BOR3**BỘ CÔNG THƯƠNG**

**TRƯỜNG CAO ĐẲNG KỸ THUẬT CAO THẮNG**

**KHOA ĐIỆN TỬ - TIN HỌC**

…..  …..



**BÁO CÁO ĐỒ ÁN TỐT NGHIỆP**

**­**

**Đề Tài: Phát Triển Game 2D trên Unity**

**GVHD**: **Tôn Long Phước**

**SVTH: Vũ Đức Duy**

**LỚP**: **CĐTH12C**

TP.HỒ CHÍ MINH, tháng 07 năm 2016

LỜI MỞ ĐẦU

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 những Game hay và hấp dẫn đã 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 Middleware 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 do có thể phát triển nhiều Game từ một Game Engine.

Từ xu hướng phát triển và những bất cập trên, 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) bắn máy bay 2D. Chuẩn bị kiến thức và kỹ năng cho định hướng nghề nghiệp (phát triển Game) sau này của chúng em, góp phần vào sự phát triển của ngành công nghiệp Game nước nhà. Các chương đầu trong tài liệu sẽ trình bày lần lượt các khái niệm chung về Game Engine và thế giới 2D.

Các chương tiếp theo sẽ giới thiệu về Engine Unity bao gồm các đặc điểm, tính năng, công cụ và thành phần trong Engine này. Sau những nội dung về thiết kế, tài liệu sẽ trình bày về việc lập trình trên Unity khi giới thiệu về ngôn ngữ lập trình cùng các lớp, hàm trong thư viện dựng sẵn của Unity thông qua các ví dụ thực tế khi phát triển một game bắn máy bay 2D. Cuối cùng là giới thiệu về bối cảnh, cốt truyện, tài liệu hướng dẫn sử dụng của trò chơi Demo cùng chương tổng kết về các khó khăn và những thành quả trong suốt quá trình phát triển trò chơi Demo trên Unity.

**LỜI CÁM ƠN**

Chúng em xin chân thành cảm ơn Khoa Công Nghệ Thông Tin, Trường Cao Đẳng Kỹ Thuật Cao Thắng đã tạo điều kiện tốt cho chúng em thực hiện đề tài này.

Em xin chân thành cảm ơn Thầy Tôn Long Phước, là người đã tận tình hướng dẫn em, chỉ bảo em trong suốt thời gian thực hiện đề tài. Em cũng xin cảm ơn Thầy Lê Quang Song đã có những trao đổi, những chỉ dẫn giúp chúng em giải quyết các vấn đề và hoàn thiện đề tài.

Em cũng xin gửi lời cảm ơn sâu sắc đến quý Thầy Cô trong Khoa đã tận tình giảng dạy, trang bị cho chúng em những kiến thức quí báu trong những năm học vừa qua.

Chúng em xin gửi lòng biết ơn sâu sắc đến Ba, Mẹ, các anh chị và bạn bè đã ủng hộ, giúp đỡ và động viên chúng em trong những lúc khó khăn cũng như trong suốt thời gian học tập và nghiên cứu.

Mặc dù chúng em đã cố gắng hoàn thành luận văn trong phạm vi và khả năng cho phép, nhưng chắc chắn sẽ không tránh khỏi những thiếu sót, kính mong sự cảm thông và tận tình chỉ bảo của quý Thầy Cô và các bạn.

**SV Thực Hiện :** **Vũ Đức Duy**

**NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN**

1. **Mục đích và nội dung của đồ án:**

………………………………………………………………………………

………………………………………………………………………………

………………………………………………………………………………

1. **Kết quả đạt được:**

………………………………………………………………………………

………………………………………………………………………………

………………………………………………………………………………

1. **Ý thức làm việc của sinh viên:**

………………………………………………………………………………

………………………………………………………………………………

………………………………………………………………………………

Tp. Hồ Chí Minh, ngày**…….**tháng**…….**năm**…….**

**Giáo viên hướng dẫn**

(Ký và ghi rõ họ tên)

**NHẬN XÉT CỦA GIÁO VIÊN PHẢN BIỆN**

1. **Mục đích và nội dung của đồ án:**

………………………………………………………………………………

………………………………………………………………………………

………………………………………………………………………………

1. **Kết quả đạt được:**

………………………………………………………………………………

………………………………………………………………………………

………………………………………………………………………………

1. **Ý thức làm việc của sinh viên:**

………………………………………………………………………………

………………………………………………………………………………

………………………………………………………………………………

Tp. Hồ Chí Minh, ngày**…….**tháng**…….**năm**…….**

**Giáo viên phản biện**

(Ký và ghi rõ họ tên)

Mục Lục

[**CHƯƠNG 1:** **TỔNG QUAN VỀ ENGINE UNITY VÀ KHÔNG GIAN 2D, 3D** 1](#_Toc455278726)

[I. TỔNG QUAN VỀ ENGINE UNITY 1](#_Toc455278727)

[1.1.1. Unity Là Gì ? 1](#_Toc455278728)

[1.1.2. Sơ Lược Lịch Sử Hình Thành Và Phát Triển Của Unity 3](#_Toc455278729)

[a. Hình Thành 3](#_Toc455278730)

[b. Giải Thưởng 3](#_Toc455278731)

[c. Khách Hàng 4](#_Toc455278732)

[1.1.3. Tính Năng Của Engine Unity 4](#_Toc455278733)

[II. TỔNG QUAN VỀ GAME ENGINE 6](#_Toc455278734)

[2.1.1. Khái Niệm 6](#_Toc455278735)

[2.1.2. Mục Đích 6](#_Toc455278736)

[2.1.3. Ảo hóa phần cứng (trừu tượng hóa phầncứng - Hardware Abstraction) 7](#_Toc455278737)

[2.1.4. Lịch sử 7](#_Toc455278738)

[2.1.5. Xu hướng hiện tại 8](#_Toc455278739)

[2.1.6. Phần mềm trung gian 8](#_Toc455278740)

[2.1.7. Phân loại Game Engine theo mức độ chuyên biệt 9](#_Toc455278741)

[a. Roll-your-own Game Engine 9](#_Toc455278742)

[b. Mostly-ready Game Engine 10](#_Toc455278743)

[c. Point-and-click Game Engine 10](#_Toc455278744)

[III. TỔNG QUAN KIẾN TRÚC GAME ENGINE 10](#_Toc455278745)

[3.1.1. Kiến trúc tổng quan 10](#_Toc455278746)

[**CHƯƠNG 2:** **UNITY ENGINE** 13](#_Toc455278747)

[I. GIỚI THIỆU CHUNG 13](#_Toc455278748)

[II. CÁC ĐẶC ĐIỂM VÀ TÍNH NĂNG CỦA UNITY 14](#_Toc455278749)

[2.2.1. Rendering (kết xuất hình ảnh) 14](#_Toc455278750)

[2.2.2. Lighting (ánh sáng) 15](#_Toc455278751)

[2.2.3. Terrains (địa hình) 15](#_Toc455278752)

[2.2.4. Substances (Texture thông minh) 15](#_Toc455278753)

[2.2.5. Physics (vật lí) 16](#_Toc455278754)

[2.2.6. . Pathfinding (tìm đường) 16](#_Toc455278755)

[2.2.7. Audio (âm thanh) 16](#_Toc455278756)

[2.2.8. Programming (lập trình) 16](#_Toc455278757)

[2.2.9. Networking 16](#_Toc455278758)

[III. CÁC THÀNH PHẦN TRONG UNITY 17](#_Toc455278759)

[2.3.1. Assets 17](#_Toc455278760)

[2.3.2. Scenes 17](#_Toc455278761)

[2.3.3. Game Object 18](#_Toc455278762)

[2.3.4. Components 19](#_Toc455278763)

[2.3.5. Scripts 19](#_Toc455278764)

[2.3.6. Prefabs 21](#_Toc455278765)

[IV. GIAO DIỆN CỦA UNITY 22](#_Toc455278766)

[2.4.1. Giao Diện 22](#_Toc455278767)

[2.4.2. Cửa sổ Scene và Hierarchy 23](#_Toc455278768)

[2.4.3. Cửa sổ Inspector 24](#_Toc455278769)

[2.4.4. Cửa sổ Project 26](#_Toc455278770)

[2.4.5. Cửa sổ Game 26](#_Toc455278771)

[**CHƯƠNG 3:** MỘT SỐ VẤN ĐỀ VÀ GIẢI PHÁP KHI XÂY DỰNG GAME ENGINE TRÊN UNITY……........ 28](#_Toc455278772)

[I. LOAD HOẠT HỌA 28](#_Toc455278773)

[3.1.1. Vấn Đề 28](#_Toc455278774)

[3.1.2. Giải Pháp 28](#_Toc455278775)

[**Kết luận** 30](#_Toc455278776)

[II. CHUYỂN ĐỘNG MÔ HÌNH NHẬT VẬT 2D 30](#_Toc455278777)

[3.2.1. Vấn Đề 30](#_Toc455278778)

[3.2.2. Giải Pháp 30](#_Toc455278779)

[**Kết Luận** 34](#_Toc455278780)

[III. THÊM SỰ KIỆN VÀO CHUYỂN ĐỘNG CỦA NHÂN VẬT 34](#_Toc455278781)

[3.3.1. Vấn đề 34](#_Toc455278782)

[3.3.2. Giải pháp 34](#_Toc455278783)

[**Kết luận** 36](#_Toc455278784)

[IV. TẠO HIỆU ỨNG PARTICLE 36](#_Toc455278785)

[3.4.1. Vấn đề 36](#_Toc455278786)

[3.4.2. Giải pháp 36](#_Toc455278787)

[**Kết luận** 42](#_Toc455278788)

[V. XÂY DỰNG GIAO DIỆN GAME 43](#_Toc455278789)

[3.5.1. Vấn đề 43](#_Toc455278790)

[3.5.2. Giải pháp 43](#_Toc455278791)

[**Kết luận** 47](#_Toc455278792)

[VI. Âm thanh trong game 47](#_Toc455278793)

[3.6.1. Vấn đề 47](#_Toc455278794)

[3.6.2. Giải pháp 47](#_Toc455278795)

[**Kết luận** 50](#_Toc455278796)

[**CHƯƠNG 4:** **ỨNG DỤNG GAME PHÁT TRIỂN TRÊN UNITY** 51](#_Toc455278797)

[I. GIỚI THIỆU GAME 51](#_Toc455278798)

[II. CÁC QUY LUẬT CHƠI CHÍNH 52](#_Toc455278799)

[4.2.1. Di chuyển 52](#_Toc455278800)

[4.2.2. Tấn Công 53](#_Toc455278801)

[4.2.3. Màn Chơi 55](#_Toc455278802)

[4.2.4. Điểm 56](#_Toc455278803)

[4.2.5. Máu 57](#_Toc455278804)

[4.2.6. Tạm Dừng Game 57](#_Toc455278805)

[III. DỮ LIỆU GAME BẮN MÁY BAY 58](#_Toc455278806)

[4.3.2. Code Trong Game 59](#_Toc455278807)

[IV. MỘT SỐ CLASS QUAN TRỌNG TRONG GAME 60](#_Toc455278808)

[4.4.1. Player Game 60](#_Toc455278809)

[4.4.2. Di chuyển Đối Tượng Trong Game 60](#_Toc455278810)

[4.4.3. Điểm 61](#_Toc455278811)

[V. PHÂN TICH SƠ ĐỒ 62](#_Toc455278812)

[4.5.1. Sơ Đồ Use Case 62](#_Toc455278813)

[4.5.2. Sơ Đồ Sequence 62](#_Toc455278814)

[**CHƯƠNG 5:** **KẾT LUẬN** 64](#_Toc455278815)

[ **Đạt Được** 64](#_Toc455278816)

[ **Chưa Đạt Được** 64](#_Toc455278817)

[ **Thuận Lợi** 65](#_Toc455278818)

[ **Khó Khăn** 65](#_Toc455278819)

[ **Kinh Nghiệm Rút Ra** 65](#_Toc455278820)

[ **Hướng Phát Triển** 65](#_Toc455278821)

1. **TỔNG QUAN VỀ ENGINE UNITY VÀ KHÔNG GIAN 2D, 3D**
2. TỔNG QUAN VỀ ENGINE UNITY
   * 1. Unity Là Gì ?

Đã qua rồi thời kỳ làm game trên nền Flash căn bản và buồn chán với những chuyển động thật cứng nhắc. Unity mang lại sức mạnh kỳ diệu cho nhân vật mà chúng ta muốn thể hiện sống động hơn trong không gian 3 chiều đầy huyền ảo. Công nghệ cao này tạo ra một bước đột phá mới về sự khác biệt trong công nghệ làm game hiện nay, mang đến cho người chơi 1 cảm giác rất khác lạ và hào hứng trong từng chuyển động, tương lai công nghệ này được áp dụng vào game Việt Nam sẽ mở ra một trang mới trong thế giới game 2D, 3D huyền ảo.

Unity được dùng để làm video game, hoặc những nội dung có tính tương tác như thể hiện kiến trúc, hoạt hình 2D, 3D thời gian thực. Unity hao hao với Director, Blender game engine, Virtools hay Torque Game Builder trong khía cạnh dùng môi trường đồ họa tích hợp ở quá trình phát triển game là chính.

Unity là một trong những engine được giới làm game không chuyên cực kỳ ưa chuộng bởi khả năng tuyệt vời của nó là phát triển trò chơi đa nền. Trình biên tập có thể chạy trên Windows và Mac OS, và có thể xuất ra game cho Windows, Mac, Wii, iOS, Android. Game cũng có thể chơi trên trình duyệt web thông qua plugin Unity Web Player. Unity mới bổ sung khả năng xuất ra game trên widget cho Mac, và cả Xbox 360, PlayStation 3.



Hình 1.1.1 Hình minh họa đa nền

Chỉ với khoản tiền bỏ ra khá khiêm tốn (1.500 USD) là phiên bản pro đã nằm trong tay của chúng ta, dĩ nhiên tại Việt Nam số tiền này vẫn là quá lớn nhưng thật may là đã có phiên bản Unity Free. Tuy nhiên, nhiều tính năng quan trọng (Network) bị cắt giảm nhưng đó không phải là vấn đề quá lớn nếu muốn phát triển một tựa game tầm trung.

Vào năm 2009, Unity nằm trong top 5 game engine tốt nhất cho việc sản xuất game với chỉ sau 4 năm phát triển. Unity đứng thứ 4, xếp sau Unreal Engine 3, Gamebryo Engine (được VTC Studio mua về phát triển SQUAD) và Cry Engine 2. Lượng tài liệu hướng dẫn Unity rất phong phú. Hơn thế nữa nó còn có sẵn một cộng đồng cực lớn với diễn đàn riêng. Bất cứ điều gì không hiểu chúng ta đều có thể thoải mái hỏi và nhận được câu trả lời nhanh chóng, tận tâm.

Quá trình tạo địa hình cũng như truy xuất từ các phần mềm 3DSMax, Maya, Cinema4D... rất nhanh chóng. Sức mạnh và sự tiện lợi của Unity là vô cùng lớn.

* Sức mạnh: Unity có thể tạo ra được nhiều loại game 2D, 3D đa dạng, dễ sử dụng với người làm game chưa chuyên nghiệp, chất lượng cao, chạy hầu hết trên các hệ điều hành.
* Sự tiện lợi: nếu chúng ta là một người chuyên dùng 3Dmax, hay Maya hoặc phần mềm mã nguồn mở Blender thì quả là thật tuyệt, chúng ta sẽ có một lợi thế lớn khi viết game trên Unity này, bởi công việc tạo các mô hình 2D, 3D sẽ trở lên dễ dàng hơn rất nhiều, việc kết hợp giữa người lập trình và người thiết kế các mô hình sẽ nhanh và hiệu quả hơn. Trong Unity chúng ta có thể import trực tiếp các file mô hình đang thiết kế và sẽ thiết kế hoàn thiện tiếp nếu chưa xong trong khi đó công việc import chỉ diễn ra một lần. Không như việc phải dùng các công cụ khác để thực hiện viết game chúng ta sẽ phải xuất chúng ra một dạng nào đó và mỗi lần sửa lại phần mô hình chúng ta lại phải import lại, và như thế là quá mất thời gian trong việc tạo và chỉnh sửa các mô hình theo ý muốn. Ngoài ra Unity còn cho chúng ta trực tiếp tạo các mô hình nếu muốn. Việc đặt các thuộc tính vật lý trong Unity cũng cực kỳ dễ dàng và hỗ trợ sẵn nhiều chức năng.
  + 1. Sơ Lược Lịch Sử Hình Thành Và Phát Triển Của Unity

1. 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 và bây giờ là phiên bản Unity 5. Có thể thấy tốc độ phát triển của Unity khá nhanh.

1. 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.

1. 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. Tính Năng Của Engine Unity
* Môi trường phát triển được tích hợp với tính năng kế thừa, khả năng chỉnh sửa đồ họa, chức năng kiểm tra chi tiết, và đặc biệt tính năng xem trước game ngay trong lúc xây dựng (live game preview).
* Triển khai được trên nhiều nền tảng:
* Chương trình độc lập trên Windows và Mac OS.
* Trên web, thông qua Unity Web Player plugin cho Internet Explorer, Firefox, Safari, Opera, Chrome, cho cả Windows và Mac OS.
* Trên Mac OS Dashboard widget.
* Cho Nintendo Wii (cần mua license thêm.)
* Cho iPhone, iPad application (cần mua license thêm.)
* Cho Google Android (cần mua license thêm.)
* Cho Microsoft Xbox 360 (cần mua license thêm.)
* Cho Sony PlayStation 3 (cần mua license thêm.)
* 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.
* Graphics engine sử dụng Direct3D (Windows), OpenGL (Mac, Windows), OpenGL ES (iPhone OS), và các API khác trên Wii.
* Hỗ trợ bump mapping, reflection mapping, parallax mapping, Screen Space Ambient Occlusion v...v...
* Unity Asset Server: Đây là một tính năng khá mới của Unity, theo đó Unity sẽ cung cấp một hệ thống quản lý theo dạng phiên bản cho tất cả asset và cả script. Đây là một kho chứa các tài nguyên cần thiết cho việc làm game. Khi import cũng như sửa chữa, trạng thái của asset ngay lập tức được cập nhật. Server chạy trên database opensource PostgreSQL và có thể truy cập trên cả Mac lẫn Windows, Linux. Asset Server đòi hỏi một khoản phí phụ trội là $499 cho mỗi bản copy Unity, và một license Unity Pro.

1. TỔNG QUAN VỀ GAME ENGINE
   * 1. Khái Niệm

Một Game Engine (hay công cụ tạo Game / động cơ Game) là một phần mềm được viết cho mục đích thiết kế và phát triển video Game. 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) 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), 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. Mục Đích

Game Engine cung cấp một bộ các công cụ phát triển trực quan và có thể tái sử dụng từng thành phần trong đó. Nói chung các bộ công cụ này cung cấp một môi trường phát triển tích hợp được đơn giản hóa. Phát triển ứng dụng nhanh (Rapid Application Development) cho Game theo cách lập trình hướng dữ liệu. Những Game Engine này đôi khi còn được gọi là các "phần mềm trung gian cho Game" (Game Middleware), như ý nghĩa của thuật ngữ, chúng cung cấp một nền tảng phần mềm linh hoạt và dễ dàng sử dụng lại với mọi chức năng cốt lõi cần thiết ngay trong nó để có thể phát triển một ứng dụng Game đồng thời giảm giá thành, độ phức tạp, và kịp thời hạn phát hành - tất cả các yếu tố quan trọng trong ngành công nghiệp Game đầy cạnh tranh. Giống như các phần mềm trung gian khác, Game Engine thường cung cấp một nền tảng trừu tượng hóa, cho phép một Game có thể chạy trên nhiều hệ máy bao gồm các hệ console hoặc máy tính cá nhân với một vài thay đổi (nếu cần) trong mã nguồn của Game đó. Thông thường, phần mềm trung gian cho Game được thiết kế với một nền tảng kiến trúc dựa trên các thành phần khác, cho phép các hệ thống khác nhau trong Engine có thể thay thế hoặc mở rộng với các phần mềm trung gian khác chuyên biệt hơn như là Havok cho hệ thống vật lý trong Game, Miles Sound System cho âm thanh, hay Bink cho các đoạn video. Một số Game Engine chỉ cung cấp khả năng dựng hình (kết xuất) 3D thời gian thực hay một khả năng riêng biệt nào khác thay vì rất nhiều chức năng trong phạm vi rộng mà Game yêu cầu. Loại Engine này thường được gọi là: "Graphics Engine", "Rendering Engine," hay "3D Engine" thay vì thuật ngữ bao quát hơn là "Game Engine”. Một vài ví dụ cho các Engine đồ họa là: RealmForge, Truevision3D, OGRE, Crystal Space, Genesis3D, Vision Engine, Irrlicht và JMonkey Engine.

* + 1. Ảo hóa phần cứng (trừu tượng hóa phầncứng - Hardware Abstraction)

Thông thường, Engine 3D hay hệ thống kết xuất đồ họa trong Game Engine thường được xây dựng trên một nền tảng giao diện lập trình ứng dụng đồ họa như Direct3D hay OpenGL cung cấp sẵn một hệ ảo hóa cho GPU hay card đồ họa. Thư viện mã nguồn cấp thấp (cấp cơ bản) như DirectX, SDL, và OpenAL (thư viện âm thanh mã nguồn mở) cũng thường được sử dụng trong các Game khi chúng cung cấp khả năng truy cập phần cứng độc lập.

* + 1. Lịch sử

Thuật ngữ "Game Engine" xuất hiện vào giữa những năm 90, đặc biệt là trong mối quan hệ giữa Game 3D. Như các thuơng hiệu nổi tiếng của id Sortware: Doom và Quake, thay vì phải làm việc từ đầu, các nhà phát triển khác (nếu được cấp phép) sẽ có quyền truy nhập vào phần lõi (mã nguồn) của Game và thiết kế các hình ảnh, nhân vật, vũ khí, và các màn chơi của riêng họ - gọi là Game Content (nội dung Game) hay "Game Assets" (tài sản Game). Các Game sau này, như Quake III Arena và sản phẩm năm 1998 của Epic Games: Unreal được thiết kế với cách tiếp cận mới này, Game Engine và nội dung Game được chia thành các phần riêng biệt để phát triển. Engine có thể tái sử dụng khiến việc phát triển Game tiếp theo nhanh hơn và dễ dàng hơn, một thuận lợi to lớn trong ngành công nghiệp cạnh tranh này. Game Engine hiện đại là một trong những ứng dụng được viết ra (bằng các ngôn ngữ lập trình) phức tạp nhất, thường xuyên phải có rất nhiều tinh chỉnh trong hệ thống để đảm bảo kiểm soát chính xác trải nghiệm người dùng. Sự phát triển liên tục của Game Engine đã tạo ra một sự phân chia mạnh mẽ giữa các công việc dựng hình, viết kịch bản, thiết kế hình ảnh, và thiết kế màn chơi. Hiện nay thông thường một đội ngũ phát triển Game điển hình phải có số lượng họa sĩ gấp vài lần số lượng lập trình viên.

* + 1. Xu hướng hiện tại

Với công nghệ tạo Game Engine càng phát triển và trở nên thân thiện hơn với người sử dụng, ứng dụng của nó càng được mở rộng, và giờ đây được sử dụng để tạo các Game mang mục đích khác với giải trí đơn thuần như: mô phỏng, huấn luyện ảo, y tế ảo, và mô phỏng các ứng dụng quân sự. Nhiều Game Engine đang được tạo ra bằng các ngôn ngữ lập trình cấp cao như Java và C# hay .NET (ví dụ: TorqueX, và Visual3D.NET) hay Python (Panda3D). Vì hầu hết các Game 3D hiện nay đều có giới hạn cho GPU (giới hạn bởi sức mạnh của card đồ họa), khả năng gây chậm máy của các ngôn ngữ lập trình cấp cao trở nên không đáng kể, trong khi việc tăng năng suất được cung cấp bởi các ngôn ngữ này lại có lợi cho các nhà phát triển Game Engine.

* + 1. Phần mềm trung gian

Một số hãng hiện nay chuyên tập trung vào phát triển các bộ phần mềm gọi là phần mềm trung gian (Middleware). Các nhà phát triển Middleware cố gắng phát triển một bộ phần mềm mạnh mẽ bao gồm hầu hết các yếu tố mà một nhà phát triển Game cần đến để xây dựng một Game. Phần lớn các Middleware cung cấp các cơ sở để phát triển game một cách dễ dàng, như hình ảnh, âm thanh, hệ thống vật lý và chức năng AI. Gamebryo và RenderWare là hai phần mềm trung gian được sử dụng khá rộng rãi. Một số Middleware chỉ làm một chức năng nhưng nó còn hiệu quả hơn cả toàn bộ Engine. Ví dụ, SpeedTree được sử dụng để dựng hình ảnh cây cối thêm thực tế trong Game nhập vai The Elder Scrolls IV: Oblivion. Bốn gói phần mềm trung gian được sử dụng rộng rãi nhất, thường được cung cấp dưới dạng một chức năng bổ trong một hệ thống Engine gồm có Bink của hang RAD Game Tools, hãng Firelight với FMOD, Havok, và GFx của Scaleform. RAD Game Tools phát triển Bink cho dựng hình video cơ bản, đi cùng với âm thanh Miles, và dựng hình 3D Granny . FMOD của Firelight là một thư viện và bộ công cụ phần mềm âm thanh mạnh mẽ giá rẻ. Havok cung cấp một hệ thống mô phỏng vật lý mạnh, cùng với các bộ phần mềm về hình ảnh động và mô phỏng hành vi. Scaleform đem lại GFx dùng cho các biểu diễn cao cấp trên nền giao diện Flash, cùng với các giải pháp trình chiếu video chất lượng cao, và một Input Method Editor (IME) phần bổ sung nhằm hỗ trợ trò chuyện trong Game bằng ngôn ngữ Châu Á.

Phân loại Game Engine theo mức độ chuyên biệt

1. Roll-your-own Game Engine

Roll-your-own Engine có thể được xem như những Engine ở mức thấp nhất. Ngày nay rất nhiều công ty tự tạo ra Engine cho riêng họ với những thứ có thể. Họ tận dụng những gói ứng dụng phổ biến, các API như XNA, DirectX, OpenGL, các API và SDL của Windows và Linux. Ngoài ra, họ có thể dùng những thư viện từ nhiều nguồn khác nhau, được mua hoặc Open-Source. Những thư viện đó có thể bao gồm cả những thư viện vật lý như Havok và ODE, những thư viện Scene-Graph như OpenSceneGraph và các thư viện GUI như AntTweakBar. Thực tế, những hệ thống “tự sản xuất” đó cho phép những lập trình viên dễ tương tác hơn rất nhiều vì họ biết cần gì và chọn những thành phần phù hợp từ đó tạo nên chính xác thứ cần thiết. Nhưng cũng vì vậy mà tốn rất nhiều thời gian. Thêm vào đó, những lập trình viên thường gặp rất nhiều lỗi khi các thư viện từ nhiều nguồn không tương thích với nhau. Điều này đã góp phần làm cho những Engine dạng này kém hiệu quả và không được ưa chuộng ngay cả với Engine được thiết kế một cách chuyên nghiệp.

1. Mostly-ready Game Engine

Mostly-ready Game Engine, những Engine nằm ở tầng trung. Các nhà thiết kế Game nghiệp dư rất thích những Engine dạng này. Chúng được thiết kế rất đầy đủ với mọi tính năng cần thiết, vẽ, tương tác, GUI, vật lý… ngoài ra chúng còn kèm theo rất nhiều tool mạnh. Những Engine tầng này khá phong phú từ ORGE và Genesis3D (open-source), rẻ tiền như Torge hoặc đắc tiền như Unreal, id Tech và Gamebryo. Về mức độ, những Engine kể trên đều cần lập trình thêm để gắn kết mọi thứ với nhau nhằm tạo nên một Game hoàn chỉnh. Hầu như mọi Mostly-ready Game Engine đều có phần hạn chế một ít so với roll-your-own Engine. Vì mặc dù nó là sản phầm của rất nhiều người tham gia phát triển trải qua thời gian dài nhằm mang đến cho khách hàng những “tiện nghi” tốt nhất nhưng không phải khách hàng nào cũng giống nhau và phát triển những Game tương tự nhau.

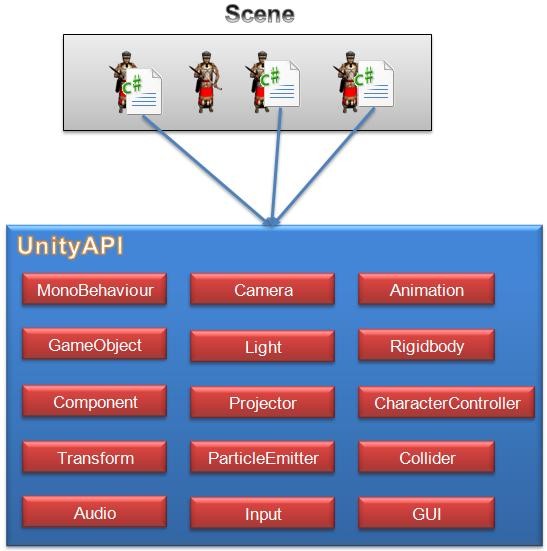
1. Point-and-click Game Engine

Point-and-click Engine được xếp ở tầng cao nhất. Những Engine này ngày càng trở nên phổ biến. Nó có tất cả mọi tool với đầy đủ chức năng cho phép chúng ta chỉ việc chọn, click và chọn để tạo ra Game. Những Engine như vậy, kể cả GameMaker, Torque Game Builder, và Unity3D được thiết kế để càng thân thiện với người dùng càng tốt. Nó cũng yêu cầu ít lập trình hơn. Điều này không có nghĩa việc lập trình không cần, nhưng nếu so với 2 loại Engine kể trên trên thì loại Engine này cần ít nhất. Nhược điểm rất lớn của những Engine point-and-click này đến từ việc những nhà phát triển bị hạn chế rất nhiều, vì với loại Engines này, chúng chỉ có thể cho ra một loại giao diện như tương tự nhau hoặc khác nhau một ít. Nhưng điều này không có nghĩa chúng vô dụng. Vì khi đối mặt với những hạn chế đó, những sáng tạo của con người sẽ càng được kích thích cao. Ngoài ra, Engine này cho phép chúng ta có thể làm việc nhanh hơn, làm ra Game nhanh hơn mà không phải làm quá nhiều.

1. TỔNG QUAN KIẾN TRÚC GAME ENGINE
   * 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 1.1.2 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. **UNITY ENGINE**
2. GIỚI THIỆU CHUNG



Hình 2.1.1: Logo của Unity Engine

* Nhà phát triển: Unity Technologies
* Phiên bản mới nhất : 5 (phát hành vào ngày 11/4/2012)
* Được viết bởi ngôn ngữ : C++, C#
* Phát triển Game cho các hệ điều hành : Windows, Mac OS X (tạo và đóng gói), Wii, iPhone/iPad, Xbox 360, Android, PS3 (chỉ đóng gói ; cần giấy phép bổ sung cho từng nền tảng)
* Giấy phép: Độc quyền
* Website: [www.unity3d.com](http://www.unity3d.com)

Unity là một 3D Game Engine, là một công cụ thiết kế Game dành cho PC, Mac và nhiều hệ máy di động khác.

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. Những đoạn code do chúng ta viết sẽ được Unity biên dịch ra Mono trước khi nó được thực thi. Điều này rất quan trọng cho Game để thực thi code vào những thời điểm cần thiết trong suốt thời gian chạy (Runtime).

Ngoài thư viện Mono, Unity cũng tận dụng chức năng của những thư viện phần mền 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 thế chúng ta không cần phải lo lắng về việc phải học thế nào để sử dụng chúng một cách riêng lẽ.

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

Có thể sản xuất các trò chơi theo tiêu chuẩn chuyên nghiệp, xuất bản 3D cho cả Mac và PC cũng như sở hữu riêng một Web Player của riêng mình, Unity là một trong những Game Engine có tốc độ phát triển nhanh nhất. Unity cũng có phiên bản phát triển Game cho hệ máy Wii của Nintendo và Iphone của Apple, có nghĩa là một khi chúng ta làm chủ được những vấn đề cơ bản, không chỉ phát triển Game cho máy tính cá nhân mà chúng ta còn có thể phát triển Game cho các hệ máy console và thiết bị di động.

1. CÁC ĐẶC ĐIỂM VÀ TÍNH NĂNG CỦA UNITY
   * 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 (Redering) cùng nhiều hỗ trợ cho phép áp dụng các công nghệ phổ biến trong lĩnh vực đồ họa 3D nhằm cải thiện chất lượng hình ảnh. Các phiên bản gần đây nhất của Unity được xây dựng lại thuật toán nhằm cải thiện hiệu suất kết xuất hình ảnh đồng thời tăng cường chất lượng hình ảnh sau khi kết xuất.

Một số hỗ trợ:

* Unity cung cấp sẵn 100 Shaders với đầy đủ các loại phổ biến nhất.
* Hỗ trợ Surface Shaders, Occlusion Culling, GLSL Optimizer.
* Hỗ trợ LOD.
  + 1. Lighting (ánh sáng)

Ánh sáng là một điều thiết yếu giúp môi trường trở nên đẹp và thực tế hơn. Unity cũng 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. Terrains (địa hình)

Terrains còn gọi chung là địa hình bao gồm phần đất nền của môi trường trong trò chơi cùng các đối tượng gắn liền như cây, cỏ, …

Unity cung cấp một công cụ hỗ trợ 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 hình và lát Texture cho địa hình. Cùng với Terrain 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. Substances (Texture thông minh)

Substances có thể hiểu đơn giản là một dạng tùy biến Textures nhằm làm đa dạng chúng trong nhiều điều kiện môi trường khác nhau. Unity cung cấp khả năng này thông qua các API dựng sẵn trong thư viện, hỗ trợ lập trình viên lập trình để tùy biến hình ảnh được kết xuất của Texture

* + 1. Physics (vật lí)

PhysX là một Engine mô phỏng và xử lí vật lý cực kỳ 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 vật lý. Một số vấn đề vật lý được hỗ trợ bởi Unity như: Soft Bodies, Rigitbodies, Ragdolls, Joints, Cars, …

* + 1. . Pathfinding (tìm đường)

Đây là một tính năng rất mới mẻ đến 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. Tuy nhiên ở phiên bản 3.5 đến nay, Unity hỗ trợ cho chúng ta tính năng Pathfinding cho phép tạo ra khả năng tìm đường cho AI nhờ vào khái niệm lưới định hướng (NavMesh).

* + 1. Audio (âm thanh)

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

* + 1. Programming (lập trình)

Lập trình là một trong những yếu tố quan trọng nhất trong phát triển Game. Lập trình cho phép nhà phát triển tạo nên khả năng tương tác, trí thông minh và yếu tố Gameplay cho trò chơi.

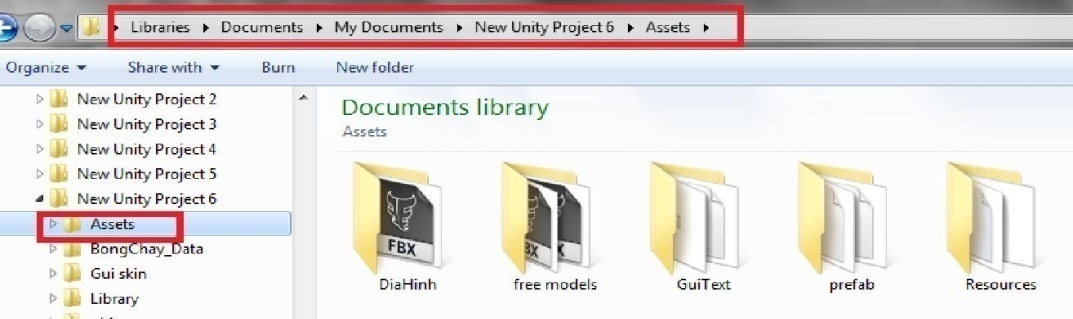
Unity cho phép chúng ta lập trình bằng nhiều ngôn ngữ mạnh mẽ và phổ biến với các lập trình viên như: C#, Java Scrip và Boo.

* + 1. Networking

Networking cho phép chúng ta tạo ra các trò chơi trực tuyến (online) – một trong những thể loại trò chơi thu hút được nhiều người chơi nhất. Tính năng này sẽ hỗ trợ đầy đủ để chúng ta tạo nên các khía cạnh phổ biến trong Game online như hệ thống điểm kinh nghiệm , chat và tương tác thời gian thực, …

Một số tính tăng cung cấp bởi Networking như: State Synchronization, Realtime Networking, Remote Procedure Calls, Backend Connectivity, Web Browser Integration, Web Connectivity.

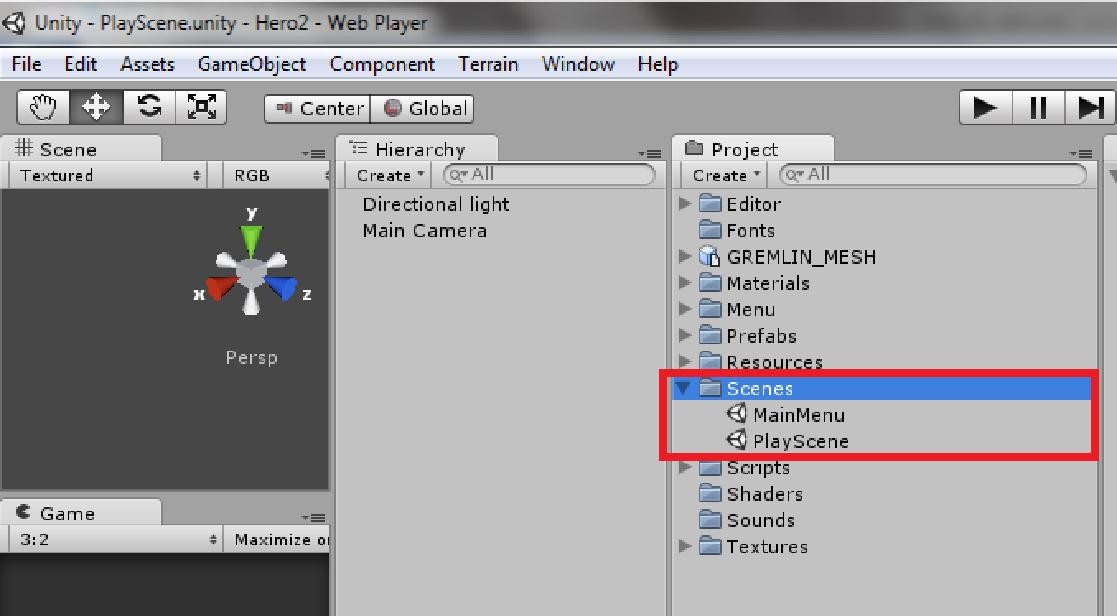
1. CÁC THÀNH PHẦN TRONG UNITY
   * 1. Assets

Assets là những tài nguyên xây dựng nên một dự án Unity. Từ những tập tin hình ảnh, mô hình 3D đến các tập tin âm thanh. Unity gọi các tập tin mà chúng ta dùng để tạo nên trò chơi là tài sản (Assets). Điều này lí giải tại sao tất cả các tập tin, thư mục của các dự án Unity đều được lưu trữ trong một thư mục có tên là “Assets”.

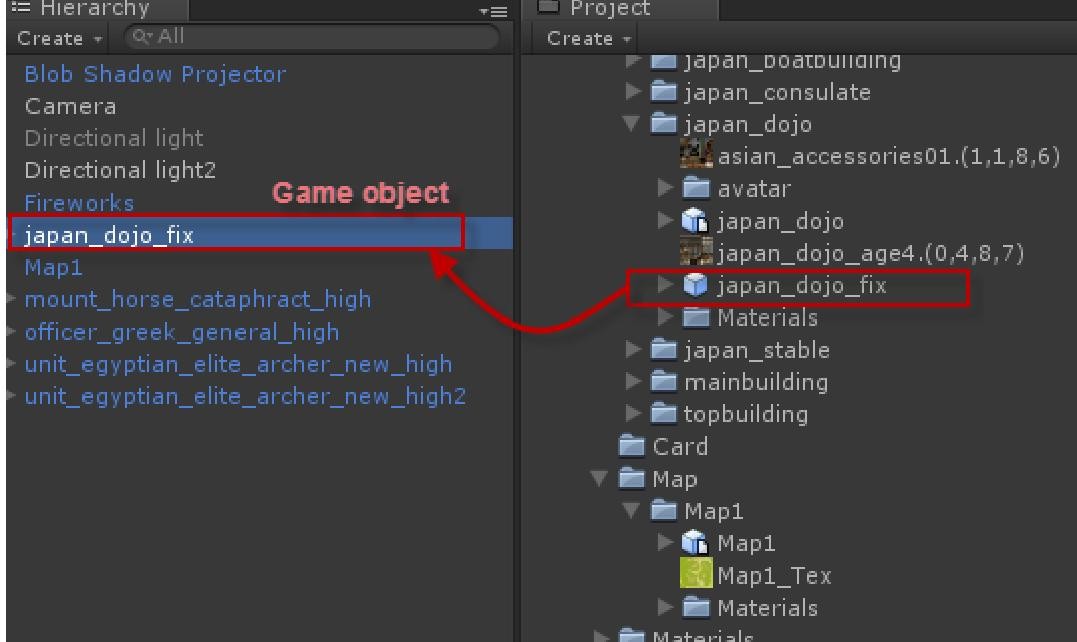
Hình 2.1.2 Asset trong Uinty

* + 1. Scenes

Trong Unity, chúng ta cần hiểu một cảnh (hay một phân đoạn) nghĩa là một màn chơi riêng biệt hoặc một khu vực hay thành phần có trong nội dung của trò chơi (ví dụ như Game menu). Bằng cách tạo nên nhiều Scenes cho trò chơi, chúng ta có thể phân phối thời gian tải hoặc kiểm tra các phần khác nhau của trò chơi một cách riêng lẽ.

Hình 2.1.3 Các Scene của Unity

* + 1. Game Object

Khi Assets 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 xoay 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 sẽ tạo ra Game Object với các thành phần khác, bổ sung chức năng cần thiết để xây dựng nên bất kỳ một thành phần nào trong kịch bản Game mà chúng ta đã tưởng tượng.

Hình 2.1.4 Kéo tài nguyên vào Scene để sử dụng

* + 1. Components

Components có nhiều hình thức khác nhau. Chúng có thể xác định hành vi, cách xuất hiện,… hay ảnh hưởng đến các khía cạnh khác trong chức năng của Game Object trong trò chơi. Bằng cách “gắn” chúng vào trong Game Object, chúng ta ngay lập tức có thể áp dụng tác động của chúng lên đối tượng. Những Components phổ biến trong quá trình phát triển trò chơi đều được Unity hỗ trợ sẵn. Ví dụ như thành phần Rigidbody đã được đề cập hay các yếu tố đơn giản khác như ánh sáng, Camera và nhiều thành phần khác. Để tạo nên các yếu tố tương tác trong trò chơi, chúng ta sẽ sử dụng Script (mã kịch bản), chúng cũng được xem như là một Components trong Unity.

Hình 2.1.5 Các thành phần trong đối tượng Camera

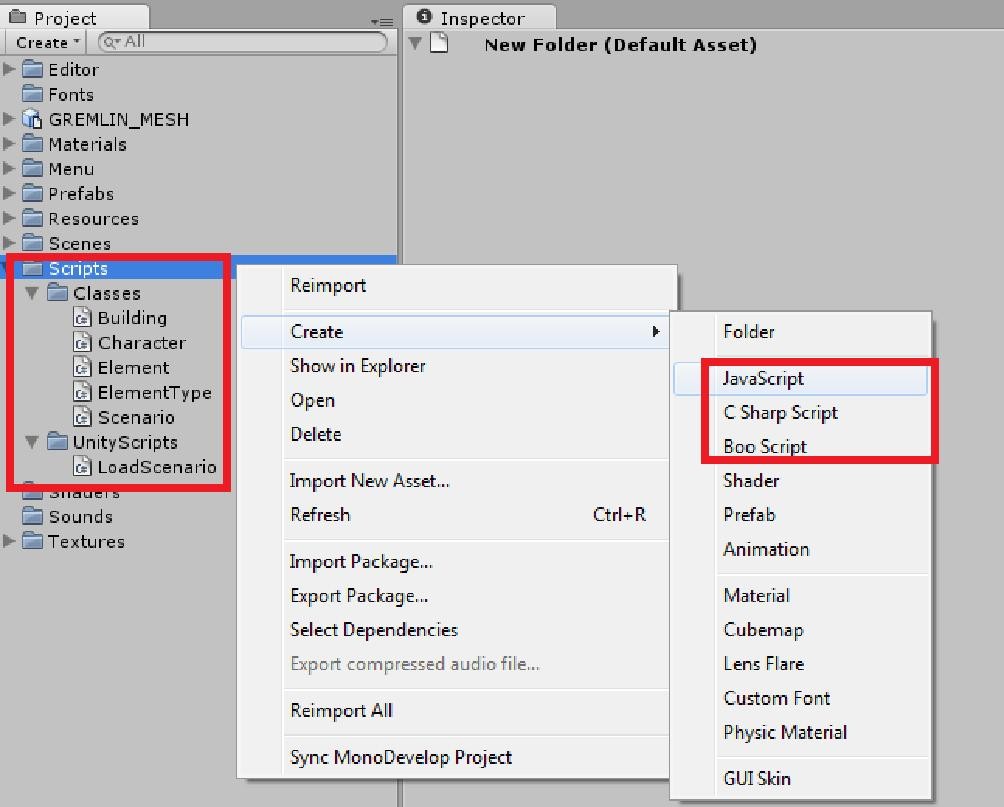
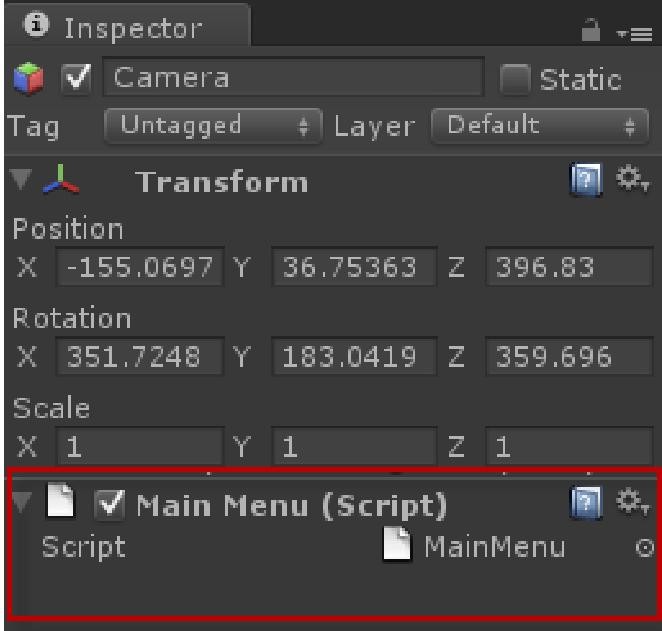
* + 1. Scripts

Được Unity xem như một Components, Script là một thành phần thiết yếu trong quá trình phát triển trò chơi và đáng được đề cập đến như một khái niệm “chìa khóa”. Unity cung cấp cho chúng ta khả năng viết Script bằng cả 3 loại ngôn ngữ là: JavaScript, C# và Boo (một dẫn xuất của ngôn ngữ Python).

Unity không đòi hỏi chúng ta phải học làm thế nào để lập trình trong Unity, nhưng hầu như chúng ta phải sử dụng Script tại mỗi thành phần trong kịch bản mà chúng ta phát triển. Unity đã xây dựng sẵn một tập hợp đa dạng các lớp, hàm mà chúng ta hoàn toàn có thể ứng dụng trong quá trình lập trình cho trò chơi của mình.

Để viết script, chúng ta sẽ làm việc với một trình biên tập Script độc lập của Unity, hoặc với chương trình Mono Developer được tích hợp và đồng bộ với Unity trong những phiên bản mới nhất hiện nay.

Mono developer là một IDE khá tốt để lập trình khi cung cấp nhiều chức năng tương tự như Visual studio. Mã nguồn viết trên Mono Developer sẽ được cập nhật và lưu trữ trong dự án Unity.

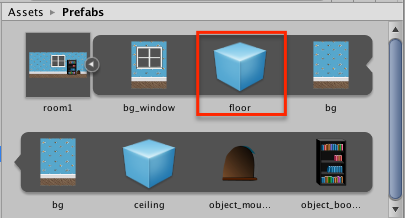
Hình 2.1.6 Cách tạo file script mới

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

Hình 2.1.7 Một file Script đang gắn vào đối tượng

* + 1. Prefabs

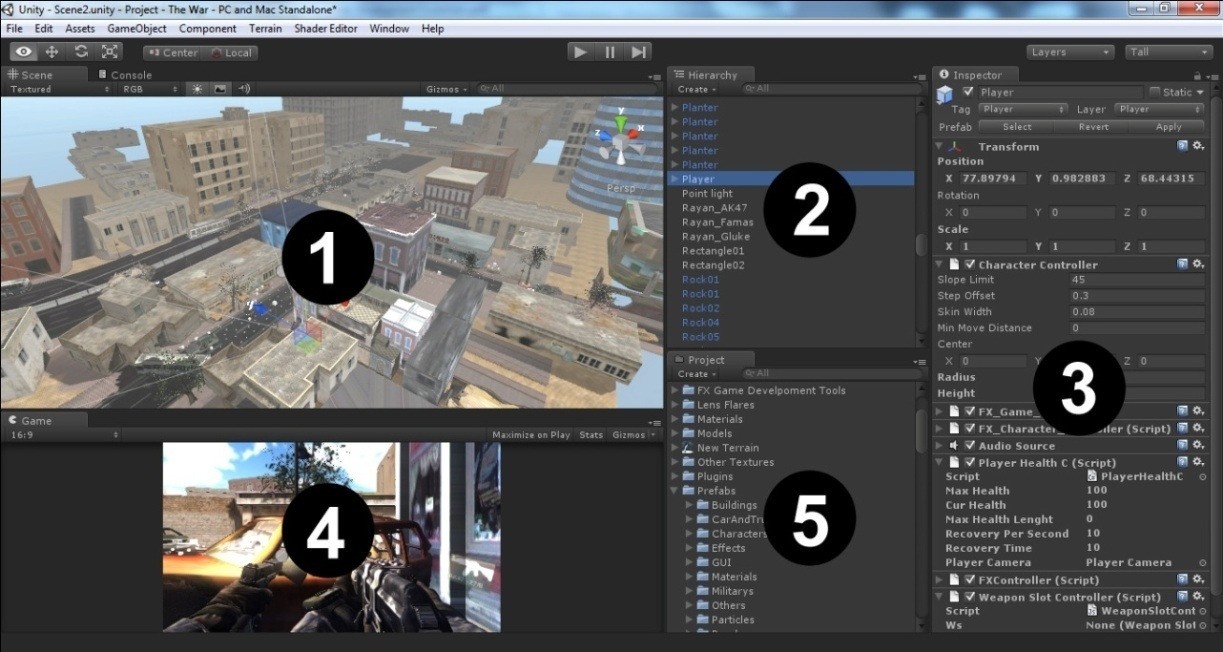
Prefabs cho phép chúng ta lưu trữ các đối tượng với những Components và những thiết đặt hoàn chỉnh. Có thể so sánh với khái niệm cơ bản là MovieClip trong Adobe Flash, Prefabs chỉ đơn giản là một Container (một đối tượng chứa) rỗng mà chúng ta có thể đưa bất kì một đối tượng hay dữ liệu mẫu nào mà chúng ta muốn tái sử dụng về sau.



Hình 2.1.8 Một số đối tượng trong Prefabs

1. GIAO DIỆN CỦA UNITY
   * 1. Giao Diện

Giao diện của Unity có khả năng tùy chỉnh bố trí tương tự như nhiều môi trường làm việc khác. Dưới đây là một kiểu bố trí điển hình trong Unity:



Hình 2.1.9 Giao diện của Unity.

∗ Chú thích:

* Scene (1): Nơi mà trò chơi sẽ được xây dựng.
* Hierarchy (2): Danh sách các Game Object trong scene.
* Inspector (3): Những thiết lập, thành phần, thuộc tính của đối tượng (hoặc Asset) đang được chọn.
* Game (4): Cửa sổ xem trước, nó chỉ hoạt động trong chế độ “Play” (Preview – xem trước).
* Project (5): Danh sách các Assets của dự án, được ví như thư viện của dự án.
  + 1. Cửa sổ Scene và Hierarchy

Cửa sổ scene là nơi mà chúng ta sẽ xây dựng các thực thể, đối tượng của dự án vào đó. Cửa sổ cung cấp góc nhìn phối cảnh (Perspective (góc nhìn 3D), chúng ta có thể chuyển qua các góc nhìn khác như từ trên xuống hoặc từ dưới lên (Top Down), từ trái sang phải hoặc phải sang trái (Side On), từ trước ra sau hoặc sau đến trước (Front On). Cửa sổ này sẽ kết hình xuất đầy đủ những hình ảnh trong thế giới của trò chơi mà chúng ta tạo ra dưới dạng một vùng biên tập mà chúng ta có thể biên tập, chỉnh sửa trực tiếp thế giới đó.

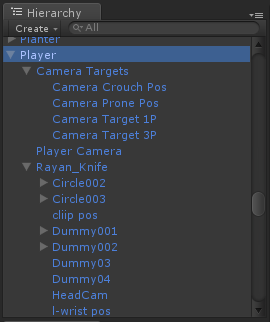
Khi kéo thả Assest vào cửa sổ Scene, Assets sẽ trở thành Game Object. Cửa sổ Scene được ràng buộc cùng với cửa sổ Hierarchy, cửa sổ Hierarchy liệt kệ danh sách các Game Object có trong Scene và được sắp xếp theo thứ tự chữ cái từ A-Z.



Hình 2.1.10. Các nút chức năng cho cửa sổ Scene**.**

Cửa sổ Scene còn đi kèm với 4 bốn nút chức năng hữu ích được hiển thị dưới dạng hình ảnh như trên. Chúng có thể được lựa chọn thông qua các phím tắt Q, W, E và R. Những nút này có các chức năng như sau:

* Công cụ bàn tay (Q): Công cụ này cho phép chúng ta di chuyển đến một khu vực nào đó trong Scene bằng thao tác kéo thả thuộc trái.
* Công cụ di chuyển (W): Công cụ này cho phép chúng ta chọn một đối tượng trong cảnh và thực hiện thao tác di chuyển, thay đổi vị trí của đối tượng đó. Khi chọn, tại vị trí của đối tượng sẽ hiển thị các trục và mặt phẳng gắn liền với đối tượng cho phép chúng ta di chuyển đối tượng trượt theo các trục, mặt phẳng hoặc di chuyển một cách tùy ý.
* Công cụ xoay (E): Công cụ này có đặc điểm và cách sử dụng giống với công cụ di chuyển, tuy nhiên thay vì để di chuyển vị trí của đối tượng thì công cụ này giúp chúng ta xoay đối tượng xoay quanh trục hay tâm của đối tượng.
* Công cụ điều chỉnh tỉ lệ (R): Cũng tương tự như công cụ di chuyển và xoay, công cụ này cho phép chúng ta tùy chỉnh kích thước, tỉ lệ của đối tượng một cách tùy ý.



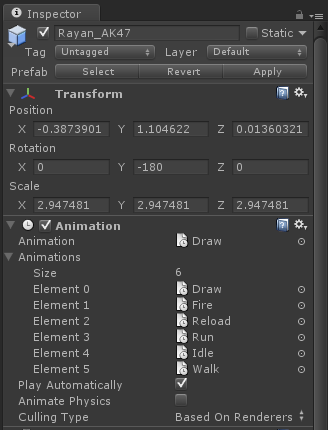
Hình 2.1.11 Cửa sổ Hierarchy.

* + 1. Cửa sổ 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 Game Object hoặc Assets đang được chọn.

Cửa sổ này sẽ hiển thị đầy đủ các Components 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-dowm Menu…

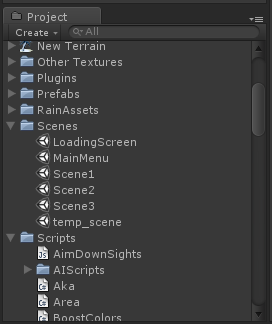
Ngoài việc hiển thị các Component của đối tượng được chọn, cửa sổ Inspector còn hiển thị các thiết đặt chung của hệ thống hay của trò chơi khi ta chọn chúng từ menu Edit.



Hình 2.1.12 Cửa sổ Inspector.

Trong hình trên, chúng ta thấy cửa sổ Inspector đang hiển thị một vài thuộc tính, Components của một đối tượng đang được chọn. Trong đó, bao gồm 2 Components là Transform và Animation. Cửa sổ Inspector sẽ cho phép chúng ta thay đổi các thiết đặt trên. Các Components này còn có thể được tạm thời vô hiệu hóa vào bất kỳ lúc nào chúng ta muốn bằng cách bỏ chọn Checkbox ở góc trên bên trái của mỗi Component, việc này sẽ rất hữu ích cho chúng ta khi muốn kiểm tra hay thử nghiệm các Components này. Ngoài ra, cửa Inspector còn cho phép chúng ta vô hiệu hóa toàn bộ một đối tượng đang được chọn bằng cách bỏ chọn Checkbox ở trên cùng góc trái của cửa sổ Inspector.

* + 1. Cửa sổ Project



Hình 2.1.13 Cửa sổ Project.

Cửa sổ Project là cửa sổ cho phép chúng ta nhìn thấy trực tiếp nội dung của thư mục Assets của dự án. Mỗi dự án Unity đều được chứa trong một thư mục cha. Trong đó có 3 thư mục con là Assets, Library và Temp (chỉ có khi Unity đang chạy). Đặt tất cả các Assets vào thư mục Assets có nghĩa là ngay lập tức chúng ta sẽ thấy chúng xuất hiện trong cửa sổ Project. Ngoài ra, khi thay đổi vị trí của Assets trong thư mục Assets hay lưu tập tin lại từ một chương trình ứng dụng thứ 3 nào khác (ví dụ như Photoshop), sẽ làm cho Unity nhập lại (Re-Import) Assets, phản ánh sự thay đổi này ngay lâp tức trong cửa sổ Project và Scene có sử dụng Assets vừa được thay đổi.

Cửa sổ Project được tích hợp nút Create, nút này cho phép chúng ta tạo mới bất kì một Assets mới nào, ví dụ như Script, Prefabs, Materials, …

* + 1. Cửa sổ Game

Cửa sổ Game sẽ được gọi khi chúng ta nhấn vào nút Play (là một hành động thực hiện test trò chơi). Cửa sổ này cho phép chúng ta tùy chọn về thiết đặt tỉ lệ màn hình, nó phản ánh phạm vi trong Scene mà người chơi có thể thấy được với mỗi tỉ lệ màn hình tương ứng, ví dụ như với mỗi tỉ lệ màn hình 4:3, 16:9 thì người chơi sẽ có một phạm vi nhìn thấy khác nhau.

Sau khi nhấn vào nút Play, chúng ta sẽ ở chế độ Testing, lúc này mọi thay đổi về các thuộc tính, Components, … của đối tượng sẽ chỉ là tạm thời. Tức là chúng sẽ trở về như ban đầu (trước khi nhấn nút Play) sau khi kết thúc chế độ Testing.



Hình 2.1.14 Các loại hình ảnh trong cửa sổ game.

1. MỘT SỐ VẤN ĐỀ VÀ GIẢI PHÁP KHI XÂY DỰNG GAME ENGINE TRÊN UNITY
2. LOAD HOẠT HỌA
   * 1. Vấn Đề

Game 2D được xây dựng từ nhiều mô hình 2D được đặt lên không gian 2 chiều sao cho hài hòa với nhau để tạo thành cảnh vật trong game. Do đó việc nạp và hiển thị được mô hình 2D trong game là vô cùng quan trọng.

Mô hình 2D được cấu tạo từ rất nhiều đa giác để tạo nên khối vật thể. Ngày nay, trong một mô hình 2D không chỉ đơn thuần chứa một khối vật thể mà nó bao gồm nhiều khối vật thể được gắn kết với nhau trên một khung xương. Điều này giúp cho mô hình không bị gắn chết một chuyển động vào bên trong và dễ dàng thay đổi chuyển động cho mô hình.

* + 1. Giải Pháp

Các mô hình 2D, 3D thông thường được thiết kế sẵn bằng các phần mềm thiết kế 3D chuyên dụng như [Adobe InDesign](http://www.dayhocdohoa.com/search/label/tai-lieu-indesign), [Adobe Photoshop](http://www.dayhocdohoa.com/search/label/tai-lieu-photoshop), [CorelDRAW](http://www.dayhocdohoa.com/search/label/tai-lieu-corel-draw), ... Sau đó, mô hình sẽ được đưa vào game engine để sử dụng.

Engine Unity hỗ trợ rất nhiều định dạng mô hình 2D, 3D khác nhau như: PNG, JPG... khi mô hình được load vào project ta sẽ chuyển mô hình sang mục Prefab để có thể tái sử dụng nhiều lần.

Unity có hỗ trợ load mô hình bằng cách kéo thả Prefab vào vị trí bất kỳ trong Scene.

Tuy nhiên, để linh hoạt hơn thì chúng ta có thể xử lý bằng code.

Trước tiên, ta tạo ra một file script và gắn nó vào một đối tượng trong game bất kì để đoạn script có thể thức thi. Trong file script này, ta khai báo một đối tượng kiểu GameObject để lưu mô hình và dung hàm Instantiate() để khởi tạo mô hình này ở vị trí góc quay mong muốn.

Tuy nhiên, câu hỏi là đối tượng GameObject trên chứa mô hình nào.ở đây có 2 giải pháp để trỏ GameObject vào mô hình

public class LoadObject : MonoBehaviour

{

public GameObject Obj;

void Awake()

{

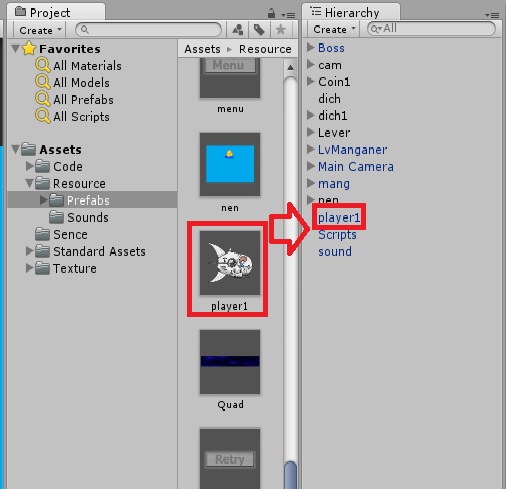
GameObject NewObj = Instantiate(Obj, new Vector3(0, 0, 0), Quaternion.identity);

}}

* *Giải pháp thứ nhất:*

Load mô hình từ prefab chứa bên ngoài resource

Trên cửa sổ Inspector của đối tượng game được gắn script vào xuất hiện thuộc tính Obj. Ta chọn Prefab mong muốn và kéo thả vào thuộc tính Obj.



Hình 3.1.1 Minh họa kéo thả prefab vào thuộc tính của script

* *Giải pháp thứ hai*:

Load mô hình từ Prefab chứa trong resource

Để load được Prefab từ resource, ta phải đặt Prefab đó trong thư mục Resources của project. Sau đó, dung hàm Resources.Load() để load Prefab với tham số là đường dẫn của Prefab tính từ thư mục Resources.

Obj = Resources.Load("Knights");

//... Instantiate mô hình



Hình 3.1.2 Prefab trong resources

**Kết luận**

Rõ ràng giải pháp thứ hai cho thấy sự linh động hơn trong việc load mô hình từ Prefab, đặc biệt trong lúc runtime. Mọi thao tác biến đổi, xoay, chuyển động sau đó đều thực hiện trên đối tượng GameObject này.

1. CHUYỂN ĐỘNG MÔ HÌNH NHẬT VẬT 2D
   * 1. Vấn Đề

Chúng ta đã load được mô hình 2D vào trong game, vậy làm sao để mô hình 2D này có thể chuyển động trong game.

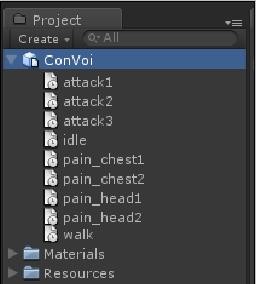
* + 1. Giải Pháp

Trước tiên mô hình 2D cần phải có sẵn animation bên trong. Khi import mô hình vào Unity, animation trong mô hình được tự động chuyển thành một **AnimationClip**. Điều này giúp animation này có thể dùng cho các mô hình khác trong project.

Trước hết ta phải tạo AnimationClip từ animation có sẵn của mô hình. Có 2 loại mô hình 2D có sẵn animation:

* *Loại thứ nhất:*

Mô hình 2D có chứa nhiều animation bên trong.

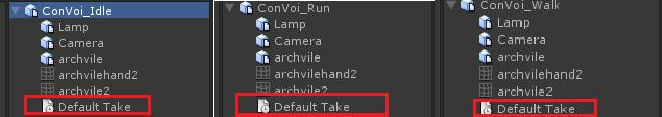


Hình 3.2.1 Mô hình 2D bên trong chứa nhiều animation

Mô hình 2D trên sau khi import vào project game, bên trong đã có sẵn 9 animation, mỗi animation sẽ tự động được tạo thành một AnimationClip bên trong đối tượng game.

* *Loại thứ hai:*

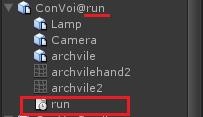
Mô hình 2D chỉ chứa một animation.

Hình 3.2.2 Mô hình 2D chứa một animation

Trong 3 mô hình trên, mỗi mô hình sau khi import có một AnimationClip duy nhất có tên “Default Take”. Về bản chất hình dạng mô hình là như nhau, chỉ khác nhau animation (Idle, Run, Walk). Vậy làm sao chúng ta kết hợp các AnimationClip này vào một đối tượng game duy nhất. Unity quy định như sau:

* Lấy một mô hình làm mô hình chính, có thể không cần animation kèm theo cũng được.
* Các mô hình còn lại, tên phải có 2 phần cách nhau bởi ‘@’, phần đầu phải trùng tên với mô hình chính đã chọn, phần thứ 2 sẽ là tên của animation.

Với cách đặt tên như vậy, khi đưa các mô hình này vào project để sử dụng thì Unity sẽ tự động đổi tên animation mặc định trong mô hình thành tên trùng với phần tên mô hình nằm sau chữ ‘@’. Lưu ý là phải đổi tên cho mô hình từ bên ngoài project tức trên Windows vì nếu như chúng ta đổi tên trực tiếp trong project thì tên của animation của mô hình đó sẽ không bị thay đổi theo phần tên sau dấu ‘@’.



Hình 3.2.3 Hình minh họa sau khi đổi tên và import vào project

Như hình trên, mô hình sẽ có 3 AnimationClip bên trong (Default Take, idle, run).

Tiếp theo, để gọi thực hiện một AnimationClip trong một đối tượng ta dùng hàm CrossFade() của thuộc tính animation trong GameObject. Đoạn code sau đây đang được gắn vào đối tượng game cần chạy animation.

animation.CrossFade("Run"); // Run là tên của AnimationClip trong mô hình

Đôi khi ta muốn chạy các chuyển động khác nhau của mô hình một cách tuần tự. Để làm điều này ta dùng hàm ***CrossFadeQueued()*** của thuộc tính **animation** trong GameObject.

// AnimationClip Idle sẽ chạy sau khi AnimationClip Run chạy xong animation.CrossFade("Run");

animation.CrossFadeQueued("Idle");

Để điều khiển cách chạy của AnimationClip, ta dùng thuộc tính **wrapMode**

animation.wrapMode = WrapMode.Loop; // chuyển động lặp lại liên tục

**Kết Luận**

Việc gọi thực hiện các animation của đối tượng là khá đơn giản. Tuy nhiên phải quyết định chọn mô hình loại nào để có thể thêm hoặc bớt animation cho mô hình dễ dàng. Nếu chọn mô hình loại 1 thì chúng ta phải import vào các chương trình hỗ trợ làm animation cho mô hình để chỉnh sửa thêm xóa animation rồi import vào Unity lại, còn chọn mô hình loại 2 thì chúng ta chỉ cần xóa hay thêm file mô hình là xong, rất linh hoạt và nhanh chóng.

1. THÊM SỰ KIỆN VÀO CHUYỂN ĐỘNG CỦA NHÂN VẬT
   * 1. Vấn đề

Trong lúc lập trình kịch bản game, chúng ta muốn biết khi nào một nhân vật chuyển động xong để có bước xử lý tiếp theo. Ví dụ như sau khi nhân vật thực thi chuyển động chết thì chúng ta phải hủy đối tượng đó khỏi bộ nhớ. Trong Unity, khi một chuyển động chạy xong không tự phát ra sự kiện.

* + 1. Giải pháp

Unity hỗ trợ lớp **AnimationEvent** giúp thêm sự kiện vảo frame bất kỳ trong một AnimationClip. Trước hết ta phải có một file script chứa hàm sẽ thực thi sau khi sự kiện xảy ra. Chẳng hạn, file script có nội dung như sau (chứa hàm hủy đối tượng):

public class UnitDieCallback : MonoBehaviour

{

void Die()

{

// hủy đối tượng được attach file script này Destroy(gameObject);

}

}

Sau đó chúng ta tạo file script khác dùng để thêm event và gắn file script này vào đối tượng bất kỳ. Đoạn script mẫu bên dưới đang được gắn vào một mô hình 2D .

public class PlayAnimation : MonoBehaviour

{

void Start()

{

AnimationEvent animEvent = new AnimationEvent(); animEvent.functionName = "Die";

animEvent.time = obj.animation["die"].clip.length;

gameObject.animation["die"].clip.AddEvent(animEvent);

}

}

Với đoạn code trên, chúng ta đã thêm một sự kiện vào frame cuối cùng của AnimationClip “die”. Hàm được thực thi khi sự kiện xảy ra là hàm **Die()**.

Nếu chạy đoạn code trên sẽ xảy ra lỗi thực thi vì không tìm thấy hàm Die(). Chúng ta phải thêm file script chứa hàm Die() vào đối tượng với đoạn code sau:

gameObject.AddComponent<UnitDieCallback>();

Khi chạy game, lúc đối tượng chạy animation “die” xong thì sự kiện sẽ được xảy ra. Nếu chạy animation khác thì sự kiện không xảy ra.

##### **Kết luận**

Việc thêm sự kiện vào mô hình giúp ta kiểm soát chuyển động dễ dàng hơn và đưa ra các xử lý thích hợp ở thời điểm nhất định.

1. TẠO HIỆU ỨNG PARTICLE
   * 1. Vấn đề

Các hiệu ứng thường gặp trong game như mưa, tuyết rơi, khói, lửa, hiệu ứng phép,… sẽ làm cho game sinh động và ấn tượng hơn, nhất là với game 3D thì các hiệu ứng này càng cần thiết hơn. Các hiệu ứng này được gọi chung là hiệu ứng particle.

* + 1. Giải pháp

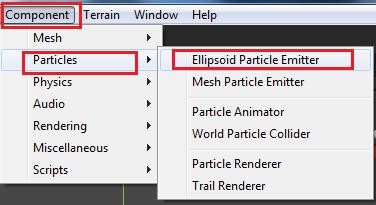
Để làm được điều này, Unity hỗ trợ người dùng Particle Systems để tạo ra bất kỳ hiệu ứng particle nào mà người dùng mong muốn. Particle muốn hiển thị được phải có 3 thành phần chính quan trọng sau:

* **Particle Emitter**: để sinh ra các hạt.
* **Particle Animator**: để làm di chuyển các hạt theo thời gian.
* **Particle Renderer**: để vẽ các hạt.

Để tạo một particle, chúng ta thực hiện như sau:

* *Bước 1:*

Tạo thành phần quan trọng nhất để sinh ra các hạt - thành phần **Particle Emitter**. Thành phần này không thể tạo trực tiếp từ code mà chỉ có thể thêm từ giao diện Editor của Unity bằng cách chọn menu **Component**  **Particles**  **Ellipsoid Particle Emiter**.

Hình 3.4.1 Thêm thành phần Ellipsoid Particle Emitter

* *Bước 2*

Tạo 2 thành phần còn lại là Particle Animator và Particle Renderer. Bước này chúng ta có thể thực hiện bằng code hoặc trên giao diện.

public class Particle : MonoBehaviour

{

void Start ()

{

//tạo thành phần ParticleRenderer để vẽ các hạt

ParticleRenderer pRen = gameObject.AddComponent<ParticleRenderer>();

//tạo thành phần ParticleAnimator để chạy các hạt

ParticleAnimator pAmin = gameObject.AddComponent<ParticleAnimator>();

}

}

Phương thức **AddComponent<ParticleRenderer>()** để thêm một thành phần **Particle Renderer** vào gameObject. Tương tự như vậy với thành phần **Particle Animator**. Cách lấy và gán thuộc tính thông qua biến trả ra của hàm AddComponent().

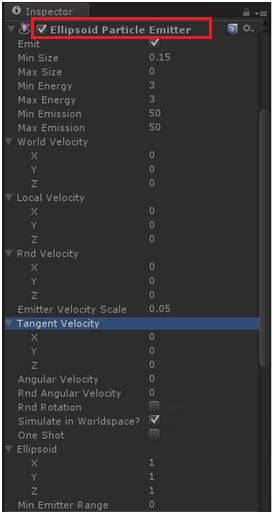
Để gọi và thay đổi các thuộc tính của thành phần Particle Emiter chúng ta sử dụng thuộc tính **particleEmiter** của GameObject:

particleEmitter.maxSize = 1F; particleEmitter.minSize = 0.15F;

* *Bước 3:*

Tùy chỉnh các thuộc tính riêng của từng thành phần để có được một hiệu ứng như mong muốn. Xem qua các thuộc tính của 3 thành phần trên giao diện editor để thấy rõ hơn:

Hình 3.4.2 Các thuộc tính của Particle System



* Các thuộc tính của thành phần tạo hạt, **Elipsoid Particle Emiter** được trình bày trong bảng 3.1

Bảng 3.4.3 Các thuộc tính của **Elipsoid Particle Emiter**

|  |  |  |
| --- | --- | --- |
| **Thuộc tính** | **Ý nghĩa** | **Hình minh họa** |
| Emit | Nếu enable hiệu ứng sẽ phát ra. | Mặc định |
| Min Size / Max Size | Kích thước nhỏ nhất/lớn nhất có thể của mỗi hạt tại thời điểm sinh ra. | Max Size = 0.7 |
| Min Energy / Max Energy | Thời gian sống nhỏ nhất/lớn nhất của hạt, tính bằng giây. | Max Energy = 7 |
| Min Emisson / Max Emisson | Số lượng tối thiểu/tối đa của hạt được phát ra, tính bằng giây. | Max Emission = 1 |
| World Velocity | Tốc độ bắt đầu của hạt theo các chiều x, y, z trong không gian. | WorldVelocity.x = 1 |
| Local Velocity | Tốc độ bắt đầu của các hạt trong cùng một vùng x, y, z | LocalVelocity.x = 1 |
| Rnd Velocity | Vận tốc ngẫu nhiên của các hạt cùng chiều x, y, z. | RndVelocity.x=3 |
| Tangent Velocity | Vận tốc khởi đầu cho các hạt cùng chiều trên bề mặt của Emitter. | TangentVelocity.x= 0.5 |
| OneShot | Nếu enable thì hiệu ứng hiện một lần rồi tắt, sau đó hiện lên lại. Nếu tắt thì hiệu ứng hiện liên tục. | Hiện và ẩn  cả khối hiệu ứng |

* Các thuộc tính của thành phần vẽ, **ParticleRenderer** được trình bày trong bảng 3.2

Bảng 3.4.4 Các thuộc tính của **ParticleRenderer**

|  |  |  |
| --- | --- | --- |
| **Thuộc tính** | **Ý Nghĩa** | **Hình minh họa** |
| Materials | Chất liệu của particle, có thuộc tính size và texture, texture để chứa các texture bên ngoài đưa vào giúp hiệu ứng đẹp hơn. | Texture: |
| Stretch Particles | Độ căng giản của các hạt. | StretchParticles= HorizontalBillboard |
| Length Scale | Tỷ lệ độ dài của các hạt | StretchParticles= Stretched, LengthScale=10 |
| Velocity Scale | Vận tốc của các hạt nếu dựa vào thuộc tính “*Stretch Particles*”. | Velocity Scale =15 |

* Các thuộc tính của thành phần làm chuyển động, **ParticleAnimator** được trình bày trong bảng 3.3

Bảng 3.4.5 Các thuộc tính của Particle Animator

|  |  |  |
| --- | --- | --- |
| **Thuộc tính** | **Ý Nghĩa** | **Hình minh họa** |
| Color Animation | Màu sắc của hạt, chu kỳ màu của hạt sẽ mất nhanh nếu hạt nào có tốc độ nhanh hơn. | DoesAnimateColor= true |
| Does Animate Color | Nếu không bật thì các chu kỳ màu loang sẽ mất, nếu bật thì các chu kỳ màu loang sẽ có tác dụng. | DoesAnimateColor= false |
| World  Rotation Axis/ Local Rotation Axis | Phép quay quanh các trục x, y, z. | WorldRotationAxis.x =5 |
| Force | Tưởng tượng như có một sức gió đang thổi vào các hạt theo các chiều x, y, z. | Force.x =-1 |
| Damping | Sự tắt dần chuyển động của từng hạt. | Damping.x=2.4 |

##### **Kết luận**

Nắm rõ các thuộc tính của 3 thành phần chính của Particle System trong Unity và biết cách áp dụng một cách linh hoạt, chúng ta hoàn toàn có thể tạo ra các hiệu hạt khác nhau và đẹp mắt để đưa vào game của mình.

1. XÂY DỰNG GIAO DIỆN GAME
   * 1. Vấn đề

Giao diện đồ họa người dùng là một phần quan trọng không thể thiếu trong khi xây dựng một ứng dụng game hay bất cứ một ứng dụng nào để vẽ các đối tượng đồ họa như Button, Label, Checkbox, Slider, … lên màn hình.

* + 1. Giải pháp

Để làm được điều này chúng ta dùng lớp GUI, GUI là chữ viết tắt của “Graphical User Interface” – “Giao diện đồ họa người dùng”. Hệ thống GUI của Unity được gọi là GUIUnity. Để sử dụng được các phương thức trong GUI ta phải gọi thực hiện từ trong hàm **OnGUI()** giống như sự kiện Paint trong C#. Ví dụ sau đây sẽ tạo ra một button đơn giản:

**void OnGUI()**

**{**

**if (GUI.Button(new Rect(10, 10, 150, 100), "I am a button"))**

**{**

**print("You clicked the button!");**

**}**

**}**



Hình 3.5.1 GUI Button

Để tạo một Button hay một thành phần trong GUI đẹp hơn, chúng ta sử dụng **GUIStyle** - style cài đặt cho các thành phần của GUI. GUIStyle chứa thông tin về font chữ, vị trí đặt biểu tượng, hình nền, khoảng cách, … Chỉ cần định nghĩa và áp dụng cho một bất kỳ thành phần nào như Button, Label, Checkbox,… Ta có thể xem GUIStyle như CSS khi thiết kế web site vậy.

Ví dụ sau tạo một Label theo một định nghĩa GUIStyle, với kích thước font là 14, chữ in đậm, canh lề chữ ở giữa, màu chữ đỏ, font Beckasin (được lưu trước trong thư mục Resources), vị trí vẽ ra (Screen.width/2, Screen.height/2), kích thước hình chữ nhật bao quanh 150x50:

void OnGUI()

{

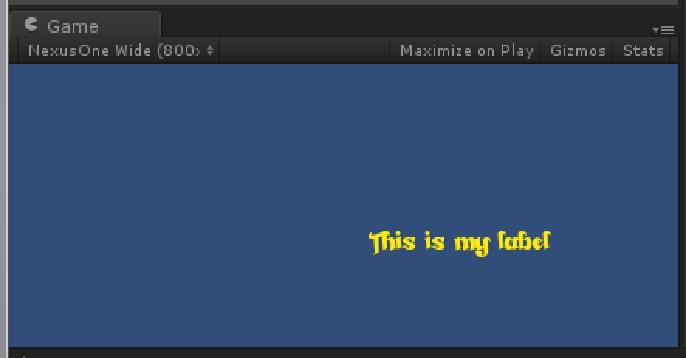
GUIStyle myStyle = new GUIStyle(); myStyle.fontSize = 14; myStyle.fontStyle = FontStyle.Bold;

myStyle.alignment = TextAnchor.MiddleCenter; myStyle.normal.textColor = Color. yellow;

myStyle.font = (Font)Resources.Load("Fonts/Beckasin");

GUI.Label(new Rect(Screen.width/2,Screen.height/2, 150,50),"This is my label",myStyle);

}



Hình 3.5.2 Áp dụng GUIStyle lên Label

Ví dụ sau đây sẽ vẽ ra một Button với background là hình. Vì chỉ vẽ hình nên chúng ta không cần quan tâm đến định dạng text trong GUIStyle nữa, chúng ta chỉ quan tâm đến hình nền của button với các sự kiện chuột tương tác trên button này mà thôi.

void OnGUI()

{

GUIStyle myStyle = new GUIStyle();

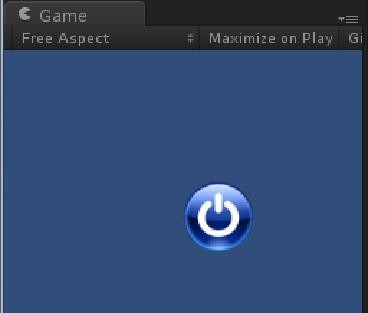
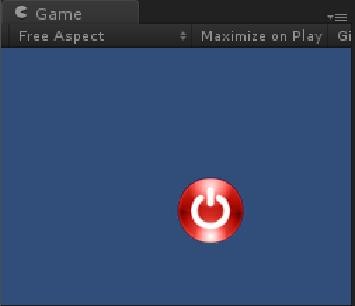
myStyle.normal.background = (Texture2D)Resources.Load("Button/exit"); myStyle.hover.background = (Texture2D)Resources.Load("Button/exit\_hover");

if (GUI.Button(new Rect(Screen.width / 2, Screen.height / 2, 50, 50), "My button", myStyle)) {

//do something....

**}**

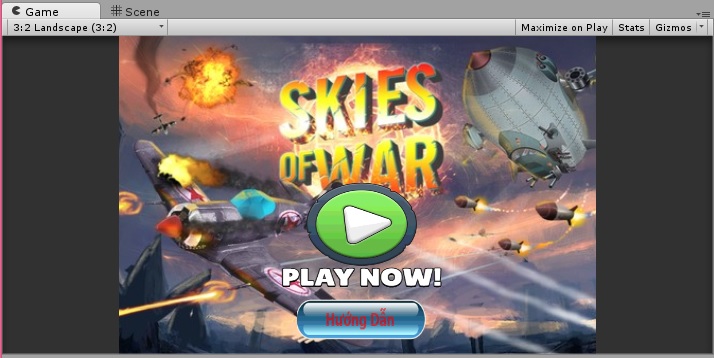
**}**



Hình 3.5.3 Button khi rê chuột và không rê chuột

Để vẽ một Image ra màn hình, chúng ta sử dụng Texture2D, nó tương tự đối tượng Bimap hay Image bên ngôn ngữ C#. Để load hình từ Resources và vẽ ra màn hình chúng ta dùng phương thức **DrawTexture()** như sau:

Texture2D myImage = (Texture2D)Resources.Load("background\_image"); GUI.DrawTexture(new Rect(0,0,Screen.width,Screen.height), myImage);



Hình 3.5,4 Vẽ hình ảnh trên GUI

**Kết luận**

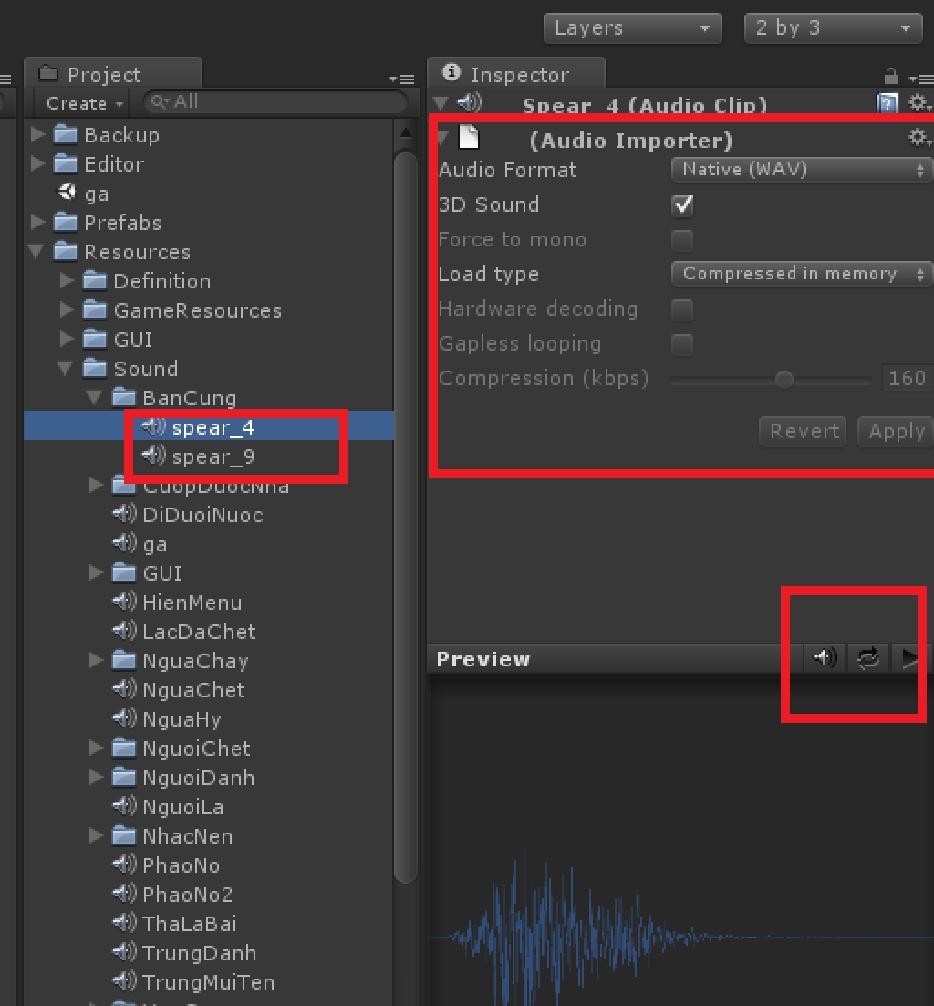
Với lớp GUI trong Unity, chúng ta hoàn toàn có xây dựng nên một giao diện tuyệt vời cho ứng dụng game. Ngoài các phương thức của lớp GUI đã nêu trên thì còn rất nhiều phương thức vẽ các đối tượng khác như Radio, Checkbox, Slider…

1. Âm thanh trong game
   * 1. Vấn đề

Âm thanh là yếu tố không kém phần quan trọng trong ứng dụng game. Thật nhàm chán khi một cảnh đánh đánh nhau, bắn nhau hay các hiệu ứng đẹp mắt mà không có âm thanh. Âm thanh 3 chiều sẽ làm cho game thực hơn và sống động hơn.

* + 1. Giải pháp

Để chơi được một file âm thanh trong Unity có 2 cách: bằng code hoặc trên giao diện. Dù chọn cách nào thì trước hết chúng ta phải có sẵn các file âm thanh và import vào project. Sau khi import âm thanh vào project, nếu file hợp lệ chúng ta sẽ thấy như hình sau và có thể nhấn nút play để nghe thử.

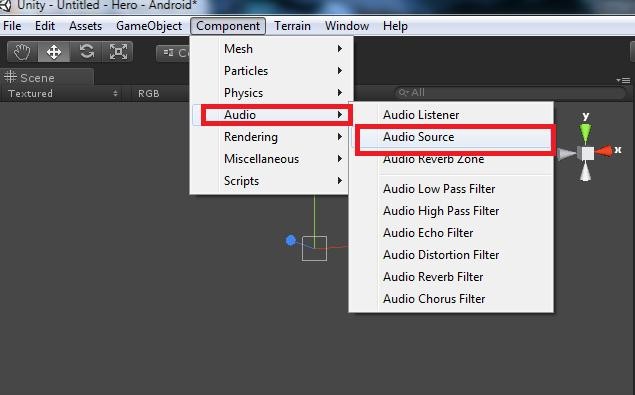


Hình 3.6.1 Thông tin file âm thanh

* *Cách 1:* Tạo trên giao diện

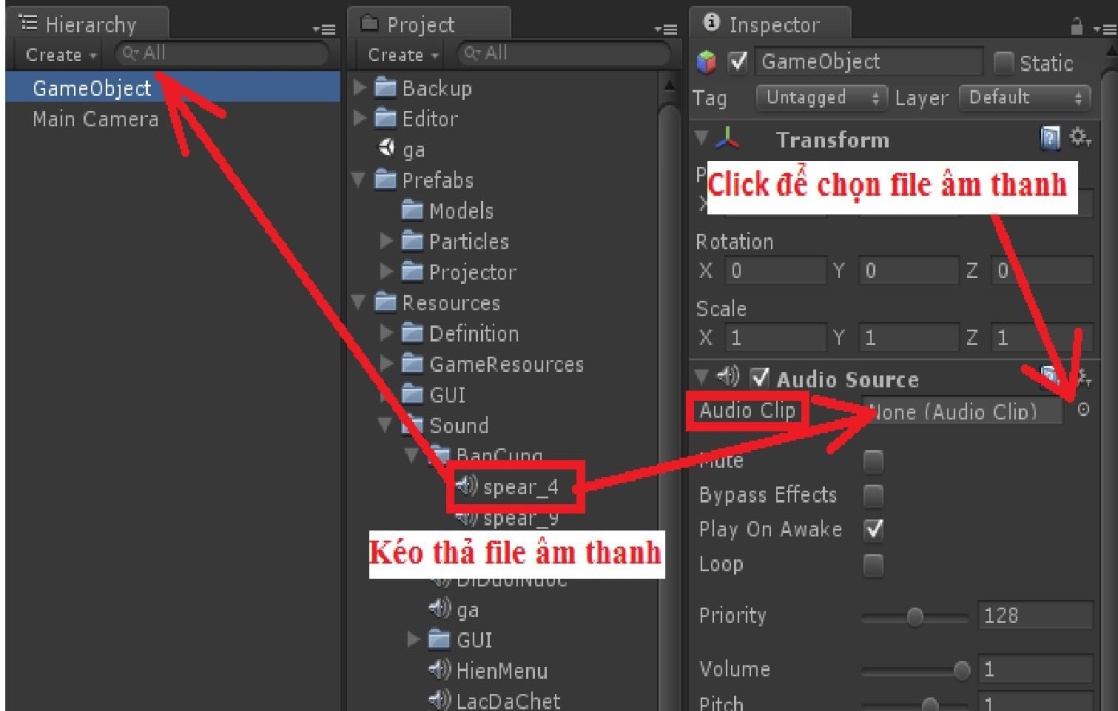
Trên menu của Unity, vào **GameObject  Create Empty**.

Chọn đối tượng vừa tạo và gắn thành phần “AudioSource” cho đối tượng này. AudioSource là một đối tượng âm thanh. Muốn Play hay Stop, thay đổi cách lặp, tăng giảm volume nhạc thì phải thông qua đối tượng này.



Hình 3.6.2 Thêm thành phần Audio Source

Sau khi gắn thành phần âm thanh cho đối tượng vừa tạo, chúng ta dễ dàng chỉnh sửa các thông số và gán file âm thanh cho thành phần AudioSource này.



Hình 3.6.3 Thêm file âm thanh cho thành phần AudioSource

Nếu thực hiện xong các bước trên, chúng ta có thể chạy game để nghe thử. Việc tạo đối tượng âm thanh trên giao diện khá đơn giản, nhưng để áp dụng vào cho game thì không được linh hoạt bằng cách sử dụng script.

* *Cách 2:* Cách chơi nhạc bằng code

Trước tiên chúng ta cần import file âm thanh vào project trước.

Khởi tạo đối tượng game âm thanh **AudioSource** như sau:

AudioSource audioSource = (AudioSource)gameObject.AddComponent("AudioSource");

Gán đường dẫn file nhạc cho âm thanh:

//play nhạc 1 lần audioSource.PlayOneShot();

//play nhạc lặp đi lặp lại audioSource.loop = true; audioSource.Play();

//stop chơi nhạc audioSource.Stop();

audioSource.clip = (AudioClip)Resources.Load("TenFileAmThanh");

Sau đó chỉ cần gọi các phương thức Play() để chạy file âm thanh:

##### **Kết luận**

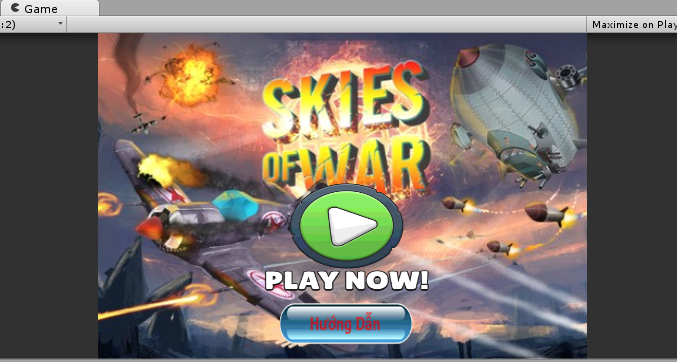
Để chơi được âm thanh trong Unity thì chỉ cần áp dụng các kỹ thuật nêu trên là đủ. Ngoài ra còn nhiều thành phần khác như: AudioListener, AudioSetting để tạo hiệu ứng âm thanh 3 chiều thực hơn cho game.

1. **ỨNG DỤNG GAME PHÁT TRIỂN TRÊN UNITY**
2. GIỚI THIỆU GAME

Đặc trưng của dòng game theo màn mà em đã xây dựng tại 1 thời điểm chỉ có một người chơi, trò chơi sẽ có 4 màn hình chơi, người chơi có thể chọn bất kì màn nào mà mình muốn, mỗi màn chơi người chơi sẽ được cung cấp 1 lượng máu và vũ khí nhất định, người chơi sẽ bắn để loại bỏ quân địch và cố gắng ăn càng nhiều tiền thì điểm càng cao, mỗi màn sẽ có 1 lượng tiền rất lơn và càng về sau tiền càng nhiều nhưng độ khó sẽ tăng lên, khi người chơi bị trúng đạn quá nhiều dẫn đến hết máu thì người chơi sẽ chết và màn hình GameOver sẽ xuất hiên thong báo cho người chơi số điểm, người chơi cũng có thể xem điểm cao ở button HighScore.

Màn hình sẽ di chuyển và quân địch cũng từ đó mà xuất hiện, càng về sau quân địch càng đông và khó chết vì vậy người chơi phải khéo léo di chuyển để tránh né đạn của địch giết càng nhiều địch thì càng có nhiều cơ hội ăn vàng để nâng cao điểm số.

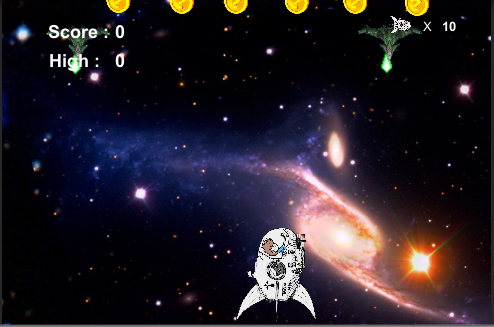
Sau khi người chơi kết thúc 1 màn chơi người chơi sẽ được hỏi có chơi tiếp màn tiếp theo hay không, nếu đồng ý người chơi sẽ click vào buton và chuyển qua màn kế tiếp.



Hình 4.1.1 màn hình để bắt đầu chơi

1. CÁC QUY LUẬT CHƠI CHÍNH
   * 1. Di chuyển

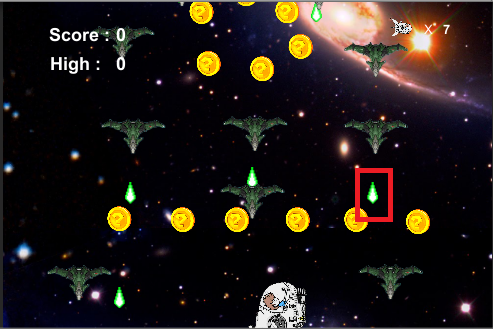
Người chơi sẽ di chuyển bằng các nút trên bàn phím, di chuyển qua trái, qua phải, lên, xuống để né đạn và tiêu diệt kẻ thù, làm sao chơi được càng lâu thì càng được nhiều điểm, người chơi chỉ có thể di chuyển trong phạm vi màn hình camara.



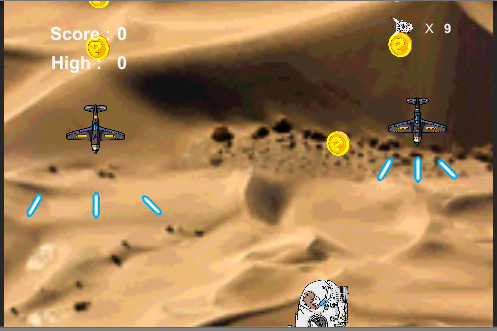
Hình 4.2.2 Màn hình để chơi game

* + 1. Tấn Công

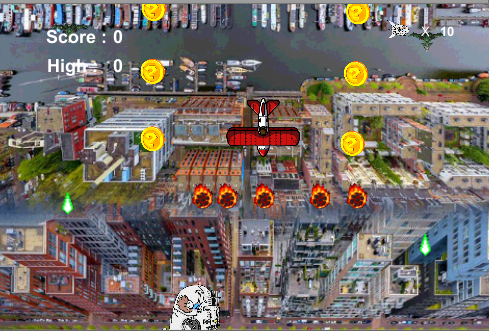
Người chơi sẽ bắn đạn để tiêu diệt kẻ thù, mỗi kẻ thù sẽ có một lượng máu nhất định, có máy bay ít máu và có máy bay nhiều máu nên người chơi cần phải bắn liên tục để tiêu diệt kẻ thù, máy bay màu xanh sẽ có lượng máu từ 1 đến 2 máu và bắn ra 1 đạn, màu đen có lượng máu lớn hơn 2 đến 3 máu, bắn ra 3 viên đạn và máy bay đỏ sẽ có từ 3 đén 4 máu, bắn ra bốn viên đạn và sát thương cũng mạnh hơn.



Hình 4.2.3 đạn của máy bay xanh



Hình 4.2.4 đạn của máy bay đen



Hình 4.2.5 đạn của máy bay đỏ

* + 1. Màn Chơi

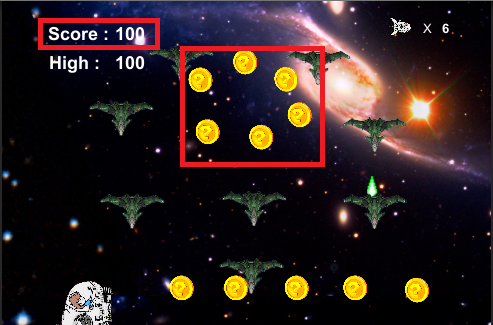
Màn chơi sẽ có 4 màn chơi, người chơi sẽ được chọn 1 trong 4 màn để chơi game, màn đâu tiên sẽ dễ hơn và càng về sau màn chơi sẽ càng khó quân địch sẽ xuất hiên nhiều và đạn bắn ra sát thương sẽ mạng hơn, mỗi màn chơi sẽ có những địa hình khác nhau màn 1 người chơi sẽ được bay lượn bên trên những vì sao, bắn hạ kẻ thù đang xâm chiếm để bảo vệ trái đất còn màn 2 người chơi sẽ di chuyển ngoài xa mạc, màn 3 người chơi sẽ được trải nghiệm cảm giác bay bên trên thành phố tuyệt đẹp, và màn 4 sẽ là đồi núi.



Hình 4.2.6 có 4 màn để người chơi lựa chọn

* + 1. Điểm

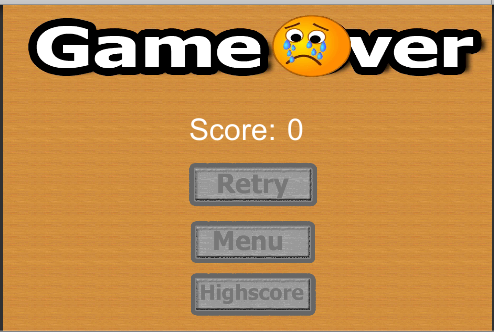
Người chơi sẽ được cộng điểm khi ăn được 1 vàng, mỗi lần ăn được 1 vàng người chơi sẽ được cộng 100 điểm, nếu người chơi va chạm với máy bay địch thì điểm sẽ bị trừ đi một nửa đồng tiền vàng có nghĩa là trừ đi 50 điểm.



Hình 4.2.7 Vàng và điểm cho người chơi

* + 1. Máu

Mỗi màn người chơi sẽ được cho 1 lượng máu khác nhau, màn càng khó lượng máu cho càng nhiều. Máu sẽ bị giảm đi khi người chơi bị trúng đạn, hết số máu người chơi sẽ chết và GameOver.



Hình 4.2.8 Màn Hình GameOver

* + 1. Tạm Dừng Game

Khi đang chơi game nếu người người muốn tạm dừng chỉ cần ấn nút p game sẽ được tạm dừng và hiện lên màn hình paused.



Hình 4.2.9 màn Hình paused

1. DỮ LIỆU GAME BẮN MÁY BAY
   * 1. Màn Hình Bắt Đầu Lập Trình

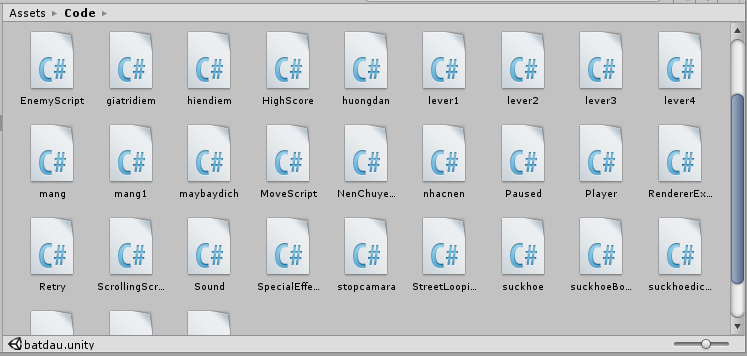
Game được hình thành bởi 10 Sence trên màn hình và rất nhiều những hoạt họa mà em đã tạo ra. Em sẽ giới thiệu các hoạt họa ở bên dưới.



Hình 4.3.1 Tổng thể các Sence để lập trình game

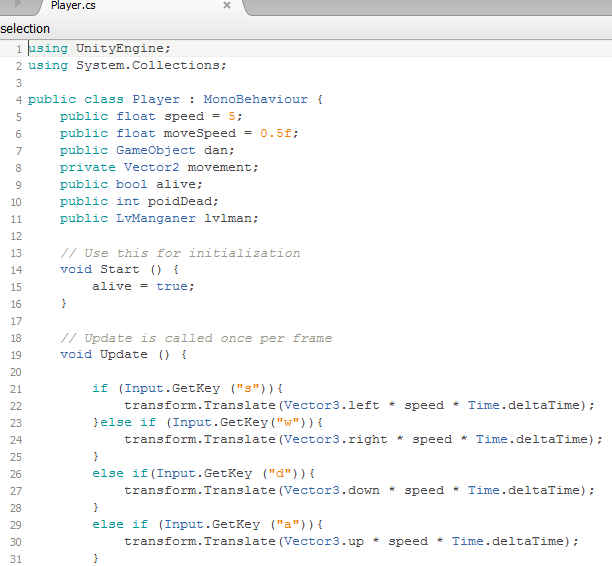
* + 1. Code Trong Game

Ngôn ngữ em chọn để lập trình game Bắn Máy Bay là ngôn ngữ C# trên MonoDeveloper được Unity hỗ trợ.



Hình 4.3.2 Code C# có trong game

1. MỘT SỐ CLASS QUAN TRỌNG TRONG GAME
   * 1. Player Game



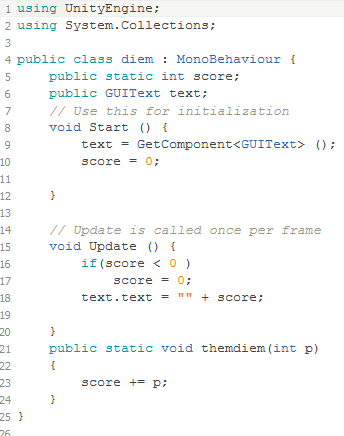
Hình 4.4.1 Class để người chơi di chuyển nhân vật

* + 1. Di chuyển Đối Tượng Trong Game



Hình 4.4.2 Di Chuyển các đối tượng có trong game

* + 1. Điểm



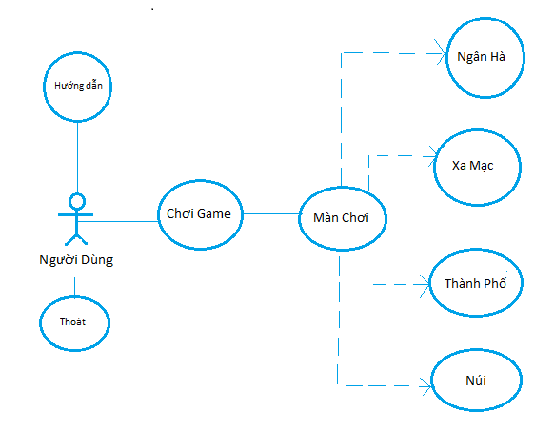
Hình 4.4.3 Cộng điểm khi ăn coin

* + 1. Đạn



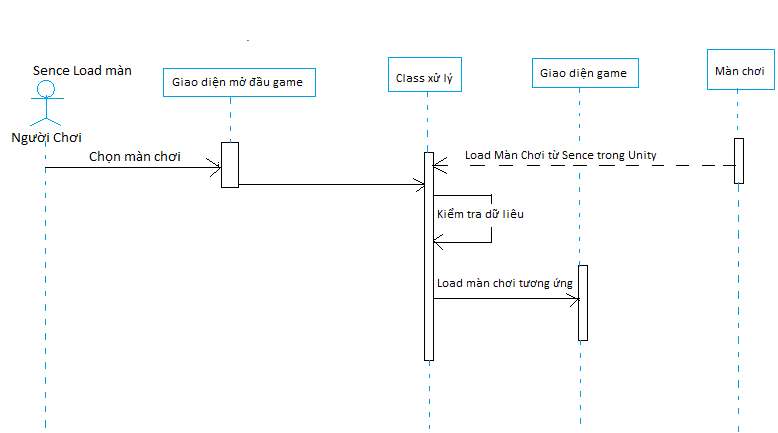
Hình 4.4.4 Nạp đạn cho đối tượng trong game

1. PHÂN TICH SƠ ĐỒ
   * 1. Sơ Đồ Use Case

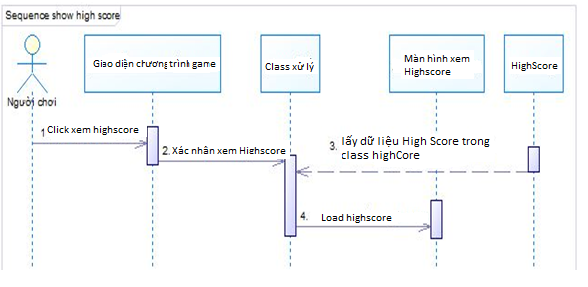


Hình 4.5.1 Sơ đồ use case game

* + 1. Sơ Đồ Sequence



Hình 4.5.2 Sơ đồ Sequence Load màn chơi



Hình 4.5.3 Sơ đồ Sequence xem high scores

1. **KẾT LUẬN**

Sau nhiều ngày suy nghĩ kĩ để chọn đề tài tốt nghiệp, cuối cùng em cũng đã chọn được đề tài làm game 2d trên Unity Game Bắn Máy Bay.

Bước đầu tìm hiểu về Unity và Game Engine em gặp rất nhiều khó khăn trong việc tìm tài liệu và xây dựng ý tưởng game vì tài liệu về unity đa phân bằng tiếng anh, nhưng sau khoảng 1 tháng cố gắng tìm tòi, học hỏi các bạn đi trước thì em cũng đã làm cho mình được 1 game gần hoàn chỉnh để phục vụ cho đề tài tốt nghiệp. Trong thời gian làm đề tài em cũng có một số những vấn đề đạt được và chưa đạt được.

* **Đạt Được**
* Hiểu được cấu trúc, cách thức hoạt động, cách thức phát triển ứng dụng trên Unity
* Nâng cao kiến thức lập trình và tiếng anh chuyên ngành thông qua những tài liệu về Unty bằng tiếng anh
* Xây dựng được các nhân vật trong game bằng phần mêm hỗ trợ photoshop
* Tổ chức load các màn tương ứng thông qua các button
* Xây dựng các chức năng như điểm, máu, màn chơi.. thông qua ngôn ngữ lập trình C# trên MonoDevelop
* Tạo âm thanh cho cho trò chơi và 1 số chức năng khác.

Ngoài những thứ đặt được thì em còn rất nhiều những vấn đề chưa đạt được.

* **Chưa Đạt Được**
  + Do kiến thức của em về Unity còn hạn hẹp nên các chức năng trong game còn nghèo nàn
  + Chưa xây dựng được hoàn chỉnh 1 game như mong muốn
  + Các Script trong Unity tổ chức chưa thực sự rõ ràng
  + Xắp xếp nhân vật và màn chơi chưa được logic
* **Thuận Lợi**
* Vận dụng được các kiến thức mà em đã học trong tường để làm đồ án này
* Được sự chỉ dạy tận tình của thầy và các bạn
* Tài liệu rất nhiều trên Internet
* **Khó Khăn**
* Unit là một môi trường lập trình mới đối với em nên khi bắt tay vào làm thì em không biết làm từ đâu và cảm thấy bỡ ngỡ
* **Kinh Nghiệm Rút Ra**
* Phải biết sắp xếp thời gian 1 cách hợp lý nếu không sẽ gặp rất nhiều khó khăn
* Để hoàn thành tốt 1 game không thể làm 1 mình mà phải biết chia sẻ và phân công công việc cho mọi người
* Làm việc phải tập trung mới có thể hoàn thành tốt được
* **Hướng Phát Triển**
* Tối ưu hóa để game có thể chạy nhanh hơn
* Tăng thêm màn chơi để người chơi có nhiều lựa chọn
* Thêm vũ khí, dạn dược, nâng cấp máy bay và vũ khí
* Cung cấp thêm công cụ để người chơi có thể chọn và chỉnh sửa tạo riêng cho mình những nhân vật, hiệu ứng âm thanh….
* Cải thiện trí thông minh của máy

**Danh mục tài liệu tham khảo**

[1] Ryan Henson Creighton, *Unity 3D Game Development by Example*, Packt Publishing, 2010.

[2] Will Goldstone, *Unity Game Development Essential*, Packt Publishing, 2009. [3] Efraim Meulenberg, *Game Development with Unity*, VTC, 2010.

[4] Efraim Meulenberg, *Game Development for iPhone/iPad Using Unity iPhone*, VTC, 2010.

[5] TornadoTwins**,** *Wormgame Cartoony Series*

<http://www.unityprefabs.com/wormgame-cartoony-series-tutorial.html>

[6] Zak Parrish, *Unity Fundamentals*

<http://www.3dbuzz.com/vbforum/content.php?176>

[7] Chad and Eric, *Unity Training*

<http://walkerboystudio.com/html/unity_training> free .htm