Mục Lục

[LỜI MỞ ĐẦU 2](#_Toc495176980)

[Chương 1 3](#_Toc495176994)

[TỔNG QUAN LẬP TRÌNH GAME VỚI ENGINE UNITY 3](#_Toc495176995)

[1.1. CƠ SỞ LÝ THUYẾT 3](#_Toc495176996)

[1.1.1. Game Engine 3](#_Toc495176997)

[1.1.2. Tương tác người máy 6](#_Toc495176998)

[1.1.3. Lý thuyết trí tuệ nhân tạo 9](#_Toc495177036)

[1.2. ENGINE UNITY 11](#_Toc495177049)

[1.2.1. Giới thiệu Engine Unity 11](#_Toc495177050)

[1.2.2. Vì sao lại chọn Unity 14](#_Toc495177072)

[1.1.6. Cơ bản về giao diện 28](#_Toc495177181)

[1.1.7. Căn bản về lập trình trên Engine Unity 32](#_Toc495177214)

[Chương 2 37](#_Toc495177286)

[PHÂN TÍCH THIẾT KẾ HỆ THỐNG 37](#_Toc495177287)

[2.1. MÔ TẢ BÀI TOÁN 37](#_Toc495177288)

[2.1.1. Giới thiệu game 37](#_Toc495177289)

[2.1.2. Các quy luật chơi chính 38](#_Toc495177292)

[2.1.3. Đối tượng người dùng 38](#_Toc495177296)

[2.2. ĐẶC TẢ YÊU CẦU HỆ THỐNG 38](#_Toc495177298)

[2.2.1. Yêu cầu phi chức năng 38](#_Toc495177299)

[2.2.2. Yêu cầu  chức năng 39](#_Toc495177304)

[2.3. PHÂN TÍCH THIẾT KẾ HỆ THỐNG 39](#_Toc495177309)

[2.3.3. Biểu đồ lớp 42](#_Toc495177362)

[2.3.3. Biểu đồ lớp 50](#_Toc495177481)

[2.3.4. Biểu hoạt động 50](#_Toc495177483)

[Chương 3 54](#_Toc495177496)

[TRIỂN KHAI VÀ GIẢI PHÁP KHI XÂY DỰNG GAME 54](#_Toc495177497)

[3.1. XÂY DỰNG CÁC ĐỐI TƯỢNG CHÍNH TRONG GAME 54](#_Toc495177498)

[3.2. TẠO MAP 56](#_Toc495177510)

[3.3. TRÍ TUỆ NHÂN TẠO CHO ENEMY 58](#_Toc495177519)

[3.4. CÀI ĐẶT, THIẾT LẬP MÔI TRƯỜNG VÀ BUILD PROJECT 79](#_Toc495177784)

[KẾT LUẬN 92](#_Toc495177852)

[**TÀI LIỆU THAM KHẢO** 94](#_Toc495177877)

# LỜI MỞ ĐẦU

Lời đầu tiên, em xin chân thành cảm ơn sự hướng dẫn tận tình của thầy Nguyễn Việt Hùng, Bộ môn Công Nghệ Phần Mềm, Khoa Công Nghệ Thông Tin, Học Viện Kỹ Thuật Quân Sự.

Em xin chân thành cảm ơn các quý thầy, cô trong bộ môn Công Nghệ Phần Mềm cũng như các thầy cô trong trường đã giảng dạy, giúp đỡ em trong suốt 45 năm học qua. Nhờ các thầy, cô đã cung cấp cho chúng em những kiến thức nền tảng và kiến thức chuyên môn để em có thể hoàn thành luận văn và xa hơn là phục vụ cho công việc sau này.

Và tôi xin cảm ơn tất cả những người bạn của tôi, những người đã và đang luôn động viên, chia sẻ và giúp đỡ tôi tận tình trong suốt thời gian học tập cũng như thực hiện đồ án này.

Cho đến hôm nay, đồ án tốt nghiệp của em đã được hoàn thành cũng chính là nhờ công sức, tâm huyết của tất cả các thầy cô và các bạn đã tận tình giúp đỡ.

Trong thời đại công nghệ thông tin như hiện nay, sản phẩm công nghệ ngày càng chịu sự đánh giá khắt khe hơn từ phía những người dùng, đặc biệt là về sản phẩm Game được nhận rất nhiều sự đánh giá từ phía các Game thủ, hay chỉ là những người chơi bình thường. Ngành công nghiệp Game hiện nay có thể nói là bùng nổ, với tốc độ phát triển đến chóng mặt, rất nhiều game hay và hấp dẫn được ra đời trong thời gian qua. Phía sau những game phát triển và nổi tiếng như vậy đều có một Game Engine. Game Engine là một công cụ hỗ trợ, một phần mềm trung gian giúp người phát triển viết Game một cách nhanh chóng và đơn giản, đồng thời cung cấp khả năng tái sử dụng các tài nguyên và mã nguồn cao.

Tại Việt Nam, ngành công nghiệp Game mới chỉ đang phát triển chủ yếu vẫn là phát hành Game. Các công ty hiện nay đa số đều chỉ nhen nhóm những dự án nhỏ và kém hấp dẫn gây nên không ít thất vọng từ những Game thủ ủng hộ Game nước nhà.

Từ xu hướng phát triển và những bất cập trên, Em đã quyết định xây dựng đề tài: ***“Xây dựng game bắn súng phiêu lưu “Counter Terrorist 3D” chạy đa nền tảngtrên thiết bị di dộng với Unity”.***

Đồ án này sẽ khảo sát và nghiên cứu về Engine Unity – một Game Engine rất phổ biến và không kém mạnh mẽ  hiện nay nhằm thực nghiệm việc phát triển một trò chơi (Demo) thể loại phiêu lưu bằng chính Unity. Chuẩn bị kiến thức và kỹ năng cho định hướng nghề nghiệp (phát triển Game) sau này , góp phần vào sự phát triển của ngành công nghiệp Game nước nhà.

Do thời gian không có nhiều và kiến thức của em còn nhiều hạn chế nên không thể tránh khỏi thiếu sót hoặc sai lầm trong phát triển ứng dụng, em mong thầy cô và bạn bè góp ý giúp em ngày càng hoàn thiện Game.

Mọi góp ý xin gửi về email mhoangn95@gmail.com.

Một lần nữa, em xin chân thành cảm ơn !

Hà Nội, ngày 25 tháng  09 năm 2017

**Nguyễn Minh Hoàng**

# Chương 1

# TỔNG QUAN LẬP TRÌNH GAME VỚI ENGINE UNITY

## 1.1. CƠ SỞ LÝ THUYẾT

### 1.1.1. Game Engine

*1.1.1.1 Game Engine là gì:*

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 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](http://vi.wikipedia.org/w/index.php?title=2D&action=edit&redlink=1)) hay 3 chiều([3D](http://vi.wikipedia.org/wiki/3D)), công cụ vật lý (hay công cụ tính toán và phát hiện va chạm), âm thanh, mã nguồn, hình ảnh động (animation), trí tuệ nhân tạo, phân luồng, tạo dòng dữ liệu xử lý , quản lý bộ nhớ, dựng ảnh đồ thị, và kết nối mạng. 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.

*1.1.1.2 Tại sao lại cần đến Game Engine:*

Như chúng ta đã biết ở phần trước, việc xây dựng một ứng dụng đồ họa chỉ sử dụng thuần túy OpenGL sẽ tốn rất nhiều thời gian do chúng ta phải dùng từng hàm của OpenGL để vẽ các điểm, các đường thẳng, các mặt phẳng để tạo nên hình mẫu vật thể, cũng như phải tạo màu, tạo ánh sáng … cho môi trường. Để làm giảm thiểu thời gian này, người ta tạo ra các Engine , được xây dựng trên một nền tảng là OpenGL hoặc Direct3D, gồm các hàm trực quan hơn đối với người lập trình, giúp người lập trình tạo ra các vật thể, thiết lập môi trường, ánh sáng một cách đơn giản hơn và không cần viết các hàm bằng OpenGL nữa.

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 điều khiển luồng, quản lý bộ nhớ, điều khiển âm thanh… Nếu Engine chỉ có chức năng render hình ảnh thì thường được gọi là Graphic Engine.

*1.1.1.3 Tại sao cần lựa chọn Game Engine trước khi phát triển:*

Việc lựa chọn một Game Engine trước khi phát triển sẽ giúp chúng ta giảm thiểu những sai sót không đáng có trong quá trình phát triển phần mềm. Có một số tiêu chí chúng ta cần chú ý như sau:

***1/ Giá thành:*** Một số game Engine có giá thành cao, tuy có rất nhiều chức năng và khả năng hỗ trợ tốt. 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$ cho 1 Engine rõ ràng là một vấn đề lớn. Thông thường hiện nay có khá nhiều Engine không tính phí cho hiệu năng cao. Từ phần tiếp theo chúng ta sẽ chỉ bàn đến các Engine free này mà thôi.

***2/ 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 FPS (Frame Per Second) trong khi 1 Engine khác cho 30 FPS với cùng một cách lập trình.

***3/ 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.

***4/ 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.

***5/ 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.

***6/ 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.

Qua 6 điểm trên, chúng ta có thể đánh giá và đưa ra quyết định của mình khi sử dụng một Engine nào đó.

### 1.1.2. Tương tác người máy

*1.1.2.1 Nhu cầu thiết kế khi thay đổi công nghệ*

Để bán được nhiều, máy tính cần được thiết kế tốt hơn nữa, tiến đến máy tính tiện dùng có tri thức. Các nhà tâm lý, xã hội học nhất trí rằng tính phức tạp của các hệ thống tính toán hầu hết là do giao diện người dung với máy tính quá nghèo nàn. D.Norman cho rằng hai nguyên tắc đảm bảo giao diện tốt là:

- Tính hiển hiện: người dung cần thể hiện được các đối tượng và thể hiện là rõ rang.

- Tính tác động: là tính tự đề xuất các chức năng đã có.

*1.1.2.2 Thách thức của HCI*

Trong chục năm qua, với kiến thức hạn chế người ta vẫn cảm thấy có nhiều yêu cầu đối với thiết kế giao diện hiệu quả. Thiết kế HCI là thách thức do nền công nghệ phát triển nhanh, tốc độ phát minh, môi trường mạng, mạng tích hợp số ISDN. Hai thách thức quan trọng là: - Cách tồn tại chung với thay đổi của công nghệ.

- Cách đảm bảo thiết kế cho ra HCI tốt như là nền cho các công nghệ mới.

Thông qua quá trình tương tác với máy điện thoại, các thao tác nối đường, chuyển thong tin, ngắt đường truyền… người ta thấy sự cần thiết của giao diện tương tác trong đời sống.

*1.1.2.3 Đích của HCI*

Đích của việc nghiên cứu HCI là cho ra hệ thống dung được và an toàn. Đích được tóm tắt như là phát triển hay cải thiện tính an toàn, tiện dụng và hiệu qủa của hệ thống có máy tính, trong đó hệ thống có nghĩa gồm cả môi trường trong lí thuyết hệ thống. Tính tiện ích nhằm vào các chức năng của hệ thống, là những điều có thể làm được. Việc cải thiện tính hiệu quả là yếu tố tự than. 5 Để có giao diện sử dụng tốt, đặc tả HCI cần:

- Hiểu hết các nhân tố, như tâm lí, thực tiễn, tổ chức và nhân tố xã hội. Điều này cho phép xác định cách thao tác và tạo điều kiện cho công nghệ có hiệu quả

- Phát triển các công cụ và kĩ thuật để người thiết kế đảm bảo hệ thống máy tính phù hợp với các hoạt động của con người dung máy.

- Đạt được tương tác an toàn, hiệu quả cả về thuật ngữ tương tác người – máy và tương tác nhóm.

*1.1.2.4 Mục đích của HCI*

- Mô tả HCI: thông qua cách mô tả ta biết được một phần nội dung bên trong

- HCI quan trọng trong việc phát triển của công nghệ

- HCI lien quan đến hiệu suất , sự an toàn của người sử dụng

*1.1.2.5 Các thành phần của HCI*

- HCI thuộc đa lĩnh vực

- Các lĩnh vực hỗ trợ: công nghệ, triết học, ngôn ngữ học….

- Mô hình khái niệm của HCI

- Thiết kế HCI

*1.1.2.6 Khái niệm giao diện người dùng*

Giao diện người dùng là gì và nó đóng vai trò như thế nào trong tương tác người máy? Tương tác người máy, có thể định nghĩa một cách đơn giản, là tương tác xảy ra giữa con người và hệ thống máy tính. Giao diện người dùng là bộ mặt, hay còn gọi là thành phần trung gian để thực hiện giao tiếp, giữa con người với máy tính. Nó là nơi người sử dụng nhập thông tin vào hệ thống máy tính (đầu vào) và nhận thông tin phản hồi từ máy tính (đầu ra). Giao diện thì có rất nhiều loại khác nhau (như là bàn phím điện thoại, màn hình máy tính, màn hình của các bộ điều khiển VCR) nhưng về mặt cấu trúc thì đều giống nhau. Chúng đều bao gồm người dùng, hệ thống, đầu vào và đầu ra.

*1.1.2.7 Tại sao cần thiết kế giao diện*

Trong nhiều năm người sử dụng phàn nàn rằng các hệ thống máy tính khó sử dụng, ngôn ngữ khó hiểu và không nhạy bén. Trong khi đó người thiết kế lại là những người hầu như không gặp người sử dụng bao giờ và ngay cả khi gặp và nhận được những lời phàn nàn từ phía người sử dụng họ cũng không để tâm tới. Tình hình này đã được thay đổi từ khi các máy tính các nhân ra đời. Những người sử dụng trực tiếp điều khiển hệ thống, nhiều người có thể lập trình các ứng dụng riêng của mình và lúc này máy tính không chỉ được sử dụng ở văn phòng mà còn xâm nhập vào các gia đình. Vì vậy việc thiết kế một giao diện tốt trở nên quan trọng do :

* Thị trường (người sử dụng) lựa chọn phần mềm đóng gói sẵn: Người ta mua một sản phẩm là do họ thích hình dáng của nó và cái họ xem xét chính là giao diện.
* Người sử dụng phản đối các giao diện tồi: Đó là khi thị trường đặt ra các tiêu chuẩn thì người sử dụng không còn muốn dùng các giao diện tồi của hệ thống đã được thiết kế trước đây mà họ đàng sử dụng trong môi trường làm việc của mình.
* Giao diện tồi có thể dẫn tới việc không sử dụng được hoặc sử dụng không có hiệu quả hệ thống. Một giao diện tốt có thể hạn chế được nhược điểm của một thiết kế phần mềm tồi nhưng một giao diện tồi khiến không sử dụng được hệ thống ngay cả khi hệ thống đó có thiết kế phần mềm tuyệt vời. Một động cơ thúc đẩy nữa là tương tác người máy ngày càng được điều chỉnh nhanh chóng nhờ các tiêu chuẩn.
* Tổ chức tiêu chuẩn quốc tế ISO kết hợp với tổ chức quốc gia như BSI(Anh), ANSI(Mỹ), DIN(Đức) cho ra đời tiêu chuẩn áp dụng cho thiết kế tương tác người máy với số hiệu tiêu chuẩn ISO9241 51
* Viện tiêu chuẩn vương quốc Anh cấp chứng chỉ “Nhãn con diều hâu” đối với những sản phẩm có tương tác người máy tốt. Đối với những sản phẩm được cấp chứng chỉ này sẽ có ưu thế cạnh tranh trên thị trường. Mặt khác, về pháp lí, theo luật Anh và Châu Âu người thiết kế giao diện tồi có thể sẽ bị kiện ra tòa. Và trong tương lai luật này có thể trở thành luật của ISO về thiết kế tương tác người máy.

Do sự ra đời của các tiêu chuẩn và nhận thức của người sử dụng về các sản phẩm tin học nên số lượng các mã được viết cho đầu vào và đầu ra cho giao diện tăng lên, khoảng 50% mã được dùng cho xử lý giao diện trong một sản phẩm.

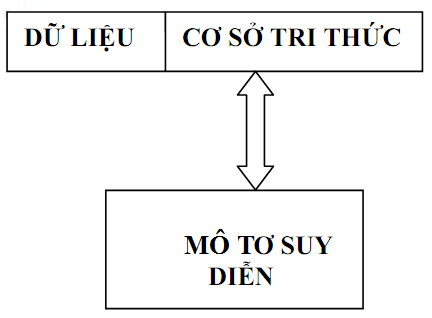
Người ta cũng nhận thấy rằng giao diện không chỉ là phần bị phê bình nhiều nhất của hệ thống mà còn là phần quan trọng và tốn nhiều chi phí nhất của nhiều chương trình nghiên cứu. Thông qua một số công trình đã được nghiên cứu, người ta nhận thấy rằng thiết kế giao diện tồi có thể gây ra những hậu quả như sau:

* Tần số mắc lỗi trong điều hành hệ thống và nhập liệu tăng lên. Để sửa lại các lỗi này thí sẽ tốn rất nhiều thời gian và tiền của, còn những lỗi không sửa được cho đúng có thể gây hậu quả có hại nếu các quyết định được đưa ra trên cơ sở dữ liệu không chính xác.
* Hệ thống hoạt động kém. Hệ thống có thể không thể quản lí được khối lượng chuyển tác đã được thiết kế, hoặc không đạt được độ chính xác của đầu ra, do hệ thống khố sử dụng và khó học.
* Hệ thống không sử dụng được do người sử dụng phản đối (Bộ quốc phòng Mỹ cho rằng sự thất bại của hệ thống là do thiết kế giao diện tồi và phân tích yêu cầu không đầy đủ).
* Người sử dụng thất vọng khi sử dụng hệ thống. Điều này làm cho năng suất làm việc thấp. Người sử dụng bị căng thẳng và trong trường hợp đặc biệt họ sẽ phá hủy hệ thống.

### 1.1.3. Lý thuyết trí tuệ nhân tạo

Trí tuệ nhân tạo (Artificial Intelligence) là trí tuệ được biểu diễn bởi bất cứ một hệ thống nhân tạo nào.

Chương trình trí tuệ nhân tạo được cấu tạo từ hai thành phần là **cơ sở tri thức** *(knowledge base)* và **mô tơ suy diễn** *(inference engine).*

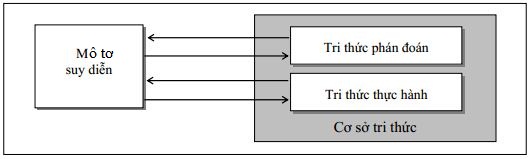


***Hình 23.  Cấu trúc chương trình trí tuệ nhân tạo***

Cơ sở tri thức là tập hợp các tri thức liên quan đến vấn đề mà chương trình quan tâm giải quyết. Cơ sở tri thức chứa các kiến thức được sử dụng để giải quyết các vấn đề (bài toán) trong trí tuệ nhân tạo.

Mô tơ suy diễn là phương pháp vận dụng tri thức trong cơ sở tri thức để giải quyết vấn đề. Là công cụ (chương trình hay bộ xử lý) tạo ra sự suy luận bằng cách quyết định xem những luật nào sẽ làm thỏa mãn các sự kiện, các đối tượng,… chọn ưu tiên các luật thỏa mãn, thực hiện các luật có tính ưu tiên cao nhất.

Quan hệ giữa **mô tơ suy diễn** và **cơ sở tri thức**



***Hình 24. Quan hệ giữa mô tơ suy diễn và cơ sở tri thức***

Trong một cơ sở tri thức, người ta phân biệt hai loại tri thức là *tri thức phán đoán* (assertion knowledge) và *tri thức thực hành* (operating knowledge).

Các tri thức phán đoán mô tả các tình huống đã được thiết lập hoặc sẽ được thiết lập. Các tri thức thực hành thể hiện những hậu quả rút ra hay những thao tác cần phải hoàn thiện khi một tình huống đã được thiết lập hoặc sẽ được thiết lập trong lĩnh vực đang xét. Các tri thức thực hành thường được thể hiện bởi các biểu thức dễ hiểu và dễ triển khai thao tác đối với người dùng.

Từ việc phân biệt hai loại tri thức trên, người ta nói mô tơ suy diễn là công cụ triển khai các cơ chế (hay kĩ thuật) tổng quát để tổ hợp các tri thức phán đoán và các tri thức thực hành.

## 1.2. ENGINE UNITY

### 1.2.1. Giới thiệu Engine Unity

*1.2.1.1. Giới thiệu chung*



***Hình 1: Logo của Engine Unity***

+   Nhà phát triển: Unity Technologies.

+   Phiên bản mới nhất: 5.3.4.

+   Được viết bởi ngôn ngữ:  C++,C#.

+   Phát triển Game cho các hệ điều hành: Windows, Mac OS X (tạo và đóng gói), Nintendo Wii, iPhone/iPad, Xbox 360, Android, Sony PlayStation 3 (chỉ đóng gói; cần giấy phép bổ sung cho từng nền tảng).

+   Giấy phép: Độc quyền.

+   Website: www.unity3d.com.

Unity được sự hỗ trợ của Just-In-Time Compilation (JIT), sử dụng thư viện mã nguồn mở C++ Mono. Bằng việc sử dụng JIT, những Engine như Unity có thể tận dụng lợi thế của tốc độ biên dịch. Ngoài thư viện Mono, Unity cũng tận dụng chức năng của những thư viện phần mềm khác vào chức năng của nó, như Engine mô phỏng vật lý PhysicX của Nvidia, OpenGL và DirectX cho kết xuất hình ảnh 3D, OpenAL cho âm thanh. Tất cả các thư viện này được xây dựng thành những tính năng tự động hoặc công cụ trực quan vào Unity, vì vậy chúng ta không cần phải lo lắng về việc học thế nào để sử dụng chúng một cách riêng lẻ.

Unity có một cộng đồng người dùng rất lớn mạnh, luôn chia sẻ những Plugins, công cụ của họ dưới phần mềm gói bổ sung.

*1.2.1.2. Sơ lược hình thành và phát triển*

· *Hình thành:*

Phần lõi của Unity ban đầu được viết bởi Joachim Ante vào năm 2001. Sau đó công ty được hình thành vào năm 2005 và bắt đầu với phiên bản 1.0. Đến năm 2007, Unity được nâng lên phiên bản 2.0. Unity bắt đầu hỗ trợ iPhone vào năm 2008. Vào tháng 6/2010, Unity chính thức hỗ trợ Android và cho ra đời phiên bản 3.0 có hỗ trợ Android vào tháng 9/2010. Có thể thấy tốc độ phát triển của Unity khá nhanh***.***

· *Giải thưởng:*

Unity đã đoạt được nhiều giải lớn với những giải chính sau:

+  Năm 2006, Unity đạt "Best Use of Mac OS X Graphics" tại Apple's WWDC. Đây là lần đầu tiên một công cụ phát triển game đạt được chất lượng do giải thưởng uy tín này đưa ra.

+  Năm 2009, Unity Technologies có tên trong "Top 5 công ty game của năm" do Gamasutra tổ chức.

+  Năm 2010, Unity đoạt giải Best Engine Finalist do Develop Magazine bình chọn, giải Technology Innovation Award của Wall Street Journal ở thể loại phần mềm

· *Khách hàng :*

Unity được trên 250.000 người đăng ký sử dụng gồm Bigpoint, Cartoon Network, Coca-Cola, Disney, Electronic Arts, LEGO, Microsoft, NASA, Ubisoft, Warner Bros, các hãng phim lớn nhỏ, các chuyên gia độc lập, sinh viên và những người đam mê.

### 1.2.2. Vì sao lại chọn Unity

*Unity là một engine game đa nền tảng*: Unity có  lợi thế rất lớn khi nó có thể xây dựng được game cho rất nhiều các nền tảng khác nhau như: Linux, Max, Windows , Windows 8, Android, iOs, Windows Phone 8, BlackBerry, HTML5 … Chính việc hỗ trợ các nền tảng cho PC phục vụ quá trình xây dựng game, và khả năng build cho 3 nền tảng di động phổ biến nhất đã tạo nên sức hấp dẫn của Unity. Về môi trường phát triển, Unity hỗ trợ các lập trình viên trên 3 ngôn ngữ: C#, Boo, JavaScript với các công cụ lập trình phổ biến như: Visual Studio, MonoDeveloper.

*Unity  là một engine game mã nguồn mở:* Nó được xây dựng hoàn toàn bởi cộng đồng phát triển game. Unity hoàn toàn miễn phí, luôn luôn được chỉnh sửa, phát triển, toàn bộ mã nguồn của Unity có thể tải trên trang chủ.

*Unity  là một engine game có hiệu suất cao*: Được xây dựng trên nền tảng đồ ngôn ngữ lập trình C++ và C# nên Unity có hiệu suất khá cao. Đặc biệt, trên các nền tảng di động, Unity tỏ rõ lợi thế của mình với độ tương thích cao cho các nền tảng di động: Android, IOS, Windows Phone 8/8.1/10. Với các nền tảng này, game với khối lượng đồ họa dạng trung bình, chạy trên cấu hình di động tầm trung hiện nay, có thể đạt được  60FPS – một con số khá cao.

*Unity  là một engine game kiểm chứng bởi cộng đồng*:  Hiện tại, có khoảng 10000 game được phát triển trên nền tảng Unity, 25% game 2D trên thị trường sử dụng Unity, với những game khá thành công như: Monument Valley, Tiger Woods PGA Tour, Temple Run 2, Global Conflicts … Hiện tại, ở Việt Nam, khá nhiều công ty làm game đã sử dụng Unity là nền tảng chính cho phát triển game, tiêu biểu như:  VTC, VNG, Glass-Egg…

### 1.2.3. Các đặc điểm và tính năng của Unity

*1.2.3.1. Rendering (Kết xuất hình ảnh)*

Giống  như tất cả các Engine hoàn chỉnh khác, Unity hỗ trợ đầy đủ khả năng kết xuất hình ảnh cùng nhiều hỗ trợ cho phép áp dụng các công nghệ phổ biến trong lĩnh vực 3D nhằm cải thiện chất lượng hình ảnh.

Một số hỗ trợ:

-  Unity cung cấp 100 Shaders với đầy đủ các loại phổ biến nhất.

-  Hỗ trợ Surface Shaders, Occlusion Culling, GLSL Optimizer.

-  Hỗ trợ LOD*.*

*1.2.3.2. Lighting (Ánh sáng)*

Unity cung cấp nhiều giải pháp đa dạng cho phép chúng ta áp dụng ánh sáng một cách tốt nhất vào môi trường trong trò chơi với nhiều loại nguồn sáng như ánh sáng có hướng (Directional Light), ánh sáng điểm (Point Light) ,...

Một số công nghệ và kỹ thuật về ánh sáng được Unity hỗ trợ: Lingtmapping, Realtime Shadows, hiệu ứng Sunshafts và Lens Flares.

*1.2.3.3. Terrains (Địa hình)*

Unity cung cấp một công cụ rất tốt khả năng này với tên gọi là Terrains Tools cho phép chúng ta thiết kế địa hình với các công cụ vẽ dưới dạng Brush có nhiều thông số tùy chỉnh để tạo và lát Texture cho địa hình. Cùng với Terrains Tools là Tree Creator, một công cụ mạnh mẽ cho phép chúng ta tạo ra cây cối với hình dạng, kích thước và kiểu cách đa dạng.

*1.2.3.4. Substances (Texture thông minh)*

Substances cho phép các trò chơi và các ứng dụng sáng tạo sử dụng tất cả các sức mạnh của nó để cung cấp những trải nghiệm độc đáo cho người dùng cuối:

- Được tạo ra ngay trong khi game được tạo

- Cung cấp nội dung tải về nhỏ

- Sử dụng sức mạnh của các textures động tạo ra các hiệu ứng hình ảnh độc đáo

Substances được xử lý giống như các bitmap khác trong Unity: chỉ cần kéo và thả nó trong project là đã có thể sẵn sàng sử dụng ngay. Substances có kích thước tự nhiên nhỏ hơn bitmap nên khi sử dụng Substances thay cho bitmap sẽ làm giảm đáng kể kích thước của game cuối cùng.

*1.2.3.5. Physics (Vật lý)*

PhysX là một Engine mô phỏng và xử lý vật lý cực kì mạnh mẽ được phát triển bởi nhà sản xuất card đồ họa hàng đầu thế giới NVIDIA. Unity đã tích hợp Engine này vào để đảm nhận mọi vấn đề về vật lý.

*1.2.3.6. Pathfinding (Tìm đường)*

Đây là tính năng mới mẻ từ phiên bản Unity 3.5. Với các phiên bản trước, để phát triển khả năng tìm đường cho trí thông minh nhân tạo (AI), nhà phát triển phải hoàn toàn tự xây dựng cho mình một hệ thống tìm đường riêng biệt.

*1.2.3.7. Audio (Âm thanh)*

Unity tích hợp FMOD – công cụ âm thanh thuộc hàng mạnh nhất hiện nay, hỗ trợ chúng ta nhập và sử dụng nhiều định dạng tập tin âm thanh khác nhau.

*1.2.3.8. Progamming (Lập trình)*

Unity được viết trên nền C++, tuy nhiên để sử dụng Unity, người dùng chỉ cần biết 1 trong 3 ngôn ngữ là C#, JavaScript, hoặc Boo, với JavaScript là đơn giản và dễ sử dụng nhất, tuy nhiên với 1 số hạn chế nên C# được đa số chuyên gia ưa chuộng hơn, ngôn ngữ Boo được cho là tốt nhất, tuy nhiên do ngôn ngữ được rất ít người sử dụng, nên gần như không bao giờ thấy tài liệu hoặc code liên quan đến Unity được viết bằng ngôn ngữ Boo.

*1.2.3.9. Networking*

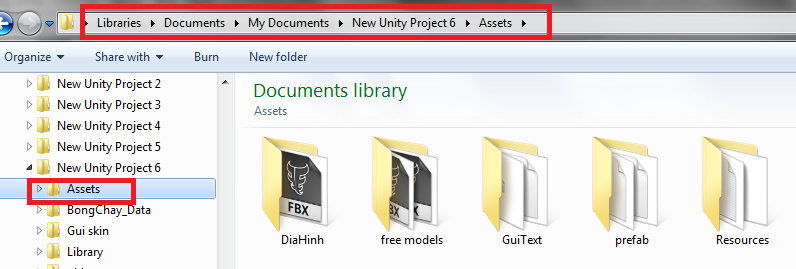
Networking cho phép chúng ta tạo ra các trò chơi trực tuyến (online), hỗ trợ đầy đủ giúp ta tạo nên các khía cạnh phổ biến trong Game online như hệ thống điểm kinh nghiệm, chat và tương tác với thời gian thực, ...

Tài nguyên (model, âm thanh, hình ảnh, ...) được tải vào trong Unity và tự động cập nhật nếu tài nguyên có sự thay đổi. Unity hỗ trợ các kiểu định dạng từ 3DS Max, Maya, Blender, Cinema 4D và Cheetah3D.

### 1.2.4. Các thành phần cơ bản trong Unity ứng dụng vào game

*1.2.4.1. Asset*

Đây là kho tài nguyên cho việc xây dựng game trong một project của Unity. Các tài nguyên này có thể là hình ảnh, âm thanh, hoặc một mô hình 3D có sẵn. Unity sẽ tham chiếu đến các tập tin chúng ta sẽ sử dụng để tạo ra các tài nguyên cho trò chơi. Đây là lý do tại sao trong bất kỳ thư mục chứa project sử dụng Unity thì tất cả các tập tin tài nguyên phải được lưu trữ trong một thư mục con tên là Assets.

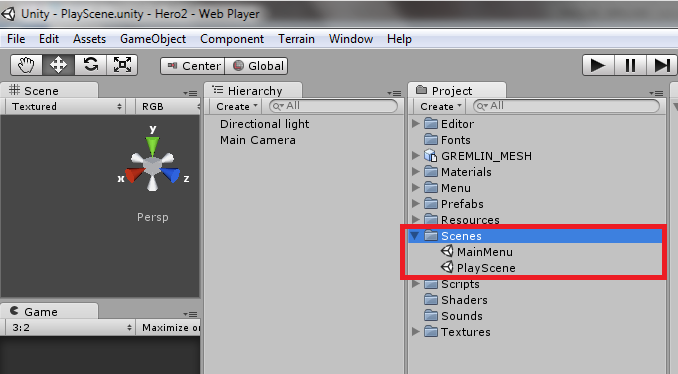


***Hình 2:  Asset trong Unity***

*1.2.4.2. Scene*

Trong Unity, chúng ta có thể xem Scenes là các màn chơi, cấp độ chơi riêng lẻ, hoặc các vùng của nội dung trò chơi. Ví dụ như Main menu, Options, About …

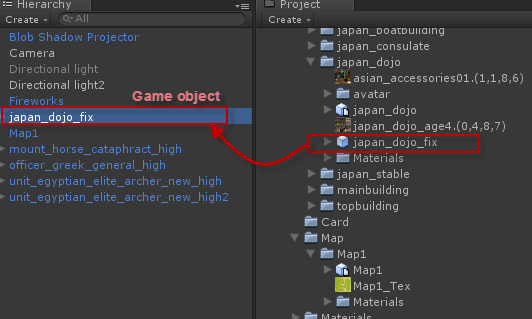
Bằng cách xây dựng trò chơi với nhiều cảnh, chúng ta sẽ có thể phân phối thời gian tải và thử nghiệm các phần khác nhau của trò chơi riêng lẻ một cách nhanh chóng và chính xác.



***Hình 3:  Các scene của Unity***

*1.2.4.3. Game Object*

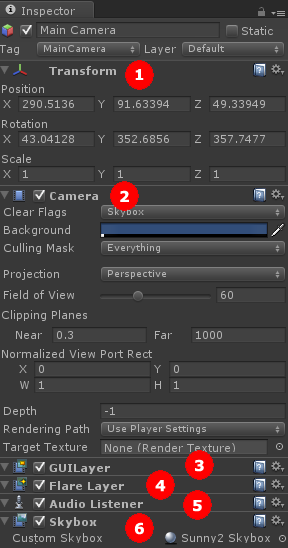
Khi Asset được sử dụng trong Scene, chúng trở thành Game Object – một thuật ngữ được sử dụng trong Unity (đặc biệt là trong mảng lập trình). Tất cả các Game Object đều chứa ít nhất một thành phần là Transform. Transform là thông tin về vị trí, góc quay và tỉ lệ của đối tượng, tất cả được mô tả bởi bộ 3 số X, Y, Z trong hệ trục tọa độ. Thành phần này có thể được tùy biến lại trong quá trình lập trình nhằm thay đổi vị trí, góc quay và tỉ lệ của đối tượng qua các đoạn mã. Từ các thành phần cơ bản này, chúng ta sec tạo ra Game Object với các thành phần khác, bổ sung chức năng cần thiết để xây dựng nên bất kì một thành phần nào trong kịch bản Game mà chúng ta đã tưởng tượng.



***Hình 4:  Kéo tài nguyên vào Scene để sử dụng***

*1.2.4.4. Component*

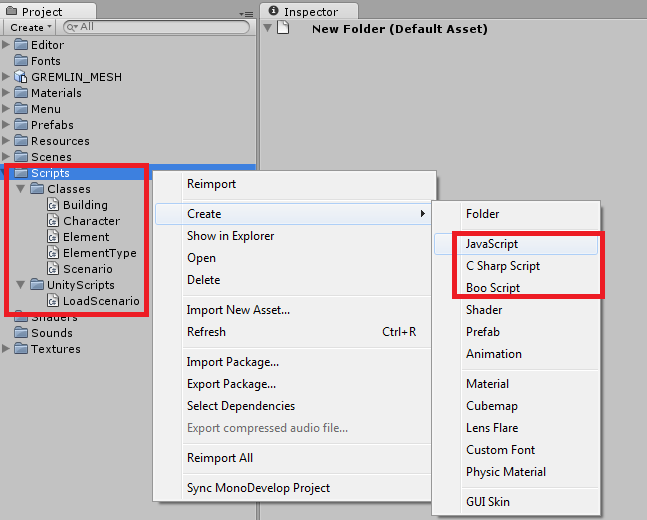
Component là các thành phần trong một Game Object của Unity. Bằng cách đính kèm các thành phần vào cho một đối tượng, chúng ta có thể áp dụng ngay các phần mới của game engine vào đối tượng. Thông thường các thành phần này được Unity xây dựng sẵn như ánh sáng, camera, particle, hiệu ứng vật lý…



***Hình 5:  Các thành phần trong đối tượng Camera***

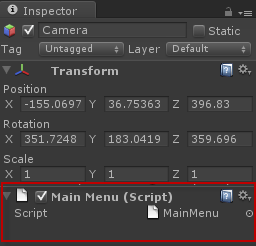
*1.2.4.5. Script*

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



***Hình 6:  Cách tạo file script mới***

Một đoạn script muốn thực thi được thì nó phải được gắn vào một đối tượng



***Hình 7:  Một file script đang gắn vào đối tượng***

*1.2.4.6. Prefab*

Hãy tưởng tượng Prefab là một cái thùng rỗng, mà bên trong nó chúng ta có thể chứa đựng các thành phần hay đối tượng khác nhau, chúng ta có thể viết kịch bản cho hành động của Prefab (khởi tạo, di chuyển, hay hủy đối tượng). Chúng ta có thể sử dụng đối tượng này nhiều lần trong trò chơi, và cũng có thể sử dụng lại cho project khác. Prefab cho phép chúng ta lưu trữ cá đối tượng, toàn bộ thành phần bên trong và cấu hình hiện tại.

*1.2.4.7. Particals System*



***Hình 8. Hiệu ứng khói, lữa-một ví dụ đại diện cho partical***

Particales là một thành phần rất quan trọng và không thể thiếu trong các trò chơi. Partical là tập hợp các hạt được đính kèm các hình ảnh, vật liệu với hình dáng và màu sắc cụ thể nhằm mô phỏng một hiệu ứng nào đó trong thực tế, ví dụ như hiệu ứng lửa, khói, hiệu ứng nổ của một vật nào đó…Partical trong Unity được giới thiệu dưới dạng một hệ thống bao gồm các Components đảm nhận các vai trò khác nhau.

*1.2.4.8. Animations*

Animation là tập hợp các hình ảnh của một nhân vật hay sự vật nào đó mà khi chúng hiển thị liên tiếp lên trên cùng một vị trí sẽ làm ta thấy nhân vật hay sự vật đó dường như đang chuyển động.



***Hình 9.  Minh họa 1 Animation***

Trong  Unity  ta có thể tạo animation bằng cách vẽ nhiều sprite liên tiếp với nhau hoặc dùng file .plist. Tuy nhiên, trong đây tôi sẽ giới thiệu về cách sử dụng animation bằng file .plist.

Ưu điểm khi sử dụng file .plist là sẽ làm cho việc quản lí hình ảnh dễ dàng hơn, đọc ảnh nhanh hơn, load ảnh mượt. Tuy nhiên vẫn còn một số hạn chế là sẽ phải load hết tất cả các ảnh vào cache, nếu khi nào dùng mới load thì nó cũng không khác gì việc load từ một ảnh gốc, làm cho việc load chậm hơn. Vì vậy, nên thực hiện  load hết tất cả lúc đầu, và chỉ remove khi không còn dùng đến cái ảnh ấy một lần nữa.

Nội dung file.plist là một file định dạng xml lưu trữ mọi thông tin về bức ảnh mà ta cần để load lên. Mỗi frame và animation trong đó sẽ mang một giá trị “key” riêng để ta có thể dựa vào đó mà vẽ chúng lên Layer.

*1.2.4.9. Tile map*

Tilemap là kỹ thuật làm map được hiểu là màn hình của người chơi bao gồm nhiều mảnh nhỏ ghép lại. Các mảnh này được lưu trong 1 file cấu hình và được nạp vào game thành 1 layer. Tilemap được dùng phổ biến trong hầu hết các game vì tính linh hoạt của nó.

Có 3 loại Tilemap: Orthogonal, Isometric và Hexagonal.

|  |  |  |
| --- | --- | --- |
| Description: http://www.cocos2d-x.org/attachments/download/1647 | 1648 | 1651 |
| Orthogonal map | Isometric map | Hexagonal map |

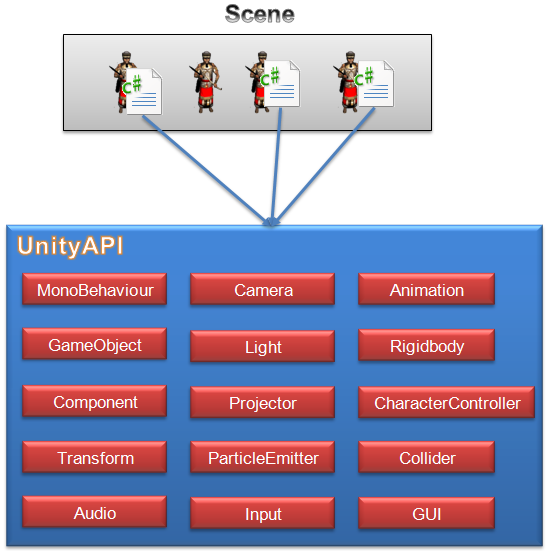
***Hình 10.  Các loại tilemap***

### 1.2.5. Kiến trúc Engine Unity

*1.2.5.1.  Kiến trúc tổng quan*

Engine Unity hỗ trợ cho chúng ta UnityAPI để viết script game. UnityAPI là API lập trình game trong Unity rất mạnh. UnityAPI chứa các đối tượng và phương thức hỗ trợ hầu hết các đối tượng và các loại thành phần trong Unity.

Trong một scene thường có nhiều đối tượng game. Mỗi đối tượng này có thể có hoặc không có đoạn script nào gắn lên đó. Nếu muốn gắn script vào đối tượng, ta bắt buộc phải kế thừa class đó từ lớp **MonoBehaviour** của UnityAPI và tên class phải trùng với tên file script. Mỗi script khi gắn lên đối tượng game đều được đối tượng game xem như một thành phần bên trong và được cấp phát vùng nhớ khi chạy game.



***Hình 11.  Tổng quan kiến trúc Unity***

Bên trong UnityAPI chứa rất nhiều lớp hỗ trợ lập trình game, trong đó có một số lớp quan trọng như :

·        *MonoBehaviour*: tất cả các script muốn gắn vào một đối tượng game bắt buộc phải kế thừa từ lớp này.

·        *GameObject*: lớp cha của tất cả các thực thể trong scene.

·        *Component*: lớp cha của tất cả các thành phần có thể gắn vào đối tượng.

·        *Transform*: giúp thay đổi vị trí, xoay, biến đổi tỉ lệ mô hình.

·        *Input*: hỗ trợ lập trình với chuột, cảm ứng đa điểm, cảm biến gia tốc.

·        *Camera*: giúp lập trình camera.

·        *Light*: giúp tạo ánh sáng trong game.

·        *Projector*: giúp chiếu texture lên bề mặt vật thể.

·        *ParticleEmitter*: hỗ trợ tạo các hiệu ứng particle đẹp mắt.

·        *Audio*: hỗ trợ lập trình với âm thanh.

·        *Animation*: chạy chuyển động của mô hình nhân vật.

·        *Rigidbody*: giúp tạo hiệu ứng vật lý liên quan đến trọng lực như bóng nẩy, lăn, ..

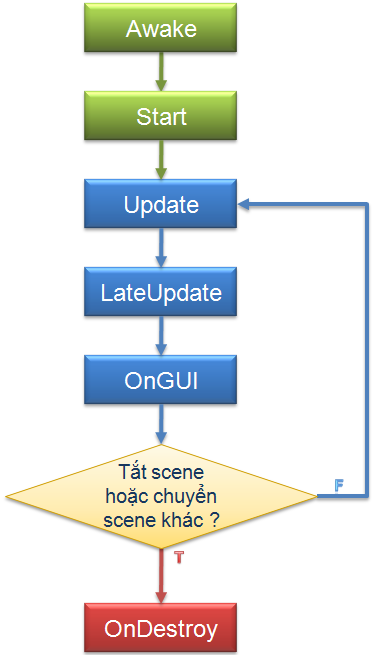
·        *CharacterController*: giúp điều khiển nhân vật di chuyển theo độ cao địa hình.

·        *Collider*: hỗ trợ lập trình va chạm giữa các vật thể.

·        *GUI*: giúp lập trình giao diện người dùng trên Unity.

*1.2.5.2. Chu kỳ sống của thành phần Script gắn trên đối tượng Game*

Chu kỳ sống của một thành phần script được tính kể từ khi scene được chạy cho đến lúc scene bị tắt hoặc chuyển sang scene khác.



***Hình 12.  Chu kỳ sống của thành phần script***

Tùy theo trạng thái của scene mà sự kiện tương ứng sẽ được gọi.

·        **Awake**: được gọi khi script được load xong

·        **Start**: được gọi khi script được load xong nếu thành phần script đó không bị vô hiệu hóa.

·        **Update**: đây là sự kiện thường sử dụng nhất và được gọi liên tục từng frame (khung) để vẽ lại màn hình.

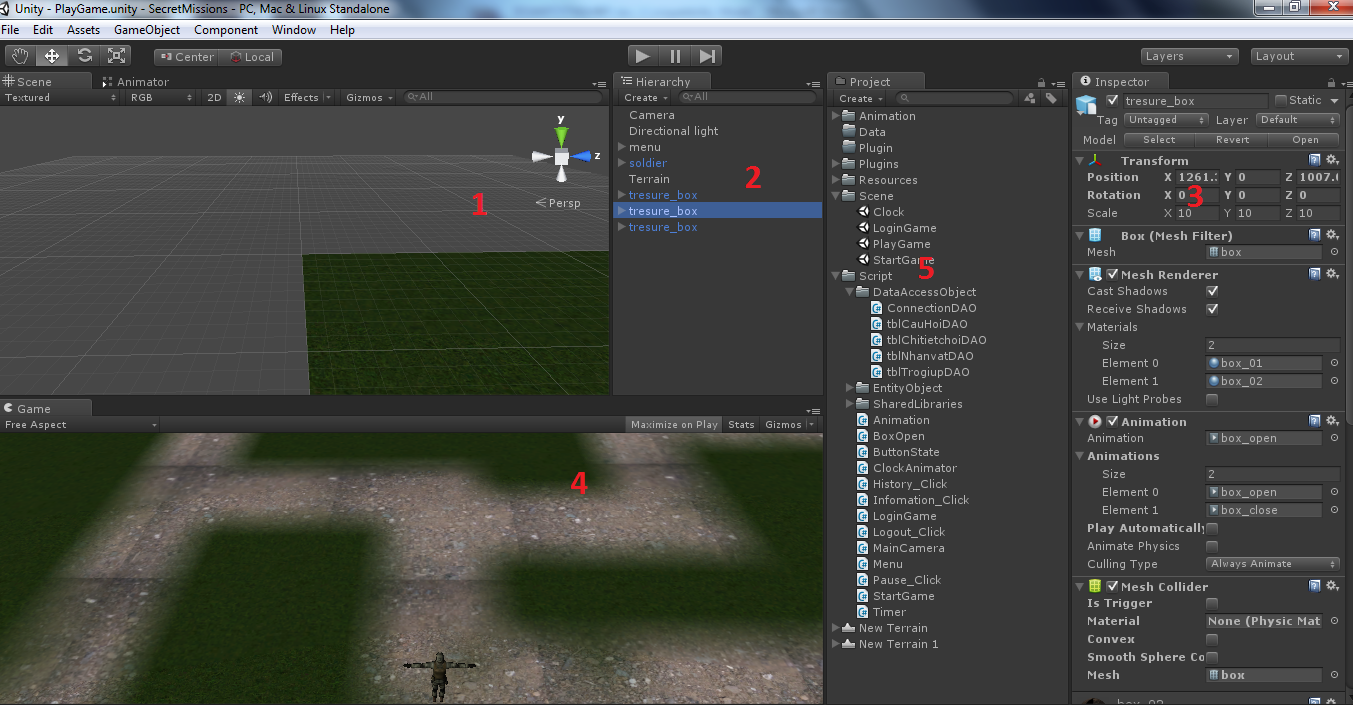
·        **LateUpdate**: sự kiện này chạy sau sự kiện Update và được gọi liên tục từng frame.

·        **OnGUI**: sự kiện này dùng để vẽ GUI và được gọi liên tục từng frame, chỉ trong sự kiện này ta mới có thể sử dụng các lớp hỗ trợ tạo giao diện người dùng của UnityAPI.

·        **OnDestroy**: được gọi khi thành phần script bị hủy khỏi bộ nhớ.

### 1.1.6. Cơ bản về giao diện

Giao diện Unity, giống như nhiều môi trường làm việc khác, layout có thể tùy chỉnh. Layout của Unity bao gồm nhiều tab khác nhau và có thể bật tắt. Dưới đây là một kiểu bố trí điển hình trong Unity.



***Hình 13.  Giao diện của Unity***

\*        **Chú thích**

-         Scene [1] – nơi xây dựng trò chơi.

-         Hierarchy [2] – danh sách các đối tượng trong một cảnh game.

-         Inspector [3] – màn hình cài đặt cho tài nguyên/đối tượng đang được chọn.

-         Game [4] – cửa sổ xem trước game, chỉ hoạt động ở chế độ chơi (khi nhấn Play).

-         Project [5] – danh sách các tài nguyên trong project, đóng vai trò như một thư viện.

*1.3.7.1. Cửa sổ Scene và Hierarchy*

Cửa sổ Scene là nơi chúng ta sẽ xây dựng toàn bộ các đối tượng trong game. Cửa sổ cung cấp nhiều góc nhìn khác nhau, có thể nhìn dạng phối cảnh hoặc dạng song song. Chúng ta có thể kéo thả đối tượng trên cửa sổ này, di chuyển, xoay…



***Hình 14. Các nút chúc năng cho cửa sổ Scene***

Cửa sổ Scene cũng kèm theo bốn nút điều khiển tiện lợi như hình trên. Truy cập từ bàn phím bằng cách sử dụng các phím Q, W, E, và R. Các phím thực hiện các hoạt động sau đây:

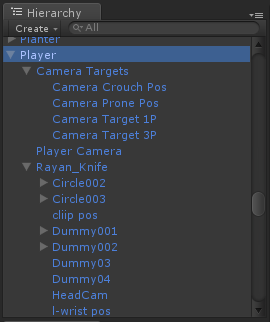
· Công cụ bàn tay [Q]: công cụ này cho phép di chuyển trong cửa sổ Scene, xoay góc nhìn, phóng to, thu nhỏ góc nhìn.

· Công cụ di chuyển [W]: Công cụ này dùng để di chuyển một đối tượng.

· Công cụ xoay [E]: Công cụ này cho phép chúng ta xoay nhân vật theo một trục nào đó trong không gian.

· Công cụ tỷ lệ [R]: Công cụ này cho phép chúng ta tăng giảm tỷ lệ kích thước của đối tượng.

Khi chọn một đối tượng trong cửa sổ Scene, đối tượng này sẽ được tự động chọn trong cửa sổ Hierarchy và ngược lại.

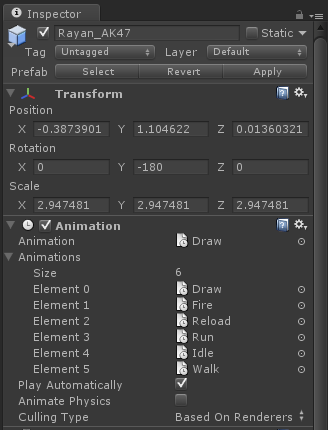


***Hình 15. Cửa sổ Hierarchy***

*1.3.7.2.**Cửa số**Inspector*

Cửa sổ Inspector có thể được xem như một công cụ cho phép chúng ta tùy chỉnh các thiết đặt, các thành phần của GameObject hoặc Asset đang được chọn. Nó hiển thị đầy đủ các Component của đối tượng mà chúng ta chọn. Nó cho phép chúng ta điều chỉnh các biến của Components dưới các hình thức như: Textbox, Slider, Button, Drop-down Menu…

Ngoài việc hiển thị các Component của đối tượng được chọn, cửa sổ Inspector còn hiển thị các thiết đặt chung của hệ thống hay của trò chơi khi ta chọn chúng từ menu Edit.



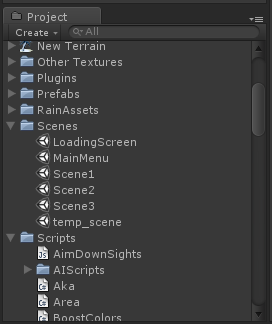
***Hình 16. Cửa sổ Inspector***

*1.3.7.3. Cửa sổ Game*

Cửa sổ này sẽ hiển thị những gì có trong cửa sổ Scene và sẽ hoạt động khi nhấn nút Play. Trong cửa sổ này chúng ta có thể chọn các kích cỡ hiển thị khác nhau để build cho các loại máy khác nhau. Chúng ta có thể chơi thử game trên cửa sổ này khi đã nhấn nút Play. Lưu ý rằng khi cửa sổ này hoạt động rồi thì mọi chỉnh sửa trên cửa sổ Scene và cài đặt cho các đối tượng chỉ là tạm thời và khi nhấn nút Stop, cửa sổ này về lại trạng thái tĩnh thì mọi chỉnh sửa trước đó là không còn.

*1.3.7.4. Cửa số Project*

Cửa sổ Project thể hiện nội dung bên trong thư mục Assets của project chúng ta. Khi thêm tài nguyên vào thư mục Assets ngay lập tức chúng sẽ tự động được cập nhập vào project Unity của chúng ta.



***Hình 17. Cửa sổ Project***

### 1.1.7. Căn bản về lập trình trên Engine Unity

Engine Unity hỗ trợ các ngôn ngữ C#, Javascript, Boo mỗi ngôn ngử có ưu, nhược điểm khác nhau tại một hoàn cảnh nhất định. Các đoạn code được viết bằng C# sẽ được thực thi nhanh hơn, sử dụng C# là một trong các biện pháp tối ưu hóa cho Unity Game. Bên cạnh đó, C# có điểm mạnh về mặt “thoáng” khi trao đổi dữ liệu giữa các Component. Các Component khác nhau được viết bằng các ngôn ngữ khác nhau vẩn có thể trao đổi thông tin và được truy cập bình thường , điều này có nghĩa là developer được thoải mái trong việc chọn ngôn ngử để giải quyết vấn đề.

*1.1.7.1.  Lớp components*

Là lớp cơ bản nhất được tích hợp vào các GameObject (danh từ chỉ các đối tượng trong game), lớp Component bao gồm các thuộc tính lưu trử những thông tin cơ bản và quan trọng nhất đối với một GameObject như vị trí của đối tượng trong không gian 3D, độ xoay của đối tượng, tên, tag…vv…

|  |  |
| --- | --- |
| **Thuộc tính** | **Chú thích** |
| transform | Thông tin về vị trí, độ xoay |
| rigidbody | Giả lập khối lượng và trọng lượng cho GameObject |
| camera | Truy xuất đến các thuộc tính của class Camera nếu GameObject được tích hợp một camera |
| light | Truy xuất đến các thuộc tính của class Light nếu GameObject được tích hợp một Light |
| animation | Truy xuất đến các thuộc tính của class Animation nếu GameObject được tích hợp các Animation |
| constantForce | Truy xuất đến các thuộc tính của class ConstantForce nếu GameObject được tích hợp ConstantForce |
| renderer | Truy xuất đến các thuộc tính của class Animation nếu GameObject được tích hợp các Animation |
| audio | Truy xuất đến các thuộc tính của class Audio nếu GameObject được tích hợp Audio |
| guiText | Truy xuất đến các thuộc tính của class GUIText nếu GameObject được tích hợp GUIText |
| networkView | Truy xuất đến các thuộc tính của class NetworkView nếu GameObject được tích hợp NetworkView |
| guiTexture | Truy xuất đến các thuộc tính của class GUITexture nếu GameObject được tích hợp GUITexture |
| collider | Truy xuất đến các thuộc tính của class Collider nếu GameObject được tích hợp Collider (được dùng để kiểm tra các va chạm). |
| hingeJoint | Truy xuất đến các thuộc tính của class HingeJoint nếu GameObject được tích hợp HingeJoint |
| particleEmitter | Truy xuất đến các thuộc tính của class ParticleEmitter nếu GameObject được tích hợp ParticleEmitter |
| particleSystem | Truy xuất đến các thuộc tính của class ParticleSystem nếu GameObject được tích hợp ParticleSystem |
| gameObject | Truy xuất đến đối tượng mà nó tích hợp đến |
| tag | Tag của component |

*Chú ý*: Mặc định khởi tạo một đối tượng, đối tượng đó sẽ mang tên “GameObject” và được tích hợp sẵn thành phần Transform.

|  |  |
| --- | --- |
| **Phương thức** | **Chú thích** |
| GetComponent | Trả về kiểu Component được tích hợp vào đối tượng |
| GetComponentInChildren | Trả về kiểu Component được tích hợp vào đối tượng, hoặc con của đối tượng |
| GetComponentsInChildren | Giống GetComponentsInChildren nhưng trả về nhiều Component |
| GetComponents | Giống GetComponent nhưng trả về nhiều Componet |
| CompareTag | Đối tượng có thuộc tag nào không ? |
| SendMessageUpwards | Gửi yêu cầu thực hiện một phương thức đến các thành phần được tích hợp trong cùng một đối tượng |
| SendMessage | Gửi yêu cầu thực hiện một phương thức đến các thành phần được tích hợp trong cùng một đối tượng |
| BroadcastMessage | Gửi yêu cầu thực hiện một phương thức đến các đối tượng con |

*1.1.7.2.  Các phương thức cơ bản*

Khi khởi tạo một script, mặc định nó đã chứa các phương thức Start(), Update(), đây là hai phương thức rất hữu dụng, ngoài ra còn một số phương thức khác được liệt kê dưới đây:

- **Update():** Những đoạn code thuộc phương thức này sẽ được gọi lại mỗi frame.

- **FixedUpdate():** Giống với Update nhưng ta có thể hiệu chỉnh được số khung hình ban đầu, và không bị ảnh hưởng khi tần số frame ổn định.

- **Awake():** Được gọi khi script ở giai đoạn khởi tạo, thường được dung để thiết lập hoặc tải dữ liệu ban đầu cho các Component.

- **Start():** Được gọi thực hiện trước phương thức Update() nhưng lại sau Awake(), Khác nhau cơ bản Start và Awake là phương thức Start() chỉ được gọi khi script không bị đình chỉ hoạt động

- **OnCollisionEnter():** Thực thi khi đối tượng chứa nó bao gồm một Collider và va chạm với một Collider hoặc Rigibody của một đối tượng khác.

- **OnMouseDown():** Thực thi khi chuột của người chơi click vào một đối tượng có tích hợp thành phần GUIElement hoặc Collider.

*1.1.7.3.  Debugging*

Unity cung cấp lớp Debug để hổ trợ lập trình viên theo dõi và kiểm soát các lỗi, ở đây chúng ta quan tâm đến phương thức Debug.Log().

Phương thức Log() cho phép người dùng gửi đến một thông tin đến Unity Console nhằm mục đích :

- Chứng minh rằng đoạn mã này đang được thực hiện.

- Báo cáo tình trạng hiện tại của biến

# Chương 2

# PHÂN TÍCH THIẾT KẾ HỆ THỐNG

## 2.1. MÔ TẢ BÀI TOÁN

### 2.1.1. Giới thiệu game

**“Counter Terrorist 3D”** là game bắn súng góc nhìn thứ nhất với khung cảnh 3D với các tòa nhà, công trình … nơi mà là căn cứ cũng như nơi bọn khủng bố xuất hiện. Người chơi sẽ là một người lính, một chiến sĩ một mình đột phá và tiêu diệt hết bọn khủng bố với các vũ khí được trang bị như súng ống, dao, lựu đạn,…

Trò chơi sẽ có nhiều màn chơi và bản đồ khung cảnh đa dạng , với nhiệm vụ từng màn và hệ thống tính sao sẽ đem lại cho người chơi niềm vui cũng như những giây phút kịch tính đầy nghẹt thở, cuốn hút theo từng màn chơi.

### 2.1.2. Các quy luật chơi chính

**Game** sẽ có 2 lối chơi chính là single mode và endless mode.

**Single mode** là kiểu chơi theo màn, người chơi sẽ chơi theo từng màn , tiêu diệt hết kẻ địch và làm các nhiệm vụ được giao để kiếm sao cho mình, với số sao kiếm được sẽ giúp người chơi mở dần các màn tiếp theo.

**Endless mode** là kiểu chơi sinh tồn, khi người chơi chỉ chơi một bản đồ khung cảnh duy nhất, kẻ địch xuất hiện theo từng đợt và người chơi cố gắng tiêu diệt để sống sót càng lâu càng tốt.

### 2.1.3. Đối tượng người dùng

Đối tượng người dùng mà game hướng đến mong muốn là tất cả mọi lứa tuổi, những người cần niềm vui, sự giải trí sau mỗi giờ học tập làm việc căng thẳng. Người chơi không chỉ tìm được niềm vui mà còn giúp tăng khả năng phản xạ cũng như sự tập trung.

## 2.2. ĐẶC TẢ YÊU CẦU HỆ THỐNG

### 2.2.1. Yêu cầu phi chức năng

- Thiết bị sử dụng hệ điều hành Windows, Windows phone hoặc Androi, có kết nối internet.

- Đồ họa trong game dễ nhìn.

- Game chạy mượt, không giật.

- Người chơi có thể dễ dàng chạm vào màn hình để thực hiện các thao tác.

### 2.2.2. Yêu cầu  chức năng

- Sử dụng chạm màn hình để thao tác các hoạt động khi chơi game.

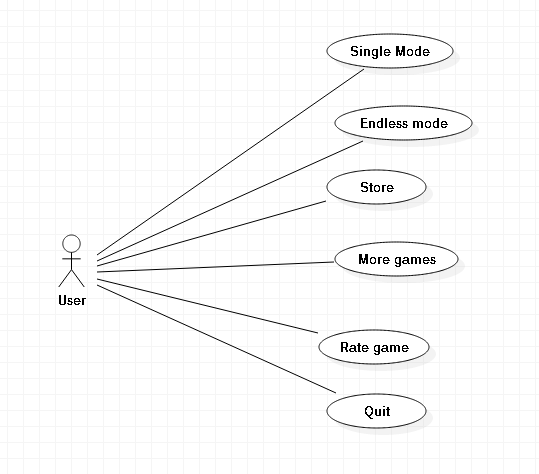
- Cung cấp các vũ khí trong game để người chơi có thể sử dụng tùy biến.

- Cho phép người chơi mua vũ khí trong game của mình thông qua việc mua đồ ở Store

- Thông báo các nhiệm vụ, hệ thống thay đổi vũ khí trong game (Súng, Dao, lựu đạn, …)

## 2.3. PHÂN TÍCH THIẾT KẾ HỆ THỐNG

### 2.3.1. Mô hình Use case



***Hình 18. Mô hình Use case***

***Bảng 1. Danh sách các ca sử dụng***

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Tên chức năng** | **Mô tả ngắn gọn chức năng** | **Chức năng** |
| F01 | Single Mode | Chế độ chơi theo màn | Cho phép người chơi chơi theo từng màn, từng map khác nhau, mở map mới thông qua kiếm sao sau mỗi màn chơi |
| F02 | Endless Mode | Chế độ chơi sinh tồn | Cho phép người chơi chơi theo chế đô sinh tồn, cố gằng sống sót sau mỗi đợt tấn công của kẻ địch |
| F03 | Store | Hệ thống bán vật phẩm vũ khí, trang bị mua thông qua tiền trong game | Cho phép người chơi mua các trang bị vũ khí mới trong cửa hàng này |
| F04 | More games | Hiển thị thêm các trò chơi khác trong store | Cho phép người chơi chọn các trò chơi khác trong store |
| F05 | Rate games | Đánh giá game | Cho phép người chơi đánh giá game |
| F06 | Quit | Thoát trò chơi | Cho phép người chơi thoát khỏi trò chơi hiện tại. |
|  |  |  |  |
| ID | Tên chức năng | Mô tả ngắn gọn chức năng | Chức năng |
| F01 | Play | Vào chọn màn chơi | Cho phép người chơi chọn thế giới để chơi và vượt qua |
| F02 | Store | Hệ thống bán vật phẩm, kĩ năng mua thông qua vàng | Cho phép người chơi mua các món đồ với cấp độ khác nhau, hệ thống kĩ năng được nâng cấp và mua trong đây. |
| F03 | More Games | Hiển thị thêm các trò chơi khác trong store | Cho phép người chơi chọn các trò chơi khác trong store |
| F04 | Quit | Thoát trò chơi | Cho phép người chơi thoát khỏi trò chơi hiện tại. |
|  |  |  |  |

2.3.2. Mô tả các Use case sử dụng

***Bảng 1. Use case Single mode***

|  |  |
| --- | --- |
| **F01 : Single Mode** | |
| **Mục đích:** | Chọn màn để chơi và vượt qua |
| **Mô tả:** | Cho phép người chơi bắt đầu chơi game .  Cho phép người chơi chơi theo màn, chơi theo lần lượt từng map, hoàn thành các nhiệm vụ đưa ra để kiếm sao, đủ số sao cần thiết có thể mở màn chơi mới, map mới |
| **Tác nhân :** | Người chơi |
| **Điều kiện trước:** | Đang ở màn hình chính của game. |
| **Luồng sự kiện chính :** | 1. Chọn chức năng Single Mode từ Menu chính  2. Thực hiện load game chơi  3. Chọn màn chơi  4. Chọn map chơi |
| **Điều kiện sau:** | Hiển thị màn hình giao diện chơi game cho người chơi |
| **Điều kiện thoát:** | -         Khi chức năng thực hiện thành công  -         Khi người dùng chọn thoát  -         Khi người dùng chọn chức năng khác |

***Bảng 2. Use case Endless mode***

|  |  |
| --- | --- |
| **F02 : Endless Mode** | |
| **Mục đích:** | Chọn màn để chơi và sống sót càng lâu càng tốt sau mỗi đợt tấn công của kẻ địch |
| **Mô tả:** | Cho phép người chơi bắt đầu chơi game .  Cho phép người chơi sinh tồn, chơi theo màn và map có ở single mode, cố gằng tiêu diệt càng nhiều kẻ địch và sống sót sau mỗi đợt tấn công của kẻ địch |
| **Tác nhân :** | Người chơi |
| **Điều kiện trước:** | Đang ở màn hình chính của game. |
| **Luồng sự kiện chính :** | 1. Chọn chức năng Endless Mode từ Menu chính  2. Thực hiện load game chơi  3. Chọn màn chơi  4. Chọn map chơi |
| **Điều kiện sau:** | Hiển thị màn hình giao diện chơi game cho người chơi |
| **Điều kiện thoát:** | -         Khi chức năng thực hiện thành công  -         Khi người dùng chọn thoát  -         Khi người dùng chọn chức năng khác |

**Bảng 3: Use case Store**

|  |  |
| --- | --- |
| **F03 : Store** | |
| **Mục đích:** | Mua trang bị vũ khí cho Player |
| **Mô tả:** | Cho phép người chơi mua các loại vũ khí khác nhau để tăng khả năng tiêu diệt kẻ địch cũng như đa dạng trong lối chơi |
| **Tác nhân :** | Người chơi |
| **Điều kiện trước:** | Đang ở màn hình chính của game. |
| **Luồng sự kiện chính :** | 1. Chọn chức năng Store từ Menu chính  2. Thực hiện load store game  3. Chọn chọn Trang bị cần mua |
| **Điều kiện sau:** | Hiển thị màn hình giao diện Trang bị của Store |
| **Điều kiện thoát:** | -         Khi chức năng thực hiện thành công  -         Khi người dùng chọn thoát  -         Khi người dùng chọn chức năng khác |

**Bảng 4: Use case More Games**

|  |  |
| --- | --- |
| **F04 : More Games** | |
| **Mục đích:** | Chơi thêm các game khác |
| **Mô tả:** | Cho phép người chơi, chơi thêm các game khác trong store. |
| **Tác nhân :** | Người chơi |
| **Điều kiện trước:** | Đang ở màn hình chính của game. |
| **Luồng sự kiện chính :** | 1. Chọn chức năng More Games  2. Chọn game khác |
| **Điều kiện sau:** | Hiển thị màn hình của Google Play |
| **Điều kiện thoát:** | -         Khi chức năng thực hiện thành công  -         Khi người dùng chọn thoát  -         Khi người dùng chọn chức năng khác |

**Bảng 5: Use case Rate Games**

|  |  |
| --- | --- |
| **F05 : Rate Games** | |
| **Mục đích:** | Đánh giá game |
| **Mô tả:** | Cho phép người chơi đánh giá |
| **Tác nhân :** | Người chơi |
| **Điều kiện trước:** | Đang ở màn hình chính của game. |
| **Luồng sự kiện chính :** | 1. Chọn chức năng More Games  2. Chọn game khác |
| **Điều kiện sau:** | Hiển thị màn hình của Google Play |
| **Điều kiện thoát:** | -         Khi chức năng thực hiện thành công  -         Khi người dùng chọn thoát  -         Khi người dùng chọn chức năng khác |

**Bảng 6: Use case Quit**

|  |  |
| --- | --- |
| **F06 : Quit** | |
| **Mục đích:** | Chơi thêm các game khác |
| **Mô tả:** | Cho phép người chơi tắt ứng dụng game |
| **Tác nhân :** | Người chơi |
| **Điều kiện trước:** | Đang ở màn hình chính của game. |
| **Luồng sự kiện chính :** | 1. Chọn chức năng Quit |
| **Điều kiện sau:** | Hiển thị màn hình của Google Play |
| **Điều kiện thoát:** | -         Khi chức năng thực hiện thành công |

### 

### 2.3.3. Biểu đồ lớp

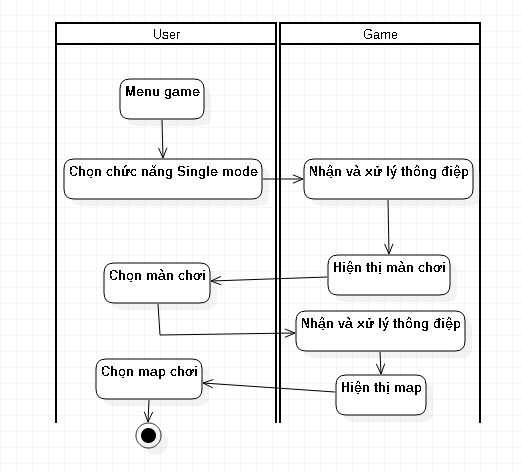


***Hình 19. Biểu đồ lớp Enemy***

### 

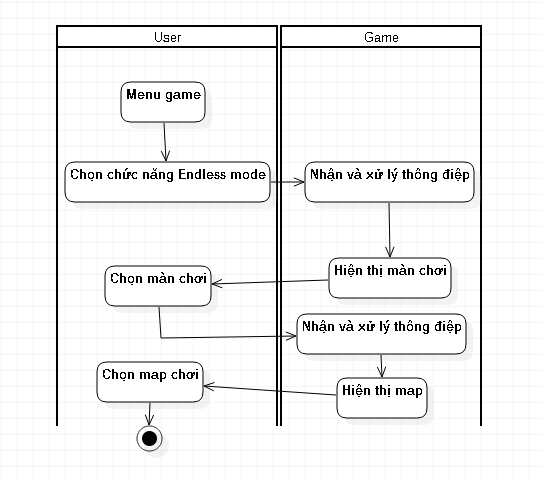
### 2.3.4. Biểu hoạt động

*2.3.4.1. Single mode*



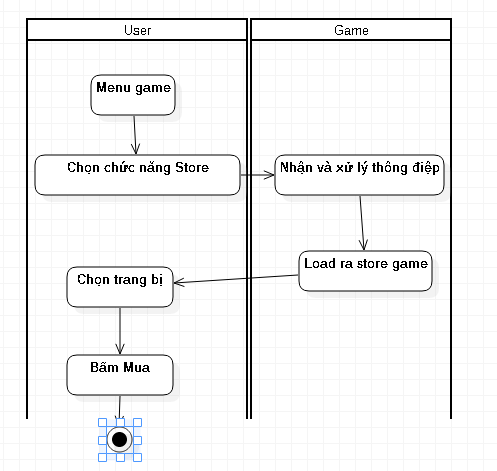
***Hình 20. Biểu đồ hoạt động Single mode***

*2.3.4.2. Endless mode*



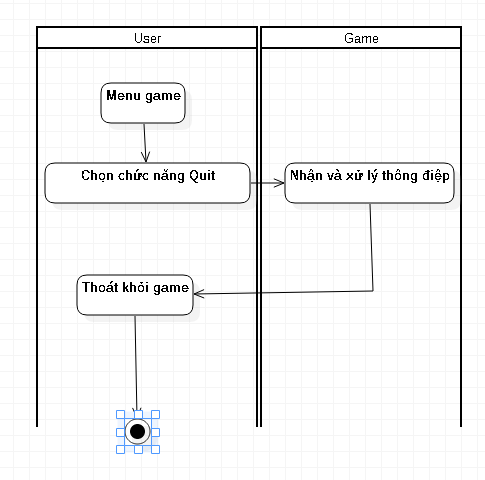
***Hình 21. Biểu đồ hoạt động Endless mode***

*3.3.4.3. Store*



***Hình 22. Biểu đồ hoạt động Store***

*3.3.4.4. Quit*



***Hình 23. Biểu đồ hoạt động Quit***

# Chương 3

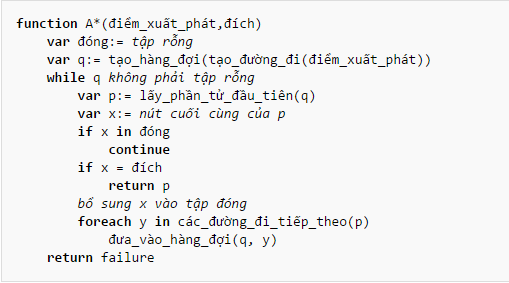
# TRIỂN KHAI VÀ GIẢI PHÁP KHI XÂY DỰNG GAME

## 3.1. XÂY DỰNG CÁC ĐỐI TƯỢNG CHÍNH TRONG GAME

### 3.1.1. Enemys

Các 3D game engine đều có hỗ trợ xử lý vấn đề tìm đường cho AI  (trí  tuệ nhân  tạo),  thuật  toán  thông  dụng  nhất  được  sử dụng  là  A\*.

Thuật toán này tìm một đường đi từ một [nút](https://vi.wikipedia.org/w/index.php?title=N%C3%BAt_(l%C3%BD_thuy%E1%BA%BFt_%C4%91%E1%BB%93_th%E1%BB%8B)&action=edit&redlink=1) khởi đầu tới một nút đích cho trước (hoặc tới một nút thỏa mãn một điều kiện đích), sử dụng một "đánh giá heuristic" để xếp loại từng nút theo ước lượng về tuyến đường tốt nhất đi qua nút đó.



Liên hệ từ thuật toán A\* cơ bản , các Waypoint đóng vai trò tương tự như các node, sẽ chứa thông tin về PreviousNode và NextNode, trong không gian 2D các waypoint đơn thuần chỉ là GameObject cơ bản chứ script.

Áp dụng giải thuật này vào cho các Enemy. Khi thực hiện đánh , Enemys sẽ xuất hiện từ vị trí biên của map và di chuyển. Lúc này, Enemys sẽ tự tìm tới Player bằng đường đi ngắn nhất và tấn công.

Liên hệ từ thuật toán A\* cơ bản, các Waypoint đóng vai trò tương tự như các node, sẽ chứa thông tin về PreviousNode và NextNode, trong không gian 2D các waypoint đơn thuần chỉ là GameObject cơ bản chứ script.

### 3.1.2. Player

Vị trí xuất hiện, di chuyển và sử dụng kĩ năng bằng joystick, và hệ thống đổi vũ khí, sử dụng dao, lựu đạn để tiêu diệt kẻ địch

### 3.1.3. Store

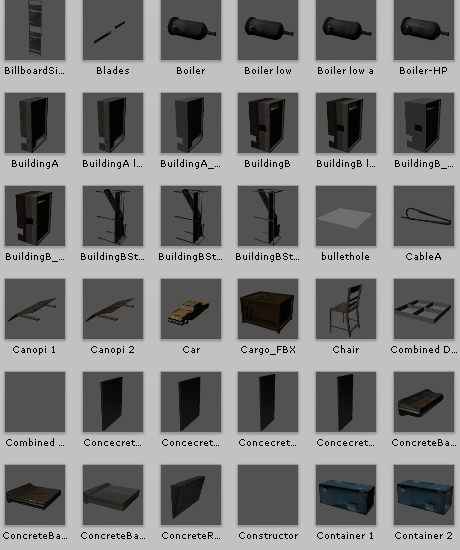
Hệ thống bán trang bị mua thông qua tiền kiếm được mỗi khi tiêu diệt kẻ địch. Các trang bị bị đa dạng gồm 12 loại súng khác nhau, lựu đạn và mua đạn riêng cho mỗi loại súng.

## 3.2. TẠO MAP

Các scene là nơi chứa các đối tượng object trong game của bạn. Chúng có thể được sử dụng để tạo một menu chính, các màn chơi riêng lẻ, và bất kỳ cái gì tương tự. Hãy hình dung là mỗi scene là một màn chơi riêng lẻ. Trong mỗi scene, bạn sẽ tạo một môi trường, đặt các chướng ngại vật, và chúng được trang trí, đây là cong việc thiết kế và xây dựng game một cách thực tế.

Khi bạn tạo một project mới trong Unity, cửa sổ màn chơi scene sẽ hiển thị một màn chơi scene được tạo mới. Đây là một file scene mới tạo và chưa được lưu. Scene sẽ không có gì cả ngoại trừ các đối tượng mặt định - hoặc là camera trực giao hoặc là camera phối cảnh - và một nguồn sáng, điều này phụ thuộc vào việc bạn chọn khởi đầu project là 2D hay 3D.

Sau khi tạo được scene , ta tạo một object rỗng để chứa các đối tượng địa hình, và sắp đặt chúng để tạo nên khung cảnh của màn chơi. Các đối tượng object này có thể tạo bởi một phần mềm đồ họa 3D hoặc download nguồn có sẵn và miễn phí trên mạng



***Hình 24: Hình ảnh các đối tượng địa hình***

Sau khi đặt các đối tượng vật thể và những vật thể rỗng point patrol đại diện cho các mốc điểm đi tuần của enemys:



***Hình 25: Hình ảnh một scene đã xây dựng hoàn thành***

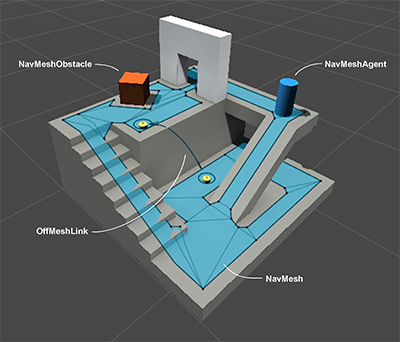
## 3.3. TRÍ TUỆ NHÂN TẠO CHO ENEMY

### 3.3.1. Bài toán tìm đường cho Enemy

Trong Unity đã cung cấp Navigation xử lý điều này.

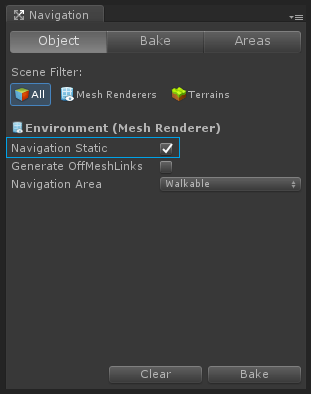
Tổng quan về Navigation với các khái niệm:

* NavMesh: Cấu trúc data cho việc tìm đường đi cho đối tượng.
* NavMesh Agent: Component giúp bạn tạo di chuyển cho đối tượng kết hợp với NavMesh.
* Off-Mesh Link: Component cho phép di chuyển tắt qua nhau mà không theo quy luật Navigation bình thường là phải qua từng điểm một.
* NavMesh Obstacle: Component mô tả đối tượng nào bị tránh khi trên đường di chuyển.



***Hình 26: Các thành phần của Navigation***

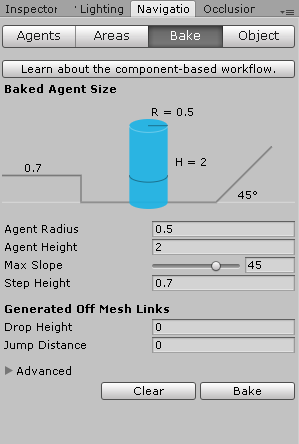
Sau khi tạo xong map, mở cửa sổ Navigation từ Windows menu. Chọn All và tích Navigation Static nghĩa là các đối tượng này không di chuyển trong suốt run time.



***Hình 27: Hình ảnh của Navigation***

Rồi chuyển tab Bake, như hình các layer màu xanh đó biểu thị cho những vị trí được đi qua.

* Mình có thể thay đổi vị trí đi qua bằng cách tham đổi các biến số trong Bake tab.
* Giờ mình đã tạo được map với các NavMesh cơ bản rồi. Tiếp đến, chọn đối tượng di chuyển để thêm NavMesh Agent component và mình có thể tùy chỉnh các thông số ở đây cho thích hợp.



***Hình 28: Hình ảnh Bake tùy chỉnh các thông số***

* Bước cuối cùng chỉ là tạo cho đối tượng để di chuyển nữa là xong. Tạo script EnemyAI.cs và add vào đối tượng với code:

Khai báo : UnityEngine.AI.NavMeshAgent nav;

void Start() // Tao moi enemy

{

audi = GetComponent<AudioSource>();

nav = GetComponent<UnityEngine.AI.NavMeshAgent>();

anim = GetComponent<Animator>();

eneHealth = GetComponent<EnemyHealth>();

playerMask = LayerMask.GetMask("Player");

player = GameObject.FindGameObjectWithTag("Point Player").transform;

heroHealth = player.GetComponentInParent<HeroHealth>();

pointTarget.position = pointPatrol[indexPointCome].transform.position;

pointTarget.GetComponent<PatrolController>().id = idEnemy;

}

Và có hàmnav.SetDestination(pointTarget.position); cho phép đối tượng luôn di chuyển đến các điểm mình đặt tùy theo các hành động của enemy và tự động tránh các vật cản trên đường.

### 3.3.2. Các hành động của enemy

Enemy đã được chỉ định đi qua các điểm đi tuần quanh khung cảnh màn chơi, với nhiều enemy tuy rằng vẫn điểm đi tuần là như nhau nhưng thứ tự đi tuần của mỗi enemy lại được tạo random.

Khi tới mỗi điểm đi tuần thì enemy sẽ gọi hành động nhìn xung quanh. Khi nhìn xung quanh nếu ko thấy enemy thì sẽ gọi đi tuần đến điểm tiếp theo. Nếu thấy player thì enemy sẽ chạy tới điểm mà vừa thấy player. Nếu thấy player mà đủ tầm bắn thì enemy sẽ bắn.

Khi bị player bắn enemy sẽ nhìn về hướng bị bắn và chạy tới điểm thấy player. Trong khi bắn nhau với enemy luôn chạy hàm CheckLook để kiểm tra xem Player có trong khoảng nhìn thấy và tầm bắn của Enemy không để thực hiện các hành động tiếp theo.

Khi enemy bị player bắn chết, đợi sau 4 giây rồi thực hiện ẩn enemy.

Code xử lý với enemy:

|  |
| --- |
| using UnityEngine;  using System.Collections;  public class EnemyAI : MonoBehaviour  {  public enum EnemyState  {  Move\_patrol  Look\_around  Have\_seen\_player  Player\_fire  Going\_to\_point\_player,  Came\_point\_player  Fighting\_player  Die  }  [SerializeField]  GameObject pointFire, holeFire, lightGun;  [SerializeField]  Transform spine, rotationPoint;  [SerializeField]  float range;  public int fieldOfViewEnemy, turnSpeed, enemyDam;  public float cooldownFire, coolDownShot, hitchFire;  Ray shootRay;  RaycastHit shootHit;  int playerMask;  float[] Speed = { 0, 1.5f, 3f };  AudioSource audi;  EnemyHealth eneHealth;  Transform player;  UnityEngine.AI.NavMeshAgent nav;  bool isLookAround;  Animator anim;  HeroHealth heroHealth;  public int indexPointCome, direc; //chi so cua diem dang tuan, huong tuan theo mang  public GameObject[] pointPatrol;  public int idEnemy;  public Transform pointTarget  public EnemyState stateEnemy  void OnEnable()  {  stateEnemy = EnemyState.Move\_patrol;  direc = Random.Range(-1, 1);  isLookAround = false;  }  void Start()  {  audi = GetComponent<AudioSource>();  nav = GetComponent<UnityEngine.AI.NavMeshAgent>();  anim = GetComponent<Animator>();  eneHealth = GetComponent<EnemyHealth>();  playerMask = LayerMask.GetMask("Player");  player = GameObject.FindGameObjectWithTag("Point Player").transform;  heroHealth = player.GetComponentInParent<HeroHealth>();  pointTarget.position = pointPatrol[indexPointCome].transform.position;  pointTarget.GetComponent<PatrolController>().id = idEnemy;  }  void OnAnimatorIK(int layerIndex)  {  if (stateEnemy == EnemyState.Have\_seen\_player)  {  anim.SetBoneLocalRotation(HumanBodyBones.Spine, rotationPoint.localRotation);  }  }  void Update()  {  rotationPoint.localRotation = Quaternion.Euler(0, -90,  -90 + Mathf.Sign(player.position.y - rotationPoint.position.y) \* Vector3.Angle(player.position - rotationPoint.position, transform.forward));  switch (stateEnemy)  {  case EnemyState.Move\_patrol:  anim.SetBool("Move\_patrol", true);  nav.speed = Speed[1];  nav.SetDestination(pointTarget.position);  CheckLook();  break;  case EnemyState.Look\_around:  if (!isLookAround)  {  anim.SetBool("Move\_patrol", false);  anim.SetBool("Look\_around", true);  StartCoroutine(LookAround());  }  break;  case EnemyState.Came\_point\_player:  if (!isLookAround)  {  anim.SetBool("Going\_to\_point\_player", false);  anim.SetBool("Came\_point\_player", true);  pointTarget.position = pointPatrol[indexPointCome].transform.position;  StartCoroutine(LookAround());  }  break;  case EnemyState.Going\_to\_point\_player:  if (nav.enabled)  {  nav.speed = Speed[2];  nav.SetDestination(pointTarget.position);  CheckLook();  }  break;  case EnemyState.Player\_fire:  TakeDame();  break;  case EnemyState.Die:  nav.speed = Speed[0];  nav.enabled = false;  anim.SetBool("Have\_seen\_player", false);  anim.SetBool("Look\_around", false);  anim.SetBool("Move\_patrol", false);  anim.SetBool("Player\_fire", false);  anim.SetBool("Going\_to\_point\_player", false);  anim.SetBool("Came\_point\_player", false);  anim.SetBool("Die", true);  audi.Stop();  lightGun.SetActive(false);  StartCoroutine(Die());  break;  case EnemyState.Have\_seen\_player:  transform.LookAt(new Vector3(player.position.x, transform.position.y, player.position.z));  break;  }  }  IEnumerator Die()  {  yield return new WaitForSeconds(4);  //Debug.Log("Da vao");  transform.parent.gameObject.SetActive(false);  StopAllCoroutines();  }  public void TakeDame()  {  //state = EnemyState.Player\_fire;  anim.SetBool("Look\_around", false);  anim.SetBool("Move\_patrol", false);  anim.SetBool("Player\_fire", true);  if (!isLookAround)  {  transform.LookAt(new Vector3(player.position.x, transform.position.y, player.position.z));  StartCoroutine(LookAround());  }  }  IEnumerator LookAround()  {  isLookAround = true;  //cooldownLook = 0;  nav.speed = Speed[0];  Vector3 direcLast = transform.forward;  transform.Rotate(Vector3.up \* 10);  while (isLookAround && stateEnemy != EnemyState.Have\_seen\_player)  {  transform.Rotate(Vector3.up \* Time.deltaTime \* turnSpeed);  if (Vector3.Angle(direcLast, transform.forward) <= 5)  {  isLookAround = false;  }  CheckLook();  yield return new WaitForSeconds(Time.deltaTime);  }  if (stateEnemy == EnemyState.Look\_around || stateEnemy == EnemyState.Player\_fire)  {  anim.SetBool("Look\_around", false);  anim.SetBool("Player\_fire", false);  stateEnemy = EnemyState.Move\_patrol;  }  else if (stateEnemy == EnemyState.Came\_point\_player)  {  anim.SetBool("Came\_point\_player", false);  anim.SetBool("Move\_patrol", true);  stateEnemy = EnemyState.Move\_patrol;  }  isLookAround = false;  }  void CheckLook()  {  if (Vector3.Distance(player.position, pointFire.transform.position) < range)  {  shootRay.origin = pointFire.transform.position;  shootRay.direction = player.position - pointFire.transform.position  if (Vector3.Angle(shootRay.direction, transform.forward) <= fieldOfViewEnemy)  {  if (Physics.Raycast(shootRay, out shootHit, 500) && shootHit.transform.tag.Contains("Player"))  {  isLookAround = false;  stateEnemy = EnemyState.Have\_seen\_player;  pointTarget.position = player.transform.position;  StartCoroutine(Fire());  return;  }  else if (stateEnemy == EnemyState.Fighting\_player)  {  anim.SetBool("Have\_seen\_player", false);  anim.SetBool("Going\_to\_point\_player", true);  stateEnemy = EnemyState.Going\_to\_point\_player;  }  }  }  }  IEnumerator Fire()  {  anim.SetBool("Have\_seen\_player", true);  anim.SetBool("Look\_around", false);  anim.SetBool("Move\_patrol", false);  anim.SetBool("Player\_fire", false);  anim.SetBool("Going\_to\_point\_player", false);  anim.SetBool("Came\_point\_player", false);  nav.speed = Speed[0];  if (!heroHealth.isDie && !GameManager.Instance.isGameEnd)  {  for (int i = 0; i < 5; i++)  {  if (eneHealth.isDie)  {  break;  }  pointFire.transform.LookAt(player);  audi.Stop();  audi.Play();  lightGun.SetActive(true);  pointFire.transform.Rotate(-Random.Range(-hitchFire, hitchFire), Random.Range(-hitchFire, hitchFire), 0);  shootRay.origin = pointFire.transform.position;  shootRay.direction = pointFire.transform.forward;  if (Physics.Raycast(shootRay, out shootHit, 500))  {    if (shootHit.collider.tag.Contains("Player"))  {  HeroHealth playerHealth = shootHit.transform.GetComponent<HeroHealth>();  playerHealth.Takedame(enemyDam);  }  else  {  GameObject obj = (GameObject)  Instantiate(holeFire, shootHit.point, Quaternion.FromToRotation(Vector3.up, shootHit.normal));  Destroy(obj, 3);  }  }  yield return new WaitForSeconds(0.1f);  lightGun.SetActive(false);  yield return new WaitForSeconds(coolDownShot);  }  pointFire.transform.localRotation = Quaternion.Euler(Vector3.zero);  audi.Stop();  yield return new WaitForSeconds(cooldownFire);  transform.LookAt(new Vector3(player.position.x, transform.position.y, player.position.z));  stateEnemy = EnemyState.Fighting\_player;  CheckLook();  }  }  } |

## 3.4. CÀI ĐẶT, THIẾT LẬP MÔI TRƯỜNG VÀ BUILD PROJECT

### 3.4.1. Môi trường phát triển ứng dụng và các công cụ

- Môi trường phát triển: Mono-devolop, Engine Unity 5.6.0 ,

- Môi trường triển khai: Mobile/PC/MAC.

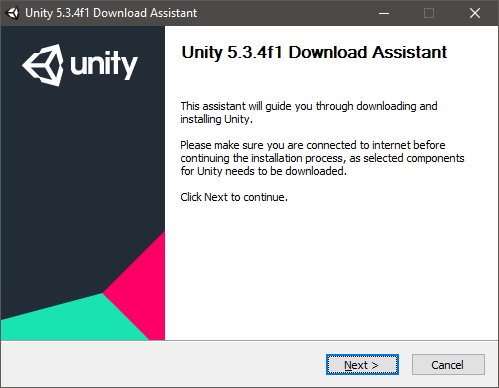
- Các thư viện: Các thư viện debug và quản lý lỗi debugger, các thư viện đồ họa 3D của Engine Unity.

### 3.4.2. Cài đặt, thiết lập môi trường

***Bước 1:*** Tải và cài đặt Engine Unity tại:

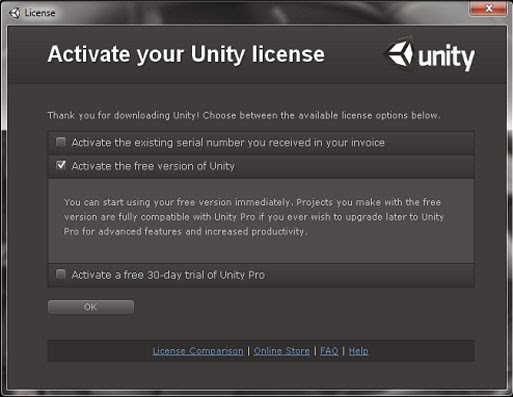
<https://unity3d.com/get-unity/download?ref=personal>  (nên chọn bản mới nhất)

- Sau khi tải xuống hoàn tất, double click vào icon của Unity để tiến hành cài đặt. Việc cài đặt Unity diễn ra bình thường và tương tự như cài đặt các chương trình khác.



***Hình 29. Giao diện cài đặt Unity***

- Khởi động Uniy, hộp thoại kích hoạt sẽ xuất hiện, có thể chọn việc sử dụng bản miễn phí, trả phí (Unity Pro) hay dùng thư bản trả phí Unity Pro trong 30 ngày.



***Hình 30. Hộp thoại Unity***

- Chọn ra sự lựa chọn và bấm OK. Nên đăng nhập hoặc tạo tài khoản.

- Đăng kí và đăng nhập để kích hoạt phiên bản Unity

***Bước 2:***Tải và cài đặt Java JDK tại:

<http://java.sun.com/javase/downloads/index.jsp> (nên chọn bản mới nhất)

***Bước 3:*** Tải và cài đặt Android Studio gói này sẽ chứa các thành phần: ADT plugin, Android SDK Tools, Android Platform tools, Android API mới nhất, Android system image mới nhất cho Emulator tại :<http://developer.android.com/sdk/index.html>.

***Bước 4:*** Tải và cài đặt Visual Studio:

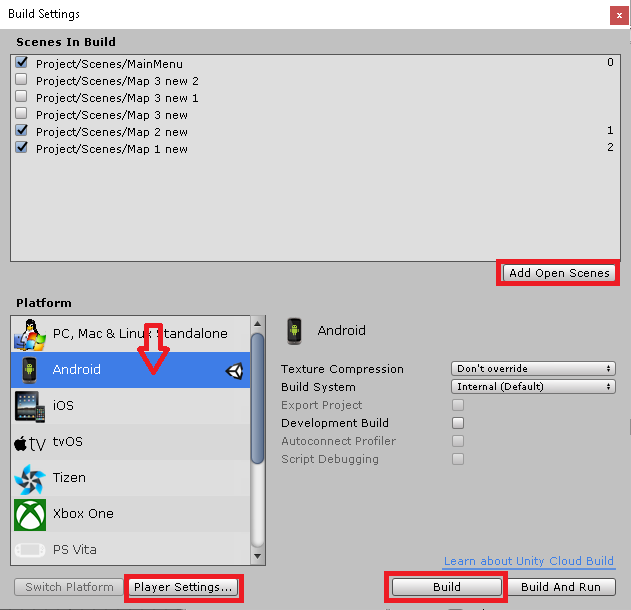
https://www.visualstudio.com/en-us/downloads/download-visual-studio-vs.aspx

Chú ý : Bước 3,4,5 là thiết lập môi trường để build cho HĐH Android và Windows Store/ Windows Phone.

### 3.4.3. Hướng dẫn build cho các hệ điều hành

*3.4.3.1. Hệ điều hành Android*

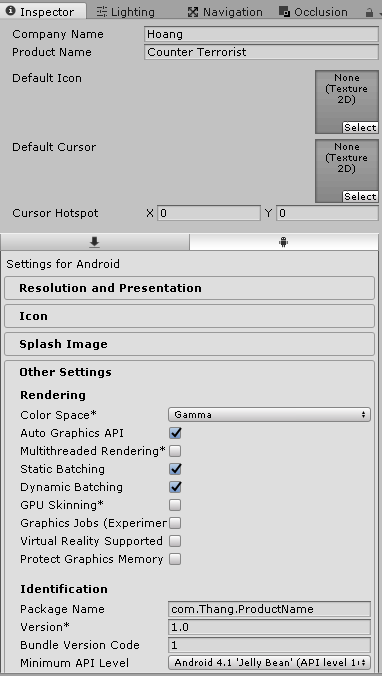
***Bước 1:*** Vào Tab “File” chọn mục “Build Setting” hoặc “Ctrl + Shift B”**.** Một giao diện như hình bên dưới hiện ra.

****

***Hình 31. Giao diện build***

***Bước 2:***  Add current để chọn các scene định build, thứ tự bắt đầu từ 0. 0 là màn hình chạy đầu tiền, tương tự cho các màn hình tiếp theo

***Bước 3:*** Tiếp theo chọn Player settings, thay đổi các thuộc tính tùy theo game của mình.



***Hình 32. Giao diện PlayerSetting***

Phần 1: PlayerSettings:

- Company Name: tên công ty/ Studio

- Product Name: điền tên sản phẩm

- Default Icon: hình đại diện cho game

- Default Cursor: hình con trỏ chuột (chỉ dùng cho game destop).

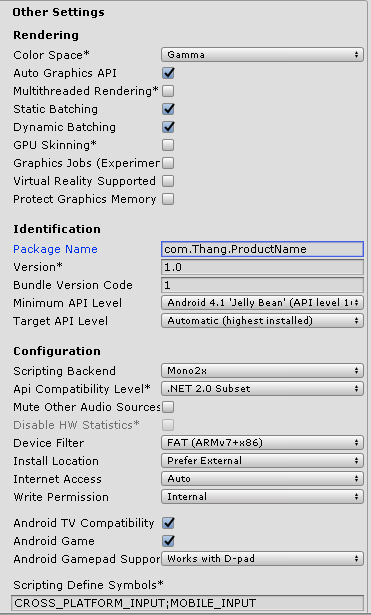
Phần 2: Setting cho Android

- Resolution and Presentation

- Icon: phần này hiện icon

- Splash Image: chọn hình lúc mới load game vào.

- Other Settings :



***Hình 33. Giao diện Other Settings của Android***

Bulder Identifier :  Tên gói  game, định danh trên GooglePlay.

***Bước 4:*** Sau khi thiết lập các tùy chỉnh chọn Build để kết thúc quá trình build file Apk.

## 3.5. HƯỚNG DẪN SỬ DỤNG

**Người chơi sẽ phải làm gì ?**

Nhập vai vào Player của mình  để đi tiêu diệt hết khủng bố tại căn cứ của chúng cũng như những nơi chúng chiếm đóng

Có 3 map chơi đại diện cho những khung cảnh khác nhau, mỗi map gồm nhiều màn với nhiệm vụ khó tăng dần để kiếm sao

Mỗi màn chơi sẽ khi tiêu diệt kẻ địch sẽ giúp Player kiếm được tiền dùng để mua các trang bị cũng như nạp đạn ở trong cửa hàng Store

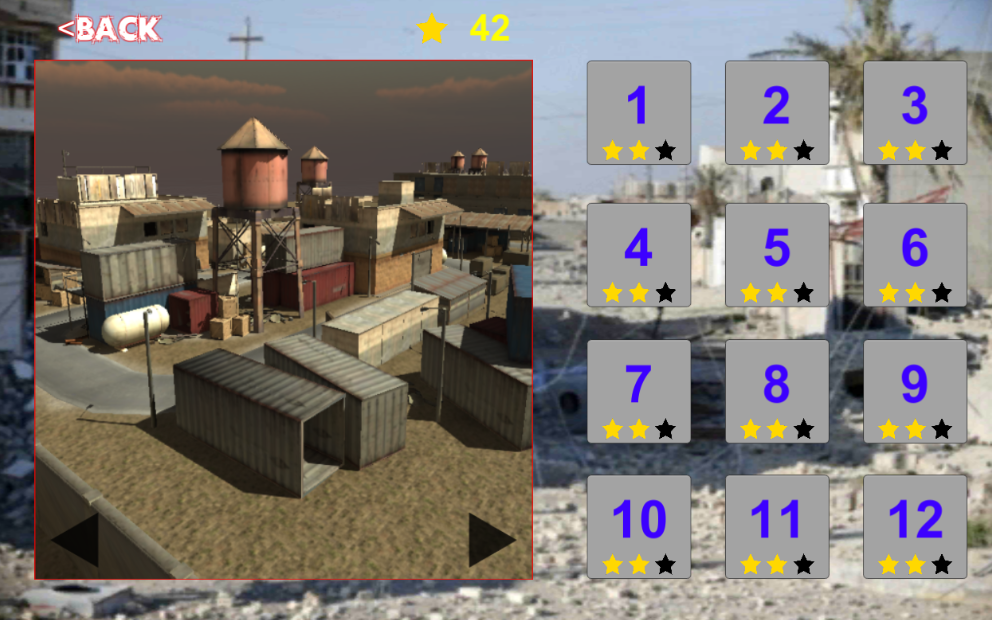
Ngoài ra còn có chế độ chơi sinh tồn Endless mode thử thách hơn …

Bắt đầu vào game…



***Hình 34: Menu chính***

Chọn Single mode…



***Hình 35: Giao diện chọn màn chơi***

Chọn một màn chơi bất kỳ…



***Hình 36: Giao diện hiển thị nhiệm vụ của màn chơi***

Ấn Play để trải nghiệm…

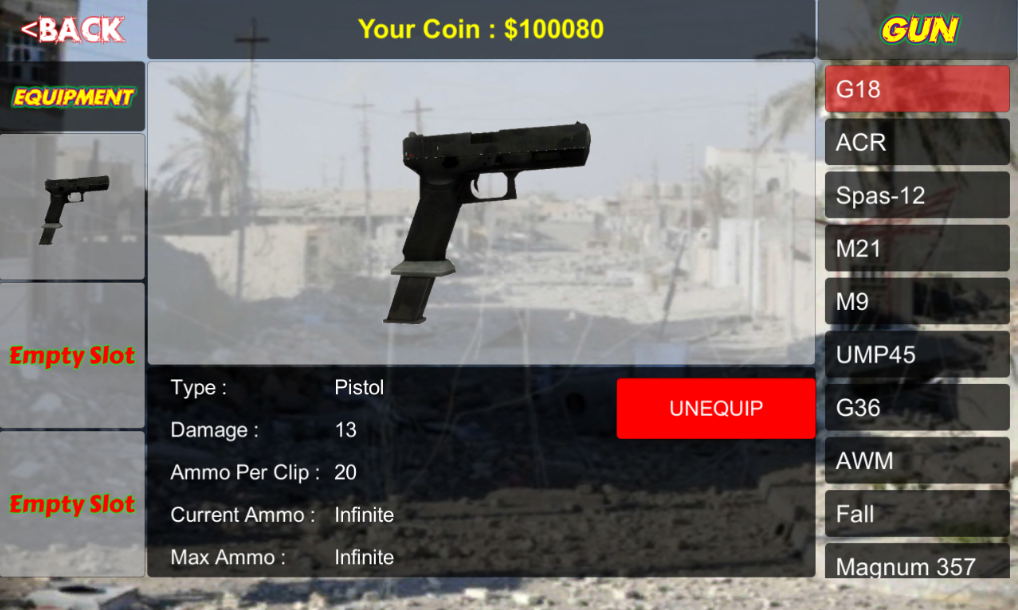


**Hình 37: Giao diện khi trải nghiệm game**

Phía bên trái là nút để điều khiển Player ,nút bên phải là nút để điều kiển thay đổi vũ khí, nạp đạn, ngắm, ném lựu và nhảy. Phía trên góc phải là hiển thị điểm sinh lực, số đạn và số enemy cần tiêu diệt. Cạnh đó có các nút hỗ trờ và dừng trò chơi.



***Hình 38: Giao diện kết thúc màn chơi***



***Hình 39: Giao diện Cửa hàng Store***

# KẾT LUẬN

**1. Kết quả đạt được**

v *Về mặt lập trình*

- Áp dụng được hầu hết các kỹ thuật thường được sử dụng trong game.

- Hiểu và áp dụng Navigation vào hệ thống tìm đường.

- Biết cách xây dựng các hệ thống giao tiếp với Designer.

- Nắm được quy trình xây dựng một trí thông minh nhân tạo qua đặc tả.

- Các phương pháp Debug.

- Hiểu và xử lí được một số lỗi thường gặp trong lập trình game.

v *Về mặt thiết kế*

- Nắm được các kỹthuật xây dựng môi trường, nhân vật.

- Biết cách sử dụng các phần mềm tương  ứng  để thiết kế mô hình, texture, vật liệu cho các đối tượng đem vào triển khai thực tế trong Unity

- Áp dụng  được kỹ thuật tối  ưu qua công cụ của Unity cho phép trò chơi có mức khung hình trên giây ổn định.

v *Kết quả chung của đề tài*

- Xây dựng thành công các chức năng đã được miêu tả của ứng dụng.

- Ứng dụng đáp ứng được các yêu cầu phi chức năng, tương thích nhiều thiết bị khác nhau.

- Ứng dụng đã triển khai được với nhiều thiết bị thật trên thực tế.

- Đề tài đã giúp sinh viên hiểu thêm về cách làm game tương tác giữa những người chơi khác nhau.

**2. Hạn chế còn tồn tại**

- Chưa khai thác được hết các khả năng về thiết kế của Unity cũng như các phần mềm hỗ trợ.

- Các kỹthuật dựng hình tân tiến và chuyên nghiệp chưa được tận dụng tối đa.

- Khả năng tối ưa hóa các tài nguyên đồ họa (mô hình, vật liệu, …) chưa tốt.

- Hiệu năng của game chưa được tối ưu hóa đến mức tối đa

**3. Hướng phát triển**

Counter Terrorist 3D hiện tại mới chỉ đáp ứng được mục tiêu ban đầu là tìm hiểu công nghệ mới Engine Unity và tìm hiểu trí tuệ nhân tạo trong phạm vi dừng lại ở mức áp dụng trực tiếp vào việc xây dựng Counter Terrorist 3D. Vì chỉ mang tính chất demo nên Game còn thiếu nhiều chi tiết của một ứng dụng hoàn chỉnh (giới hạn thử thách, số lượng trợ giúp, cấp độ, hành động, nhiệm vụ,…v…v). Hướng mở rộng của Game trong tương lai là phát triển Game thành một ứng dụng hoàn chỉnh ,thêm các đối tượng khác nhau vào trò chơi,  đồng thời cải thiện và tối ưu các thuật toán hơn nữa để Game có thể chạy trơn tru hơn, mượt hơn và chạy được trên nhiều cấu hình máy tính khác nhau. Ngoài ra game sẽ Kết nối mạng xã hội (facebook, G+..) Đưa lên các store và gắn quảng cáo để thu lợi nhuận.

**TÀI LIỆU THAM KHẢO**

*Tiếng Việt:*

[1]. [Đinh Nguyễn Anh Dũng](http://nxbdhqghcm.edu.vn/tac-gia/1205/dinh-nguyen-anh-dung.htm) **-**[Vũ Thanh Nguyên](http://nxbdhqghcm.edu.vn/tac-gia/1250/vu-thanh-nguyen.htm)**.** *Nhập môn phát triển game* .Nhà xuất bản ĐHQG-HCM,2015.

[2]. Phạm Thọ Hoàn, Phạm Thị Anh Lê. *Trí tuệ nhân tạo*. Nhà xuất bản Đại học sư phạm Hà Nội, 2011.

[3]. TS. Nguyễn Thanh Bình. *Phân tích và thiết kế hệ thống hướng đối tượng*. Nhà xuất bản Đại học Đà Nẵng, 2009.

*Tiếng Anh:*

[4]. Alan R.Stagner .*Unity Multiplayer Games.* by Birmingham - Munbai, 2013

[5]. Alan Thorn . *pro Unity Game Development with C#.* by Technology In Action*,* 2014

[6]. Dave Mark. *Behavioral Mathematics for  Game AI* . Charles River Media (Applied Mathematics),2009

*Nguồn tham khảo từ Internet:*(nguồn từ trang chủ của UNITY)

[7]. Website: https://unity3d.com/learn

[8]. Website:<http://www.unity3dstudent.com/>

[9]. Website:<http://www.gamedev.net/>

[10]. Website:<http://forum.unity3d.com/threads/26785-Unity-Jump-Start-Video-Tutorials>