity để xác định cấu trúc của các đối tượng Shader.

* Shader graph- một công cụ tạo shader mà không cần viết mã.
* **Cách sử dụng**

+ Tạo shader: create-> shader-> chọn loại shader

+Sửa đổi Shader:Double-click vào shader mới bạn vừa tạo để mở nó trong trình chỉnh sửa.Sử dụng ngôn ngữ lập trình shader (thường là HLSL hoặc Cg trong Unity) để chỉ định cách mà đối tượng được vẽ trên màn hình.

+Áp dụng Shader cho vật thể: Kéo và thả shader từ Project Window lên đối tượng bạn muốn áp dụng shader trong Scene hoặc Inspector Window.

+Tinh chỉnh các thuộc tính của Shader: Nếu shader có các thuộc tính có thể tùy chỉnh (ví dụ: màu sắc, độ trong suốt, ánh sáng, vv.), bạn có thể điều chỉnh chúng trong Inspector Window.

+Hiệu chỉnh Shader (nếu cần): Bạn có thể cần chỉnh sửa shader sau khi đã áp dụng nó cho các đối tượng để đạt được hiệu ứng mong muốn.

### 1.4.2 Tổng quan về texture và cách sử dụng

Texture là hình ảnh hoặc dữ liệu đồ họa được sử dụng để trang trí hoặc làm cho vật thể trong trò chơi của bạn trở nên thực tế hơn. Texture được sử dụng để tạo ra các hiệu ứng hình ảnh như màu sắc, độ bóng, độ nhám, và chi tiết.

Dưới đây là một số khái niệm cơ bản về texture trong Unity và cách sử dụng chúng:

**Loại Texture**: Có nhiều loại texture trong Unity, bao gồm:

Texture 2D: Dùng cho hầu hết các loại vật thể.

Texture 3D: Dùng cho các vật thể 3D.

Cubemap: Dùng cho các texture phản chiếu, như ánh sáng môi trường.

Render Texture: Dùng để lưu trữ hình ảnh được render từ camera hoặc shader.

**Sử dụng Texture trong Shader**: Shader là mã được sử dụng để xử lý cách mà vật thể được vẽ trên màn hình. Bạn có thể sử dụng texture trong shader để thêm màu sắc, chi tiết và hiệu ứng vào vật thể.

**Importing Textures**: Trước khi sử dụng texture trong Unity, bạn cần import chúng vào dự án của mình. Unity hỗ trợ nhiều định dạng file hình ảnh như PNG, JPG, TGA, v.v.

**Texture Properties:** Texture có các thuộc tính quan trọng như Filter Mode (chế độ lọc), Wrap Mode (chế độ bọc), và Compression (nén) để tối ưu hóa hiệu suất và bộ nhớ.

**Texture Atlas và Sprite Sheet**: Để tối ưu hóa hiệu suất, có thể kết hợp nhiều hình ảnh nhỏ vào một texture lớn (texture atlas hoặc sprite sheet) và sử dụng chỉ một texture này cho nhiều đối tượng.

**Sử dụng Texture trong Material**: Material là một thành phần quan trọng trong Unity để xác định cách mà vật thể sẽ phản ánh ánh sáng và màu sắc, có thể gán texture vào các thuộc tính của material như Albedo (màu sắc cơ bản), Normal (hướng bề mặt), Metallic (độ kim loại), Roughness (độ nhám), v.v.

**Sử dụng Texture trong UI**: có thể sử dụng texture để trang trí giao diện người dùng trong Unity, chẳng hạn như làm nền cho các button, panel, hoặc các hình ảnh khác

# **2. Physics**

-Vật lý trong unity chịu trách nhiệm mô phỏng và kiểm soát hành vi vật lý của các object trong trò chơi. Unity sử dụng một công cụ vật lý để xử lý các tác vụ như phát hiện va chạm, động lực học của rigidbody và các tương tác khác của object trong trò chơi.

-Một số thành phần chính trong unity physic:

+Rigitbody

+Collider

+Physic material

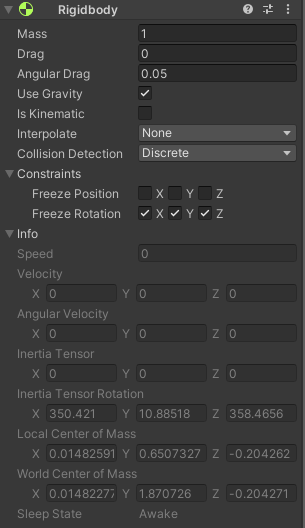
+ Joint

## 2.1 RigidBody

### 2.1.1Tổng quan về Rigidbody

* RigidBody là thành phần chính để kích hoạt các hành vi vật lý cho gameobject.
* Khi có RigidBody thì vật đó ngay lập tức sẽ bị ảnh hưởng bởi trọng lực, nó sẽ xuyên qua các object khác nếu nó không có colider.
* Khi thêm colider thì vật sẽ di chuyển ( thay đổi vận tốc, hướng đi, góc xoay..) do các va chạm.
* Vận tốc hoặc tốc độ xoay nhỏ hơn một mức độ giới hạn thì hệ thống vật lý sẽ coi nó đã dừng lại. Có thể dùng Rigidbody.Sleep(); để ngừng một cách chủ động ( ngắt nó khỏi tính toán vật lý) : nó sẽ bật lại cho đến khi có lực tác dụng vào nó đủ lớn hoặc dùng Rigidbody.Wakeup(); để bật nó trở lại.

### 2.1.2 Các thành phần trong rigidbody:



* + - * **Mass**
* khối lượng của vật thể tính bằng kg. Cùng một lực tác dụng vào thì cách di chuyển của vật sẽ khác nhau khi khối lượng khác nhau.
* Rigidbody.mass=x;
* Khối lượng tương đối của mỗi rigidbody trong một vụ va chạm quyết định cách chúng phản ứng khi va chạm với nhau.
  + - * **Drag**
* mô phỏng lực cản không khí hay lực ma sát nó làm cho vật di chuyển chậm dần theo thời gian. Drag =0 không có lực cản.
* Sử dụng Rigidbody.Drag=x; để thay đổi giá trị hoặc thay đổi trong unity editor.
  + - * **Angular Drag**
* tương tự như Drag nhưng cho chuyển động quay.
* Thay đổi giá trị sử dụng Rigidbody.angularDrag =x; hoặc thay đổi trong editor.
  + - * **Use Gravity**
* dùng để bật tắt ảnh hưởng bởi trọng lực.
* Rigidbody.useGravity = true; bật ảnh hưởng bởi trọng lực, Rigidbody.useGravity =flase là tắt.
  + - * **Is Kinematic**
* khi bật thì đối tượng sẽ không bị ảnh hưởng bởi vật lý chỉ có thể di chuyển bằng Transform.
* Rigidbody.isKinematic = true; để bật. Rigidbody.isKinematic = false; để tắt.
  + - * **Interpolate**
* sử dụng khi tốc độ cập nhật mô phỏng vật lý chậm hơn so với tốc độ khung hình của ứng dụng, làm mượt chuyển động của Rigidbody nếu nó có vẻ giật trong thời gian chạy.
* Có 3 tùy chọn:

- **None**: không áp dụng phép nội suy hoặc ngoại suy.

- **Interpolate**: nguyên lý hoạt động là sử dụng tư thế vận tốc của hai bản cập nhật vật lý trước đó để tính toán và áp dụng tư thế trong khung hình hiện tại. Interpolate chính xác hơn Extrapolate nhưng có đọ trễ cao hơn. Thường dùng trong các trường hợp mà độ chính xác là quan trọng. Ví dụ như vận tốc thay đổi hoặc có các thành phần vật lý khác ảnh hưởng đến di chuyển.

- **Extrapolate**: Sử dụng tư thế và vận từ bản cập nhật vật lý trước đó và dự đoán tư thế trong bản cập nhật vật lý tiếp theo để tính toán và dự đoán tư thế trong khung hiện tại. Thường dùng trong các trường hợp vận tốc không đổi hoặc không bị ảnh hưởng bởi thành phần vật lý khác.

* + - * **Collision Detection**
* Xác định cách hệ thống vật lý phát hiện các va chạm giữa collider này và colliser khác trong hiện trường. Có 4 tùy chọn:

-**Discrete**: Phương pháp này kiểm tra xung đột tại các thời điểm khác nhau, thường là ở những khoảng thời gian cố định.

Ứng dụng:Thường được sử dụng trong những tình huống mà thời gian giữa các khung hình tương đối lớn.

Hạn chế: Có thể bỏ lỡ va chạm nếu đối tượng di chuyển quá nhanh hoặc nếu bước thời gian giữa các khung hình quá lớn.

-**Continuous**: Không giống như phát hiện rời rạc, phát hiện va chạm liên tục nhằm mục đích tìm ra thời điểm va chạm chính xác trong quá trình chuyển động của vật thể.

Ứng dụng: Thích hợp cho các tình huống yêu cầu độ chính xác cao, đặc biệt khi xử lý các vật thể chuyển động nhanh hoặc mô phỏng trong đó độ chính xác là rất quan trọng.

-**Continuous Dynamic**: Được thiết kế đặc biệt cho các vật thể chuyển động liên tục hoặc thay đổi hình dạng động.

Ứng dụng: Hữu ích khi xử lý các vật thể có thể biến dạng, vật thể có khớp nối hoặc các tình huống trong đó vật thể có thể thay đổi hình dạng theo thời gian.

-**Continuous Speculative**: Cách tiếp cận này liên quan đến việc dự đoán các va chạm có thể xảy ra trước khi chúng xảy ra, dựa trên trạng thái hiện tại và quỹ đạo của các vật thể.

Ứng dụng: Có thể có ích trong các tình huống mà việc dự đoán và ngăn ngừa va chạm trước là có lợi, chẳng hạn như trong các ứng dụng thời gian thực.

* + - * **Constraints**
* đặt các hạn chế di chuyển

-**Freeze Position**: Dừng Rigidbody di chuyển trong các trục X, Y và Z in world space.

-**Freeze Rotation**: Dừng Rigidbody quay quanh các trục X, Y và Z cục bộ.

* + - * **Info**
* cho biết các thông tin như:

*-***Speed**: Tốc độ mà Vật rắn đang di chuyển, được đo bằng (m/s).

**-Velocity**: Vận tốc của vật rắn dọc theo mỗi vectơ, được đo bằng (m/s).

**-Angular Velocity**: Vận tốc của vật rắn dọc theo mỗi vectơ, được đo bằng độ trên giây.

*-***Inertia Tensor**: Tenxơ quán tính của vật rắn này. Giống như khốilượng, một tenxơ quán tính xác định lực hoặc mô men xoắn cần thiết để làm cho Vật rắn chuyển động; tuy nhiên, trong khi khối lượng ảnh hưởng đến chuyển động tuyến tính thì tensor quán tính ảnh hưởng đến chuyển động quay. Giá trị Tensor quán tính càng cao thì càng cần nhiều mô-men xoắn để làm cho Vật cứng quay trên trục của nó.

*-***Intertia Tensor Rotation**: Sự quay của tensor quán tính. Theo mặc định, Unity tự động tính toán độ xoay tensor quán tính, dựa trên tất cả các máy va chạm được gắn vào Rigidbody

*-***Local Center of Mass**: Vị trí cục bộ của khối tâm của vật rắn dọc theo mỗi vectơ. Có thể thay đổi tâm khối thông qua tập lệnh (xem Rigidbody.centerOfMass = vector3; hoặc Unity có thể thay đổi nó dựa trên vị trí của collider..

*-***World Center of Mass**: Vị trí toàn cầu của khối tâm của vật rắn dọc theo mỗi vectơ.

**-Sleep State**: Trạng thái ngủ của Rigidbody. Hiển thị trạng thái thức nếu Rigidbody thức và được đưa vào tính toán vật lý. Hiển thị Đang ngủ nếu Rigidbody đang ngủ và không được đưa vào tính toán vật lý.

## 2.2 Collider

### 2.2.1 Tổng quan về Collider

* Collider xác định danh giới vật lý của một Rigidbody.
* Để xảy ra va chạm phải thêm collider vào gameobject cùng với một rigidbody.
* Nếu 2 rigidbody va chạm với nhau, hệ thống vật lý chỉ tính toán va chạm khi cả 2 đều được gán collider. Nếu 1 gameobject có rigidbody nhưng không có collider thì nó chuyển qua gameobject khác và unity không đưa nó vào tính toán va chạm.

### 2.2.2 Một số điểm quan trọng về Collider

- **Xác định va chạm**: Collider được sử dụng để xác định khi nào và như thế nào các vật thể sẽ tương tác với nhau. Khi hai Collider của hai GameObject khác nhau va chạm vào nhau, hệ thống vật lý sẽ tính toán các sự kiện như va chạm, đâm vào.

-**Yêu cầu sử dụng với RigidBody**: Để va chạm giữa các vật thể được tính toán, cả hai GameObject đều cần phải có Collider và ít nhất một trong chúng cần phải có RigidBody. Nếu một GameObject có RigidBody mà không có Collider, nó sẽ được xem xét là không tham gia vào tính toán va chạm.

-**Loại Collider**: Unity cung cấp nhiều loại Collider khác nhau, phù hợp cho các hình dạng và loại tương tác khác nhau. Ví dụ, Box Collider cho phép xác định va chạm cho các hình hộp, Sphere Collider cho hình cầu, và Mesh Collider cho các hình dạng phức tạp hơn dựa trên mesh của đối tượng.

* **Các loại collider**: các loại collider phụ thuộc vào cấu hình collider và rigidbody của gameobject

+**Static collider**: Collider gắn vào GameObject không có thành phần Rigidbody. Loại máy va chạm này được sử dụng cho các vật thể không cần tương tác vật lý

+**Rigidbody collider**: Collider được gắn vào GameObject có thành phần Rigidbody.

-*Dynamic collider:*The Rigidbody is dynamic (Is Kinematic disabled)

*-Kinematic collider*: The Rigidbody is kinematic (Is Kinematic enabled).

* **Trigger colliders:** là một loại thành phần collider cho phép phát hiện khi các vật thể khác đi vào hoặc thoát khỏi một khu vực nhất định mà không va chạm vật lý với chúng. Chúng thường được sử dụng để triển khai các cơ chế chơi trò chơi như kích hoạt khu vực, nhặt vật phẩm hoặc kích hoạt các sự kiện khi người chơi đi vào một khu vực cụ thể.
  + **OnTriggerEnter(Collider other):**
    - **Mô tả:** Được gọi khi một máy va chạm khác đi vào vùng kích hoạt của GameObject.
    - **Cách sử dụng:** Thường được sử dụng để phát hiện khi người chơi đi vào một khu vực cụ thể, kích hoạt các sự kiện như hội thoại, điểm kiểm tra hoặc kích hoạt các yếu tố trong trò chơi.
  + **OnTriggerStay(Collider other):**
    - **Mô tả:** Được gọi ở mọi khung hình miễn là một máy va chạm khác vẫn nằm trong vùng kích hoạt của GameObject.
    - **Cách sử dụng:** Hữu ích cho việc tương tác liên tục, chẳng hạn như gây sát thương theo thời gian, theo dõi vị trí của người chơi khi ở trong trình kích hoạt hoặc cập nhật một số hiệu ứng.
  + **OnTriggerExit(Collider other):**
    - **Mô tả:** Được gọi khi một máy va chạm khác thoát khỏi vùng kích hoạt của GameObject.
    - **Cách sử dụng:** Thường được sử dụng để xử lý các sự kiện khi người chơi rời khỏi một khu vực cụ thể, tắt các phần tử, đặt lại các biến hoặc kích hoạt hoạt ảnh thoát.
* **Hình dạng của các collider**
  + **Hình dang nguyên thủy:** là loại hiệu tính toán quả nhất, là các hình đơn giản như:
    - Box collider
    - Sphere collider
    - Capsule collider

Collider nguyên thủy có hiệu quả nhưng chúng có những hạn chế. Ví dụ: bạn không thể thay đổi hoặc làm biến dạng hình dạng của chúng mà chỉ có tỷ lệ của chúng.

* + **Mesh collider**
    - tạo ra mô hình phù hợp với model 3d
  + **Wheel collider**
    - collider hình bánh xe
  + **Terrain collider**
    - tạo ra collider phù hợp với terrains

### 2.2.3 Collider events và điều kiện đẻ xảy ra

* + **Collision events**:

void OnCollisionEnter(Collision collision) {

// Code to execute when a collision occurs

}

void OnCollisionStay(Collision collision) {

// Code to execute while colliders are in contact

}

void OnCollisionExit(Collision collision) {

// Code to execute when a collision ends

}

* + **Trigger events**:

void OnTriggerEnter(Collider other) {

// Code to execute when a trigger collision occurs

}

void OnTriggerStay(Collider other) {

// Code to execute while colliders are in trigger contact

}

void OnTriggerExit(Collider other) {

// Code to execute when a trigger collision ends

}

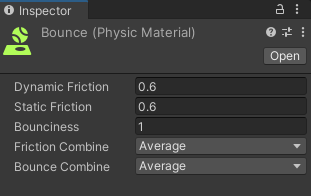
* **Điều kiện xảy ra OnCollision**:
  + Cả 2 đểu phải có collider
  + Ít nhất 1 trong 2 có rigidbody
  + Không cái nào bật is trigger
* **Điều kiện xảy ra OnTrigger**:
  + Cả 2 đểu phải có collider
  + Ít nhất 1 trong 2 có rigidbody
  + Ít nhất 1 trong 2 có is trigger = true;

## 2.3 Physic materials

### 2.3.1 Tổng quan về Physic materials

* Physic materials được sử dụng để điều chỉnh hiệu ứng ma sát và nảy của các object
* Để tạo physic materials **Assets > Create > Physic Material**, sau đó kéo vật liệu vào collider

### 2.3.2Các thành phần trong physic materials



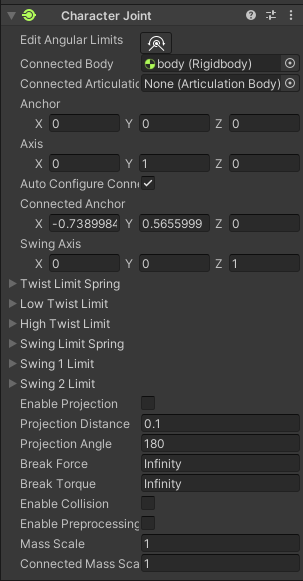
* + - * Dynamic Friction
* Xác định mức độ ma sát của bề collider với collider khác các collider ***chuyển động hoặc trượt vào nhau***. Giá trị này nằm trong khoảng từ 0 đến 1. Giá trị 0 có nghĩa là không có ma sát (như băng), trong khi giá trị 1 có nghĩa là ma sát rất cao (như cao su). Theo mặc định, Ma sát động được đặt thành 0,6.
* Unity sử dụng giá trị ma sát của cả hai máy va chạm tiếp xúc nhau để tính toán ma sát giữa chúng, dựa trên thuộc tính Friction Combine (bên dưới).
  + - * Static Friction
* Xác định mức độ ma sát của bề mặt máy va chạm với máy va chạm khác khi các máy va chạm ***không chuyển động***. Giá trị này nằm trong khoảng từ 0 đến 1. Giá trị 0 có nghĩa là không có ma sát (như băng), trong khi giá trị 1 có nghĩa là ma sát rất cao (như cao su). Theo mặc định, Ma sát tĩnh được đặt thành 0,6.
  + - * Bounciness
* Xác định độ nảy của bề mặt và mức độ các máy va chạm khác có thể bật ra khỏi nó. Giá trị 0 có nghĩa là bề mặt hoàn toàn không nảy (như đất sét mềm) và các máy va chạm khác mất động năng khi chạm vào nó. Giá trị bằng 1 có nghĩa là bề mặt rất nảy (như cao su) và các máy va chạm khác nảy lên mà không bị mất động năng.
  + - * Friction Combine
* Xác định cách hệ thống vật lý tính toán ma sát giữa hai máy va chạm, dựa trên ma sát của mỗi máy va chạm. Lựa chọn này áp dụng cho cả Ma sát động và Ma sát tĩnh
* Các loại kết hợp
  + Maximum: chọn cái có giá trị lớn nhất
  + Multiply: nhân các giá trị với nhau
  + Minimum: chọn cái có giá trị nhỏ nhất
  + Average: tính trung bình cộng các giá trị
    - * Bounce Combine
* Xác định cách hệ thống vật lý tính toán độ nảy giữa hai máy va chạm, dựa trên giá trị Độ nảy của mỗi máy va chạm .
* Các loại kết hợp
  + Maximum: chọn cái có giá trị lớn nhất
  + Multiply: nhân các giá trị với nhau
  + Minimum: chọn cái có giá trị nhỏ nhất
  + Average: tính trung bình cộng các giá trị

## 2.4 Physic join

* Joint nối một rigidbody này với một rigidbody khác hoặc một điểm cố định trong không gian.
* Các joint tác dụng lực để di chuyển các rigidbody và joint hạn chế giới hạn các di chuyển đó.
* Unity's cung cấp các thành phần khớp sau đây để áp dụng các lực và giới hạn khác nhau cho các thành phần của Rigidbody và tạo cho các vật thể đó chuyển động khác nhau.

### 2.4.1 Character Joint

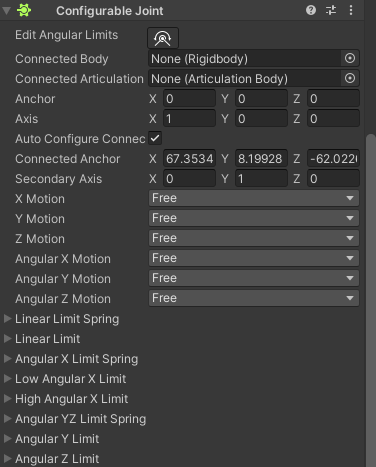
* Thường được sử dụng trong Ragdoll



* ***Connected Body***:Một tham chiếu tùy chọn đến Rigidbody mà khớp này phụ thuộc vào. Nếu không được thiết lập, khớp sẽ kết nối với thế giới.
* ***Anchor***: Điểm trong không gian cục bộ của GameObject nơi khớp quay xung quanh.
* ***Axis***: Xác định trục xoắn. Được hình dung bằng hình nón màu cam.
* ***Auto Configure Connected Anchor***: Nếu được bật, vị trí Connected Anchor sẽ được tính tự động để phù hợp với vị trí toàn cầu của thuộc tính anchor. Nếu tắt, bạn có thể cấu hình vị trí connected anchor thủ công.
* ***Connected Anchor***: hình thủ công vị trí connected anchor.
* ***Swing Axis***: Xác định trục swing. Được hình dung bằng hình nón màu xanh.
* ***Low Twist Limit***: Giới hạn thấp của khớp xoắn.
* ***High Twist Limit***: Giới hạn cao của khớp xoắn.
* ***Swing 1 Limit***: Giới hạn quay xung quanh một phần của trục Swing đã xác định.
* ***Swing 2 Limit***: Giới hạn chuyển động xung quanh một phần của trục Swing đã xác định.
* ***Break Force***: Lực cần được áp dụng để làm hỏng khớp này.
* ***Break Torque***: Mô-men xoắn cần được áp dụng để làm hỏng khớp này.
* ***Enable Collision***: Khi chọn, điều này kích hoạt va chạm giữa các thể có khớp.
* ***Enable Preprocessing***: Tắt xử lý trước giúp ổn định các cấu hình không thể thực hiện được.

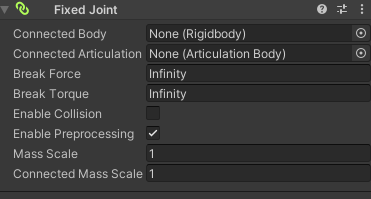
### 2.4.2Configurable Joint

* kết hợp tất cả các chức năng của các loại joint khác và cung cấp khả năng kiểm soát chuyển động của nhân vật tốt hơn. Chúng đặc biệt hữu ích khi bạn muốn tùy chỉnh chuyển động của ragdoll và thực thi một số tư thế nhất định trên nhân vật của mình. Bạn cũng có thể sử dụng chúng để thích ứng khớp vào các khớp chuyên dụng cao theo thiết kế riêng.



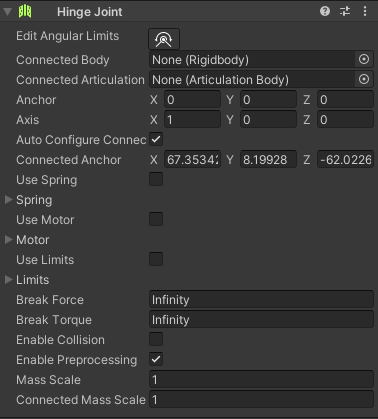
### 2.4.3 Fixed Joint

* Fixed joint làm cho object này phụ thuộc vào chuyển động của object khác gấn giống với parenting nhưng được thực hiện qua vật lý



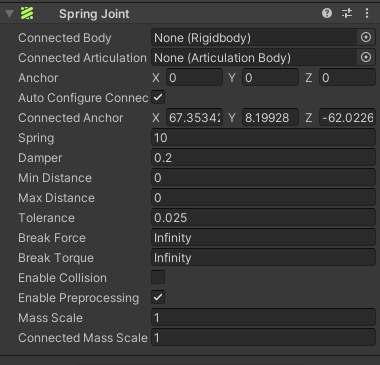
### 2.4.4Hinge Joint

* Hinge joint kết nối 2 rigidbody với nhau làm cho chuyển động giống như bản lề , thường dùng cho cánh cửa, con lắc



### 2.4.5 Spring Joint

* Spring joint kết nối 2 rigidbody với nhau khoảng cách giữa chúng thay đổi như thể chúng được nối với nhau bằng một lò xo



# **3.Scripting**

## 3.1 Tổng quan về script

* Script là các thành phần hành vi có thể được áp dụng cho GameObjects và được sửa đổi trong Unity Inspector.
* Một script được tạo bằng c# được chạy khi game trong trạng thái play.
* Create > C# Script> đổi tên( Khi đổi tên script, điều quan trọng là tên của Script phải khớp với tên của lớp trong mã. Unity sẽ tạo ra một cảnh báo và kịch bản sẽ không biên dịch. > Kéo và thả tập lệnh vào GameObject cần thiết.

## 3.2 Một số khái niệm về Script

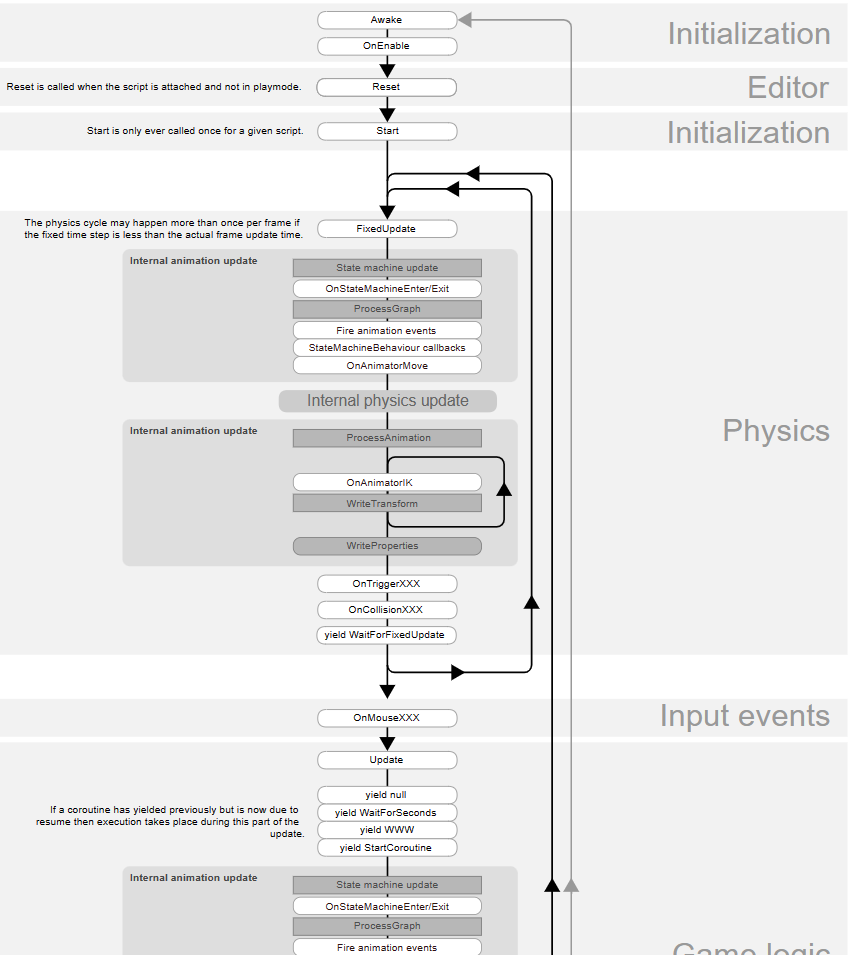
### 3.2.1 Biến và Inspector

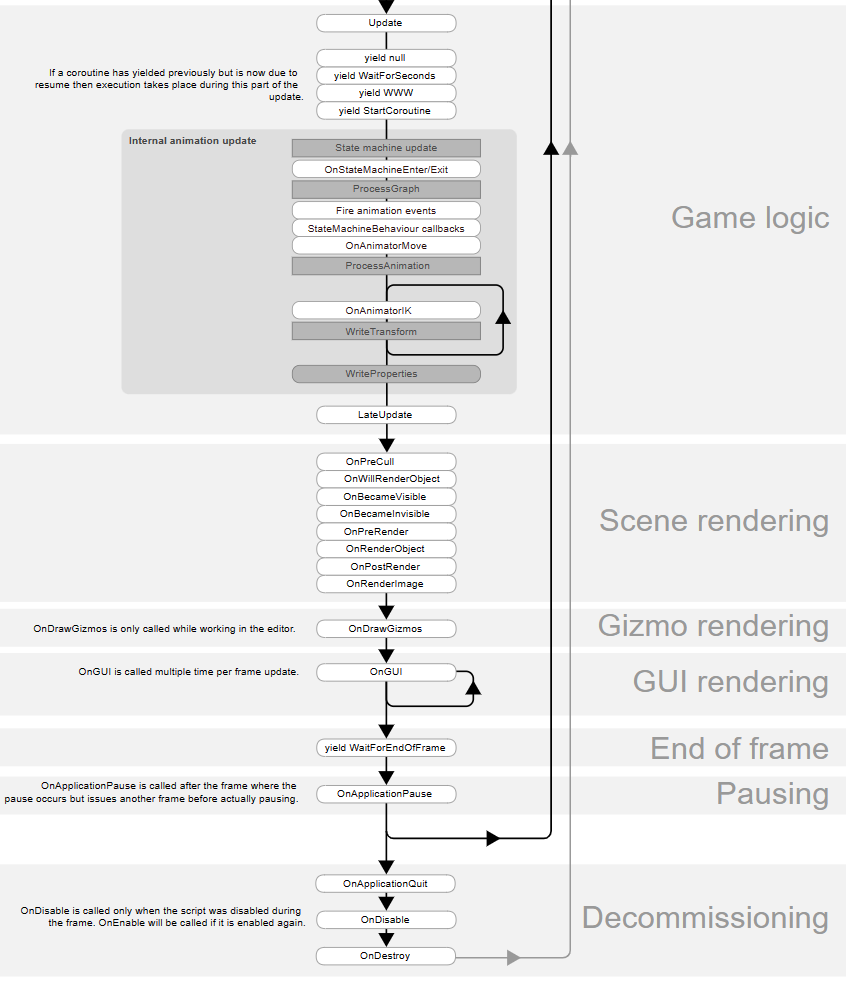
* Khi tạo một script unity sẽ tự đọng cấp một mẫu kế thừa từ MonoBehavior. Khi kế thừa thì nó có thể hoạt động giống như một component mà có thể đính kèm vào gameobject
* Khi có MonoBehavior thì có thể chỉnh sửa các thuộc tính và giá trị trong tập lệnh trong cửa sổ Inspector.
* Với các trường Private có thể sử dụng [SerializeField]. Ngược lại có thể dùng [HideInInsoector] để ẩn các trường public.
* Có thể thay đổi khi game đang chạy để xem thay đổi trực tiếp. khi dừng lại thì giá trị sẽ quay trở lại giá trị ban đầu trước khi chạy.
* Ngược lại có thể thay đổi các thuộc tính của ibject bằng cách tao tham chiếu đến nó.

### 3.2.2 Instantiate prefabs trong thời gian chạy

* Prefabs rất tiện dụng khi tạo các đối tượng phức tạp hoặc tợp hợp các gameobject trong thời gian chạy.
* Để khởi tạo Prefab trong thời gian chạy, script cần tham chiếu đến Prefab đó. Có thể thực hiện tham chiếu này bằng cách tạo một biến công khai trong csript để giữ tham chiếu Prefab. Biến công khai trong script xuất hiện dưới dạng trường có thể gán trong Trình kiểm tra. Sau đó,có thể chỉ định Prefab thực tế muốn sử dụng trong Trình kiểm tra.

### 3.2.3 Thứ tự thực hiện của các event fuctions





### 3.2.4 Event Function

* Unity chuyển quyền kiểm soát sang một tập lệnh không liên tục bằng cách gọi các hàm nhất định được khai báo bên trong nó. Khi một hàm đã thực thi xong, quyền kiểm soát được chuyển trở lại Unity. Các Event Function này được gọi là chức năng sự kiện vì chúng được Unity kích hoạt để đáp ứng với các sự kiện xảy ra trong quá trình chơi trò chơi.
* Event cập nhật thường xuyên:

+ Update

+FixedUpdate

+LateUpdate

* Event khởi tạo:

+Start

+Awake

* Event GUI

+OnGUI

* Physic Event

+OnCollosionEnter

+OnCollisionStay

+OnCollisionExit

+OnTriggerEnter

+OnTriggerStay

+OnTriggerExit

### 3.2.5 Coroutines

* Coroutines cho phép trải rộng các nhiệm vụ trên một số frame.
* Coroutine là một phương thức có thể tạm dừng thực thi và trả lại quyển kiểm soát cho unity nhưng sau đó tiếp tục nơi nó dừng lại trên frame sau.
* Trong hầu hết các trường hợp khi gọi một phương thức nó sẽ chạy cho đến khi nó hoàn thành và trả về một giá trị nhất định-> bất kì hành động nào diễn ra phải diễn ra trong một frame.
* Coroutines không phải là luồng. Các hoạt động đồng bộ chạy trong một coroutine vẫn thực thi trên luồng chính.

### 3.2.6 Null Reference Exceptions

* Xảy ra khi chuy cập vào một object mà chưa có tạo reference đến nó

## 3.3 Một số class quan trọng

### 3.3.1 MonoBehaviour

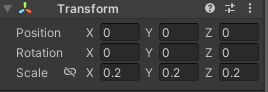
* MonoBehaviour là một base class mà từ đó mọi tập lện của unity bắt nguồn.
* Khi tạo script nó sẽ tự động kế thừa từ MonoBehaviour và cung cấp một tập lệnh mẫu.

### 3.3.2 GameObject

* Class GameObject đại diện cho bất cứ thứ gì có thể tồn tại trong một scene
* GameObjects là các khối xây dựng cho các scene trong Unity và hoạt động như một vùng chứa cho các thành phần chức năng xác định GameObject trông như thế nào và GameObject làm gì.
* Class GameObject cung cấp một tập hợp các phương thức cho phép bạn làm việc với chúng trong mã của mình, bao gồm tìm, tạo kết nối và gửi tin nhắn giữa GameObjects, thêm hoặc xóa các thành phần được đính kèm với GameObject và đặt các giá trị liên quan đến trạng thái của chúng trong scene.

### 3.3.3 Transform

* Transform cung cấp nhiều các khác nhau để làm việc với position , rotation, scale, cũng như mối quan hệ của nó với GameObject cha và con.



### 3.3.4 Vector

* Vector là một khái niệm toán học cơ bản cho phép bạn mô tả một hướng và độ lớn.
* Trong các trò chơi và ứng dụng, vectơ thường được sử dụng để mô tả một số thuộc tính cơ bản như vị trí của nhân vật, tốc độ di chuyển của một vật thể hoặc khoảng cách giữa hai đối tượng.

### 3.3.5 Quaternion

* Quaternion được sử dụng để đại diện cho phép quay
* Unity suer dụng Quaternion class để lưu trữ định hướng chiều xoay của gameobject, có thể dùng để mô tả một còng quay tương đối từ hướng này sang hướng khác.
* Tạo Rotations:

+Quaternion.LookRotation

+Quaternion.Angle

+Quaternion.AngleAxis

+Quaternion.FromToRotation

* Thao tác xoay:

+Quaternion.Slerp

+Quaternion.Inverse

+Quaternion.RotateTowards

+Lớp Transform cũng cung cấp các phương thức cho phép bạn làm việc với các phép quay Quaternion:

Transform.Rotate và Transform.RotateAround

### 3.3.6 ScripableObject

* ScriptableObject là một lớp trong Unity, cho phép tạo ra các đối tượng chứa dữ liệu mà không cần liên kết với bất kỳ game object cụ thể nào. Điều này hữu ích khi bạn muốn lưu trữ dữ liệu có thể chia sẻ giữa nhiều đối tượng và không bị phụ thuộc vào vị trí của chúng trong scene.

### 3.3.7 Time

* Time là một lớp trong Unity cung cấp thông tin về thời gian trong trò chơi.
* Các thuộc tính như **Time.deltaTime** được sử dụng để tính toán khoảng thời gian giữa các frame, giúp đảm bảo tính ổn định và đồng đều trong trò chơi

### 3.3.8 Mathf

* Mathf là một lớp trong Unity cung cấp các hàm toán học cho việc thực hiện các phép toán như làm tròn, căn bậc hai, giữ giá trị trong một khoảng cụ thể, v.v.

### 3.3.9 Random

* Random là một lớp trong Unity được sử dụng để tạo số ngẫu nhiên.
* Hàm như Random.Range được sử dụng để sinh ra số ngẫu nhiên trong một khoảng cụ thể.

### 3.3.10 Gizmos and Handles

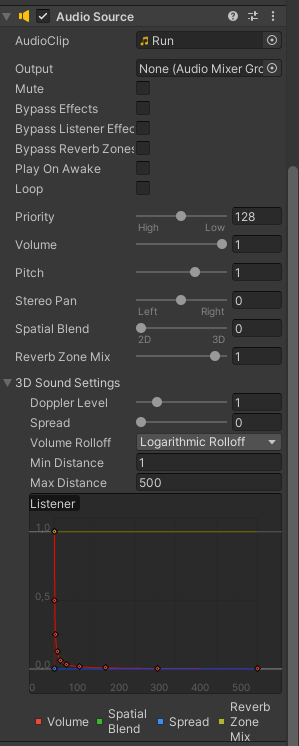
* Gizmos và Handles là các công cụ hỗ trợ trong Unity để hiển thị và tương tác với các đối tượng trong Editor Mode.
* Gizmos thường được sử dụng để vẽ các đối tượng đồ họa như hình tròn, hình vuông, v.v.
* Handles thường được sử dụng để điều chỉnh và thao tác trực tiếp với các thuộc tính của các đối tượng trong Editor Mode.

# **4. Audio**

## 4.1 Tổng quan về audio

* Hệ thống âm thanh của Unity có thể nhập hầu hết các định dạng tệp âm thanh tiêu chuẩn, phát âm thanh trong không gian 3D và áp dụng các hiệu ứng tùy chọn như tiếng vang và lọc. Unity cũng có thể ghi lại âm thanh từ bất kỳ micrô có sẵn nào trên máy của bạn để sử dụng trong khi chơi trò chơi hoặc để lưu trữ và truyền tải.
* Unity có thể nhập các tệp âm thanh ở định dạng AIFF, WAV, MP3 và Ogg giống như các tài sản khác.
* Kéo các tệp vào bảng điều khiển Dự án. Nhập tệp âm thanh để tạo Clip âm thanh sau đó bạn có thể kéo đến Nguồn âm thanh hoặc sử dụng từ tập lệnh. Trang tham chiếu Audio Clip có thêm chi tiết về các tùy chọn nhập có sẵn cho tệp âm thanh.

## 4.2 AudioSource

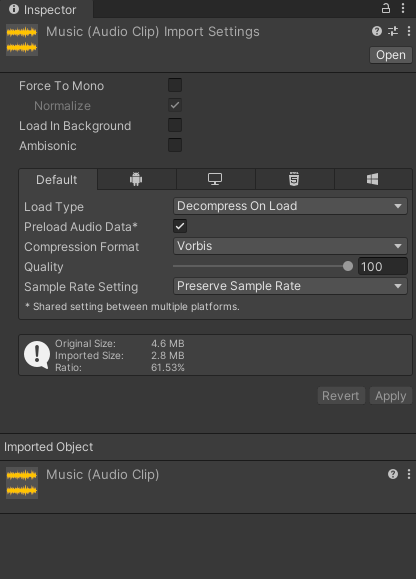
* Audio source phát audioclip in scene, clip có thể phát cho một audio listener hoặc thông qua audio mixer.
* 
  + AudioClip: reference đến âm thanh sẽ được phát ra
  + Output: theo mặc định sẽ được xuất trực thiếp đến audio listener in scene. Sử dụng thuộc tính này đẻ xuất clip sang audio mixer
  + Mute: bật tắt âm thanh
  + Bypass Effects: bật tắt không gian tùy chỉnh cho nguồn âm thanh.
  + Bypass Listener Effects: Điều này là để nhanh chóng bật / tắt tất cả các hiệu ứng Listener.
  + Bypass Reverb Zones: Điều này là để nhanh chóng bật / tắt tất cả các Vùng hồi âm.
  + Play on Awake: Nếu được bật, âm thanh sẽ bắt đầu phát ngay khi cảnh khởi động. Nếu bị tắt, bạn cần khởi động nó bằng lệnh Play() từ scripting.
  + Loop: lặp lại clip âm thanh.
  + Priority: Xác định mức độ ưu tiên của nguồn âm thanh này trong số tất cả các nguồn cùng tồn tại trong cảnh. (Ưu tiên: 0 = quan trọng nhất. 256 = ít quan trọng nhất. Mặc định = 128.). Sử dụng 0 cho các bản nhạc để tránh thỉnh thoảng bị hoán đổi.
  + Volume: chỉnh âm lượng
  + Pitch: chỉnh tốc độ phát của clip
  + Stereo Pan: Đặt vị trí trong trường âm thanh nổi của âm thanh 2D.
  + 3D source setting: Cài đặt được áp dụng tương ứng với tham số Spatial Blend.
  + Doppler Level: Xác định mức độ hiệu ứng doppler sẽ được áp dụng cho nguồn âm thanh này (nếu được đặt thành 0, thì không có hiệu ứng nào được áp dụng).
  + Volume Rolloff: Âm thanh mờ dần nhanh như thế nào. Giá trị càng cao, Người nghe càng phải ở gần trước khi nghe âm thanh. (Điều này được xác định bởi một Biểu đồ).
  + Min Distance: Trong MinDistance, âm thanh sẽ ở mức to nhất có thể. Bên ngoài MinDistance, nó sẽ bắt đầu suy giảm. Tăng MinDistance của âm thanh để làm cho nó 'to hơn' trong thế giới 3d và giảm nó để làm cho nó 'yên tĩnh hơn' trong thế giới 3d.
  + Max Distance: Khoảng cách mà âm thanh ngừng suy giảm tại. Ngoài thời điểm này, nó sẽ ở mức âm lượng, nó sẽ ở các đơn vị MaxDistance từ người nghe và sẽ không suy giảm nữa.

## 4.3 AudioListener

* Audio Listener hoạt động như một thiết bị giống như micrô. Nó nhận đầu vào từ bất kỳ audio source trong scene và phát âm thanh qua loa máy tính.
* 
* Audio Listener không có thuộc tính. Nó chỉ đơn giản là phải được thêm vào để làm việc. Nó luôn được thêm vào Camera chính theo mặc định.

## 4.4 AudioClip

* Clip âm thanh chứa dữ liệu âm thanh được sử dụng bởi Nguồn âm thanh
* Unity hỗ trợ tài sản âm thanh đơn âm, âm thanh nổi và đa kênh lên đến tám kênh. Bạn có thể nhập các định dạng tệp âm thanh sau trong Unity: .aif, .wav, .mp3 và .ogg.



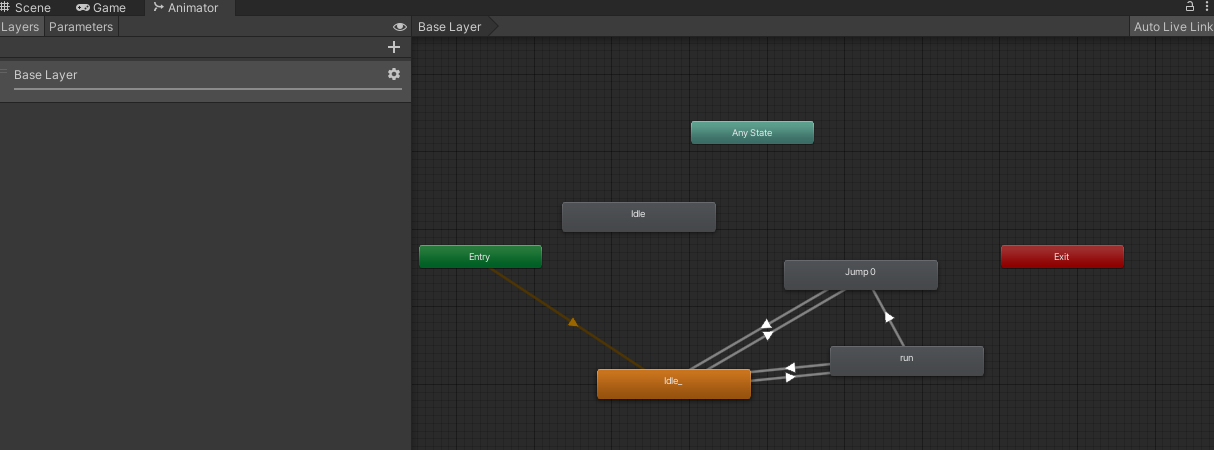
* Force to Mono: Khi tùy chọn này được bật, âm thanh đa kênh sẽ được trộn thành một bản đơn âm trước khi đóng gói.
* Normalize: Khi tùy chọn này được bật, âm thanh sẽ được chuẩn hóa trong quá trình trộn "Force To Mono".
* Ambisonic: Các nguồn âm thanh Ambisonic lưu trữ âm thanh ở định dạng đại diện cho trường âm thanh có thể được xoay dựa trên hướng của người nghe. Nó hữu ích cho video 360 độ và XR Ứng dụng. Bật tùy chọn này nếu tệp âm thanh của bạn chứa âm thanh được mã hóa Ambisonic
* Load in background : Khi tùy chọn này được bật, việc tải clip sẽ diễn ra vào thời điểm chậm trễ trên một luồng riêng biệt mà không chặn luồng chính.

# **5. Animation**

* Animation trong Unity thường được sử dụng để tạo ra sự chuyển động và sự tương tác cho các đối tượng trong game.

## 5.1 Animator

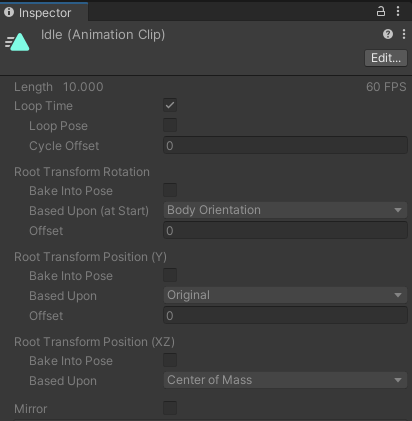
* Animator cho phép sắp xếp và duy trì một bộ animation cho một nhân vật hoặc Gameobject.



* ***Chức năng chính***: Animator trong Unity được sử dụng để tổ chức và duy trì một bộ animation cho một nhân vật hoặc Gameobject.
* ***Thao tác với Animation Clip***: Animator tham chiếu đến các Animation Clip được sử dụng bên trong nó. Các Animation Clip này đại diện cho các đoạn hoạt hình cụ thể.
* ***Quản lý trạng thái***: Animator quản lý các trạng thái hoạt hình khác nhau và quá trình chuyển đổi giữa chúng thông qua việc sử dụng State Machine (máy trạng thái).
* ***Chuyển đổi linh hoạt***: Animator cho phép chuyển đổi giữa các hoạt ảnh và trạng thái khác nhau dựa trên điều kiện xác định trong trò chơi.
* Bộ điều khiển animator cuối cùng được áp dụng cho một đối tượng bằng cách gắn một thành phần Animator tham chiếu đến chúng.

## 5.2 Animation

## 5.3 AnimationClip (Đoạn hoạt hình)

* Mô tả: Animation Clip là đơn vị cơ bản của hoạt ảnh trong Unity, đại diện cho một đoạn hoạt hình cụ thể.
* Chức năng chính của Animation Clip:
  + Làm nền tảng cho sự chuyển động của đối tượng trong game.
  + Được sử dụng bởi Animator để tham chiếu và kích thích sự chuyển động.
  + 

## AvatarController

* Mô tả: Avatar Controller kết hợp với Animator để điều khiển Avatar, đại diện cho nhân vật trong game.
* Chức năng chính của Avatar Controller:
  + Quản lý việc ánh xạ giữa Animation Clip và cử động của Avatar.
  + Được sử dụng để tinh chỉnh và kiểm soát các cử động của nhân vật.

## 5.5 AvatarMask

* Mô tả: Avatar Mask là một công cụ giúp kiểm soát phạm vi của Animation Clip trên một Avatar.
* Chức năng chính của Avatar Mask:
  + Xác định các phần cụ thể của Avatar sẽ tham gia vào một Animation Clip cụ thể.
  + Được sử dụng để tạo ra hiệu ứng tương tác chính xác với một phần của nhân vật.

# **6. Navigation**

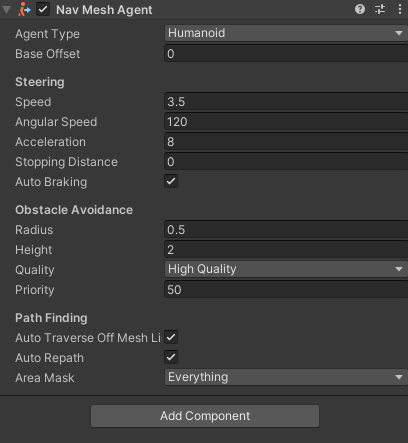
## 6.1 Tổng quan về Navigation

* Navigation cho phép tạo các nhân vật có thể điều hướng thế giới trò chơi. Nó giúp nhân vật của bạn có khả năng hiểu rằng họ cần đi cầu thang bộ để lên tầng hai hoặc nhảy để vượt qua một con mương.

## 6.2 NavMesh (viết tắt của Navigation Mesh)

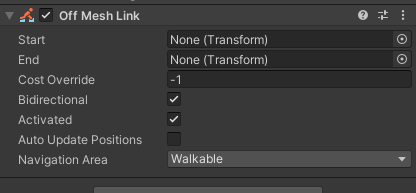
* Navmesh là cấu trúc dữ liệu mô tả các bề mặt có thể đi bộ trong thế giới trò chơi và cho phép tìm đường đi từ vị trí có thể đi bộ này đến vị trí khác trong thế giới trò chơi. Cấu trúc dữ liệu được xây dựng hoặc nướng tự động từ hình học cấp độ của bạn.

## 6.3 NavMesh Agent

* NavMesh Agent giúp bạn tạo các nhân vật tránh mặt nhau trong khi tiến tới mục tiêu của họ. Các đặc vụ suy luận về thế giới trò chơi bằng cách sử dụng NavMesh và họ biết cách tránh nhau cũng như tránh các chướng ngại vật di chuyển. 

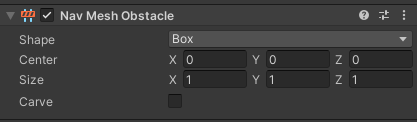
## 6.4 OffMesh Link

* Offmesh Link cho phép bạn kết hợp các phím tắt điều hướng không thể biểu diễn bằng bề mặt có thể đi bộ được. Ví dụ: nhảy qua mương hoặc hàng rào hoặc mở cửa trước khi đi qua nó đều có thể được mô tả là liên kết OffMesh.



## 6.5 NavMesh Obstacle

* NavMesh Obstacle cho phép bạn mô tả các chướng ngại vật di chuyển mà các đặc vụ nên tránh khi điều hướng thế giới. Một cái thùng hoặc một cái thùng được điều khiển bởi hệ thống vật lý là một ví dụ điển hình về chướng ngại vật. Trong khi chướng ngại vật đang di chuyển, các đặc vụ cố gắng hết sức để tránh nó, nhưng một khi chướng ngại vật đứng yên, nó sẽ khoét một lỗ trên lưới điều hướng để các đặc vụ có thể thay đổi đường đi để tránh nó hoặc nếu chướng ngại vật cố định đang chặn đường Bằng cách này, các đặc vụ có thể tìm ra một con đường khác.



# **7. Assets**

## 7.1 Tổng quan về assets

* Asset là bất kỳ mục nào bạn sử dụng trong dự án Unity để tạo trò chơi hoặc ứng dụng của mình. Nội dung có thể đại diện cho các yếu tố hình ảnh hoặc âm thanh trong dự án của bạn, chẳng hạn như mô hình 3D, kết cấu, Sprites, hiệu ứng âm thanh hoặc nhạc.
* Asset cũng có thể đại diện cho các mục trừu tượng hơn như gradient màu, mặt nạ hoạt hình hoặc dữ liệu số hoặc văn bản tùy ý cho bất kỳ mục đích sử dụng nào.
* Asset có thể đến từ tệp được tạo bên ngoài Unity, chẳng hạn như Mô hình 3D, tệp âm thanh hoặc hình ảnh.

## 7.2 Importing assets

* Bạn có thể đưa các tài sản được tạo bên ngoài Unity vào dự án Unity của mình. Để thực hiện việc này, bạn có thể xuất tệp trực tiếp vào thư mục cho dự án của mình hoặc sao chép tệp vào thư mục đó. Đối với nhiều định dạng phổ biến, bạn có thể lưu tệp nguồn của mình trực tiếp vào thư mục của dự án và Unity có thể đọc nó. Unity cũng phát hiện khi bạn lưu các thay đổi mới vào tệp và nhập lại tệp nếu cần.
* Vào projects in unity để xem nội dung thư mục.

## 7.3 Các loại asset thường dùng

* ***Tệp mô hình 3D*** Unity hỗ trợ định dạng tệp FBX, có nghĩa là bạn có thể nhập dữ liệu từ bất kỳ phần mềm mô hình 3D nào hỗ trợ FBX. Unity cũng nguyên bản hỗ trợ nhập các tệp SketchUp. Để biết danh sách phần mềm mô hình 3D mà Unity hỗ trợ, hãy xem Định dạng tệp mô hình.Các tệp Mô hình 3D có thể chứa nhiều loại nội dung, chẳng hạn như lưới, hoạt ảnh, Vật liệu và kết cấu.
* ***Tệp hình ảnh*** Unity nhập các tệp hình ảnh dưới dạng kết cấu. Unity hỗ trợ hầu hết các loại tệp hình ảnh phổ biến, chẳng hạn như BMP, TIF, TGA, JPG và PSD. Nếu bạn lưu các tệp Photoshop (.psd) phân lớp trong thư mục của mình, Unity sẽ nhập chúng dưới dạng hình ảnh phẳng
* ***Tệp âm thanh*** Unity hỗ trợ nhiều định dạng tệp âm thanh. Nói chung, tốt nhất là nhập các định dạng tệp âm thanh không nén.
* ***Văn bản, HTML, XML, JSON*** Unity có thể nhập dữ liệu tùy ý từ các tệp, cho phép lưu trữ và sử dụng dữ liệu từ các nguồn bên
* ***Assets unity*** Có một loạt các loại nội dung có nguồn gốc từ Unity Editor. Bạn có thể tạo nội dung thuộc các loại này bằng các tính năng của Trình chỉnh sửa. Khi bạn tạo các tệp này, Unity sẽ lưu các tệp đại diện cho chúng dưới dạng tệp asset trong thư mục assets của dự án.