**Chương I. Công cụ và ngôn ngữ lập trình**

**1.**    **Tổng quan về Unity engine**

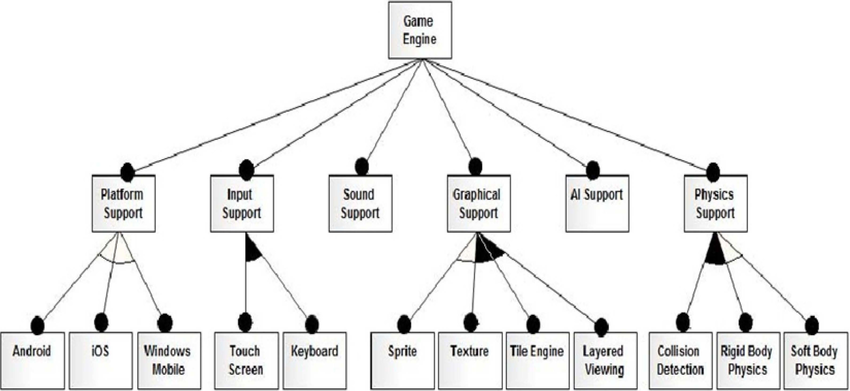
**1.1 Sơ lược về lịch sử - quá trình phát triển của Unity**

Ra mắt đầu tiên vào năm 2005 tại sự kiện Apple’s Worldwide Developer Conference bởi nhà sáng lập David Helgason. Trải qua nhiều năm phát triển, Unity đã có version 5.5 hoàn thiện hơn về nhiều mặt. Tháng 5-2012 theo cuộc khảo sát Game Developer Magazine được công nhận là Game engine tốt nhất cho mobile. Năm 2004 Unity thắng giải Best Engine tại UK’s annual Develop Industry Excellence. Phiên bản mới nhất hiện nay là 2021.1.24f1 với nhiều tính năng nổi trội

Game engine về cơ bản là phần mềm cung cấp nhiều chức năng thường được sử dụng để xây dựng một trò chơi điện tử. Hầu hết các công cụ trò chơi cho phép hiển thị hình dạng 2D hoặc 3D bằng cách nhận tệp được tạo trong phần mềm của bên thứ ba như Blender hoặc 3D Studio Max hoặc Maya, nếu bạn có đủ khả năng chi trả.

Game engine được xem là hệ thống cung cấp nguyên liệu xây dựng, trang thiết bị và các phương thức lắp ghép cơ bản vào một trò game. Kỹ sư làm game chỉ việc vẽ ra bản thiết kế, tự mình lắp ghép các nguyên liệu, trang trí hay sắp xếp chúng thành một ngôi nhà hoàn chỉnh.

Chức năng chính của Game engine là: cung cấp phương tiện dựng hình cho các hình ảnh 2D hoặc 3D, phương tiện vật lý, âm thanh, mã nguồn, hình ảnh động, trí tuệ nhân tạo, phân luồng, tạo dòng dữ liệu xử lý, quản lý bộ nhớ, dựng ảnh đồ thị và kết nối mạng. Nhờ có các engine mà công việc làm game trở thành ít tốn kém và đơn giản hơn. Một số engine nổi tiếng như Unity3D Engine, Cryengine, Unreal Engine, Source Engine, Id Tech Engine, IW Engine…



*Hình 1.Chức năng của Game engine*

Một số thống kê về về Unity:

* Tính đến quý 3 năm 2016 đã có 5 tỷ lượt download game và ứng dụng được phát triển bởi Unity.
* 2,4 tỷ thiết bị di động đã từng ít tải ít nhất 1 ứng dụng bởi Unity.
* Trong top 1000 game Mobile miễn phí thì số lượng game tạo ra bởi Unity chiếm tới 34%.

*(nguồn: thống kê từ internet)*

**1.2 Unity là gì?**



*Hình 2.Logo của Unity.*

Unity là một cross-platform game engine, tạm hiểu là công cụ phát triển game đa nền tảng, được phát triển bởi Unity Technologies. Game engine này được sử dụng để phát triển game trên PC, consoles, thiết bị di động (Android, IOS) và trên website.

Unity3D là phần mềm làm game trực tiếp theo thời gian thực mà không cần render, cho phép người design game có thể thiết kế Interface Graphic, Map hay Character,... từ một phần mềm thứ 2 (thường là các phần mềm thiết kế đồ họa chuyên nghiệp như 3DsMax, Blender, Maya, XSL, Cinema 4D, Cheetah3D, Modo,...) sau đó chỉ việc import nó vào trong Unity với định dạng của tập tin phù hợp. Là môi trường phát triển tích hợp, mạnh mẽ, hỗ trợ thao tác kéo thả, tùy biến giao diện nhanh chóng, trực quan.

 Nhờ sự nỗ lực theo thời gian của đội ngũ phát triển Unity mà ngày nay Unity có thể ứng dụng vào các lĩnh vực khác nhau không chỉ là Game mà còn vào các công nghệ thực tế ảo AR/VR.

Một số khái niệm liên quan đến Unity:

Trong unity có rất nhiều công cụ nhỏ khác giúp phục vụ cho các mục đích khác nhau trong quá trình lập trình viên code game. Chúng ta hãy cùng tìm hiểu qua một số khái niệm liên quan dưới đây:

1. **GameObject**

Đây là một đối tượng cụ thể trong game như là nhân vật hay là đồ vật nào đó (con người, nhà cửa, cây cối, xe cộ,...). Khi một tài nguyên được sử dụng trong một scene, khi đó chúng ta có thể coi tài nguyên này là một GameObject mới. Mỗi GameObject phái chứa ít nhất một thành phần, đó là thành phần Transform. Transform chứa các phép để biến đổi góc quay, tỷ lệ hay tịnh tiến của đối tượng.

1. **Component**

Đối với mỗi gameobject đều sẽ có nhiều thành phần khác nhau cấu tạo nên như là hình ảnh, các hành động, tính toán vật lý, thành phần để xử lý các va chạm, các mã điều khiển,... và mỗi thành phần này sẽ được gọi là component.

1. **Sprite**

đây là hình ảnh 2D của game và đó có thể sẽ là các hình ảnh đầy đủ hoặc cũng có thể chỉ là một bộ phận nhất định nào đó trong game.

1. **Animation**

tập hợp hình ảnh động dựa trên những biến đổi liên tục của các sprite khác trong game.

1. **Keyframe**

trạng thái của hình ảnh động và được tạo nên từ 1 hình ảnh 2D hoặc là nhiều hình ảnh 2D khác nhau.

1. **Prefabs**

Khái niệm này được dùng để chỉ việc sử dụng lại các đối tượng giống nhau trong game chỉ cần thực hiện thao tác khởi tạo lại các giá trị của vị trí cùng các tỉ lệ biến dạng, góc quay từ đối tượng ban đầu.

1. **Sounds**

Hiểu đơn giản chính là hệ thống âm thanh có trong game.

1. **Script**:

Script là thành phần quan trọng nhất trong Unity, có thể xem script như là linh hồn của game. Chúng ta có thể viết kịch bản cho game bằng C#, JavaScript hoặc Boo (Một dẫn xuất của ngôn ngữ Python). Theo nhiều người đã sử dụng Unity thì code bằng C# sẽ giúp game chạy nhanh hơn và giúp kiểm soát code tốt hơn do tất cả các biến phải được khai báo rõ ràng. Mỗi file script C# là một class bắt buộc kế thừa từ lớp MonoBehaviour, có tên class phải trùng với tên file script.

1. **Scenes**

Trong Unity, chúng ta có thể xem Scenes là các màn chơi, cấp độ chơi riêng lẻ hoặc các vùng của nội dung trò chơi. Ví dụ như Main menu, Options, About,... Bằng cách xây dựng trò chơi với nhiều cảnh, chúng ta sẽ có thể phân phối thời gian tải và thử nghiệm các phần khác nhau của trò chơi riêng lẻ một cách nhanh chóng và chính xác nhất.

1. **Assets**

Bao gồm toàn bộ những công cụ khác phục vụ cho một dự án game đó là sprite, sound, script, animation, scenes,... Đây là kho tài nguyên cho việc xây dựng game trong một project của unity. Các tài nguyên này có thể là hình ảnh, âm thanh,... Unity sẽ tham chiếu đến các tập tinh chúng ta sẽ sử dụng để tạo ra các tài nguyên cho trò chơi.

1. **Camera**

Đây là một gameobject đặc biệt có trong scenes và được sử dụng để xác định tầm nhìn cũng như quan sát các đối tượng khác có trong game.

1. **Transform**

Hiểu đơn giản đây là ba phép biến đổi trong game bao gồm có tịnh tiến, phóng to - thu nhỏ và quay theo các trục của một đối tượng nhất định

**1.3 Ứng dụng của Unity**

* Ứng dụng Unity vào thực tế ảo AR/VR
* Ứng dụng Unity vào game
* Ứng dụng vào phần mềm di động
* Ngoài ra Unity còn có các tiện ích:
* Trong lĩnh vực AR/VR Unity 3D:
* ARCore: tool này hỗ trợ về AR App trên Android
* ARKit: tool này hỗ trợ về AR App trên IOS
* AVFoundation: tool này hỗ trợ tạo ra AR App cả trên Android và IOS
* Vuforia: tool này hỗ trợ tạo ra VR App cả trên Hololens
* Các lĩnh vực chung: (chỉ liệt kê một số tiện ích hay dùng, còn rất nhiều tiện ích khác)
* [Unity Timeline](https://docs.unity3d.com/Manual/TimelineSection.html): (dùng để làm animation hoặc dựng phim ngắn, trailer không cần code)
* [Unity Probuilder](https://unity3d.com/unity/features/worldbuilding/probuilder): (dùng để dựng cảnh và cho phép bạn cấu trúc chi tiết hơn)
* [PathCreator](https://assetstore.unity.com/packages/tools/utilities/b-zier-path-creator-136082):  (như tên gọi đây là plugin giúp bạn tạo những đoạn đường theo ý muốn của bản thân mà không cần code)
* [Unity Shader Graph](https://unity.com/shader-graph):  (giúp bạn dễ dàng điều chỉnh shader cho object mà không cần code)
* [DOTween](https://assetstore.unity.com/packages/tools/animation/dotween-hotween-v2-27676): (tool này cũng là 1 dạng plugin giúp bạn đơn giản hóa việc di chuyển các object 3D bằng những dòng code đơn giản)

**1.4 Tại sao nên dùng Unity**

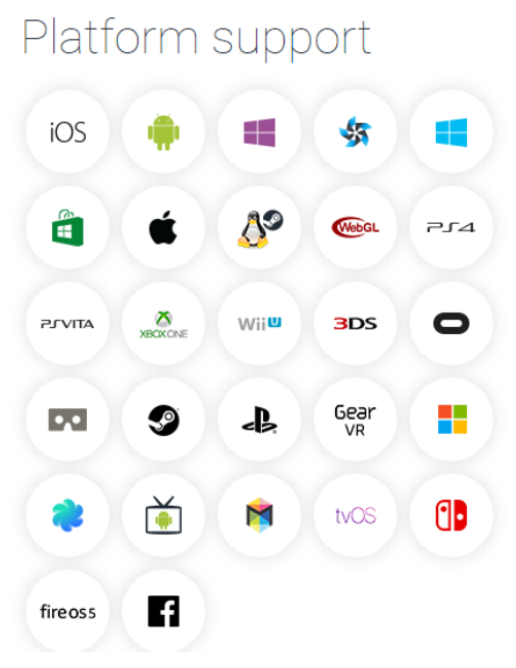
Ngoài những chức năng cơ bản của một hệ sinh thái đó, Unity còn có những ưu việt vượt trội so với các engine khác:

* Sử dụng ngôn ngữ lập trình phổ biến tại Việt Nam là **C#**.
* Hỗ trợ đa nền tảng: Lập trình viên dùng Unity3D Engine và ngôn ngữ C# hoặc script để phát triển game hoàn thiện, sau đó Unity cho phép bạn “build” ra các phiên bản cho các nền tảng khác mà không cần viết thêm dòng code nào, giúp bạn rút ngắn rất nhiều thời gian xây dựng game cũng như nâng cao mức độ an toàn khi lập trình game. Những nền tảng mà Unity đang hỗ trợ gồm PlayStation 3, Xbox 360, Wii U, iOS, Android, Windows, Blackberry 10, OS X, Linux, trình duyệt web.
* Dễ sử dụng, ngay cả với Lập trình viên nghiệp dư, do Unity3D được xây dựng trong một môi trường phát triển tích hợp, cung cấp một hệ thống toàn diện cho các lập trình viên, từ soạn thảo mã nguồn, xây dựng công cụ tự động hóa đến trình sửa lỗi.
* Tính kinh tế cao: Những cá nhân và doanh nghiệp có doanh thu dưới 100.000 USD/năm được dùng miễn phí Unity3D, và Unity Technology chỉ thu phí 1.500 USD/năm cho bản Pro- một con số rất khiêm tốn so với những gì engine này mang lại.
* Rất được ưa chuộng tại Việt Nam, ngay cả trong các game studio lớn như VTC, VNG, Glass-Egg.
* Thư viện phong phú, đa dạng: Unity có nhiều thư viện, các công cụ hỗ trợ làm game nhanh hơn, thông minh hơn, các đối tượng được tạo sẵn, và tất cả các thư viện này đều “mở”, cho phép cộng đồng tự do sử dụng và sáng tạo nên các sản phẩm của chính mình, thậm chí có thể bán trên Asset Store của Unity.
* Cộng đồng rất lớn mạnh: là engine phổ biến nhất trên thế giới, Unity có cộng đồng sử dụng rất lớn mạnh. Mọi thắc mắc của bạn về Unity đều sẽ được trả lời trên website cộng đồng ( [http://answers.unity3d.com](http://answers.unity3d.com/)).
* **Hỗ trợ Networking để phát triển MMO game**.

**1.5 Ưu và nhược điểm của Unity**

**Ưu điểm:**

* Unity có một cộng đồng rất lớn về asset và plugin – trong đó có rất nhiều resources free và có nhiều thứ rất đáng bỏ tiền
* Chức năng cốt lõi đa dạng bao gồm: cung cấp công cụ dựng hình (kết xuất đồ họa) cho các hình ảnh 2D hoặc 3D, công cụ vật lý (tính toán và phát hiện va chạm), âm thanh, mã nguồn, hình ảnh động, trí tuệ nhân tạo, phân luồng, tạo dòng dữ liệu xử lý, quản lý bộ nhớ, dựng ảnh đồ thị và kết nối mạng. Nhờ có các engine mà công việc làm game trở nên ít tốn kém và đơn giản hơn.
* Unity có bộ công cụ rất trực quan và editor có thể mở rộng bằng plugins.
* Hỗ trợ đa nền tảng: Một trong các thế mạnh của Unity3D chính là khả năng hỗ trợ gần như toàn bộ các nền tảng hiện có bao gồm: PlayStation 3, Xbox 360, Wii U, IOS, Android, Windows, Blackberry 10, OS X, Linux, trình duyệt Web và cả Flash. Nói cách khác, chỉ với một gói engine, các studio có thể làm game cho bất kỳ hệ điều hành nào và dễ dàng convert chúng sau những hệ điều hành khác nhau. Đồng thời, đây cũng là giải pháp cho các game online đa nền tảng - có thể chơi đồng thời trên nhiều hệ điều hành.
* Unity hỗ trợ rất nhiều định dạng asset khác nhau và có thể tự động chuyển đổi đến định dạng phù hợp nhất với nền tảng thích hợp
* Unity hỗ trợ nhiều nền tảng: di động, desktop, web và console
* Việc triển khai đến các nền tảng khác nhau cũng khá dễ quản lý
* Bạn có thể dễ dàng xây dựng một game 3D mà không cần cấu hình quá phức tạp
* Unity bản free có hầu hết những tính năng quan trọng nhất
* Unity bản trả phí phù hợp với các developer chuyên nghiệp



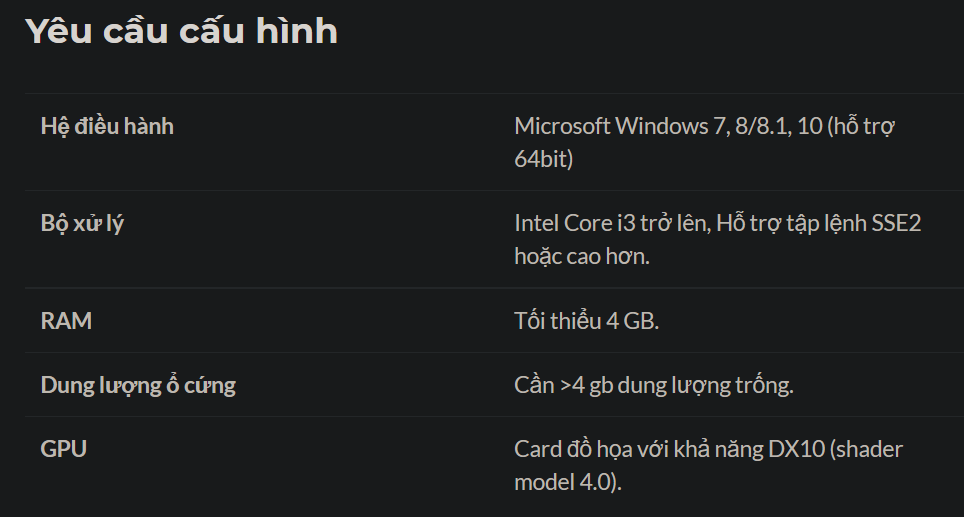
*Hình 4. Danh sách platform mà Unity support.*

**Nhược điểm:**

* Việc hợp tác rất khó khăn. Unity sử dụng một server asset rất hiệu quả để hỗ trợ các đội phát triển phần mềm hợp tác với nhau. Tuy nhiên nếu bạn không sử dụng nó thì việc chia sẻ code và asset giữa các thành viên trong team có thể gây ra những vấn đề nghiêm trọng. Lựa chọn tốt nhất là sử dụng một số công cụ quản lý resource bên ngoài nhưng có một vài binary file không thể merge được với nhau và việc cập nhật asset có thể gây nên một số vấn đề trong scenes, mất kết nối đến script và các đối tượng khác
* Hiệu năng chưa thật sự ấn tượng cho đến khi Unity 5 ra mắt. Unity 5 đã chạy hầu hết trên một luồng duy nhất và hầu như không sử dụng thêm 1 nhân phụ nào trên các thiết bị di động. Bộ biên dịch chưa được tối ưu tốt cho các bộ xử lý ARM trên hầu hết các thiết bị di động. Để giải quyết vấn đề này thì Unity đã quyết định transpile (source-to-source compiler) sang C++ và sử dụng LLVM để tối ưu được nhiều hơn thay vì giải quyết vấn đề này trực tiếp trên các phiên bản sau này
* Mã nguồn của engine không được công bố kể cả cho những người dùng chấp nhận trả tiền. Điều đó có nghĩa là nếu bạn gặp một bug với engine bạn phải chờ Unity fix chúng trong các bản tiếp theo. Điều này có thể gây nên những vấn đề nghiêm trọng với project của bạn

**2.Cài đặt Unity**

**2.1 Những yêu cầu khi cài đặt Unity**



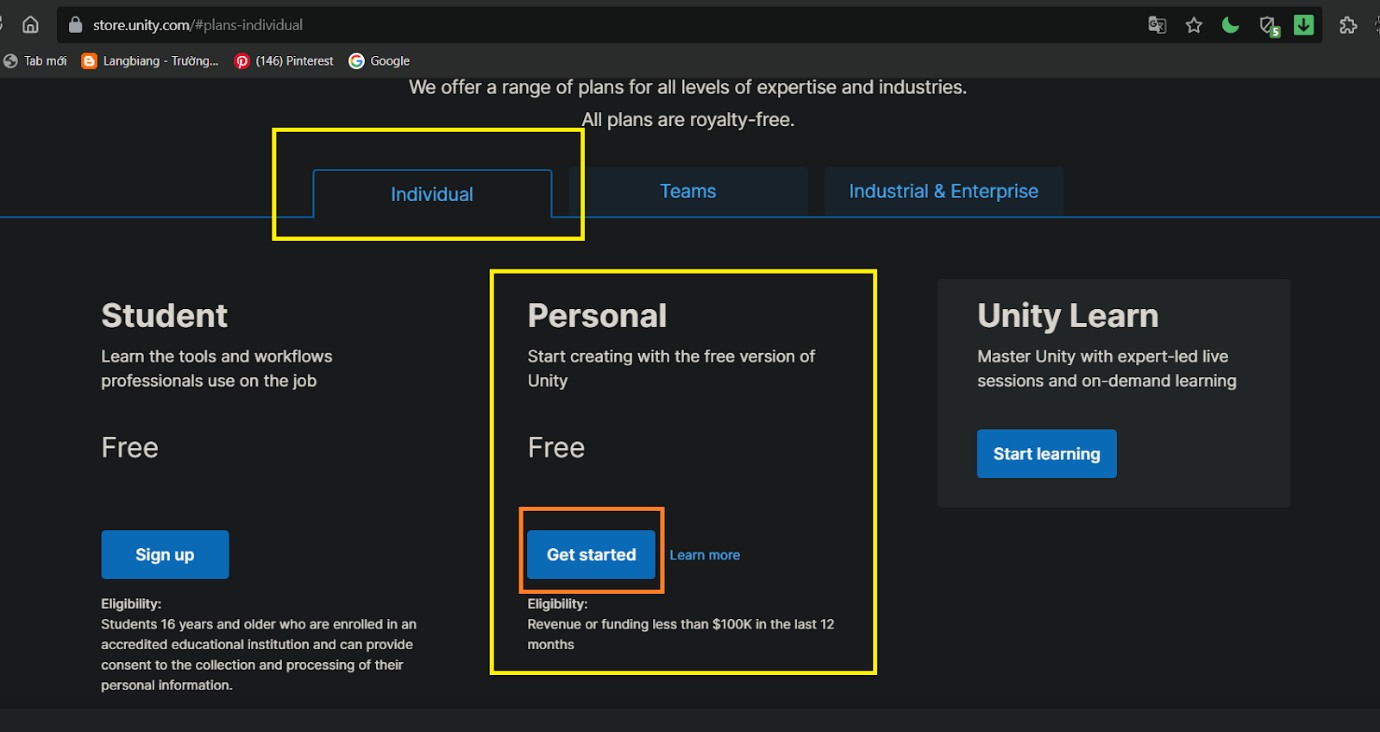
*Hình 5. Cấu hình tối thiểu để cài đặt Unity.*

**2.2 Cài đặt Unity**



*Hình 6. Trang chủ Unity.*

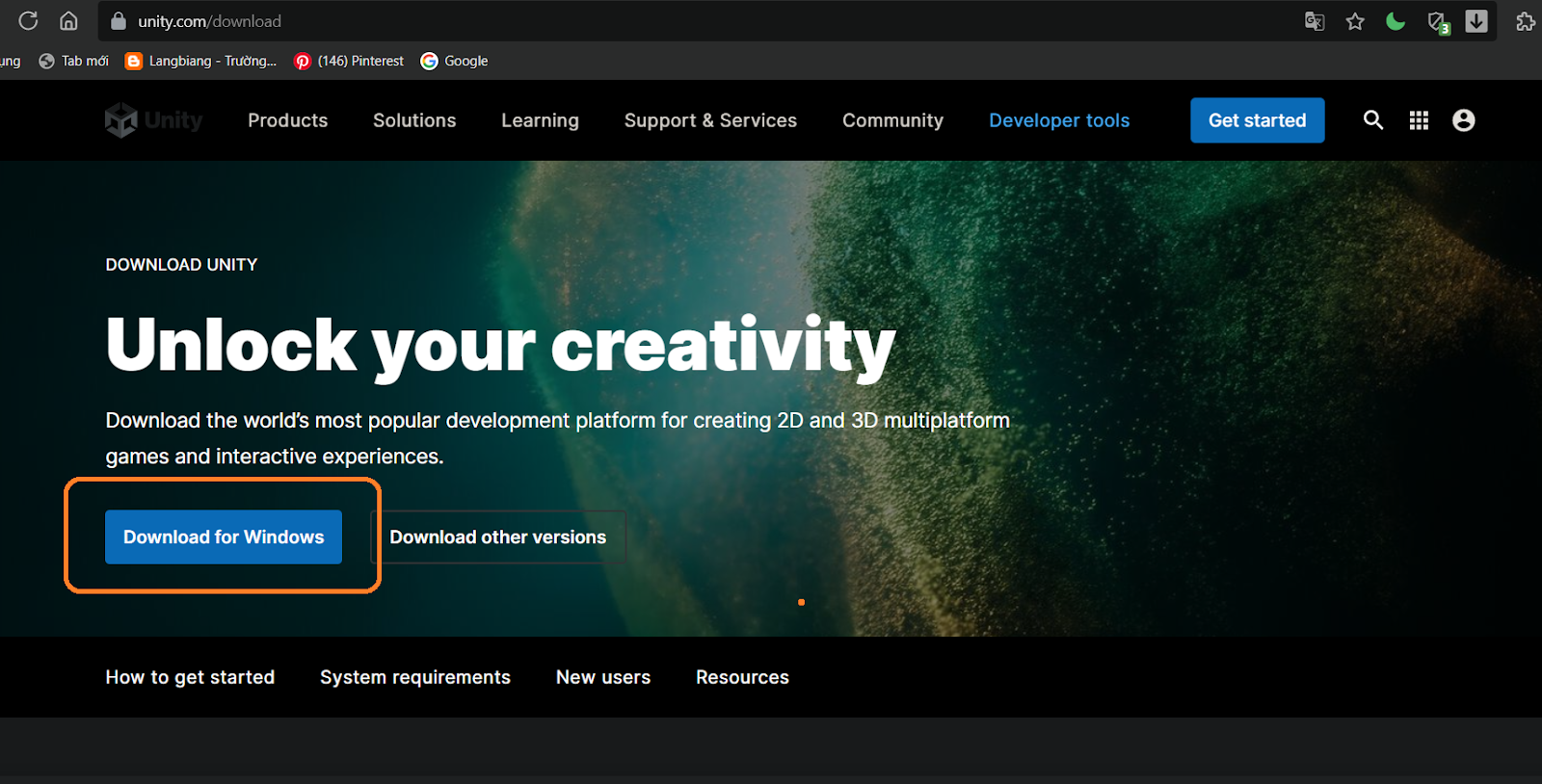
Chọn mục “Choose your Unity + download”



*Hình 7. Hướng dẫn cài đặt Unity (1).*

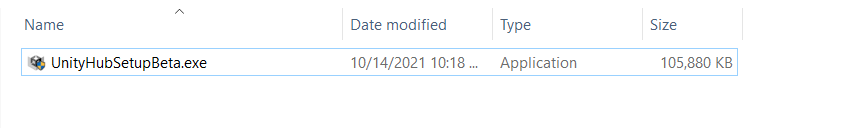
Nhấn vào mục “individual” và chọn “Personal” để tải Unity miễn phí

Chọn “Get started” để bắt đầu

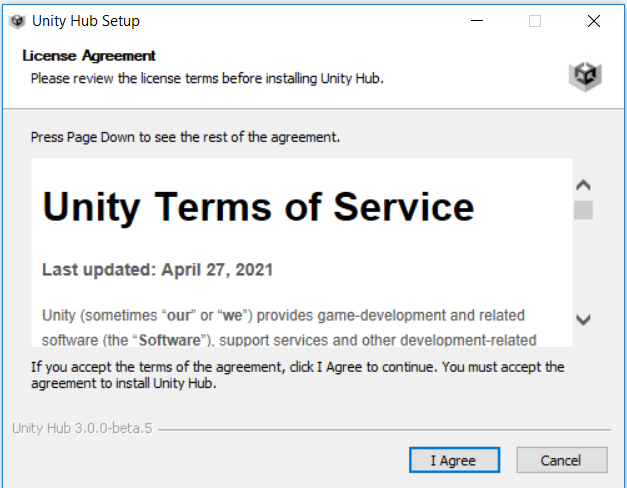


*Hình 8. Hướng dẫn cài đặt Unity (2).*

Nếu bạn sử dụng hệ điều hành window thì nhấn vào “Download for Windows”. Nếu bạn sử dụng hệ điều hành khác, hãy chọn “Download Other Versions”

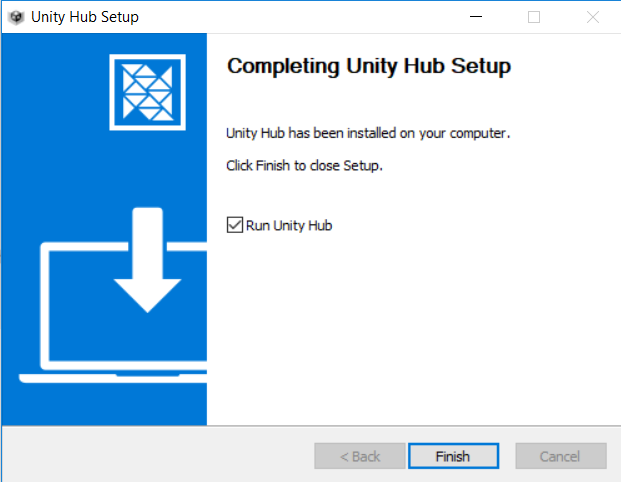


*Hình 9. Hướng dẫn cài đặt Unity (3).*



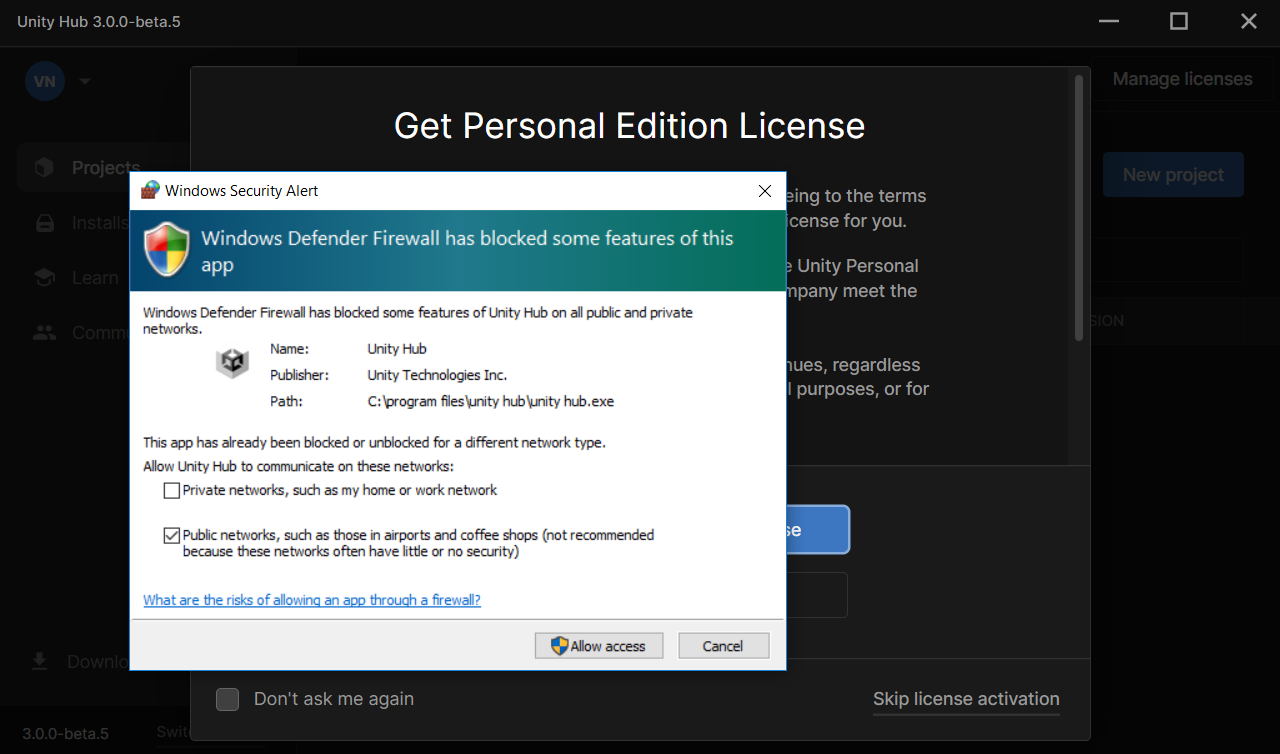
*Hình 10. Hướng dẫn cài đặt Unity (4).*

Chọn “I Agree”



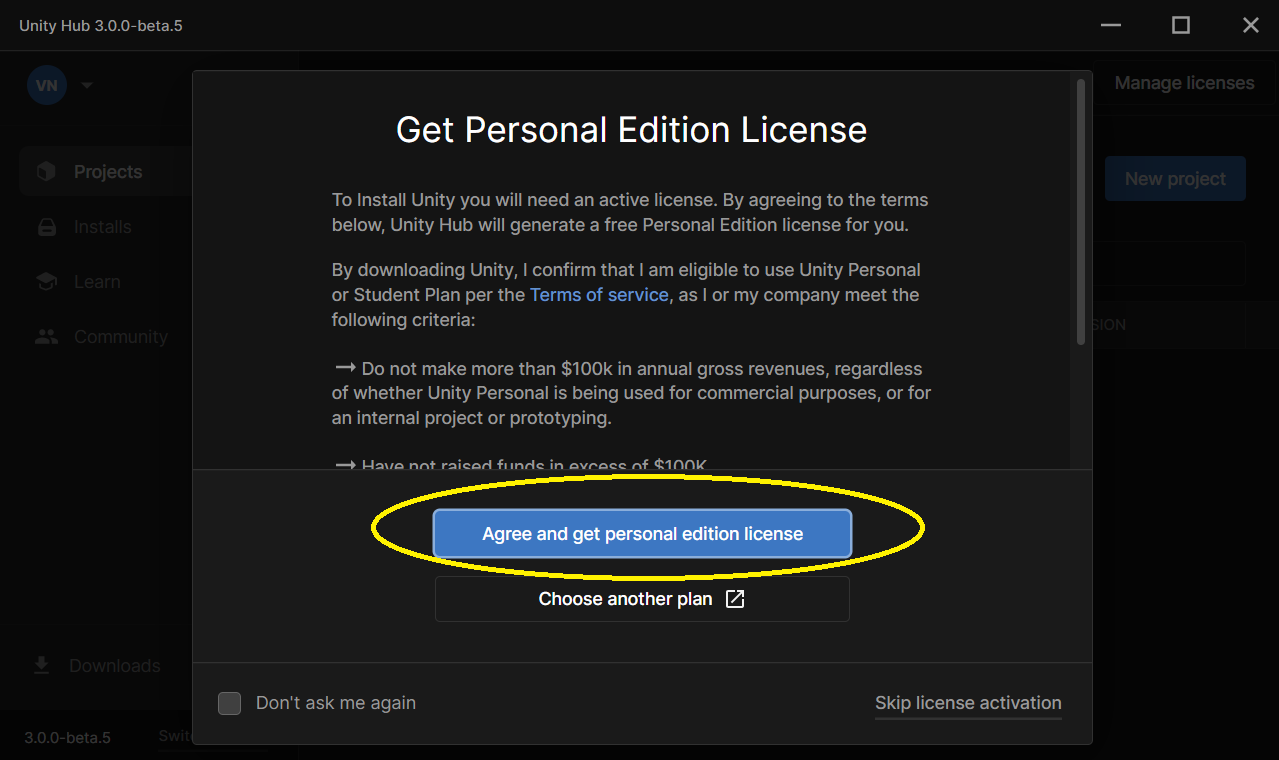
*Hình 11. Hướng dẫn cài đặt Unity (5).*

chọn “Finish”



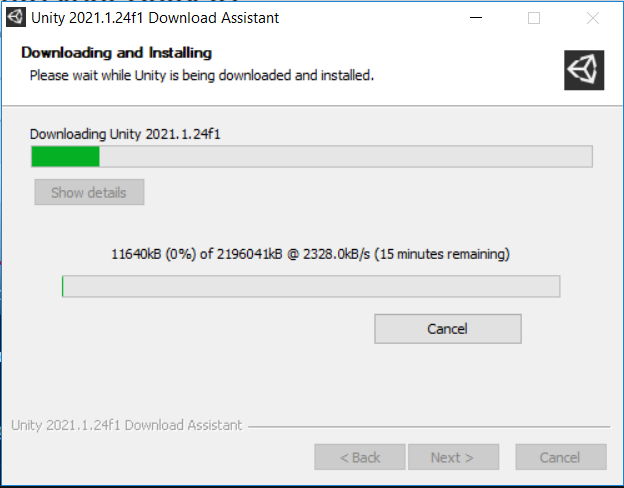
*Hình 12. Hướng dẫn cài đặt Unity (6).*

Chọn “check” vào Public networks và nhấn “Allow access”



Hình 13. Hướng dẫn cài đặt Unity (7).

Chọn “Agree and get personal edition license”



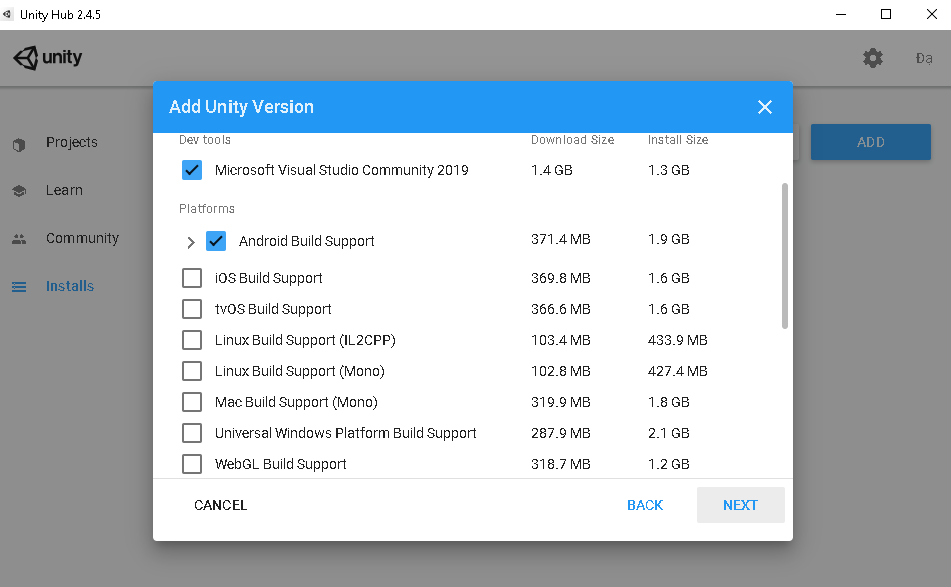
*Hình 14. Hướng dẫn cài đặt Unity (8).*

Làm theo hướng dẫn, cho tới khi chương trình cài đặt thành công.

**2.3 Thiết lập môi trường**

Sau khi cài đặt Unity thành công, chúng ta cần thiết lập môi trường làm việc của Unity. Tùy theo nhu cầu của mỗi người, ở đây chúng ta sẽ làm một game trên thiết bị di động, nên chúng ta cần cài đặt các gói (package) phù hợp.

Để làm việc với mã nguồn C#, Unity yêu cầu người dùng phải cài đặt Dev tools là Microsoft Visual Studio để việc soạn thảo và biên dịch code trở nên dễ dàng. Ngoài ra, chúng ta cần chọn thêm Platforms Android Build Support là công cũ hỗ trợ lập trình game trên điện thoại di động.



*Hình 15. Thiết lập môi trường lập trình.*

**3. Ngôn ngữ C#**

C# là một ngôn ngữ lập trình hướng đối tượng được phát triển bởi Microsoft, là phần khởi đầu cho kế hoạch .Net của họ. Tên của ngôn ngữ bao gồm ký tự thăng theo Microsoft nhưng theo ECMA là C#, chỉ bao gồm dấu số thường. Microsoft phát triển C# dựa trên C++ và Java. C# được miêu tả là ngôn ngữ có được sự cân bằng giữa C++, Visual Basic, Delphi và Java. C# được thiết kế chủ yếu bởi Anders Heijsberg - kiến trúc sư phần mềm nổi tiếng với các sản phẩm Turbo Pascal, Delphi, J++, WFC. Tuy nhiên, ngôn ngữ C# có ý nghĩa cao khi nó thực thi những khái niệm lập trình hiện đại. C# bao gồm tất cả những hỗ trợ cho cấu trúc, thành phần component, lập trình hướng đối tượng. Sau đây là một số điểm nổi bật của ngôn ngữ C#.

Cú pháp C# hàm ý, nhưng nó cũng đơn giản và dễ dàng để học. Nếu có kinh nghiệm với C++, C hoặc Java thì có thể bắt đầu làm việc hiệu quả trong C# với thời gian rất ngắn, vì C# có cú pháp tương tự, tuy nhiên C# đơn giản hơn so với C++ và cung cấp các tính năng mạnh mẽ như các loại giá trị nullable, enum, delegate, biểu thức lambda và việc truy cập bộ nhớ trực tiếp mà chúng ta không thể tìm thấy được trong ngôn ngữ Java. C# hỗ trợ kiểu dữ liệu và phương thức chung giống (Java), ứng dụng C# nhanh bảo mật và mã nguồn có thể tái sử dụng.

Là một ngôn ngữ hướng đối tượng, C# hỗ trợ các khái niệm về đóng gói (encapsulation), kế thừa (inheritance) và đa hình (polymorphism). Tất cả các biến và phương thức, bao gồm cả phương thức override virtual methods, được đóng gói trong Class. Một lớp có thể kế thừa trực tiếp từ một lớp cha, nhưng nó có thể có thực hiện nhiều interfaces khác nhau… Ngoài những nguyên tắc cơ bản của ngôn ngữ hướng đối tượng C#, C# còn có những sáng tạo riêng: Encapsulation, Properties, Attribute, XML, Language-Integrated Query (LinQ).

Cách đây không lâu, Unity 3D đã hỗ trợ ba ngôn ngữ chính. Đầu tiên là Boo, được đi kèm với JavaScript, sau đó là C#. Kể từ phiên bản Unity mới nhất, C# được coi là ngôn ngữ lập trình chính hoặc hạng nhất. Cả JavaScript và Boo đều đã bị loại bỏ.

Khi các phiên bản mới của .NET được giới thiệu, ngôn ngữ đi cùng với nó cũng được nâng cấp, mặc dù với các số phiên bản khác nhau. Gây nhầm lẫn? Đúng. Thật không may, để tăng thêm sự nhầm lẫn, còn có ba phiên bản .NET bao gồm .NET Standard, .NET Framework và .NET Core. Sự khác biệt giữa những điều này không ảnh hưởng đến những gì chúng ta sẽ tìm hiểu ở đây, vì vậy tôi sẽ để nghiên cứu đó tùy bạn thực hiện. Hiện tại, chúng ta không quá quan tâm đến phiên bản của nó, miễn là các tính năng chúng ta tìm hiểu về ngôn ngữ C # hoạt động trong ngữ cảnh của Unity.

C # được sử dụng bởi một cộng đồng lớn các lập trình viên ứng dụng nói chung, không chỉ các kỹ sư trò chơi. Unity 3D cũng là một công cụ trò chơi đa nền tảng và một nhà phát triển làm việc trên OSX hoặc Windows đều có thể chia sẻ và sử dụng các tệp cho Unity 3D. C # giống nhau trên cả Mac và PC cũng như Android và iOS dành cho thiết bị di động, vì vậy chúng ta không cần lo lắng về hệ điều hành nào sẽ sử dụng.

**3.1. Cấu trúc cơ bản của C#**

1. **Using**

Thành phần using có tác dụng khai báo thư viện nào sẽ sử dụng trong chương trình (thư viện và tập hợp các phương thức, kiểu dữ liệu có sẵn nào đó được tạo ra nhằm hỗ trợ cho việc lập trình nhanh chóng và hiệu quả hơn).

Ví dụ, khi chúng ta muốn viết chương trình game thì phải sử dụng thư viện đồ họa có sẵn, như vậy việc lập trình sẽ nhanh chóng và dễ dàng hơn việc tự viết thủ công.

Cú pháp để khai báo thư viện như sau: using <tên thư viện>;

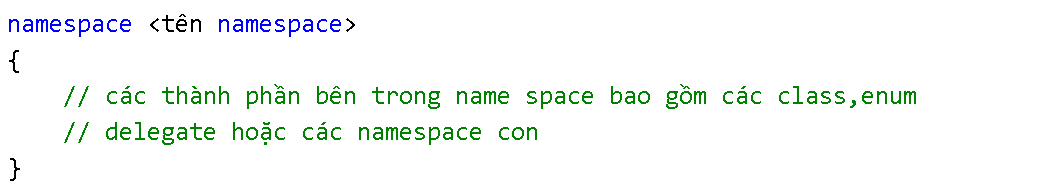
vd: using System.UnityEngine;

1. **Namespace**

Một namespace trong C# được thiết kế để phân nhóm toàn bộ các kiểu dữ liệu theo một cấu trúc phân cấp. Nhờ có namespace, kiểu dữ liệu được quản lý tốt hơn và tránh được hiện tượng xung đột tên biến. Các tên biến được khai báo trong một namespace không xung đột với cùng tên biến đó nhưng được khai báo ở một namespace khác.

Namespace báo cho trình biên dịch biết rằng các thành phần nằm bên trong khối “{ … }” ngay dưới namespace thuộc về namespace đó, điều này sẽ giúp chúng ta tránh được những xung đột về code.

Một định nghĩa namespace trong C# bắt đầu với từ khóa namespace được theo sau bởi tên của namespace đó, như sau:

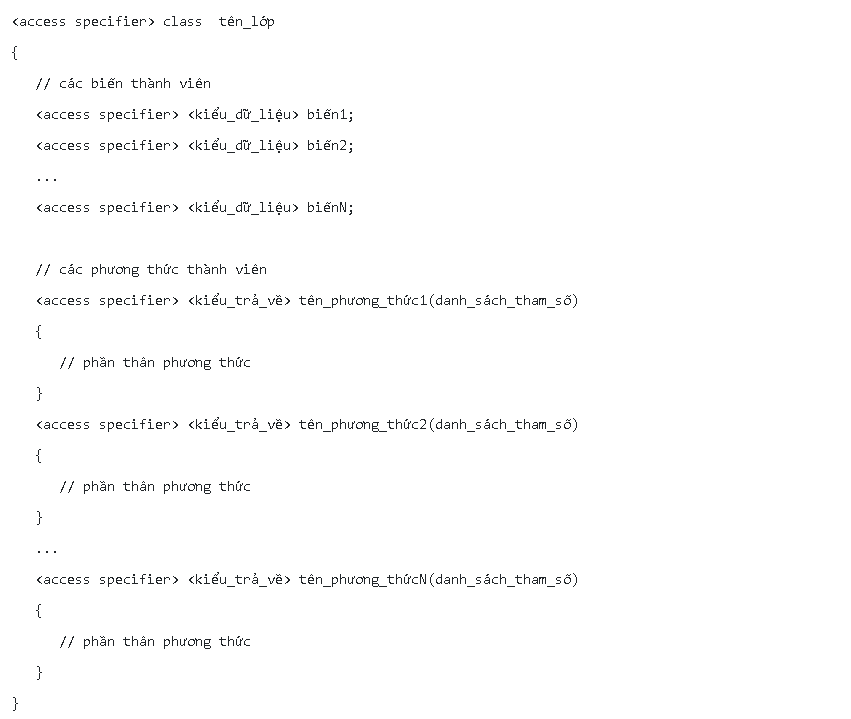


*Hình 16. namespace trong C#*

1. **Class**

Khi chúng ta định nghĩa một lớp (class) trong C#, chúng ta định nghĩa một blueprint cho một kiểu dữ liệu. Điều này không thực sự định nghĩa bất kỳ dữ liệu nào, nhưng nó định nghĩa ý nghĩa của tên lớp đó. Tức là, một đối tượng của lớp đó gồm những cái gì, các hoạt động nào có thể được thực hiện trên đối tượng đó. Các đối tượng là instance (sự thể hiện) của một lớp. Các phương thức và các biến mà cấu tạo nên một lớp được gọi là các thành viên của lớp đó.

Một định nghĩa lớp trong C# bắt đầu với từ khóa class được theo sau bởi tên lớp và phần thân lớp được bao quanh bởi các dấu ngoặc ôm. Dưới đây là form chung của một định nghĩa lớp trong C#.



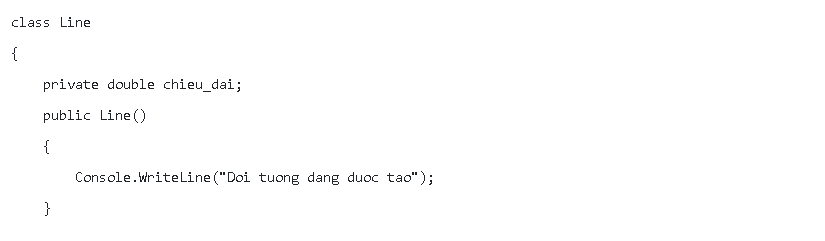
*Hình 17. Class trong C#*

Ghi chú:

* Access specifier xác định quy tắc truy cập cho các thành viên cũng như chính lớp đó. Nếu không được đề cập, thì Access Specifier mặc định cho một kiểu lớp là Internal. Chế độ truy cập mặc định cho các thành viên là private.
* kiểu\_dữ\_liệu xác định kiểu biến và trả về kiểu dữ liệu mà phương thức trả về.
* Để truy cập các thành viên lớp, chúng ta sử dụng toán tử dot (.).
* Toán tử dot (.) liên kết tên của mổ đối tượng với tên của một thành viên.

Khi nhắc đến class, không thể thiếu Constructor. Một constructor của một class là một thành viên đặc biệt của một lớp. Được thực thi bất cứ khi nào chúng ta tạo ra các đối tượng mới của lớp đó.

Một constructor có tên giống như tên lớp và nó không có bất kỳ kiểu trả về nào. Dưới đây là ví dụ minh họa khái niệm constructor trong C#:



*Hình 18. Constructor trong C#*

Một constructor mặc định trong C# không có bất kỳ tham số nào, nhưng nếu cần, một constructor có thể có tham số. Những constructor này được gọi là constructor **được tham số hóa**.

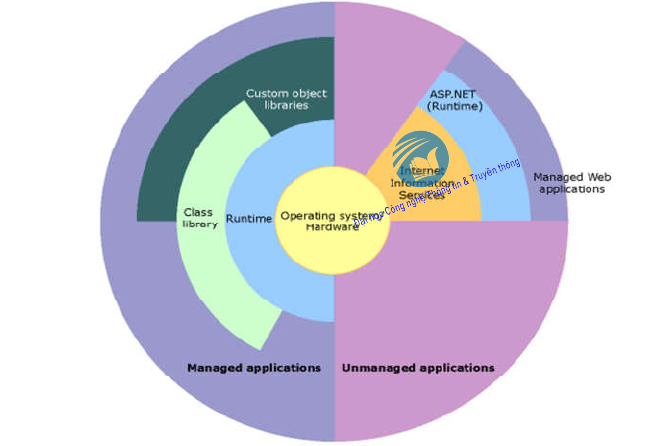
Ngoài ra, chúng ta có thể định nghĩa các thành viên lớp là static bởi sử dụng từ khóa static trong C#. Khi chúng ta khai báo một thành viên lớp là static, nghĩa là, dù cho có bao nhiêu đối tượng của lớp được tạo, thì chỉ có một bản sao của thành viên static.

Từ khóa static ngụ ý rằng chỉ có một instance (sự thể hiện) của thành viên tồn tại cho một lớp đó. Các biến static được sử dụng để định nghĩa các hằng số (constant) bởi vì giá trị của chúng có thể được thu nhận bằng việc gọi lớp đó mà không cần tạo một instance của nó. Các biến static có thể được khởi tạo bên ngoài hàm thành viên hoặc định nghĩa lớp. Chúng ta cũng có thể khởi tạo các biến static bên trong định nghĩa của lớp.

1. **Cách ứng dụng C# hoạt động**

Các ngôn ngữ .NET như C#, VB.NET, VJ#, VC++ .NET đều có cách thức hoạt động giống nhau và là một thành phần của .NET Framework do đó chúng ta không thể chạy ứng dụng này trên các máy không cài đặt .NET Framework version thích hợp.

Trước hết cần có cái nhìn khái quát về .Net Framework, gồm 2 phần chính: common language runtime (CLR) và .Net Framework class library.



*Hình 19.*

Trong hình 19, những gì chúng ta viết ra sẽ là ***custom object libraries***, ***class library***sẽ chứa những class chung, có thể sử dụng nhiều lần. Các class này định nghĩa console, window hay button, thứ mà chúng ta cần để ứng dụng chạy mà không cần viết lại.

***Runtime*** chính là nền tảng của .NET Framework, cách dễ hiểu nhất là coi nó như máy ảo có nhiệm vụ quản lý bộ nhớ, quản lý luồng, cho phép truy cập các dữ liệu khác để chương trình có thể hoạt động trơn tru.

Khi chúng ta viết xong một chương trình, qua quá trình biên dịch sẽ cho ta cùng một kết quả, đó là ***Microsoft Intermediate Language*** (MSIL code), sau đó ***JIT compiler*** sẽ chuyển mã MSIL code sang mã máy, thứ mà CPU có thể hiển thị thực thi.

**3.2. Kiến trúc .Net Framework**

Như đã giới thiệu ở phần trước, .NET Framework bao gồm 2 phần chính là Common Language Runtime (CLR) và Base Class Libraries (BCL), nhưng trên thực tế thì BCL bao gồm ASP.NET và Windows Forms, nên một số tài liệu đã giới thiệu .NET Framework bao gồm ba phần là bộ thực thi ngôn ngữ chung (Common Language Runtime), các lớp lập trình hợp nhất hay còn gọi là các thư viện lớp cơ sở (Base Class Libraries) và một phiên bản cấu thành của Microsoft Active Server Pages gọi là Microsoft ASP.NET. Một trong các thành phần này đều có vai trò cực kỳ quan trọng trong việc phát triển các dịch vụ và ứng dụng .NET.

**Common Language Runtime** (gọi tắt là bộ thực thi) được xây dựng trên các dịch vụ hệ điều hành. Nó chịu trách nhiệm thực hiện các ứng dụng và đảm bảo các phần liên quan đến ứng dụng đều được đáp ứng như: quản lý bộ nhớ, an ninh bảo mật, tích hợp ngôn ngữ,... Bộ thực thi bao gồm nhiều dịch vụ hỗ trợ phát triển và triển khai ứng dụng cũng như cải thiện tính đáng tin cậy của ứng dụng. Tuy vậy, những nhà phát triển trên thực tế không tương tác với CLR. Họ sử dụng một tập các thư viện lớp cơ sở được xây dựng bên trên bộ thực thi này thông qua các ngôn ngữ lập trình. Như là một phần của các lớp cơ sở, .NET Framework còn cung cấp một mô hình lập trình ứng dụng Web ASP.NET và Windows Form (để xây dựng các ứng dụng trên Windows). Riêng ASP.NET cung cấp các thành phần và các dịch vụ ở mức cao hơn nhằm mục đích phát triển các dịch vụ và các ứng dụng Web XML.

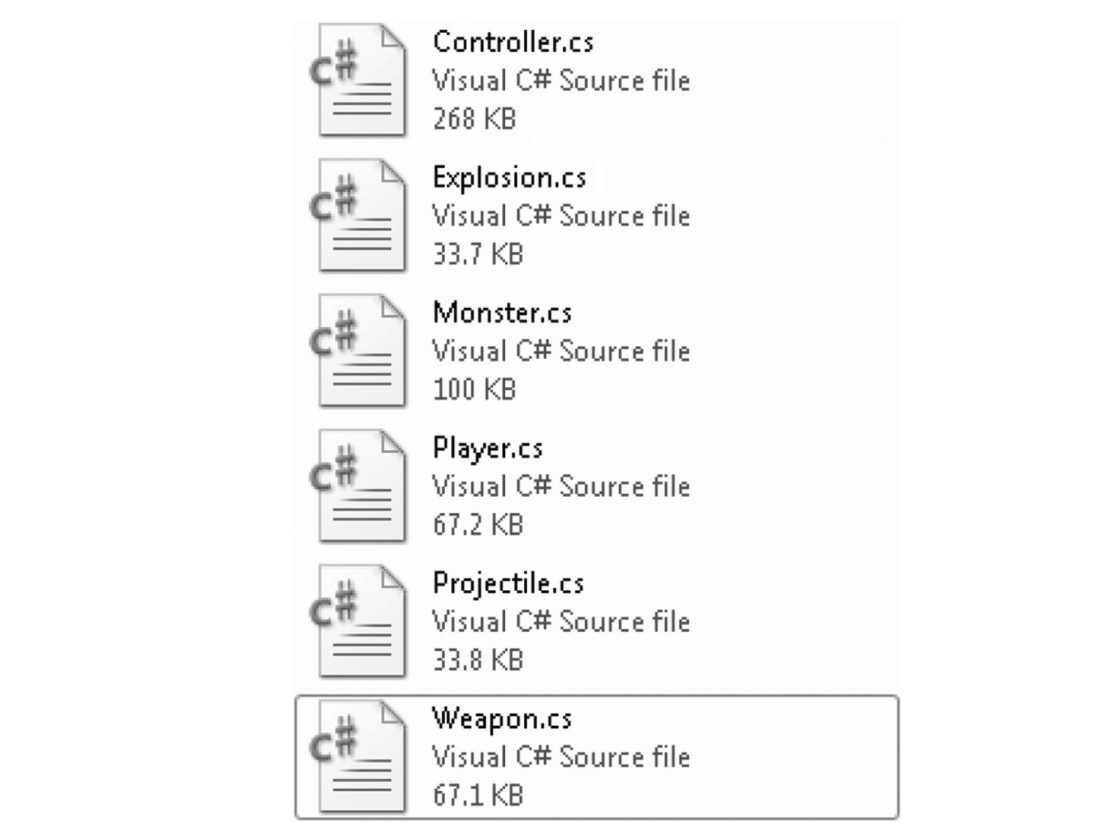
Trong phần tiếp theo, chúng ta sẽ đi sâu vào phân tích cách Unity vận hành chương trình với mã nguồn C#.

**4. Thao tác cơ bản trên Unity**

Unity 3D và Microsoft Visual Studio sẽ là môi trường phát triển tích hợp (IDE) của chúng tôi. IDE có nhiều dạng khác nhau. Nhiều lập trình viên Windows chuyên nghiệp sử dụng Visual Studio và các nhà phát triển. Apple thường sử dụng Xcode trên OSX. Microsoft cũng đã cung cấp Visual Studio cho OSX. IDE cung cấp nhiều công cụ giúp lập trình dễ dàng hơn bằng cách đánh dấu các dòng mã và tự động hoàn thành nhiều tác vụ thường được lặp lại.

Lập trình về cơ bản là viết các hướng dẫn chi tiết về những gì cần làm và khi nào. Nói tóm lại, lấy dữ liệu rồi làm gì đó với dữ liệu. Một phần lớn của lập trình là tìm ra nơi lấy thông tin, chúng ta sẽ gọi đây là quản lý dữ liệu. Sau đó, sử dụng logic để xác định phải làm gì với dữ liệu khi chúng ta có nó. Logic lập trình được tổ chức thành các thuật toán được tìm thấy trong các câu lệnh được tổ chức thành các hàm.

Unity 3D cung cấp các công cụ để di chuyển các ký tự và đối tượng bằng mã nguồn của chúng ta. Tất cả các nội dung này đều nằm trong thư mục dự án dưới dạng các tệp riêng biệt mà người dùng có thể kiểm soát. Unity 3D cung cấp các cảnh để tạo cấu trúc cho nội dung của người dùng, mô hình 3D, hệ thống các hiệu ứng đặc biệt. Tất cả đều dựa trên những mã nguồn C# được các lập trình viên xử lý logic.



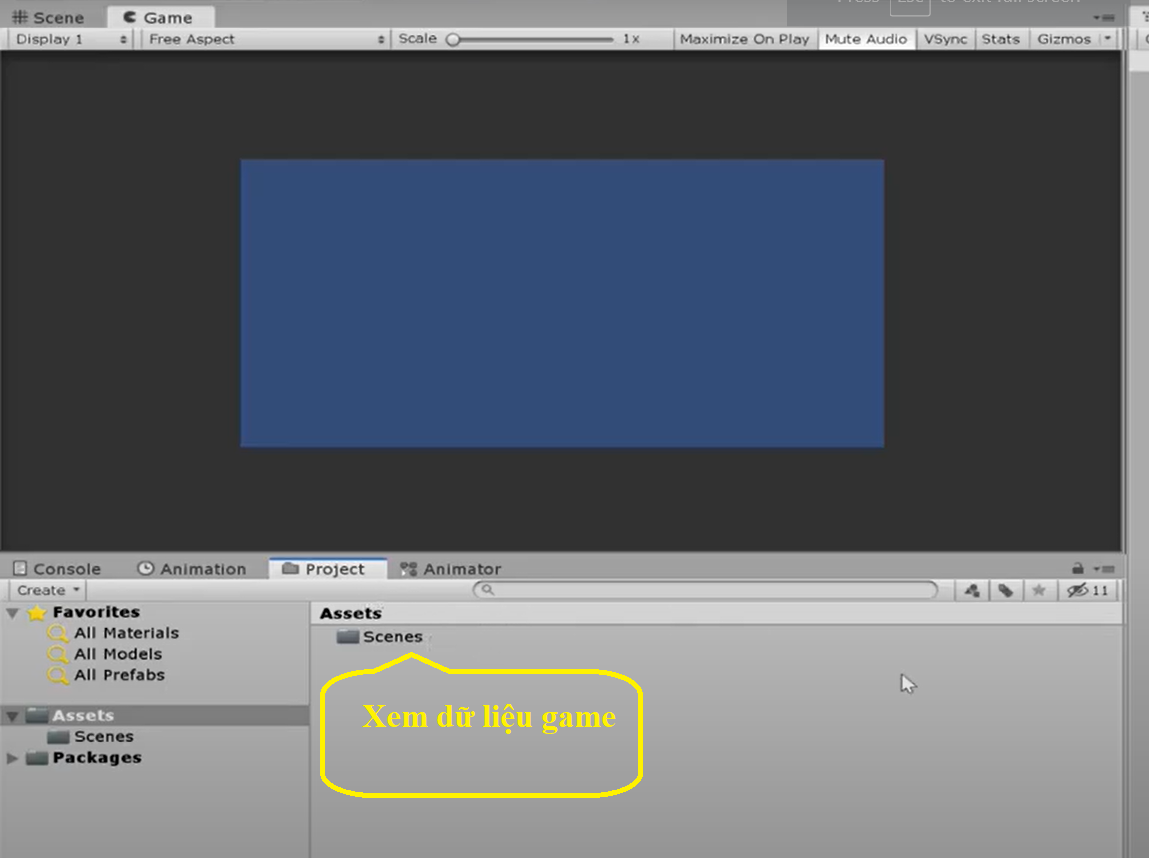
*Hình 19. Visual C# Source file.*

**4.1 Các thành phần và bố trí**

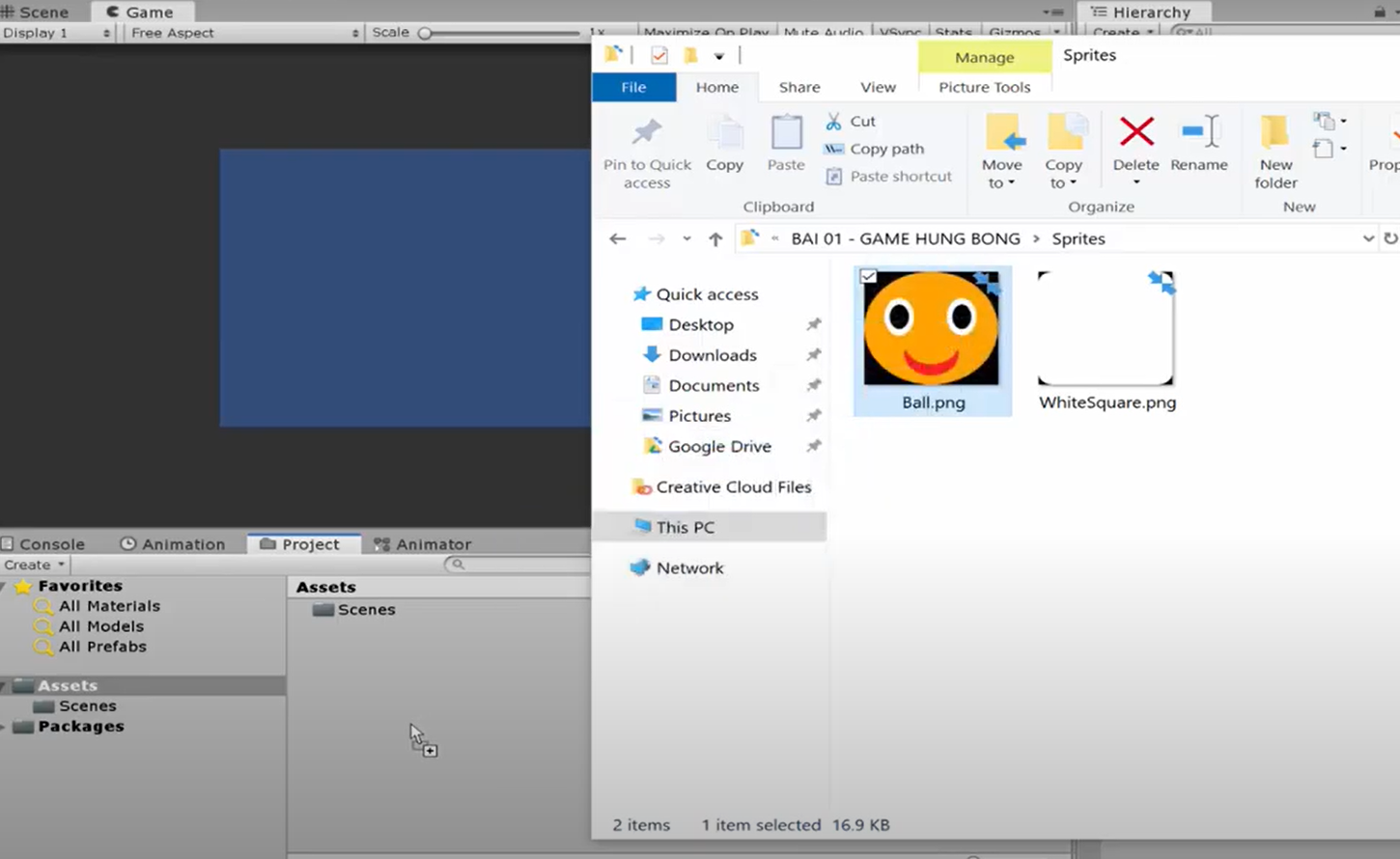
1. Project -> Assets

Mỗi Project của Unity đều chứa một thư mục Assets. Nội dung của thư mục này được hiển thị trong Project view. Đây là nơi chứa tất cả các assets để tạo Game như Scenes, Script, 3D models, Textures, Audio, Prefabs. Chúng ta không nên di chuyển assets bằng cách sử dụng Window Explorer (hay Finder trên MAC), vì nó sẽ phá vỡ cấu trúc siêu dữ liệu của assets với nhau, và luôn sử dụng Project view để cấu trúc assets.

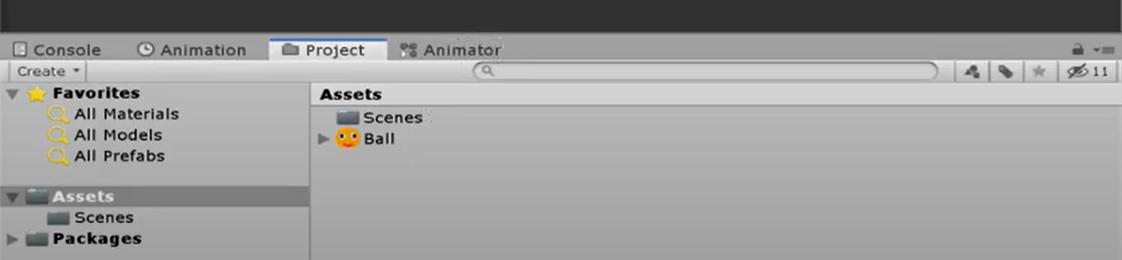
Để thêm assets vào project, chúng ta có thể kéo thả bất kỳ file nào vào trong Project view. Scenes cũng được lưu trữ trong Project view, vì đây là một level độc lập mang tính cá nhân. Chúng ta dễ dàng tạo một assets game trong Unity bằng cách chọn hình tam giác nhỏ nằm bên phải Create trong cửa sổ Project hoặc click chuột phải vào trong Project view, sau đó chọn assets tương ứng.



*Hình 20.Cửa sổ Project.*



*Hình 21. Kéo thả file vào thư mục Assets.*

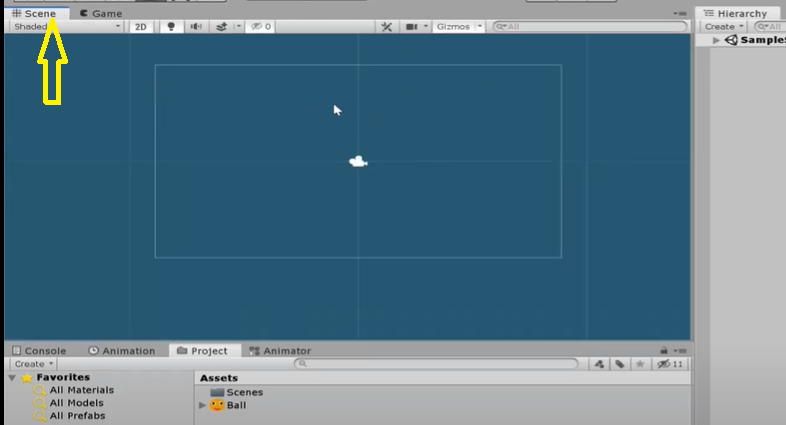


*Hình 22. File trong thư mục Assets.*

1. Scene View

Là nơi Design Game, đối tượng Maneuvering và Importanting trong Scene view (chuyển động và điều khiển) là hai trong số các chức năng quan trọng của Unity, ở góc bên phải của Scene hiển thị hướng nhìn trong không gian của camera trong Scene view hiện thời, cho phép thay đổi góc nhìn trực quan và nhanh chóng.

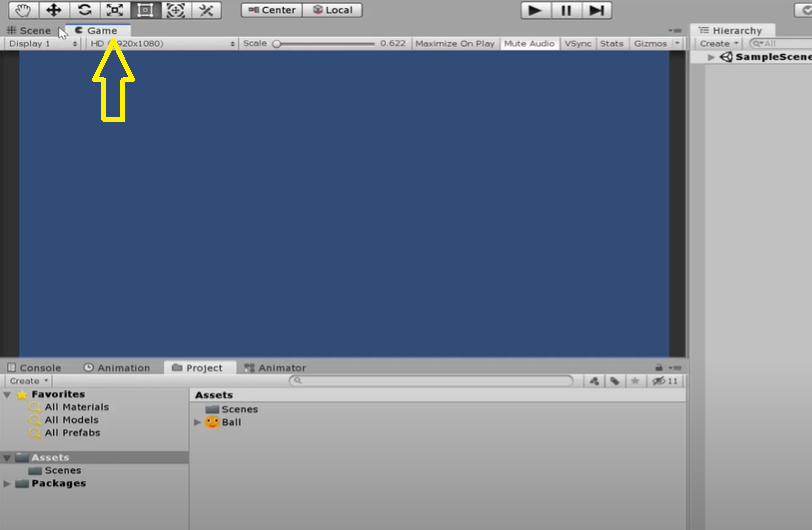
Khi xây dựng một game, chúng ta sẽ đặt rất nhiều đối tượng vào trong Game, khi đó ta có thể sử dụng các công cụ Transform Tools ở trong  Toolbar để di chuyển, xoay, phóng to, thu nhỏ từng đối tượng. Nếu chọn một đối tượng trong Scene view, xung quanh đối tượng được chọn sẽ có những thay đổi tương ứng với từng chế độ trong Transform Tools. Ta có thể thay đổi đối tượng tùy ý.



*Hình 23. Cửa sổ Scene.*

1. Game View

Game view được rendered từ những Camera trong Game. Đó là những gì được nhìn thấy khi hoàn tất, khi Game được xuất bản. Chúng ta sẽ cần ít nhất là một hoặc nhiều hơn số lượng các Camera để quyết định những gì mà người chơi sẽ nhìn thấy khi họ chơi Game.

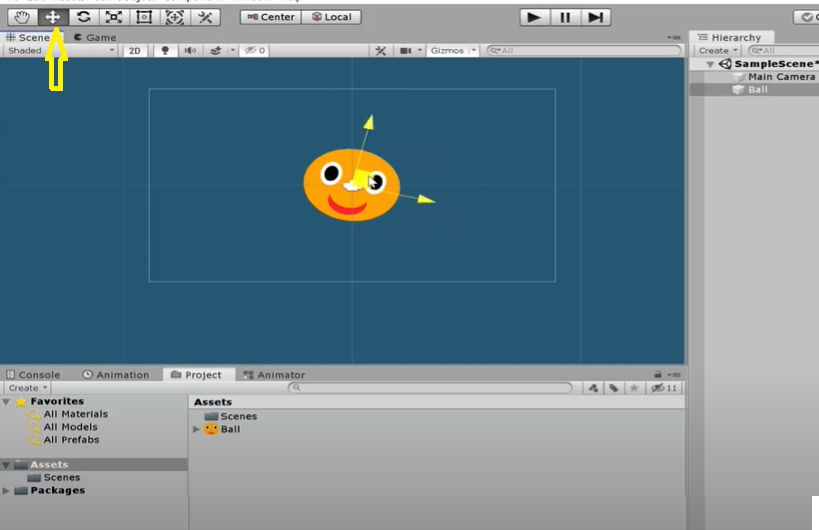


*Hình 24. Cửa sổ Game view.*

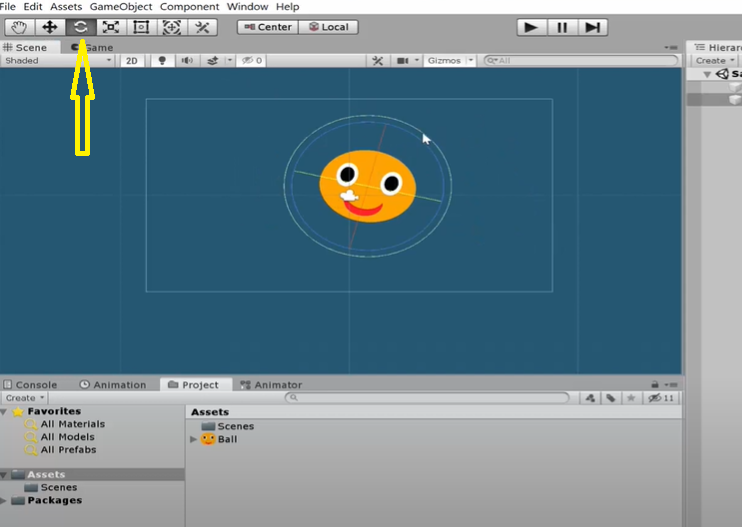
1. Toolbar

Toolbar chứa 5 loại điều khiển cơ bản, mỗi loại giữ một vai trò quan trọng trong Editor.

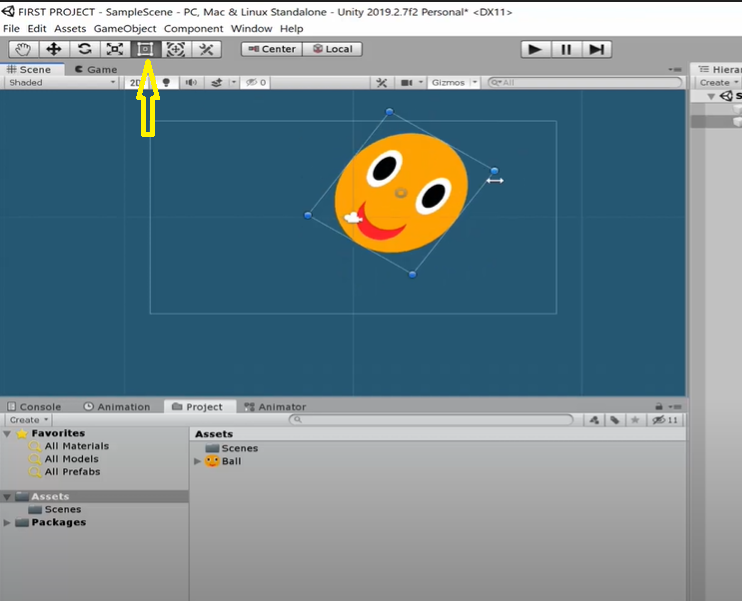
* Transform Tool: được dùng với Scene view, như quay trái, phải, lên, xuống, phóng to, thu nhỏ đối tượng.
* Transform Toggle: dùng cho việc thể hiện Scene view.
* Play/Pause/Stop Buttons: dùng cho Game view, chạy game trong Editor để kiểm tra.
* Layer Dropdown kiểm soát đối tượng nào đang được thực hiện trong Scene view.
* Layout Dropdown kiểm soát sự sắp xếp giữa các View.



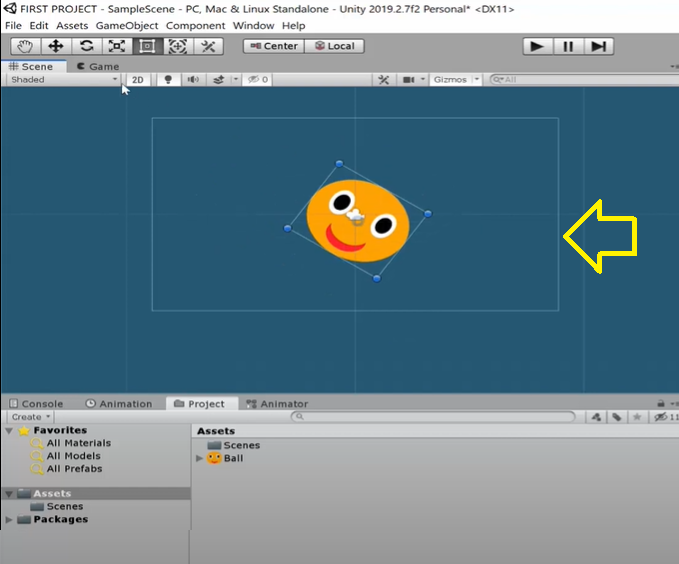
*Hình 24. Phóng to Object.*



*Hình 25. Xoay Object.*



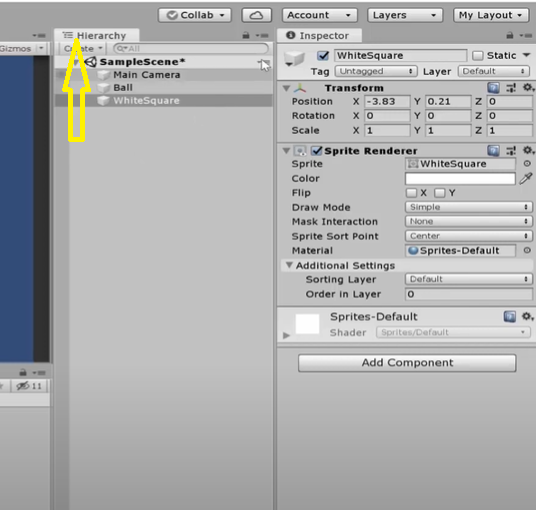
*Hình 26. Thay đổi kích thước Object.*



*Hình 27. Phạm vi hiển thị của Object.*

1. Hierarchy

Trong Hierarchy chứa các GameObject hiện thời, một số có thể trực tiếp tới những file Assets như 3D models, một số khác đại diện cho Prefabs - những đối tượng đã được tùy biến, dùng làm các công việc khác nhau trong Game. Chúng ta có thể chọn và parenting Object trong Hierarchy. Một Object có thể được thêm vào hay loại bỏ trong Scene và có thể thấy nó mất đi hay xuất hiện trong Hierarchy.

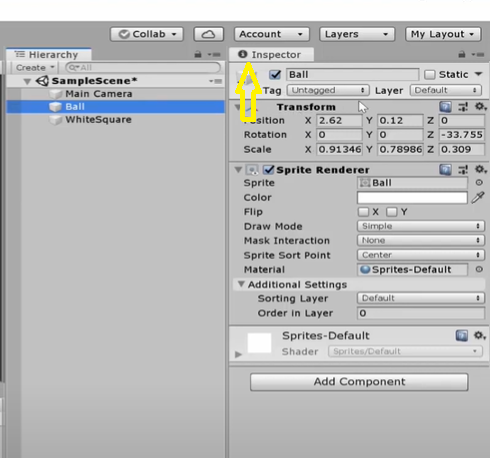


*Hình 28. Cửa sổ Hierarchy.*

1. Inspector

Games trong Unity được tạo ra bởi tập hợp rất nhiều Game Object, trong đó bao gồm meshes, script, âm thanh, hay những đối tượng Graphic như nguồn sáng v..v… Inspector sẽ hiển thị mọi thông tin về đối tượng đang làm việc một cách chi tiết, kể cả những Components được đính kèm và những thuộc tính của nó. Tại đây ta có thể điều chỉnh, thiết lập mọi thông số chức năng của những mối liên kết GameObject - Component.

Mọi thuộc tính thể hiện trong Inspector đều có thể được tùy biến một cách trực tiếp. Ngay cả với những biến trong script cũng có thể được hiệu chỉnh mà không cần xem mã. Trong script cũng có thể được hiệu chỉnh mà không cần xem mã. Trong script, nếu chúng ta định nghĩa một giá trị là public cho một kiểu đối tượng như GameObject, ta có thể drag-drop một GameObject hay một Prefabs vào trong Inspector để gán giá trị cho nó.



*Hình 29. Cửa sổ Inspector.*

**4.2 Các phần mềm hỗ trợ khác**

1. Photoshop

Adobe Photoshop là phần mềm đồ họa có tác dụng chính là chỉnh sửa ảnh. Được phát triển và phát hành bởi Adobe năm 1988. Hiện nay Ps được xem là phần mềm đồ họa bitmap mạnh nhất trên thị trường, và được hầu hết những người làm thiết kế đồ họa cũng như các photographer sử dụng. Photoshop cũng được đưa vào giáo trình giảng dạy chính thức của các trường đào tạo nghề cũng như các trường đại học.

Photoshop giúp tạo ra các bản vẽ nhân vật, cắt ghép các chi tiết, tạo chữ,...

1. After Effects

Adobe After Effect là phần mềm đồ họa tổng hợp được phát triển bởi hãng Adobe Systems. Phần mềm này được ứng dụng nhiều trong đồ họa động và kỹ xảo chuyên nghiệp. Chức năng chính của After Effects là tạo ra được những chuyển động đồ họa ấn tượng, đẹp mắt. Đây là một công cụ tuyệt vời cho ngành thiết kế đồ họa, kỹ xảo trong quảng cáo hay game hoặc truyền hình. After Effects tương thích ngược với nhiều phần mềm khác của Adobe vì vậy ta có thể lấy sản phẩm tạo ra từ Photoshop để xử lý tiếp, hoàn thiện hiệu ứng nhân vật trong game.

1. Visual Studio

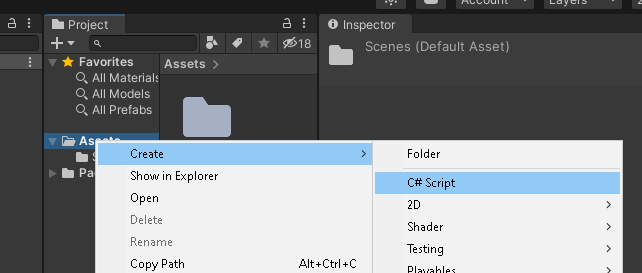
Visual Studio là (*IDE - Integrated Development Environment*) một bộ công cụ phát triển phần mềm do Microsoft phát triển. Visual Studio cũng là một tool được sử dụng bởi các lập trình viên để xây dựng nên các sản phẩm phần mềm.



*Hình 30. Visual Studio*

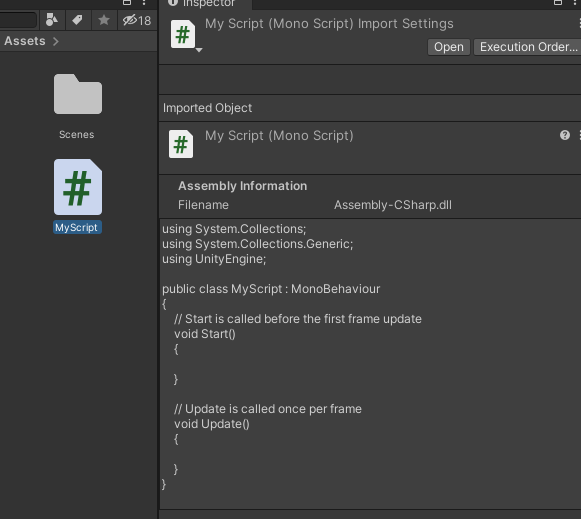
**4.3 Sử dụng C# trong Unity**

Như đã trình bày ở trên, Unity sử dụng mã nguồn C# để thực thi các hành động logic trong game. Để tạo một file script C# trong Unity, chúng ta Click chuột phải vào Project, chọn Create:



*Hình 31.Tạo một file Script C#.*

Thao tác này sẽ tạo một tệp C# mới trong thư mục Assets. Chúng ta sẽ đi vào tổ chức cấu trúc các tệp này theo một logic sau, hiện tại chúng ta sẽ chỉ để nó trong thư mục Assets.



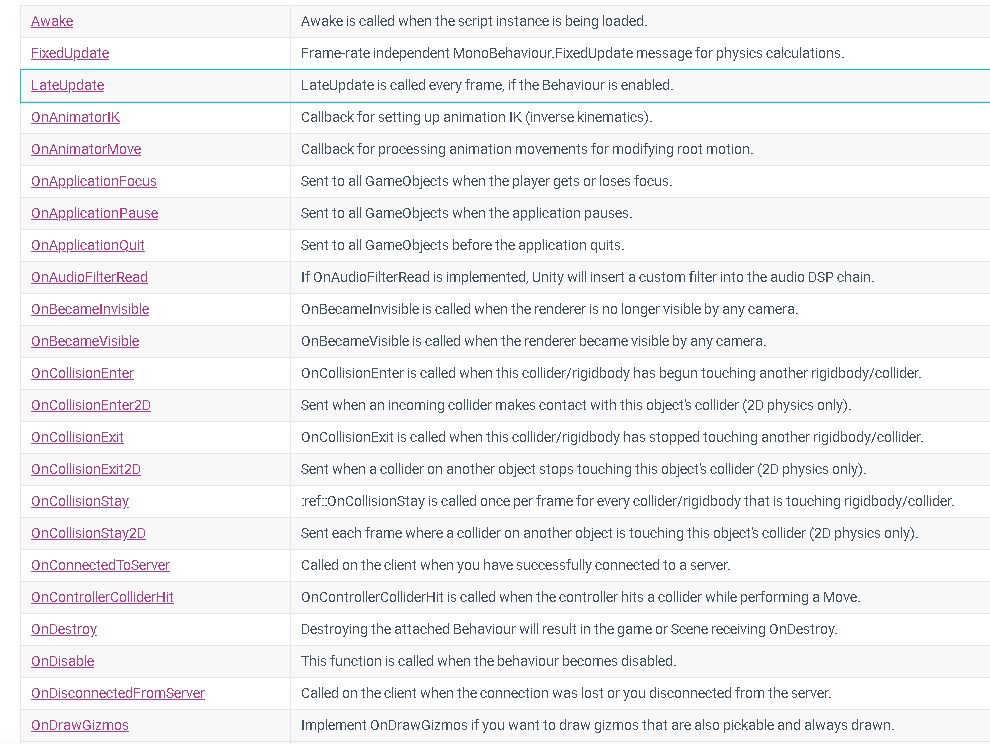
*Hình 32*

Click đúp vào file vừa tạo để mở Visual Studio, nơi mà chúng ta sẽ soạn thảo mã nguồn C#. Visual Studio là một IDE phổ biến có nhiều tính năng hiện đại như tự động hoàn mã nguồn, giúp tô sáng cú pháp và quan trọng nhất là trình gỡ lỗi.

**4.3.1 MonoBehaviour**

Trong C#, tập lệnh thao tác các biến bằng cách sử dụng các hàm (function). Mặc định, khi ta tạo một file script C#, phần Class sẽ trùng tên với tên file và kế thừa lớp MonoBehaviour, là một lớp cơ sở mà mọi tập lệnh Unity đều bắt nguồn từ đó, khi chúng ta sử dụng C#, chúng ta bắt buộc phải kế thừa từ lớp MonoBehaviour.

Lớp MonoBehaviour hỗ trợ những hàm riêng cho việc lập trình game Unity.

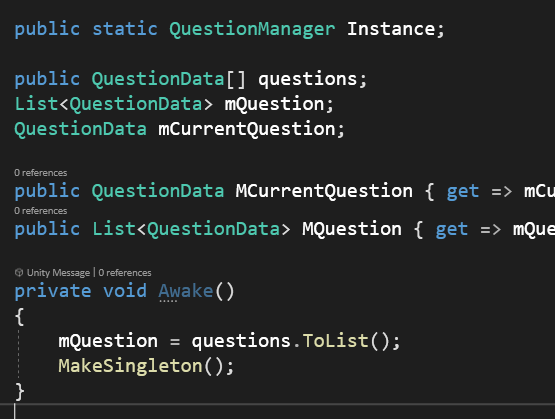


*Hình 33.Function MonoBehaviour support.*

1. **Awake**

Awake được gọi khi một GameObject đang hoạt động có chứa tập lệnh được khởi tạo khi một Scene được tải, khi một GameObject không hoạt động trước đó được đặt thành hoạt động hoặc sau khi một GameObject được tạo bằng Object.Instantiate được khởi tạo. Sử dụng Awake để khởi tạo các biến hoặc trạng thái trước khi ứng dụng khởi động.

Unity gọi Awake chỉ một lần trong suốt thời gian chương trình, các mã script được thực thi. Thời gian tồn tại của một tập lệnh kéo dài cho đến khi Scene chứa nó được ngừng thực thi. Nếu Scene được tải lại, Unity sẽ tải lại phiên bản script, do đó, Awake sẽ được gọi lại. Nếu Scene được tải cộng dồn nhiều lần, Unity sẽ tải Awake tùy theo số lần cộng dồn. Chúng ta nên sử dụng Awake để thiết lập tham chiếu giữa các tập lệnh và sử dụng Start, được gọi sau khi tất cả các lệnh gọi Awake kết thúc, để chuyển qua lại bất kỳ thông tin nào.

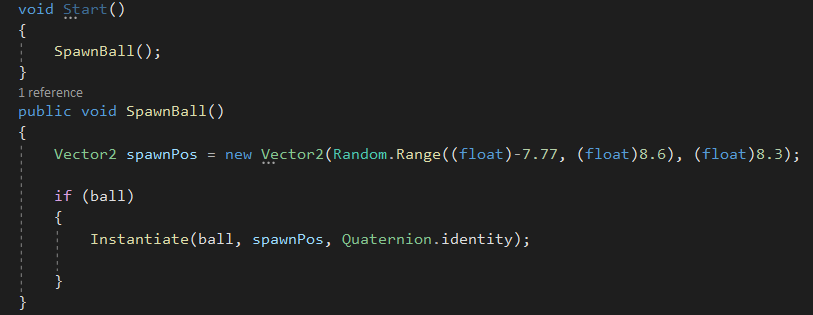


*Hình 34. Hàm Awake.*

1. **Start**

Tương tự Awake, hàm Start sẽ được gọi chính xác một lần trong thời gian tồn tại của script. Tuy nhiên,Awake được gọi khi đối tượng script được khởi tạo bất kể script có được kích hoạt hay không. Start có thể không được gọi trên cùng một khung (frame) với Awake nếu tập lệnh không được kích hoạt tại thời điểm tạo.

Hàm Awake được gọi trên tất cả các đối tượng trong Scene trước khi hàm Start của bất kỳ đối tượng nào được gọi. Thực tế này hữu ích trong trường hợp mã khởi tạo của một đối tượng A cần dựa vào đối tượng B đã được khởi tạo. Khi đó, hàm khởi tạo của B nên được thực hiện trong Awake, trong khi A nên được thực hiện trong Start. Ngoài ra, khi các đối tượng được khởi tạo trong quá trình trò chơi được chạy, Awake được gọi sau khi các hàm Start của các đối tượng trên Scene đã xong.



*Hình 34. Hàm Start.*

*Ví dụ hình 34, ta tạo một trò chơi hứng bóng, khởi tạo hàm SpawnBall() để sinh bóng ngẫu nhiên trong tọa độ random cho trước. Khi bắt đầu game, những quả bóng sẽ tự động rơi xuống trong tọa độ định sẵn.*

1. **OnEnable**

Awake và Start chỉ được gọi một lần duy nhất khi GameObject được tạo ra, nhưng với OnEnable thì lại khác.

OnEnable được gọi bất cứ khi nào GameObject chuyển từ trạng thái deactive (ngừng hoạt động) sang trạng thái active (hoạt động) hoặc thay đổi giá trị của enable từ false sang true. Do đó, chúng ta hầu như không dùng đến OnEnable đối với gameobject. Thay vào đó, đối với các object trong Canvas (UI) *phải deactive và active nhiều thì OnEnable mới thực sự hữu ích.*

*Thứ tự thực thi của ba hàm thực thi (a, b, c) khi GameObject được tạo ra là:*

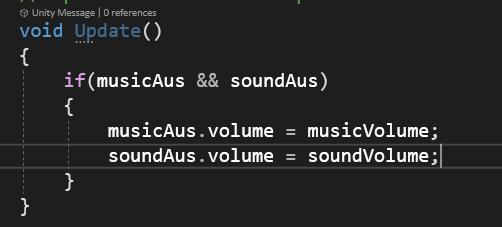
*Awake => OnEnable => Start*

*Nếu ta sử dụng Awake với quá nhiều GameObject cùng khởi tạo một lúc, thì với CPU yếu rất dễ dẫn đến việc bị nghẽn và bị bỏ qua. Do vậy, Start được cho là phương án an toàn trong việc xử lý các tác vụ của GameObject khi mới được khởi tạo.*

1. **Update**

Hàm Update được gọi một lần cho mỗi khung. Đây là nơi chúng ta đặt mã script để xác định logic chạy liên tục như hoạt ảnh, âm thanh, AI và các thành phần khác của trò chơi phải được cập nhật liên tục. Như việc chúng ta viết một hàm vào trong hàm Update, game đang chạy ở mức 60fps (frame per second) tức là một giây hàm của chúng ta sẽ được thực hiện 60 lần. Điều này ảnh hưởng rất nhiều tới performance. Do vậy, chúng ta chỉ sử dụng hàm Update khi thực sự cần thiết như những thao tác của người chơi đòi hỏi phải được xử lý liên tục.

Ví dụ bên dưới cho ta thấy ứng dụng của hàm Update khi phát nhạc nền trong game.



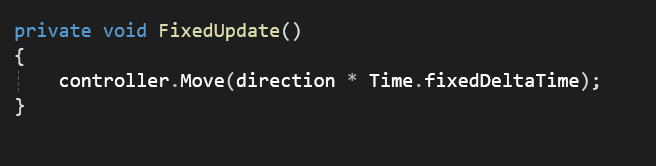
*Hình 36. Hàm Update.*

1. **FixedUpdate**

Như chúng ta đã biết, các đại lượng vật lý thường phụ thuộc vào thời gian. Việc mô phỏng vật lý trong Unity cũng vậy, nó phụ thuộc vào thời gian chứ không phải fps (frame per second). FixedUpdate thường dùng để xử lý các tác vụ liên quan đến vật lý trong game, thường là game 3D.

Đó cũng là điểm khác biệt giữa Update và FixedUpdate được gọi sau mỗi khoảng fixedDeltaTime mặc định trong Unity là 0.02 giây, tức là một giây được gọi 50 lần. Vì vậy, dù game có chạy ở 60fps hay 120fps thì số lần gọi FixedUpdate cũng không khác nhau.

Do đó, nếu GameObject đang được mô phỏng vật lý (có gán rigidbody), việc di chuyển GameObject đó nên được thực hiện trong FixedUpdate bằng AddForce hoặc MovePosition.



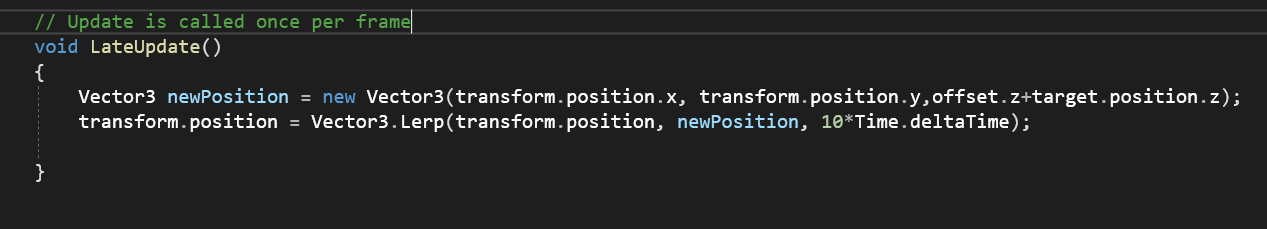
*Hình 37. Hàm FixedUpdate.*

1. **LateUpdate**

LateUpdate cũng giống với Update, đều được gọi mỗi frame (khung). Điểm khác nhau là LateUpdate được gọi sau khi Update được gọi. Vì vậy, các lưu ý khi dùng LateUpdate cũng giống như khi dùng Update.

Vậy ta cần LateUpdate làm gì khi đã có Update?

Ví dụ có một gameobject di chuyển trong hàm Update. Nếu ta muốn Camera của mình di chuyển theo gameobject đó thì mình sẽ viết hàm di chuyển của Camera vào trong LateUpdate. Đó cũng là cách di chuyển Camera của một vật mượt mà nhất.



*Hình 38. LateUpdate*

**4.3.2. Xử lý va chạm**

Va chạm và xử lý va chạm là những thành phần không thể thiếu khi lập trình game. Va chạm trong game là xảy ra khi chúng ta có 2 object đi vào không gian của nhau. Ví dụ như : trúng đạn, trúng bom, chạm phải quái vật, xuất phát, tới đích, trúng mũi tên ….. đều là các sự kiện va chạm và khi lập trình game chúng ta cần phải xử lý các va chạm đó. Chúng ta sẽ có 2 quá trình xử lý với va chạm như sau:

Quá trình 1 : phát hiện và thông báo sự kiện va chạm.

Quá trình 2 : xử lý sự kiện va chạm.

Trong Unity có 2 loại va chạm đó là :

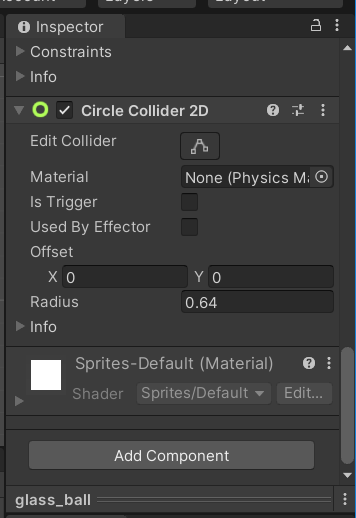
* Collision : là loại va chạm mà 2 đối tượng sẽ không đi xuyên qua nhau, khi đối tượng này gặp đối tượng kia thì sẽ bị cản lại, bật lại tùy theo tính chất vật lý mà chúng ta xét cho đối tượng. Ví dụ : Quả bóng rơi trên giá hứng bóng
* Trigger : là loại va chạm mà các đối tượng này có thể đi xuyên qua đối tượng kia, chúng ta sẽ sử dụng trigger trong các hoạt cảnh như làm cho tiếng nhạc bật lên khi đối tượng đi qua loa hay làm cho cây đổ khi người chơi đi tới, hay lửa …

Xem ảnh minh họa sau đây để hiểu rõ hơn (ví dụ Unity 2D):

* Đầu tiên: Tạo không gian Collider

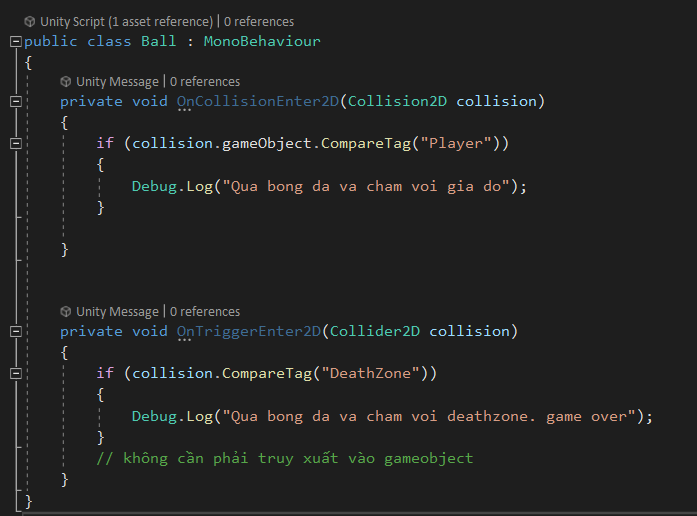
Chọn một trong các loại collider phù hợp dưới đây:

* Circle collider 2D: collider dạng hình tròn
* Box collider 2D: collider dạng hình vuông
* Edge collider 2D: collider dạng đoạn thẳng
* Polygon collider 2D: collider dạng đa giác tùy biến



*Hình 38. Chọn collider*

* Tiếp theo: Viết code xử lý va chạm



*Hình 39. Xử lý va chạm Unity 2D*

**4.3.3 UI Component**

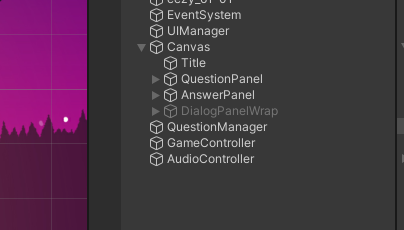
Giao diện người dùng (User Interface - UI) là thành phần không thể thiếu đối với bất kỳ game nào. UI cung cấp các thông tin trực quan cần thiết cho người chơi, giúp người chơi có cái nhìn toàn diện về các khả năng của mình (thời gian, điểm số,...) và có chiến thuật thích hợp để vượt qua được các thử thách trong game.

Thiết kế giao diện người dùng đơn giản là sử dụng các Assets có sẵn (hình ảnh, font chữ, các hiệu ứng,..), sắp xếp chúng theo một bố cục được Designer, Artist trong dự án thiết kế. Hầu hết các đối tượng UI đã được trừu tượng hóa thành các class và được đặt trong 1 namespace có tên là UnityEngine.UI.

Các thành phần cơ bản trong thiết kế UI bao gồm Canvas, Text, Image, Button,...

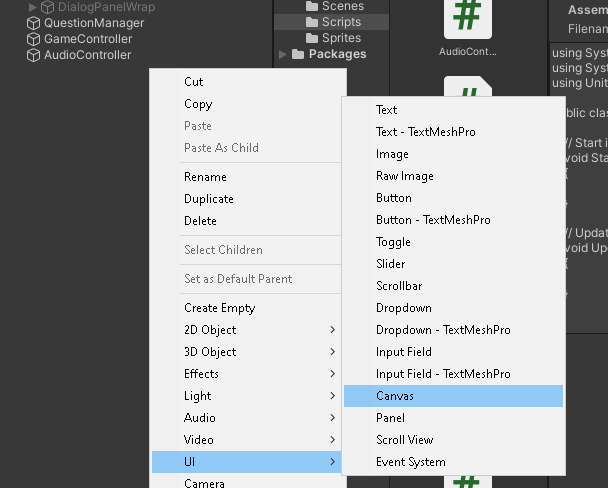
1. Canvas

Canvas là thành phần chính không thể thiếu trong thiết kế UI, các thành phần UI khác khi được khởi tạo bắt buộc phải nằm trong 1 Canvas. Khi khởi tạo một thành phần UI, Unity sẽ tự động tạo ra 1 Canvas nếu chưa tồn tại trong Scene



*Hình 40. Canvas trong Unity.*

Để khởi tạo 1 Canvas, trong cửa sổ Hierarchy, chọn Create - UI -Canvas. Các đối tượng UI khác cũng được khởi tạo tương tự.



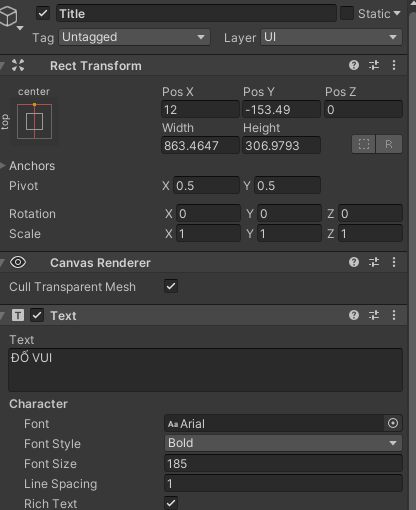
*Hình 41. Cách tạo Canvas.*

Các đối tượng con của một Canvas sẽ được render theo thứ tự từ trên xuống trong cửa sổ Hierarchy, đối tượng nào ở trên sẽ được render trước và có thể bị che khuất bởi đối tượng phía dưới.

1. Text

UI Text được sử dụng để hiển thị các thông tin trên màn hình như Score, Lives, Times,... một số game sử dụng các texture riêng để hiển thị thông tin thay cho text, tuy nhiên text vẫn là lựa chọn phổ biến hơn vì tính đơn giản và dễ thao tác.

Sau khi khởi tạo một đối tượng Text, cửa sổ Inspector có dạng như sau:



*Hình 42. Cửa sổ Text.*

Rect Transform: quản lý vị trí, kích thước, góc quay,... của Text.

Text lưu trữ chuỗi ký tự cần hiển thị ra màn hình. Unity hỗ trợ 1 số chức năng để tùy biến chuỗi như font, kiểu chữ, cỡ chữ, màu chữ,... Tùy chọn Best Fit sẽ tự động điều chỉnh kích thước font chữ phù hợp với kích thước được quy định trong Rect Transform.



*Hình 43. Ví dụ về UI Text.*

1. Image

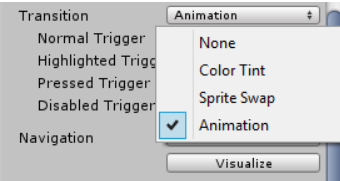
UI Image được sử dụng rất phổ biến, như thiết kế background, các button, title,...

Chọn hình ảnh và kéo thả vào khung Source Image, ngoài ra còn có một số tùy chọn thay đổi màu sắc, chất liệu,...

1. Button

UI Button là một thành phần quan trọng, giúp người chơi tương tác với game. Unity cung cấp khá nhiều tinh năng giúp hiển thị và thao tác với button được dễ dàng và đẹp hơn.

Về phần hiển thị, Unity hỗ trợ 4 loại hiển thị trong phần Transition:



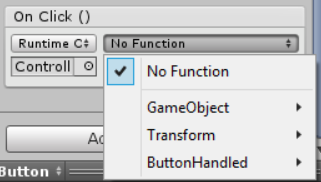
*Hình 44. Hiển thị Button*

None: sử dụng hình ảnh button được truyền vào thành phần image. Với tùy chọn này, hình hiển thị button sẽ không thay đổi mỗi khi người chơi thao tác.

Color Tint, Sprite Swap và Animation có các tùy chọn hiển thị riêng cho từng trạng thái của button. Với Color Tint, button sẽ thay đổi màu sắc theo từng trạng thái, Sprite Swap có hỗ trợ thay đổi sprite theo trạng thái, Animation hỗ trợ thêm các đặc tính như scale, transform,... Tùy vào yêu cầu của game mà lựa chọn tùy chọn phù hợp.

Về phần tương tác, Unity cung cấp sẵn event OnClick. Khi nhấn vào button, tất cả các hành động được thiết lập sẵn trong event OnClick sẽ được thực hiện, do đó có thể thực hiện đồng thời nhiều hành động tùy theo nhu cầu.

Các đối tượng được thêm vào có thể là một GameObject trong cửa sổ Hierarchy hoặc chính button đó. Khi đó event sẽ tự động nhận được các thành phần của GameObject và hiển thị trong menu như trong hình dưới.



*Hình 45. Sự kiện OnClick.*