**ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

**KHOA CÔNG NGHỆ PHẦN MỀM**

**LÊ TRỌNG THIÊN**

**KHÓA LUẬN TỐT NGHIỆP**

**XÂY DỰNG GAME TRÊN NỀN TẢNG UNITY 3D**

**KỸ SƯ NGÀNH KỸ THUẬT PHẦN MỀM**

**TP. HỒ CHÍ MINH, 2015**

**ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

**KHOA CÔNG NGHỆ PHẦN MỀM**

**LÊ TRỌNG THIÊN**

**KHÓA LUẬN TỐT NGHIỆP**

**XÂY DỰNG GAME TRÊN NỀN TẢNG UNITY 3D**

**KỸ SƯ NGÀNH KỸ THUẬT PHẦN MỀM**

**GIẢNG VIÊN HƯỚNG DẪN**

**NGUYỄN PHƯƠNG ANH**

**TP. HỒ CHÍ MINH, 2015**

DANH SÁCH HỘI ĐỒNG BẢO VỆ KHÓA LUẬN

Hội đồng chấm khóa luận tốt nghiệp, thành lập theo Quyết định số …………………… ngày ………………….. của Hiệu trưởng Trường Đại học Công nghệ Thông tin.

* 1. …………………………………………. – Chủ tịch.
  2. …………………………………………. – Thư ký.
  3. …………………………………………. – Ủy viên.
  4. …………………………………………. – Ủy viên.

LỜI CẢM ƠN

Trước hết em xin gởi lời cảm ơn chân thành đến thầy Nguyễn Phương Anh, khoa Công Nghệ Phần Mềm, Trường Đại học Công Nghệ Thông Tin. Trong suốt thời gian thực hiện khóa luận, thầy đã dành rất nhiều thời gian và tâm huyết trong việc hướng dẫn em. Thầy đã luôn có những định hướng, góp ý, sửa chữa những chỗ sai sót giúp em có thể đi được đúng hướng. Khoá luận có được kết quả ngày hôm nay là nhờ sự nhắc nhở, đôn đốc và giúp đỡ nhiệt tình của thầy.

Em cũng xin trân trọng cảm ơn tất cả thầy cô trong khoa Công Nghệ Phần Mềm, cũng như các thầy cô đã giảng dạy, giúp đỡ em trong suốt thời gian học tập ở trường. Những kiến thức nền tảng và chuyên môn mà em học được từ các thầy cô đã giúp em có thể hoàn thành khóa luận này.

Xin cảm ơn anh Phong đã tạo điều kiện, giảm thiểu công việc ở công ty để em có nhiều thời gian hơn làm khóa luận. Cảm ơn các anh em trong công ty và bạn bè đã chơi thử game và cho em những lời khuyên hữu ích.

Con xin cảm ơn ba mẹ đã nuôi dạy con nên người, và đã chăm sóc con hết sức chu đáo trong khoảng thời gian hơn một tuần về quê để tập trung làm khóa luận.

Mặc dù đã rất cố gắng để hoàn thành nhưng vẫn không thể tránh khỏi những thiếu sót. Em kính mong nhận được thêm sự góp ý của quý thầy cô.

Sinh viên thực hiện khóa luận

Lê Trọng Thiên

MỤC LỤC

[Chương 1. TỔNG QUAN 6](#_Toc438721854)

[1.1. Khảo sát các sản phẩm tương tự 6](#_Toc438721855)

[1.1.1. Fun Run – Multiplayer Race 6](#_Toc438721856)

[1.1.2. NinJump Dash: Multiplayer Race 6](#_Toc438721857)

[1.2. Bối cảnh nghiên cứu 7](#_Toc438721858)

[1.3. Động lực nghiên cứu 7](#_Toc438721859)

[1.4. Giới hạn đề tài 8](#_Toc438721860)

[Chương 2. CƠ SỞ CÔNG NGHỆ 9](#_Toc438721861)

[2.1. Game engine 9](#_Toc438721862)

[2.1.1. Các loại game engine 9](#_Toc438721863)

[2.1.2. Một số game engine trên thị trường 10](#_Toc438721864)

[2.2. Unity 12](#_Toc438721865)

[2.2.1. Giới thiệu 12](#_Toc438721866)

[2.2.2. Các tính năng chính 13](#_Toc438721867)

[2.2.3. Ngôn ngữ lập trình 16](#_Toc438721868)

[2.2.4. Nền tảng Mono 18](#_Toc438721869)

[2.2.4.1. Kiến trúc 18](#_Toc438721870)

[2.2.4.2. Các thành phần 19](#_Toc438721871)

[2.2.4.3. Các tính năng 19](#_Toc438721872)

[2.2.5. Kiến trúc Component 20](#_Toc438721873)

[2.2.6. Các Component trong Unity 22](#_Toc438721874)

[2.2.7. Script 28](#_Toc438721875)

[2.2.8. Physics 31](#_Toc438721876)

[2.2.8.1. Các Component vật lý trong Unity 33](#_Toc438721877)

[2.3. Photon 37](#_Toc438721878)

[2.3.1. Giới thiệu về SaaS 37](#_Toc438721879)

[2.3.1.1. SaaS là gì? 37](#_Toc438721880)

[2.3.1.2. So sánh SaaS và on Premise 38](#_Toc438721881)

[2.3.1.3. Hạn chế của SaaS 39](#_Toc438721882)

[2.3.2. Photon Cloud và Photon Server 40](#_Toc438721883)

[2.3.3. Các ứng dụng chạy trên Photon Cloud 42](#_Toc438721884)

[2.3.4. So sánh PUN, PUN + và Unity Networking 43](#_Toc438721885)

[2.3.5. So sánh Photon Cloud với các Service khác 43](#_Toc438721886)

[2.4. Parse 44](#_Toc438721887)

[2.4.1. Giới thiệu 44](#_Toc438721888)

[2.4.2. Kiến trúc 46](#_Toc438721889)

[2.4.3. NoSQL 46](#_Toc438721890)

[2.4.4. MongoDB 47](#_Toc438721891)

[2.4.4.1. Các khái niệm 48](#_Toc438721892)

[2.4.4.2. Tính năng chính 49](#_Toc438721893)

[2.4.4.3. Sử dụng 50](#_Toc438721894)

[Chương 3. PHÂN TÍCH THIẾT KẾ 52](#_Toc438721895)

[3.1. Tổng quan chức năng 52](#_Toc438721896)

[3.2. Kiến trúc tổng quát 53](#_Toc438721897)

[3.3. Game Race of Dragons 53](#_Toc438721898)

[3.3.1. Yêu cầu 53](#_Toc438721899)

[3.3.1.1. Tổng quan 53](#_Toc438721900)

[3.3.1.2. Lý do thành công của dự án 54](#_Toc438721901)

[3.3.1.3. Cốt truyện và gameplay 54](#_Toc438721902)

[3.3.2. Phân tích 55](#_Toc438721903)

[3.3.3. Cấu trúc dữ liệu 55](#_Toc438721904)

[3.3.4. Sơ đồ use case 56](#_Toc438721905)

[3.3.5. Sơ đồ tuần tự 57](#_Toc438721906)

[3.3.6. Giải thuật 59](#_Toc438721907)

[3.3.7. Danh sách các lỗi 62](#_Toc438721908)

[3.4. Map Editor 64](#_Toc438721909)

[3.4.1. Sơ đồ use case 65](#_Toc438721910)

[3.4.2. Sơ đồ tuần tự 66](#_Toc438721911)

[3.4.3. Thiết kế giao diện 68](#_Toc438721912)

[3.4.4. Cấu trúc dữ liệu 70](#_Toc438721913)

[3.5. CMS 72](#_Toc438721914)

[3.5.1. Sơ đồ use case 73](#_Toc438721915)

[3.5.2. Sơ đồ tuần tự 74](#_Toc438721916)

[3.5.3. Thiết kế cơ sở dữ liệu 75](#_Toc438721917)

[3.5.4. Thiết kế giao diện 77](#_Toc438721918)

[Chương 4. CÀI ĐẶT MINH HOẠ 83](#_Toc438721919)

[4.1. Traffic Game 83](#_Toc438721920)

[4.1.1. Giao diện 83](#_Toc438721921)

[4.1.2. Hướng dẫn sử dụng 84](#_Toc438721922)

[4.2. Map Editor 86](#_Toc438721923)

[4.2.1. Giao diện 86](#_Toc438721924)

[4.2.2. Hướng dẫn sử dụng 86](#_Toc438721925)

[4.3. CMS 87](#_Toc438721926)

[4.3.1. Yêu cầu phần cứng 87](#_Toc438721927)

[4.3.2. Giao diện 87](#_Toc438721928)

[Chương 5. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 90](#_Toc438721929)

[5.1. Kết luận 90](#_Toc438721930)

[5.2. Hướng phát triển 90](#_Toc438721931)

[TÀI LIỆU THAM KHẢO 91](#_Toc438721932)

DANH MỤC HÌNH VẼ

Hình 2.1 Kiến trúc Mono 17

Hình 2.2 Thực thể theo kiến trúc component 19

Hình 2.3 Các hệ thống xử lý các component 20

Hình 2.4 Các loại Physic Material mặc định trong Unity 32

Hình 2.5 Các thuộc tính chung của các Physic Material 32

Hình 2.6 Tỷ lệ sử dụng các framework PHP năm 2013 35

Hình 2.7 Vòng đời một Request trong Laravel 37

Hình 2.8 Request Filter trong Laravel 40

Hình 3.1 Kiến trúc tổng quát hệ thống 47

Hình 3.2 Làn đường (bên trái) và đường (bên phải) 49

Hình 3.3 Biển báo giao thông là nơi chứa thông tin ràng buộc 51

Hình 3.4 Làn đường là nơi chứa thông tin ràng buộc 51

Hình 3.5 Vấn đề kiểm tra lỗi người chơi 52

Hình 3.6 Sơ đồ use case của Traffic Game 54

Hình 3.7 Sơ đồ tuần tự use case Login 55

Hình 3.8 Sơ đồ tuần tự use case Logout 56

Hình 3.9 Sơ đồ tuần tự use case Show history 56

Hình 3.10 Sơ đồ tuần tự use case Select map 56

Hình 3.11 Sơ đồ tuần tự use case Post score 57

Hình 3.12 Vượt đèn đỏ 57

Hình 3.13 Sơ đồ logic phát hiện lỗi vượt đèn đỏ 58

Hình 3.14 Lấn tuyến 58

Hình 3.15 Sơ đồ logic phát hiện lỗi lấn tuyến 59

Hình 3.16 Sơ đồ logic lỗi chuyển hướng không tín hiệu 59

Hình 3.17 Sơ đồ use case Map Editor 62

Hình 3.18 Sơ đồ tuần tự use case New map 63

Hình 3.19 Sơ đồ tuần tự use case Save map 64

Hình 3.20 Sơ đồ tuần tự use case Open map 64

Hình 3.21 Giao diện chính Map Editor 65

Hình 3.22 Giao diện New map và Edit map 67

Hình 3.23 Sơ đồ use case CMS 70

Hình 3.24 Sơ đồ tuần tự use case Upload accounts 71

Hình 3.25 Sơ đồ tuần tự use case Update accounts 72

Hình 3.26 Giao diện chung các trang của CMS 75

Hình 3.27 Giao diện Đăng nhập 76

Hình 3.28 Giao diện Update accounts 76

Hình 3.29 Giao diện Update maps 77

Hình 3.30 Giao diện Scores 78

Hình 4.1 Giao diện Traffic Game 80

Hình 4.2 Các thành phần giao diện Traffic Game 81

Hình 4.3 Giao diện Map Editor 83

Hình 4.4 Giao diện Home / Scores trong CMS 84

Hình 4.5 Giao diện Update accounts trong CMS 85

Hình 4.6 Giao diện Upload map trong CMS 85

Hình 4.7 Giao diện Update maps trong CMS 86

DANH MỤC BẢNG

Bảng 1.1 Miêu tả ứng dụng Học luật giao thông 3

Bảng 1.2 Miêu tả game City Car Driving 4

Bảng 2.1 Tỷ lệ sử dụng các ngôn ngữ trong Unity 15

Bảng 2.2 Các thuộc tính của MonoBehaviour 27

Bảng 2.3 Các phương thức của MonoBehaviour 27

Bảng 2.4 Các phương thức cần thừa kế của MonoBehaviour 29

Bảng 2.5 So sánh tốc độ xử lý vật lý của CPU và GPU 30

Bảng 2.6 Các thuộc tính của Rigidbody 31

Bảng 2.7 Các loại Collider 34

Bảng 2.8 So sánh các thuật ngữ trong RDBMS và MongoDB 43

Bảng 3.1 Danh sách Actor trong sơ đồ use case của Traffic Game 54

Bảng 3.2 Danh sách use case trong Traffic Game 55

Bảng 3.3 Danh sách các lỗi vi phạm 61

Bảng 3.4 Danh sách Actor trong Map Editor 62

Bảng 3.5 Danh sách use case của Map Editor 63

Bảng 3.6 Chú giải giao diện chính Map Editor 66

Bảng 3.7 Chú giải giao diện New map 67

Bảng 3.8 Danh sách Actor trong CMS 70

Bảng 3.9 Danh sách use case CMS 71

Bảng 3.10 Các thuộc tính mặc định trong MongoDB 72

Bảng 3.11 Collection accounts 73

Bảng 3.12 Collection maps 73

Bảng 3.13 Collection scores 74

Bảng 3.14 Các giao diện trong CMS 74

Bảng 3.15 Chú giải giao diện chung các trang CMS 75

Bảng 3.16 Chú giải giao diện Đăng nhập 76

Bảng 3.17 Chú giải giao diện Update accounts 77

Bảng 3.18 Chú giải giao diện Update maps 78

Bảng 3.19 Chú giải giao diện Scores 79

Bảng 4.1 Các thành phần giao diện Traffic Game 81

Bảng 4.2 Các phím điều khiển trong Traffic Game 82

DANH MỤC TỪ VIẾT TẮT

|  |  |
| --- | --- |
| **Từ viết tắt** | **Diễn giải** |
| API | Application Programming Interface |
| CMS | Content Management System |
| GUI | Graphical User Interface |
| MVC | Model-View-Controller |
| CSDL | Cơ sở dữ liệu |
| RDBMS | Relational Database Management System |
| SQL | Structured Query Language |
| NoSQL | Not Only Structured Query Language |
| 2D, 3D | 2-Dimensional, 3-Dimensional |
| PC | Personal Computer |
| ORM | Object-Relational Mapping |
| T-SQL | Transact-Structured Query Language |

TÓM TẮT KHÓA LUẬN

Với sự phát triển mạnh mẽ của đất nước trong những năm gần đây, đời sống con người càng ngày càng được cải thiện, nhu cầu giải trí cũng từ đó tăng lên. Smart phone trở thành một món đồ phổ biến, hầu như ai cũng có. Nhưng cũng do đó, con người dần trở nên rời xa nhau. Các bạn trẻ thì bị chứng nghiện mạng xã hội, ngay cả khi gặp mặt bạn bè cũng lấy điện thoại check in, chơi game. Ý tưởng làm một game mà ở đó, bạn bè, người thân có thể chơi cùng với nhau cũng hình thành từ thực trạng này.

Từ kết quả nghiên cứu, tìm hiểu về các game engine, em đã quyết định sử dụng Unity 3D để xây dựng nên game Race of Dragons - là sự tổng hợp và pha trộn giữa các thể loại action, platform, racing và multiplayer.

Trước hết, quá trình nghiên cứu Unity bao gồm tìm hiểu về các vấn đề cơ bản: game engine là gì, so sánh các game engine đang phổ biến, và lý do lựa chọn Unity. Tiếp theo là các vấn đề chuyên sâu của engine này, bao gồm: công nghệ Mono, kiến trúc Component, script, physic system và terrain.

Cuối cùng là việc áp dụng Unity vào việc xây dựng game. Bên cạnh việc sử dụng Unity, để có thể hoàn thành các chức năng multiplayer của game, em đã tìm hiểu thêm về Photon và Parse để xây dựng hệ thống mutiplayer real-time, hệ thống quản lý dữ liệu người chơi.

Kết quả cuối cùng là một game tương đối hoàn chỉnh về nội dung, gameplay và các chức năng cơ bản như quản lý tài khoản, đồng bộ dữ liệu, kết bạn, tìm trận ngẫu nhiên, mời bạn cùng chơi…

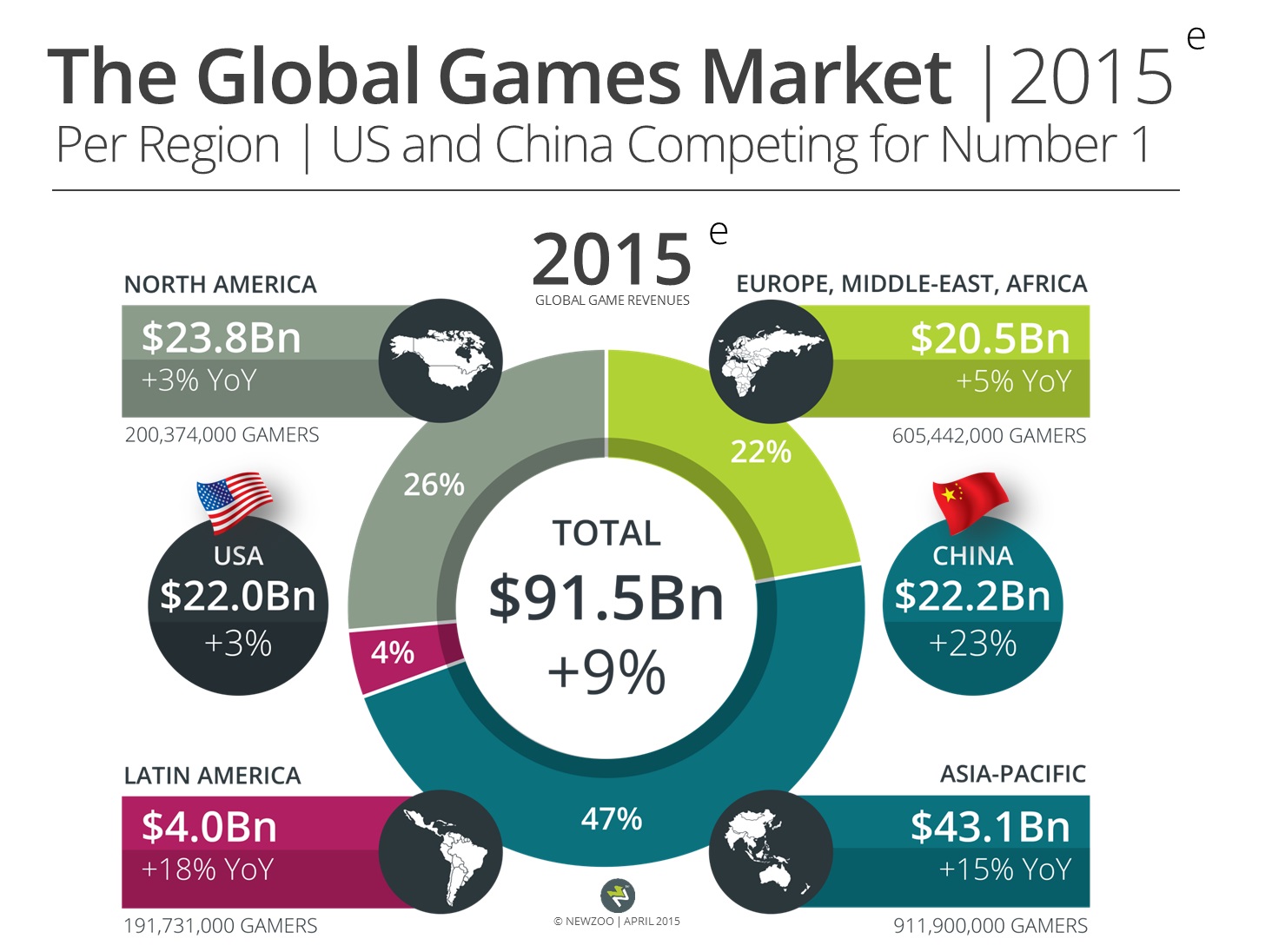
MỞ ĐẦU

* Tình hình ngành công nghiệp game trên thế giới

Công ty nghiên cứu thị trường trò chơi giải trí Newzoo dự báo tổng doanh thu ngành công nghiệp game toàn cầu trong năm 2015 sẽ sớm cán mốc 91,5 tỷ USD, tương ứng tỷ lệ tăng trưởng hằng năm 9,6%.

Cũng theo dự báo về thị trường game toàn cầu trong tương lai của Newzoo, doanh thu sẽ tiếp tục tăng lên đến con số 107 tỷ USD vào năm 2017. Khi thị trường tổng thể bắt đầu lớn mạnh, doanh thu tại những thị trường cụ thể cũng sẽ thay đổi theo.

Newzoo nhận định rằng, trong năm nay 2015, sự thay đổi về doanh thu đáng kể là tại thị trường game Trung Quốc, khi mà thị trường tại quốc gia này gần như đã vượt mặt thị trường Mỹ, mang lại 22,2 tỷ USD (tăng 23%) so với 22 tỷ USD tại thị trường Mỹ (tăng 3%).



Song song đó, châu Á - Thái Bình Dương sẽ là khu vực có doanh thu cao nhất trong nền công nghiệp game, chiếm 43,1 tỷ USD (tăng 15%), trong khi đó khu vực châu Mỹ La-tinh sẽ có doanh thu thấp nhất trong 4 thị trường trọng điểm, nhưng đồng thời cũng là khu vực có mức tăng trưởng nhanh nhất, tăng 18% so với năm liền kề trước đó.

Các nền tảng (hệ máy) mà người chơi sử dụng cũng sẽ thay đổi liên tục. Các loại máy tính bảng dự kiến sẽ tăng thêm 27% so với năm 2014, và chạm mốc 9,7 tỷ USD.

Một số thiết bị di động khác như smartphone, smartwatch sẽ có doanh thu tăng 21% lên đến 20,6 tỷ USD. Tuy nhiên, máy tính cá nhân (PC) vẫn là nền tảng phổ biến nhất cho các tựa game, mang lại 27,1 tỷ USD (tăng 8%) từ các tựa game tiêu chuẩn hoặc game thuộc thể loại MMO, trong khi các webgame bình thường sẽ còn đem đến 6,6 tỷ USD (tăng 2%).

Được biết, Newzoo phân loại TV, máy console và thiết bị thực tế ảo vào cùng một hạng mục, đồng thời dự đoán các thiết bị công nghệ này sẽ đem lại doanh thu 25,1 tỷ USD cho ngành công nghiệp game trong năm 2015.

Một phần thị trường chưa thấy sự tăng trưởng đáng kể tại thời điểm hiện nay chính là của các thiết bị chơi game cầm tay chuyên dụng, theo Newzoo. Ở nhóm sản phẩm này, Newzoo dự báo doanh thu trong năm 2015 là 2,7 tỷ USD, tương ứng mức giảm 16%.

* Ngành công nghiệp game ở Việt Nam

Chưa bao giờ ngành game mobile Việt Nam có tốc độ tăng trưởng mạnh như hiện nay. Năm 2014, doanh thu của thị trường game Việt Nam ước đạt khoảng 6.000 tỷ đồng, bằng với năm 2013, trong đó game mobile chiếm khoảng 1.000 tỷ đồng. Tiềm năng của game mobile sẽ tạo nên diện mạo mới của ngành công nghiệp game Việt Nam.

Do chi phí để sản xuất game cài đặt trên PC khá cao nên các nhà phát triển game trong nước đang hướng đến sản xuất game chạy trên nền tảng iOS hoặc Android của smartphone. Cả nước hiện có khoảng 40 doanh nghiệp, studio tham gia sản xuất và phát hành game.

Các studio game của Việt Nam tập trung chủ yếu TP.HCM, Hà Nội, Đà Nẵng, có thể kế đến như: Colobox, Pine, Canvas Games Studio, Banana Studio..., có những studio game chỉ làm outsoursing (thuê ngoài) cho nhà sản xuất ngoại nhưng một số lại kiêm luôn vai trò sản xuất.

Nhìn vào bức tranh chung của ngành công nghiệp game thế giới, một nghiên cứu của Newzoo cho thấy, thị phần game mobile toàn cầu đang có xu hướng gia tăng. Cụ thể, trong năm 2013, game mobile chiếm 17,4% doanh thu (tương ứng 12,3 tỷ USD) và năm 2016, tỷ lệ này tăng 27,8% (khoảng 23 tỷ USD).

Riêng tại Việt Nam, từ thị trường không có gì, game chạy trên nền tảng iOS và Android của thiết bị smartphone ước tính đạt doanh thu 2 triệu USD/tháng trong năm 2012; 25,5 triệu USD vào năm 2013, nay dự báo đạt 35 triệu USD và đến năm 2016 là 52 triệu USD.

Năm 2014, doanh thu của thị trường game Việt Nam ước đạt khoảng 6.000 tỷ đồng, bằng với năm 2013, trong đó game mobile chiếm khoảng 1.000 tỷ đồng.

Gần đây, ngành công nghiệp game Việt Nam nổi lên với sự kiện Flappy Bird của Nguyễn Hà Đông (từng đứng đầu AppStore); đồng thời, Việt Nam cũng có một số game studio tạo ra các sản phẩm mobile game khá thành công như trường hợp DivMob, một trong những nhà phát triển game Việt đã viết hơn 10 game trên nền Android và iOS, thu hút hơn 20 triệu lượt tải về trên toàn cầu năm 2013.

Trước tình hình ngành công nghiệp game trên thế giới và tại Việt Nam đang dần biến đổi, cùng với lập trình game chính là đam mê từ nhỏ của em, em xin chọn đề tài “Xây dựng game trên nền tảng Unity 3D” để có thể tạo ra một sản phẩm game chất lượng.

# TỔNG QUAN

## Khảo sát các sản phẩm tương tự

### http://2.bp.blogspot.com/-5LdjcM6TvSs/UQIFSGDlJLI/AAAAAAAAAcY/DY2Xi1ucZxQ/s1600/fun%2Brun-icon.jpgFun Run – Multiplayer Race

* Nhà phát triển: Dirtybit
* Thể loại: Platform, Runner, Multiplayer, 2D
* Mô tả: Fun Run là một game online real-time multiplayer cho smart phone và tablets cho Android và iOS. Người chơi có thể là bất cứ ai trên thế giới. Cùng chơi với bạn bè hoặc chơi ngẫu nhiên với các người chơi khác!
* Rating: hơn 450.000 rating trên google play trong đó có hơn 300.000 rating 5\*
* Download: > 50M
* Một số hình ảnh trong game



### NinJump Dash: Multiplayer Race



* Nhà phát triển: Backflip Studio
* Thể loại: Platform, Runner, Multiplayer, 2D
* Mô tả: Ninjump Dash là một game live, multiplayer racing với các nhân vật từ game trong series game NinJump. Người chơi sẽ tham gia vào cuộc đua với 3 người còn lại, có thể là bạn bè hoặc ngẫu nhiên. Nhảy qua các vật cản, thu thập những vật phẩm đặc biệt, sử dụng chúng đánh hạ đối thủ để về đích sớm nhất và dành chiến thắng. Người chơi có thể kiếm tiền để mở khóa các nhân vật mới.
* Rating: 150.000 rating trên google play với hơn 95.000 rating 5\*
* Download: >5.000.000
* Một số hình ảnh trong game:



## Bối cảnh nghiên cứu

* Ngành công nghiệp game trên thế giới vẫn đang phát triển với tốc độc rất nhanh
* Thị trường game di động lên ngôi
* Game online nhiều người cùng chơi, tương tác với nhau đang thống trị dòng game mobile

## Động lực nghiên cứu

* Niềm đam mê với game đặc biệt là game online từ khi còn nhỏ
* Thử thách bản thân với một chủ đề khó và nặng đó là chủ đề game nhiều người chơi, real-time
* Nghiên cứu, tìm hiểu thêm các công nghệ mới, tích lũy thêm kiến thức cho bản thân
* Tạo ra một sản phẩm tốt, một game thuần việt mới đủ sức cạnh tranh với các game khác trên store

## Giới hạn đề tài

Nghiên cứu tập trung vào kiến thức game 2D và môi trường phát triển của Unity. Bên cạnh đó nghiên cứu cũng bao hàm các kiến thức về Photon Network và Parse.

Mục tiêu nghiên cứu nhằm xây dựng game 2D online real-time, sử dụng Photon Cloud và Parse.

Mục tiêu cụ thể bao gồm:

* Nghiên cứu các kiến thức 2D và môi trường phát triển Unity.
* Nghiên cứu Photon Network
* Nghiên cứu Parse
* Xây dụng game Race of Dragons với các tính năng sau:
  + Đăng ký
  + Đăng nhập
  + Đồng bộ dữ liệu người chơi từ server
  + Chơi game với 3 mode: Practice, Quick Race và Invite friend
  + Hỗ trợ 4 người chơi online real time trong cùng 1 match
  + Kết bạn và mời bạn bè cùng chơi

# CƠ SỞ CÔNG NGHỆ

## Game engine

Game engine là một bộ ứng dụng được xây dựng để phát triển game, hỗ trợ các tính năng để quá trình phát triển game trở nên đơn giản, dễ dàng và nhanh chóng. Các tính năng được hỗ trợ bởi các game engine thường bao gồm:

* Dựng hình 2D, 3D
* Chỉnh sửa mã nguồn
* Âm thanh
* Vật lý
* Tạo hiệu ứng động
* Quản lý bộ nhớ
* Xây dựng trí tuệ nhân tạo
* Kết nối mạng
* ...

Game engine giúp tiết kiệm thời gian và chi phí cho các dự án rất đáng kể.

### Các loại game engine

Việc phân loại game engine rất phức tạp vì có rất nhiều tiêu chí để so sánh giữa chúng. Tuy nhiên, dựa vào mức độ chuyên biệt, mức độ hỗ trợ của các engine mà ta chia chúng thành 3 cấp độ khác nhau, từ thấp lên cao: Roll-Your-Own, Most-Ready và Point-And-Click.

* Roll-Your-Own

Roll-Your-Own engine được xem như những engine ở mức thấp nhất. Thường thì chúng không phổ biến do được phát triển ở mỗi công ty với những mục đích riêng. Những engine này sử dụng những ứng dụng giao diện được công bố rộng rãi, các API như XNA, DirectX, OpenGL, SDL để phát triển phần đồ hoạ. Thêm vào đó, chúng có thể dùng những thư viện từ nhiều nguồn khác nhau để bổ sung các tính năng khác. Những thư viện đó có thể bao gồm thư viện vật lý như Box2D, Chipmunk, thư viện âm thanh như irrKlang, PortAudio và thư viện GUI như AntTweakBar.

Thực tế, Roll-Your-Own engine cho phép những lập trình viên có thể tương tác sâu hơn vào hệ thống, tuỳ biến theo mong muốn một cách dễ dàng. Tuy nhiên, họ cũng sẽ gặp phải rất nhiều lỗi do sự thiếu tương thích giữa các thành phần với nhau.

* Most-Ready

Most-Ready engine là những engine tầng trung. Chúng được thiết kế với đầy đủ các tính năng cần thiết, kèm theo đó là rất nhiều công cụ thoả mãn hầu hết các mục đích sử dụng của người dùng. Những engine loại này khá phong phú, bao gồm OGRE, Genesis3D, Torge, Gamebryo và cả Unreal, IdTech.

Những lập trình viên cần lập trình thêm để gắn kết các thành phần trong engine lại với nhau để tạo nên một game hoàn chỉnh. Vì tính chuyên biệt cao hơn, nên với loại engine này các lập trình viên sẽ không còn nhiều khả năng tương tác sâu như các engine loại Roll-Your-Own.

* Point-And-Click

Point-And-Click là loại game engine dễ sử dụng nhất. Nói như vậy không có nghĩa là mọi người đều có thể sử dụng chúng để làm ra game, vẫn cần một ít kiến thức lập trình để có thể làm được, tuy nhiên, lượng công việc đó đã được tối giản xuống mức tối thiểu. Các Point-And-Click engine có chứa hầu như tất cả những thứ cần thiết để làm game, người dùng chỉ cần lựa chọn và kéo thả. Các engine loại này bao gồm GameMaker, Torque Game Builder và Unity.

### Một số game engine trên thị trường

* CryEngine

CryEngine ban đầu được Crytek phát triển như một bản demo công nghệ cho Nvidia. Khi cảm thấy tiềm năng của CryEngine, Crytek đã phát triển nó và tung ra game đầu tiên Far Cry khá thành công. Qua thời gian CryEngine ngày càng phát triển mạnh mẽ hơn với những tựa game nổi tiếng như Far Cry, Crysis, Crysis Warhead, Crysis 2 và Aion: Tower of Eternity. Phiên bản hiện tại, CryEngine 3 được phát hành vào ngày 4/10/2009. Theo Crytek, CryEngine 3 được tạo ra với tham vọng trở thành công cụ phát triển game tất cả trong một, ứng dụng trên cả Windows, PlayStation 3 và Xbox 360. Riêng đối với nền PC, CryEngine 3 sẽ hỗ trợ tốt cả DirectX 9, 10 và 11.

Điểm nổi trội của CryEngine so với các engine đương thời là tập trung vào mặt xử lý hình ảnh, ngoài ra hiệu ứng âm thanh và chuyển động cũng được mô phỏng xuất sắc.

* OGRE

OGRE (Object-Oriented Graphics Rendering Engine) là một engine dựng hình 3D linh hoạt, tập trung vào dựng hình hơn là một công cụ tạo game. Bộ thư viện của OGRE trừu tượng hóa các thư viện ở mức hệ thống như DirectX 3D và OpenGL để cung cấp một giao diện lập trình dựa trên các đối tượng thế giới thực và các lớp cấp cao.

Tuy chỉ tập trung vào dựng hình và không cung cấp các thành phần hỗ trợ âm thanh, vật lý, tuy nhiên đó lại được xem là một ưu điểm của OGRE khi nó cho phép các lập trình viên có thể tuỳ ý lựa chọn các bộ thư viện để tích hợp vào để đáp ứng yêu cầu của từng dự án.

* Panda3D

Panda3D là một game engine mã nguồn mở, một thư viện dựng hình 3D và phát triển game được xây dựng bằng ngôn ngữ Python và C++. Bản thân engine được viết bằng C++ và sử dụng một bộ sinh mã tự động để tạo thành các chức năng hoàn chỉnh cho engine thông qua giao diện của Python. Cách tiếp cận này giúp tận dụng được những ưu điểm của giao diện lập trình Python nhưng vẫn giữ lại được tốc độ của C++ trong nhân của engine. Panda3D thường được dùng trong các dự án ở trường học, hoặc các dự án thương mại cỡ lớn. Những game sử dụng Panda3D như: Toondown Online, Pirates of the Caribbean, A Vampyre Story, Signal Ops.

* Unreal Engine

Unreal Engine được phát triển bởi Epic Games, ra mắt vào năm 1998 với game Unreal. Unreal được xem là đối thủ trực tiếp của CryEngine trên thị trường engine làm game FPS. Được xây dựng dựa trên C++ nên Unreal có thể chạy tốt trên hầu hết mọi nền tảng như Windows, Linux, Android, iOS, Mac OS hay các máy chơi game Xbox, PS, Dream Cast.

## Unity

### Giới thiệu

Unity là một engine làm game 3D, dựng hình thời gian thực, thuộc loại Point-And-Click, được phát triển bởi Unity Technologies. Unity có thể chạy trên hệ điều hành Windows và Mas OS X. Sản phẩm tạo ra từ Unity có thể chạy trên hầu hết các nền tảng Windows, Mac, Linux , Wii, iOS, Android. Bên cạnh đó, Unity Engine có khả năng phát triển game nền web hỗ trợ hầu hết các trình duyệt chính.

Xét về mặt đồ hoạ, Unity không phải là engine mạnh nhất, CryEngine vẫn đang dẫn đầu với ưu thế đồ họa 3D cực kỳ chân thực. Tuy nhiên, để có được những trải nghiệm tốt như vậy, CryEngine hay Unreal đều cần những máy cấu hình từ khá đến cao, không giống như Unity, có thể chạy tốt trên những thiết bị cấu hình trung bình. Tuy đồ họa không thể so sánh nhưng chất lượng hình ảnh mà Unity mang lại cũng quá đủ để thỏa mãn những game thủ hiện nay. Hơn nữa, Unity còn có những ưu điểm mà không phải game engine nào cũng có:

* Đa nền tảng

Unity được phát triển dựa trên kết quả của Mono Project, một dự án mã nguồn mở đa nền tảng trên ngôn ngữ C#. Chính vì điều đó, Unity có thể chạy tốt trên Windows, Mac OS, Linux, PlayStation 3, Xbox 360, Wii U, iOS, Android, Windows Phone, Blackberry 10, và thậm chí cả web và Flash. Với gói phần mềm của Unity, các studio có thể phát triển các game và phát hành trên nhiều nền tảng một cách nhanh nhất và tiết kiệm nhất.

* Hỗ trợ 2D và 3D

Unreal Engine và CryEngine đều chỉ hỗ trợ làm game 3D trong khi Unity hỗ trợ cả game 2D. Đây là một trong những ưu điểm vượt trội của Unity cho phép các studio vừa và nhỏ có thể sử dụng để phát triển các game từ nhỏ đến lớn, tuỳ vào kinh phí của họ. Phiên bản mới ra mắt của Unity là 5.3 với nhiều cập nhật mới hỗ trợ cho game 2D tốt hơn nữa với nhiều chức năng mới.

* Dễ sử dụng

Đối tượng hướng đến của Unity là cả các lập trình viên không chuyên và các studio chuyên nghiệp, do vậy nó được thiết kế để dễ dàng sử dụng nhất. Unity cung cấp một hệ thống toàn diện các công cụ từ thiết kế, soạn thảo mã nguồn đến sửa lỗi. Các tính năng được ưa thích của Unity là cho phép các lập trình viên có thể xây dựng các phần plugin để gắn vào editor của Unity và tính năng xem trước, cho phép nhà phát triển vừa kiểm lỗi, vừa thiết kế.

Assets Store có rất nhiều thư viện, plugin (free hoặc có phí) hỗ trợ rất tốt cho lập trình viên.

Hơn nữa Unity là một trong những engine phổ biến nhất thế giới, do đó, các nhà phát triển có thể dễ dàng tìm kiếm sự giúp đỡ từ cộng đồng rất nhanh.

* Tính kinh tế cao

Những cá nhân và doanh nghiệp có doanh thu < 100.000$ / năm được sử dụng miễn phí Unity, và 1.500$ cho bản Pro, một con số khá khiêm tốn so với các Engine game khác.

### Các tính năng chính

Với mong muốn tạo ra công cụ hỗ trợ tốt nhất, Unity giúp người dùng có thể tập trung phát triển ý tưởng của mình hơn là quan tâm giải quyết các vấn đề kỹ thuật, các khó khăn công nghệ.

Các tính năng chính của Unity bao gồm:

* Dựng hình
* DirectX 11 được hỗ trợ làm cho hiệu suất dựng hình được nâng cao đáng kể so với các phiên bản trước
* Kỹ thuật chiếu sáng Pre-Pass và đa luồng cho ra ánh sáng chân thực nhất
* 100 kiểu shader sẵn có, từ đơn giản đến phức tạp phục vụ mọi mục đích của nhà phát triển
* Cho phép can thiệp xuống tầng thấp của quá trình dựng hình để tối ưu hoá
* Ánh sáng
* Tích hợp bộ công cụ Beast của Autodesk cho phép thực hiện kỹ thuật Lightmap một cách dễ dàng
* Kỹ thuật Lightmap kép được dùng cho các khung cảnh lớn giúp hạn chế số lượng lightmap được tạo ra
* Hard-shadow và soft-shadow tuỳ vào phần cứng cho trải nghiệm tốt nhất trên từng thiết bị
* Kỹ thuật đặc biệt
* Vô số hiệu ứng tiền xử lý toàn màn hình cho phép chất lượng hình ảnh tuyệt vời hơn
* Tăng tính chân thực với các hiệu ứng vẽ lên texture
* Chất lỏng với nhiều trạng thái và màu sắc
* Quản lý nhiều particle cùng lúc hiệu quả với Particle Editor
* Âm thanh
* Thao tác đưa âm thanh vào game một cách đơn giản, hiệu quả
* Hỗ trợ streaming với file kích thước lớn
* Cung cấp rất nhiều bộ lọc để tạo ra các hiệu ứng âm thanh khác nhau
* Tài nguyên
* Công nghệ Substance của Allegorithmic giúp giảm kích thước tài nguyên và tạo ra các hiệu ứng ấn tượng
* Địa hình
* Tạo địa hình đơn giản nhưng hiệu quả hoàn toàn bằng thao tác kéo thả
* Công nghệ Lightmap của Beast cho ánh sáng địa hình được chân thực
* Vật lý
* Tích hợp engine vật lý NVIDIA PhysX cho hiệu suất xử lý vật lý cực cao
* Thư viện Box2D được sử dụng cho game 2D giúp giảm chi phí tính toán
* Xây dựng game đua xe đơn giản hơn với Wheel Collider
* Trí tuệ nhân tạo
* Bộ thư viện tìm đường đi được tích hợp hỗ trợ phát triển các đối tượng tự động
* 2D and 3D
* Unity cho phép sử dụng model từ các phần mềm phổ biến như Maya, 3ds Max, Modo, Cinema 4D, Blender
* Xây dựng các màn chơi đơn giản chỉ với thao tác kéo thả
* Công cụ Profiler cho phép kiểm soát tài nguyên, tối ưu hoá hệ thống tốt hơn
* Hỗ trợ nhiều camera cho phép tích hợp các đối tượng 2D và 3D trong cùng một khung cảnh dễ dàng
* Hoạt hoạ

Từ phiên bản 4.0, Unity đã cung cấp thêm công cụ xây dựng hoạt hoạ. Các thao tác trực tiếp trên khung xương ở chế độ 2D hay 3D đều đơn giản và dễ dàng.

* Hiệu suất
* Giải pháp tiền xử lý Occlusion Culling độc quyền của Unity, phát triển cùng với Umbra Software đảm bảo chỉ những đối tượng được nhìn thấy mới cần phải vẽ
* LODGroups hỗ trợ thực hiện kỹ thuật Level-Of-Detail dễ dàng
* Cơ chế tự động tối ưu bằng cách gắn nhiều khối nhỏ thành một khối lớn
* Cơ chế tự động loại bỏ các tài nguyên không sử dụng
* Bộ tối ưu mã GLSL trên mobile giúp tăng tốc độ lên 2-3 lần
* Đa nền tảng
* Tất cả các nền tảng mà Unity hỗ trợ bao gồm: iOS, Android, Windows Phone 8, Blackberry 10, Windows, Windows Store Apps, Universal 10, Mac OS, Linux, Web Player, PS3, PS4, PlayStation Vista, PlayStation Mobile, Xbox One, Xbox 360 và Wii U
* Quá trình build đơn giản với một thao tác
* Giao diện người dùng

Bộ thư viện GUI mới của Unity hỗ trợ đầy đủ các loại điều khiển từ cơ bản đến phức tạp, hỗ trợ layout và anchor giúp giao diện phù hợp với nhiều kích thước màn hình.

### Ngôn ngữ lập trình

Với Unity, ta có thể sử dụng một hoặc kết hợp ba ngôn ngữ để phát triển là: C#, JavaScript và Boo.

Theo một khảo sát của trang <http://forum.unity3d.com> vào tháng 7/2013 thì ngôn ngữ được sử dụng nhiều nhất trong Unity là C#.

|  |  |
| --- | --- |
| Ngôn ngữ | Tỷ lệ |
| Boo only | 3.28% |
| C# only | 40.10% |
| JavaScript only | 27.92% |
| Boo & C# | 1.97% |
| Boo & JavaScript | 0.52% |
| JavaScript & C# | 25.29% |
| Boo, C# and JavaScript | 0.92% |

Bảng . Tỷ lệ sử dụng các ngôn ngữ trong Unity

Giữa 3 ngôn ngữ này, không có sự khác biệt lớn về tốc độ và hiệu năng, tuy nhiên ta có một vài sự khác biệt giữa chúng.

* C#

Ưu điểm:

* Lượng người sử dụng lớn.
* Tốc độ nhanh hơn trong một vài trường hợp.
* Ràng buộc nghiêm ngặt, có thể phát hiện lỗi ngay lúc build.
* .NET Framework hỗ trợ nhiều, có cộng đồng lớn ngoài cộng đồng Unity.

Nhược điểm: Chưa thấy

* JavaScript

Ưu điểm:

* Dễ học, dễ hiểu.
* Có lượng người sử dụng lớn.

Nhược điểm:

* Không nghiêm ngặt, chỉ có thể phát hiện lỗi lúc chạy.
* Boo

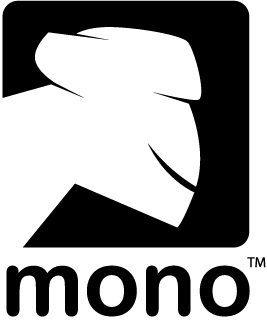
Ưu điểm:

* Đẹp, dễ đọc, dễ hiểu.
* Nghiêm ngặt, có thể phát hiện lỗi ngay lúc build.

Nhược điểm:

* Không có nhiều người sử dụng.

### Nền tảng Mono



Mono là một phiên bản mã nguồn mở đa nền tảng, hiện thực Microsoft .NET Framework dựa trên tiêu chuẩn ECMA Common Language Runtime. Được tài trợ bởi Novell và hiện tại là Xamarin, dự án Mono có một cộng đồng đóng góp tích cực và nhiệt tình. Mono bao gồm các công cụ phát triển và cơ sở hạ tầng để có thể chạy được các ứng dụng .NET ở client và server.

Unity được phát triển dựa trên phiên bản Mono 2.6. Hiện tại Unity chỉ hỗ trợ API của .NET 2.0.

#### Kiến trúc



Hình . Kiến trúc Mono

Ứng dụng được viết bằng ngôn ngữ C# hoặc các ngôn ngữ khác dựa trên các lớp thư viện tiện ích của Mono. Khi được thực thi, các lớp này sẽ gọi đến Common Language Runtime để thực thi các lệnh. Ở đây, tuỳ vào hệ điều hành mà các lệnh khác nhau sẽ được chuyển xuống từng hệ điều hành tương ứng. Do đó, ở khía cạnh nhà phát triển ứng dụng thì các lớp thư viện đều như nhau ở mọi hệ điều hành, họ không cần quan tâm đến chi tiết của mỗi nền tảng trong ứng dụng của mình.

#### Các thành phần

* C# Compiler

Mono C# Compiler hỗ trợ đầy đủ tính năng cho C# 1.0, 2.0, 3.0, 4.0 và 5.0.

* Mono Runtime

Mono Runtime hiện thực tiêu chuẩn ECMA Common Language Infrastructure (CLI). Máy ảo cung cấp một trình biên dịch Just-In-Time (JIT), một trình biên dịch Ahead-Of-Time (AOT), một library-loader, một garbage-collector, một threading-system và các thư viện chức năng. Các thư viện đó bao gồm:

* **Base Class Library**: Mono hiện thực tất cả các lớp cơ bản để có thể phát triển hoàn thiện một ứng dụng. Các lớp này hoàn toàn tương thích với các lớp của Microsoft .NET Framework.
* **Mono Class Library**: Mono còn cung cấp các lớp nâng cao, tiện dụng và đặc biệt là tập trung cho các ứng dụng trên nền tảng Linux, ví dụ như các lớp cho Gtk+, Zip file, LDAP, OpenGL, Cairo, POSIX, ...

#### Các tính năng

* Đa nền tảng

Mono hỗ trợ cho cả hệ thống 32bit và 64bit của tất cả các kiến trúc máy tính. Các hệ điều hành được hỗ trợ bao gồm: Linux, Mac OSX, iOS, Sun Solaris, OpenBSD, FreeBSD, NetBSD, Microsoft Windows, Nintendo Wii, Sony PlayStation 3

* Đa ngôn ngữ

Rất nhiều ngôn ngữ có thể được sử dụng với nền tảng Mono. Bất kỳ ngôn ngữ nào có thể biên dịch được sang ngôn ngữ IL (Intermediate Language) đều có thể chạy được trên nền tảng Mono. Danh sách các ngôn ngữ có thể chạy được, bao gồm: C#, F#, Scala, Java, Boo, Nemerle, Visual Basic .NET, JavaScript, Operon, PHP, Object Pascal, Lua, Cobra và nhiều ngôn ngữ khác.

* Tương thích với các API của Microsoft .NET Framework

Mono có thể chạy được các ứng dụng ASP.NET, ADO.NET, SilverLight và Windows.Form mà không cần phải biên dịch lại.

* Mã nguồn mở, ứng dụng miễn phí

Mono được phân phối dưới các giấy phép sau:

* C# Compiler xuất bản dưới 2 giấy phép MIT/X11 và GPL
* Các công cụ xuất bản dưới giấy phép GPL
* Các thư viện runtime xuất bản dưới giấy phép LGPL 2.0
* Các lớp thư viện xuất bản dưới giấy phép MIT X11
* Các thư viện liên quan với Microsoft được xuất bản dưới giấy phép Microsoft Permissive License, và một vài thư viện có giấy phép Apache2

### Kiến trúc Component

Kiến trúc Component là hướng lập trình hướng dữ liệu (Document-Oriented). Trong đó một đối tượng thường được gọi là entity (thực thể). Các thực thể là những đối tượng rỗng, không chứa bất kỳ dữ liệu nào riêng biệt và không có bất kỳ sự khác nhau nào giữa chúng. Một thực thể có thể gắn thêm nhiều component khác nhau để thực hiện các mục đích riêng. Mỗi thực thể sẽ mang những thuộc tính và hành vi của component được thêm vào, khi đó các thực thể mới thực sự khác nhau. Trong Unity, các thực thể này gọi là GameObject.



Hình . Thực thể theo kiến trúc component

Trong thực thể ở trên, có các component là: Position, Velocity, Sprite và Health.

Mỗi component thông thường sẽ có hai thành phần là:

* Attributes: dữ liệu riêng của component.
* Behaviours: các hàm xử lý các dữ liệu.

Mỗi component trong một thực thể sẽ hoàn toàn độc lập với nhau và được thực thi trên những hệ thống riêng biệt, chính vì thế, tốc độ thực thi của kiến trúc Component nhanh hơn kiến trúc cây thừa kế rất nhiều.



Hình . Các hệ thống xử lý các component

Trong kiến trúc Component hầu như không tạo ra một đối tượng nhất định giống như kiến trúc OOP mà tất cả các đối tượng trong component đều là thực thể. Không có kiến trúc theo cây thừa kế trong kiến trúc Component.

* Ưu điểm
* Tái sử dụng code tốt, giảm độ phức tạp của dự án. Một component có thể được sử dụng cho nhiều thực thể khác nhau có chung một số đặc điểm về component đó.
* Tốc độ xử lý cực nhanh vì mỗi component sẽ chạy trên một hệ thống riêng biệt nhau và không phải trả chi phí cho các hàm thừa kế trừu tượng như trong cấu trúc cây thừa kế.
* Dễ vận hành và bảo trì game: mỗi component gần như hoàn toàn độc lập với nhau nên rất dễ thích nghi với những thay đổi trong game, ít ảnh hưởng tới các component còn lại.
* Dễ dàng cho các dự án có nhiều người cùng tham gia khi mỗi người có thể quản lý mỗi component riêng biệt.
* Nhược điểm
* Không thích hợp với các dự án nhỏ.
* Thời gian phát triển ban đầu lớn.

### Các Component trong Unity

* Nhóm Mesh



Nhóm Mesh dùng để hiển thị các model 3D trong Unity.

Bao gồm các component:

* Mesh

Lớp dữ liệu, dùng để lưu các model 3D được import từ assets.

* Mesh Filter

Dùng để truyền một Mesh từ assets vào cho Mesh Renderer hiển thị.

* Text Mesh

Một Mesh đặc biệt dùng để hiển thị chữ trong không gian 3D.

* Mesh Render

Dùng để hiển thị Mesh và Text Mesh trong không gian 3D. Một GameObject phải được gắn momponent Mesh Renderer mới có thể hiển thị được model 3D trong không gian.

Các định dạng model 3D được Unity hỗ trợ gồm 2 loại:

* + Các định dạng đã kết xuất: .FBX, .OBJ
  + Các định dạng chưa kết xuất: .MAX, .BLEND nhận từ các phần mềm 3D Studio Max hoặc Blender
* Nhóm Effect



Tạo ra các hiệu ứng trong game như: khói, hơi nước, hiệu ứng khí quyển... Nhóm này bao gồm các component:

* Particle System

Dùng để hiện thực một hiệu ứng particle nào đó.

* Trail Renderer

Tạo hiệu ứng lưu vết phía sau cho các đối tượng di chuyển trong không gian.

* Line Renderer

Tạo ra các đường vẽ trong không gian, dựa vào các điểm được xác định. Đường được vẽ không phải là các điểm mà là các texture có độ dày nhất định.

* Lens Flare

Được dùng để thêm vào một nguồn sáng để tạo hiệu ứng cho nguồn sáng đó. Hiệu ứng được tạo ra giống như khi một camera quay trực tiếp vào một nguồn sáng mạnh.

* Halo

Tạo hiệu ứng vầng sáng xung quanh một đối tượng. Halo được dùng chủ yếu cho hiệu ứng của một điểm sáng (Point Light).

* Projector

Tạo ra hiệu ứng đổ bóng lên một mặt phẳng.

* Nhóm Renderring



Tạo các thành phần hiển thị trong không gian 3D, hoặc trong giao diện 2D. Nhóm này bao gồm các component:

* Camera

Dùng để quay lại tất cả các đối tượng có trong không gian đưa lên màn hình. Đây là component không thể thiếu trong game. Nếu không có đối tượng nào chứa component này, sẽ không có gì được thể hiện lên màn hình.

* Flare Layer

Được gắn vào Camera để tạo ra hiệu ứng Lens Flare.

* GUI Layer

Tương tự Flare Layer, cần được gắn vào Camera để thể hiện các GUI Text và GUI Texture.

* GUI Text

Dùng để hiển thị chữ lên màn hình 2D.

* GUI Texture

Dùng để hiển thị Texture lên màn hình 2D.

* Light

Tạo ra một nguồn sáng, chiếu sáng các thành phần hoặc toàn bộ các đối tượng có trong không gian. Có 4 loại Light:

* + Direction: nguồn sáng song song, chiếu từ một nơi rất xa, có tác dụng với tất cả các đối tượng trong không gian. Ví dụ: ánh sáng mặt trời.
  + Point: điểm sáng, chiếu từ 1 điểm ra tất cả các hướng. Ví dụ: bóng đèn tròn.
  + Spot: nguồn sáng hình chóp. Ví dụ: đèn pin.
  + Area: Chỉ có tác dụng khi tạo light mapping, và không có tác dụng lúc thực thi.
* Skybox

Dùng để giả lập một không gian vô hạn bao quanh toàn bộ không gian. Ví dụ: bầu trời.

* Nhóm Physics



Unity sử dụng NVIDIA PhysX để giả lập các tương tác vật lý trong game. Để tạo các chuyển động vật lý, va chạm, phản hồi, ... cần gắn một component physic vào đối tượng. Các component nhóm này bao gồm:

* Rigidbody

Cho phép các đối tượng có thể chịu tác động của lực, bao gồm cả trọng lực trái đất, lực cản không khí… Đối với các đối tượng tĩnh, không cần phản ứng với lực thì không cần tới component này.

* Character Controller

Được dùng để gắn vào các đối tượng mà người chơi điều khiển trong game như: nhân vật góc nhìn thứ nhất, góc nhìn thứ 3. Đối tượng này không chịu sự tác động của các lực vật lý để đảm bảo người chơi có thể di chuyển giống thực, nhưng nó có khả năng gây ra lực lên các đối tượng khác khi va chạm.

* Constant Force

Tạo ra một lực lập tức tác động liên tục lên đối tượng ở mọi khung hình thay vì chỉ một khung hình nếu sử dụng hàm *Rigidbody.AddForce*. Được dùng nhiều cho các loại tên lửa hay viên đạn bay.

* Collider

Cho phép các đối tượng có thể va chạm với nhau. Nếu không có một Collider nào được gắn vào đối tượng, thì chúng sẽ di chuyển xuyên qua nhau. Có các loại Collider sau:

* + Box Collider: Khối lập phương
  + Sphere Collider: Khối cầu
  + Capsule Collider: Khối trụ có 2 đầu tròn.
  + Mesh Collider: Khối được tạo nên từ một Mesh.
  + Wheel Collider: Một dạng đặc biệt, dùng cho các loại xe.
* Nhóm Audio

Cung cấp các component liên quan đến âm thanh trong game. Bao gồm các component:

* Audio Listener

Thể hiện một chiếc micro để lắng nghe các âm thanh có trong không gian 3D và thường được gắn vào Camera chính. Trong môi trường 2D thì có thể gắn ở bất kỳ đâu cũng được. Ở một scene, chỉ được phép tồn tại một Audio Listener.

* Audio Source

Đại diện cho một đối tượng phát ra âm thanh trong không gian. Trong môi trường 3D, nếu một Audio Source có vị trí càng gần Audio Listener thì âm thanh mà người dùng nghe được sẽ càng to hơn.

* Nhóm Terrain



Tạo ra địa hình cho game như mặt đất, đường, đồi núi, cây cối, biển, sông suối,... Nhóm này chỉ có một component là Terrain. Các thao tác chỉnh Terrain đều được thực hiện trên giao diện đồ hoạ.

### Script

Trong Unity, một file script định nghĩa một class cũng được xem là một component khi class đó thừa kế từ class MonoBehaviour.

Vì script là một phần không thể thiếu và được dùng vô cùng phổ biến, nên MonoBehaviour cũng là một class vô cùng quan trọng. Đây là class sẽ được hầu hết các class của người dùng thừa kế. Với JavaScript, tất cả các file script đều tự động thừa kế từ class này, tuy nhiên, với C# và Boo thì việc thừa kế phải được thực hiện tường minh.

Các thành phần quan trọng của class này bao gồm.

Thuộc tính:

|  |  |
| --- | --- |
| Tên | Ý nghĩa |
| enabled | Nếu enabled thì hàm Update của MonoBehaviour sẽ được gọi mỗi frame. |
| tag | Tag của GameObject, dùng để phân nhóm các GameObject. |
| name | Tên của GameObject, dùng để định danh các GameObject. |

Bảng . Các thuộc tính của MonoBehaviour

Phương thức:

|  |  |
| --- | --- |
| Tên | Ý nghĩa |
| GetComponent | Trả về component được gắn vào GameObject. |
| GetComponentInChildren | Trả về component được gắn vào GameObject hoặc bất kỳ đối tượng con của GameObject. |
| SendMessage | Gọi phương thức của GameObject theo tên có trong bất kỳ MonoBehaviour nào của đối tượng đó. |
| Invoke | Gọi phương thức theo tên sau một khoảng thời gian. |
| InvokeRepeating | Gọi lặp lại phương thức theo tên sau một khoảng thời gian. |
| CancelInvoke | Hủy hoặc dừng việc thực hiện 2 hàm trên. |
| IsInvoking | Kiểm tra phương thức có được chờ để gọi. |
| StartCoroutine | Bắt đầu một coroutine (tương tự thread). |
| StopCoroutine | Dừng tất cả coroutine chạy trên phương thức có tên này. |
| StopAllCoroutines | Dừng tất cả coroutine chạy trên đối tượng này. |

Bảng . Các phương thức của MonoBehaviour

Bên cạnh đó, có một vài hàm rất quan trọng, thường được sử dụng để thừa kế lại trong các script của người dùng.

|  |  |
| --- | --- |
| Tên | Ý nghĩa |
| Update | Sẽ được gọi ở mỗi frame, nếu MonoBehaviour được kích hoạt. |
| LateUpdate | Sẽ được gọi ở mỗi frame, sau hàm Update nếu MonoBehaviour được kích hoạt. |
| FixedUpdate | Được gọi ở mỗi frame cố định, nếu MonoBehaviour được kích hoạt, được dùng chủ yếu cho các xử lý vật lý, cần thời gian chính xác. |
| Awake | Được gọi khi đối tượng được load (trước hàm Start) và một lần duy nhất cho một đối tượng, thường được dùng để thực hiện các tiền xử lý. |
| Start | Được gọi đầu tiên (sau hàm Awake), thường được dùng để khởi tạo đối tượng. |
| Reset | Thiết lập lại các giá trị mặc định. |
| OnMouseEnter | Sẽ được gọi một lần khi đưa chuột vào GUIElement hoặc Collider. |
| OnMouseOver | Sẽ được gọi mỗi frame khi chuột nằm trên GUIElement hoặc Collider. |
| OnMouseExit | Sẽ được gọi một lần khi đưa chuột ra khỏi GUIElement hoặc Collider. |
| OnMouseDown | Sẽ được gọi lúc người dùng nhấn chuột và chuột đang nằm trên GUIElement hoặc Collider. |
| OnMouseUp | Sẽ được gọi khi thả chuột. |
| OnMouseDrag | Được gọi khi click lên GUIElement hoặc Collider và giữ trạng thái đó rồi di chuyển chuột. |
| OnTriggerEnter | Được gọi một lần khi các collider khác bắt đầu va chạm với trigger. |
| OnTriggerStay | Được gọi một lần cho từng frame cho tất cả các collider khác đang chạm vào trigger. |
| OnTriggerExit | Được gọi một lần khi collider khác không chạm vào trigger nữa. |
| OnCollisionEnter | Được gọi một lần khi collider/rigidbody bắt đầu chạm vào một collider/rigidbdy khác. |
| OnCollisionStay | Được gọi một lần mỗi frame cho tất cả các collider/rigidbody khác va chạm với collider/rigidbody này. |
| OnCollisionExit | Được gọi một lần khi collider/rigidbody khác không còn va chạm với collider/rigidbody này. |

Bảng . Các phương thức cần thừa kế của MonoBehaviour

### Physics

Vật lý trong game là tất cả những thứ liên quan đến sự chuyển động và tác động qua lại của các vật thể nhằm mô phỏng như trong thế giới thật, bao gồm sự chuyển động phức tạp của các vật thể, sự va chạm, phản hồi, ma sát, dòng chảy... dựa trên sự tác động của lực đẩy, trọng lực, gió...

Thông thường, CPU đảm nhiệm vai trò tính toán logic và vật lý trong game, GPU chỉ nhận nhiệm vụ dựng hình cho game. Chỉ những máy tính chơi game chuyên dụng, có card vật lý (PPU) thì các tính toán vật lý mới được thực hiện chuyên nghiệp ở đây.

|  |  |  |
| --- | --- | --- |
| PhysX | Core 2 Quad | GeForce GTX 280 |
| Cores | 4 | 240 |
| GFLOPS | 96 | 930 |
| Fluids | 1 | 15 |
| Soft Bodies | 1 | 12x |
| Cloth | 1 | 13x |

Bảng . So sánh tốc độ xử lý vật lý của CPU và GPU

Một lý do chính để chuyển việc tính toán vật lý từ CPU sang GPU là do sức mạnh to lớn trong tính toán của GPU cao hơn CPU rất nhiều lần. Qua bảng so sánh bên trên có thể thấy ưu thế rõ rệt của GPU so với CPU trong việc xử lý các hiệu ứng vật lý.

Hiện nay chỉ duy nhất engine vật lý PhysX của NVIDIA là có thể vừa được xử lý thông qua CPU lại vừa được xử lý thông qua GPU trong khi những engine như Havok, Box2D, Chipmunk lại chỉ có một cách xử lý duy nhất là thông qua CPU.

Unity sử dụng engine vật lý NVIDIA PhysX, điều này cho phép Unity có thể trình diễn những chức năng và hiệu ứng vật lý rất mượt mà và độc đáo.

Các tính năng của PhysX bao gồm:

* Giả lập đối tượng vật lý
* Phát hiện va chạm
* Điều khiển nhân vật
* Hiệu ứng particles
* Phương tiện giao thông
* Các loại trang phục

Với việc tích hợp hệ thống engine NVIDIA PhysX, Unity đã kế thừa được sức mạnh của hệ thống này để hổ trợ các xử lý vật lý của mình mượt mà nhất.

#### Các Component vật lý trong Unity

* Rigidbody

Để làm cho một đối tượng chịu ảnh hưởng của các yếu tố vật lý (trọng lực, va chạm,...) thì ta chỉ cần thêm component Rigidbody cho đối tượng đó.

Rigidbody có các thuộc tính vật lý, được dùng cho những GameObject có khả năng chịu tác động lực và tác động lực lên đối tượng khác.

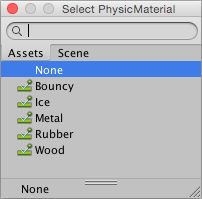
Các thuộc tính bao gồm:

|  |  |
| --- | --- |
| Tên | Ý nghĩa |
| Mass | Khối lượng của đối tượng. |
| Drag | Sức cản không khí lên đối tượng khi di chuyển. |
| Angular Drag | Sức cản không khí lên đối tượng khi quay. |
| Use Gravity | Nếu được kích hoạt, đối tượng sẽ chịu tác động bởi trọng lực. |
| Is Kinematic | Nếu được kích hoạt, đối tượng sẽ không chịu tác động bởi hệ thống vật lý, nó chỉ được điều khiển thông qua Transform. |
| Interpolate | Chế độ làm mượt chuyển động. Gồm các tham số:   * None: Không áp dụng * Interpolate: Transform sẽ mượt mà dựa trên Transform của khung hình trước đó. * Extrapolate: Transform sẽ mượt mà dựa trên dự đoán Transform của khung hình kế tiếp. |
| Collision Detetion | Được dùng để phát hiện va chạm trong các trường hợp vật di chuyển quá nhanh. |
| Constraints | Giới hạn phạm vi chịu tác động lựa của các Rigidbody, gồm giới hạn di chuyển và giới hạn quay theo 3 trục. |

Bảng . Các thuộc tính của Rigidbody

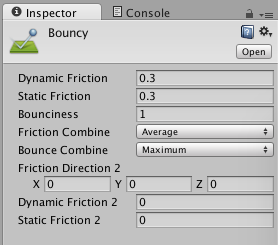
* Physic Material

Các tính chất của vật liệu như độ ma sát, độ đàn hồi, độ cứng... được định nghĩa trong Physics Material. Standard Assets có chứa hầu hết các loại vật liệu vật lý phổ biến. Ta cũng có thể tạo một loại vật liệu riêng bằng cách tùy chỉnh các thông số vật lý cho vật liệu.



Hình . Các loại Physic Material mặc định trong Unity

Các Physic Material đều có các thuộc tính chung sau:



Hình . Các thuộc tính chung của các Physic Material

Thuộc tính:

* *Dynamic Friction*: Ma sát động hay ma sát khi đã di chuyển. Có giá trị từ 0 đến 1. Giá trị 0 sẽ làm đối tượng trượt hoàn toàn trên bề mặt, giá trị 1 sẽ làm cho nó dừng nhanh chóng trừ khi có một lực lớn tác động lên vật.
* *Static Friction*: Ma sát nghỉ hay ma sát khi đứng yên. Có giá trị từ 0 đến 1. Giá trị 0 sẽ làm đối tượng trượt hoàn toàn trên bề mặt, giá trị 1 sẽ làm đối tượng khó có thể di chuyển.
* *Bounciness*: Độ đàn hồi. Giá trị 0 là không đàn hồi, giá trị 1 là đàn hồi hoàn toàn mà không mất năng lượng.
* *Friction combine mode*: Công thức xác định giá trị ma sát của 2 đối tượng va chạm khi kết hợp:
  + Average: giá trị ma sát là trung bình cộng của cả 2.
  + Min: giá trị nhỏ nhất trong 2 giá trị.
  + Max: giá trị lớn nhất trong 2 giá trị.
  + Multiply: tích của 2 giá trị.
* *Bounce combine*: Xác định giá trị đàn hồi của 2 đối tượng va chạm khi kết hợp, giống như Friction Combine mode.
* *Friction Direction 2*: Ma sát không cùng hướng được kích hoạt nếu biến này có giá trị khác Vector Zero. Dynamic Friction 2 và Static Friction 2 sẽ được áp dụng theo Friction Direction 2.
* *Dynamic Friction 2*: Nếu ma sát không cùng hướng được kích hoạt, Dynamic Friction 2 sẽ được áp dụng theo Friction Direction 2.
* *Static Friction 2*: Nếu ma sát không cùng hướng được kích hoạt, Static Friction 2 sẽ áp dụng theo Friction Direction 2.
* Các component Collider

Collider là một loại component phải được thêm vào cùng với rigidbody để cho phép va chạm xảy ra. Nếu 2 rigidbody va đập vào nhau, hệ thống vật lý sẽ không tính toán va chạm trừ khi cả hai đối tượng đều có gắn một component Collider.

Trong các trường hợp khác, Collider được sử dụng để xác định vùng không gian xung quanh vật thể để nhận các sự kiện mouse, touch hoặc để hàm raycast xác định được chính xác đối tượng.

Có các loại Collider gồm:

|  |  |
| --- | --- |
| Tên | Miêu tả |
| Box Collider | Hình dáng cơ bản của một khối hộp. |
| Sphere Collider | Hình dáng cơ bản của một khối cầu. |
| Capsule Collider | Hình dáng cơ bản của một khối trụ, hai đầu tròn (hình con nhộng). |
| Mesh Collider | Một Collider được tạo ra từ một Mesh của GameObject, không thể va chạm với Mesh Collider khác. |
| Wheel Collider | Một loại Collider đặc biệt sử dụng cho đối tượng xe cộ trong game. |

Bảng . Các loại Collider

* Joint

Joint được dùng để kết nối các đối tượng vật lý trong Unity lại với nhau, tạo thành một hệ chuyển động, tương tác với nhau. Một ví dụ như: con lắc đơn, lò xo, ...

Có ba loại Joint cơ bản: Fixed Joint, Spring Joint và Hinge Joint với các mục đích sử dụng khác nhau.

* Fixed Joint: Gắn liền các đối tượng lại với nhau
* Spring Joint: Kết nối các đối tượng dạng lò xo
* Hinge Joint: Kết nối các đối tượng dạng sợi dây
* Constant Force

Constant Force giúp ta tạo lực tác động liên tục vào Rigidbody. Điều này hoạt động tuyệt vời cho những đối tượng tăng tốc tức thời như tên lửa, viên đạn, ...

Thuộc tính:

* *Force*: Vector lực tác dụng trong hệ tọa độ thế giới.
* *Relative*: Vector lực tác dụng trong hệ tọa độ địa phương.
* *Torque*: Vector của mô-men xoắn tác dụng trong hệ tọa độ thế giới.
* *Relative Torque*: Vector của mô-men xoắn tác dụng trong hệ tọa độ địa phương.

## Photon



### Giới thiệu về SaaS

#### SaaS là gì?

SaaS (Software as a Service) là một dạng chuyển giao phần mềm cho phép truy cập dữ liệu từ bất kì thiết bị nào có kết nối internet và cài đặt trình duyệt web. Trong mô hình chạy trên web này, các nhà cung cấp phần mềm thực hiện nhiệm vụ lưu trữ và bảo trì các máy chủ, cơ sở dữ liệu và các đoạn mã để cấu thành ứng dụng. Đây là sự cải tiến đặc biệt so với mô hình on-premise do doanh nghiệp không cần đầu tư vào phần cứng dung lượng lớn để lưu trữ phần mềm. Doanh nghiệp không phải chịu các trách nhiệm kỹ thuật như khắc phục sự cố và bảo trì hệ thống. Các nhà cung cấp dịch vụ SaaS sẽ lo việc đó.



Ngoài việc cho phép truy cập từ xa vào các phần mềm ứng dụng và dữ liệu thông qua web, SaaS cũng khác với phần mềm on-premise về phương diện chi phí. Phần mềm on-premise thường được mua thông qua một giấy phép vĩnh viễn, nghĩa là doanh nghiệp sẽ sở hữu phần mềm đó. Ngoài ra, doanh nghiệp còn phải trả từ 15% đến 20% phí bảo trì và phí hỗ trợ hàng tháng. Ngược lại, SaaS cho phép người mua trả một khoản phí thuê bao hàng năm hoặc hàng tháng, chi phí đã bao gồm giấy phép sử dụng phần mềm, phí hỗ trợ và hầu hết các chi phí khác. Lợi ích chính của SaaS là có thể linh hoạt mở rộng quy mô dịch vụ để phù hợp với nhu cầu của doanh nghiệp theo thời gian, từ đó giúp tiết kiệm chi phí.

#### So sánh SaaS và on Premise

|  |  |  |
| --- | --- | --- |
|  | SaaS | On Premise |
| Chi phí cố định ban đầu | Không | Có – mua một giấy phép vĩnh viễn và thuộc sỡ hữu doanh nghiệp |
| Phí thuê bao hàng tháng | Có | Không |
| Phí bảo trì và phí hỗ trợ hàng tháng | Không – đã bao gồm trong phí thuê bao | Có – phải trả thêm 15 – 20% trên năm |
| Mở rộng quy mô dịch vụ | Có thể mở rộng bất kỳ lúc nào, tùy theo nhu cầu sử dụng | Phải mua bản nâng cấp mới |
| Trách nhiệm kỹ thuật (khắc phụ sự cố hoặc bảo trì hệ thống) | Nhà cung cấp dịch vụ SaaS | Doanh nghiệp |
| Quy mô doanh nghiệp | Vừa và nhỏ | Lớn |
| Quyền sở hữu dữ liệu | Doanh nghiệp sở hữu, cần quy định trong cam kết mức độ dịch vụ (SLA) | Thuộc sở hữu doanh nghiệp hoàn toàn |
| Cập nhật phiên bản mới | Nhà cung cấp dịch vụ SaaS tự động cập nhật thường xuyên, không cần trả phí | Cập nhật theo nhu cầu doanh nghiệp, cần trả phí |

#### Hạn chế của SaaS

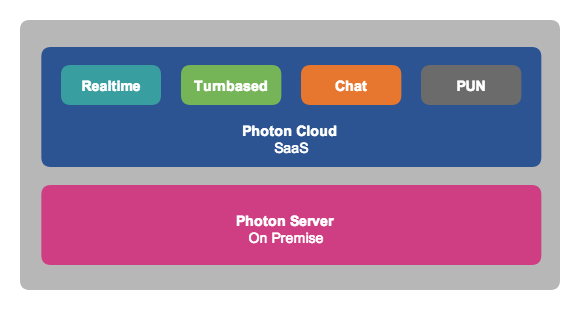
Nhược điểm chính của SaaS là nó phụ thuộc vào đường truyền kết nối Internet. Trong khi nhiều người tin rằng hệ thống on-premise là đáng tin cậy hơn thì thực tế không có hệ thống nào hoàn toàn không chịu ảnh hưởng của “downtime”. Phần mềm on-premise dễ bị mất điện, lỗi phần cứng và một loạt các rủi ro khác. Một số nhà cung cấp SaaS đã phát triển chức năng “offline” cho phép mọi người tiếp tục làm việc trong trường hợp rớt mạng như một biện pháp dự phòng. Khi internet được kết nối lại, tất cả dữ liệu sẽ được đồng bộ hóa với hệ thống.

Ngoài việc kết nối Internet, một số doanh nghiệp còn lo lắng về khả năng tương thích với hệ điều hành. Hầu hết các hệ thống của doanh nghiệp được xây dựng để chạy trên nền Windows hoặc Linux. Gần đây bắt đầu có sự thay đổi, nhiều nhà cung cấp còn hỗ trợ cả hệ điều hành Mac. Hy vọng việc hỗ trợ chức năng để tăng tốc độ hệ điều hành Mac sẽ được chú ý hơn khi càng có nhiều doanh nghiệp dùng Mac. Hơn nữa, hầu hết các công ty SaaS đều hỗ trợ nhiều trình duyệt web. Vậy nên, bất kể người dùng sử dụng hệ thống nào, đều có thể truy cập các vào các ứng dụng SaaS của họ.

### Photon Cloud và Photon Server

Photon Cloud là một Software as a Service (SaaS) nhanh và mạnh nhất để xây dựng các MMOs, FPS hoặc bất cứ thể loại game, ứng dụng multiplayer cho PC, Mac, trình duyệt web, Console, Mobile, .. .

Photon Server là một On Premise ứng dụng server mà bạn có thể chạy và host trên bất cứ máy chủ nào bạn muốn. Có thể tùy ý tùy chỉnh các configure và cài đặt multiplayer game backend theo cách bạn muốn.



Photon Cloud thì chạy trên Photon Server.

Photon Realtime, Photon Turnbased và Photon Chat là các ứng dụng chạy trên Photon Cloud.

Photon Unity Networking (PUN) là một plugin cho Unity 3D, cung cấp các API hỗ trợ Photon Realtime.

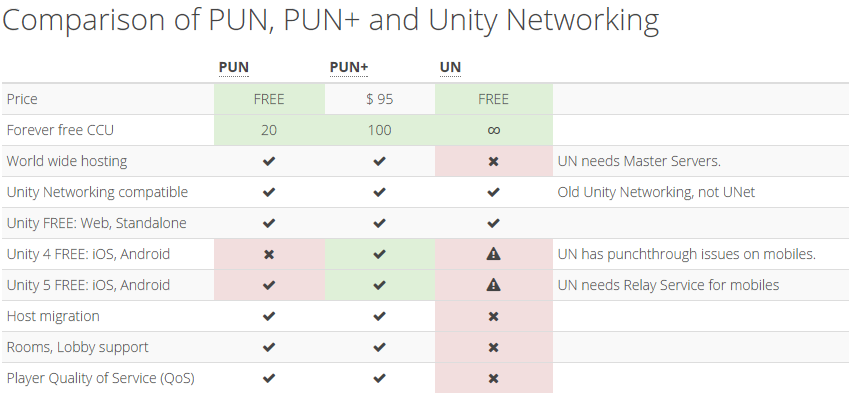
|  |  |  |
| --- | --- | --- |
|  | Photon Cloud | Photon Server |
| Quản trị server | Không cần server riêng, Photon Cloud sẽ tự động phần này. | Cần phải có server riêng để deploy ứng dụng. |
| Khả năng mở rộng | Tự động mở rộng để đáp ứng tất cả người dùng của ứng dụng. | Không tự động mở rộng. Nhà phát triển phải tự làm việc đó. |
| Game logic | Với Photon Realtime, Photon Turnbased, Photon Chat và PUN, nhà phát triển có thể tạo ra tất cả các loại game multiplayer. Có rất nhiều ví dụ chứa mã nguồn để developer dễ dàng bắt đầu. | Server logic có thể tùy chỉnh hoàn toàn với C#. Một số ứng dụng demo trong gói SDK cung cấp một framework hiệu suất cao cho lập trình viên. Dễ dàng thêm các chức năng mới. |
| Khả năng bắt đầu | Photon Cloud cho phép lập trình viên đăng ký và ngay lập tức chạy ứng dụng mà không cần phải setup gì thêm, không cần server deploy. | Sau khi tải về SDK, có thể bắt đầu ngay chỉ với hơn 5 phút. Photon Server rất dễ dàng cài đặt và chạy trên local hay remote. |
| Giấy phép | Có nhiều gói để chọn sao cho phù hợp với nhu cầu. Cho dù bạn là một indie, startup hay AAA studio, bạn sẽ tìm thấy một gói phù hợp. Gói miễn phí cho phép 20 người chơi tại cùng 1 thời điểm. | Gồm có loại trả theo tháng hoặc mua một lần trọn gói trong 60 tháng. Gói miễn phí cho phép 100 người chơi tại cùng 1 thời điểm. |

### Các ứng dụng chạy trên Photon Cloud

|  |  |
| --- | --- |
|  |  |
| Không cần server riêng. Photon Realtime chạy trên nền Photon Cloud. | Không cần server riêng. Photon Turnbased chạy trên nền Photon Cloud. |
| Nhanh và mạnh với các giao thức: R-UDP, TCP, Websockets, HTTP. | Đồng bộ hoặc không đồng bộ matchmaking và gameplay. |
| Cung cấp các API đầy đủ cho việc kết nối, tạo trận đấu và gameplay. | Chứng thực với Facebook, Steam hoặc Backend riêng. |
| Có hơn 130.000 lập trình viên sử dụng. | Hỗ trợ kết nối lại game sau khi rớt mạng. |
| Tự mở rộng liền mạch. | Code Webhooks với Heroku, Parse, Azure … |

|  |  |
| --- | --- |
|  |  |
| Không cần server riêng. Chạy trên nền Photon Cloud. | Không cần server riêng. Chạy trên nền Photon Cloud. |
| Subscribe và publish trong kênh. | Tích hợp đầy đủ với Unity. |
| Hỗ trợ tin nhắn văn bản và tin nhắn dữ liệu binary. | Hỗ trợ đầy đủ cả nền tảng: Browser, PC, Mobile và Console |
| Lưu lịch sử tin nhắn và trò chuyện | Bao gồm tất cả chức năng của Photon Realtime |

### So sánh PUN, PUN + và Unity Networking



### So sánh Photon Cloud với các Service khác

|  |  |  |  |
| --- | --- | --- | --- |
|  | Photon cloud | Smartfox | Gamesparks |
| Giao thức | HTTP, UDP, TCP | TCP | TCP |
| Data type | Tất cả | String | String |
| Game logic | Hỗ trợ rất nhiều ứng dụng với API rõ ràng (Photon realtime, Turnbased, Chat, PUN…) | Hỗ trợ các API cần thiết để viết các chức năng, game logic | Hỗ trợ các API cần thiết để viết các chức năng, game logic |
| Platform support | Android, iOS, Cocos2d-x, Unity, .Net, Flash, Linux, Javascript, HTML5, Xamarin, Xbox, Unreal, PS | Flash, Unity, HTML5, iOS, Windows, Android, Standard C++ | Unity, Marmalade, Unreal, Cocos2d-x, JavaScript, HTML5, Action Script, C++ |
| Price | Free nếu <20 CCU | Free nếu <20 CCU | Free cho 10.000 người chơi đầu tiên. Phải trả phí cho những người chơi tiếp theo. |

## Parse



### Giới thiệu

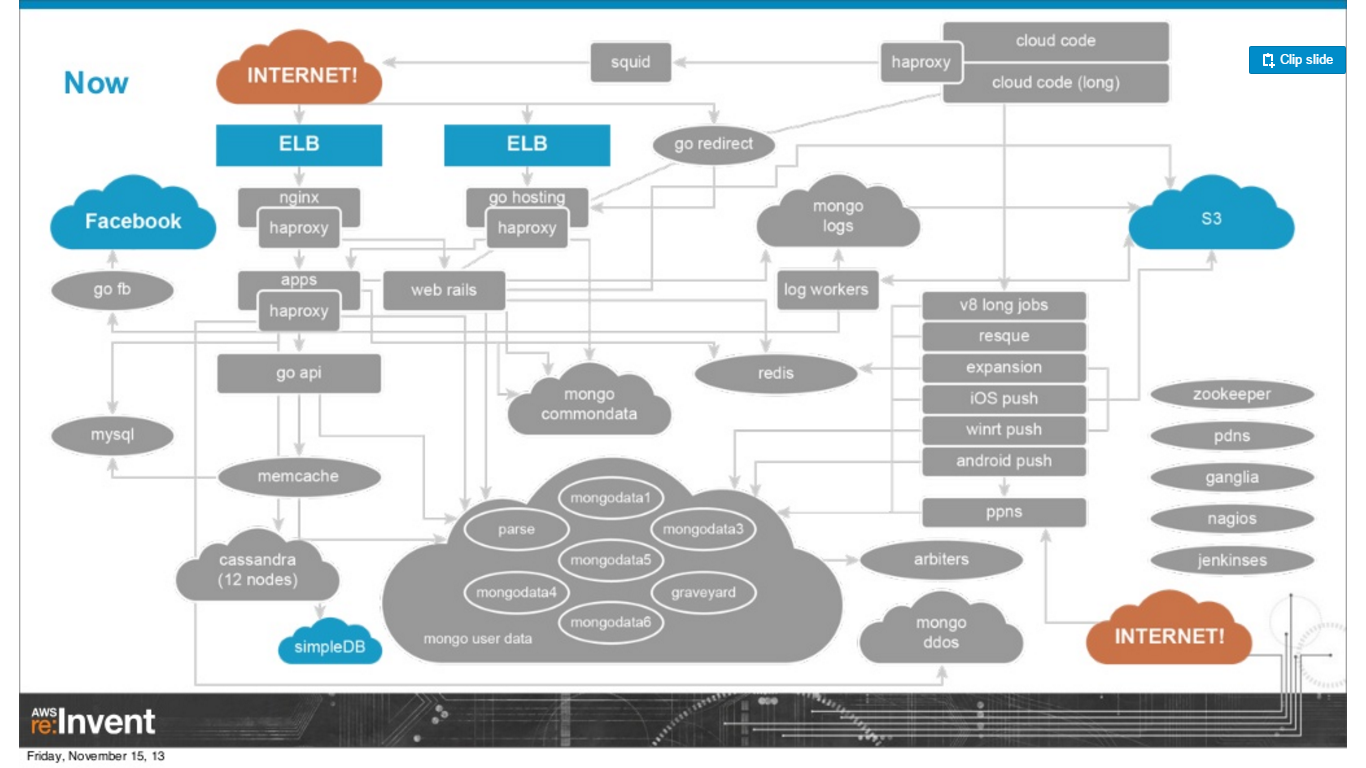
Parse là một Mobile Backend as a Service (MBaaS) – Nền tảng đám mây cung cấp các API và dịch vụ đám mây cho các ứng dụng Desktop, Mobile, Web, IoT …



Các dịch vụ đám mây Parse cung cấp bao gồm:

* Core (Cloud-based Storage)
* Parse xử lý tất cả mọi thứ cần thiết để lưu trữ dữ liệu một cách an toàn và hiệu quả trên cloud. Lưu trữ từ các loại dữ liệu cơ bản tới địa điểm, media và truy cập chúng trên bất kỳ nền tảng nào mà không cần thông qua một server duy nhất nào.
* Local Datastore cho phép ứng dụng sử dụng tất cả các truy vấn và bảo mật của Parse ngay cả khi offline.
* Quản lý người dùng dễ dàng hơn. Parse cung cấp hệ thống quản lý người dùng rất chi tiết và đầy đủ. Bao gồm các chức năng đăng ký, xác nhận email, reset password, đăng nhập bằng facebook.
* Lập trình server-side logic cho mọi nền tảng dễ dàng với Cloud Code. Với nó, lập trình viên có thể dễ dàng tùy chỉnh các trigger (before save, after save…), cài đặt các hàm xử lý logic với Webhooks. Ngoài ra ta có thể sử dụng Parse Cloud để xây dựng các Background Jobs tự động thực hiện các xử lý dưới nền từ server xuống các client như Push Notification, cập nhật dữ liệu, gửi mail…
* Push
* Dễ dàng gửi notification tới client chỉ với vài dòng code. Các mobile SDK của Parse đều được hỗ trợ tính năng push notification mặc định. Lập lịch cho các thông báo tới tất cả client, một nhóm hoặc một client cụ thể chỉ với vài click từ Parse Dashboard.
* Tạo các kênh để lấy về thông báo nhanh hơn.
* Phân khúc đối tượng mục tiêu dựa trên bất kỳ điều kiện bào gồm tuổi, địa phương và ngôn ngữ để gửi thông báo phù hợp.
* Bản địa hóa cho các thông báo tin nhắn.
* Analytics
* Nắm bắt quá trình dùy trì và phát triển của ứng dụng. Với Analytics, nhà phát triển có thể theo dõi các thông tin về các cài đặt, sô lần mở ứng dụng, tần suất hoạt động, và nhiều hơn nữa. Qua đó hiểu được thái độ của người dùng đối với ứng dụng, để làm ứng dụng tốt hơn.
* Theo dõi bất kỳ sự kiện chỉ với vài dòng code.
* Loại bỏ lag ứng dụng với công nghệ truy vấn chậm. Parse tự động nhận diện các truy vấn chậm, không hiểu quả để lập trình viên biết chính xác vị trí cần sửa lỗi, từ đó cải thiện chất lượng ứng dụng.
* Parse Explorer cung cấp một ngôn ngữ truy vấn mạnh mẽ để giúp nhà phát triển tìm hiểu những gì đang xảy ra. Sử dụng nó để khai thác thông tin từ các log và tìm hiểu thêm về cách người dùng sử dụng ứng dụng.

### Kiến trúc



Parse được xây dựng trên Amazon Web Services và sử dụng nhiều công nghệ như MongoDB, Go, Rails, …

Parse chạy cơ sở dữ liệu MongoDB của nó trong các bộ replica gồm 1 co sở dữ liệu chính và 2 cơ sở dữ liệu phụ. Parse sử dụng Amazon Elastic Block Store (Amazon EBS) để tạo những snapshot cho từng phân đoạn MongoDB, mà sau đó được tải lên một Amazon Simple Storage Service (Amazon S3). Nếu cần thiết, Parse có thể mang tới một node mới trong vài phút sử dụng Amazon với S3 và đưa nó vào một cluster.

### NoSQL

NoSQL là 1 dạng CSDL mã nguồn mở không sử dụng T-SQL để truy vấn thông tin. NoSQL viết tắt bởi: None-Relational SQL, hay có nơi thường gọi là Not-Only SQL.

NoSQL được phát triển trên JavaScript framework với kiểu dữ liệu JSON và dạng dữ liệu theo kiểu key-value. NoSQL ra đời như là 1 mảnh vá cho những khuyết điểm và thiếu sót cũng như hạn chế của mô hình dữ liệu quan hệ RDBMS về tốc độ, tính năng, khả năng mở rộng, memory cache,...

* Ưu điểm

Khi sử dụng SQL Server hoặc MySQL thì giữa các bảng trong CSDL sẽ có các mối quan hệ khóa chính-khóa ngoại vì vậy sẽ gặp nhiều khó khăn trong việc mở rộng dữ liệu sau này. Nhưng trong NoSQL thì khác, ta có thể mở rộng dữ liệu mà không lo tới những việc như tạo khóa ngoại, khóa chính, kiểm tra ràng buộc, ...

* Nhược điểm

Vì NoSQL không hạn chế việc mở rộng dữ liệu nên tồn tại nhiều nhược điểm như: sự phụ thuộc của từng bản ghi, tính nhất quán, tính toàn vẹn dữ liệu,... nhưng chúng ta có thể chấp nhận những nhược điểm đó để khiến ứng dụng có được hiệu suất cao hơn khi giải quyết những bài toán lớn về hệ thống thông tin, phân tán hay lưu trữ dữ liệu.

* Sử dụng

*NoSQL được sử dụng ở đâu?*

NoSQL được sử dụng ở rất nhiều công ty, tập đoàn lớn, ví dụ Facebook sử dụng Cassandra do Facebook phát triển, Google phát triển và sử dụng BigTable,... và rất nhiều công ty lớn khác.

### MongoDB

MongoDB là 1 hệ thống CSDL mã nguồn mở được phát triển và hỗ trợ bởi 10gen (hiện tại là MongoDB Inc), là CSDL NoSQL hàng đầu được hàng triệu người sử dụng. Vì được viết bởi C++ nên nó có khả năng tính toán với tốc độ cao, hơn hẳn các hệ quản trị CSDL hiện nay.

Thay vì lưu trữ dữ liệu dưới dạng bảng và các tuple như trong các CSDL quan hệ thì nó lưu trữ dữ liệu dưới dạng JSON (trong MongoDB được gọi là dạng BSON vì nó lưu trữ dưới dạng binary từ 1 JSON document).

Hiện nay MongoDB đang được sử dụng tại một số công ty lớn như: MTV Networks, Craigslist, Foursquare, Forbes, Expedia, Bosch, và nhiều công ty lớn khác.

#### Các khái niệm

* Collection

Collection trong MongoDB là nhóm các tài liệu (document), nó tương đương với một bảng (table) trong CSDL thông thường nên mỗi Collection sẽ thuộc về một database duy nhất. Tuy nhiên, có một sự khác biệt đó là không có ràng buộc giữa các Collection như trong các hệ quản trị CSDL khác nên việc truy xuất rất nhanh. Đồng thời mỗi Collection có thể chứa nhiều loại dữ liệu khác nhau không giống như các field cố định trong table của MySQL.

* Document

Document trong MongoDB có cấu trúc tương tự như kiểu dữ liệu JSON, nghĩa là sẽ có các cặp key-value nên nó có tính năng động rất lớn. Ta có thể hiểu Document giống như các record dữ liệu trong MySQL, tuy nhiên, có một sự khác biệt là các cặp key-value có thể không giống nhau ở mỗi Document.

Để rõ hơn, ta có bảng so sánh các thuật ngữ giữa Cơ sở dữ liệu quan hệ (RDBMS) và MongoDB:

|  |  |
| --- | --- |
| RDBMS | MongoDB |
| Database | Database |
| Table | Collection |
| Tuple/Row | Document |
| Column | Field |
| Table Join | Embedded Documents |
| Primary Key | Primary Key (mặc định là \_id) |

Bảng . So sánh các thuật ngữ trong RDBMS và MongoDB

#### Tính năng chính

* Document-Oriented

Thay vì chia nhỏ các thuộc tính của một đối tượng để lưu trong các bảng khác nhau như trong SQL thì MongoDB lưu toàn bộ đối tượng trong một Document của một Collection. Ví dụ: Thay vì lưu *TênSách* và *TácGiả* ở hai bảng khác nhau như trong MySQL, thì MongoDB lưu toàn bộ đối tượng *Sách* vào một Document của một Collection, trong đó chứa toàn bộ các thông tin của đối tượng *Sách*.

* Server-side JavaScript Execution

JavaScript có thể được sử dụng ở phía server để thao tác trực tiếp với MongoDB.

* Indexing

Tất cả các key-value trong MongoDB đều có thể được dùng để index. Việc index một key-value giúp tăng tốc độ truy vấn đối với key-value đó lên rất nhiều lần.

* Ad hoc querries

MongoDB cho phép tìm kiếm theo key, theo miền giá trị, sử dụng regular expression để tìm kiếm. Truy vấn có thể trả về một vài key-value của dữ liệu, và thậm chí có thể trả về một hàm được người dùng định nghĩa.

* Replication

MongoDB cung cấp cơ chế tạo bản sao hiệu quả, qua đó tăng cường khả năng phân tán dữ liệu, một trong những tính năng được sử dụng rất nhiều trong những hệ thống lớn hiện nay.

* Load Balancing

MongoDB cho phép mở rộng cơ sở dữ liệu theo chiều dọc dựa trên cơ chế phân mảnh (Shard). Người dùng có thể chọn một *Shard Key* để xác định cách mà dữ liệu được phân mảnh. Dữ liệu sẽ được cắt nhỏ thành những miền giá trị và phân ra nhiều mảnh. MongoDB cho phép chạy trên nhiều server, cho phép tạo bản sao để phòng ngừa các trường hợp một server trong hệ thống gặp lỗi.

* File Storage

MongoDB cho phép phân tán các file nhằm tăng cường khả năng Load Balancing và Replication dữ liệu.

* Capped Collections

MongoDB cho phép tạo ra các Collection có kích thước cố định, và khi dữ liệu đầy, nó sẽ được ghi đè ở đầu bên kia của Collection. Cơ chế giống như hàng đợi vòng tròn (circular queue).

#### Sử dụng

MongoDB thật sự rất tốt nhưng không phải lúc nào ta cũng sử dụng, những trường hợp nên sử dụng:

* Nếu website có tính chất INSERT cao, bởi vì mặc định MongoDB có sẵn cơ chế ghi với tốc độ cao và an toàn.
* Hệ thống rộng lớn và không đáng tin cậy. MongoDB cung cấp cơ chế làm việc trên nhiều server, replication và recover dữ liệu tự động, an toàn cho phép hệ thống có thể thoả mãn điều kiện này.
* Dữ liệu dựa trên khu vực. MongoDB đã xây dựng sẵn các chức năng không gian, vì vậy việc tìm kiếm dữ liệu có liên quan đến các địa điểm cụ thể rất nhanh chóng và chính xác.
* Website ở dạng thời gian thực nhiều, nghĩa là nhiều người thao tác với ứng dụng. Nếu trong quá trình load bị lỗi tại một điểm nào đó thì nó sẽ bỏ qua phần đó nên sẽ an toàn.
* Website có lượng dữ liệu lớn. Giả sử website có đến 10 triệu record thì đó là cơn ác mộng với MySQL. Bởi vì MongoDB có khả năng tìm kiếm thông tin liên quan khá nhanh nên trường hợp này nên dùng nó.

Những trường hợp không nên sử dụng:

* Các ứng dụng cần sử dụng nhiều transaction như ngân hàng
* Các ứng dụng cần ràng buộc chặc chẽ

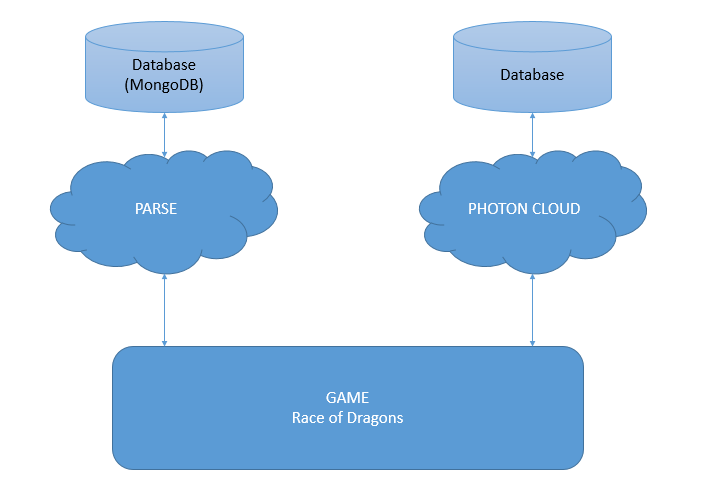
# PHÂN TÍCH THIẾT KẾ

## Tổng quan chức năng

Game Race of Dragons gồm các chức năng chính sau:

* Đăng ký user
* Đăng nhập và đồng bộ dữ liệu với server
* Kết bạn, xem thông tin bạn bè, mời bạn cùng chơi
* Hệ thống shop mua hàng, nâng cấp items
* Hỗ trợ 3 mode chơi chính:
  + Practice: chế độ luyện tập
  + Quick race: chế độ matching ngẫu nhiên người chơi
  + Invite friend: chế độ mời bạn bè cùng chơi
* Hệ thống xếp hạng người chơi

## Kiến trúc tổng quát



Hình . Kiến trúc tổng quát hệ thống

Trong đó:

* Parse giữ chức năng quản lý data user bao gồm quản lý tài khoản, quản lý bạn bè, hệ thống ranking, quản lý dữ liệu người chơi, …
* Photon Cloud giữ vai trò cầu nối, quản lý hệ thống mutiplayer, tạo room, join room, …

## Game Race of Dragons

### Yêu cầu

#### Tổng quan

* Tên game: Race of Dragon
* Thể loại: Action, Multiplayer, Racing
* Đối tượng hướng đến: 6+
* Nền tảng: Android, iOs
* Chủ đề: Đua rồng, lấy cảm hứng từ bộ phim “How to train your dragon”



* Thời gian dự kiến: 3 tháng

#### Lý do thành công của dự án

* Chủ đề hấp dẫn: Rồng
* Thể loại Running kết hợp với Action và Multiplayer rất thú vị
* Xây dựng cách chơi mới lạ

#### Cốt truyện và gameplay

* **Cốt truyện:** Thế giới, nơi mà ngày xưa đã từng là những cuộc chiến liên miên giữa Người và Rồng, đến khi một cậu bé tìm được một chú rồng bị thương và cứu chữa nó, sau đó cả hai trở thành bạn bè, cùng nhau chinh phục bầu trời. Từ đó trở đi, Người và Rồng cùng chung sống với nhau, cùng sinh hoạt, làm việc, vui chơi. Trong đó, trò chơi lôi cuốn nhất, chính là đua rồng.
* **Gameplay:** Người chơi sẽ là một phần trong thế giới đó, cùng với chú rồng của mình vui chơi, rèn luyện để trở nên mạnh mẽ và đoạt lấy chiếc vương miện của Rồng trong cuộc đua kỳ thú. Người chơi sẽ được chọn ngẫu nhiên một chủng loại rồng để làm bạn đồng hành với mình lúc mới bắt đầu game. Trải qua những cuộc đua luyện tập với AI hoặc đua với những người chơi khác, cố gắng kiếm thật nhiều tiền để nâng cấp trang bị, kỹ năng cho rồng.

### Phân tích

Để thiết kế và lập trình game Race of Dragons với các chức năng đề ra ta cần phải giải quyết các vấn đề sau:

* Vấn đề multiplayer và xử lý đồng bộ
* Vấn đề quản lý dữ liệu người dùng và đồng bộ dữ liệu khi người chơi đăng nhập trên các device khác nhau
* Vấn đề kết bạn và mời bạn bè chơi cùng

### Cấu trúc dữ liệu

* Player

Là đối tượng chính của game. Những thông số của Player được lưu trên cả client và server. Khi người chơi đăng nhập, dữ liệu từ server và client được đồng bộ với nhau. Các thông số của Player:

* Level: cấp độ của người chơi
* Exp: kinh nghiệm
* Gold: số vàng người chơi kiếm được
* Gem: số đá quý của người chơi
* Played: số trận đấu người chơi đã chơi
* Win: số trận thắng của người chơi
* Dragons: danh sách rồng người chơi sỡ hữu
* Items: danh sách vật phẩm của người chơi
* Emojis: danh sách các mô tả trạng thái cảm xúc của người chơi
* Dragon

Là lớp quản lý các thuộc tính của từng loại rồng mà người chơi sở hữu. Các thông số gồm:

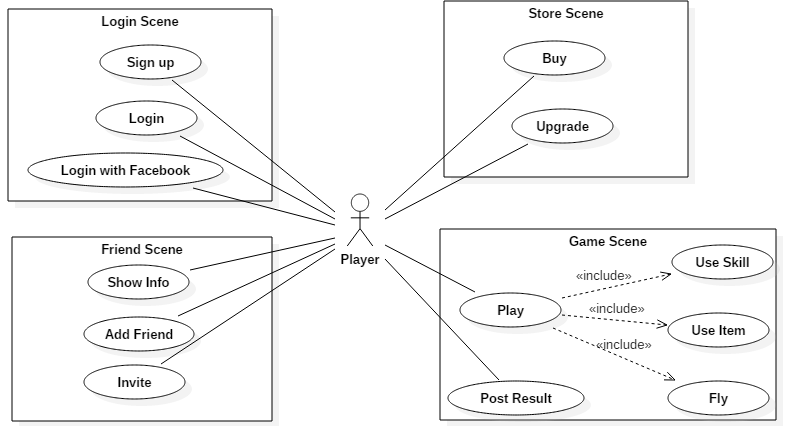
* Level: cấp độ của rồng
* Element: chủng loại rồng (lửa, băng, sấm sét, … )
* TimeCooldown: thời gian hồi chiêu của rồng (dựa vào chủng loại và cấp độ để tính toán thông số này)
* Item

Là lớp quản lý các thuộc tính của vật phẩm trong game khi người chơi nhặt được. Các thông số gồm:

* Level: cấp độ của vật phẩm
* ItemType: loại vật phẩm
* TimeDuration: thời gian hiệu lực của vật phẩm (dựa vào cấp độ và chủng loại vật phẩm để tính toán thông số này)

### Sơ đồ use case

Race of Dragons có 6 màn hình chính, do đó, sơ đồ use case được chia thành 6 subsystem tương ứng với các màn hình: Login Scene, Menu Scene, Select Scene và Game Scene.



Hình . Sơ đồ use case của Race of Dragons Game

Danh sách Actor:

|  |  |  |
| --- | --- | --- |
| STT | Tên Actor | Ý nghĩa |
| 1 | Player | Người chơi chính |

Bảng . Danh sách Actor trong sơ đồ use case của Traffic Game

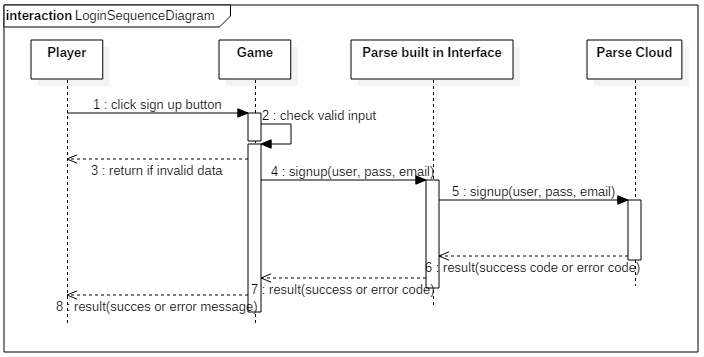
Danh sách use case:

|  |  |  |
| --- | --- | --- |
| STT | Tên use case | Ý nghĩa |
| 1 | Sign up | Đăng ký user |
| 2 | Login | Đăng nhập vào game |
| 3 | Login with Facebook | Đăng nhập vào game bằng tài khoản Facebook |
| 4 | Show Info | Hiển thị thông tin người chơi |
| 5 | Add Friend | Thêm người chơi vào danh sách bạn bè |
| 6 | Invite | Mời người chơi vào trận đấu |
| 7 | Buy | Mở khóa chủng loại rồng mới, vật phẩm mới |
| 8 | Upgrade | Nâng cấp rồng, vật phẩm |
| 9 | Play | Chơi game |
| 10 | Post Result | Cập nhật và lưu dữ liệu kết quả lên server và xuống database |
| 11 | Use Skill | Sử dụng kỹ năng của rồng |
| 12 | Use Item | Sử dụng vật phẩm đang có |
| 13 | Fly | Điều khiển rồng bay |

Bảng . Danh sách use case trong Race of Dragons Game

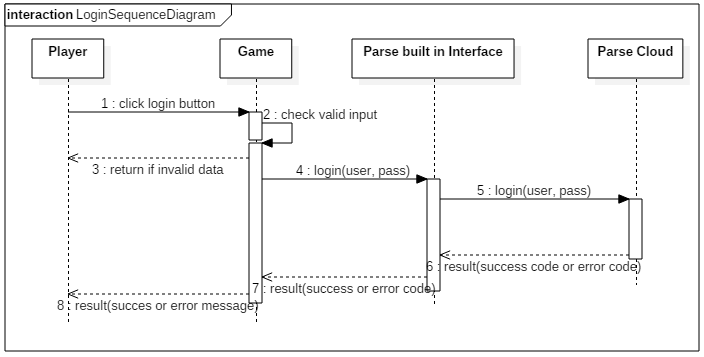
### Sơ đồ tuần tự

* Sơ đồ tuần tự use case Sign Up



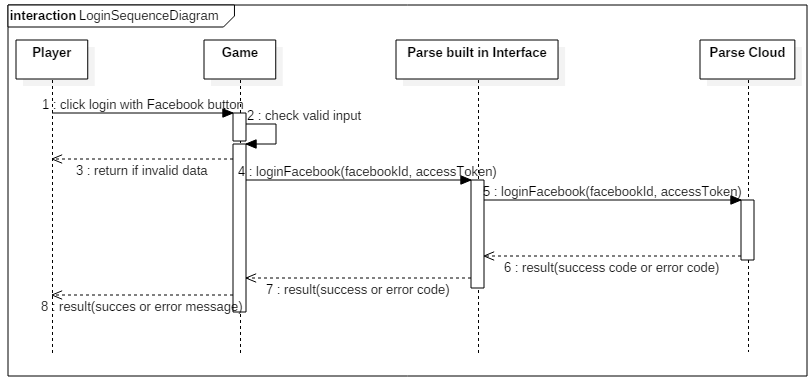
Hình . Sơ đồ tuần tự use case Sign Up

* Sơ đồ tuần tự use case Login



Hình . Sơ đồ tuần tự use case Login

* Sơ đồ tuần tự use case Login with Facebook



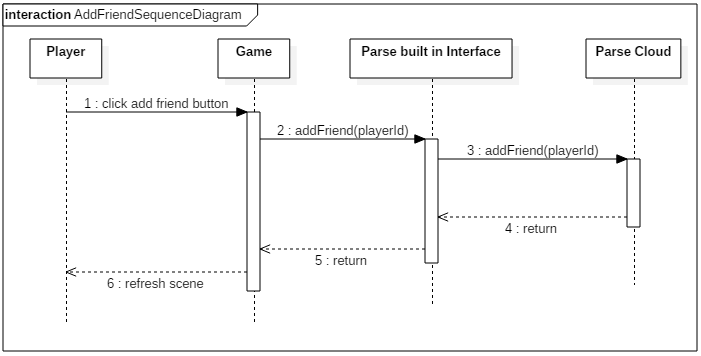
Hình . Sơ đồ tuần tự use case Login with Facebook

* Sơ đồ tuần tự use case Show Info



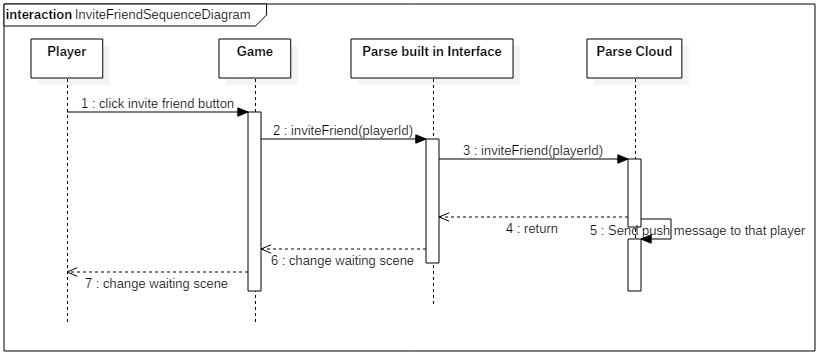
Hình . Sơ đồ tuần tự use case Show Info

* Sơ đồ tuần tự use case Add Friend



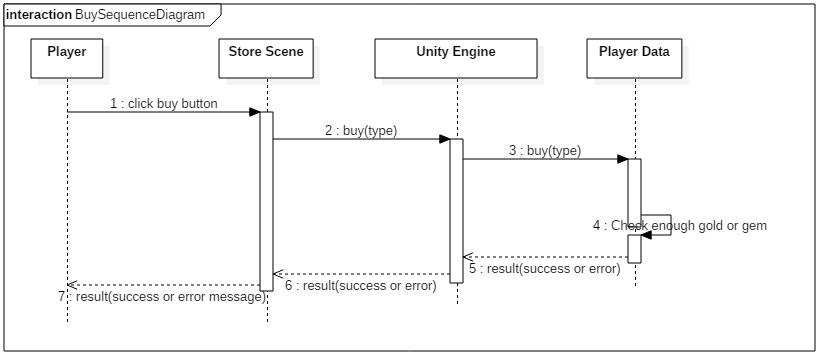
Hình . Sơ đồ tuần tự use case Add Friend

* Sơ đồ tuần tự usecase Invite Friend



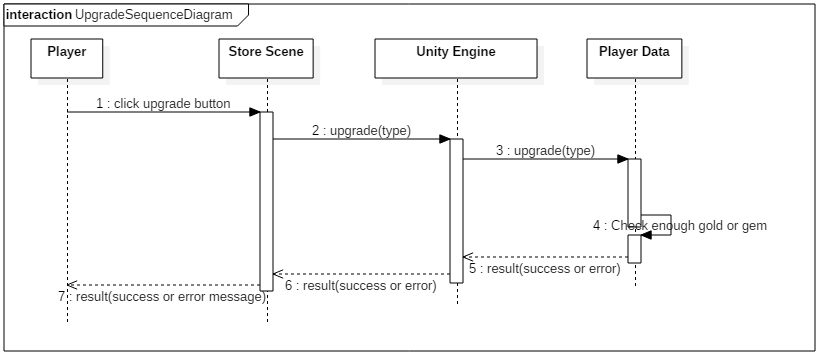
Hình 3.12 Sơ đồ tuần tự use case Add Friend

* Sơ đồ tuần tự usecase Buy



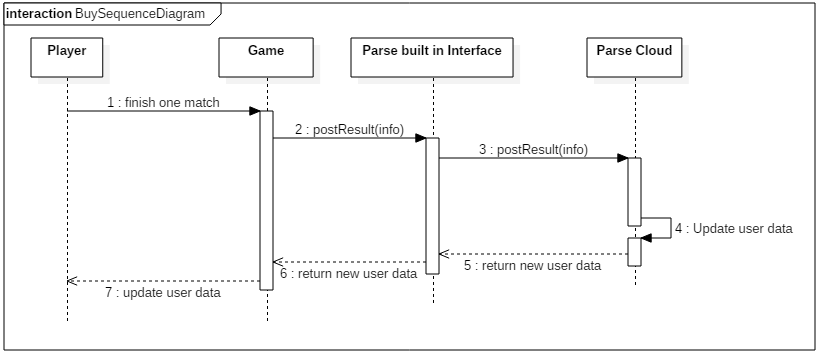
Hình 3.13 Sơ đồ tuần tự use case Buy

* Sơ đồ tuần tự usecase Upgrade



Hình 3.14 Sơ đồ tuần tự use case Upgrade

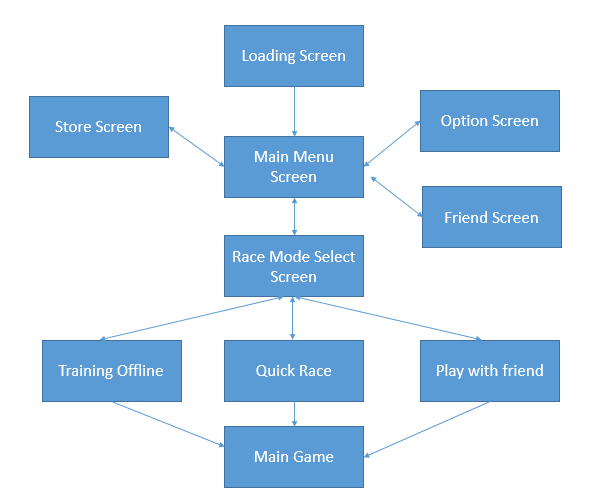
* Sơ đồ tuần tự usecase Post Result



Hình 3.14 Sơ đồ tuần tự use case Post Result

### Thiết kế giao diện

#### Game flow



#### Màn hình Loading



Nhiệm vụ chính:

* Load các config từ file
* Load hình ảnh, âm thanh trong game
* Kiểm tra bản cập nhật mới nếu có

#### Màn hình MainMenu



Nhiệm vụ chính:

* Hiển thị các UI để dẫn tới các màn hình chức năng chính (Game, Store, Friends, Option)
* Hiển thị tên game

Chú giải giao diện:

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Tên | Chức năng | Ghi chú |
| 1 | Button Play | Chuyển sang màn hình chọn mode chơi |  |
| 2 | Button Store | Chuyển sang màn hình Cửa hàng |  |
| 3 | Button Friends | Hiển thị dialog danh sách bạn bè | Có animation |
| 4 | Button Option | Hiển thị dialog các cài đặt trong game và một số thông tin về credit và support user | Có animatio |

#### Màn hình Cửa hàng



Nhiệm vụ chính:

* Hiển thị thông tin về số gold và số gem của người chơi đang có
* Hiển thị thông tin về các loại rồng người chơi đang sỡ hữu
* Hiển thị thông tin level các vật phẩm của người chơi
* Hiển thị thông tin các cảm xúc của người chơi
* Cho phép người chơi mua rồng và nâng cấp rồng, vật phẩm và chỉnh sửa trạng thái cảm xúc
* Cho phép người chơi mua thêm gold và gem

Chú giải giao diện:



|  |  |  |  |
| --- | --- | --- | --- |
| STT | Tên | Chức năng | Ghi chú |
| 1 | Text Gems | Hiển thị số lượng gem mà người chơi đang có |  |
| 2 | Text Golds | Hiển thị số lượng gold người chơi đang có |  |
| 3 | Button Character | Hiển thị danh sách các loại rồng trong listview bên phải |  |
| 4 | Button Item | Hiển thị danh sách các vật phẩm trong listview bên phải |  |
| 5 | Button Chat | Hiển thị danh sách các text trạng thái cảm xúc của người chơi |  |
| 6 | Text Title và Level | Hiển thị tên của item và level của nó |  |
| 7 | Image | Hiển thị hình ảnh của item | Có animation |
| 8 | Button Buy/Upgrade | Hiển thị giá mua/ nâng cấp và chức năng mua/ nâng cấp |  |
| 9 | Button Previous | Xem item phía trước |  |
| 10 | Button Next | Xem item kế tiếp |  |
| 11 | Button Back | Thoát khỏi màn hình Cửa hàng, quay trở lại màn hình Main Menu |  |

# CÀI ĐẶT MINH HOẠ

## Traffic Game

Traffic Game khi cài đặt thực tế có thể chạy tốt trên tất cả các trình duyệt phổ biến có cài đặt Unity Web Player, bao gồm: Chrome, Safari, Firefox, Opera, Internet Explorer.

Game có thể phát hiện ra: 30 tình huống vi phạm giao thông, chủ yếu là các tình huống vi phạm khi đang di chuyển trên đường.

Hiện tại, Traffic Game đã được cài đặt tại địa chỉ:

<http://widocom.com/projects/trafficgame/game>

Tài khoản khả dụng:

* username: user
* password: user

### Giao diện

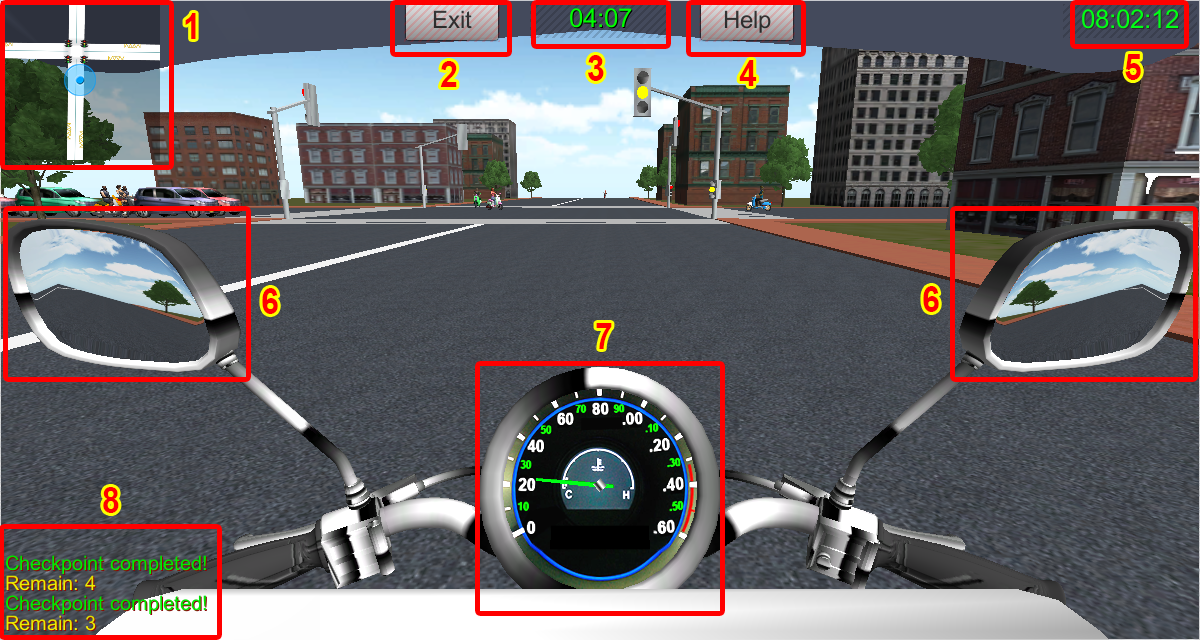
Giao diện khi cài đặt thực tế của Traffic Game như sau:



Hình . Giao diện Traffic Game

### Hướng dẫn sử dụng

* Hiển thị



Hình . Các thành phần giao diện Traffic Game

Danh sách các thành phần:

|  |  |  |
| --- | --- | --- |
| STT | Tên | Ý nghĩa |
| 1 | Mini Map | Bản đồ thu nhỏ |
| 2 | Exit | Thoát màn chơi |
| 3 | Timer | Thời gian còn lại cho màn chơi |
| 4 | Helper | Hướng dẫn chơi |
| 5 | Clock | Đồng hồ hiển thị thời gian trong hệ thống |
| 6 | Mirror | Gương chiếu hậu |
| 7 | Velocity Clock | Đồng hồ tốc độ của xe |
| 8 | Notifier | Bảng thông báo |

Bảng . Các thành phần giao diện Traffic Game

* Điều khiển

Điều khiển trong Traffic Game bao gồm các phím sau:

|  |  |
| --- | --- |
| Tên | Ý nghĩa |
| Left/A | Rẽ trái |
| Right/D | Rẽ phải |
| Up/W | Đi thẳng, tăng tốc |
| Down/S/Space | Phanh xe |
| Q | Bật đèn rẽ trái (tắt đèn rẽ phải) |
| E | Bật đèn rẽ phải (tắt đèn rẽ trái) |
| L | Bật/tắt đèn chiếu trước |
| F | Bật đèn chiếu xa |
| N | Bật đèn chiếu gần |
| B | Bấm còi |

Bảng . Các phím điều khiển trong Traffic Game

Trước khi xuất phát, cần đội mũ bảo hiểm bằng cách click chọn vào mũ bảo hiểm.

* Luật chơi

Mỗi bản đồ sẽ có tổng thời gian khác nhau để hoàn thành. Nếu trong thời gian đó, người chơi không hoàn thành thì cuộc chơi sẽ dừng tại đó.

Trong mỗi bản đồ, sẽ có các trạm dừng mà người chơi phải đi qua và điểm đến cuối cùng là trạm dừng quan trọng nhất.

Hãy hoàn thành bản đồ và đi đúng luật giao thông. Tất cả lỗi vi phạm sẽ được ghi lại và hiển thị cuối màn chơi.

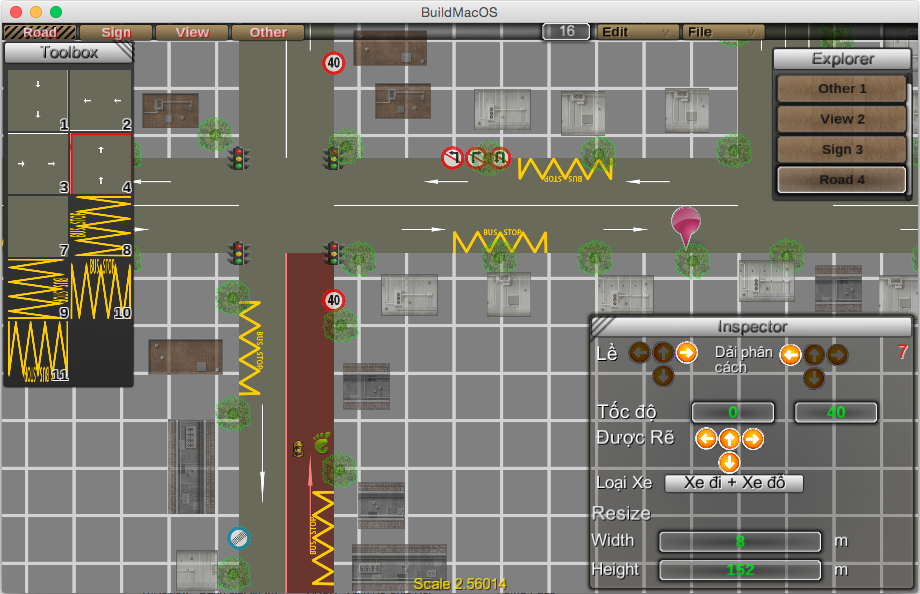
## Map Editor

Map Editor có chạy trên nhiều nền tảng khác nhau, bao gồm Windows, Mac OSX, Linux và Web thông qua Unity Web Player, điều này cho phép nhà quản trị có thể sử dụng ở mọi lúc, mọi nơi.

Hiện tại, Map Editor đã được cài đặt tại địa chỉ:

<http://widocom.com/projects/trafficgame/editor>

### Giao diện



Hình . Giao diện Map Editor

### Hướng dẫn sử dụng

* Tạo map mới bằng menu File -> New map
* Lựa chọn layer phù hợp để chỉnh sửa ở Explorer
* Lựa chọn đối tượng cần tạo ở Toolbox
* Kéo thả đối tượng ở Grid
* Click vào đối tượng để chọn
* Chỉnh sửa các thuộc tính của đối tượng ở Inspector
* Lưu file ở menu File -> Save map

## CMS

### Yêu cầu phần cứng

Yêu cầu về máy chủ web để có thể cài đặt CMS:

* Hệ điều hành: Linux
* Web server: Apache
* MongoDB 2.6
* PHP 5.3 trở lên, có cài đặt MongoDB driver.
* FTP Server

Hiện tại, hệ thống CMS đã được cài đặt tại địa chỉ:

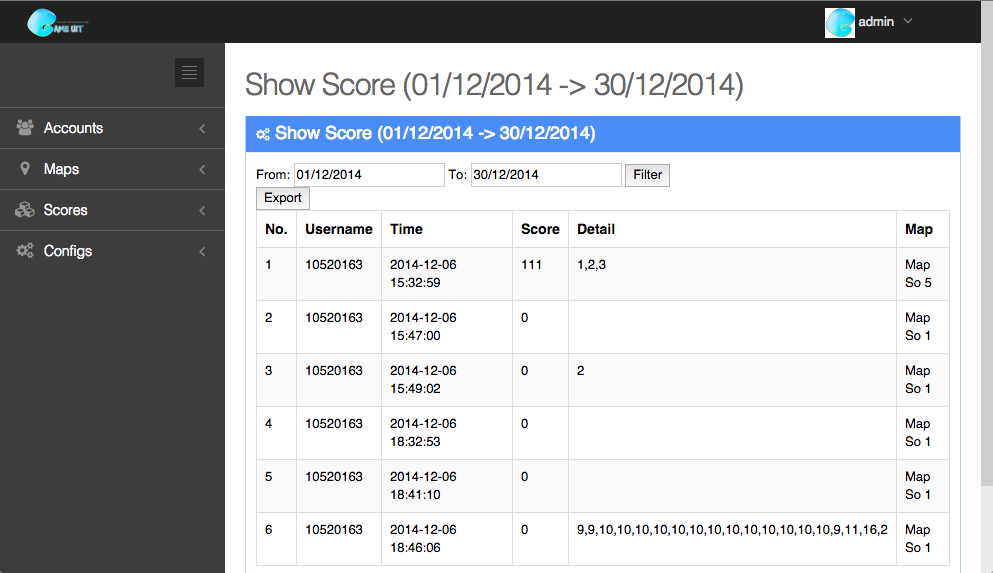
<http://widocom.com/projects/trafficgame/backend>

Tài khoản khả dụng:

* username: admin
* password: 123456

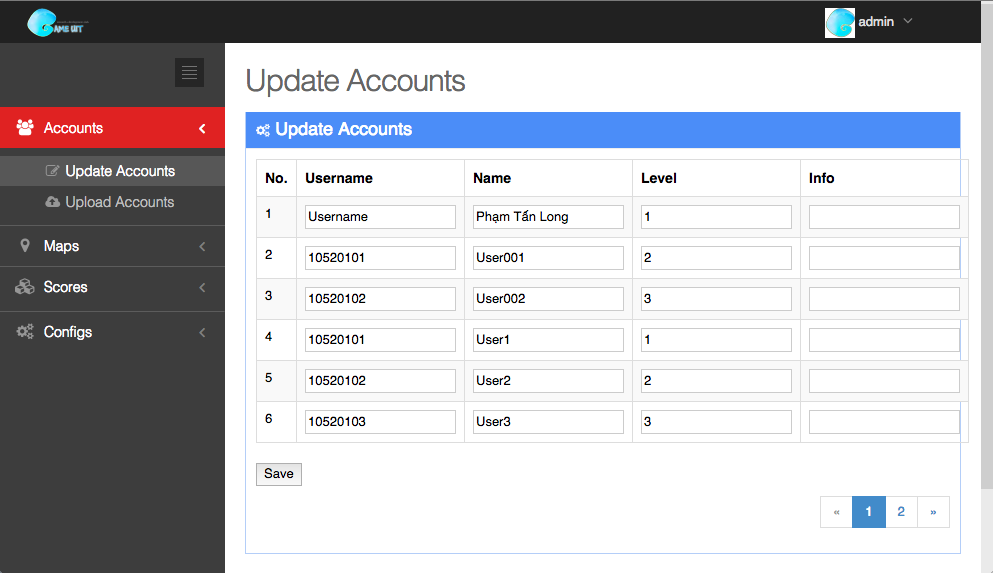
### Giao diện

* Home / Scores



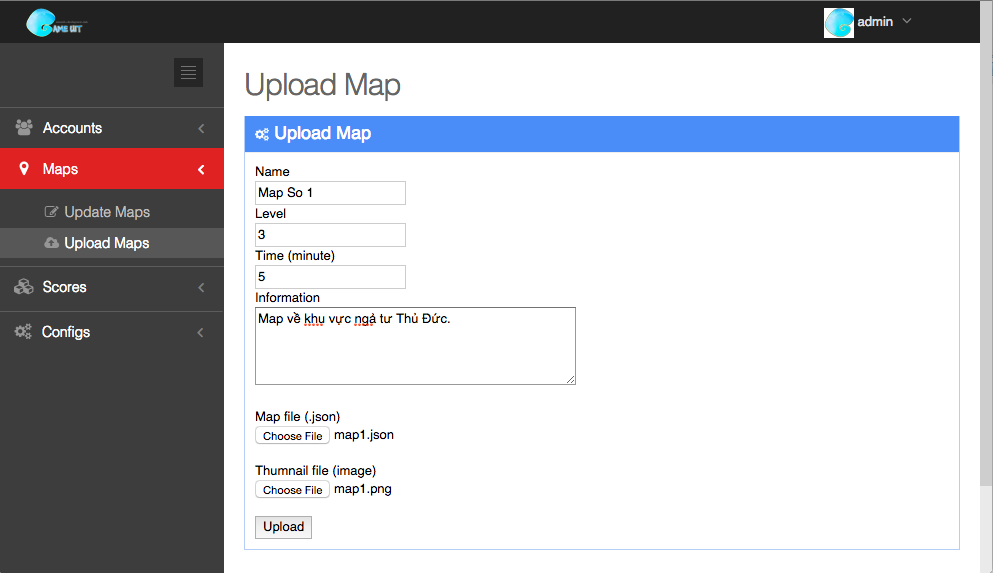
Hình . Giao diện Home / Scores trong CMS

* Update accounts



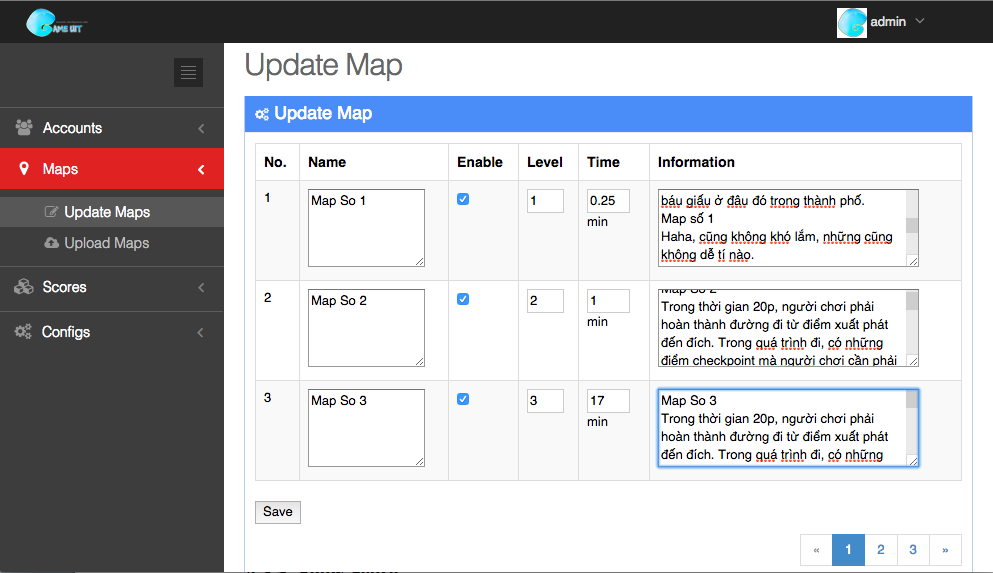
Hình . Giao diện Update accounts trong CMS

* Upload map



Hình . Giao diện Upload map trong CMS

* Update maps



Hình . Giao diện Update maps trong CMS

# KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

## Kết luận

Từ những cấu trúc, mô hình và phương pháp tiếp cận nêu trên, nhóm đã xây dựng thành công hệ thống ứng dụng học luật giao thông. Trong đó, Traffic Game đã có thể phát hiện được 30 tình huống vi phạm giao thông thường gặp và cung cấp thông tin về các lỗi vi phạm đó.

Nhóm đã cài đặt hệ thống tại địa chỉ: <http://widocom.com/projects/trafficgame> với giao diện đơn giản nhằm mục đích thử nghiệm hoạt động của hệ thống.

## Hướng phát triển

Trong thời gian tới, hệ thống sẽ có thể cung cấp thêm các tính năng:

* Hỗ trợ lựa chọn nhiều phương tiện giao thông
* Hỗ trợ thêm nhiều kiến trúc hạ tầng phức tạp hơn
* Xây dựng hệ thống trí tuệ nhân tạo tốt hơn cho các xe tự động
* Nâng cấp giao diện
* Cải thiện tốc độ Traffic Game

# TÀI LIỆU THAM KHẢO

1. A. Sahni and D. Segleau, NoSQL and SQL Introspective. California: Oracle, 2013
2. K. Chodorow and M. Dirolf, MongoDB The Definitive Guide. California: O'Reilly, 2010.
3. R. Saunier, Getting Started with Laravel 4. Birmingham, UK: Packt, 2014.
4. S. Blackman, Beginning 3D Game Development with Unity 4, 2nd ed. New York: Apress, 2013.
5. Tình hình tai nạn giao thông năm 2014 [Online]. Available: <http://antoangiaothong.gov.vn/den-xanh/nam-2014-ca-nuoc-giam-hon-4-ngan-vu-tngt-62911.html>
6. Luật giao thông đường bộ [Online]. Available: <http://www.moj.gov.vn/vbpq/_layouts/print.aspx?id=12345>
7. Quy định xử phạt vi phạm hành chính trong lĩnh vực giao thông [Online]. Available: <http://www.moj.gov.vn/vbpq/_layouts/print.aspx?id=25264>
8. Nghị định số 171/2013/NĐ-CP của Chính phủ [Online]. Available: <http://moj.gov.vn/vbpq/Lists/Vn%20bn%20php%20lut/Attachments/28758/VanBanGoc_171.2013.ND.CP.pdf>