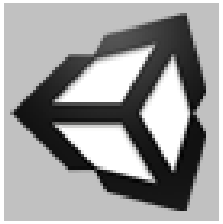
LẬP TRÌNH GAME VỚI UNITY

Bài 6: Physics Game

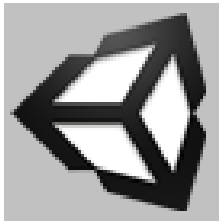
ThS. Thái Duy Quý

Đà Lạt, Tháng 03 năm 2016



Nội dung

- Material
- Character Controller
- Rigid body
- Joint
- Audio
- Animation

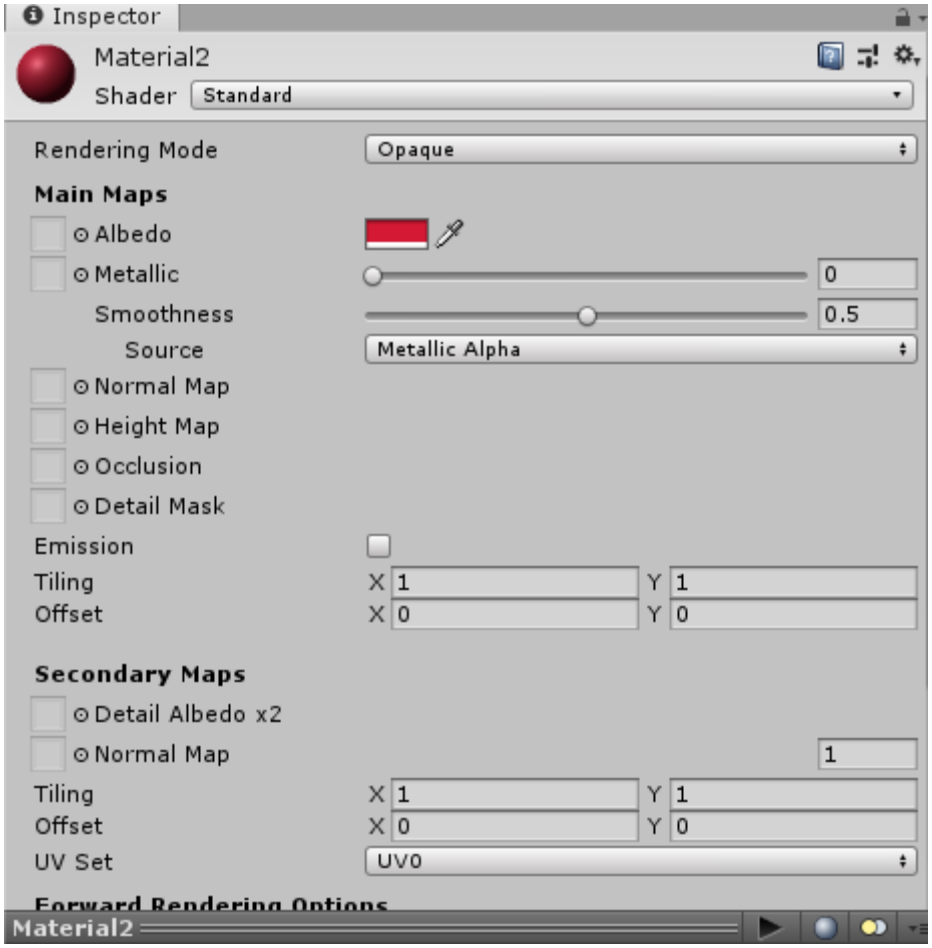


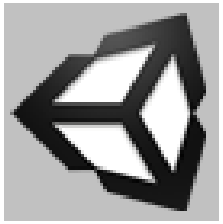
Material

- Quá trình render muốn thực hiện, phải cần có ba yếu tố là Material, Shader và Texture.
- Sau đây là các mối liên hệ giữa ba yếu tố này:
  - **Material**: là một thành phần biểu diễn diện mạo cho mô hình bằng cách kết hợp giữa đổ bóng (shader) và ảnh (texture).
  - **Shader**: Là các script nhỏ chứa các thuật toán tính toán màu và ánh sáng trên từng pixel.
  - **Texture**: Là các ảnh bitmap biểu diễn một phần đối tượng.

Sử dụng Material

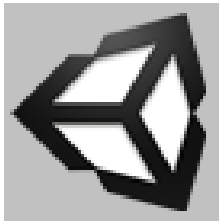
- Material được tạo bằng cách vào Assets->Create->Material.





Sử dụng Material

- Thuộc tính Shaders:
  - FX: Lighting and glass effects;
  - GUI and UI: For user interface graphics;
  - Mobile: Shader for mobile devices.
  - Nature: For trees and terrain;
  - Particles: Particle system effects;

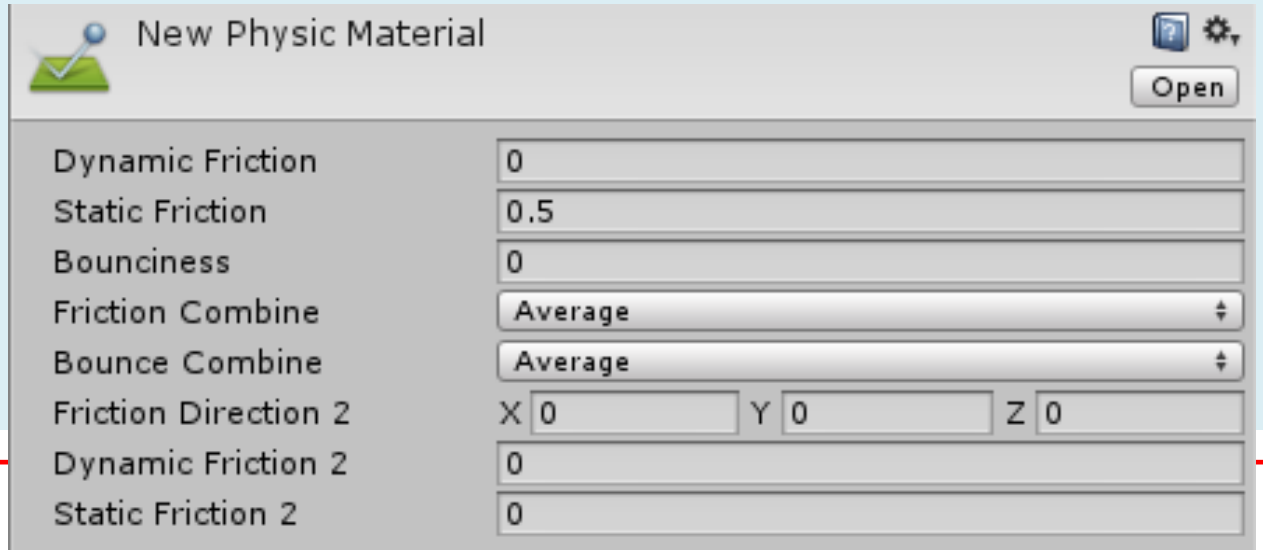


Sử dụng Material

- Thuộc tính Shaders (tiếp):
  - Skybox: For rendering background environments behind all geometry;
  - Sprites: For use with the 2D sprite system;
  - Toon: Cartoon-style rendering;
  - Unlit: For rendering that entirely bypasses all light & shadowing;
  - Legacy: The large collection of older shaders which were superseded by the Standard Shader

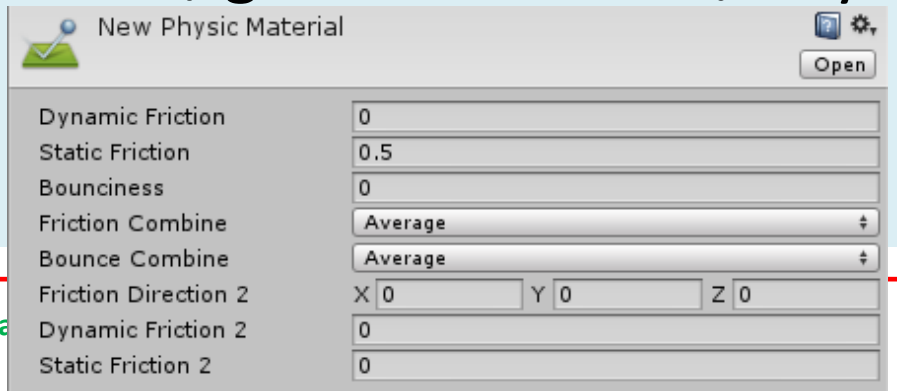
Physics Material

- Dùng để gán thuộc tính ma sát & độ nẩy lên đối tượng Game.
- Vào Assets->Create->Physics Material , kéo và rê Physics Material từ Project View vào một đối tượng có Collider trong Scene.



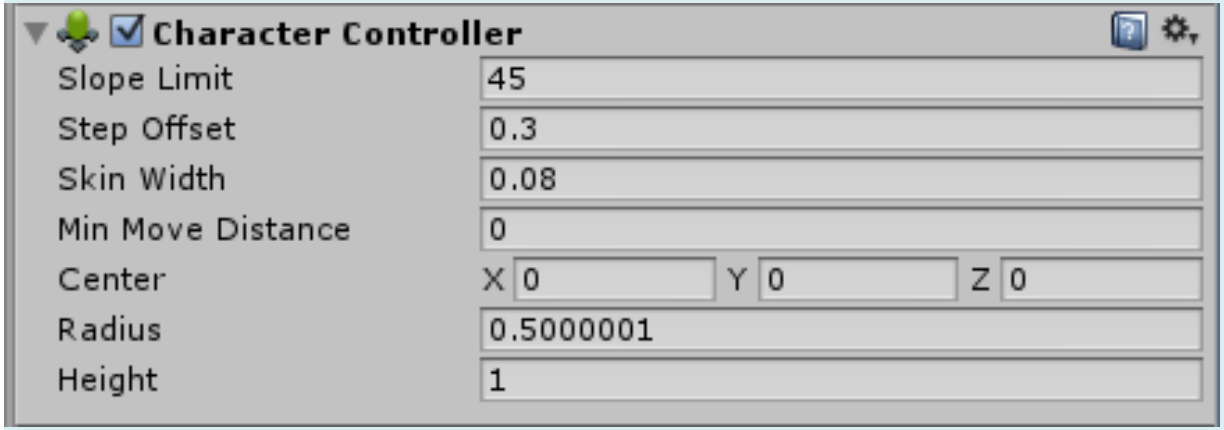
Physics Material

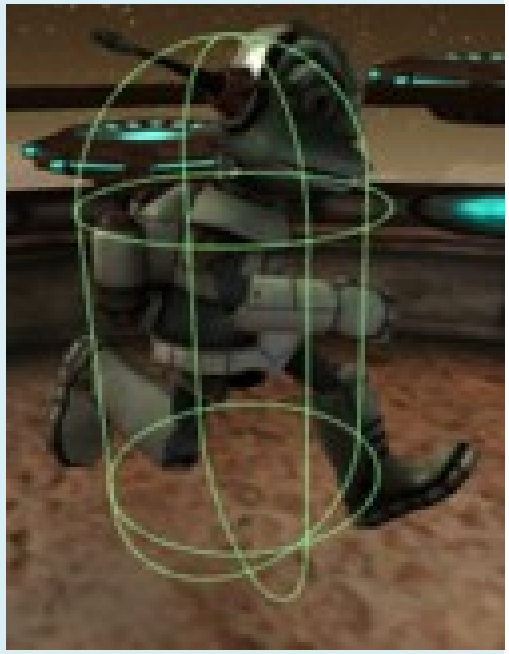
- Các thuộc tính chính:
  - **Dynamic Friction**: ma sát khi chuyển động, có giá trị từ 0 đến 1 (0: trượt trên băng).
  - **Static Friction**: ma sát khi đặt lên bề mặt (0->1).
  - **Bounciness**: độ nẩy (0->1: không có trọng lực).
  - **Friction Combine**: quan hệ giữa ma sát & collider.
  - **Bounce Combine**: quan hệ giữa ma sát & độ nẩy.
- Ví dụ demo



Character Controller

- Dùng để biểu diễn trạng thái nhân vật khi xây dựng một nhân vật.



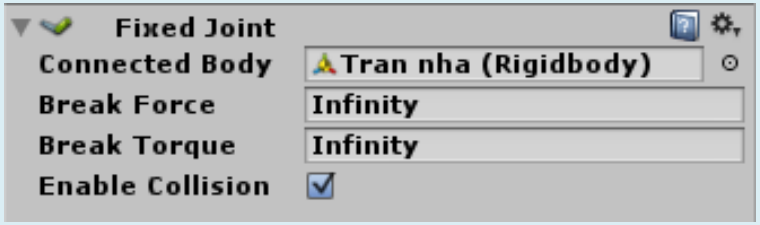


Character Controller

- Các thuộc tính:
  - Height, Radius: chiều cao và bán kính.
  - Slope Limit: giới hạn về độ dốc khi leo núi.
  - Step Offset: Độ cao của bậc thang khi nhân vật leo cầu thang.
  - Min Move Distance: Độ dài bước chân tối thiểu.
  - Skin width: Độ dày của da, dùng cho va chạm.
  - Center: Tâm của collider.

Fixed Joint

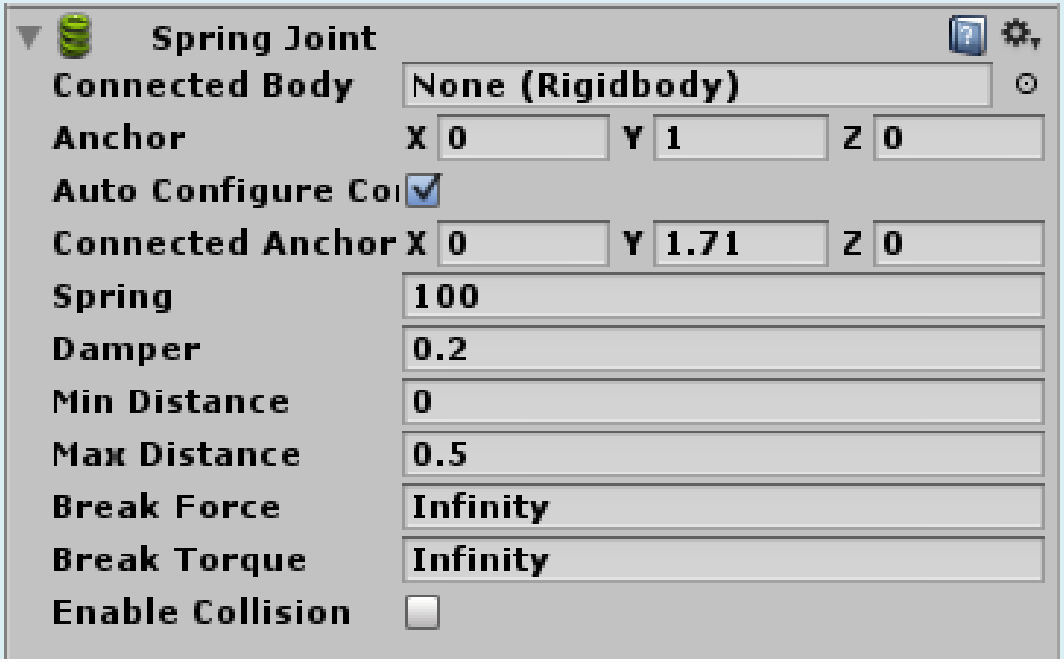
- Là một dạng liên kết đơn giản, giúp vật gắn cố định vào vật khác.



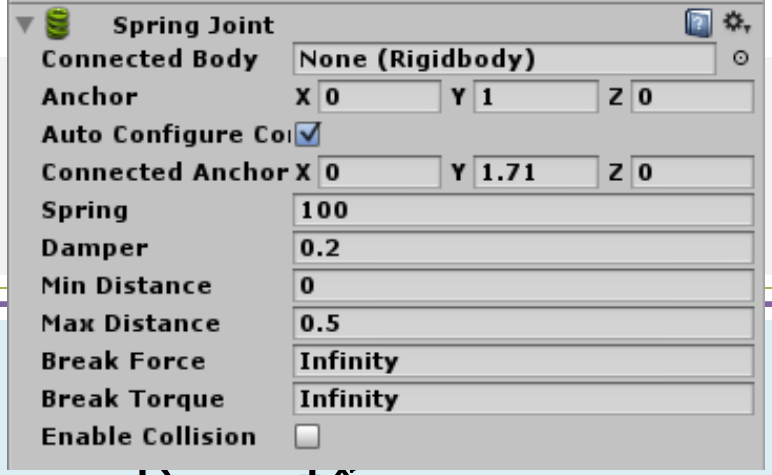
- Các thuộc tính:
  - Connected Body: Đối tượng (chứa rigidbody) dùng để neo.
  - Break Force: Lực dùng để bẻ gãy.
  - Break Torque: Lực momen quay bẻ gãy.
  - Enable Collision: cho phép va chạm hay không.

Sping Joint

- Spring Joint là dạng liên kết lò xo, có vị trí cân bằng ở chế độ nghỉ, nếu bị kéo, lò xo sẽ kéo ngược lại.



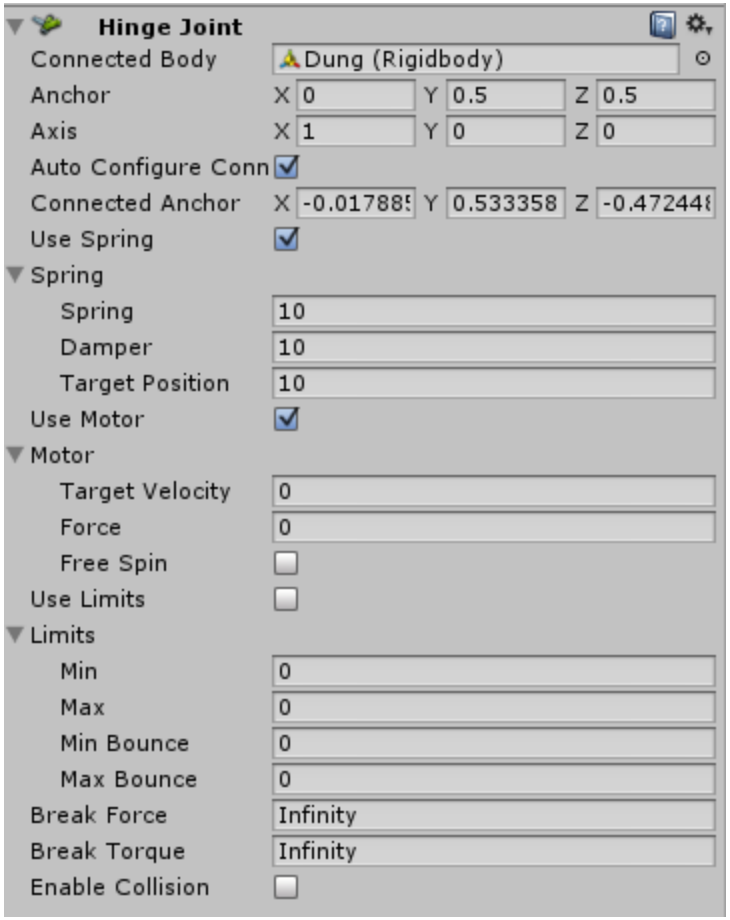
Sping Joint



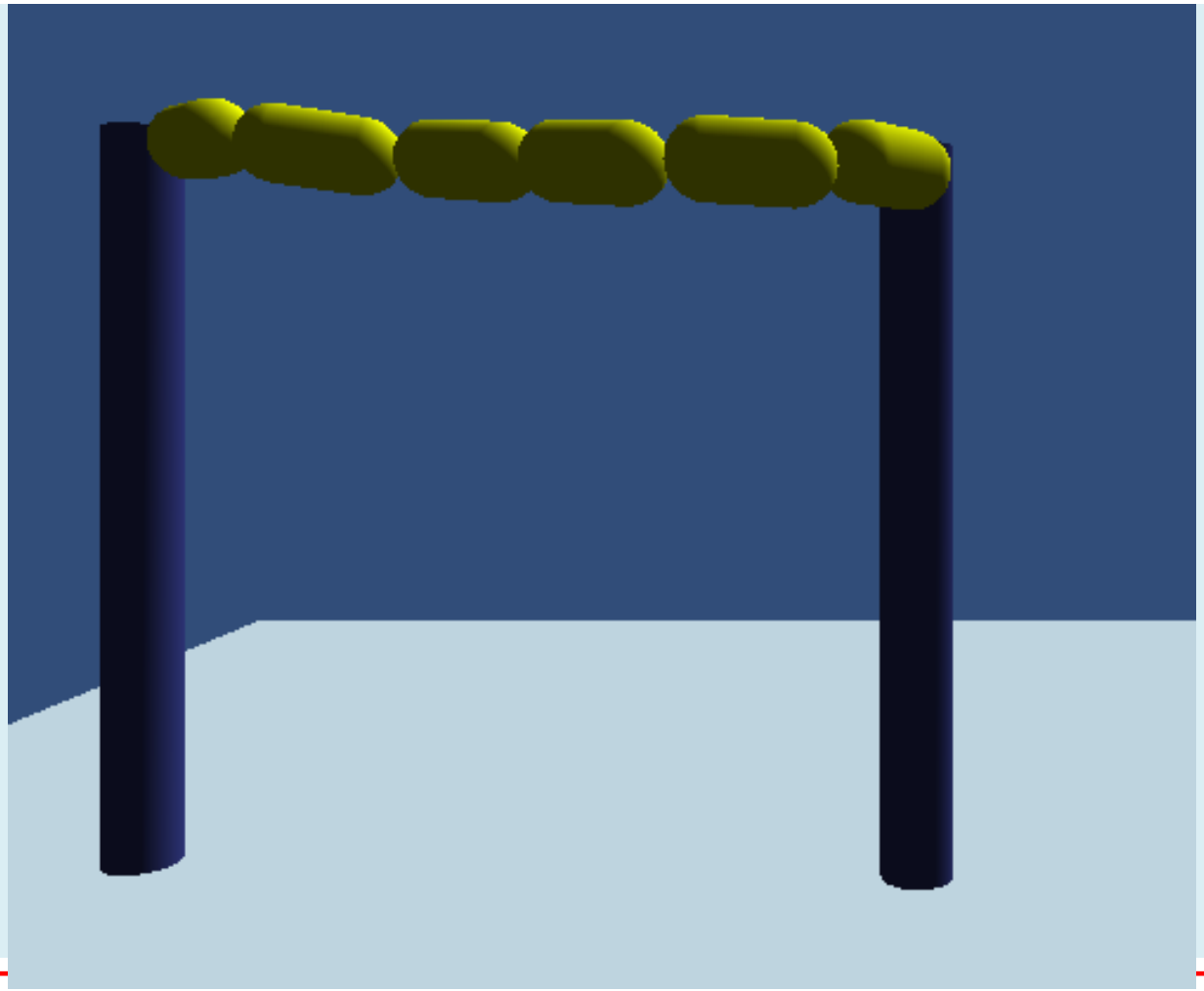
- Các thuộc tính:
  - **Connected Body**: Đối tượng dùng để neo.
  - **Anchor**: Điểm neo, dựa trên hệ tọa độ cục bộ.
  - **Connected Anchor**: Tâm cục bộ của khớp nối
  - **Spring**: độ cứng của lò xo.
  - **Damper**: mức độ nảy của lò xo.
  - **Min, max distance**: Khoảng cách để lò xo kéo vật trở lại.
  - **Force, Torque**: Lực & momen quay phá vỡ lò xo.

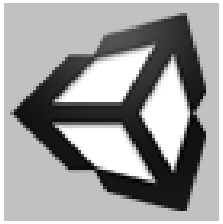
Hinge Joint

- Là dạng neo tương tự bản lề cánh cửa, dùng để neo một vật cố định vào một vật khác.
- Các thuộc tính của Hinge joint tương tự như Fixed & Spring joint



Ví dụ Hinge Joint



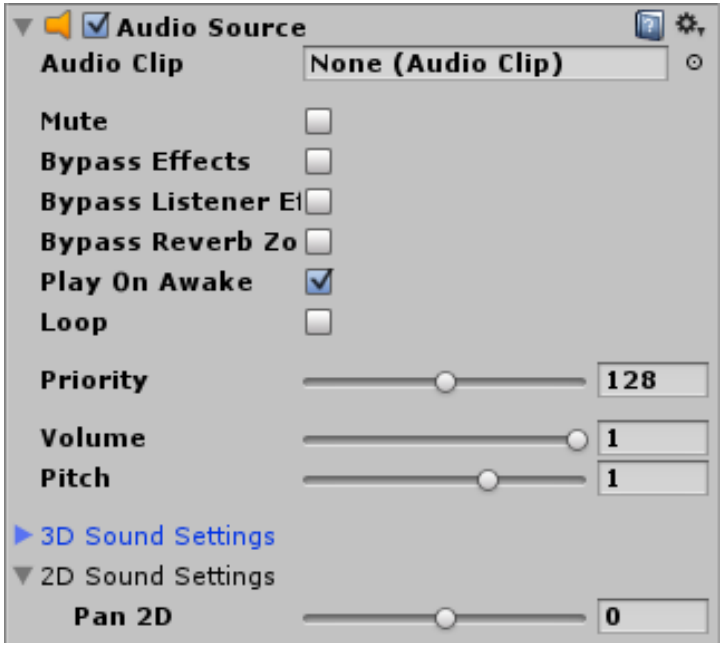


Audio

- Âm thanh là một trong những yếu tố không thể thiếu.
- Âm thanh nhằm kích thích các giác quan của người chơi giúp cho game trở nên thu hút và hấp dẫn hơn.
- Để quản lý Audio, người ta thường dùng Audio Clip & Audio Source

Audio Source Component

- Là một component được xây dựng để hỗ trợ phát các file âm thanh trong game.
- Chọn **Add > Component > Audio > Audio Source**

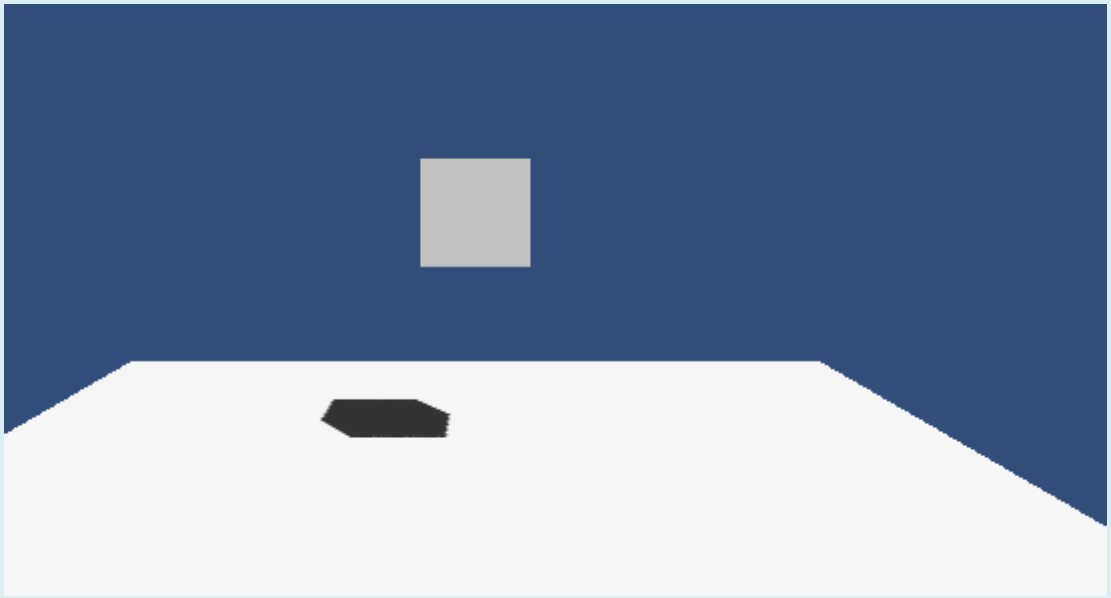


Audio Source Component

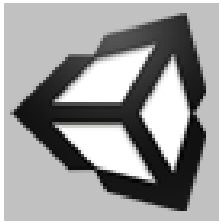
- Một số thuộc tính cơ bản:
  - **AudioClip**: Tập tin âm thanh.
  - **Play On Awake**: âm thanh được phát ngay khi Game Object được kích hoạt.
  - Loop: Tuỳ chọn lặp lại liên tục.
  - Mute: ngắt kết nối với các thiết bị output.

Audio vs Script

- Tạo đối tượng như sau:



- Gắn Rigid Body cho khối hộp, sàn đặt tên là Plane
- Tạo một script có tên là “PlayAudio”

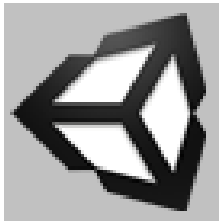


Audio vs Script

- Các script để khi khối hộp rơi xuống, sẽ chơi một âm thanh.
- Dùng một trong hai cách sau:

```
// Khai báo Audio Source
AudioSource au;
void Start()
{
    // Lấy audio Source
    au = GetComponent<AudioSource>();
}
// Nếu xảy ra va chạm
void OnCollisionEnter(Collision coll)
{
    if (coll.gameObject.name == "Plane")
    {
        Debug.Log("Chao");
        au.Play();
    }
}
```

```
// Khai báo một đối tượng audio clip
public AudioClip au;
void Start()
{
}
// Nếu xảy ra va chạm
void OnCollisionEnter(Collision coll)
{
    if (coll.gameObject.name == "Plane")
    {
        audio.PlayOneShot(au); // Lấy audio source và chơi
    }
}
```

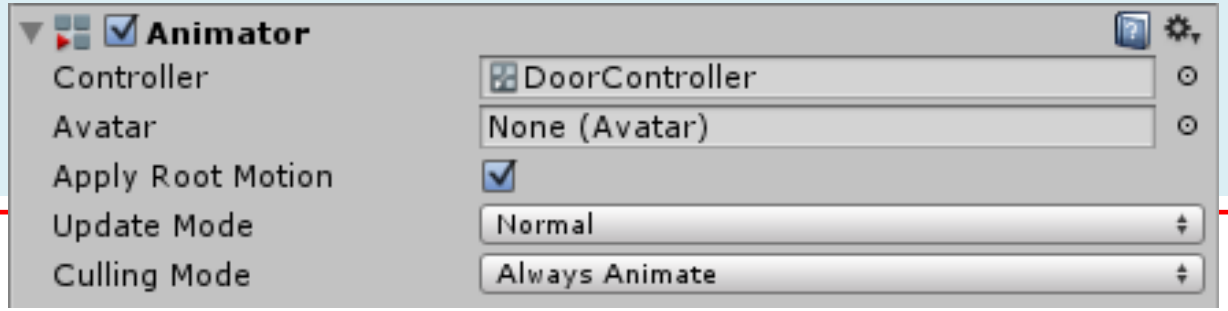


Animation

- Animation trong Unity là một công cụ giúp điều khiển sự hoạt động của game.
- Animation bao gồm:
  - Animator
  - Animator Controller
  - State
  - Transitions
  - Parameters
  - Animator Scripting

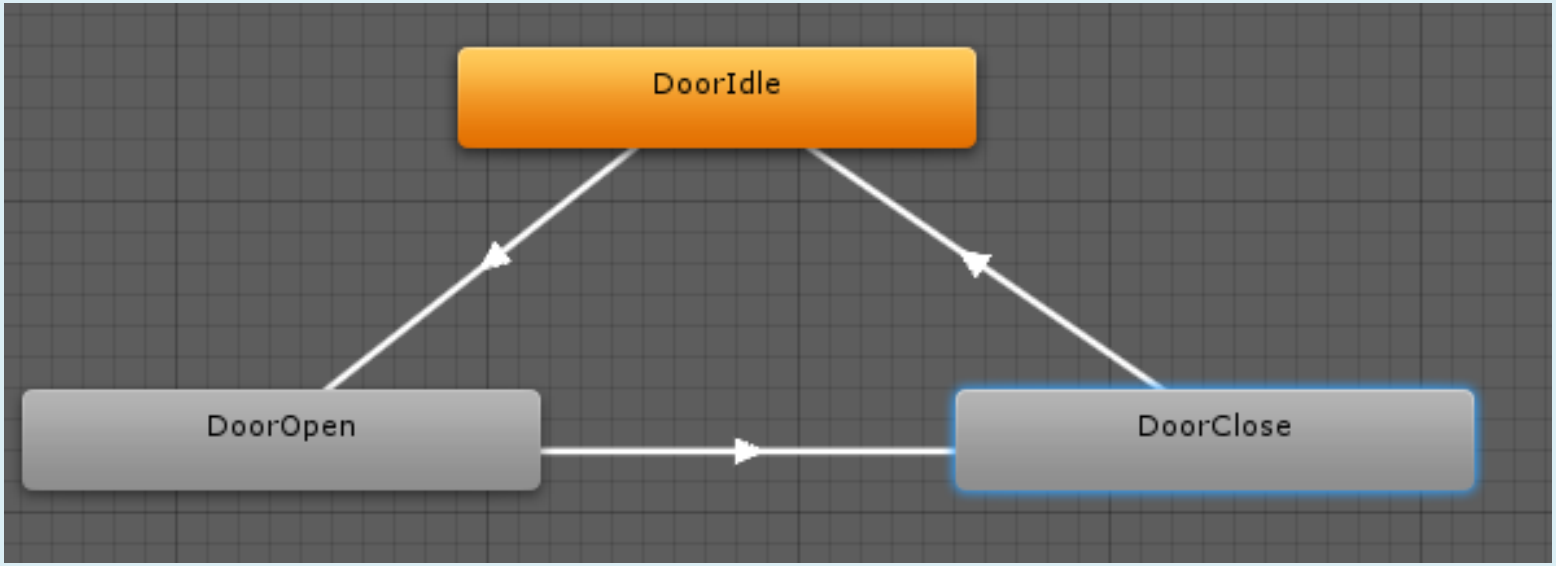
Animator

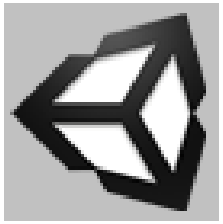
- Là component được gắn với hoạt hình.
- Vai trò Animator là chuyển thông tin từ game object vào controller.
- Các thuộc tính chính:
  - *Animator Controller :* điều khiển hoạt hình để chuyển trạng thái.
  - *Avatar*: dùng để thay đổi hoạt hình trong Model.



Animator Controller

- Là component của Animator, nơi các trạng thái và logic của hoạt hình được tạo ra.
- Animator là một máy trạng thái.



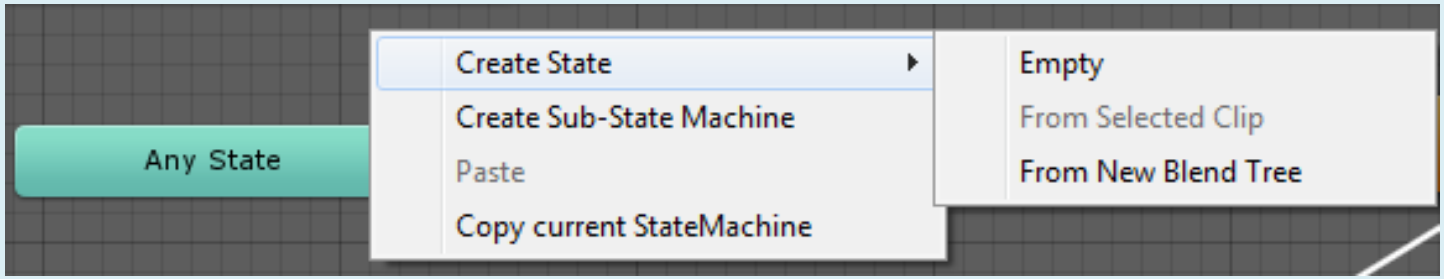


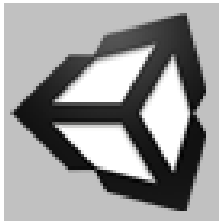
State

- Là một hình thức chuyển đổi giữa đối tượng này qua đối tượng khác.
- Một số trạng thái chính:
  - *Animations*: là một hoạt hình đơn, có những thuộc tính như: di chuyển, tốc độ và Mirror.
  - *Blendspaces:* Điều khiển nhiều animation khác nhau trong một nút

State

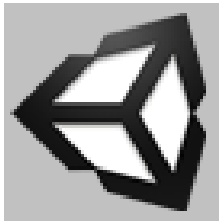
- Một số trạng thái chính:
  - *Default State:* trạng thái ban đầu, mặc định
  - *Any State:* cho phép di chuyển đến trạng thái này từ bất kỳ trạng thái khác.
  - *Exit:* Sẽ thông báo khi trạng thái này chuyển sang trạng thái khác.





Transitions

- Dùng để chuyển từ trạng thái này sang trạng thái kia.
- Click chuột phải vào Animator, chọn “Create Transition” và kéo sang trạng thái khác.
- Sau đây là những điều kiện của Transition:
  - **Exit Time**: xác định thời điểm thoát khỏi hoạt hình khi trạng thái chạy xong.
  - **Parameters**: một số tham số nhằm giúp thõa mãn trạng thái khi thay đổi.

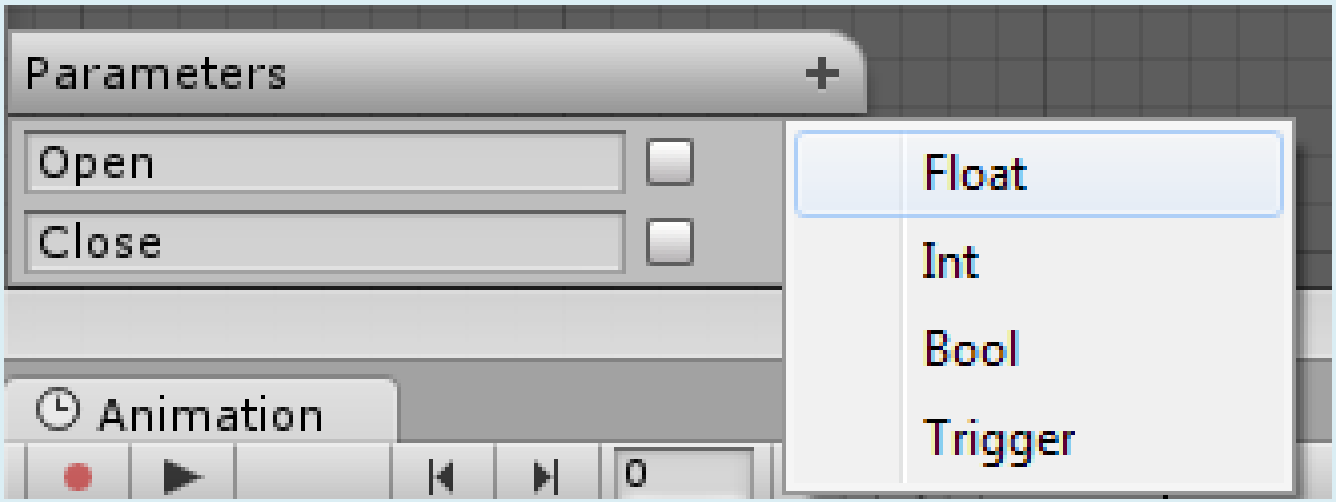


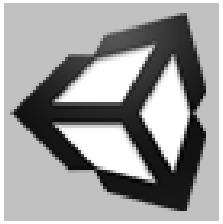
Parameters

- Là các tham số dùng để điều khiển hoạt động của nhân vật.
- Để tạo ra parameter, mở animation controller, nhấn nút "+" để tạo các tham số.
- Những parameter có thể dùng:
  - **Float:** điều khiển tốc độ giữa hai trạng thái.
  - **Integer:** xác định liệu trạng thái có thể được di chuyển tùy vào giá trị.
  - **Bool:** tạo ra điều kiện chuyển đổi trạng thái.

Parameters

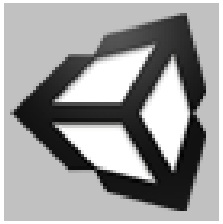
- Những parameter có thể dùng:
  - **Trigger:** là trạng thái sự kiện dạng true/false sẽ lặp lại hoạt hình liên tiếp chừng nào nó không chuyển thành giá trị khác.





Animator Scripting

- Sử dụng trong Code, để điều khiển các Animation một các linh hoạt.
- Một số phương thức thường dùng:
  - *GetComponent<Animator>():* lấy ra thành phần Animator của đối tượng.
  - *Animator.SetInteger(String, Int):* áp dụng lên trạng thái(tên string) với giá trị số.
  - *Animator.SetTrigger(String):* Truyền giá trị mặc định lên trạng thái sự kiện.



- Demo