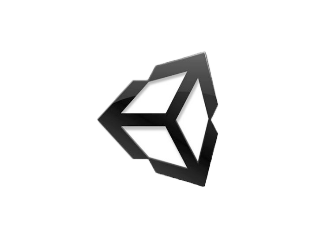
Nguyễn Văn Mạnh



UNITY

# Chương 1.Phụ lục

[**I.** **Graphics** 1](#_Toc161158923)

[**1.** Lighting 1](#_Toc161158924)

[**2.** Cameras 2](#_Toc161158925)

[**3.** Material 3](#_Toc161158926)

[**4.** Shaders and Textures 4](#_Toc161158927)

[**II.** **Physcis** 4](#_Toc161158928)

[**1.** RigidBody 4](#_Toc161158929)

[**2.** Collider 8](#_Toc161158930)

[**3.** Physic materials 9](#_Toc161158931)

[**4.** Physic join 10](#_Toc161158932)

[**III.** **Scripting** 12](#_Toc161158933)

[**IV.** **Audio** 15](#_Toc161158934)

[**1.** AudioSource 15](#_Toc161158935)

[**2.** AudioListener 18](#_Toc161158936)

[**3.** AudioClip 19](#_Toc161158937)

[**V.** **Animation** 24](#_Toc161158938)

[1. Animator 24](#_Toc161158939)

[2. Animation 28](#_Toc161158940)

[3. AnimationClip 28](#_Toc161158941)

[4. AvatarController 29](#_Toc161158942)

[5. AvatarMaskt 30](#_Toc161158943)

[**VI.** **Navigation** 31](#_Toc161158944)

[**VII.** **UI** 32](#_Toc161158945)

[**VIII.** **Assets** 33](#_Toc161158946)

# Chương 2. Nội dung

1. **Graphics**
2. Lighting

Lighting trong Unity là quá trình tạo ra hiệu ứng ánh sáng và bóng trên các đối tượng trong môi trường 3D của trò chơi hoặc ứng dụng. Nó là một phần quan trọng trong việc tạo ra một không gian 3D sống động và thú vị.



\*Cơ chế của Lighting trong Unity:

-Phân Tán Ánh Sáng (Global Illumination):

Unity hỗ trợ các kỹ thuật phân tán ánh sáng như Enlighten hoặc Progressive Lightmapper để tạo ra hiệu ứng ánh sáng tự nhiên và phản xạ trên các bề mặt trong môi trường.

Các kỹ thuật này tính toán cách ánh sáng phản xạ và lan truyền qua các vật liệu và không gian.

-Các nguồn Ánh Sáng (Light Sources):

Unity hỗ trợ nhiều loại nguồn ánh sáng như đèn điểm, đèn spot, đèn hình học, và đèn dải.

Các nguồn ánh sáng này phát ra ánh sáng và tạo ra các hiệu ứng chiếu sáng và bóng trên các đối tượng trong môi trường.

-Vật Liệu Phản Xạ Ánh Sáng (Material Reflectance):

Các vật liệu trong Unity được thiết lập để phản xạ ánh sáng theo các thuộc tính khác nhau như màu sắc, độ bóng, độ mờ, và ánh sáng phản xạ.

-Shadow Mapping:

Unity sử dụng kỹ thuật shadow mapping để tạo ra các bóng đổ tự nhiên từ các nguồn ánh sáng.

Các bóng đổ được tính toán và vẽ lên bề mặt phía dưới đối tượng, tạo ra các hiệu ứng bóng đổ thực tế.

-Baking và Real-Time Lighting:

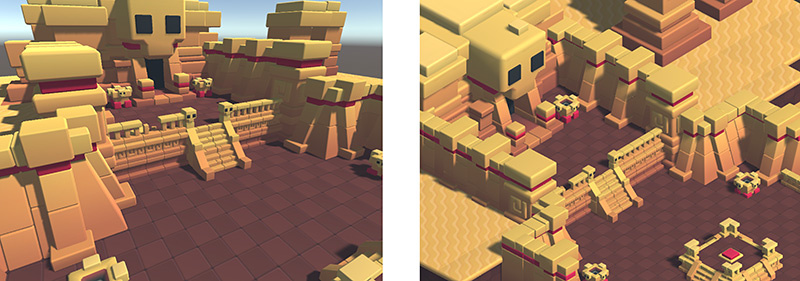
Unity cho phép sử dụng cả hai phương pháp baking ánh sáng (tính toán trước và lưu trữ ánh sáng trong bản đồ ánh sáng) và ánh sáng thời gian thực (tính toán ánh sáng trong thời gian chạy) để tạo ra hiệu ứng ánh sáng.

Baking được sử dụng cho các môi trường tĩnh, trong khi ánh sáng thời gian thực thích hợp cho các hiệu ứng động và tương tác người chơi.

Qua cơ chế này, Unity cho phép người phát triển tạo ra các hiệu ứng ánh sáng phong phú và đa dạng để tạo ra môi trường 3D sống động và chân thực.

1. Cameras

Camera là một thành phần quan trọng để hiển thị môi trường 3D cho người chơi. Camera là một đối tượng ảo trong không gian 3D, và nó được sử dụng để xác định góc nhìn và khu vực của màn hình mà người chơi sẽ nhìn thấy.



\*Cơ chế của Cameras trong Unity:

-Xác định Góc Nhìn (View Frustum):

Camera trong Unity xác định một khu vực 3D gọi là "View Frustum" hoặc "Hình hộp nhìn".

Khu vực này định rõ những đối tượng nào sẽ được hiển thị trên màn hình và những đối tượng nào sẽ bị cắt bỏ do nằm ngoài khu vực này.

-Thu Thập Hình Ảnh (Rendering):

Camera thu thập thông tin về môi trường 3D từ góc nhìn của nó.

Thông tin này sau đó được chuyển đến quá trình rendering để tạo ra hình ảnh cuối cùng mà người chơi sẽ thấy trên màn hình.

-Các Thuộc Tính Camera:

Unity cung cấp nhiều thuộc tính để tùy chỉnh Camera, bao gồm vị trí, hướng, góc nhìn, tỷ lệ khung hình, và khoảng cách cắt.

Các thuộc tính này có thể được điều chỉnh để tạo ra các hiệu ứng đặc biệt như chuyển động, zoom, và hiệu ứng chuyển cảnh.

-Chế độ Render:

Camera có thể được cấu hình để render ở chế độ Perspective (theo góc nhìn phối cảnh) hoặc Orthographic (theo góc nhìn phẳng).

-Multiple Cameras:

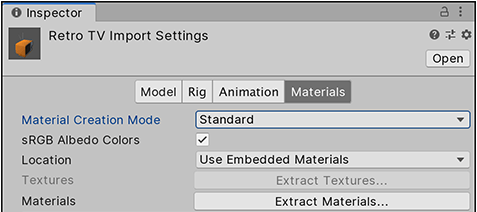
Unity hỗ trợ sử dụng nhiều Camera trong cùng một scene.

Điều này cho phép tạo ra các hiệu ứng như hồng ngoại, góc nhìn đa dạng, và chế độ xem đa phương tiện.

Camera trong Unity chịu trách nhiệm xác định góc nhìn và nội dung được hiển thị trên màn hình, tạo ra trải nghiệm thực tế và hấp dẫn cho người chơi.

1. Material

Là một thành phần quan trọng để xác định cách mà đối tượng trong môi trường 3D sẽ được hiển thị, bao gồm màu sắc, ánh sáng, bóng, và các thuộc tính khác.



Cơ chế của Material trong Unity:

-Thông Tin Vật Liệu (Material Properties):

Mỗi Material trong Unity bao gồm một tập hợp các thông tin vật liệu, bao gồm màu sắc, texture, ánh sáng, độ bóng, độ mờ, và các thuộc tính khác.

-Shader Programs:

Material trong Unity sử dụng Shader để xác định cách mà nó sẽ hiển thị trên màn hình.

Shader là một chương trình đặc biệt được viết bằng ngôn ngữ lập trình shader (như HLSL hoặc Cg) để mô tả cách mà vật liệu sẽ phản ứng với ánh sáng và các hiệu ứng khác.

-Rendering Pipeline:

Khi một đối tượng được render, thông tin về Material của nó được chuyển đến Rendering Pipeline của Unity.

Pipeline sử dụng Shader được gán cho Material để tính toán cách mà đối tượng sẽ được hiển thị, bao gồm cả màu sắc, ánh sáng, bóng, và các hiệu ứng khác.

-Texture Mapping:

Material có thể được thiết lập để sử dụng texture, là các hình ảnh được áp dụng lên bề mặt của đối tượng để tạo ra các hiệu ứng phức tạp như vân đá, gỗ, hoặc áo.

-Phản ứng với Ánh Sáng (Lighting Response):

Material có thể được cấu hình để phản ứng với ánh sáng theo nhiều cách khác nhau, từ mờ đến phản xạ hoặc lấp lánh.

Điều này cho phép tạo ra các hiệu ứng ánh sáng đa dạng và thú vị trên các đối tượng.

1. Shaders and Textures
2. **Physcis**
3. RigidBody

Sử dụng thành phần Rigidbody để áp dụng Rigidbody cho GameObject của bạn. Rigidbody cung cấp một phương pháp dựa trên vật lý để điều khiển chuyển động và vị trí của GameObject. Thay vì các thuộc tính Biến đổi, bạn có thể sử dụng các lực và mô-men xoắn vật lý mô phỏng để di chuyển GameObject và để công cụ vật lý tính toán kết quả. Để biết thêm thông tin, xem Giới thiệu về Vật lý vật rắn. Để theo dõi hiệu suất của Rigidbody, hãy sử dụng công cụ Trực quan gỡ lỗi vật lý.

| **Property** | **Function** |
| --- | --- |
| **Mass** | Xác định khối lượng của GameObject (tính bằng kilôgam). Khối lượng được đặt thành 1 theo mặc định. Giống như trong đời thực, khối lượng không ảnh hưởng đến tốc độ rơi của một vật thể dưới tác dụng của trọng lực. Để mô phỏng lực cản làm chậm chuyển động, hãy sử dụng Kéo. |
| **Drag** | Xác định tốc độ phân rã của vận tốc tuyến tính của Vật rắn, để mô phỏng lực cản, lực cản không khí hoặc ma sát. Giá trị thấp tạo ra tốc độ phân rã chậm hơn, do đó GameObject di chuyển nhanh hơn trong thời gian dài hơn (điều này rất hữu ích để mô phỏng các vật thể nặng trong thế giới thực). Giá trị cao tạo ra tốc độ phân rã nhanh hơn, do đó GameObject chậm lại trong một khoảng thời gian ngắn (điều này rất hữu ích để mô phỏng các vật thể nhẹ trong thế giới thực). |
| **Angular Drag** | Xác định tốc độ phân rã của vận tốc quay của Vật cứng, để mô phỏng lực cản, lực cản không khí hoặc ma sát. Giá trị thấp tạo ra tốc độ phân rã chậm hơn, do đó GameObject di chuyển nhanh hơn trong thời gian dài hơn (điều này rất hữu ích để mô phỏng các vật thể nặng trong thế giới thực). Giá trị cao tạo ra tốc độ phân rã nhanh hơn, do đó GameObject chậm lại trong một khoảng thời gian ngắn (điều này rất hữu ích để mô phỏng các vật thể nhẹ trong thế giới thực). Lưu ý rằng bạn không thể làm cho GameObject ngừng quay chỉ bằng cách đặt Angular Drag của nó thành vô cùng. Angular Drag được đặt thành 0,05 theo mặc định. |
| **Automatic Center Of Mass** | Kích hoạt Khối tâm tự động để sử dụng khối tâm dự đoán của hệ thống vật lý cho Vật cứng, dựa trên hình dạng và tỷ lệ của nó. Tắt để đặt tọa độ X, Y và Z của riêng bạn cho khối tâm. |
| **Automatic Tensor** | Bật Bộ kéo căng tự động để sử dụng phép quay tensor và tensor dự đoán của hệ thống vật lý cho Vật cứng, dựa trên tất cả các máy va chạm được kết nối. Giống như khối lượng, 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. Thay vào đó, hãy tắt chức năng đặt tọa độ X, Y và Z của riêng bạn cho tensor (xem các thuộc tính bên dưới). |
| - **Inertia Tensor** | Xác định tensor quán tính của vật rắn này. Giá trị Tensor quán tính càng cao thì càng cần nhiều mô-men xoắn để làm cho Vật cứng quay trên trục của nó. |
| - **Inertia Tensor Rotation** | Xác định góc quay của tensor quán tính |
| **Use Gravity** | Chuyển đổi tác dụng của trọng lực lên Rigidbody. Nếu được bật, hệ thống vật lý sẽ áp dụng một lực để di chuyển GameObject theo hướng trọng lực mô phỏng (theo mặc định là dọc theo trục y). Sử dụng Gravity được bật theo mặc định. |
| **Is Kinematic** | Chuyển đổi giữa chuyển động dựa trên vật lý và chuyển động động học cho GameObject. Khi Is Kinematic được bật, hệ thống vật lý không thể tác dụng lực để di chuyển hoặc xoay GameObject, thay vào đó, Unity chỉ có thể di chuyển và xoay nó thông qua Transform. Xem phần Giới thiệu về vật lý vật cứng: Vật cứng không có chuyển động dựa trên vật lý để biết chi tiết. Theo mặc định, Kinetic bị tắt. |
| **Interpolate** | Cài đặt Nội suy trên Rigidbody cung cấp hai tùy chọn để làm mượt sự xuất hiện của chuyển động của Rigidbody nếu nó có vẻ giật cục trong thời gian chạy. Các tùy chọn này là Nội suy và Ngoại suy. Cả phép nội suy và ngoại suy đều tính toán tư thế của Vật rắn (nghĩa là vị trí và góc quay) giữa các lần cập nhật vật lý. Bạn nên chọn cái nào tùy thuộc vào tùy chọn nào tạo ra kết quả hình ảnh tốt nhất cho trường hợp sử dụng của bạn. Để biết thông tin chi tiết về thuộc tính Nội suy, hãy xem Áp dụng phép nội suy cho Thân cứng.. |
| - **None** | Không áp dụng phép nội suy hoặc ngoại suy. Đây là tùy chọn mặc định. |
| - **Interpolate** | Sử dụng tư thế và vận tốc của Thân cứng từ hai bản cập nhật vật lý trước đó để tính toán và áp dụng tư thế của Thân cứng trong khung hiện tại. Nội suy chính xác hơn Ngoại suy, nhưng nó có độ trễ thời gian của một lần cập nhật vật lý.  Nó thường là tốt nhất cho những tình huống mà độ chính xác là quan trọng; ví dụ: nếu vận tốc của Vật cứng thay đổi hoặc nếu có các yếu tố vật lý khác ảnh hưởng đến chuyển động của Vật cứng. |
| - **Extrapolate** | Sử dụng tư thế và vận tốc của Vật cứng nhắc từ bản cập nhật vật lý trước đó và dự đoán tư thế của Vật cứng cứng trong bản cập nhật vật lý tiếp theo để tính toán và dự đoán tư thế trong khung hình hiện tại.  Phép ngoại suy làm cho Vật cứng có vẻ di chuyển về phía trước một chút so với vị trí cần đến và có thể hơi không chính xác. Nó thường là tốt nhất cho những tình huống mà độ chính xác không quan trọng; ví dụ: nếu Vật cứng nhắc di chuyển với vận tốc không đổi và không có yếu tố vật lý nào khác ảnh hưởng đến chuyển động của Vật cứng nhắc. |
| **Collision Detection** | Xác định cách hệ thống vật lý phát hiện các va chạm giữa máy va chạm của vật cứng này và các máy va chạm khác trong hiện trường.  Unity tạo ra một va chạm trên mỗi cặp máy va chạm và xác định phương pháp phát hiện va chạm dựa trên thuộc tính Phát hiện va chạm này. Theo mặc định, Phát hiện va chạm được đặt thành Rời rạc. Để biết thêm thông tin về từng loại phát hiện va chạm, hãy xem Phát hiện va chạm liên tục.. |
| - **Discrete** | Hệ thống vật lý sử dụng khả năng phát hiện va chạm rời rạc để tính toán va chạm cho máy va chạm của vật thể cứng này. Chọn Rời rạc nếu Rigidbody này không liên quan đến bất kỳ va chạm chuyển động nhanh nào. Phát hiện va chạm rời rạc không đòi hỏi nhiều tính toán. |
| - **Continuous** | Hệ thống vật lý sử dụng CCD dựa trên quét để tính toán các va chạm giữa máy va chạm của vật cứng này và bất kỳ máy va chạm tĩnh nào (nghĩa là các máy va chạm không có vật thể cứng liên quan). Chọn Liên tục nếu Vật cứng này có liên quan đến các va chạm chuyển động nhanh với máy va chạm tĩnh. CCD dựa trên quét có tính toán chuyên sâu hơn so với suy đoán rời rạc hoặc liên tục. |
| - **Continuous Dynamic** | Hệ thống vật lý sử dụng CCD dựa trên quét để tính toán các va chạm giữa máy va chạm của Vật cứng này và tất cả các máy va chạm khác, ngoại trừ những máy được đặt thành Phát hiện va chạm rời rạc. Chọn Động liên tục nếu Vật cứng này có liên quan đến các vụ va chạm chuyển động nhanh với bất kỳ máy va chạm nào. CCD dựa trên quét có tính toán chuyên sâu hơn so với suy đoán rời rạc hoặc liên tục. |
| - **Continuous Speculative** | Hệ thống vật lý sử dụng tính năng phát hiện va chạm liên tục mang tính suy đoán để tính toán các va chạm giữa máy va chạm của vật cứng này và tất cả các máy va chạm khác. Chọn Suy đoán liên tục nếu độ chính xác va chạm không quan trọng đối với Vật cứng này. Phát hiện xung đột suy đoán đòi hỏi nhiều tính toán hơn so với Rời rạc, nhưng ít tính toán hơn so với Động liên tục hoặc Động liên tục. |
| **Constraints** | Đặt các hạn chế đối với chuyển động của vật rắn |
| - **Freeze Position** | Dừng Rigidbody di chuyển trong các trục X, Y và Z của thế giới một cách có chọn lọc. |
| - **Freeze Rotation** | Dừng Rigidbody quay quanh các trục X, Y và Z cục bộ một cách có chọn lọc. |

1. Collider

Khi hai trình va chạm thực hiện liên hệ, bạn có thể gọi các chức năng để kích hoạt các sự kiện khác trong dự án của bạn thông qua kịch bản. Hai loại chức năng thiết yếu để tương tác va chạm là Oncollision và Ontrigger.

| **Topic** | **Description** |
| --- | --- |
| [Use collisions to trigger other events](https://docs.unity3d.com/Manual/collider-interactions-other-events.html) | Tổng quan về các sự kiện và trình kích hoạt xung đột cũng như API tập lệnh bạn cần sử dụng. |
| [OnCollision events](https://docs.unity3d.com/Manual/collider-interactions-oncollision.html) | Cách máy va chạm có thể gọi các sự kiện khi chúng va chạm vật lý. |
| [OnTrigger events](https://docs.unity3d.com/Manual/collider-interactions-ontrigger.html) | Cách máy va chạm có thể gọi các sự kiện khi một người đi vào không gian của người khác trong một vụ va chạm phi vật lý. |
| [Create and configure a trigger collider](https://docs.unity3d.com/Manual/collider-interactions-create-trigger.html) | Tạo trình thu thập trình kích hoạt và định cấu hình GameObject được liên kết của nó để gọi chính xác các sự kiện trên các tương tác trình kích hoạt. |
| [Example scripts for collider events](https://docs.unity3d.com/Manual/collider-interactions-example-scripts.html) | Tập lệnh mẫu cho các sự kiện OnCollision và OnTrigger. Các tập lệnh mẫu này thể hiện một số cách sử dụng tiềm năng cho các sự kiện và tương tác xung đột theo tập lệnh. |

1. Physic materials

Vật liệu Vật lý là tài sản vật chất mà bạn có thể đặt trên GameObject . Vật liệu xác định các đặc tính trên bề mặt của máy va chạm, chẳng hạn như ma sát và độ nảy. Để tạo Vật liệu Vật lý, hãy đi tới Tài sản > Tạo > Vật liệu Vật lý, sau đó kéo Vật liệu Vật lý từ cửa sổ Dự án vào máy va chạm trong cảnh. Nếu không có bộ Vật liệu Vật lý, máy va chạm sẽ sử dụng cài đặt bề mặt mặc định. Để điều chỉnh cài đặt mặc định của dự án, hãy sử dụng Cài đặt Vật lý.

| **Property** | **Description** |
| --- | --- |
| **Dynamic Friction** | Xác định mức độ ma sát của bề mặt máy va chạm với máy va chạm khác khi các máy va chạm 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 Kết hợp ma sát (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. 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 Kết hợp ma sát (bên dưới). |
| **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. Theo mặc định, Độ nảy được đặt thành 0. 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 Kết hợp ma sát.  Lưu ý rằng các phép tính gần đúng nảy của hệ vật lý vẫn có thể bổ sung thêm một lượng nhỏ năng lượng cho mô phỏng. |
| **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. Theo mặc định, Kết hợp ma sát được đặt thành Trung bình. Để biết chi tiết, hãy tham khảo Cách kết hợp các giá trị bề mặt máy va chạm.. |
| **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. Theo mặc định, Kết hợp Thoát được đặt thành Trung bình. Để biết chi tiết, hãy tham khảo Cách kết hợp các giá trị bề mặt máy va chạm. |

1. Physic join

Join kết nối một Rigidbody với một Rigidbody khác hoặc một điểm cố định trong không gian. Các khớp tác dụng lực làm di chuyển vật thể cứng và các giới hạn của khớp hạn chế chuyển động đó. Các khớp cung cấp cho các vật cứng các mức độ tự do sau:

A diagram of a diagram

Description automatically generated

| **Property:** | **Function:** |
| --- | --- |
| [Character Joint](https://docs.unity3d.com/Manual/class-CharacterJoint.html) | Mô phỏng khớp cầu và ổ cắm, giống như khớp hông hoặc vai. Hạn chế chuyển động cứng nhắc của cơ thể dọc theo tất cả các bậc tự do tuyến tính và cho phép tất cả các góc tự do. Các vật cứng được gắn vào Khớp ký tự sẽ định hướng quanh mỗi trục và xoay quanh một điểm gốc chung. |
| [Configurable Joint](https://docs.unity3d.com/Manual/class-ConfigurableJoint.html) | Mô phỏng bất kỳ khớp xương nào, giống như khớp xương trong một con ragdoll. Bạn có thể cấu hình khớp này để buộc và hạn chế chuyển động của cơ thể cứng nhắc ở bất kỳ mức độ tự do nào. |
| [Fixed Joint](https://docs.unity3d.com/Manual/class-FixedJoint.html) | Hạn chế chuyển động của một vật rắn theo chuyển động của vật rắn mà nó gắn vào. Điều này hữu ích khi bạn cần các vật thể cứng nhắc dễ dàng tách rời khỏi nhau hoặc bạn muốn kết nối chuyển động của hai vật thể cứng nhắc mà không cần quản lý theo hệ thống phân cấp Biến đổi. |
| [Hinge Joint](https://docs.unity3d.com/Manual/class-HingeJoint.html) | Gắn một vật rắn vào một vật rắn khác hoặc một điểm trong không gian có chung một gốc tọa độ và cho phép các vật rắn quay quanh một trục cụ thể từ gốc đó. Hữu ích cho việc mô phỏng cửa và khớp ngón tay. |
| [Spring Joint](https://docs.unity3d.com/Manual/class-SpringJoint.html) | Giữ các vật cứng cách xa nhau nhưng để khoảng cách giữa chúng giãn ra một chút. Lò xo hoạt động giống như một miếng đàn hồi cố gắng kéo hai điểm neo lại với nhau về cùng một vị trí. |

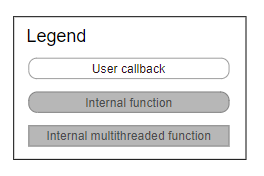
Khớp 2D có chữ 2D ở cuối tên (ví dụ Hinge Joint 2D). Để biết tóm tắt về các khớp 2D, hãy xem tài liệu về Khớp 2D. Khớp cũng có các tùy chọn khác mà bạn có thể kích hoạt để tạo ra các hiệu ứng cụ thể. Ví dụ: bạn có thể đặt một khớp bị gãy khi một vật cứng tác dụng một lực lên nó vượt quá một ngưỡng nhất định. Một số khớp cho phép lực truyền động xảy ra giữa các Thân cứng được kết nối để khiến chúng chuyển động tự động.

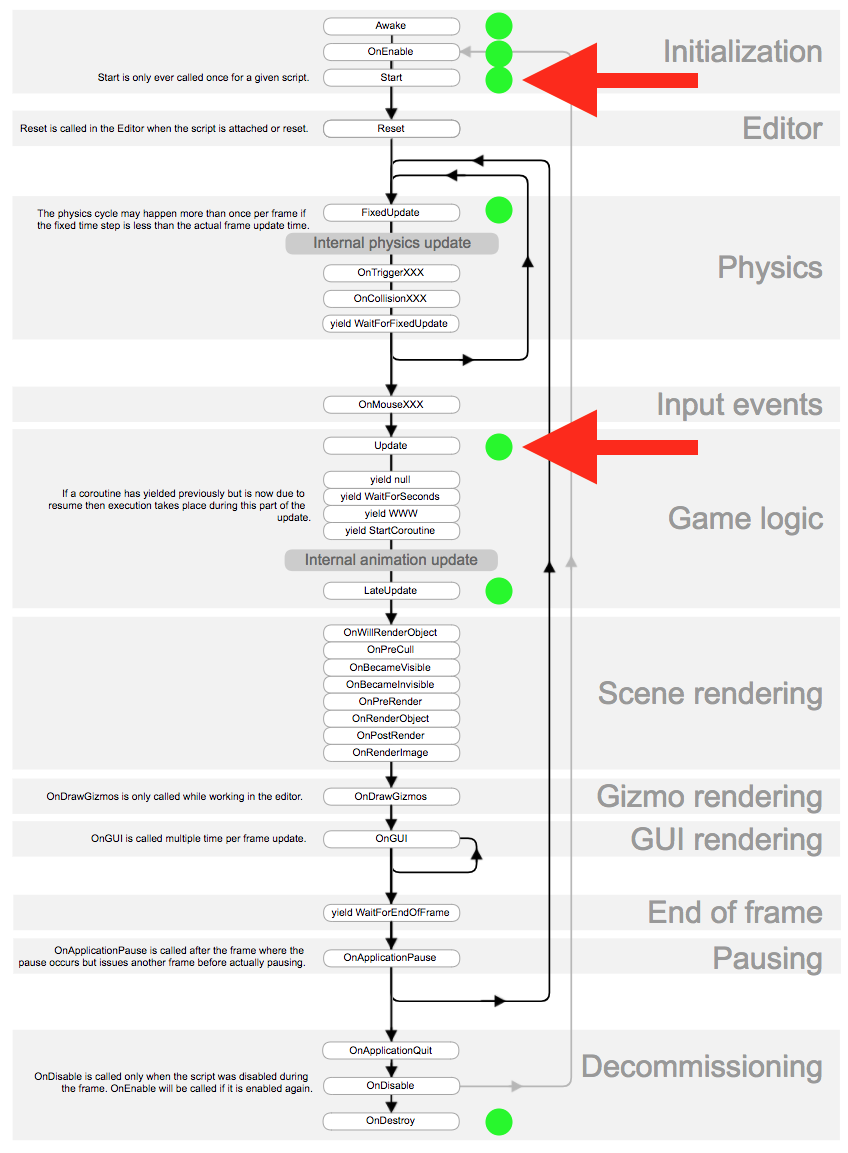
Lưu ý: Nếu bạn muốn xây dựng chuỗi động học trong bối cảnh ứng dụng công nghiệp, chẳng hạn như để mô phỏng cánh tay robot với hành vi vật lý thực tế, bạn nên sử dụng các khớp nối vật lý thay vì các khớp nối thông thường được mô tả dưới đây.

1. **Scripting**



Scripting là một thành phần thiết yếu trong tất cả các ứng dụng bạn tạo trong Unity. Hầu hết các ứng dụng đều cần tập lệnh để phản hồi thông tin đầu vào từ người chơi và sắp xếp các sự kiện trong trò chơi diễn ra khi cần thiết. Ngoài ra, tập lệnh có thể được sử dụng để tạo hiệu ứng đồ họa, điều khiển hành vi vật lý của các vật thể hoặc thậm chí triển khai hệ thống AI tùy chỉnh cho các nhân vật trong trò chơi.





Cơ Bản về Scripting trong Unity:

Lớp MonoBehaviour:

Các script trong Unity thường được kế thừa từ lớp MonoBehaviour. Lớp này cung cấp các phương thức như Start, Update, FixedUpdate và các sự kiện khác để quản lý hành vi của đối tượng trong trò chơi.

Các Phương Thức Quan Trọng:

Start(): Phương thức này được gọi khi đối tượng được khởi tạo trong Scene.

Update(): Được gọi mỗi frame và thường được sử dụng để cập nhật trạng thái của đối tượng.

FixedUpdate(): Được gọi mỗi frame cố định và thường được sử dụng cho các tính toán vật lý.

LateUpdate(): Được gọi sau tất cả các Update() và thường được sử dụng cho việc đảm bảo rằng tất cả các cập nhật đã hoàn thành trước khi thực hiện hành động tiếp theo.

Tương Tác với Đối Tượng và Components:

Scripting cho phép bạn tương tác với các đối tượng và thành phần khác trong trò chơi bằng cách sử dụng các phương thức và thuộc tính của chúng.

Bạn có thể lấy trực tiếp các tham chiếu đến các đối tượng và components bằng cách sử dụng các phương thức như GetComponent<T>().

Event System:

Unity cung cấp hệ thống sự kiện để xử lý các tương tác người dùng và các sự kiện khác trong trò chơi. Bạn có thể gán các phương thức từ script của bạn vào các sự kiện này để thực hiện hành động cụ thể khi chúng xảy ra.

Coroutines:

Coroutines là một tính năng mạnh mẽ trong Unity cho phép bạn thực hiện các hành động theo thời gian mà không cần chờ đợi. Điều này có thể hữu ích cho việc thực hiện các hoạt động như delay, animation sequence, và nhiều hơn nữa.

Debugging và Logging:

Unity cung cấp các công cụ debugging như log và break points để giúp bạn kiểm tra và gỡ lỗi script của bạn.

Sử dụng Debug.Log() để ghi lại thông điệp trong quá trình chạy và theo dõi các giá trị biến.

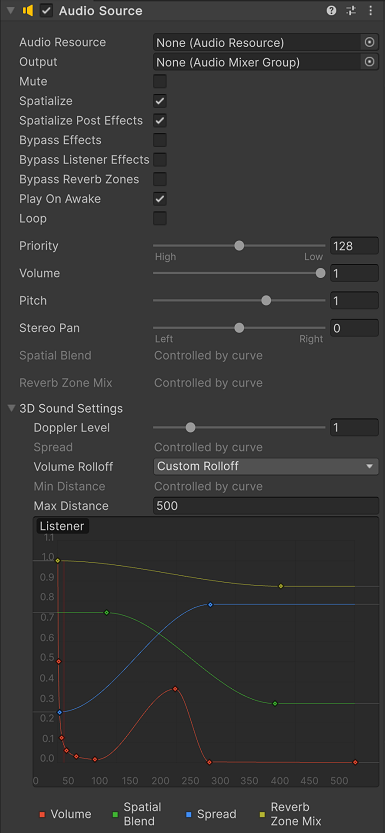
Optimization và Best Practices:

Đảm bảo rằng bạn tuân thủ các nguyên tắc tối ưu hóa và best practices khi viết script để tránh các vấn đề hiệu suất và bảo trì trong tương lai.

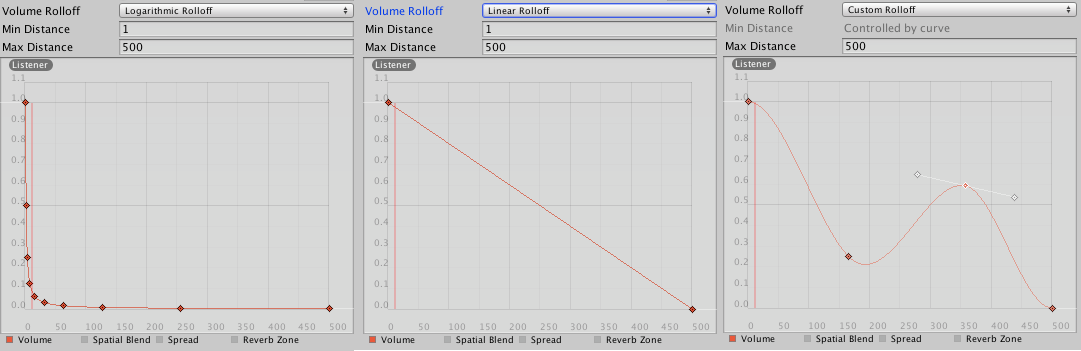
Scripting là một phần không thể thiếu của việc phát triển trò chơi và ứng dụng trong Unity. Bằng cách sử dụng các kỹ thuật và công cụ có sẵn, bạn có thể tạo ra các trò chơi phong phú và tương tác, với hành vi và tính năng mà bạn mong đợi.

1. **Audio**
2. AudioSource

Nguồn âm thanh phát lại Clip âm thanh trong cảnh. Đoạn clip có thể được phát cho người nghe âm thanh hoặc thông qua bộ trộn âm thanh. Nguồn âm thanh có thể phát bất kỳ loại Đoạn âm thanh nào và có thể được định cấu hình để phát chúng dưới dạng 2D, 3D hoặc dưới dạng hỗn hợp (SpatialBlend). Âm thanh có thể được trải ra giữa các loa (stereo to 7.1) (Spread) và biến đổi giữa 3D và 2D (SpatialBlend). Điều này có thể được kiểm soát theo khoảng cách bằng các đường cong dự phòng. Ngoài ra, nếu người nghe ở trong một hoặc nhiều Vùng Reverb, âm vang sẽ được áp dụng cho nguồn. Các bộ lọc riêng lẻ có thể được áp dụng cho từng nguồn âm thanh để có trải nghiệm âm thanh phong phú hơn nữa. Xem Hiệu ứng âm thanh để biết thêm chi tiết.



* Audio Resource: Tham chiếu đến tệp clip âm thanh sẽ được phát.
* Output: Theo mặc định, clip được xuất trực tiếp cho trình nghe âm thanh trong cảnh. Thay vào đó, sử dụng thuộc tính này để xuất clip vào bộ trộn âm thanh.
* Mute: Nếu bật âm thanh sẽ phát nhưng bị tắt tiếng
* Spatialize: Cho phép hoặc vô hiệu hóa không gian tùy chỉnh cho nguồn âm thanh. Thuộc tính này chỉ khả dụng nếu bạn đã cài đặt SDK không gian âm thanh và chọn nó trong cài đặt âm thanh toàn cầu của dự án.
* Spatialize Post Effect: Xác định xem bộ điều chỉnh không gian tùy chỉnh được áp dụng trước hay sau khi các hiệu ứng khác được gắn vào nguồn âm thanh. Cho phép thuộc tính này áp dụng bộ không gian tùy chỉnh sau các hiệu ứng khác được gắn vào nguồn âm thanh. Thuộc tính này chỉ khả dụng nếu bạn đã kích hoạt thuộc tính không gian hóa cho nguồn âm thanh.
* Bypass Effects: Điều này là để nhanh chóng các hiệu ứng bộ lọc 'By-pass' được áp dụng cho nguồn âm thanh. Một cách dễ dàng để bật/tắt tất cả các hiệu ứng.
* Bypass Listener Effects: Điều này là để nhanh chóng bật/tắt tất cả các hiệu ứng người nghe.
* **Bypass Reverb Zones: Điều này là nhanh chóng bật/tắt các vùng hồi âm**
* **Play On Awake: Nếu được bật, âm thanh sẽ bắt đầu phát vào khoảnh khắc cảnh phát ra. Nếu bị vô hiệu hóa, bạn cần bắt đầu bằng cách sử dụng lệnh play () từ tập lệnh.**
* **Loop: Kích hoạt tính năng này để tạo vòng lặp Audio Clip khi nó kết thúc**
* **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 nó thỉnh thoảng hoán đổi.**
* **Volume: Âm thanh ở khoảng cách của một đơn vị thế giới (một mét) từ trình nghe âm thanh.**
* **Pitch: Số lượng thay đổi trong cao độ do sự chậm lại/tăng tốc của clip âm thanh. Giá trị 1 là tốc độ phát lại bình thường.**
* **Stereo Pan: Đặt vị trí trong trường âm thanh nổi của âm thanh 2D.**
* **Spatial Blend: Đặt mức độ ảnh hưởng của công cụ 3D đến nguồn âm thanh.**
* **Reverb Zone Mix: Đặt lượng tín hiệu đầu ra được định tuyến đến vùng hồi âm. Lượng này là tuyến tính trong phạm vi (0 - 1), nhưng cho phép khuếch đại 10 dB trong phạm vi (1 - 1.1), điều này có thể hữu ích để đạt được hiệu ứng của âm thanh trường gần và âm thanh xa.**
* **3D Sound Settings: Các 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).**
* **Spread: Đặt góc lan truyền thành âm thanh nổi 3D hoặc âm thanh đa kênh trong không gian loa.**
* **Min Distance: Trong MinDistance, âm thanh sẽ ở mức to nhất có thể. Bên ngoài MinDistance nó sẽ bắt đầu giảm dần. Tăng MinDistance của âm thanh để làm cho âm thanh 'to hơn' trong thế giới 3d và giảm nó để làm cho âm thanh 'im lặng hơn' trong thế giới 3d.**
* **Max Distance: Khoảng cách mà âm thanh ngừng suy giảm. Ngoài thời điểm này, nó sẽ giữ nguyên âm lượng ở mức Khoảng cách tối đa tính từ người nghe và sẽ không giảm nữa.**
* **Rolloff Mode: Âm thanh biến mất nhanh như thế nào. Giá trị càng cao thì Listener càng phải ở gần hơn trước khi nghe được âm thanh. (Điều này được xác định bởi đồ thị).**
* **Logarithmic Rolloff: Âm thanh to khi bạn ở gần nguồn âm thanh, nhưng khi bạn rời xa vật thể thì nó giảm nhanh đáng kể.**
* **Linear Rolloff: Càng đi xa nguồn âm thanh, bạn càng ít nghe thấy nó.**
* **Custom Rolloff: Âm thanh từ nguồn âm thanh hoạt động tùy theo cách bạn thiết lập biểu đồ cuộn.**



1. AudioListener

AudioListener hoạt động như một thiết bị giống như micrô. Nó nhận đầu vào từ bất kỳ Nguồn âm thanh cụ thể nào trong cảnh và phát âm thanh qua loa máy tính. Đối với hầu hết các ứng dụng, việc gắn thiết bị nghe vào Camera chính là điều hợp lý nhất. Nếu người nghe âm thanh nằm trong ranh giới của Vùng vang âm sẽ được áp dụng cho tất cả âm thanh nghe được trong cảnh. Hơn nữa, Hiệu ứng âm thanh có thể được áp dụng cho người nghe và nó sẽ được áp dụng cho tất cả âm thanh nghe được trong cảnh.

https://docs.unity3d.com/uploads/Main/audio_listener_inspector.png

Trình nghe âm thanh 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.

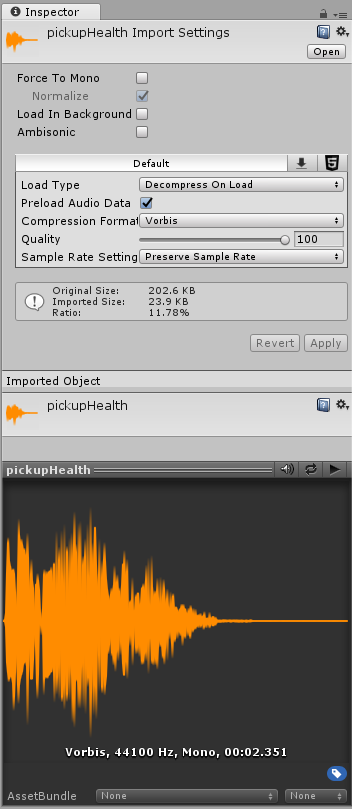
Trình nghe âm thanh hoạt động cùng với Nguồn âm thanh, cho phép bạn tạo trải nghiệm âm thanh cho trò chơi của mình. Khi Trình nghe âm thanh được gắn vào GameObject trong cảnh của bạn, mọi Nguồn đủ gần với Trình nghe sẽ được chọn và xuất ra loa của máy tính. Mỗi cảnh chỉ có thể có 1 Audio Listener để hoạt động bình thường.

Nếu Nguồn là 3D (xem cài đặt nhập trong Đoạn âm thanh), Trình nghe sẽ mô phỏng vị trí, vận tốc và hướng của âm thanh trong thế giới 3D (Bạn có thể điều chỉnh độ suy giảm và hành vi 3D/2D một cách chi tiết trong Nguồn âm thanh). 2D sẽ bỏ qua mọi xử lý 3D. Ví dụ: nếu nhân vật của bạn đi từ đường phố vào câu lạc bộ đêm, âm nhạc của câu lạc bộ đêm có thể phải là 2D, trong khi giọng nói của từng nhân vật trong câu lạc bộ phải là giọng đơn âm và vị trí thực tế của họ sẽ do Unity xử lý.

Bạn nên gắn Trình nghe âm thanh vào Camera chính hoặc GameObject đại diện cho trình phát. Hãy thử cả hai để tìm ra thứ phù hợp nhất với trò chơi của bạn. gợi ý

1. AudioClip

Các đoạn âm thanh chứa dữ liệu âm thanh được Nguồn âm thanh sử dụng. Unity hỗ trợ nội dung 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. Unity cũng hỗ trợ nhập mô-đun theo dõi ở định dạng .xm, .mod, .it và .s3m. Nội dung mô-đun theo dõi hoạt động giống như bất kỳ nội dung âm thanh nào khác trong Unity mặc dù không có bản xem trước dạng sóng trong trình kiểm tra nhập nội dung.



**Options**

**Force To Mono**

Khi tùy chọn này được bật, âm thanh đa kênh sẽ được trộn thành một bản nhạc 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".

**Load In Background**

Khi tùy chọn này được bật, quá trình tải clip sẽ diễn ra ở thời điểm trễ trên một luồng riêng biệt mà không chặn luồng chính.

**Ambisonic**

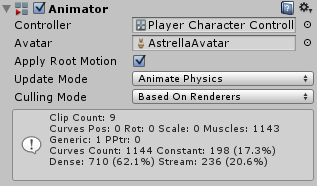
Nguồn âm thanh Ambisonic lưu trữ âm thanh ở định dạng đại diện cho trường âm thanh có thể xoay dựa trên hướng của người nghe. Nó hữu ích cho video 360 độ và ứng dụng XR. Bật tùy chọn này nếu tệp âm thanh của bạn chứa âm thanh được mã hóa Ambisonic.

**Properties:**

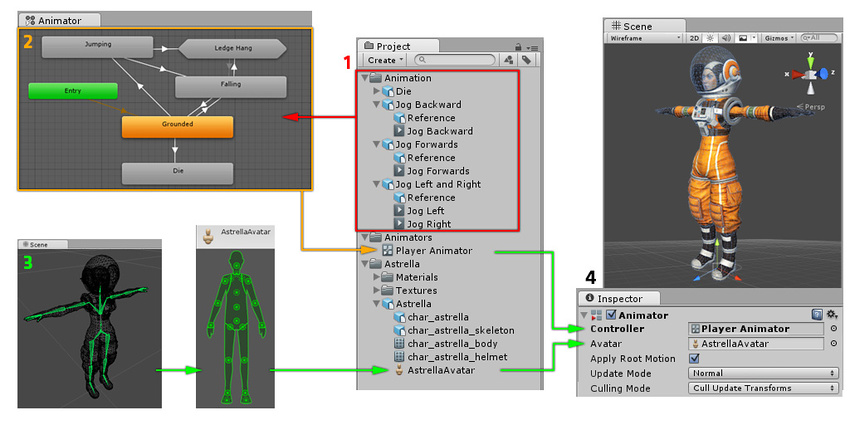
| **Property:** | | **Function:** |
| --- | --- | --- |
| **Load Type** | | Phương thức Unity sử dụng để tải nội dung âm thanh trong thời gian chạy. |
|  | **Decompress On Load** | Các tập tin âm thanh được giải nén ngay khi chúng được tải. Sử dụng tùy chọn này cho các âm thanh nén nhỏ hơn để tránh tốn kém hiệu suất khi giải nén nhanh chóng. Xin lưu ý rằng việc giải nén âm thanh được mã hóa Vorbis khi tải sẽ sử dụng bộ nhớ nhiều hơn khoảng mười lần so với việc giữ chúng ở dạng nén (đối với mã hóa ADPCM là khoảng 3,5 lần), vì vậy đừng sử dụng tùy chọn này cho các tệp lớn |
|  | **Compressed In Memory** | Giữ âm thanh được nén trong bộ nhớ và giải nén trong khi phát. Tùy chọn này có chi phí hoạt động hơi cao, đặc biệt đối với các tệp nén Ogg/Vorbis. Chỉ sử dụng nó cho các tệp tiêu thụ quá nhiều bộ nhớ cho Giải nén khi tải. Quá trình giải nén diễn ra trên luồng của bộ trộn, có thể được theo dõi trong phần CPU DSP trong khung Âm thanh của cửa sổ Profiler. |
|  | **Streaming** | Giải mã âm thanh liên tục. Phương pháp này sử dụng lượng bộ nhớ tối thiểu để đệm dữ liệu nén được đọc dần dần từ đĩa và được giải mã một cách tự nhiên. Quá trình giải nén diễn ra trên một luồng phát trực tuyến riêng biệt mà việc sử dụng CPU có thể được theo dõi trong phần CPU truyền phát trong khung Âm thanh của cửa sổ trình lược tả. Lưu ý: Các clip phát trực tuyến có tổng dung lượng khoảng 200KB, ngay cả khi không có dữ liệu âm thanh nào được tải. |
| **Compression Format** | | Định dạng cụ thể sẽ được sử dụng cho âm thanh khi chạy. Lưu ý rằng các tùy chọn có sẵn tùy thuộc vào mục tiêu xây dựng hiện được chọn. |
|  | **PCM** | Tùy chọn này cung cấp chất lượng cao hơn nhưng phải trả giá bằng kích thước tệp lớn hơn và phù hợp nhất cho các hiệu ứng âm thanh ngắn. |
|  | **ADPCM** | Định dạng này hữu ích cho những âm thanh có khá nhiều tiếng ồn và cần phát với số lượng lớn, chẳng hạn như tiếng bước chân, tác động, vũ khí. Tỷ lệ nén nhỏ hơn 3,5 lần so với PCM nhưng mức sử dụng CPU thấp hơn nhiều so với định dạng MP3/Vorbis, khiến nó trở thành lựa chọn ưu tiên cho các loại âm thanh nói trên. |
|  | **Vorbis/MP3** | Quá trình nén tạo ra các tệp nhỏ hơn nhưng có chất lượng thấp hơn một chút so với âm thanh PCM. Mức độ nén có thể được cấu hình thông qua thanh trượt Chất lượng. Định dạng này phù hợp nhất cho âm nhạc và hiệu ứng âm thanh có độ dài trung bình. |
| **Sample Rate Setting** | | Các định dạng nén PCM và ADPCM cho phép giảm tốc độ mẫu được tối ưu hóa tự động hoặc thủ công. |
|  | **Preserve Sample Rate** | Cài đặt này giữ cho tốc độ mẫu không bị sửa đổi (mặc định). |
|  | **Optimize Sample Rate** | Cài đặt này tự động tối ưu hóa tốc độ mẫu theo nội dung tần số cao nhất được phân tích. |
|  | **Override Sample Rate** | Cài đặt này cho phép ghi đè tốc độ mẫu theo cách thủ công, do đó, cài đặt này có thể được sử dụng để loại bỏ nội dung tần số một cách hiệu quả. |
| **Force To Mono** | | Khi được bật, clip âm thanh sẽ được trộn xuống thành âm thanh một kênh. Sau khi trộn xuống, tín hiệu được chuẩn hóa đỉnh vì quá trình trộn xuống thường tạo ra các tín hiệu yên tĩnh hơn tín hiệu ban đầu. Tín hiệu chuẩn hóa đỉnh cung cấp khoảng trống cho những điều chỉnh sau này thông qua thuộc tính âm lượng của AudioSource. |
| **Load In Background** | | Khi được bật, clip âm thanh sẽ tải ở chế độ nền mà không gây ra tình trạng dừng trên luồng chính. Tính năng này bị tắt theo mặc định để đảm bảo hoạt động Unity tiêu chuẩn trong đó tất cả các AudioClip hoàn tất tải ngay khi cảnh bắt đầu phát. Các yêu cầu phát trên AudioClip vẫn đang tải ở chế độ nền sẽ bị hoãn lại cho đến khi tải xong clip. Bạn có thể truy vấn trạng thái tải thông qua thuộc tính AudioClip.loadState. |
| **Preload Audio Data** | | Khi được bật, clip âm thanh sẽ được tải trước sau khi cảnh được tải. Tính năng này được bật theo mặc định để phản ánh hành vi Unity tiêu chuẩn trong đó tất cả AudioClip hoàn tất tải ngay khi cảnh bắt đầu phát. Nếu cờ này không được đặt thì dữ liệu âm thanh sẽ được tải trên AudioSource.Play AudioSource.PlayOneShot đầu tiên hoặc có thể được tải thông qua AudioClip.LoadAudioData và được tải lại thông qua AudioClip.UnloadAudioData. |
| **Quality** | | Xác định mức độ nén sẽ được áp dụng cho một clip đã nén. Không áp dụng cho các định dạng PCM/ADPCM/HEVAG. Bạn có thể xem số liệu thống kê về kích thước tệp trong trình kiểm tra. Một cách tiếp cận tốt để điều chỉnh giá trị này là kéo thanh trượt đến vị trí sao cho quá trình phát lại ở mức "đủ tốt" trong khi vẫn giữ tệp đủ nhỏ cho yêu cầu phân phối của bạn. Lưu ý rằng kích thước ban đầu liên quan đến tệp gốc, vì vậy nếu đây là tệp MP3 và Định dạng nén được đặt thành PCM (không nén) thì Tỷ lệ kết quả sẽ lớn hơn 100% vì tệp hiện được lưu trữ không nén và chiếm nhiều dung lượng hơn hơn nguồn MP3 mà nó đến từ đó. |

1. **Animation**
2. Animator

Sử dụng thành phần Animator để gán hoạt ảnh cho GameObject trong Scene của bạn. Thành phần Animator yêu cầu tham chiếu đến Bộ điều khiển hoạt hình để xác định clip hoạt ảnh nào sẽ sử dụng cũng như kiểm soát thời điểm cũng như cách trộn và chuyển đổi giữa chúng. Nếu GameObject là một nhân vật hình người có định nghĩa Avatar thì Avatar cũng phải được chỉ định trong thành phần này, như được thấy ở đây:



Sơ đồ này cho thấy cách kết hợp các nội dung khác nhau (Đoạn hoạt hình, Bộ điều khiển hoạt hình và Hình đại diện) trong Thành phần hoạt hình trên GameObject:

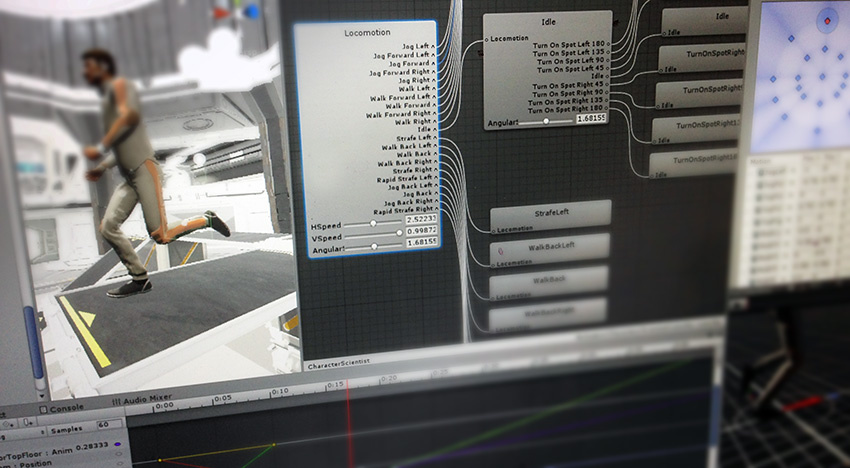


| **Property:** | | **Function:** |
| --- | --- | --- |
| **Controller** | | Bộ điều khiển hoạt hình gắn liền với nhân vật này. |
| **Avatar** | | Avatar của nhân vật này. (Nếu Animator đang được sử dụng để tạo hoạt ảnh cho nhân vật hình người) |
| **Apply Root Motion** | | Chọn kiểm soát vị trí và xoay của nhân vật từ chính hoạt ảnh hay từ tập lệnh. |
| **Update Mode** | | This allows you to select when the Animator updates, and which timescale it should use. |
|  | **Normal** | Điều này cho phép bạn chọn thời điểm Animator cập nhật và nên sử dụng khoảng thời gian nào. |
|  | **Animate Physics** | Trình hoạt ảnh được cập nhật không đồng bộ với lệnh gọi Cập nhật và tốc độ của trình hoạt hình phù hợp với khoảng thời gian hiện tại. Nếu khoảng thời gian bị chậm lại, hoạt ảnh sẽ chậm lại để phù hợp. |
|  | **Unscaled Time** | Trình hoạt ảnh được cập nhật không đồng bộ với lệnh gọi Cập nhật, nhưng tốc độ của trình hoạt ảnh bỏ qua khoảng thời gian hiện tại và hoạt hình ở tốc độ 100% bất kể. Điều này rất hữu ích để tạo hoạt ảnh cho hệ thống GUI ở tốc độ bình thường trong khi sử dụng khoảng thời gian đã sửa đổi cho các hiệu ứng đặc biệt hoặc để tạm dừng trò chơi. |
| **Culling Mode** | | Chế độ loại bỏ bạn có thể chọn cho hình ảnh động |
|  | **Always Animate** | Luôn tạo hoạt ảnh, không loại bỏ ngay cả khi ở ngoài màn hình. |
|  | **Cull Update Transforms** | Nhắm mục tiêu lại, IK và ghi Biến đổi bị tắt khi không hiển thị trình kết xuất. |
|  | **Cull Completely** | Hoạt ảnh bị tắt hoàn toàn khi trình kết xuất không hiển thị. |

Hộp thông tin ở cuối thành phần Animator cung cấp cho bạn thông tin chi tiết về dữ liệu đang được sử dụng trong tất cả các clip được Bộ điều khiển hoạt hình sử dụng. Đoạn phim hoạt hình chứa dữ liệu ở dạng "đường cong", thể hiện cách giá trị thay đổi theo thời gian. Những đường cong này có thể mô tả vị trí hoặc góc quay của một vật thể, độ uốn của cơ trong hệ thống hoạt hình hình người hoặc các giá trị hoạt hình khác trong clip chẳng hạn như màu vật liệu thay đổi. Bảng này giải thích ý nghĩa của từng mục dữ liệu:

| ***Label*** | ***Description*** |
| --- | --- |
| **Clip Count** | Tổng số clip hoạt hình được bộ điều khiển hoạt hình sử dụng được gán cho Trình hoạt hình này. |
| **Curves (Pos, Rot & Scale)** | Tổng số đường cong mà Unity sử dụng để tạo hiệu ứng cho vị trí, góc xoay hoặc tỷ lệ của GameObjects. Chúng dành cho các GameObject hoạt hình không phải là một phần của giàn hình người tiêu chuẩn. Khi tạo hoạt ảnh cho hình đại diện hình người, những đường cong này sẽ hiển thị số lượng xương không phải cơ bổ sung như đuôi, vải chảy hoặc mặt dây chuyền lủng lẳng. Nếu bạn có hoạt ảnh hình người và nhận thấy các đường cong hoạt ảnh không phải cơ bắp không mong muốn thì bạn có thể có các đường cong hoạt ảnh không cần thiết trong tệp hoạt ảnh của mình. |
| **Muscles** | Số đường cong hoạt hình cơ được Nhà hoạt hình này sử dụng cho hoạt hình hình người. Đây là những đường cong được sử dụng để tạo hoạt ảnh cho các cơ hình đại diện hình người tiêu chuẩn. Ngoài các chuyển động cơ tiêu chuẩn cho tất cả các xương hình người trong hình đại diện tiêu chuẩn của Unity, điều này còn bao gồm hai "đường cong cơ" lưu trữ vị trí chuyển động gốc và hoạt ảnh xoay. |
| **Generic** | Số lượng đường cong số (float) được trình hoạt hình sử dụng để tạo hoạt ảnh cho các thuộc tính khác như màu vật liệu. |
| **PPtr** | Tổng số đường cong hoạt hình sprite (được sử dụng bởi hệ thống 2d của Unity) |
| **Curves Count** | Tổng số đường cong hoạt hình kết hợp |
| **Constant** | Số lượng đường cong hoạt ảnh được tối ưu hóa dưới dạng giá trị không đổi (không thay đổi). Unity tự động chọn tùy chọn này nếu tệp hoạt ảnh của bạn chứa các đường cong có giá trị không thay đổi. |
| **Dense** | Số lượng đường cong hoạt ảnh được tối ưu hóa bằng phương pháp lưu trữ dữ liệu "dày đặc" (các giá trị rời rạc được nội suy giữa tuyến tính). Phương pháp này sử dụng ít bộ nhớ hơn đáng kể so với phương pháp "stream". |
| **Stream** | Số lượng đường cong hoạt ảnh sử dụng phương pháp lưu trữ dữ liệu "luồng" (các giá trị có dữ liệu thời gian và tiếp tuyến cho phép nội suy đường cong). Dữ liệu này chiếm nhiều bộ nhớ hơn đáng kể so với phương pháp "dày đặc". |

1. Animation



Các tính năng Hoạt hình của Unity bao gồm hoạt ảnh có thể nhắm mục tiêu lại, kiểm soát hoàn toàn trọng lượng hoạt ảnh trong thời gian chạy, gọi sự kiện từ trong quá trình phát lại hoạt ảnh, phân cấp và chuyển tiếp máy trạng thái phức tạp, hòa trộn hình dạng cho hoạt ảnh khuôn mặt, v.v. Đọc phần này để tìm hiểu cách nhập và làm việc với hoạt ảnh đã nhập cũng như cách tạo hoạt ảnh cho các đối tượng, màu sắc và bất kỳ tham số nào khác trong chính Unity.

1. AnimationClip

AnimationClip trong Unity là một tập hợp các khung hình (frames) hoặc quỹ đạo (trajectories) được sắp xếp theo thời gian, được sử dụng để tạo ra hiệu ứng hoạt hình cho các đối tượng trong trò chơi. Bạn có thể sử dụng AnimationClip để điều khiển các thuộc tính như vị trí, quay, tỉ lệ và màu sắc của các thành phần trong Unity.

Dưới đây là một số điều bạn có thể làm với AnimationClip trong Unity:

Tạo AnimationClip:

Trong cửa sổ Project, bạn có thể tạo một AnimationClip bằng cách nhấp chuột phải và chọn Create > AnimationClip.

Ghi lại Animation:

Unity cung cấp các công cụ cho phép bạn ghi lại các hành động của đối tượng trong trò chơi để tạo ra AnimationClip. Bạn có thể sử dụng các công cụ như Animator hoặc Animation Window để ghi lại các hành động này.

Sửa đổi AnimationClip:

Bạn có thể chỉnh sửa AnimationClip bằng cách thêm, xóa hoặc di chuyển các khung hình. Bạn cũng có thể chỉnh sửa các thuộc tính của mỗi khung hình để thay đổi vị trí, quay, tỉ lệ hoặc màu sắc của đối tượng.

Sử dụng AnimationClip trong Animator Controller:

AnimationClip thường được sử dụng trong Animator Controller để điều khiển hoạt hình của các đối tượng trong trò chơi. Bằng cách thêm các trạng thái và chuyển đổi trong Animator Controller, bạn có thể chuyển đổi giữa các AnimationClip để tạo ra các hoạt hình phức tạp.

Chạy AnimationClip bằng mã:

Bạn cũng có thể chạy AnimationClip bằng mã thông qua script trong Unity. Bằng cách sử dụng các phương thức như Play, Stop và CrossFade, bạn có thể điều khiển việc chạy các AnimationClip trong game của mình.

AnimationClip là một công cụ mạnh mẽ trong Unity, cho phép bạn tạo ra các hoạt hình phong phú và sống động cho trò chơi của mình.

1. AvatarController

Trong Unity, AvatarController không phải là một thành phần hoặc lớp được tích hợp sẵn. Tuy nhiên, có thể bạn đang nói đến AnimatorController, một thành phần quan trọng để điều khiển hoạt hình của avatar trong trò chơi của bạn.

Dưới đây là một giải thích về AnimatorController:

Animator Controller:

Định nghĩa:

AnimatorController là một loại tệp hoặc thành phần trong Unity được sử dụng để điều khiển hoạt hình của đối tượng trong trò chơi.

Nó chứa các trạng thái (states), chuyển đổi (transitions), và các cài đặt khác để quản lý hoạt hình và trạng thái của avatar hoặc đối tượng.

Sử dụng:

Để tạo một Animator Controller, bạn có thể nhấp chuột phải trong cửa sổ Project, sau đó chọn Create > Animator Controller.

Bạn có thể gán Animator Controller cho một đối tượng trong Scene hoặc sử dụng nó thông qua script để điều khiển hoạt hình.

Các thành phần:

States (Trạng thái): Đại diện cho các trạng thái khác nhau của hoạt hình, như chạy, nhảy, đứng yên, vv.

Transitions (Chuyển đổi): Xác định cách chuyển từ một trạng thái sang một trạng thái khác.

Parameters (Tham số): Là các biến được sử dụng để kiểm soát chuyển đổi giữa các trạng thái.

Layers (Lớp): Cho phép bạn phân tách và quản lý các hoạt hình đa lớp.

Sử dụng trong Script:

Bạn có thể sử dụng script để điều khiển Animator Controller thông qua các phương thức như Play, SetTrigger, SetBool, vv.

Animator Controller là một phần quan trọng trong việc tạo ra hoạt hình chất lượng và sống động trong trò chơi Unity của bạn, cho phép bạn điều khiển và kết hợp các hoạt hình một cách linh hoạt và hiệu quả.

1. AvatarMaskt

Trong Unity, AvatarMask là một thành phần được sử dụng trong hệ thống Animation để điều chỉnh quá trình hiển thị hoạt hình của một avatar hoặc đối tượng. Dưới đây là một số thông tin về AvatarMask:

AvatarMask:

Định nghĩa:

AvatarMask là một thành phần của Animation Rigging trong Unity, cho phép bạn xác định các phần của avatar hoặc đối tượng sẽ được ảnh hưởng bởi hoạt hình.

Sử dụng:

AvatarMask được sử dụng để giới hạn hoạt hình chỉ đối với một số phần cụ thể của avatar hoặc đối tượng.

Điều này cho phép bạn tạo ra các hiệu ứng hoạt hình phức tạp bằng cách chỉ định rõ ràng những phần cần tham gia vào hoạt hình.

Cách sử dụng:

Để sử dụng AvatarMask, bạn có thể tạo hoặc chỉnh sửa nó trong cửa sổ Avatar Mask Editor của Unity.

Trong Editor này, bạn có thể xác định những phần của avatar hoặc đối tượng sẽ được ảnh hưởng bởi hoạt hình bằng cách chọn hoặc bỏ chọn các bones hoặc transform.

Ứng dụng:

AvatarMask thường được sử dụng khi bạn muốn tạo ra các hoạt hình chỉ ảnh hưởng đến một phần cụ thể của avatar, chẳng hạn như việc điều khiển riêng lẻ các phần của cơ thể (ví dụ: tay, chân) hoặc các mảnh ghép của avatar.

Tính linh hoạt:

Với AvatarMask, bạn có thể tạo ra các hoạt hình phức tạp và linh hoạt, cho phép bạn tinh chỉnh và điều chỉnh hoạt hình của avatar một cách chính xác và hiệu quả.

AvatarMask là một công cụ mạnh mẽ trong hệ thống Animation của Unity, cho phép bạn tạo ra các hiệu ứng hoạt hình chất lượng và sống động trong trò chơi của bạn bằng cách xác định rõ ràng những phần cụ thể mà hoạt hình sẽ ảnh hưởng đến.

1. **Navigation**

Trong Unity, Navigation là một thành phần được sử dụng để xác định cách mà các đối tượng có thể tương tác với nhau trong không gian 3D hoặc 2D. Navigation thường được sử dụng trong các trò chơi và ứng dụng nơi mà các đối tượng cần di chuyển và tương tác với nhau trong không gian game.

Dưới đây là một số thông tin về Navigation:

Navigation:

Định nghĩa:

Navigation là một thành phần của Unity được sử dụng để định rõ cách mà các đối tượng di chuyển và tương tác với nhau trong không gian game.

Nó thường được sử dụng để xác định con đường di chuyển của các đối tượng, các rào cản, và cách tương tác của chúng.

Sử dụng:

Navigation thường được sử dụng trong các trò chơi nơi mà các đối tượng cần di chuyển trong không gian game.

Nó cho phép bạn xác định các điểm đến, các rào cản, các khu vực cấm di chuyển và các tham số khác để điều chỉnh cách mà các đối tượng di chuyển trong trò chơi của bạn.

Các loại Navigation:

Navigation 3D: Dành cho các trò chơi 3D, cho phép các đối tượng di chuyển trong không gian 3 chiều.

Navigation 2D: Dành cho các trò chơi 2D, cho phép các đối tượng di chuyển trong không gian 2 chiều.

NavMesh: Sử dụng một mạng lưới để đại diện cho không gian di chuyển của các đối tượng.

Tính năng:

Pathfinding (Tìm đường): Navigation cung cấp các thuật toán pathfinding để tính toán các con đường di chuyển hiệu quả cho các đối tượng trong trò chơi.

Collision Avoidance (Tránh va chạm): Cho phép bạn xác định cách mà các đối tượng tránh va chạm với nhau khi di chuyển trong không gian game.

Sử dụng trong Unity:

Để sử dụng Navigation, bạn cần thêm một thành phần NavMesh hoặc NavMeshAgent vào các đối tượng trong trò chơi của bạn và cấu hình chúng để phù hợp với nhu cầu của bạn.

Navigation là một công cụ mạnh mẽ trong Unity, cho phép bạn tạo ra các trò chơi với hệ thống di chuyển và tương tác phức tạp giữa các đối tượng.

1. **UI**

Trong Unity, UI (User Interface) là thành phần quan trọng để tạo ra giao diện người dùng cho trò chơi và ứng dụng của bạn. Giao diện người dùng giúp người chơi tương tác với trò chơi của bạn, bao gồm các nút, hộp văn bản, thanh trượt, hình ảnh và các thành phần khác.

Dưới đây là một số thông tin về UI trong Unity:

UI (User Interface):

Canvas:

Canvas là thành phần cơ bản của giao diện người dùng trong Unity.

Nó đại diện cho bề mặt mà tất cả các thành phần UI được vẽ lên.

Canvas có thể được cấu hình để sử dụng đa phần tử (Multiple) hoặc chỉ một phần tử (Single), tùy thuộc vào yêu cầu của bạn.

UI Elements:

Unity cung cấp một loạt các thành phần UI như Text, Image, Button, Slider, Scrollbar, Toggle, Input Field và nhiều hơn nữa.

Các thành phần này được sử dụng để tạo ra các phần của giao diện người dùng, cho phép người chơi tương tác với trò chơi của bạn.

Layouts và Groups:

Unity cung cấp các thành phần Layouts như Vertical Layout Group, Horizontal Layout Group và Grid Layout Group để tự động sắp xếp các thành phần UI.

Bằng cách sử dụng Layouts và Groups, bạn có thể tạo ra giao diện người dùng linh hoạt và dễ dàng điều chỉnh.

Animation và Transitions:

Bạn có thể áp dụng animation và transitions cho các thành phần UI để tạo ra hiệu ứng động và chuyển đổi mượt mà trong giao diện người dùng của bạn.

Điều này giúp tăng tính tương tác và sự hấp dẫn của trò chơi hoặc ứng dụng của bạn.

Scripting:

Bạn có thể điều khiển và tương tác với các thành phần UI thông qua kịch bản (scripting) trong Unity.

Sử dụng các phương thức và sự kiện UI như onClick, onValueChanged để thực hiện các hành động khi người chơi tương tác với các thành phần UI.

Giao diện người dùng (UI) là một phần quan trọng của trò chơi và ứng dụng của bạn trong Unity, cho phép bạn tạo ra trải nghiệm tương tác tốt và dễ sử dụng cho người chơi và người dùng của bạn.

1. **Assets**

Cửa hàng Asset trong Unity là một nơi bạn có thể tìm thấy và mua các tài nguyên, gói và công cụ để phát triển trò chơi và ứng dụng của mình. Unity Asset Store cung cấp một loạt các tài nguyên, bao gồm:

3D Models và Characters: Các mô hình 3D của các đối tượng, nhân vật, môi trường và các phần khác để tạo ra thế giới của trò chơi.

Textures và Materials: Các hình ảnh, texture và vật liệu để trang trí và làm đẹp cho môi trường và các đối tượng trong trò chơi.

Audio và Sound Effects: Âm thanh, nhạc nền và hiệu ứng âm thanh để tạo ra trải nghiệm âm thanh phong phú cho trò chơi của bạn.

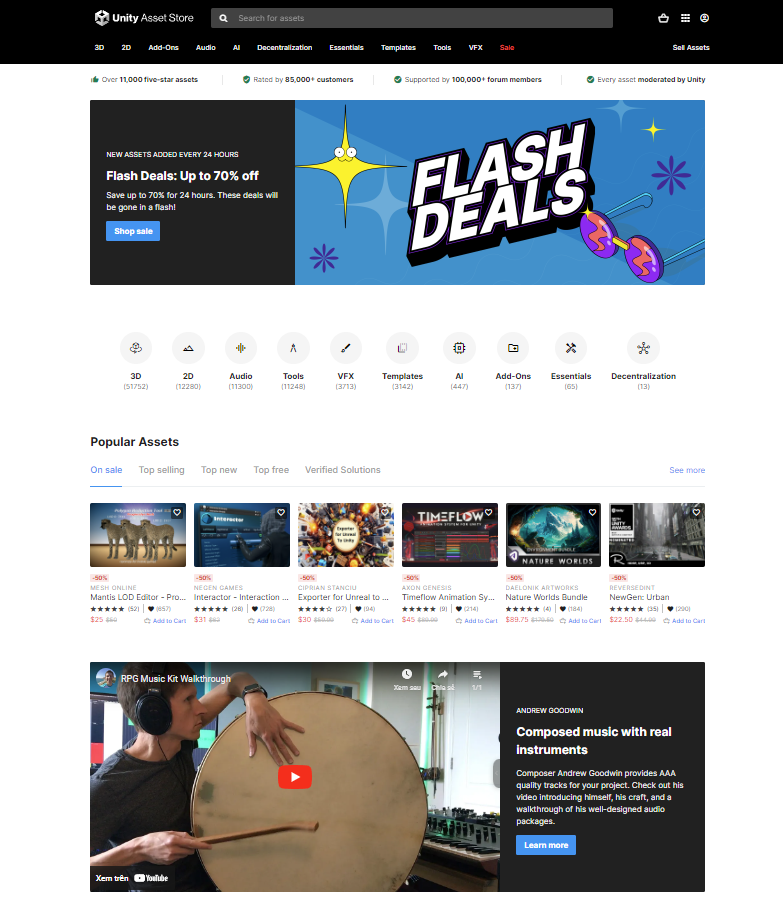
Scripts và Plugins: Các tập lệnh và plugin để mở rộng chức năng của Unity và giúp bạn thực hiện các nhiệm vụ cụ thể trong trò chơi của mình.

Animations và Particle Effects: Các hoạt ảnh và hiệu ứng hạt để tạo ra các chuyển động và hiệu ứng đặc biệt cho trò chơi của bạn.

Tools và Utilities: Các công cụ và tiện ích giúp bạn tăng tốc quá trình phát triển trò chơi và tạo ra các tính năng phức tạp.

Templates và Project Examples: Các mẫu và ví dụ dự án để bạn có thể bắt đầu nhanh chóng với việc phát triển trò chơi của mình.

Demos và Projects: Các dự án và bản demo để bạn có thể học và tham khảo để xây dựng trò chơi của riêng mình.



Bạn có thể truy cập Unity Asset Store trực tiếp từ trình chỉnh sửa Unity bằng cách chọn Window > Asset Store trong thanh menu hoặc từ trang web chính thức của Unity Asset Store. Từ đó, bạn có thể tìm kiếm, duyệt và mua các tài nguyên và công cụ phù hợp với nhu cầu của bạn.