ỦY BAN NHÂN DÂN TP. HỒ CHÍ MINH

#### **TRƯỜNG ĐẠI HỌC SÀI GÒN**

# TRẦN QUANG TÍN

**Tìm hiểu GameEngine Unity3D và ứng dụng phát triển một game thực tế**

###### **KHÓA LUẬN TỐT NGHIỆP**

**NGÀNH: CÔNG NGHỆ thÔNG tiN**

**tRÌNH ĐỘ ĐÀO TẠO: ĐẠI HỌC**

**TP. HỒ CHÍ MINH, THÁNG 12 NĂM 2017**

ỦY BAN NHÂN DÂN TP. HỒ CHÍ MINH

#### **TRƯỜNG ĐẠI HỌC SÀI GÒN**

# TRẦN QUANG TÍN

**Tìm hiểu GameEngine Unity3D và ứng dụng phát triển một game thực tế**

###### **KHÓA LUẬN TỐT NGHIỆP**

**NGÀNH: CÔNG NGHỆ THÔNG TIN**

**tRÌNH ĐỘ ĐÀO TẠO: ĐẠI HỌC**

NgƯỜI HƯỚNG DẪN: TS. NGUYỄN MINH THI

**TP. HỒ CHÍ MINH, THÁNG 12 NĂM 2017**

**LỜI CAM ĐOAN**

*Tôi xin cam đoan đây là công trình nghiên cứu của riêng tôi, các số liệu và kết quả nghiên cứu nêu trong luận văn là trung thực, được các đồng tác giả cho phép sử dụng và chưa từng được công bố trong bất kì một công trình nào khác.*

Tác giả luận văn

**Trần Quang Tín**

# LỜI CẢM ƠN

Để hoàn thành khóa luận tốt nghiệp này em xin chân thành cảm ơn quý thầy cô giáo trong khoa Công Nghệ Thông Tin trường Đại Học Sài Gòn đã tạo điều kiện để em học tập và thực hiện khóa luận.

Đặc biệt, em xin gửi đến thầy Nguyễn Minh Thi, người đã tận tình hướng dẫn, giúp đỡ em hoàn thành khóa luận tốt nghiệp này lời cảm ơn sâu sắc nhất.

Vì kiến thức bản thân còn hạn chế, trong quá trình thực hiện và bảo vệ khóa luận này em không tránh khỏi những sai sót, rất mong nhận được sự thông cảm cũng như những ý kiến đóng góp từ thầy cô.

# DANH MỤC CÁC TỪ VIẾT TẮT

|  |  |
| --- | --- |
| **UI** | **User Interface** |
| **IP** |  |
| **LAN** |  |
| **VR** |  |
| **AR** |  |
| **FBX** |  |
| **AI** |  |
| **FPS** |  |

# DANH MỤC CÁC BẢNG

# DANH MỤC CÁC SƠ ĐỒ



# MỤC LỤC

Contents

[TRẦN QUANG TÍN i](#_Toc500983002)

[TRẦN QUANG TÍN i](#_Toc500983003)

[LỜI CẢM ƠN iii](#_Toc500983004)

[DANH MỤC CÁC TỪ VIẾT TẮT iv](#_Toc500983005)

[DANH MỤC CÁC BẢNG v](#_Toc500983006)

[DANH MỤC CÁC SƠ ĐỒ vi](#_Toc500983007)

[**1** **CHƯƠNG 1: TỔNG QUAN VỀ GAME ENGINE VÀ UNITY3D** 4](#_Toc500983008)

[**1.1.** **Game Engine** 4](#_Toc500983009)

[**1.1.1.** **Khái niệm** 4](#_Toc500983010)

[**1.1.2.** **Xu hướng hiện nay** 4](#_Toc500983011)

[**1.2.** **Unity3D** 6](#_Toc500983012)

[**1.2.1.** **Tổng quan** 6](#_Toc500983013)

[**1.2.2.** **Lịch sử hình thành, phát triển và thành tựu đạt được** 6](#_Toc500983014)

[**1.2.3.** **Tại sao chọn Unity?** 8](#_Toc500983015)

[**1.2.4.** **Các sản phẩm nổi bật** 9](#_Toc500983016)

[**2** **CHƯƠNG 2: Chi tiết về Unity3D** 13](#_Toc500983017)

[**2.1.** **Các thành phần chính** 13](#_Toc500983018)

[**2.1.1.** **Các khái niệm cơ bản** 13](#_Toc500983019)

[**2.1.2.** **Physics** 15](#_Toc500983020)

[**2.1.3.** **UI** 18](#_Toc500983021)

[**2.1.4.** **Audio và video** 21](#_Toc500983022)

[**2.1.5.** **Animation** 23](#_Toc500983023)

[**2.1.6.** **Networking** 25](#_Toc500983024)

[**2.1.7.** **Scripting** 29](#_Toc500983025)

[**2.1.8.** **VirtualReality và AugmentedReality** 32](#_Toc500983026)

[**2.2.** **Giao diện người dùng** 35](#_Toc500983027)

[**2.2.1.** **Cửa sổ làm việc** 35](#_Toc500983028)

[**2.2.2.** **Cửa sổ Scene** 35](#_Toc500983029)

[**2.2.3.** **Cửa sổ Game** 36](#_Toc500983030)

[**2.2.4.** **Cửa sổ Project** 37](#_Toc500983031)

[**2.2.5.** **Cửa sổ Hierarachy** 38](#_Toc500983032)

[**2.2.6.** **Cửa sổ Inspector** 39](#_Toc500983033)

[**2.2.7.** **Cửa sổ AssetStore** 40](#_Toc500983034)

# DANH MỤC CÁC TỪ VIẾT TẮT

|  |  |  |
| --- | --- | --- |
|  | : |  |
|  | : |  |

# DANH MỤC CÁC SƠ ĐỒ

|  |  |  |
| --- | --- | --- |
| **TT** | **Tên bảng** | **Trang** |
|  | Bảng 2.1. Dân số trung bình qua các năm của đồng bằng sông Cửu Long | 16 |
|  |  |  |

**MỞ ĐẦU**

1. **CHƯƠNG 1: TỔNG QUAN VỀ GAME ENGINE VÀ UNITY3D**
   1. **Game Engine**
      1. **Khái niệm**

Một GameEngine là một phần mềm được viết để thiết kế và phát triển video game, hiểu đơn giản nó là loại phần mềm trung gian kết nối tương tác của nhiều ứng dụng trong cùng 1 hệ thống với nhau. Có rất nhiều loại GameEngine dùng để thiết kế game cho các hệ máy như hệ consoles(Xbox, PlayStation) hay máy tính cá nhân(PC). Chức năng cốt lõi của GameEngine phần lớn nằm trong công cụ dựng hình(kết xuất đồ họa) cho các hình ảnh 2 chiều(2D) hay 3 chiều(3D), công cụ vật lý (hay công cụ tính toán và phát hiện va chạm), âm thanh, mã nguồn, hình ảnh động(animation), dựng ảnh... Quá trình phát triển game tiết kiệm được rất nhiều thời gian và kinh phí vào việc tái sử dụng và tái thích ứng những tài nguyên trong cùng một GameEngine để tạo nhiều game khác nhau.

Chúng ta cũng cần chú ý, GameEngine không chỉ bao gồm thành phần tạo hình ảnh mà bao gồm rất nhiều thành phần khác như trí tuệ nhân tạo, phân luồng, quản lý bộ nhớ, và kết nối mạng… Nếu Engine chỉ có chức năng render hình ảnh thì thường được gọi là Graphic Engine.

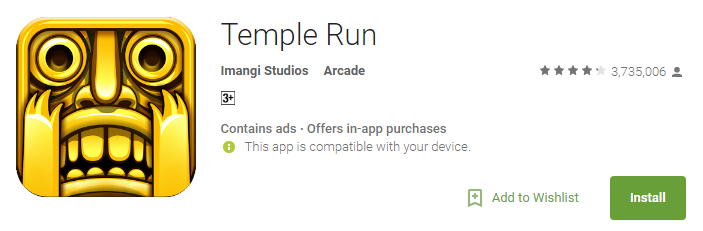
* + 1. **Xu hướng hiện nay**

Việc ngày càng có nhiều game engine mới xuất hiện và những game engine cũ được cải tiến thì cuộc cạnh tranh về thị phần nhà phát triển ngày càng gay gắt. Các game engine phải không ngừng tối ưu, cho ra những chức năng mới tiện dụng để hỗ trợ các lập trình viên. Thông thường khi khi chọn lựa một game engine để xây dựng sản phẩm, các nhà phát triển sẽ dựa vào những tiêu chí chủ yếu sau:

* **Giá thành:** Một số Game Engine có giá thành cao đi kèm theo đó có rất nhiều chức năng nâng cao và khả năng hỗ trợ tốt. Tuy vậy cũng có những game engine miễn phí nhưng hỗ trợ gần như đầy đủ các tính năng cơ bản. Do vậy chúng ta cần xem xét đến vấn đề này trước tiên. Nếu là người lập trình thông thường thì với mức giá khoảng 1500$(mức giá của Unity3D) cho 1 Engine rõ ràng là một vấn đề lớn đối với những nhà phát triển nhỏ lẻ.
* **Hiệu năng:** Trước khi chúng ta lựa chọn một Engine, chúng ta cần kiểm tra hiệu năng của Engine này trên nền tảng mà chúng ta hướng đến. Ví dụ: Một engine có hiệu năng cao trên PC nhưng lại cho hiệu năng thấp hoặc không hỗ trợ đầy đủ trên Android. Hoặc 1 Engine chỉ cho 1-2 Frame Per Second(FPS) trong khi 1 Engine khác cho 30 FPS với cùng một cách lập trình.
* **Cộng đồng:** Số lượng người sử dụng rõ ràng nói lên chất lượng của Engine đó. Đồng thời, nhiều người sử dụng và nhiều lập trình viên có nghĩa là khi bạn vướng mắc một vấn đề nào đó, khi đưa câu hỏi lên, sẽ nhanh có câu trả lời hơn, và câu trả lời chất lượng hơn.
* **Tài liệu:** Nếu một Engine không có tài liệu cụ thể, rõ ràng đó là một engine không tốt. Tài liệu cụ thể sẽ giúp bạn rất nhiều khi mới làm quen với Engine, cũng như giải quyết các vấn đề về thiết kế cũng như các vấn đề xảy ra trong khi lập trình.
* **Mã nguồn:** đôi khi các tài liệu không có đủ thông tin, chúng ta cần tìm hiểu mã nguồn để hiểu cách xử lý của Engine. Một số Engine mặc dù miễn phí nhưng lại có mã nguồn đóng. Do đó chúng ta cần có sự lựa chọn. Hoặc đôi khi mã nguồn của Engine không thực sự phù hợp với yêu cầu của chúng ta, khi đó chúng ta cần sửa đổi mã nguồn của Engine đó. Mã nguồn mở cũng giúp cho việc sửa lỗi nhanh hơn do có nhiều người có thể đóng góp hơn.
* **Các tính năng:** Rõ ràng mục đích của chúng ta khi sử dụng Engine là tiết kiệm thời gian phát triển và dễ dàng hơn trong quá trình phát triển. Do đó một Engine có nhiều tính năng hơn sẽ giúp chúng ta nhiều hơn, phù hợp hơn trong nhiều ứng dụng.
* **Store:** việc phát triển từ những gì có sẵn bao giờ cũng ít tốn thời gian và công sức hơn làm tất cả từ đầu. Vậy nên việc có một cộng đồng trao đổi và mua bán asset lớn mạnh sẽ giúp cải thiện đáng kể thời gian phát triển sản phẩm.
  1. **Unity3D**
     1. **Tổng quan**

Unity3D là một engine game đa nền tảng được phát triển bởi Unity Technologies bằng ngôn ngữ C/C++, hỗ trợ mã viết mã bằng C#, JavaScript hoặc Boo. Đây là công cụ phát triển game được cộng đồng làm game ưa thích sử dụng. Hiểu đơn giản Unity 3D Game là một công cụ trung gian tương tác giữa thiết kế và lập trình để phát triển game, gắn kết với nhiều ứng dụng với nhau trong cùng 1 hệ thống. Unity 3D hỗ trợ tốt cho game phát triển trên nhiều nền tảng khác nhau, có thể hoạt động trên nhiều nhóm thiết bị, từ trình duyệt web trên PC cho tới điện thoại hay máy tính bảng.

* + 1. **Lịch sử hình thành, phát triển và thành tựu đạt được**
* **Lịch sử hình thành và phát triển**
* Vào đầu những năm 2000, ba lập trình viên trẻ David Helgason (CEO), Nicholas Francis (CCO), and Joachim Ante (CTO) với nguồn kinh tế eo hẹp đã tập trung tại một tầng hầm ở Copenhagen và bắt đầu lập trình ra những thứ cơ bản nhất mà sau này trở thành một trong những phần mềm được ứng dụng rộng rãi nhất trong ngành công nghiệp video game.
* Một phiên bản đơn giản của Unity3D đã được phát hành vào năm 2005. Đội đã có thể hỗ trợ cho máy tính Windows và các trình duyệt Web. Đến năm 2008, Unity3D đã trở nên phức tạp hơn và doanh số bán phần mềm đã bắt đầu chi trả được cho các chi phí hoạt động và cho phép đội phát triển Unity mở rộng đến hàng chục nhân viên.
* Một bước ngoặt đã đến vào giữa năm 2008 khi Apple công bố AppStore. "Chúng tôi đã gấp rút xử lý để hỗ trợ iPhone và trở thành game engine đầu tiên làm được điều đó vào cuối năm 2008" - Helgason nói - "Nó xảy ra thực sự nhanh chóng. Đột nhiên, có rất nhiều người muốn có Unity3D."
* Một bước tiến lớn nữa đã đến vào năm 2008 là khi Cartoon Network sử dụng Unity3D để tạo ra FusionFall, một MMORPG(Massively Multiplayer Online Role-Playing Game) cho trẻ em đã có được 8 triệu người chơi.
* Electronic Arts sử dụng Unity3D trong năm 2009 để làm Tiger Woods PGA Tour Online, và thậm chí cả Microsoft và Ubisoft đã trở thành khách hàng của Unity.
* Trong năm 2011, Unity đã mua một công ty hoạt hình có tên Mecanim để thúc đẩy công nghệ cơ bản của game engine. Unity đã mua lại Applifier, một nhà cung cấp dịch vụ di động dựa vào Helsinki vào tháng 3 năm 2014. Hai thương vụ mua lại tiếp theo vào cuối năm 2014: Playnomics, một nền tảng phân tích dữ liệu cho các nhà phát triển (nay là Unity Analytics) và Tsugi, người đã từng làm việc cho một dịch vụ tích hợp liên tục, bây giờ được gọi là Unity Cloud Build.
* **Những thành tựu đạt được**
* Vào tháng 4 năm 2012, Unity cho biết có 1 triệu nhà phát triển đã đăng ký, 300.000 trong số đó đã sử dụng Unity hàng tháng. Vào tháng 4 năm 2015, số lượng các nhà phát triển đã đăng ký báo cáo đạt 4,5 triệu, với 1 triệu người dùng hoạt động hàng tháng. 47% các nhà phát triển trò chơi trên điện thoại di động sử dụng Unity3D.
* Vào ngày 10 tháng 11 năm 2010, Unity Asset Store được đưa ra như một thị trường trực tuyến cho người dùng Unity bán các asset của dự án (ảnh nghệ thuật, các đoạn mã, âm thanh, v.v.) cho nhau. Đến tháng 4 năm 2014, đã có 600.000 người dùng đăng ký tải xuống khoảng 500.000 asset mỗi tháng.
* Tại hội nghị VR/AR với mục đích "mở rộng cơ sở tri thức của bất kỳ ai phát triển thực tế ảo và thực tế tăng cường" được Unity tổ chức vào tháng 2/2016, công ty đã thông báo hợp tác với Steam VR và Google Cardboard và thành lập bộ phận VR/AR của công ty là Unity Labs.
* Từ đó đến nay Unity3D vẫn đang dẫn đầu về công nghệ VR và chiếm một thị phần rất lớn trên hai thiết bị phổ thông nhất hiện nay: 90% đối với Samsung Gear VR và 53% đối với Oculus Rift. Còn về AR thì Unity3D vẫn là lời khuyên số một để phát triển các ứng dụng cho Microsoft Hololens, tích hợp Vuforia(một trong những tool phát triển VR nổi tiếng nhất hiện nay), hỗ trợ các nền tảng mới ARKit(Apple) và ARCore(Google) ngay khi vừa được ra mắt.
* Đến nay Unity3D đã hỗ trợ trên 27 nền tảng bao gồm: Android, Android TV, Facebook Gameroom, Fire OS, Gear VR, Google Cardboard, Google Daydream, HTC Vive, iOS, Linux, macOS, Microsoft HoloLens, Nintendo 3DS family, Nintendo Switch, Oculus Rift, PlayStation 4, PlayStation Vita, PlayStation VR, Samsung Smart TV, Tizen, tvOS, WebGL, Wii U, Windows, Windows Phone, Windows Store, và Xbox One
  + 1. **Tại sao chọn Unity?**
* **Chi phí thấp:** Theo chính sách của Unity Technology, những cá nhân và doanh nghiệp có doanh thu dưới 100.000 USD/năm được dùng miễn phí engine Unity3D, và chỉ thu phí 1.500 USD/năm cho bản Profestional, một con số rất khiêm tốn so với những gì engine này mang lại.
* **Cộng đồng rất lớn mạnh:** Là engine phổ biến nhất trên thế giới nên Unity có cộng đồng sử dụng rất lớn mạnh. Mọi thắc mắc của bạn về Unity đều sẽ được thảo luận và trả lời trên website cộng đồng http://answers.unity3d.com. Các tân binh sẽ dễ dàng kế thừa, phát triển các scripts và kho đồ họa có sẵn để giúp cho sản phẩm phong phú hơn.
* **Dễ dàng tiếp cận:** theo đánh giá của các lập trình viên thì editor của unity là một điểm tích cực, giao diện cực kì trực quan và dễ sử dụng. Bên cạnh đó là cách kết hợp các component và script giúp lập trình viên dễ dàng hiểu được cách vận hành của hệ thống.
* **Dễ dàng học hỏi:** tài liệu để học trên trang web của unity được viết một cách rất bài bản từ mô tả cho đến các thuộc tính, đi kèm với đó là các ví dụ minh họa về sử dụng scripts hay các hướng dẫn thực hiện theo từng bước có cả ảnh chụp màn hình để minh họa. Ngoài ra còn có các chuỗi bài học với nhiều cấp độ từ những người mới bắt đầu cho đến những nhà phát triển đã có kinh nghiệm.
* **Cập nhật thường xuyên:** Unity 2017.2 là phiên bản mới nhất của được Unity giới thiệu vào tháng 10 năm 2017. Những tính năng ấn tượng được giới thiệu trong phiên bản mới này tạo thêm niềm tin tưởng vững chắc và sức mạnh khó vượt qua được từ Unity.
  + 1. **Các sản phẩm nổi bật**
* Temple Run

  
Hình 1.1: Temple Run trên CH Play

Đây là tựa game khá quen thuộc với người chơi Việt Nam. Được phát triển và ra mắt lần đầu năm 2011 và ngay lập tức trở thành một game hot thời điểm đó. Game được phát triển cho ba nền tảng di động là iOS, Android và WindowsPhone.

Một số hình ảnh về game:

  
Hình 1.2: Ảnh chụp game Temple Run

  
Hình 1.3: Ảnh chụp game Temple Run

* Ghost of a Tale: người chơi sẽ nhập vai một chú chuột mang tên là Tilo tham gia vào một cuộc phiêu lưu. Đây không phải là một game thành công hay nổi tiếng của Unity nhưng nó mang đến một trải nghiệm đồ họa khá đẹp mắt, thể hiện khả năng render không thua kém bất kì game engine nào nếu như được chau chuốt.

  
Hình 1.4: Ảnh chụp game Ghost of a Tale

  
Hình 1.5: Ảnh chụp game Ghost of a Tale

  
Hình 1.6: Ảnh chụp game Ghost of a Tale

1. **CHƯƠNG 2: Chi tiết về Unity3D**
   1. **Các thành phần chính**
      1. **Các khái niệm cơ bản**

* **Asset(tài sản)**: Assets là thư mục nơi chứa tất cả các file asset trong Game như: meshes, textures, videos, animations, sounds, music...... Các folder, file có thể tự sắp xếp theo cách của riêng của người dùng làm sao để dễ dàng quản lý nhất nên thông thường được đặt tên theo các chức năng hoặc ý nghĩa của chúng. Khi muốn đặt một mô hình 3D hay đơn giản chỉ là một tập tin cấu hình xml bên trong game, chúng ta sẽ cần phải khai báo(import) một cách đơn giản là kéo thả tập tin vào bên trong dự án để có thể sử dụng như là một asset của dự án. Sau khi được khai báo thì Unity sẽ tạo ra một file .meta có cùng tên với asset để quản lí. Các phương pháp trước đây chỉ cho phép nhập một file tại một thời điểm, trong khi sau này cho phép nhiều asset được gom lại thành một gói(package) và khai báo cùng một lúc. Các asset do chính những nhà phát triển game tạo ra và có thể được download miễn phí hoặc trả phí trên Unity Asset Store. Đây là một trong những tính năng rất hay của Unity. Các asset này sẽ giúp giảm thiểu rất nhiều thời gian cho việc thiết kế và lập trình game.

|  |
| --- |
| Học lập trình game cơ bản |
| Hình 2.1: Các loại file được Unity3D hỗ trợ |

* **Scene**: Trong Unity, một cảnh chơi (hoặc một phân đoạn) là những màn chơi riêng biệt, một khu vực trong game hoặc thành phần có trong nội dung của trò chơi (các menu). Các thành phần này được gọi là scene. Bằng cách tạo ra nhiều scene, chúng ta có thể phân phối thời gian và tối ưu tài nguyên, kiểm tra và xây dựng các phân đoạn trong game một cách độc lập.
* **Game Object:** Đây là một thuật ngữ thông dụng, đặc biệt trong mảng lập trình game. Mọi cảnh chơi, mọi kịch bản đều được xây dựng lên từ các gameobject và tương tác giữa chúng. Tất cả các gameobject đều chứa ít nhất một thành phần cơ bản là transform, lưu trữ thông tin về vị trí, góc xoay và tỉ lệ của gameobject. Thành phần transform này có thể được tuỳ biến và chỉnh sửa trong quá trình lập trình.
* **Component**: là các thành phần được đính vào trong gameobject, bổ sung tính năng cho các gameobject. Mỗi component có chức năng riêng biệt. Đa phần các component phụ thuộc vào một transform, vì nó lưu trữ các thông số cơ bản của gameobject. Bản chất của gameobject thật sự chỉ là một đơn vị để chứa các component, các đặc tính và khả năng của gameobject nằm hoàn toàn trong các component. Do đó chúng ta có thể xây dựng nên bất kỳ gameobject nào trong game mà chúng ta có thể tưởng tượng được bằng cách đính vào đó những component cần thiết.
* **Script**: script được Unity xem như một component. Đây là thành phần thiết yếu trong quá trình phát triển game. Bất kỳ một game nào, dù đơn giản nhất đều cần đến script để tương tác với các thao tác của người chơi, hoặc quản lý các sự kiện để thay đổi chiều hướng của game tương ứng với kịch bản game. Unity cung cấp cho lập trình viên khả năng viết script bằng ba ngôn ngữ: JavaScript, C#, Boo. Thông thường tất cả các hoạt động của gameobject hay sâu hơn là các component đều được điều khiển bởi script, điều này tạo nên một sự linh động rất lớn cho quá trình lập trình.
* **Prefab**: Prefab thực chất là một gameobject được lưu trữ lại để tái sử dụng. Các gameobject được nhân bản từ một prefab sẽ giống nhau hoàn toàn, ngoại trừ thành phần Transform và các tham chiếu ra bên ngoài prefab để phân biệt và quản lý được tốt hơn. Nhờ prefab ta có thể tận dụng tối đa những thành phần có thể tái sử dụng để giảm khối lượng công việc khi phát triển game.
  + 1. **Physics**

Để có được hành vi vật lý thuyết phục trong game, đối tượng trong trò chơi phải tăng tốc như thật và bị ảnh hưởng bởi va đập, lực hấp dẫn và các lực khác. Hệ vật lý tích hợp của Unity sẽ cung cấp một cách đầy đủ các thành phần xử lý mô phỏng vật lý. Chỉ cần một số thiết lập tham số có thể tạo ra các đối tượng hoạt động theo cách thực tế (nghĩa là chúng sẽ bị di chuyển bởi va chạm và rơi…). Bằng cách kiểm soát vật lý từ các script, bạn có thể cho một đối tượng động lực của một chiếc xe, một cái máy, hoặc thậm chí một miếng vải.

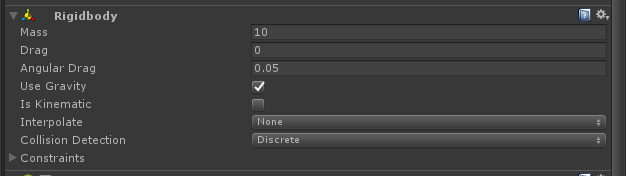
**Lưu ý:** có hai hệ vật lý riêng biệt trong Unity: một cho vật lý 3D, và một cho vật lý 2D. Các khái niệm chính giống hệt nhau giữa hai hệ vật lí nhưng chúng được thực hiện bằng các component khác nhau.

* + - 1. **Rigidbody**

Một Rigidbody là thành phần chính xử lí hành vi vật lý của một đối tượng. Với một Rigidbody đính kèm, đối tượng sẽ ngay lập tức đáp ứng những tính chất của trọng lực. Nếu một hoặc nhiều thành phần collider cũng được thêm vào, đối tượng sẽ phát hiện được va chạm và bị tác động từ lực của các va chạm.

Vì một thành phần của rigidbody chiếm quyền điều khiển transform đối tượng mà nó được gắn liền nên không thể di chuyển nó bằng cách thay đổi vị trí và độ xoay như thường lệ. Thay vào đó, nên áp dụng các lực vật lí để di chuyển đối tượng đó hoặc có thể tác động đến rigidbody để dễ kiểm soát hơn.

Trong một số trường hợp, nếu muốn một đối tượng có một rigidbody mà không muốn chuyển động của nó bị kiểm soát bởi các tác động vật lý. Ví dụ: muốn kiểm soát hành động vật lí của nhân vật trực tiếp từ script nhưng vẫn cho phép nó phát hiện được các va chạm của hệ vật lí. Kiểu chuyển động phi vật lý này được gọi là chuyển động kinematic. Thành phần rigidbody có một thuộc tính được gọi là IsKinematic loại bỏ nó khỏi sự kiểm soát của các tác động vật lý và cho phép nó được điều khiển chuyển động từ script. Có thể thay đổi giá trị kinematic từ script để cho phép vật lý được bật và tắt cho một đối tượng, nhưng điều này đi kèm với việc tiêu tốn hiệu năng xử lí nên cần cân nhắc trước khi được sử dụng.

  
Hình 2.2: Ảnh minh họa một thành phần rigidbody

* + - 1. **Collider**

Như đã đề cập ở trên, collider là thành phần đề phát hiện va chạm trong hệ vật lí của Unity. Vì vậy collider xác định hình dạng theo ý muốn của một vật thể với mục đích phát hiện va chạm vật lý. Một collider không nhìn thấy được và không nhất thiết phải có hình dạng tương tự như của vật thể trên thực tế, trong đa số trường hợp, chỉ cần một collider có hình dáng đơn giản và gần giống như vật thể là đảm bảo tính logic và hiệu suất của ứng dụng.

Một số collider thông dụng trong môi trường 3D như collider hình hộp 3D, hình cầu, hình trụ bo tròn. Trong 2D, có các loại như hình hộp 2D, hình tròn. Ngoài ra, trong 2D Unity cũng hỗ trợ tùy biến colider2D theo các hình dáng phức tạp. Đối với các mô hình 3D xuất ra từ các phần mềm đồ họa thì đã kèm theo meshcollider chính xác theo hình dáng của mô hình.

Colliders có thể được thêm vào một đối tượng mà không cần thành phần Rigidbody để tạo ra các thành phần bất động của game như tường, sàn... Nhìn chung, không nên định vị lại các collider tĩnh bằng cách thay đổi vị trí vì nó sẽ tác động mạnh mẽ đến hiệu năng của hệ vật lý vì nó không có rigidbody nên hệ vật lí không thể tính toán va chạm và các lực khác khi tác động.

  
Hình 2.3: Collider component và một số collider cơ bản

* + 1. **UI**

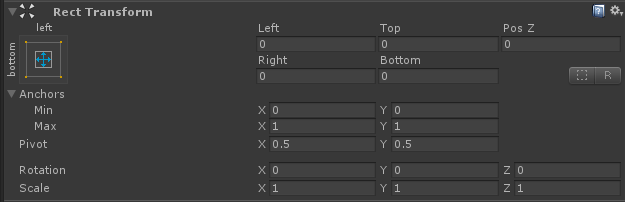
Mặc dù mang tên chính thức là Unity3D nhưng không vì thế mà unity bớt đi sự tập trung và trau chuốt cho mảng 2D của gameengine này. Bằng chứng là các thành phần để phát triển game 2D như physic2D và đa nền tảng không thua kém gì mảng 3D. Phần UI cũng không phải là ngoại lệ, UI của unity cung cấp đầy đủ các thành phần cần thiết để xây dựng giao diện người dùng. Bên cạnh đó là khả năng tùy biến vô giới hạn các thành phần, ví dụ như có thể biến một hình ảnh thành một nút bấm bằng cách thêm vào cho nó một event trigger component để tương tác với các sự kiện.

* + - 1. **RectTransform**

RectTransform là một component được dùng để thay thế cho thành phần transform trên tất cả các thành phần UI. RectTransform thể hiện một hình chữ nhật được xác định bởi chiều rộng và chiều cao liên quan đến một điểm tâm của nó (gọi là pivot).

Pivot là điểm trụ của gameobject, thông thường mặc định sẽ là ở tâm của hình chữ nhật, nếu xoay hoặc phóng to thu nhỏ RectTransform thì nó sẽ dựa trên điểm tâm này.

Thêm một thuật ngữ mà chúng ta cũng cần phải lưu ý là “neo”(anchor), một RectTransform có thể được neo tới đối tượng cha của nó nếu cha của nó cũng có một thành phần RectTransform. Anchor cho phép chúng ta di chuyển và điều chỉnh kích thước của thành phần mà nó đính kèm theo vị trí và kích thước của UI cha.

  
Hình 2.4: Một thành phần RectTransform

* + - 1. **Canvas và panel**

Canvas và panel là hai thành phần chính để mô hình tổng thể layout người dùng.

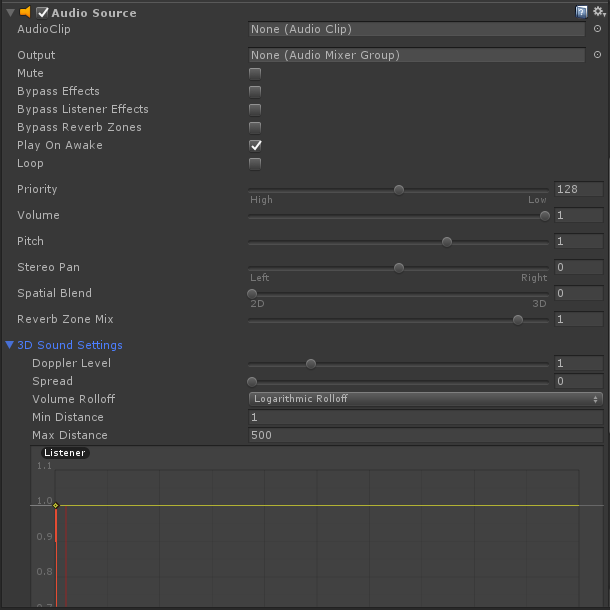
* **Canvas:** có thể hiểu là một vùng riêng cho phép chứa đựng thành phần giao diện người dùng (UI). Như vậy, khi tạo ra một UI mới, một Canvas cũng tự động được tạo ra, nếu đã có một đối tượng Canvas rồi thì UI mới được tạo ra là đối tượng con của Canvas. Một trong những mục đích quan trọng nhất của Canvas là để xử lý vấn đề tương thích đa màn hình, với thời đại smart phone hiện nay, có rất nhiều màn hình với tỷ lệ khác nhau, Canvas sẽ tự động scale canh chỉnh lại để duy trì UI của chúng ta được hiển thị tốt, không bị mất đi khi build lên những kích thước màn hình khác nhau.
* **Panel:** cơ bản panel là một thành phần để nhóm các thành phần UI liên quan lại với nhau sau đó mới canh chỉnh pannel này với canvas tổng để không bị vỡ khi chuyển đổi kích thướt màn hình.
  + - 1. **Các thành phần cơ bản khác**
* Button: button là một thành phần không thể thiếu đối với bất kì hệ UI nào, hầu như tất cả các giao diện người dùng đều có sự góp mặt của button. Unity cũng vậy, ngoài việc đem đến một sự tùy biến tuyệt vời Unity còn cung cấp một giao diện kéo thả để sử dụng các event mà không cần phải thông qua code.
* Toggle: bên cạnh button thì toggle cũng là một thành phần UI quan trọng, Unity cung cấp cho chúng ta một thành phần toggle với đầy đủ các chức năng, bao gồm cả toggle group…
* Image: đối với các tác vụ UI thì hiển thị hình ảnh là điều tất yếu cần được hỗ trợ. Unity cung cấp một component image mạnh mẽ với khả năng hiển thị nhiều loại hình ảnh với nhiều chất lượng khác nhau. Ngoài việc hiển thị một hình ảnh đã được khai báo thành asset, image của Unity còn có thể đọc và hiển thị ảnh từ bộ nhớ trong lúc thực thi chương trình.
* Scroll elements: nếu được đánh giá về các điểu mà unity làm chưa tốt trong phần UI thì những thành phần hỗ trợ xây dựng nên một scrollbar là ứng viên hàng đầu. Những component rời rạc cần ghép lại với nhau một cách hợp lí thì mới cho ra được một scrollbar hoạt động được. Điều này tốt với những lập trình viên đã có nhiều kinh nghiệm vì có thể làm bất kì scrollbar nào mà họ muốn. Nhưng đối với những người mới làm quen thì tìm hiểu để xây dựng nên một scrollbar thật là một điều khó khăn.
* Inputfield và Text: đây là hai thành phần cần thiết cuối cùng mà Unity cung cấp để xây dựng UI. Tuy đơn giản mà hiệu quả, ta có thể font chữ, canh chỉnh theo ý muốn, đặt một hình background yêu thích phía sau.

Mặc dù unity cung cấp một cách đầy đủ những thành phần cần thiết để xây dựng nên một UI hoàn chỉnh nhưng không thể so sánh được với các thành phần UI của những phần mềm chuyên nghiệp như VisualStudio, Netbean… Mặc dù vậy, điểm cộng lớn nhất của Unity là cung cấp cho các thành phần của họ một khả năng tùy biến cao, chỉ cần thêm vào các component thì nhà phát triển có thể xây dựng lên bất kì thành phần UI nào mà họ muốn. Nhìn chung, đối với UI của các game hiện nay thì như vậy là đủ để làm hài lòng đa phần lập trình viên cũng như người chơi.

* + 1. **Audio và video**
       1. **Audio**

Đối với bất kỳ game nào, âm thanh là một trong những yếu tố không thể thiếu. Âm thanh như một chất xúc tác nhằm kích thích các giác quan của người chơi giúp cho game trở nên thu hút và hấp dẫn hơn. Unity cung cấp một component gọi là AudioSource để làm việc này. Để thêm âm thanh cho game chỉ cần làm một việc đơn giản là thêm một component AudioSource và gán cho nó một AudioClip.

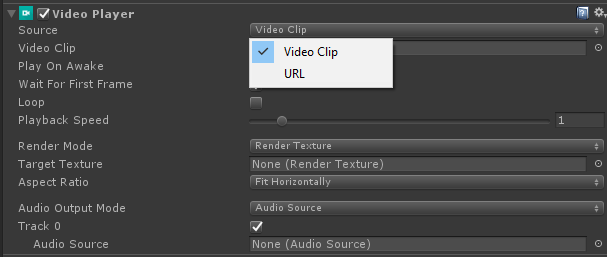
AudioClip là tên để chỉ asset âm thanh trong Unity, sau khi import các file âm thanh vào trong unity thì mỗi file được xem là một AudioClip. Unity cũng hỗ trợ gần như đầy đủ các định dạng thông dụng hiện nay.

  
Hình 2.5: Thành phần AudioSource trong Unity

* + - 1. **Video**

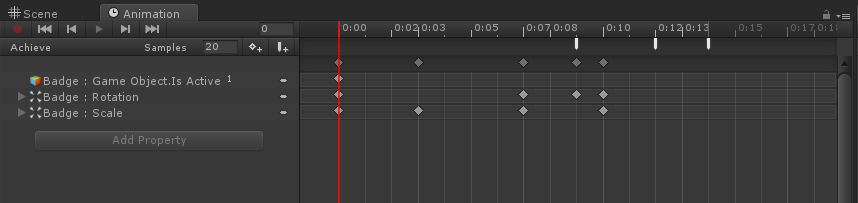
Đối với các nền tảng để phát triển ứng dụng thì khả năng chơi một video là tối cần thiết. Nhưng dường như Unity không đề cao sự cần thiết của việc phát một video trong game. Chính vì vậy nên cho tới trước khi phát hành Unity 5.6 vào ngày 31/3/2017 thì không có cách nào “chính thống” để có thể phát một video trong game của Unity kể cả trong mảng 2D. Cách duy nhất để phát video trong khoảng thời gian này về trước là nhúng một Asset của bên thứ ba(đa phần là phải trả phí) hoặc dùng plugin của QuickTime. Nhưng những cách này không mang lại sự ổn định cao cũng như phải cái đặt và sử dụng rất phức tạp. May mắn là sau khi nhận thấy nhu cầu thì Unity đã cung cấp một built-in component là VideoPlayer trong bản Unity 5.6. Điều này nhận được sự đồng thuận và ca ngợi của đa số nhà phát triển Unity và cũng là điểm nhấn đáng chú ý nhất của bản phát hành khi đó.

Về cách sử dụng thì VideoPlayer sử dụng không khác nhiều so với một AudioSource. Chỉ cần thêm một component VideoPlayer và gán cho nó một VideoClip là có thể sử dụng được. VideoClip có thể được khai báo trong phần Asset hoặc có thể là một URL dẫn đường đến file để game đọc lên trong khi thực thi. Mọi thao tác, tùy chỉnh liên quan đến phát video đều có thể điều khiển trong script nên dễ dàng quản lí trong khi khi thực thi.

  
Hình 2.6: Thành phần VideoPlayer trong Unity

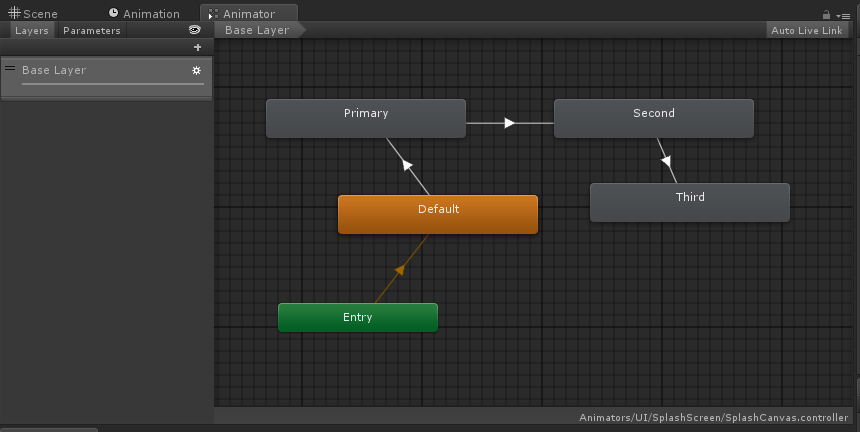
* + 1. **Animation**

Animation là một yếu tố không thể thiếu được trong bất kỳ game nào. Đối với ứng dụng thì animation mang lại sự sống động, không cứng nhắc như một bức hình. Animation giúp lập trình viên diễn tả hành động của các nhân vật một cách chân thực và sống động hơn. Với Unity, thao tác tạo và sử dụng animation cho các nhân vật trở nên dễ dàng và trực quan nhờ khả năng kéo thả, giúp lập trình viên không tốn quá nhiều thời gian để điều chỉnh và thiết lập các thông số.

  
Hình 2.7: Màn hình làm việc với Animation

Một animation trong Unity bao gồm hai thành phần chính:

* Clip: một chuyển động của nhân vật ví dụ như: chạy, nhảy, đứng yên …vv
* Animator: chứa các clip của nhân vật và quyết định khi nào thì thể hiện các clip này. Mỗi gameobject chỉ có thể có một Animator và chứ nhiều clip trong Animator đó. Mọi thao tác điều khiển animation từ trong script đều thông qua Animator này mà không gọi trực tiếp tới clip.

  
Hình 2.8: Một animator chứa những animation clip

Hiện tại, Unity chỉ hỗ trợ tạo Animation cho mảng 2D, đối với những mô hình 3D thì Unity có thể thể hiện Animation của mô hình đó nhưng không thể tạo ra nó. Animation cho mô hình 3D phải được tạo cùng mô hình đó trong các phần mềm vẽ 3D chuyên nghiệp và chuyển qua Unity để sử dụng.

Về kỹ thuật tạo Animation2D thì có hai kỹ thuật chính:

* Frames: kỹ thuật này dùng một chuỗi hình ảnh của nhân vật và thay thế liên tục để tạo cảm giác chuyển động. Ưu điểm của kỹ thuật này là dễ hiểu dễ thực hiện. Nhược điểm của kỹ thuật này là ngốn rất nhiều tài nguyên. Muốn chuyển động càng mượt thì cần phải có càng nhiều hình và thay thế chúng càng nhanh, điều này gâp áp lực lớn lên hiệu năng của game.
* Spine: kỹ thuật này tạo Animation dựa trên khung xương, mỗi khung xương sẽ có một hình ảnh tương ứng, muốn nhân vật chuyển động thì ta sẽ chuyển động khung xương và hình ảnh, nhân vật sẽ chuyển động theo. Kỹ thuật này khắc phục được tình trạng tiêu tốn tài nguyên của kỹ thuật Frames ở trên nhưng bù lại khó thực hiện hơn.

Đối với Unity thì có thể thực thi cả hai Animation được tạo theo hai kỹ thuật ở trên nhưng chỉ hỗ trợ tạo Animation cho kỹ thuật Frames. Nếu muốn tối ưu cho game thì nhà phát triển phải tạo Spine Animation bởi một phần mềm khác sau đó đưa vào Unity sử dụng.

* + 1. **Networking**

Networking là một phần không thể thiếu cho một game engine. Công nghệ ngày càng phát triển và con người không hề thỏa mãn khi chơi game một mình trên máy. Như có thể thấy mạng xã hội đã phát triển như thế nào khi kết nối con người với nhau và không game nào được tổ chức các cuộc thi mà lại chơi một mình sau đó tính điểm. Unity cũng ý thức được vấn đề này nên đã xây dựng nên một bộ networking rất đầy đủ để hỗ trợ những nhà phát triển trong việc mang game của mình lên mạng network hay xa hơn là thông qua internet.

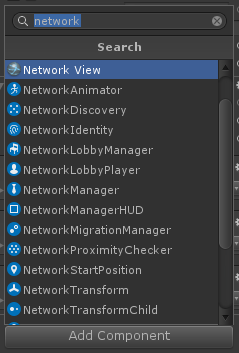
* + - 1. **Local network**

Về local network thì Unity dường như đã cung cấp đầy đủ một bộ component hỗ trợ xây dựng nên một game local với các tác vụ cơ bản. Local netwoking là một chủ đề mà Unity có hẳn một khóa hướng dẫn cho người vừa bắt đầu. Chỉ cần theo dõi những hướng dẫn từ khóa học miễn phí này là đã có thể nắm vững nền tảng và tạo ra một game multyplayer đơn giản.

* + - 1. **Internet**.

Về các game online thì về cơ bản cũng sử dụng các component làm nên local network game nhưng khác nhau ở phương thức tạo ra và liên kết những người chơi. Để sử dụng multyplayer cũng như những service khác của Unity thì nhà phát triển phải có tài khoản developper của Unity và đăng kí game cần sử dụng multyplayer bằng tài khoản của mình. Quá trình đăng kí khá đơn giản và có thể sử dụng ngay sau khi đăng kí. Tuy nhiên, Unity sẽ giời hạn băng thông đủ để các lập trình viên có thể chạy thử mà không không thể đưa game vào hoạt động.

* + - 1. **NetworkMangager và các component hỗ trợ kết nối**
* **NetworkManager:** đây là thành phần mẹ và là thành phần cần được tạo ta đầu tiên để hình thành và đăng kí network. Nắm giữ tất cả các thông số kết nối cũng như object cần đăng kí để đồng bộ giữa các người chơi. Thành phần này quyết định đâu là offline scene, đâu là online scene và chuển đổi giữa hai scene này dựa vào sự thay đổi trạng thái của network. Component này cũng đăng kí gameobject được xem như là “người chơi” và tạo ra object này khi kết nối thành công. Ngoài ra thì các thành phần cần đồng bộ giữa các người chơi đề phải đăng kí ở trong component này.
* **NetworkLoobyManager:** là một thành phần kế thừa từ NetworkManager. NetworkLoobyManager cung cấp thêm những chức năng trung gian từ khi kết nối thành công cho tới khi scene game chính được tải. Component này phụ hợp cho những game cần những cài đặt những thông số thời gian thực như màu sắc, nickname…
* **NetworkManagerHUB**: là component sinh ra để hỗ trợ và thường đi đôi với NetworkManager. NetworkManagerHUB hỗ trợ trong việc quyết định người chơi đóng vai trò máy chủ hay máy khách. Nhà phát triển có thể sử dụng UI mặc định của component này để người chơi tự kết nối nhưng thông thường khi lập trình sẽ ẩn phần UI này và chủ yếu sử dụng component này trong script vì UI của nó khá xấu và tác vụ thủ công(nếu muốn kết nối tới máy chủ phải biết địa chỉ IP)
* **NetworkDiscovery:** đây là component hỗ trợ trong việc tự động tìm kiếm những máy chủ trong mạng LAN và trả về các thông tin để NetworkManager kết nối. Component nếu được quyết định là máy chủ thì sẽ liên tục phát ra những gói tin và nếu được cài đặt thành máy khách thì sẽ lắng nghe những gói tin phát ra từ các máy chủ. Những gói tin này bao gồm thông tin IP máy chủ để kết nối và một chuỗi thông tin do máy chủ quy định, thường là tên của máy chủ hoặc tên phòng chơi mà máy chủ đang muốn tìm người chơi.
* **MatchMaker:** khác với những đối tượng được đề cập ở trên, MatchMaker không phải là một component mà là một script, cần kế thừa để sử dụng chức năng của nó. MatchMaker hỗ trợ tạo ra các phòng chơi cũng như tham gia các phòng có trước thông qua internet. MatchMaker tìm người chơi thông qua dịch vụ của Unity mà không cần biết ip những máy mà nó kết nối.
  + - 1. **NetworkTransform và các component hỗ trợ đồng bộ**
* **NetworkTransform:** là component hỗ trợ đồng bộ transform của gameobject mà nó được gắn vào. Có nhiều tùy chọn cho việc đồng bộ của nó, có thể đồng bộ toàn bộ transform hoặc có thể chỉ đồng bộ rigidbody. Ngoài ra còn có thể chỉnh thời gian giữa hai lần đồng bộ, thời gian này càng ngắn thì đồng bộ càng nhanh nhưng đi cùng với nó là hiệu năng của game bị giảm sút.
* **NetworkTrasformChild:** là component tương tự như NetworkTransform nhưng lần này là đồng bộ các gameobject con của gameobject nó được đính vào. Lưu ý là component này chỉ đồng bộ transform chứ không đồng bộ rigidbody vật lí và chỉ hoạt động khi gameobject chưa nó cũng chứa NetworkTranform.
* **NetworkAnimator:** đồng bộ animation của gameobject mà nó được đính vào.
* **NetworkPosition:** component này thường được đính vào một gameobject rỗng. Khi tạo ra một prefab thể hiện người chơi trong scene game chính, NetworkManager sẽ tự động tìm các gameobject có chứa các component này trong scene và lấy ra vị trí của chúng. Sau đó từ những vị trí này NatworkManager sẽ chọn ra một vị trí dựa theo phương thức đã được cài đặt trước đó(ngẫu nhiên hoặc xoay vòng) và gán chúng cho những người chơi mới. Vì đặc thù chức năng nên component này chỉ hoạt động trong scene game chính và không có bất kí setting nào.
* **NetworkIdentify:** bất cứ gameobject nào chuyển động, bất cứ gameobject nào cần đồng bộ, bất cứ gameobject nào có liên quan đến network đều phải chứa một component NetworkIdentify. Component này cho phép các component network khác định danh và quyền điều khiển gameobject của nó. Có hai tùy chọn là Server Only và Local Player Authority với các quyền tương ứng. Nếu không có tùy chọn nào được chọn thì NetworkManager sẽ tùy chỉnh theo ngữ cảnh.

****Hình 2.9: Các thành phần Networking

* + 1. **Scripting**

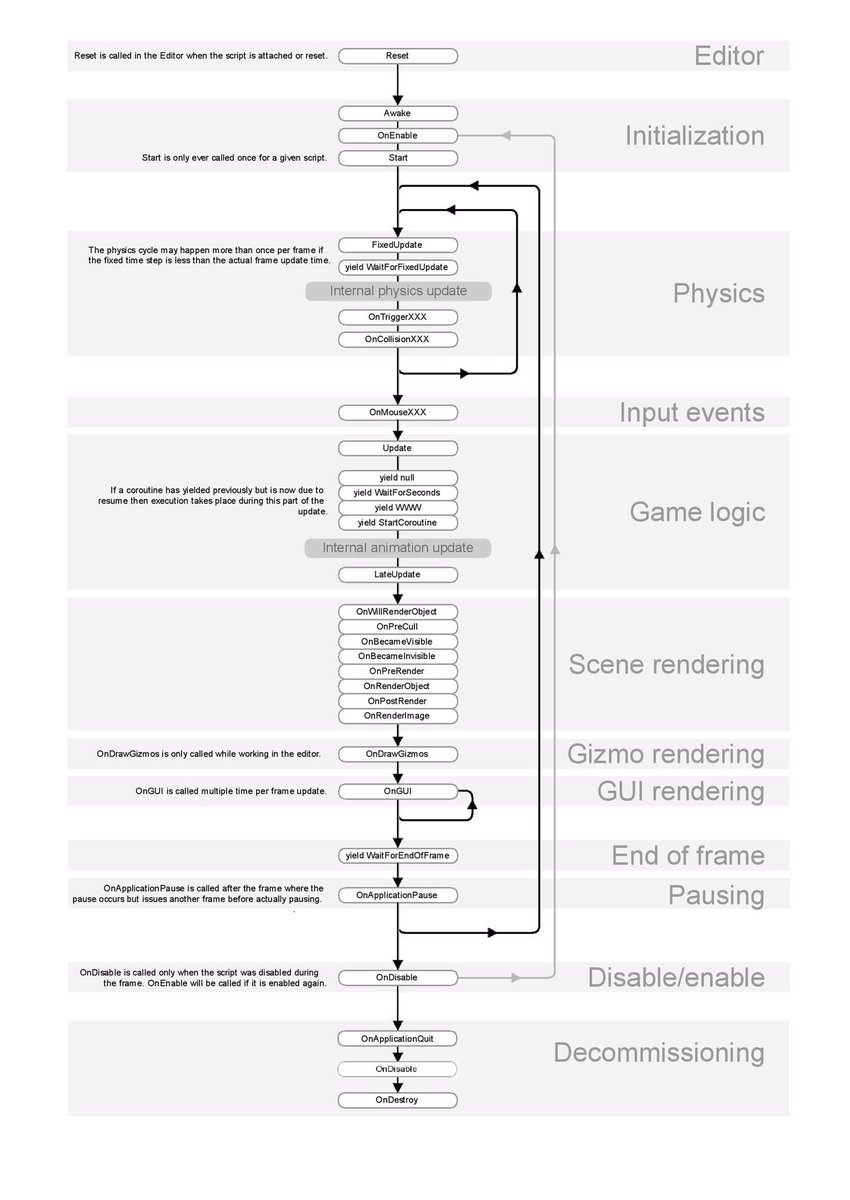
Như những thành phần mà Unity cung cấp cho chúng ta đã đề cập ở những mục trên, dường như mới nhìn vào sẽ tạo cảm giác Unity đã làm tất cả, nhưng thực tế những thành phần kia không thể nào hoạt động đúng và theo ý nhà phát triển chỉ với các thao tác kéo thả. Nói cách khác, scrpit giống như bộ não của game, điều khiển tất cả các component để thể hiện ý muốn của các nhà phát triển.

Hiện nay, Unity hỗ trợ hai loại ngôn ngữ để biết script bao gồm C# và Javascript. Trước đây Unity cũng có hỗ trợ ngôn ngữ Boo nhưng thị phần nhà phát triển quá ít nên đã được cắt giảm. Tuy hỗ trợ song song hai ngôn ngữ tuy nhiên những chức năng mới hay đặc biệt thì đa phần được hỗ trợ trên C# trước.

Script trong Unity thường được đính kèm vào trong một gameobject và có thể thao tác với bất kì gameobject hay component nào nó được tham chiếu đến. Để có khả năng đính kèm vào một gameobject như một component và có các tính chất của game thì script phải được kế thừa từ một lớp MonoBehaviour.

Những phương thức cơ bản của một Monobehaviour:

* Awake: gọi đầu tiên và duy nhất một lần trong vòng đời của gameobject, thường thì chúng ta sẽ gọi và truyền những cài đặt cho màn chơi trong hàm này. Hàm này chạy khi chưa bắt đầu render đồ họa nên không truy xuất được đến các gameobject, chỉ có thể sử dụng các biến.
* Start: hàm này chạy ngay sau Awake và lặp lại mỗi khi gameobject được tắt và bật trở lại. Ta có thể bắt đầu tác động đến những gameobject ở giai đoạn này.
* Update: được gọi lặp đi lặp lại mỗi một khung hình. Thời gian chạy của hàm này tùy thuộc vào độ phức tạp của nó. Nếu mất quá nhiều thời gian để thực hiện hàm này thì trò chơi sẽ bị giật nên cần hạn chế tối đa việc lạm dụng hàm này.
* OnDestroy: hàm này được gọi khi gameobject được xóa và giải phóng biến. Thường thì ta sẽ gọi những hàm giải phóng những tài nguyên không sử dụng nữa trong này.

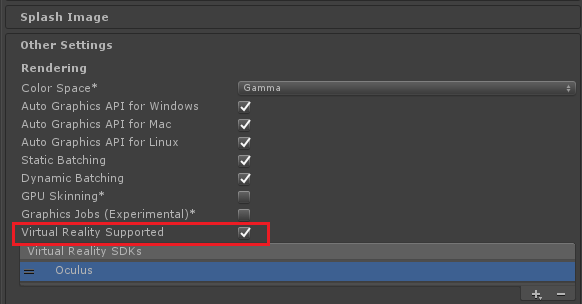
  
Hình 2.10: Vòng đời của một script

* + 1. **VirtualReality và AugmentedReality**

Công nghệ ngày càng phát triển và những thứ trên mặt phẳng 2D đã không còn thu hút được người chơi cũng như những nhà phát triển. Để đáp ứng những nhu cầu mới mẻ như vậy, Unity đã không ngừng tích hợp và hỗ trợ cho các nền tảng VR và AR lớn hiện nay.

* + - 1. **VR**

Nếu kể tên những sản phẩm kính VR phổ thông nhất hiện nay có thể kể đến kính VR Oculus Rift, HTC Vive, VR trên thiết bị di động thì ta có Gear của Samsung. Với ba thiết bị chiếm gần trọn thị phần kính VR kể trên thì Unity đã hỗ trợ trực tiếp hai thiết bị là Oculus Rift và Samsung Gear. Còn đối với HTC Vive không có nghĩa là không thể phát triển được trên Unity, thay vì kết nối thiết bị và sử dụng ngay như hai kính đã đề cập ở trên thì HTC Vive phải qua một bước cài plugin nữa, plugin này cũng có sẵn trên AssetStore và cũng chỉ cần vài phút để cài đặt. Đó là những minh chứng cho thấy Unity đang làm chủ trong việc phát triển các ứng dụng VR.

  
Hình 2.11: Cài đặt hỗ trợ VR cho game

Có thể nói VR vẫn là một công nghệ mới đối với đa số những nhà phát triển trên toàn cầu, tuy nhiên Unity đã hỗ trợ tốt để cho các nhà phát triển tiếp cận công nghệ một cách nhanh chóng và dễ dàng nhất.

* + - 1. **AR**

AR là một công nghệ ra đời và phổ biến sau VR, tuy nhiên hiện nay thị hiếu của thị trường cho thấy AR đang được ưa chuộng và có những bước tiến lớn trong khi VR đã có dấu hiệu chững lại.

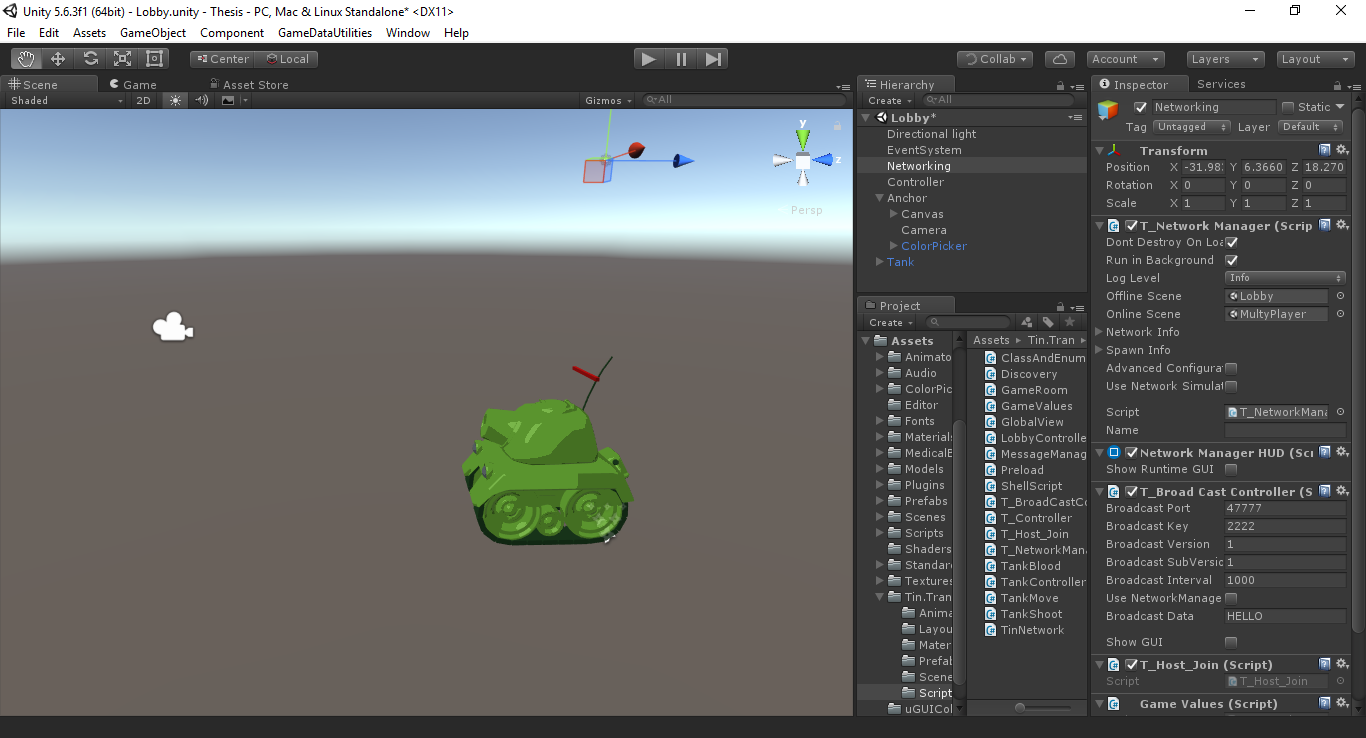
Về thiết bị đánh dấu sự ra đời và bước vào cuộc sống của AR là Microsoft Hololens. Cho đến nay thì Hololens vẫn là thiết bị hỗ trợ đầy đủ các chức năng của AR nhất. Có một câu chuyện tuy khó tin là Unity đã chính thức hỗ trợ phát triển trên Hololens ngay từ khi kính này còn chưa được ra mắt chính thức. Cho tới nay, hơn 91% ứng dụng của Hololens được phát triển bới Unity(theo trang chủ unity3d.com) Điều đó cho thấy sự quan tâm và móc nối với những nhà sản xuất khác để đón đầu công nghệ của Unity. Vì nhiều lí do mà Hololens không phát triển và đạt được thành công như Microsoft và Unity mong đợi tuy nhiên không thể phủ nhận những đóng góp của hãng cho sự phát triển của công nghệ.

Về AR trên thiết bị di động, không ảm đạm như trên những thiết bị chuyên dụng mà là một bức tranh hoàn toàn khác. Nhắc đến thiết bị di động là nhắc tới hai nền tảng Android và iOS hãy cùng xem những hãng này chuẩn bị như thế nào cho cuộc chiến thị phần:

* iOS: sự ra đời của iOS 11 đi kèm với đó là sự ra đời của ARKit, một framework hỗ trợ phát triển những ứng dụng AR trên những thiết bị iPhone và iPad của hãng. iPhone 6s là thiết bị lâu đời nhất có thể hỗ trợ iOS 11, điều đó có nghĩa ARKit sẽ được hỗ trợ từ iPhone 6s trở đi, cũng không quá tệ khi mà ARKit ra đời sau dòng 6s đến tận hai năm. Về nguyên lí hoạt động, ARKit dùng Gyro Sensor và phân tích hình ảnh thu được từ camera để phát hiện chuyển động của thiết bị từ đó điều khiển nội dung ảo bên trong ứng dụng. ARKit còn dùng AI để phân tích ánh sáng môi trường để phát hiện những mặt phẳng và xa hơn nữa là đổ bóng cho một mô hình 3D mà ta thêm vào từ trong ứng dụng. Đối với những thiết bị đời mới được trang bị camera kép thì ARKit cùng cặp camera kép này để phát hiện chiều sâu, hai camera với hai góc quay khác nhau và AI của ARKit sẽ so sánh và đưa ra được chiều sâu của từng mặt phẳng hay đối tượng mà nó phát hiện từ những phương pháp đề cập ở trên. Mới đây sự ra đời của iPhone X đã đánh dấu sự ra đời và xu hướng phần cứng của Apple trong những đời iPhone tiếp theo: chip AI riêng biệt và camera true depth. Có thể nói khó khăn lớn nhất của việc phát triển AR trên thiết bị di động là nó không thể biết được chính xác một trường 3D xung quanh, tất cả đều phải được phân tích bằng AI và độ chính xác không thể tuyệt đối. Nếu Apple thành công trong việc đưa camera true depth ra mặt sau của iPhone cộng với việc phát triển chip AI riêng có hiệu suất cao như hiện nay thì gần như họ đã nắm trọn công nghệ AR trong thiết bị của mình.
* Android: tuy được giới thiệu sau ARKit nhưng ARCore của Android mới là nền tảng được phát hành tới những nhà phát triển trước. Ngay từ khi ra mắt thì ARCore đã ngay lập tức hỗ trợ hai môi trường là AndroidStudio và Unity trong khi muốn sử dụng ARKit phải dùng plugin do Unity phát triển. Hiện nay ARCore chỉ hỗ trợ trên hai thiết bị là Google Pixel và Samsung Galaxy S8, tuy nhiên hãng cũng có lời hứa đưa ARCore lên tất cả các thiết bị chạy Android 7.0 trở đi vào mùa đông này. Về chức năng thì ARCore thua thiệt hơn so với ARKit, ARCore hoàn toàn dựa trên kết quả phân tích hình ảnh của AI mà không dùng tới bất kì tín hiệu phần cứng nào khác, chính vì cậy độ ổn định của ARCore được cho là không tốt bằng ARKit. Ngoài ra Google cũng chưa có dấu hiệu cho thấy họ đang có một chiến lược dài hạn hoàn chỉnh về phát triển AR trên thiết bị của họ.

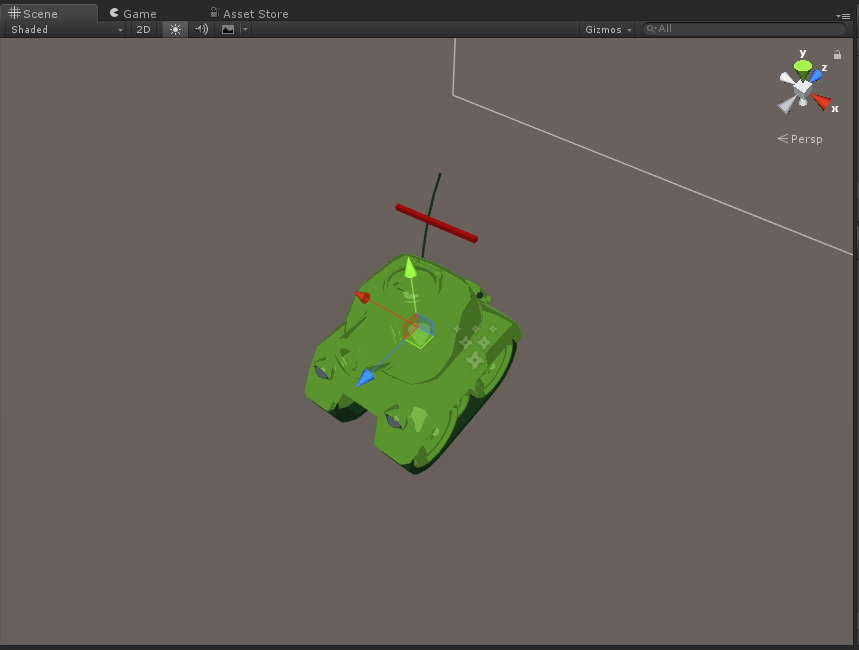
Có thể thấy, những gì hứa hẹn nhất của công nghệ AR vẫn còn ở phía trước và chúng ta vẫn mới chỉ ở bước đầu của giai đoạn phát triển này. Tuy nhiên, với những hỗ trợ tuyệt vời hiện nay thì không có lí gì Unity lại tụt hậu so với những nền tảng khác về hỗ trợ AR.

* 1. **Giao diện người dùng**
     1. **Cửa sổ làm việc**

  
Hình 2.12: Cửa sổ làm việc của Unity

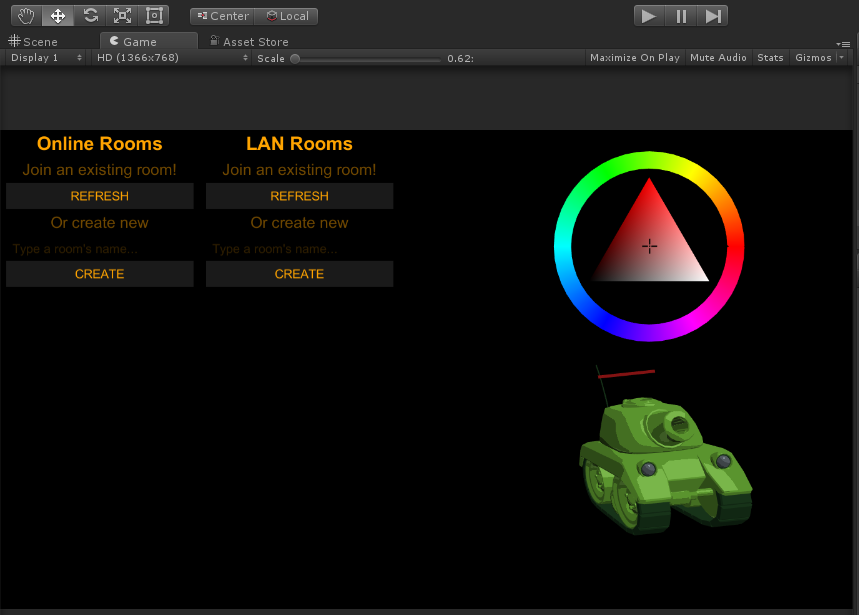
Giống như VisualStudio hoặc những editor khác, Unity editor có những phân vùng và người dùng có thể thêm những cửa sổ vào những phân vùng này sao cho hợp với ý muốn và tiện làm việc nhất.

* + 1. **Cửa sổ Scene**

  
Hình 2.13: Cửa sổ scene

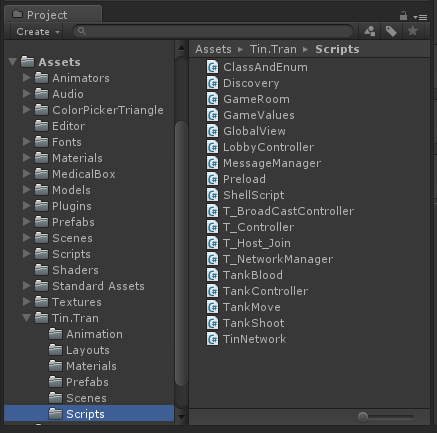
Đây là cửa sổ làm việc chính với môi trường 3D của Unity. Mỗi gameobject sau khi được chọn sẽ được đính kèm một hệ tọa độ 3D rất trực quan. Chúng ta có thể di chuyển gameobject hoặc chọn góc nhìn theo từng trục. Ta cũng có thể chọn chế độ làm việc 2D đối với những game 2D ở cửa sổ này, khi đó sẽ chỉ có hai trục x và y.

* + 1. **Cửa sổ Game**

  
Hình 2.14: Cửa sổ game

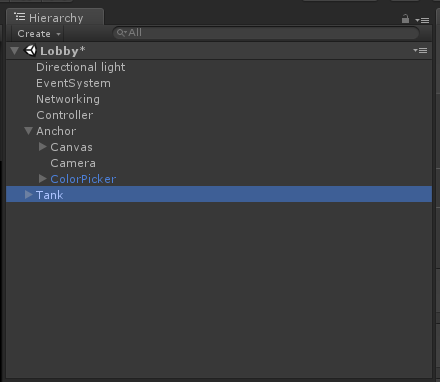
Khi chạy game Unity sẽ tự động chuyển sang màn hình game cho dù đang ở cửa sổ nào. Cửa sổ này cho chúng ta những tùy chọn về độ phân giải và màn hình xuất ra. Ở cửa sổ này ta có thể tương tác với game tương tự như game thực tế sau khi build.

* + 1. **Cửa sổ Project**

  
Hình 2.15: Cửa sổ project

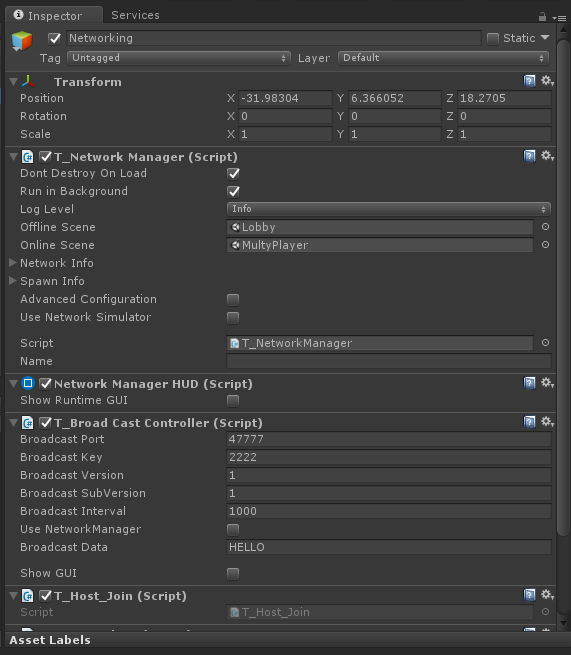
Thật ra đây là cửa sổ thể hiện các asset chứa trong game. Ta có thể đăng kí và di chuyển asset bằng cách kéo thả trong cửa sổ này. Cấu trúc thư mục phân cấp giúp chúng ta dễ dàng hơn trong việc tổ chức và quản lí dữ liệu.

* + 1. **Cửa sổ Hierarachy**

  
Hình 2.16: Cửa sổ Hierarchy

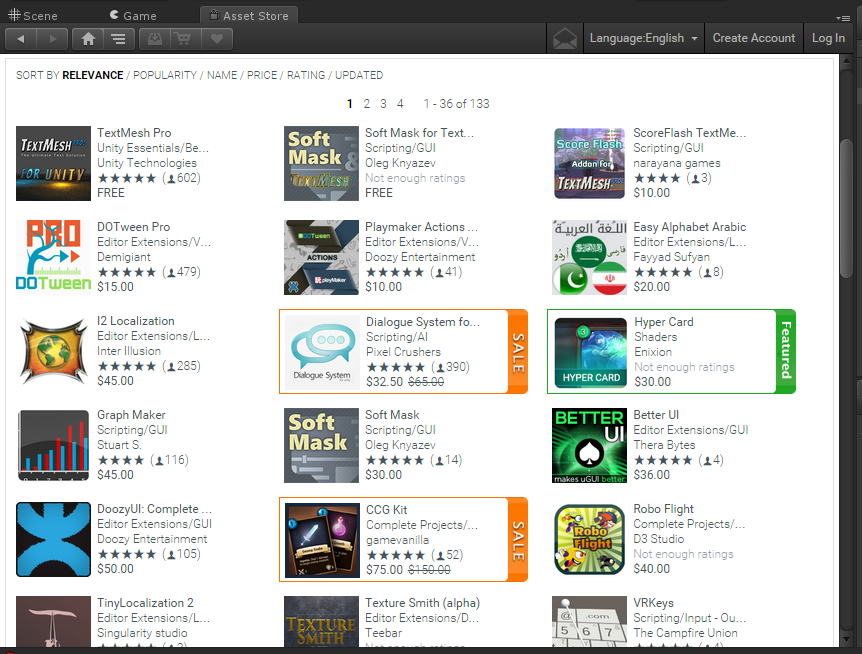
Đây là cửa sổ chứa tất cả các gameobject trong scene. Thể hiện theo cấu trúc phân cấp cha con. Muốn thêm gameobject mới hoặc các thao tác xóa sửa những gameobject sẽ diễn ra trong cửa sổ này.

* + 1. **Cửa sổ Inspector**

  
Hình 2.17: Cửa sổ Inspector

Khi chọn một gameobject ở cửa sổ Hierarchy thì ở cửa sổ Inspector sẽ hiển thị ra các component của gameobject đó. Ở cửa sổ này ta có thể chỉnh sửa các giá trị của các component, cài đặt các mối liên hệ giữa chúng, thêm hoặc xóa các component.

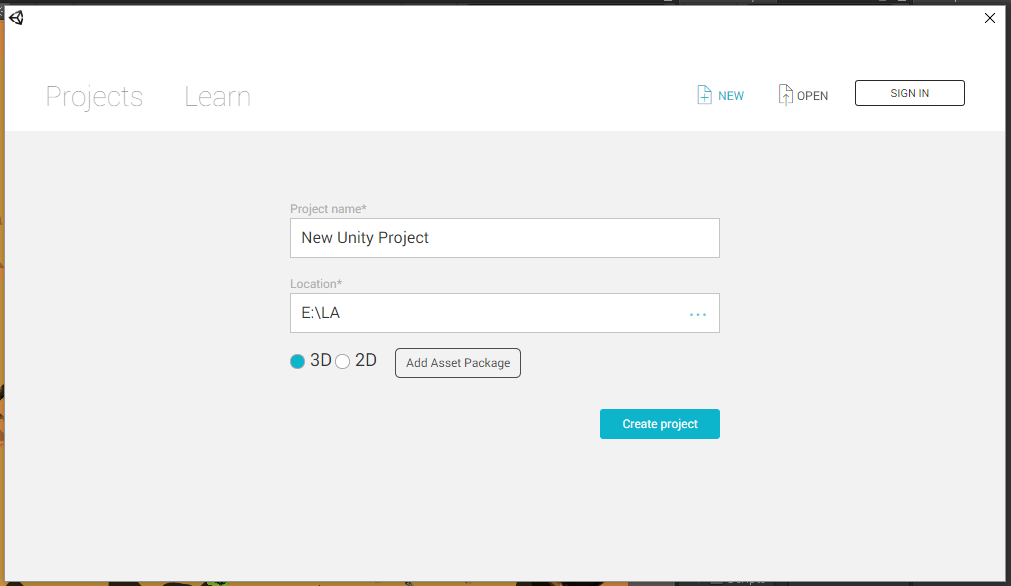
* + 1. **Cửa sổ AssetStore**

  
Hình 2.18: Cửa sổ asset store

Mặc định nếu muốn tải và cài đặt một bộ asset từ AssetStore thì người sử dụng phải đăng nhập tải xuống từ cửa sổ của Unity.

* 1. **Làm quen với các thao tác cơ bản**
     1. **Tạo dự án mới**

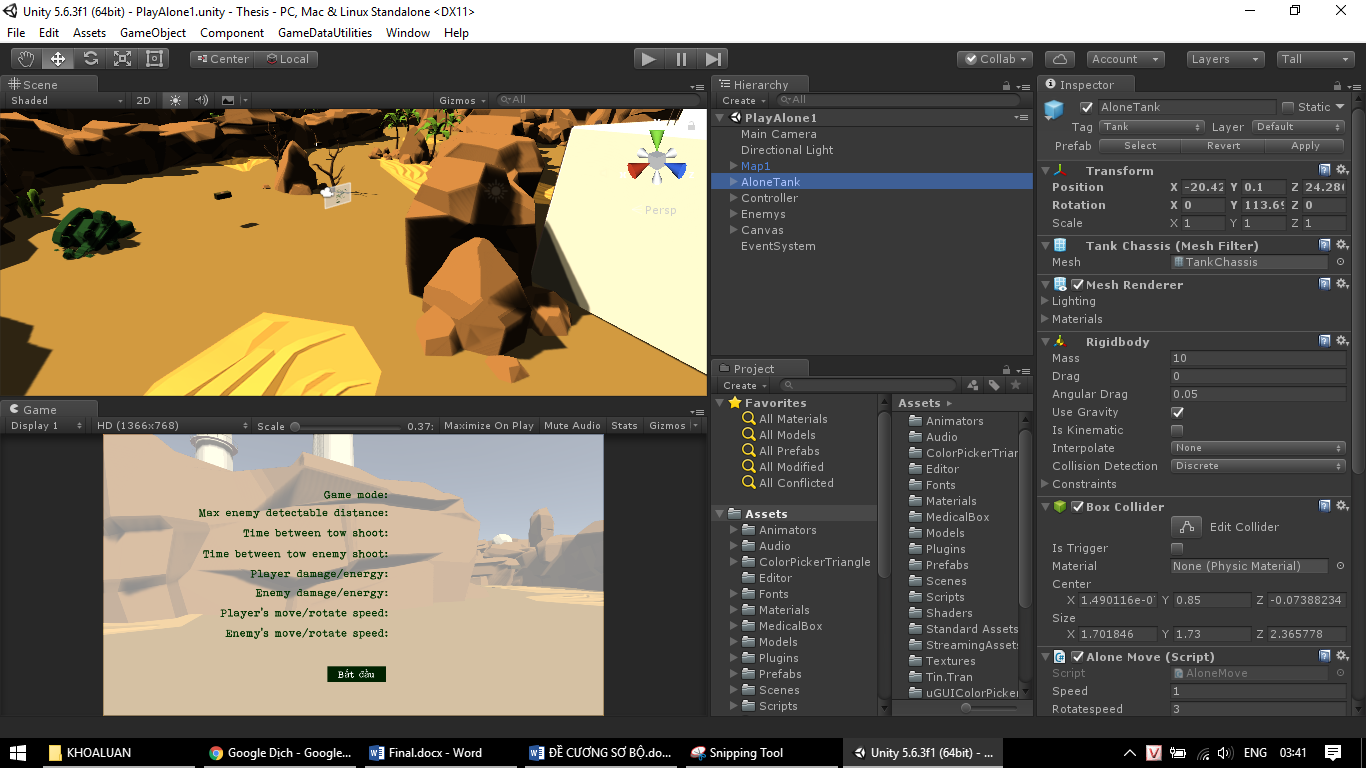
Tạo dự án mới là thao tác đầu tiên và cơ bản nhất đối với bất kì môi trường nào chứ không riêng gì Unity3D. Đây đơn giản là thao tác chỉ cho Unity biết thư mục dự án để Unity nạp những file cần thiết và bắt đầu làm việc.

  
Hình 2.19: Tạo một dự án mới

Như đã thấy như ở hình minh họa, thao tác khá đơn giản chỉ là chọn địa chỉ lưu dự án và khai báo tên dự án. Một tùy chọn phổ biến khác là ta có thể chọn loại của dự án là 2D hay là 3D. Với mỗi tùy chọn sẽ có khung làm việc riêng, asset khi import cũng sẽ có những định dạng mặc định khác nhau, ví dụ đối với chế độ 2D thì một hình ảnh khi import có định dạng mặc định là sprite còn ở chế độ 3D là một texture. Dĩ nhiên đây chỉ là định dạng mặc định, ta có thể thay đổi định dạng tùy theo mục đích sử dụng. Nếu như có nhu cầu làm việc với những gói asset(package) ta có thể chọn ở mục “Add Asset Package” để import nhanh vào dự án.

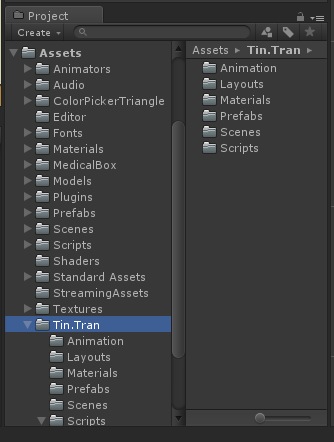
* + 1. **Sắp xếp layout làm việc**

Đối với những lập trình viên thì điều đầu tiên khi tạo xong dự án có lẽ là sắp xếp lại layout làm việc sao cho thuận tiện và thoải mái nhất. Unity chia khung làm việc ra thành nhiều miền, ta có thể tạo ra nhiều miền và đối với mỗi miền ta có thể thêm các cửa sổ mà không có bất kì ràng buộc nào. Điều này tạo nên một sự thuận lợi để bất kì ai cũng có thể tạo ra khung làm việc mà mình mong muốn.

  
Hình 2.20: Một cửa sổ làm việc cơ bản

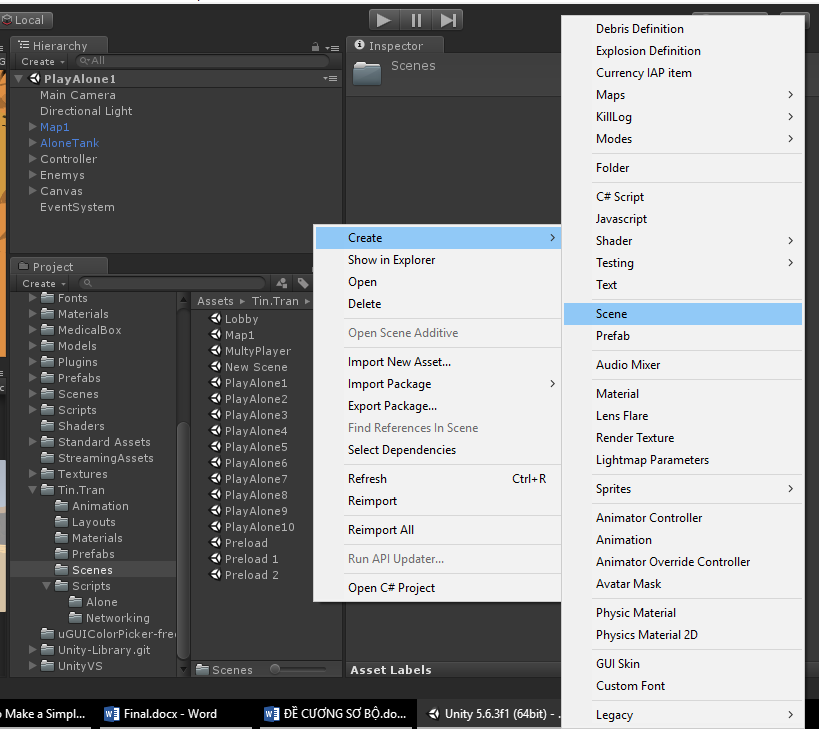
* + 1. **Cấu trúc thư mục asset**

Sau giai đoạn bắt đầu và cài đặt khung làm việc, giờ ta sẽ bắt đầu với project. Điều đầu tiên cần làm là tổ chức các asset thành những mục sao cho khoa học và dễ dàng quản lý sau này. Điều này là không bắt buộc về mặt logic của dự án, tuy nhiên để đảm bảo tính khoa học của dự án và đáp ứng yêu cầu của việc phát triển theo nhóm thì đây là yêu cầu tối thiểu. Thông thường sẽ phân chia theo chức năng, ví dụ tất cả các scene sẽ nằm trong thư mục mang tên “Scene”, tất cả các script sẽ nằm trong thư mục mang tên Script… chẳng hạn.

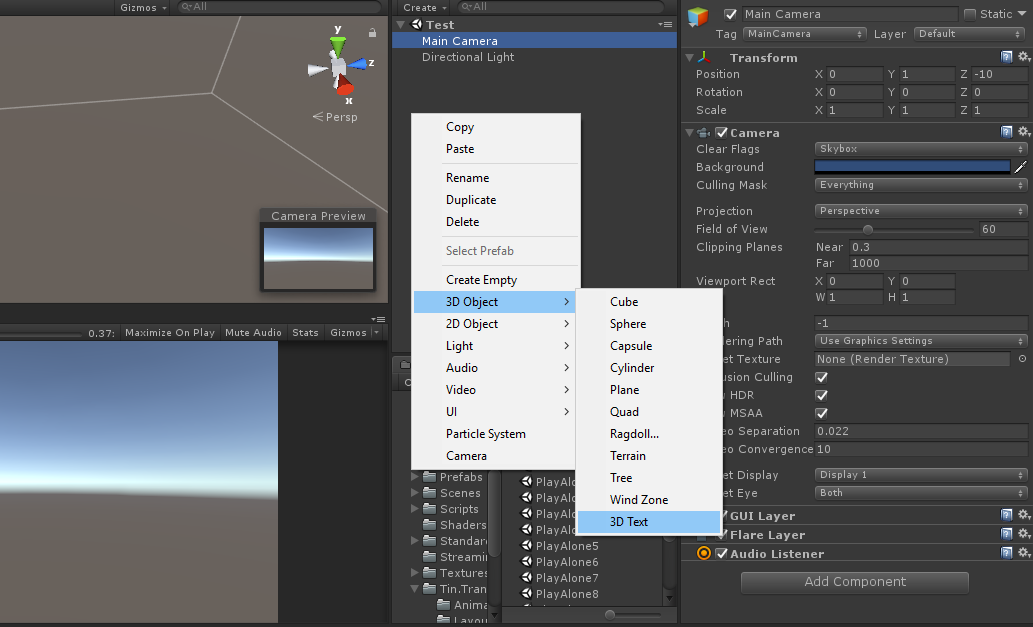
  
Hình 2.21: Một cách tổ chức cơ bản

* + 1. **Tạo scene và thao tác với các gameobject**

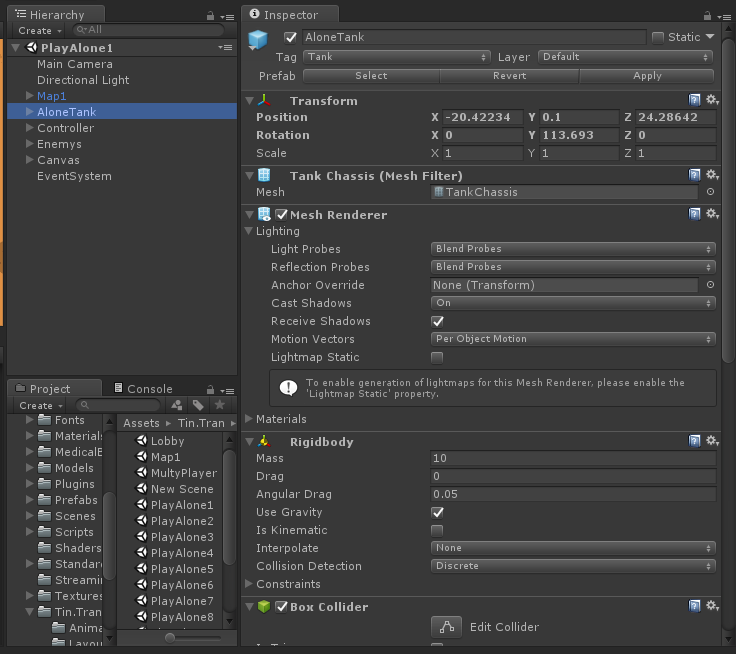
Tất cả các asset được tạo ra cũng như import vào dự án đều có chung một mục đính là để hiện diện trong game theo một cách nào đó. Và để làm được điều đó thì phải thông qua một đối tượng đó là “Scene”. Scene chứa tất cả các gameobject cần thiết để game chạy vậy nên bước tiếp theo khi tìm hiểu về Unity là tạo và làm việc với scene.

  
Hình 2.22: Minh họa cách tạo scene

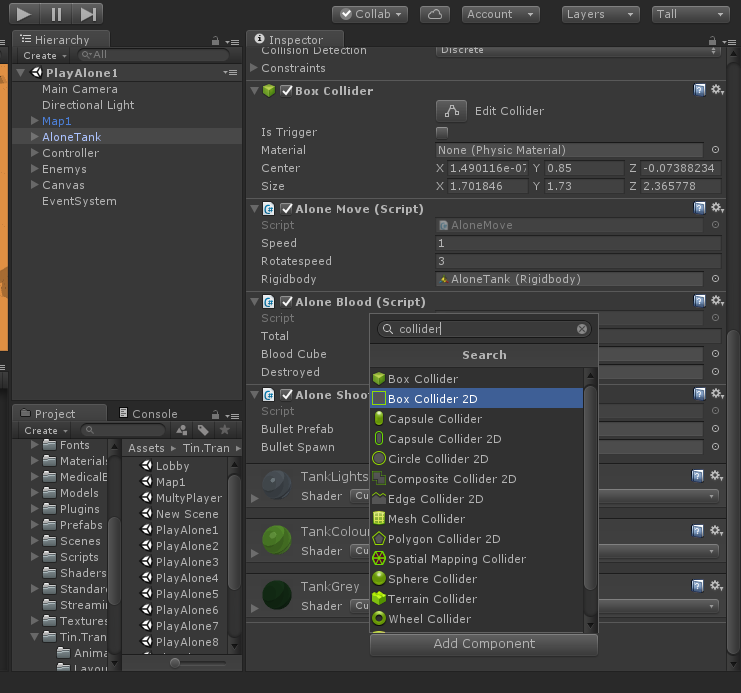
Tại thư mục muốn tạo scene ta chuột phải -> Create -> Scene và đặt tên cho scene. Sau khi tạo ra thì scene sẽ chứa hai gameobject mặc định là Main Camera để render ra hình ảnh và Directional Light cung cấp ánh sáng cho scene. Bắt đầu từ đây ta có thể thêm các gameobject để tạo ra một màn chơi hoàn chỉnh. Các gameobject trong scene sẽ được tổ chức theo mô hình cha con. Các gameobject ở lớp cao nhất thì vị trí được tính theo root transform(Global position) còn các gameobject con thì vị trí được tính dựa vào gameobject cha của nó(Local position).

  
Hình 2.23: Thêm các gameobject vào scene

Các gameobject sẽ có các component đính kèm, nói đúng hơn là các component làm nên bản sắc của một gameobject. Vậy nên khi chọn một gameobject thì những component của gameobject đó sẽ xuất hiện ở cửa sổ Inspector kèm với các thông số để có thể tùy chỉnh.

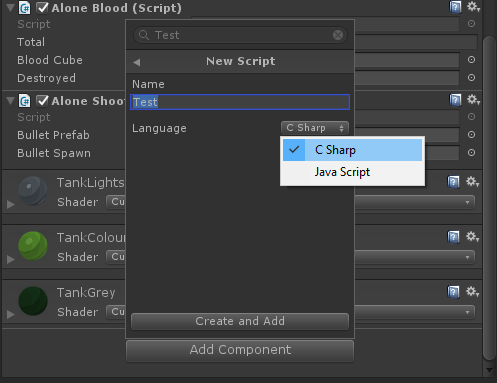
  
Hình 2.24: Một cửa sổ Inspector điển hình

Để tạo nên một gameobject riêng ngoài những gameobject cơ bản mà Unity cung cấp thì ta sẽ thêm những component mà ta mong muốn.

  
Hình 2.25: Cách thêm một component cho gameobject

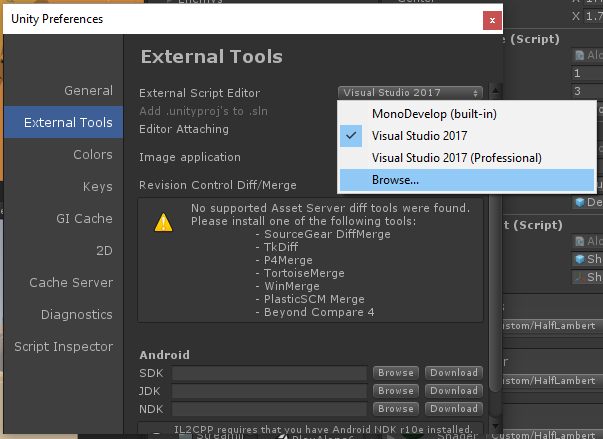
* + 1. **Tạo vào thao tác với script**

Về cơ bản thì tạo một script cũng giống như tạo một scene. Còn một cách nhanh hơn là tạo ngay trong cửa sổ Inspector khi thêm component cho gameobject(một script cũng là một component nên được thêm vào gameobject ở cửa sổ Inspector như các component khác).

  
Hình 2.26: Cách thêm nhanh một script

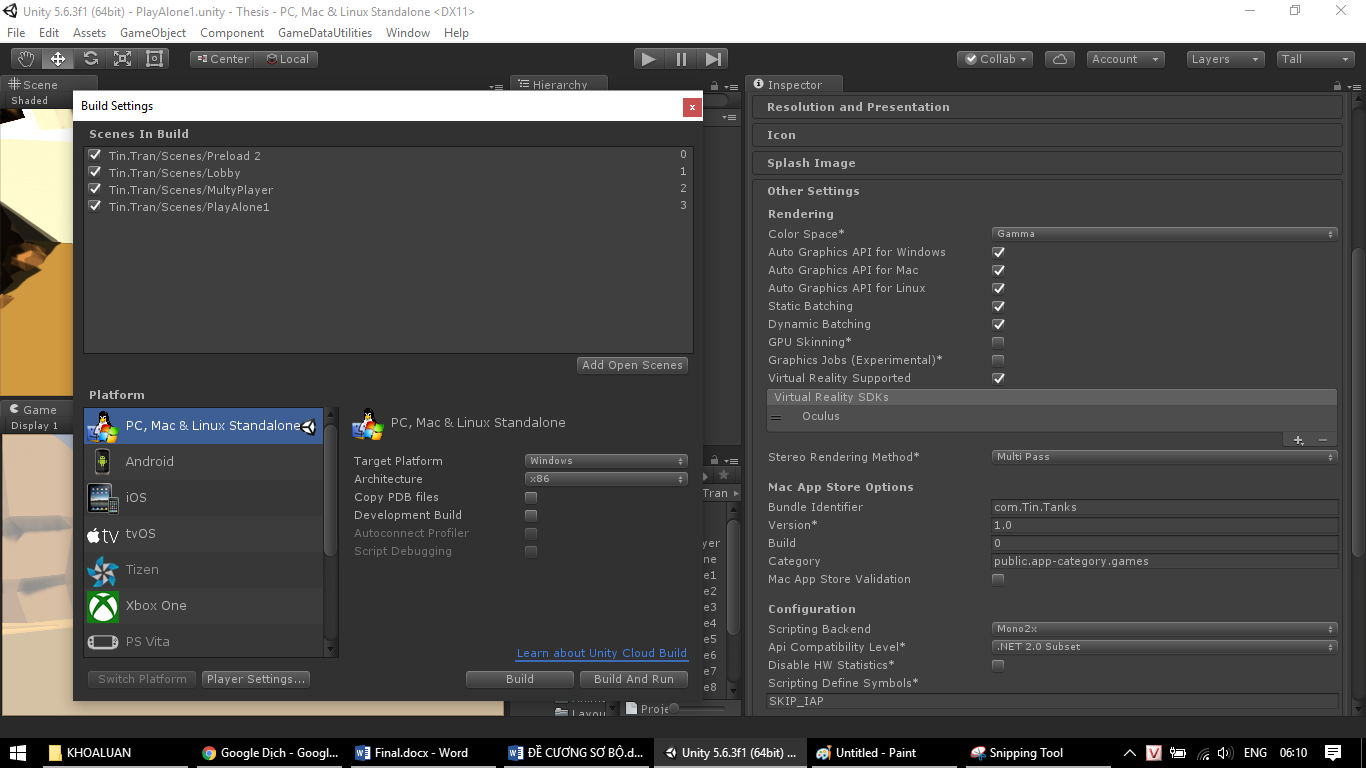
Ưu điểm của cách làm này là nhanh gọn, khi vừa được tạo ra thì script cũng được gán vào gameobject đó như một component. Tuy nhiên nhược điểm của cách làm này là không chọn được vị trí lưu script, theo mặc định script sẽ được lưu trong thư mục asset mẹ nên nếu như các asset đã được phân loại thì ta sẽ phải đưa scipt này về đúng vị trí.

Unity không hỗ trợ sửa trực tiếp script nên thông thường ta sẽ phải dùng một trình biên tập khác để làm việc với script. Có hai trình biên tập thông dụng là MonoDevelop và VisualStudio. MonoDevelop là trình biên tập đính kèm khi cài đặt Unity, đây là một trình biên tập nhẹ nên phù hợp với những máy có cấu hình yếu tuy nhiên vì nhẹ nên hỗ trợ ít chức năng. Visual studio là trình biên tập được Unity khuyến khích sử dụng. Mặc dù đã có kèm theo Monodevelop khi cài đặt nhưng Unity vẫn đề xuất hỗ trợ cài thêm Visual studio tự động. VisualStudio là một trong những trình biên tập hỗ trợ tốt nhất hiện nay, nhất là với ngôn ngữ C#. Tuy nhiên cũng nên cân nhắc khi sử dụng VisualStudio vì chiếm khá nhiều tài nguyên máy, thậm chí là ngang ngửa Unity mặc dù chỉ hỗ trợ viết script. Vậy nên để chạy song song Unity và VisualStudio thì yêu cầu về cấu hình máy khá cao.

  
Hình 2.27: Cài đặt trình biên tập mặc định cho Unity

Để làm việc với script chỉ cần nhấp đúp chuột hoặc nhấn enter sau khi chọn để mở script trong trình biên tập. Script khi được tạo ra mặc định sẽ kế thừa từ lớp MonoBehaviour, một lớp chứa những đặc trưng của một gameobject(chứa những hàm thể hiện vòng đời của một gameobject, đã được trình bày ở phần 2.17 Script).

* + 1. **Tùy chỉnh và build**

  
Hình 2.28: Cửa sổ build và phần cài đặt

Như đã đề cập ở trên, tất cả những gì muốn đến được với người chơi, đóng góp vào game đều phải thông qua scene. Vậy nên khi build ra thành phẩm cuối cùng chỉ cần khai báo các scene cần thiết và build. Khi build sẽ có các tùy chọn về platform và kiến trúc x86 hay x64. Thông thường ta sẽ chọn platform ngay từ khi bắt đầu dự án để được hỗ trợ những tính năng trên platform đó. Khi trong quá trình phát triển thì sẽ build ở chế độ Develop để được hỗ trợ debug và kiểm soát những thông tin như hiệu năng, tài nguyên…

* + 1. **Something**
  1. **Some thing**

1. **CHƯƠNG 3: ỨNG DỤNG XÂY DỰNG MỘT GAME ĐƠN GIẢN**
   1. **Mô tả game**

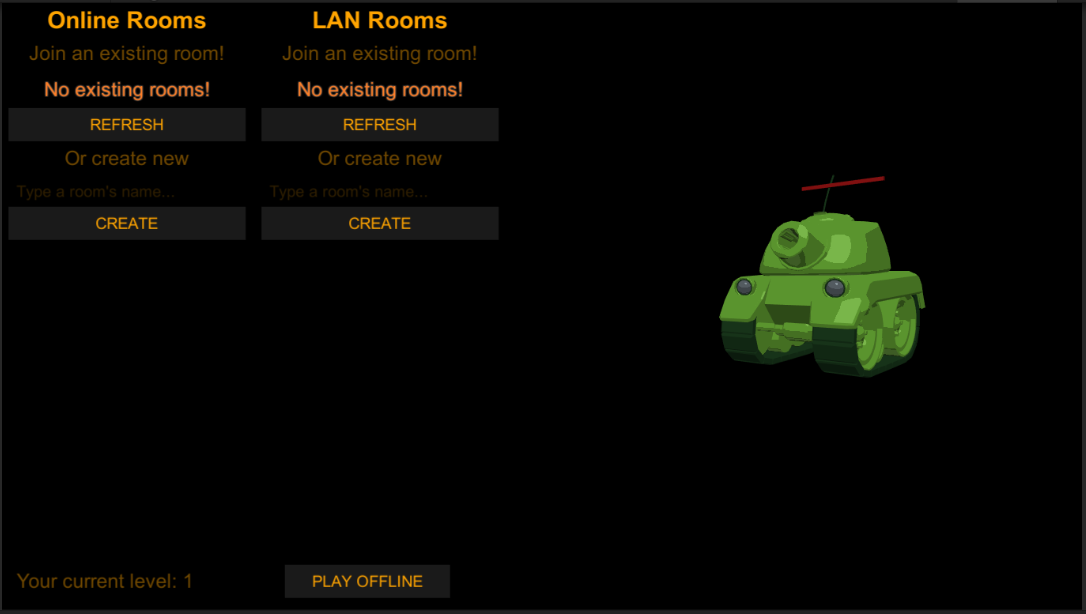
Game có hai chế độ chơi: chơi với máy và chơi với bạn bè thông qua internet hoặc mạng LAN.

Quá trình chơi chung của cả 2 hình thức chơi: mỗi người chơi sẽ có một xe tăng của riêng mình. Nhiệm vụ của người chơi là di chuyển xe tăng và tiêu diệt những xe tăng khác. Tất cả các xe tăng chơi chung đều là đối thủ cần phải tiêu diệt. Game sẽ quy định thời gian tối thiểu giữa 2 lần bắn của người chơi, sau mỗi lần bắn người chơi phải đợi hết khoảng thời gian này mới bắn được phát tiếp theo. Trong quá trình chơi, game sẽ tự động tạo ra các vật phẩm y tế tại các vị trí ngẫu nhiên, khi người chơi ăn được các vật phẩm y tế này thì sẽ được khôi phục năng lượng. Năng lượng và sức mạnh của đạn sẽ phụ thuộc vào level của người chơi và chế độ chơi.

* Đối với chế độ chơi offline với máy thì người chơi sẽ có nhiều map tương ứng với nhiều level khác nhau, người chơi phải hoàn thành các điều kiện của level này để có thể chơi ở level tiếp theo. Các điều kiện có thể là:
  + Tiêu diệt các xe tăng khác trong một khoảng thời gian nhất định
  + Tiêu diệt tất cả các xe tăng với số lần chết ít hơn hoặc bằng một số cho trước

Độ khó của game sẽ tăng theo từng level. Những tiêu chí đánh giá độ khó của game:

* + Khoảng cách địch có thể phát hiện ra người chơi(tỷ lệ thuận với độ khó)
  + Thời gian tối thiểu giữa 2 lần bắn của người chơi(tỷ lệ thuận với độ khó)
  + Năng lượng của xe tăng địch(tỷ lệ thuận với độ khó)
  + Tốc độ xe tăng của người chơi(tỷ lệ nghịch với độ khó)
  + Tốc độ xe tăng địch(tỷ lệ thuận với độ khó)
  + Thời gian xuất hiện các vật phẩm y tế
* Đối với chế độ chơi ở mạng LAN, mọi người chơi trong cùng một phòng sẽ có cùng level khi bắt đầu để đảm bảo tính công bằng. Khi người chơi đầu tiên bắt đầu sẽ thiết lập các tùy chọn của game. Mỗi lượt bắn trúng hay mỗi lượt giết được đối phương sẽ được cộng điểm. Người chơi đạt được số mạng quy định nhanh nhất sẽ là người giành chiến thắng. Sau khi kết thúc khoảng thời gian chơi mà không ai đạt được số mạng cần thiết thì ai giành nhiều điểm nhất sẽ thắng. Số lượng mạng không giới hạn nhưng người chơi phải chờ một khoảng thời gian để được hồi sinh
  1. **Cách chơi và những màn chơi chính**

****Hình 3.1: Màn hình khi vừa bắt đầu game

Khi khởi động game, người chơi sẽ được đưa đến màn hình bắt đầu. Ở đây cung cấp những thông tin về level hiện tại, những phòng hiện đang tồn tại. Ngoài ra người chơi cũng có thể tạo mới phòng và xem trước xe tăng của mình ở màn hình này. Thao tác tạo phòng chỉ đơn giản là nhập tên phòng và nhấn nút Create để tạo.

* + 1. **Chơi offline**

  
Hình 3.2: Màn hình bắt đầu khi vào chơi của chế độ chơi offline.

Khi bắt đầu một màn chơi người chơi sẽ có thời gian để đọc những thông tin về màn chơi như chế độ chơi, những thông tin về năng lượng, máu, thời gian bắn… Sau khi nắm thông tin người chơi sẽ bấm nút bắt đầu để tiếp tục.

* Thao tác chơi: người chơi sẽ dùng các nút mũi tên trên bàn phím để di chuyển và nhấn space để bắn.

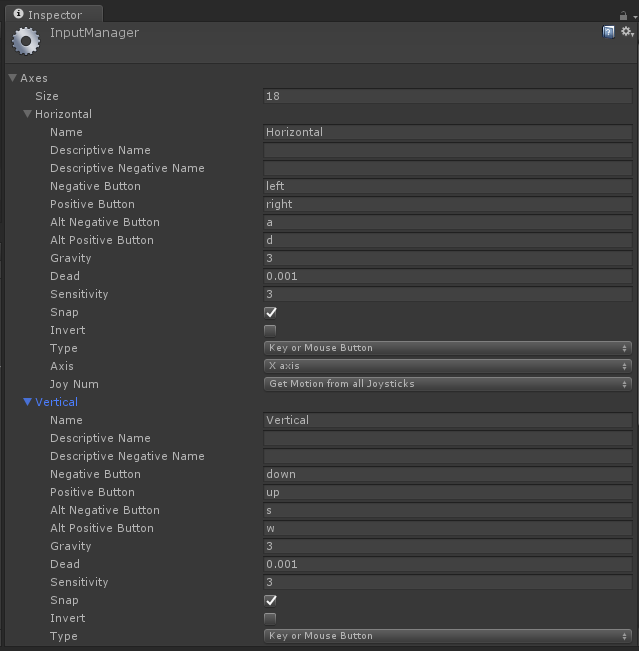
  
Hình 3.3: Màn hình thông báo chiến thắng.

  
Hình 3.4: Màn hình thông báo thất bại.

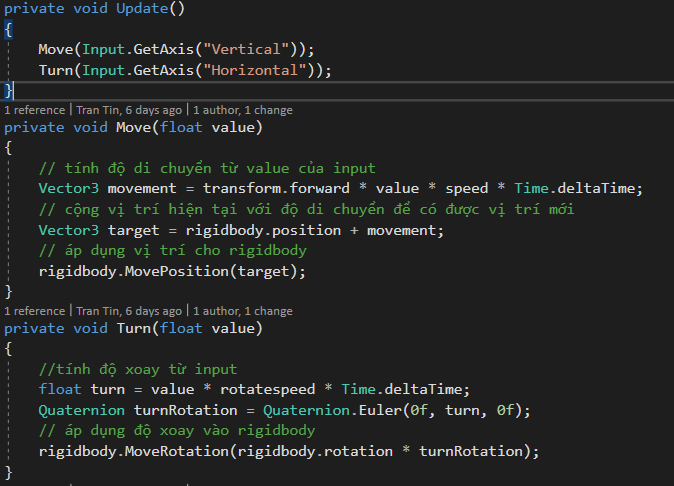
* + 1. **Chơi online**
  1. **33**

1. **CHƯƠNG 4: CÁC KỸ THUẬT CHÍNH**
   1. **Di chuyển trong trò chơi**

Một trong những tính năng cơ bản và được xây dựng đầu tiên trong các trò chơi thường là tính năng di chuyển. Đối với những tổ hợp phím phổ thông như những phím mũi tên hoặc dãy phím WASD thì Unity đã hỗ trợ ở phần input mặc định.

  
Hình 4.1: Bảng cài đặt các input của Unity3D

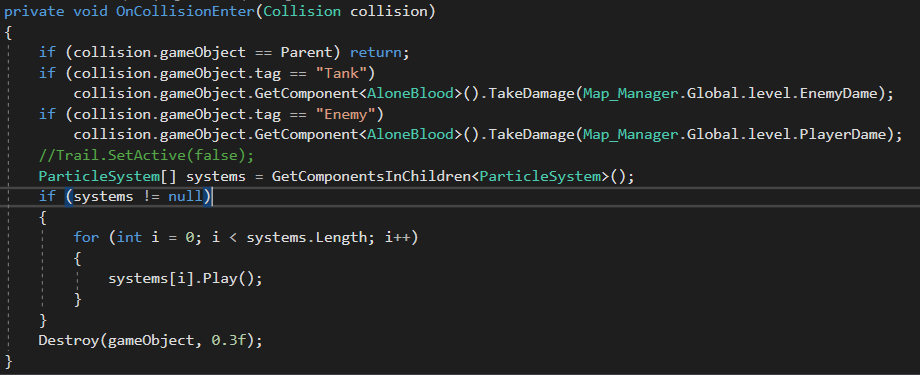
Như đã đề cập ở phần vật lý(Physic), ta không thể di chuyển gameobject người chơi theo cách thông thường là thay đổi giá trị transform vì gameobject có chứa thành phần vật lí rigidbody. Vậy nên sau khi lấy được giá trị input như ở bước trên ta sẽ can thiệp vào thành phần rigidbody để di chuyển gameobject người chơi.

  
Hình 4.2: Quá trình lấy input từ bàn phím và di chuyển đối tượng

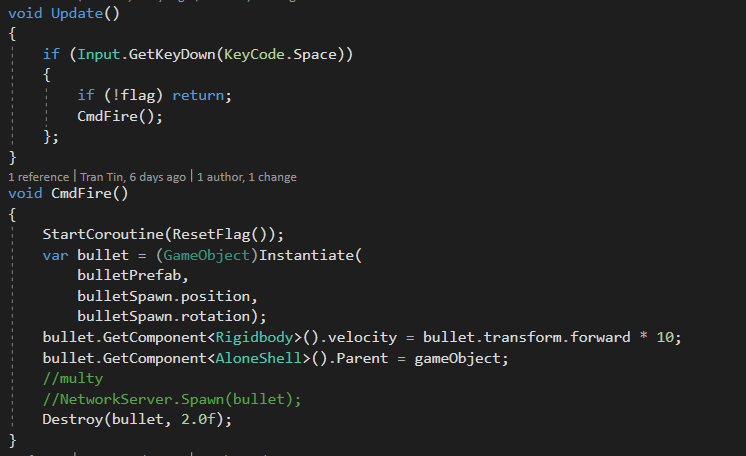
* 1. **Đạn và phương thức bắn**

Như đã đề cập ở phần vật lí, muốn có được sự kiện va chạm trong trò chơi thì cả hai phải có thành phần collider và ít nhất một trong hai phải chứa rigidbody. Trong ngữ cảnh hiện tại thì xe tăng của người chơi đã có thành phần rigidbody nên theo thông lệ ta sẽ không thêm thành phần này vào viên đạn để không làm giảm hiệu năng của game. Tuy nhiên, đạn không chỉ va chạm và làm tổn thương đến xe tăng mà còn phải tương tác với môi trường xung quanh nên bắt buộc ta phải thêm thành phàn rigidbody cho viên đạn.

  
Hình 4.3: Thông số của gameobject viên đạn

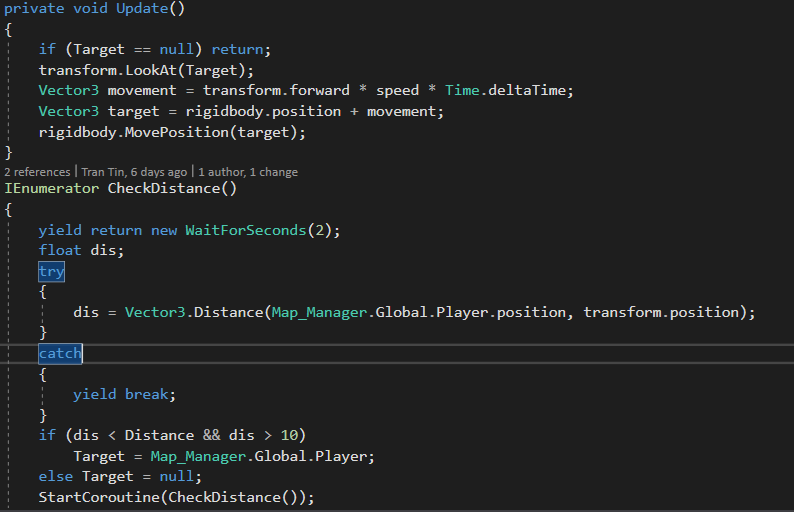
  
Hình 4.4: Đoạn mã phát hiện va chạm và gây sát thương của viên đạn

Đối với thao tác bắn, không có quy chuẩn quy định đồng bộ nên ta sẽ phải tự xác định input và gọi sự kiện thay vì được hỗ trợ như đôi với thao tác di chuyển. Unity cũng cung cấp những phương thức rất dễ dàng để nhận input từ bàn phím hoặc từ màn hình cảm ứng.

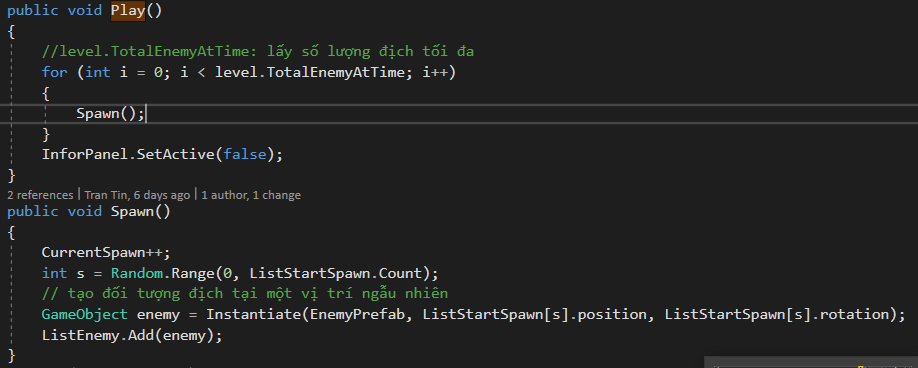
  
Hình 4.5: Đoạn mã lấy sự kiện nhấn nút Space và bắn ra viên đạn

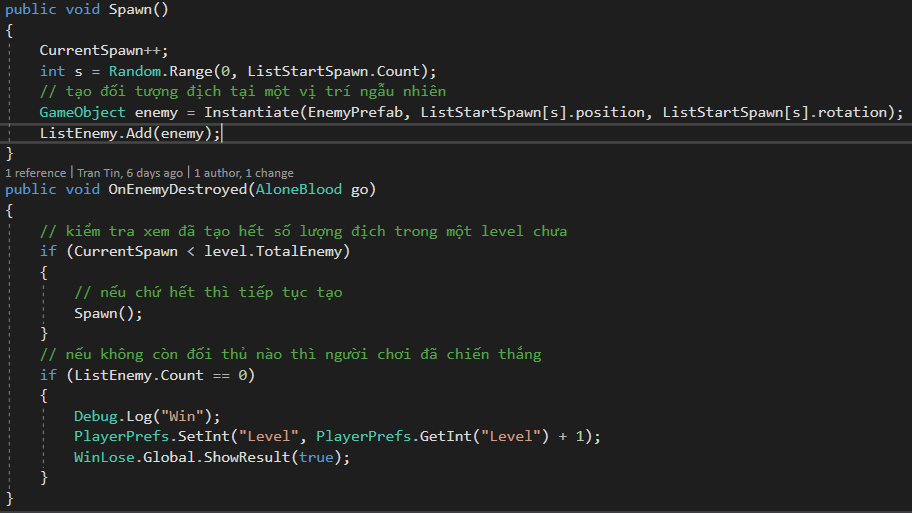
* 1. **“Kẻ thù” và phương thức tạo kẻ thù**

Nếu như đối tượng người chơi mang màu sắc và chịu sự điều khiển của người chơi thì những kẻ thù lại hoàn toàn phải dựa vào sự điều khiển của trò chơi. Giải thuật là ta sẽ lưu giữ vị trí xe tăng của người chơi và từ đó tính toán vị trí giữa nó và các xe tăng địch. Sở dĩ ta tính vị trí tương đối giữa chúng là để đáp ứng yêu cầu của trò chơi là địch sẽ phát hiện người chơi trong một khoảng cách nhất định nào đó. Sau khi tính được khoảng cách thỏa mãn ở bước này công việc còn lại là cho xe tăng định hướng về phía xe tăng của người chơi để tấn công.

  
Hình 4.7: Đoạn mã thực hiện kiểm tra khoảng cách và lần theo dấu người chơi

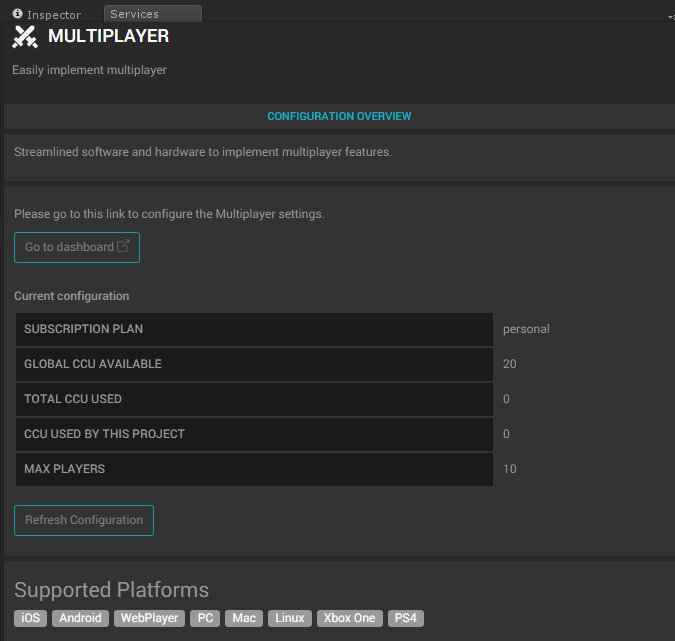
Vì thực tế môi trường trong trò chơi vẫn còn khá nhỏ nên việc đưa cùng lúc tất cả địch lên và tấn công người chơi là hơi bất công. Vậy nên cần xây dựng giải thuật sao cho địch xuất hiện từ từ và phù hợp với khả năng chiến đấu của người chơi. Đầu tiên ta sẽ cần quy định số lượng địch tối đa tại mỗi thời điểm, số lượng này phụ thuộc vào level và được để trong file config. Sau khi tạo đủ số lượng địch thì trò chơi sẽ dừng việc tạo lại để chờ một đối thủ nào đó bị tiêu diệt mới tiếp tục tạo tiếp.

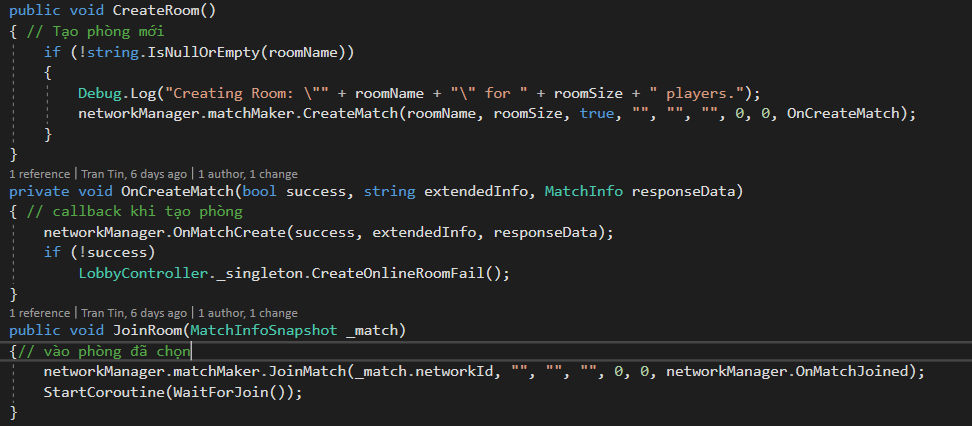
  
Hình 4.8: Đoạn mã tạo số lượng địch tối đa khi khởi động

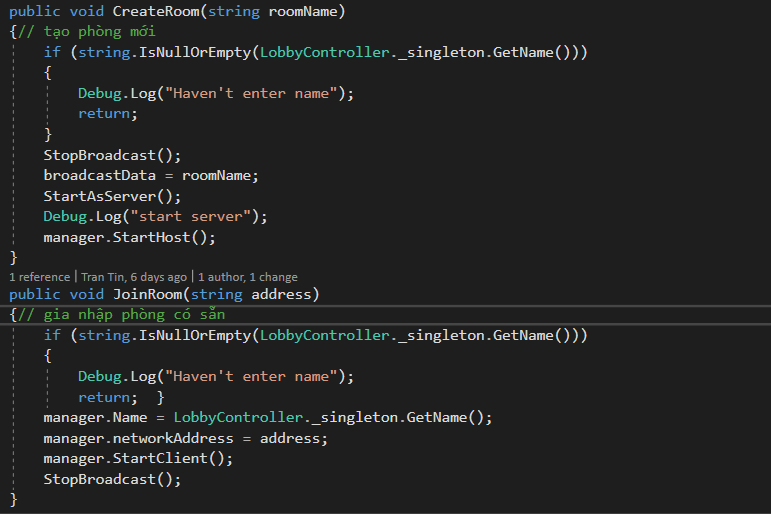
  
Hình 4.9: Kiểm tra và tạo mới mỗi khi có địch bị tiêu diệt

* 1. **Kết nối trong chế độ multyplayer**

Về chế độ kết nối giữa những người chơi với nhau qua mạng thì quy trình là như nhau đối với mạng internet và mạng LAN, chỉ khác nhau ở cách thức kết nối. Mỗi người chơi sẽ chọn mạng kết nối mà mình muốn chơi, sẽ có tùy chọn để người chơi tạo phòng mới hoặc gia nhập một phòng đã được tạo. Quy trình này được Unity hỗ trợ nên khả năng can thiệp còn hạn chế và phải sử dụng dịch vụ đám mây của Unity.

  
Hình 4.9: Bật dịch vụ của Unity để chơi qua internet

  
Hình 4.10: Đoạn mã tạo phòng hoặc gia nhập phòng internet có sẵn

  
Hình 4.11: Đoạn mã chọn phòng và tạo phòng cho mạng LAN

* 1. **Những hạn chế còn tồn đọng**

1. **TÀI LIỆU THAM KHẢO**
2. Wikipedia, (2017), *Unity (phần mềm làm game)*. Nguồn gốc: <https://vi.wikipedia.org/wiki/Unity_(ph%E1%BA%A7n_m%E1%BB%81m_l%C3%A0m_game)> ngày 15/11/2017.
3. VũLê, (2015), “Tại sao Unity 3D được nhà phát triển Game lựa chọn”, *VTCAcademy,* Nguồn gốc: <http://vtcacademy.edu.vn/tintuc/tai-sao-unity-3d-duoc-nha-phat-trien-game-lua-chon.2318.aspx> ngày 15/11/2017.
4. Unity 5 và Unreal Engine 4. Nguồn gốc: <https://viblo.asia/p/unity-5-va-unreal-engine-4-OREGwZRpvlN> ngày 15/11/2017.
5. Hành trình phát triển của Unity 3D – Engine Game phổ biến nhất thế giới hiện nay. Nguồn gốc: [http://genk.vn/dien-thoai/hanh-trinh-phat-trien-cua-unity-3d-engine-game-pho-bien-nhat-the-gioi-hien-nay-20160105165947238.chn ngày 15/11/2017](http://genk.vn/dien-thoai/hanh-trinh-phat-trien-cua-unity-3d-engine-game-pho-bien-nhat-the-gioi-hien-nay-20160105165947238.chn%20ngày%2015/11/2017).
6. TrịnhMinhCường, (2015), “Cơ bản về lập trình game Unity 2D”, *Techmaster,* Tập1. Nguồn gốc: <https://techmaster.vn/posts/33356/hoc-lap-trinh-game-unity-co-ban> ngày 15/11/2017.
7. UI System in Unity – Phần 2: Thành phần Rect Transform. Nguồn gốc: <http://hocunity.3dvietpro.vn/default.aspx?g=posts&m=46#post46> ngày 15/11/2017.
8. Trần Văn Cường, (2016), “Animation và điều khiển hành động nhân vật (Animator)”, *CNTTQN*, Bài4. Nguồn gốc: <https://cnttqn.com/threads/bai-4-animation-va-dieu-khien-hanh-dong-nhan-vat-animator.1517.html> ngày 15/11/2017.