**Bìa 1**

**Bìa 2**

# LỜI CAM ĐOAN

# 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.

# MỤC LỤC

[LỜI CAM ĐOAN i](#_Toc499004321)

[LỜI CẢM ƠN ii](#_Toc499004322)

[MỤC LỤC iii](#_Toc499004323)

[DANH MỤC CÁC TỪ VIẾT TẮT v](#_Toc499004324)

[DANH MỤC CÁC BẢNG vi](#_Toc499004325)

[DANH MỤC CÁC SƠ ĐỒ vii](#_Toc499004326)

[**1** **CHƯƠNG 1: TỔNG QUAN VỀ GAME ENGINE VÀ UNITY3D** 1](#_Toc499004327)

[**1.1.** **Game Engine** 1](#_Toc499004328)

[**1.1.1.** **Khái niệm** 1](#_Toc499004329)

[**1.1.2.** **Xu hướng hiện nay** 1](#_Toc499004330)

[**1.2.** **Unity3D** 3](#_Toc499004331)

[**1.2.1.** **Tổng quan** 3](#_Toc499004332)

[**1.2.2.** **Lịch sử hình thành, phát triển và thành tựu đạt được** 3](#_Toc499004333)

[**1.2.3.** **Tại sao là Unity?** 5](#_Toc499004334)

[**1.2.4.** **Các sản phẩm nổi bật** 6](#_Toc499004335)

[**2** **CHƯƠNG 2: Chi tiết về Unity3D** 7](#_Toc499004336)

[**2.1.** **Các thành phần chính** 7](#_Toc499004337)

[**2.1.1.** **Các khái niệm cơ bản** 7](#_Toc499004338)

[**2.1.2.** **Physics** 9](#_Toc499004339)

[**2.1.2.1.** **Rigidbody** 9](#_Toc499004340)

[**2.1.2.2.** **Colliders** 10](#_Toc499004341)

[**2.1.3.** **UI** 11](#_Toc499004342)

[**2.1.3.1.** **RectTransform** 11](#_Toc499004343)

[**2.1.3.2.** **Canvas và panel** 12](#_Toc499004344)

[**2.1.3.3.** **Các thành phần cơ bản khác** 12](#_Toc499004345)

[**2.1.4.** **Audio và video** 13](#_Toc499004346)

[**2.1.4.1.** **Audio** 13](#_Toc499004347)

[**2.1.4.2.** **Video** 14](#_Toc499004348)

[**2.1.5.** **Animation** 15](#_Toc499004349)

[**2.1.6.** **Networking** 16](#_Toc499004350)

[**2.1.6.1.** **Local network** 16](#_Toc499004351)

[**2.1.6.2.** **Internet**. 16](#_Toc499004352)

[**2.1.6.3.** **NetworkMangager và các component hỗ trợ kết nối** 17](#_Toc499004353)

[**2.1.6.4.** **NetworkTransform và các component hỗ trợ đồng bộ** 18](#_Toc499004354)

[**2.1.7.** **Scripting** 19](#_Toc499004355)

[**2.1.8.** **VirtualReality và AugmentedReality** 22](#_Toc499004356)

[**2.1.8.1.** **VR** 22](#_Toc499004357)

[**2.1.8.2.** **AR** 23](#_Toc499004358)

[**2.2.** **Giao diện người dùng** 25](#_Toc499004359)

[**2.2.1.** **Cửa sổ làm việc** 25](#_Toc499004360)

[**2.2.2.** **Cửa sổ Scene** 25](#_Toc499004361)

[**2.2.3.** **Cửa sổ Game** 26](#_Toc499004362)

[**2.2.4.** **Cửa sổ Project** 27](#_Toc499004363)

[**2.2.5.** **Cửa sổ Hierarachy** 28](#_Toc499004364)

[**2.2.6.** **Cửa sổ Inspector** 29](#_Toc499004365)

[**2.2.7.** **Cửa sổ AssetStore** 30](#_Toc499004366)

# DANH MỤC CÁC TỪ VIẾT TẮT

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

# DANH MỤC CÁC BẢNG

# DANH MỤC CÁC SƠ ĐỒ

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

Một game engine 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 game engine 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 game engine 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 đồ thị... Quá trình phát triển game tiết kiệm được rất nhiều thời gian và kinh phí vào việc tái sử dụng và tái thích ứng một engine để tạo nhiều game khác nhau.

Chúng ta cũng cần chú ý, game engine không chỉ bao gồm thành phần tạo hình ảnh mà bao gồm rất nhiều thành phần khác như trí tuệ nhân tạo, phân luồng, tạo dòng dữ liệu xử lý, 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 Free như 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**

Unity 3D Game 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 với nguồn kinh tế eo hẹp David Helgason (CEO), Nicholas Francis (CCO), and Joachim Ante (CTO) 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 Unity 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, engine 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 App Store. "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 Unity."
* 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 Unity3D.
* Trong năm 2011, Unity đã mua một công ty hoạt hình có tên Mecanim, 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 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 điện thoại di động sử dụng Unity.
* 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, Unity Labs.
* Từ đó đến nay Unity 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 unity 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 Unity 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 là 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 Unity 3D, 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**

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

* **Assets(tài sản)**: Asset là nơi chứa tất cả các file 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 gian, 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 1.1: Các loại file được Unity3D hỗ trợ |

* **Scenes**: 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 Scenes, chúng ta có thể phân phối thời gian và tối ưu tài nguyên, kiểm tra các phân đoạn trong game một cách độc lập.
* **Game Objects:** 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 Game Object 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.
* **Components**: 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 Game Object 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 lâp trình.
* **Prefabs**: Prefabs 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 chính xác 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.

* + - 1. **Colliders**

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.

* + 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 game engine 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**

Rect Transform 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 object, 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.

* + - 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. Ngoài ra text và inputfield còn hỗ trợ định dạng html, điều này nâng tầm khả năng tùy biến nội dung hiển thị lên một tầm cao mới.

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 tưởng tượng ra. 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.

* + - 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.

* + 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.

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.

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**.

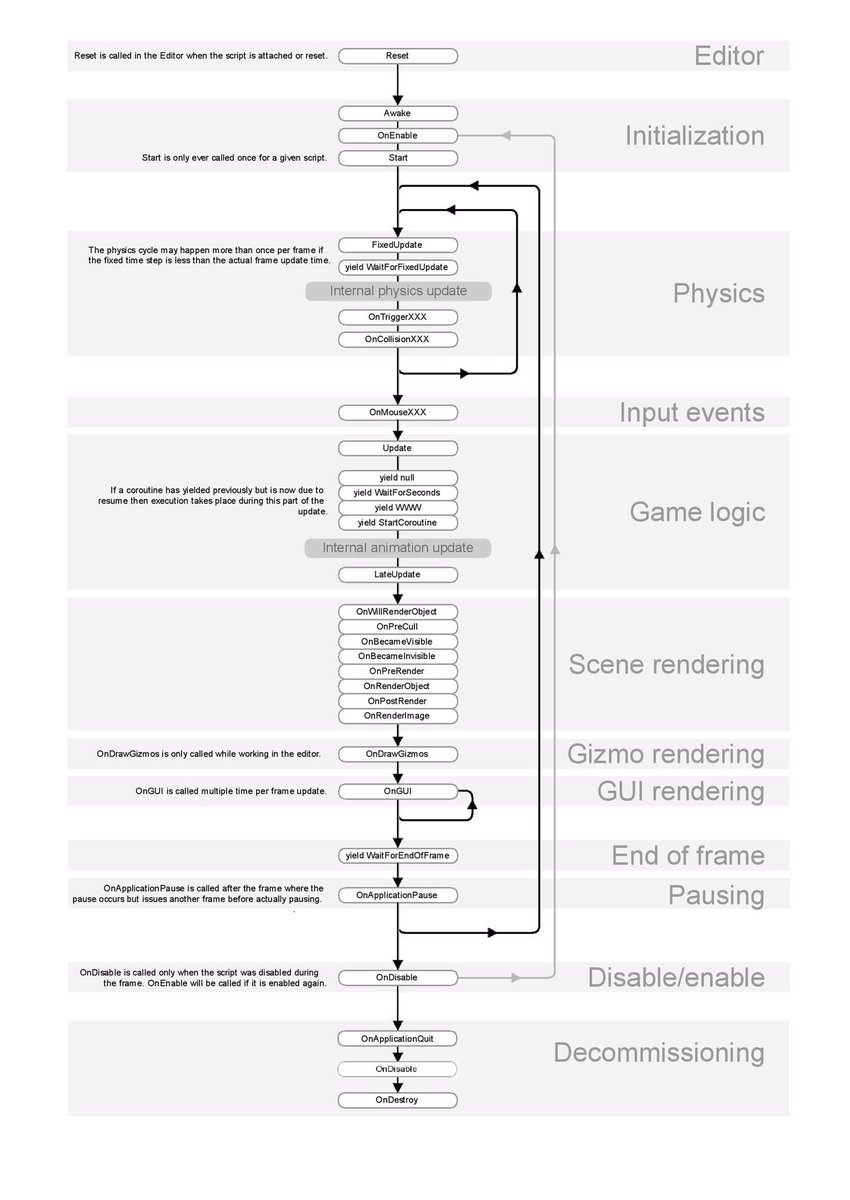
Về các game online thì về cơ bản cũng sử dụng các component làm nên local network game nhưng khác nhau ở phương thức tạo ra và liên kết những người chơi. Để sử dụng multyplayer cũng như những service khác của Unity thì nhà phát triển phải có tài khoản developper của Unity và đăng kí game cần sử dụng 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.
  + 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.

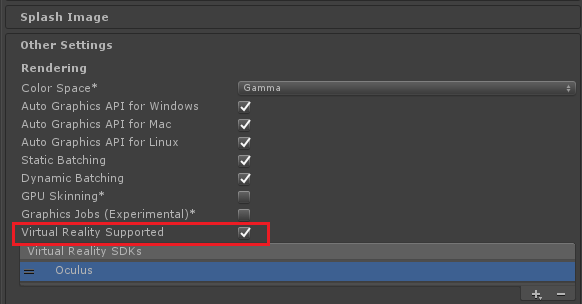
  
Vòng đời của một scripts

* + 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.



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.

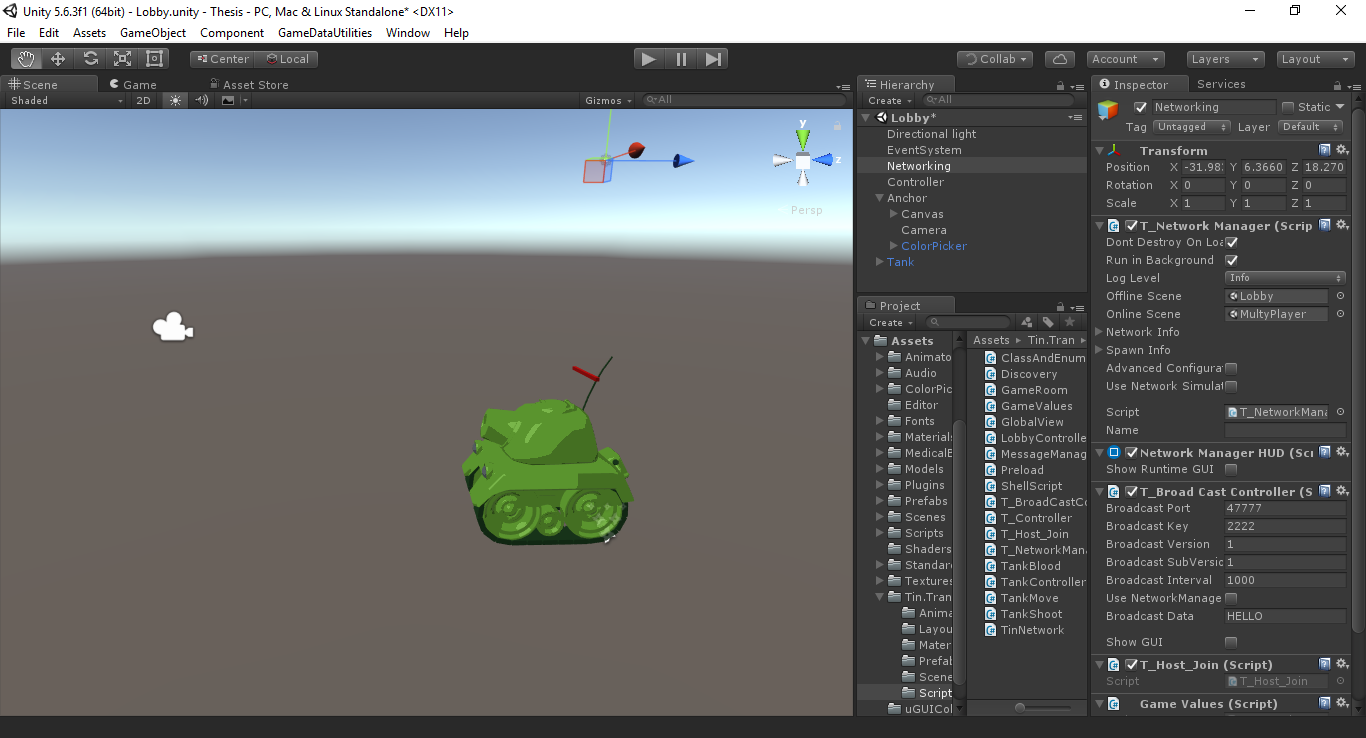
Về thiết bị đánh dấu sự ra đời và bước vào cuộc sống của AR là Microsoft Hololens. Cho đến nay thì Hololens vẫn là thiết bị hỗ trợ đầy đủ các chức năng của AR nhất. Có một câu chuyện tuy khó tin là Unity đã chính thức hỗ trợ phát triển trên Hololens ngay từ khi kính này còn chưa được ra mắt chính thức. 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.

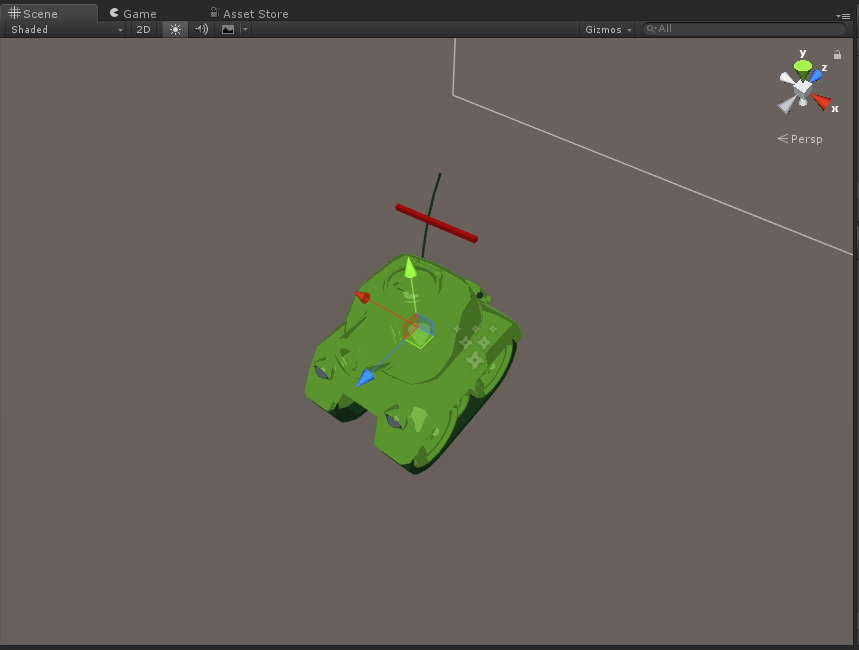
Có thể thấy, những gì hứa hẹn nhất của công nghệ AR vẫn còn ở phía trước và chúng ta vẫn mới chỉ ở bước đầu của giai đoạn phát triển này. Tuy nhiên, với những hỗ trợ tuyệt vời hiện nay thì không có lí gì Unity lại tụt hậu so với những nền tảng khác về hỗ trợ AR.

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



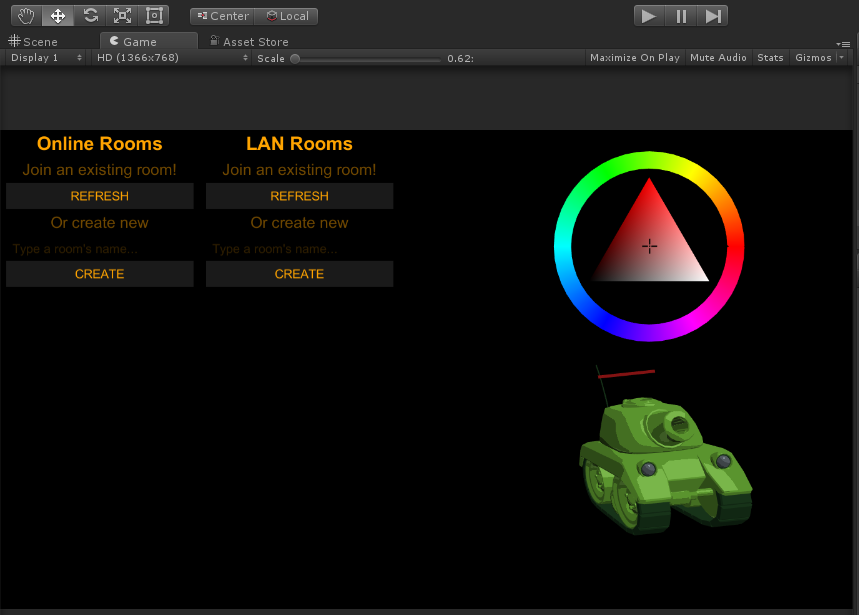
Giống như VisualStudio hoặc những editor khác, Unity editor có những phân vùng và người dùng có thể thêm những cửa sổ vào những phân vùng này sao cho hợp với ý muốn và tiện làm việc nhất.

* + 1. **Cửa sổ Scene**



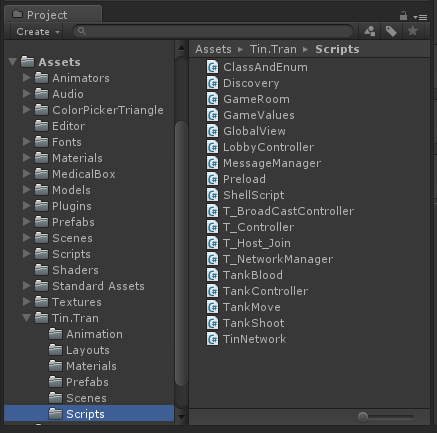
Đây là cửa sổ làm việc chính với môi trường 3D của Unity. Mỗi gameobject sau khi được chọn sẽ được đính kèm một hệ tọa độ 3D rất trực quan. Chúng ta có thể di chuyển gameobject hoặc chọn góc nhìn theo từng trục. Ta cũng có thể chọn chế độ làm việc 2D đối với những game 2D ở cửa sổ này, khi đó sẽ chỉ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

* + 1. **Cửa sổ Game**



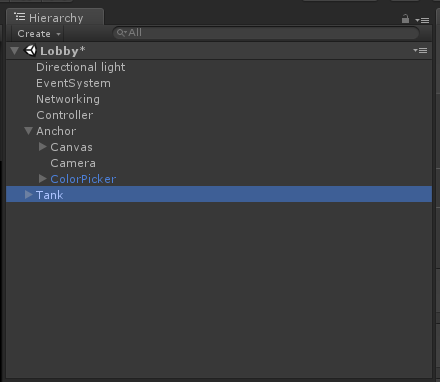
Khi chạy game Unity sẽ tự động chuyển sang màn hình game cho dù đang ở cửa sổ nào. Cửa sổ này cho chúng ta những tùy chọn về độ phân giải và màn hình xuất ra. Ở cửa sổ này ta có thể tương tác với game tương tự như game thực tế sau khi build.

* + 1. **Cửa sổ Project**



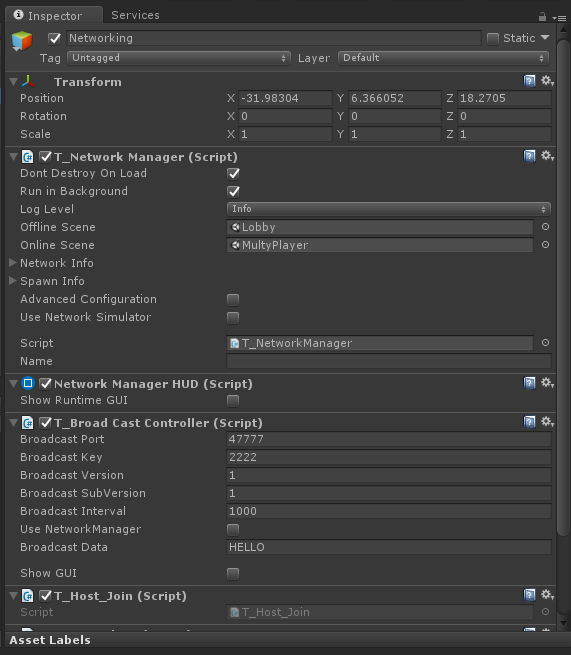
Thật ra đây là cửa sổ thể hiện các asset chứa trong game. Ta có thể đăng kí và di chuyển asset bằng cách kéo thả trong cửa sổ này. Cấu trúc thư mục phân cấp giúp chúng ta dễ dàng hơn trong việc tổ chức và quản lí dữ liệu.

* + 1. **Cửa sổ Hierarachy**



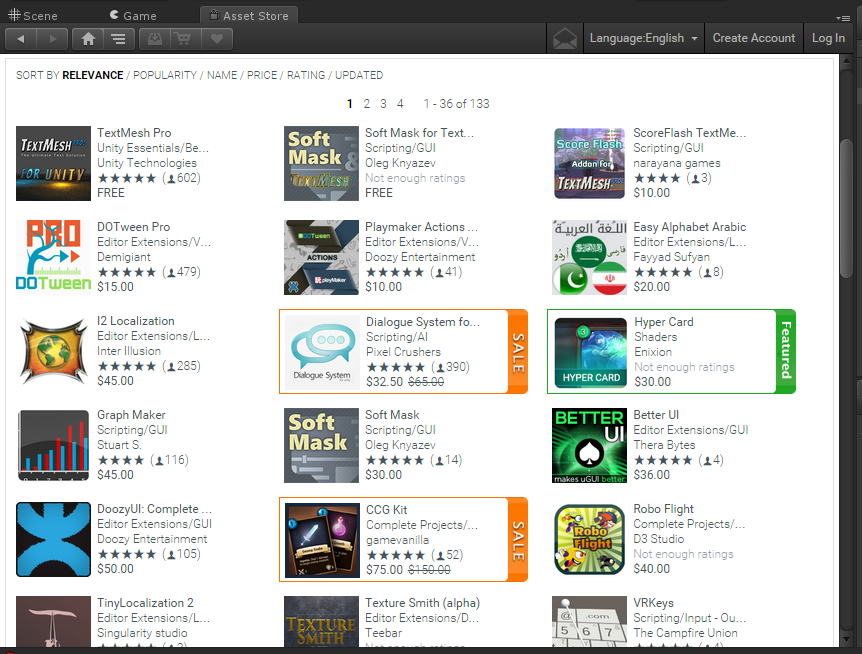
Đây là cửa sổ chứa tất cả các gameobject trong scene. Thể hiện theo cấu trúc phân cấp cha con. Muốn thêm gameobject mới hoặc các thao tác xóa sửa những gameobject sẽ diễn ra trong cửa sổ này.

* + 1. **Cửa sổ Inspector**



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**



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

* 1. **Làm quen với các thao tác cơ bản**

1. **Chương 3**
   1. **d**