**ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

**KHOA KỸ THUẬT PHẦN MỀM**

**PHẠM TẤN LONG**

**KHÓA LUẬN TỐT NGHIỆP**

**NGHIÊN CỨU UNITY VÀ XÂY DỰNG ỨNG DỤNG HỌC LUẬT GIAO THÔNG**

**KỸ SƯ NGÀNH KỸ THUẬT PHẦN MỀM**

**TP. HỒ CHÍ MINH, 2014**

**ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

**KHOA KỸ THUẬT PHẦN MỀM**

**PHẠM TẤN LONG – 10520163**

**KHÓA LUẬN TỐT NGHIỆP**

**NGHIÊN CỨU UNITY VÀ XÂY DỰNG ỨNG DỤNG HỌC LUẬT GIAO THÔNG**

**KỸ SƯ NGÀNH KỸ THUẬT PHẦN MỀM**

**GIẢNG VIÊN HƯỚNG DẪN**

**ThS. NGUYỄN TRÁC THỨC**

**TP. HỒ CHÍ MINH, 2014**

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.

MỤC LỤC

Chương 1. TỔNG QUAN 5

1.1. Bối cảnh nghiên cứu 5

1.2. Động lực nghiên cứu 5

1.3. Giới hạn đề tài 6

Chương 2. CƠ SỞ CÔNG NGHỆ 8

2.1. Game Engine 8

2.1.1. Các loại Game Engine 8

2.1.1.1. Roll-Your-Own 8

2.1.1.2. Most-Ready 9

2.1.1.3. Point-And-Click 9

2.1.2. Một số Game Engine trên thị trường 9

2.1.2.1. CryEngine 9

2.1.2.2. OGRE Engine 10

2.1.2.3. Panda3D 10

2.1.2.4. UnrealEngine 10

2.2. Unity Engine 11

2.2.1. Giới thiệu 11

2.2.2. Các tính năng tiêu biểu 12

2.2.3. Ngôn ngữ lập trình 15

2.2.4. Mono 16

2.2.4.1. Kiến trúc 17

2.2.4.2. Các thành phần 17

2.2.4.3. Các tính năng 18

2.2.5. Kiến trúc Component 18

2.2.6. Những Component trong Unity 21

2.2.6.1. Nhóm Mesh 21

2.2.6.2. Nhóm Effect 22

2.2.6.3. Nhóm Renderring 23

2.2.6.4. Nhóm Physics 24

2.2.6.5. Nhóm Audio 25

2.2.6.6. Nhóm Terrain 26

2.2.7. Script 26

2.2.7.1. Behaviour 26

2.2.7.2. MonoBehaviour 27

2.2.8. Physics 29

2.2.8.1. NVIDIA PhysX 29

2.2.8.2. Vật lý trong Unity 31

2.2.8.3. Các Component vật lý trong Unity 34

2.3. Laravel Frameword 39

2.3.1. Giới thiệu 39

2.3.2. Một số ưu điểm 40

2.3.3. Request LifeCycle 41

2.3.4. Router 43

2.3.5. View 44

2.3.6. Controller 44

2.3.7. Filter 45

2.3.8. Chủ đề cấp độ 3 47

2.3.9. Chủ đề cấp độ 3 47

2.3.9.1. Chủ đề cấp độ 4 47

Chương 3. TÊN CHƯƠNG 2 48

3.1. Chủ đề cấp độ 2 48

3.1.1. Chủ đề cấp độ 3 48

3.1.1.1. Chủ đề cấp độ 4 48

3.2. Chủ đề cấp độ 2 48

3.2.1. Chủ đề cấp độ 3 48

Chương 4. TÊN CHƯƠNG 3 49

4.1. Chủ đề cấp độ 2 49

4.1.1. Chủ đề cấp độ 3 49

4.1.1.1. Chủ đề cấp độ 4 49

4.2. Chủ đề cấp độ 2 49

DANH MỤC HÌNH VẼ

Hình 2.1 Kiến trúc thực thi Mono 17

Hình 2.2 Thực thể theo kiến trúc component 19

Hình 2.3 Các hệ thống cho các component 19

Hình 2.4 Nhóm Mesh 21

Hình 2.5 Nhóm Effect 22

Hình 2.6 Nhóm Rendering 23

Hình 2.7 Nhóm Physics 24

Hình 2.8 Nhóm Terrain 26

Hình 2.9 Các loại Physic Material mặc định trong Unity 36

Hình 2.10 Các thuộc tính chung của các Physic Material 36

Hình 2.11 Constant Force áp dụng cho tên lửa đẩy 38

Hình 2.12 Laravel Framework 39

Hình 2.13 Tỷ lệ sử dụng các framework PHP năm 2013 40

Hình 2.14 Vòng đời một Request trong Laravel 41

Hình 2.15 Bước Loading trong Request 42

Hình 2.16 Bước Running trong Request 42

Hình 2.17 Bước Running trong Request 43

Hình 2.18: Tên hình 1 47

DANH MỤC BẢNG

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

Bảng 2.2 Các thuộc tính của Behaviour 26

Bảng 2.3 Các phương thức của Behaviour 27

Bảng 2.4 Các phương thức của MonoBehaviour 27

Bảng 2.5 Các phương thức cần thừa kế của MonoBehaviour 29

Bảng 2.6 So sánh tốc độ vật lý của CPU và GPU 30

Bảng 2.7 Các thuộc tính của Rigidbody 34

Bảng 2.8 Các loại Collider 35

Bảng 2.9 Các loại Joint 37

Bảng 2.10: Tên bảng 1 47

Bảng 3.1: Tên bảng 1 48

DANH MỤC TỪ VIẾT TẮT

TÓM TẮT KHÓA LUẬN

Với sự phát triển mạnh mẽ của đất nước trong những năm gần đây, ngoài những thay đổi tích cực về đời sống con người, Việt Nam cũng đang phải đối mặt với nhiều vấn đề lớn, trong đó có vấn đề giao thông.

Với mong muốn được sử dụng công nghệ thông tin để góp phần làm thay đổi đất nước, khoá luận đã được hình thành ý tưởng như vậy.

Từ kết quả nghiên cứu, tìm hiểu về Unity Engine, nhóm đã sử dụng công cụ này để xây dựng nên phần mềm học luật giao thông nhằm hỗ trợ thêm một công cụ học tập trực quan sinh động hơn, nhằm nâng cao ý thức người dân khi tham gia giao thông, từ đó hạn chế tai nạn giao thông, cải thiện tình trạng giao thông Việt Nam.

Trước hết, quá trình nghiên cứu Unity Engine bao gồm tìm hiểu về các vấn đề cơ bản: game engine là gì, các loại game engine, các engine phổ biến, so sánh các engine đó, và lý do lựa chọn Unity. Tiếp theo là các vấn đề chuyên sâu của engine này, bao gồm: kiến trúc component, so sánh với kiến trúc OOP, particle system, animation system, physic system và terrain.

Cuối cùng là việc áp dụng Unity Engine vào việc xây dựng phần mềm học luật giao thông chạy trên nền web, thông qua Unity Web Player. Bên cạnh việc sử dụng Unity, để có thể hoàn thành hệ thống, nhóm đã tìm hiểu thêm về Laravel Framework để xây dựng nên trang quản trị nội dung cho hệ thống.

Kết quả cuối cùng, khóa luận tạo ra một hệ thống bao gồm 3 sản phẩm: Map Editor, Game và CMS.

Map Editor là một phần mềm độc lập, cho phép người quản trị xây dựng tùy biến một khu vực giao thông có hầu hết các thành phần giao thông trong thực tế gồm: làn đường, vỉa hè, vạch kẻ đường, biển báo, đèn giao thông, trạm xe buýt... Bên cạnh là các thành phần phối cảnh: nhà cửa, cây cối, người qua đường, các xe tham gia giao thông .v.v.

Game là phần mềm chạy trên web, thông qua Unity Web Player. Đây là nơi người chơi có thể tham gia chơi và học luật giao thông.

CMS là trang web cho phép nhà quản trị quản lý nội dung của game, quản lý người chơi, xem kết quả .v.v.

MỞ ĐẦU

Theo thống kê của Uỷ ban An toàn giao thông quốc gia, trong chín tháng đầu năm 2014, cả nước đã xảy ra 18.697 vụ tai nạn giao thông, làm chết 6.758, làm bị thương 17.835 người. Cũng trong chín tháng qua, lực lượng cảnh sát giao thông toàn quốc đã kiểm tra, lập biên bản hơn 3,37 triệu trường hợp vi phạm trật tự an toàn giao thông đường bộ; phạt tiền 1.943 tỷ đồng; tạm giữ 441.020 phương tiện các loại; tước 275.153 giấy phép lái xe. Với số liệu trên đã cho thấy, bên cạnh những lý do về cơ sở hạ tầng thì ý thức, kiến thức của người tham gia giao thông cũng là nguyên nhân quan trọng.

Trên thực tế, đã có rất nhiều nỗ lực tuyên truyền của nhà nước trên báo chí, truyền hình hay trực tiếp tại các trường học, tổ dân phố. Bên cạnh đó cũng đã có nhiều ứng dụng học luật giao thông, kiểm tra luật giao thông trên PC, mobile để mọi người dễ dàng tiếp cận. Tuy nhiên, tất cả các biện pháp này chỉ cung cấp thông tin về luật giao thông, các quy định xử phạt, v.v.. Nhược điểm của những biện pháp này là chỉ mang tính đối phó, thiếu tính trực quan, không thu hút người dùng và nội dung không được cập nhật, đổi mới, do đó không thể đem lại những thay đổi tích cực trong ý thức cũng như kiến thức người dân.

Như vậy, cần có một hệ thống giáo dục luật giao thông mới trực quan, sinh động hơn, cho phép người học tham gia trực tiếp vào các tình huống giao thông, tương tác như trong thực tế, đồng thời có thể dễ dàng thay đổi, cập nhật nội dung học tập, và hơn nữa là đa nền tảng để có thể đến được với mọi đối tượng người học.

Một game 3D là một ý tưởng tuyệt vời để thực hiện điều này. Các tình huống giao thông sẽ trở nên thực tế hơn, sinh động hơn trong môi trường 3D. Người tham gia có thể tương tác trực tiếp, tạo ra các tình huống vi phạm và bị xử phạt. Những kiến thức giao thông sẽ dễ dàng tiếp cận hơn khi việc học trở thành một trò chơi.

Unity đang là một game engine mạnh, phổ biến và có nhiều ưu điểm hơn hẳn các engine khác, đặc biệt khi Unity trở nên miễn phí cho cả các sản phẩm thương mại kể từ phiên bản 4.3. Người dùng cá nhân hoặc các tổ chức phi lợi nhuận có thể download, sử dụng và phát hành sản phẩm hoàn toàn miễn phí.

Với tất cả những khảo sát đó, em xin lựa chọn đề tài "Nghiên cứu Unity và xây dựng ứng dụng học luật giao thông".

# TỔNG QUAN

## Bối cảnh nghiên cứu

Theo thống kê của Bộ Giao thông Vận tải, tính đến tháng 3/2014, số lượng xe máy được sử dụng trên địa bàn cả nước lên đến xấp xỉ 37 triệu chiếc. Nhiều năm qua, không chỉ thành phố mà kể cả địa bàn nông thôn, số hộ gia đình sử dụng xe máy được "phủ sóng” gần như đạt tỷ lệ tuyệt đối. Với dân số hiện tại 90 triệu người, cứ khoảng 2,4 người thì có 1 người có xe máy. Tại nhiều nơi, nhất là khu vực đô thị, tỷ lệ người lao động với xe máy đạt hệ số 1:1. Xe máy đang chiếm hơn 85% tổng số phương tiện giao thông hiện nay.

Bình quân mỗi năm thị trường Việt Nam "khai sinh” thêm hơn 3 triệu chiếc xe máy. Với đà tăng tiến như vậy, đến 2020, tổng số lượng xe máy lưu hành trên thị trường có khả năng đạt tới 60 triệu chiếc. Là phương tiện giao thông thiết yếu của số đông người dân, nhưng lượng xe máy tăng nhanh sẽ gây ra nhiều hệ lụy rất đáng lo ngại.

Theo Uỷ ban An toàn giao thông quốc gia, mỗi năm, cả nước mất 30.000 tỷ đồng do ùn tắc giao thông. Một số liệu nghiên cứu khác cho thấy thời gian ùn tắc giao thông trung bình tại Hà Nội và TP.HCM là 45 phút/ngày, tương đương 15 giờ/tháng và 180 giờ/năm. Với thời gian lãng phí do ùn tắc gây ra, tổng thiệt hại tương ứng chỉ tính riêng Hà Nội và TP.HCM đã lên tới 18.800 tỉ đồng/năm. Thêm vào đó, mỗi năm số tiền chi ra cho khắc phục tai nạn giao thông tại Việt Nam lên tới 2,5 tỉ USD, tương đương 50.000 tỉ đồng, bằng 30% ngân sách chi cho giáo dục. 70% trong số các vụ tai nạn đó có thủ phạm chính là xe máy.

Với những số liệu đó, vấn đề nâng cao kiến thức cho người điều khiển xe gắn máy tham gia giao thông là một vấn đề cấp bách.

## Động lực nghiên cứu

Tuyên truyền về kiến thức giao thông là hoạt động cần được đẩy mạnh trong bối cảnh trên, tuy nhiên câu hỏi là phương pháp nào là hiệu quả.

Các hoạt động tuyên truyền phổ biến hiện nay như: áp phích, băng rôn, chương trình truyền hình, cuộc thi trong trường học, tuyên truyền thông qua loa phát thanh ở từng khu phố… Nhược điểm có thể thấy của các hoạt động này là chỉ có thể đề cập đến lý thuyết hay trình bày hiện trạng nhằm răng đe. Người nghe có thể thay đổi ý thức, kiến thức, nhưng chưa có cơ hội để có thể thực hành và kiểm tra trong các điều kiện giống với thực tế. Chính những thiếu sót đó dẫn đến người dân lúng túng khi xử lý các tình huống thực tế dẫn đến gây ùn tắt giao thông hoặc tệ hơn là gây tai nạn giao thông.

Điều đó đặt ra yêu cầu cần có một giải pháp tuyên truyền khác hiệu quả hơn. Giải pháp mới đòi hỏi phải trực quan, giống với thực tế, cho phép người chơi tương tác như đang điều khiển phương tiện tham gia giao thông như trong thực tế và hơn nữa, các tình huống vi phạm phải được pháp hiện kịp thời và cảnh báo để người tham gia có thể ghi nhớ.

## Giới hạn đề tài

Nghiên cứu tập trung vào kiến thức game 3D và môi trường phát triển Unity. Bên cạnh đó nghiên cứu cũng bao hàm các kiến thức về Laravel Frameword và cơ sở dữ liệu MongoDB.

Mục tiêu nghiên cứu nhằm xây dựng ứng dụng học luật giao thông chạy trên nền web và hệ thống hỗ trợ quản trị nội dung CMS (Content Management System).

Mục tiêu cụ thể bao gồm:

* Nghiên cứu các kiến thức 3D và môi trường phát triển Unity.
* Nghiên cứu Laravel Frameword
* Nghiên cứu cơ sở dữ liệu MongoDB
* Xây dụng ứng dụng học luật giao thông với các tính năng sau:
  + User
    - Đăng nhập
    - Chơi game
    - Đăng tải điểm
    - Xem lịch sử
  + Admin
    - Quản lý nội dung ứng dụng
    - Thiết kế nội dung ứng dụng
    - Quản lý tài khoản người dùng
    - Báo cáo

# CƠ SỞ CÔNG NGHỆ

## Game Engine

Một game engine là một phần mềm được viết để thiết kế và phát triển video game, hiểu đơn giản nó là loại phần mềm trung gian kết nối tương tác của nhiều ứng dụng trong cùng 1 hệ thống với nhau. 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.

### Các loại Game Engine

Game engine có nhiều loại khác nhau chia thành từng mức chuyên biệt khác nhau. Để dễ hiểu, tạm chia ra thành 3 loại: roll-your-own, mostly-ready, và point-and-click engine.

#### Roll-Your-Own

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. Điều này có nghĩa họ dùng những ứng dụng giao diện được công bố rộng rãi, các API như XNA, DirectX, OpenGL, các API và SDL của Windows và Linux. Thêm vào đó, 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 của Roll-your-own engine được 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.

#### Most-Ready

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 cả lô tool mạnh cho bạn tha hồ sử dụng. Những engine tầng này khá phong phú từ OGRE và Genesis3D (open-source), rẻ tiền như Torge hoặc đại gia 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 chút so với roll-your-own engine. Cũng đúng thôi 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 đâu phải khách hàng nào cũng giống nhau và phát triển những game tương tự nhau.

#### Point-And-Click

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 bạn 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, nó cần ít nhất.

### Một số Game Engine trên thị trường

#### CryEngine

CryEngine 3 là game engine được Crytek phát triển và phá hành vào 4 tháng 10 năm 2009. Là một bướt phát triển tiếp nối theo phiên bản CryEngine2 khá thành công của hãng. Theo Crytek, CryEngine 3 được tao ra với tham vọng trở thành công cụ phát triển game tất cả trong một, ứng dụng trên cả Microsoft windows, PlayStation 3 và Xbox 360. Riêng đối với nền PC, CryEngine 3 sẽ hỗ trợ tốt cả DirectX 9,10,11.

Điểm nổi trội 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. Những tựa game như Far Cry, Crysis, Crysis Warhead, Crysis 2 và Aion: Tower of Eternity là những các tên tiêu biểu sử dụng Cry Engine.

#### OGRE Engine

OGRE (Object-Oriented Graphics Rendering Engine) là một Engine dựng hình 3D linh hoạt và tập trung vào khung cảnh (hơn là một công của tạo game) được viết bằng C++, được thiết kế để mang lại sự đơn giản và trực quan hơn cho nhà phát triển trong công việc tạo ra những ứng dụng có khai thác phần cứng tăng tốc đồ họa 3D. Bộ thư viện của OGRE trựu tượng hóa các hóa các thư viện mức hệ thống như Direct3D và OpenGL để cung cấp một giao diện lập trình dựa trên các đối tượng thể giới thực và các lớp cấp cao.

#### Panda3D

Là một Engine Game bao gồm các chức năng về đồ họa, âm thanh, quản lý nhập xuất, hệ thống phát hiện va chạm và những chức năng khác phù hợp với nhiệm vụ phát triển Game 3D.

Ngôn ngữ kịch bản dùng trong Panda3D là python. 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 này giúp lập trình tận dụng được ưu điểm của python như hỗ trợ lập trình nhanh và quản lý bộ nhớ tốt hơn, nhưng vẫn giữ lại được ưu điểm về khả năng biên dịch của C++ trong nhân của engine. Cụ thể, Panda3D tích hợp bộ dọn rác tự đọng của Python và công cụ quản lý cấu trúc tự động.

#### UnrealEngine

Unreal engine là Engine Game được phát triển bởi Epic Games. Từ lần ra mắt đầu tiên vào năm 1998 với game Unreal, nó đã đặt nền móng cho những game sau này như Unreal Tournament, Deus Ex, Turok, Tom.

Do được viết bằng C++ nên Unreal Engine có tính cơ động cao và đồng thời là một công cụ được rất nhiều nhà phát triển game ngày nay sử dụng trên hầu hết các nền tảng hiện có như Windows, Linux, Android, iOS, Mac OS, trên máy tính cá nhân cũng như trên rất nhiều các thiết bị chơi game chuyên dụng như Dream cast, Xbox, PS.

## Unity Engine

### Giới thiệu

Unity Engine được phát triển bởi Unity Technologies – là một gói công cụ tích hợp dùng để xây dựng Game 3D hoặc các nội dung có tính tương tác khác như mô hình kiến trúc hay hoạt hình 3D thời gian thực. Unity Engine có thể chạy trên hệ điều hành Windows và Mas OS X. Sản phẩm tạo ra từ Unity Engine có thể chạy được trên các nền Windows, Mac, Linux , Wii,iOS,Android. Bên cạnh đó, Unity Engine có khả năng phát triên Game nền Web hỗ trợ cả Mac và Windows.

Unity hay Unity 3D chắc chắn không phải engine đỉnh cao nhất về mặt đồ họa. Về mặt này, Cry Engine vẫn đang dẫn đầu với ưu thế đồ họa 3D cực kỳ chân thực. Có thể cảm nhận rõ ràng điều này qua chất lượng hình ảnh các tựa game gần đây sử dụng Cry Engine như Far Cry hay Crysis 3. Tuy nhiên, những engine khủng như Cry hay Unreal rất kén chọn và yêu cầu cấu hình cũng khủng không kém. Hơn nữa cái giá để được cấp phép sử dụng Cry Engine chắc chắn cũng không dễ chịu chút nào.

Tuy đồ họa không thể so sánh với Cry Engine nhưng chất lượng hình ảnh Unity 3D mang lại cũng quá đủ để thỏa mãn game thủ hiện nay. Hơn nữa, Unity 3D còn có những ưu điểm mà không phải engine game nào cũng có.

* Hỗ trợ đa nền tảng

Một trong các thế mạnh của Unity 3D chính là khả năng hỗ trợ gần như toàn bộ các nền tảng hiện có bao gồm: PlayStation 3, Xbox 360, Wii U, iOS, Android, Windows, Blackberry 10, OS X, Linux, trình duyệt Web và cả Flash. Nói cách khác, chỉ với một gói engine, các studio có thể làm game cho bất kỳ hệ điều hành nào và dễ dàng convert chúng sang những hệ điều hành khác nhau. Đồng thời, đây cũng là giải pháp cho các game online đa nền tảng – có thể chơi đồng thời trên nhiều hệ điều hành, phần cứng khác nhau như Web, PC, Mobile, Tablet….

* Dễ sử dụng

Unity 3D được xây dựng trong một môi trường phát triển tích hợp, cung cấp một hệ thống toàn diện cho các lập trình viên, từ soạn thảo mã nguồn, xây dựng công cụ tự động hóa đến trình sửa lỗi. Do được hướng đến đồng thời cả lập trình viên không chuyên và studio chuyên nghiệp, nên Unity 3D khá dễ sử dụng. Hơn nữa, đây là một trong những engine phổ biến nhất trên thế giới, người dùng có thể dễ dàng tìm kiếm kinh nghiệm sử dụng trên các forum công nghệ.

* Tính kinh tế cao

Unity Technologies hiện cung cấp bản miễn phí engine Unity 3D cho người dùng cá nhân và các doanh nghiệp có doanh thu dưới 100.000 USD/năm. Với bản Pro, người dùng phải trả 1.500 USD/năm – một con số rất khiêm tốn so với những gì engine này mang lại.

Có thể thấy, tuy không phải engine “khủng” nhất, nhưng nếu xét toàn diện, Unity 3D đích thực là một trong những engine game đa nền tảng tốt nhất hiện nay.

### Các tính năng tiêu biểu

Unity Engine có thể sử dụng C#, JavaScript hoặc Boo để phát triển và thiết kế theo hướng component. Là một môi trường phát triển tích hợp mạnh mẽ, có khả năng xem trước kết quả, sản phẩm có thể triển khai trên nhiều nền tảng:

* Sử dụng được trên Windows và Mac OSX
* Ứng dụng chạy trên nền web (thông qua plugin Unity Web Player) trên Windows và Mac OSX
* Chức năng dashboard widget trên Mac OSX
* Ứng dụng chạy trên Nintendo Wii, iOS, Android, WP 8, PS

Unity cho phép nạp các model từ 3ds Max, Maya, Blender, Mode, Zbrush, Cineme 4D, Cheetah3D.

Engine đồ họa sử dung Direct3D (Windows), OpenGL, OpenGLES và prorietary.

Về mặt dựng hình, Unity engine có những đặc điểm nổi bật sau:

* Deferred Lightning

Cho khả năng thể hiện ánh sáng vào bóng đổ cực kỳ trung thực, mọi đối tượng đều thể hiện được hiệu ứng ánh sáng cũng như bóng tối, mức tác động ở cấp từng điểm ảnh.

* 100 Kiểu tạo bóng tích hợp

Từ đơn giản (Diffuse, Glossy,…) đến phức tạp (self Illuminated Bumped Specular….).

* Surface shaders

Bộ phát sinh mã tự động nhằm đem lại sự đơn giản trong việc lập trình Vertex Shader và Pixel Shader. Chúng ta đơn giản chỉ cần viết mã Cg và Unity sẽ biên dịch nó thành mã dựng hình, thậm chí cũng có thể biên dịch sang GLSL để chạy trên các thiết bị đi động.

* Scalability

Mở rộng khả năng tương thích với phần cứng. Unity sẽ phân tích phần cứng hiện có để bảo đảm người dùng có thể trải nghiệm được những hiệu ứng đồ họa tốt nhất.

* Full Screen Post-Processing Effects

Unity tích hợp rất nhiều hiệu ứng hậu xủa lý hình ảnh toàn màn hình như: Sun shaft, hight quality depth-of-field, lens effects, chromatic abberation, curve-driven color correction…

Nhằm tăng hiệu năng, Unity áp dụng một số kỹ thuật như sau:

* Batching

Nhằm hạn chế số lệnh gọi hàm vẽ, Unity tự động kết hợp các đối tượng hình học lại thành các tổ hợp, cách làm này sẽ giúp giảm đáng kể chi phí nhưng đồng thời vẫn giữ lại được toàn bộ hiệu ứng hình ảnh.

* Rewritten Renderring Engine

Tăng sức mạnh và tính linh động của bộ kết xuất hình ảnh lên đến 50% so với các phiên bản trước.

* Occlusion Culling

Giải pháp giúp giảm chi phí cho các sản phẩm chạy trên nền console, di động và web bằng cách giảm số lượng đối tượng cần kết xuất xuống mức tối thiểu.

* GLSL Optimizer

Mặc dù OpenGL ES cho phép sử dụng hiệu ứng đổ bóng trên nền di động song nó vẫn tỏ ra quá nặng so với khả năng của thiết bị. Vì vậy, Unity được trang bị bộ tối ưu GLSL Shader với khả năng tăng hiệu năng lên 2-3 lần.

Về phần chiếu sáng, Unity có những đặc điểm khá ưu việt:

* Deferred Renderer

Kỹ thuật chiếu sáng hiện đại cho khả năng bao phủ toàn bản đồ, mọi nguồn sáng đều tạo ra hiệu ứng chiếu sáng thậm chí là tia sáng phát ra từ một vụ nổ.

* Realtime Shadows

Hệ thống tạo bóng đổ tiên tiến có khả năng tạo bóng từ mọi nguồn sáng, đông thời sử dụng ít tài nguyên phần cứng.

* Screen Space Ambient Occlusin (SSAO)

Unity pro sử dụng công nghệ SSAO như là một công cụ tạo hiệu ứng ảnh chuẩn. Đơn giản chỉ cần thêm thành phần này vào bất kỳ góc nhìn nào và chúng ta sẽ đạt được hiệu quả tuyệt vời.

* Sun shafts & Lens Effects

Unity Pro cũng hỗ trợ thêm nhiều hiệu ứng tia sáng mặt trời, đồng thời mô phỏng hiểu ứng phản chiếu thấu kính nội và hiệu ứng lóe sáng nhằm mang lại một trải nghiệm mang tính nghê thuật cao.

* Lighting mapping

Là công nghệ duy nhất cho phếp chúng ta điều khiển không gian toàn bộ game một cách chính xác. Unity tích hợp công cụ lightmapper tốt nhất hiện nay là Beast, giúp tạo ra không gian game hoàn hảo nhất.

### Ngôn ngữ lập trình

Với Unity, ta có thể sử dụng một hoặc kết hợp 3 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#.

|  |  |
| --- | --- |
| 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 tốt độ và hiệu năng, tuy nhiên ta có một vài sự khác biệt giữa chúng.

* C#

Ưu điểm:

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

Nhược điểm: Chưa thấy

* JavaScript

Ưu điểm:

* Dễ học, dễ hiểu
* Tốc độ chậm hơn C# trong một vài trường hợp

Nhược điểm:

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

Ưu điểm:

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

Nhược điểm:

* Không có nhiều người sử dụng

### Mono

Mono là một phiên bản mã nguồn mở đa nền tảng, hiện thực từ 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 Engine 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 thực thi Mono

#### Các thành phần

Có các thành phần tạo nên Mono:

* C# Compiler

Mono C# Compiler hỗ trợ tính năng đầy đủ 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 chức năng 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 hơn các lớp cơ bản ở trên. Chúng cung cấp các hàm 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
  + BSD – 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 giấy phép OSI approved license.

### Kiến trúc Component

Component Architecture là hướng lập trình hướng dữ liệu. Trong đó một đối tượng thường được gọi là entity (thực thể). Tất cả các entity đều giống nhau không chứa bất kỳ đặc tính riêng nào. Một entity có thể gắn thêm các component. Mỗi thực thể sẽ mang những thuộc tính và hành vi của component được thêm vào.



Hình . Thực thể theo kiến trúc component

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

Mỗi component sẽ độc lập với nhau và chỉ ảnh hướng tới nhau theo từng liên kết nhất định. Mỗi component và behavior sẽ có một hệ thống xử lý riêng. Chạy trên những nhánh độc lập với nhau.

Mỗi component thông thường sẽ có hai thành phần là:

* Attributes (data): danh sách các dữ liệu cần thiết.
* Behaviours (logic): các hàm xử lý dữ liệu



Hình . Các hệ thống cho các component

Trong Component Architecture hầu như không tạo ra một đối tượng nhất định giống như OOP mà tất cả các Object trong component đều là thực thể. Không có kiến trúc theo cây thừa kế trong Component Architecture. Tốc độ xử lý theo kiến trúc Component Architecture nhanh hơn OOP rất nhiều.

* Ưu điểm
* Tốc độ xử lý cực nhanh vì mỗi component sẽ chạy trên một hệ thống system riêng biệt nhau
* Reuse code tốt. Một component có thể được sử dụng cho nhiều đối tượng khác nhau nhưng có chung một số đặc điểm về component đó. Giảm độ phức tạp của dự án.
* Dễ vận hành và bão trì game: mỗi component hoàn toàn gần như là độc lập với nhau nên dựa theo kiến trúc của component thì rất dễ thích nghi với thây đổi trong game, tránh và hạn chế ảnh hưởng tới các component còn lại.
* Dễ dàng cho làm việc với các dự án có nhiều người cùng tham gia.
* Nhược điểm
* Không thích hợp với các dự án nhỏ.
* Thời gian phát triển ban đầu lớn

### Những Component trong Unity

#### Nhóm Mesh



Hình . Nhóm Mesh

Dùng để hiển thị các model 3d vào Unity. Bao gồm các component:

* Mesh

Các model 3d được import từ asset

* Mesh Filter

Truyền một mesh từ asset vào cho Mesh Renderer hiển thị

* Text Mesh

Một mesh đặc biệt dùng để hiển thị text trong không gian 3D

* Mesh Render

Hiển thị mesh lên không gian 3D. 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



Hình . Nhóm Effect

Tạo ra các effect trong game như: khói, hơi nước, hiệu ứng khí quyển. 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 (đuôi) cho các đối tượng di chuyển trong không gian. Tương tự như Line Renderer sẽ nói ở sau.

* Line Renderer

Tạo ra các đường 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 (Light) 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 GameObject. 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



Hình . Nhóm Rendering

Tạo các thành phần có hiển thị trong không gian, hoặc trong giao diện người dùng. Bao gồm các component:

* Camera

Dùng để quay lại các GameObject có trong scene. Đây là component không thể thiếu trong game. Nếu một scene không có GameObject nào chứa component này, sẽ không có gì trong scene 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 Layer

Dùng để hiển thị Text 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 GameObject có trong scene.

* Skybox

Dùng để giả lập một không gian vô hạn bao quanh toàn bộ scene.

#### Nhóm Physics



Hình . 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 GameObject. Bao gồm:

* Rigidbody

Cho phép các GameObject có thể chịu tác động lực, bao gồm cả trọng lực trái đất, lực cản không khí… Đối với các đối tượng tĩnh, không cần phản ứng với các tác động của 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 vật ở mọi frame thay vì chỉ một frame nếu sử dụng hàm *Rigidbody.AddForce*. Được dùng nhiều cho các loại tên lửa bay, 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 GameObject, thì các GameObject sẽ di chuyển xuyên qua nhau. Bao gồm:

* + 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 chiếc xe

#### Nhóm Audio

Âm thanh trong game. Bao gồm các component:

* Audio Listener

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

* 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 sẽ càng to hơn.

#### Nhóm Terrain



Hình . Nhóm Terrain

Tạo ra địa hình cho game như mặt đất, đường, đồi núi, cây cối, biển, sông suối,...

### Script

Trong Unity, một file script định nghĩa một class cũng được xem là một Component khi class đó thừa kế thừ 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.

#### Behaviour

Đây là class cha của class MonoBehaviour. Class này gồm một số thành phần quan trọng sau.

Thuộc tính:

|  |  |
| --- | --- |
| Tên | Ý nghĩa |
| enabled | Nếu enabled thì behaviours sẽ được Update. |
| tag | Tag của game object, dùng để phân nhóm object. |
| name | Tên của game object, dùng để định danh object. |

Bảng . Các thuộc tính của Behaviour

Phương thức:

|  |  |
| --- | --- |
| Tên | Ý nghĩa |
| GetComponent | Trả lại component được gắn vào game object. |
| GetComponentInChildren | Trả lại component được gắn vào game object hoặc bất kỳ đối tượng con của game object. |
| SendMessage | Gọi phương thức của đối tượng theo tên tương ứng có trong bất kỳ MonoBehaviour của chính đối tượng đó. |

Bảng . Các phương thức của Behaviour

#### MonoBehaviour

Đây là class sẽ được hầu hết các class của người dùng thừa kế. Các thành phần quan trọng của class này bao gồm.

Phương thức:

|  |  |
| --- | --- |
| Tên | Ý nghĩa |
| Invoke | Gọi một 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 lại tất cả coroutines chạy trên phương thức có tên này. |
| StopAllCoroutines | Dừng lại tất cả coroutines chạy trên phương thức có tên này. |

Bảng . Các phương thức của MonoBehaviour

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

|  |  |
| --- | --- |
| Tên | Ý nghĩa |
| Update | Sẽ được gọi ở mỗi frame, nếu MonoBehaviour được kích hoạt. |
| LateUpdate | Sẽ được gọi ở mỗi frame, nếu MonoBehaviour được kích hoạt, gọi sau hàm Update. |
| FixedUpdate | Chức năng này được gọi là 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 khi đưa chuột đi qua GUIElement hoặc Collider. |
| OnMouseOver | Sẽ được gọi mỗi frame khi chuột nằm trên GUIElement hoặc Collider. |
| OnMouseExit | Sẽ được gọi 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 | Được gọi khi thả chuột. |
| Tên | Ý nghĩa |
| OnMouseDrag | Được gọi khi click lên GUIElement hoặc Collider và giữ trạng thái đó rồi di chuyển chuột. |
| OnTriggerEnter | Được gọi khi collider khác 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 khi collider khác không chạm vào trigger nữa. |
| OnCollisionEnter | Được gọi 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 khi collider/rigidbody khác không còn va chạm với collider/rigidbody này. |

Bảng . Các phương thức cần thừa kế của MonoBehaviour

### Physics

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. Vậy engine NVIDIA PhysX là gì?

#### NVIDIA PhysX

*Vật lý trong game là gì?*

Vật lý trong game là tất cả những thứ liên quan đến sự chuyển động và tác động qua lại của các vật thể. Về cơ bản GPU dựng hình game còn những tính toán vật lý thì chuyển cho CPU xử lý. Với những thiết kế hiện nay ta cần phải hiểu rằng, PhysX không dựng hình đồ họa nó nằm ở mức thấp hơn trong pipeline. Sự chuyển động phức tạp của vật thể, dòng chảy... dựa trên sự tác động biến thiên của trong lực, tỉ trọng, gió..... Những thành phần này sẽ được tính toán từ đó sẽ vạch ra hướng chuyển động thực tế của vật thể, quần áo, dòng chảy....

Những tính toán vật lý chung thì được thực hiện bởi CPU, với những game hiện nay bộ vi xử lý của bạn càng ngày càng phải làm việc nhiều hơn do các game càng lúc càng tích hợp nhiều trí thông minh nhân tạo và sự suy luận logic. Với những tính toán vật lý phức tạp GPU có thể xử lý được nhanh hơn rất nhiều so với CPU.

*Vậy tại sao lại cần PhysX?*

Hiện nay game có tích hợp engine vật lý đã rất phổ biến tuy vậy có một số nhà phát triển game tự xây dựng engine vật lý cho riêng game của họ những engine này đều được xử lý thông qua CPU, có thể kể tên engine vật lý Havok nổi tiếng hiện đã được Intel mua lại.

Có một thực tế là hiện nay chỉ duy nhất engine vật lý PhysX của NVIDIA là có thể vừa được tăng tốc xử lý thông qua phần mềm (CPU xử lý) lại vừa được tăng tốc xử lý thông qua phần cứng (GPU xử lý) trong khi engine Havok chỉ có một cách xử lý duy nhất là thông qua phần mềm (CPU).

|  |  |  |
| --- | --- | --- |
| 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 vật lý của CPU và GPU

Một lý chính để chuyển việc tính toán PhysX từ CPU sang GPU là do sức mạnh to lớn trong tính toán vật lý của GPU 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. Chẳng hạn khả năng xử lý dấu chấm động của chip đồ họa GeForce GTX 280 cao hơn CPU 4 nhân Core 2 Quad đến hơn 9 lần, và chính yếu tố này sẽ giúp các nhà phát triển game tạo được nhiều hiệu ứng hình ảnh sống động hơn nữa. Chẳng hạn với một ví dụ đơn giản khi bạn làm nổ tung một chiếc xe hơn, nếu chiếc PC không hỗ trợ khả năng xử lý PhysX bằng GPU, thì chiếc xe đó chỉ bị vỡ thành 20 đến 30 mảnh nhỏ, tuy nhiên nếu PC có khả năng xử lý PhysX trên GPU thì chiếc xe đó sẽ bị vỡ ra thành hàng trăm mảnh nhưng tốc độ khung hình khi chơi game vẫn không bị giảm xuống quá nhiều.

Một vài tính năng PhysX có thể làm là:

* Vụ nổ sinh ra bụi và các mảnh vỡ vụn
* Nhân vật với những hình dạng chuyển động phức tạp và sự tương tác qua lại giữa các vật thể.
* Những loại vũ khí đặc biệt mới với những khả năng phi thường.
* Quần áo được xếp hoặc xé theo hướng lực tác động.
* Khói hoặc sương mù dầy đặc bao quanh vật thể trong lúc chuyển động.

Việc thêm vào PhysX cho các game không yêu cầu phải bổ sung thêm bất cứ những cải tiến đặc biệt nào cho những nhà sản xuất CPU hoặc GPU, công việc đơn giản là đã được lớp driver PhysX (PhysX driver's HAL - Hardware Abstraction Layer) đảm nhiệm, lớp driver PhysX này sẽ chuyển các mã lệnh PhysX phù hợp đến cho CPU, GPU hoặc PhysX PPU (card vật lý) xử lý.

Vì vậy với việc tích hợp hệ thống engine PhysX, Unity sẽ kế thừa được sức mạnh của hệ thống này để hổ trợ các xử lý vật lý của mình.

#### 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 là đối tượng mô phỏng có các thuộc tính vật lý, được dùng cho những thứ mà player có thể đẩy đi hoặc di chuyển trực tiếp bằng cách tạo lực tác dụng lên nó, bằng cách dùng script.

Nếu sử dụng phép chuyển dịch đối với Rigidbody không có thuộc tính động học (non-kinematic rigidbody) thì nó sẽ không thể va chạm chính xác với những đối tượng khác. Vì vậy chúng ta chỉ nên di chuyển Rigidbody bằng cách tác dụng lực hoặc mô-men xoắn (torque) vào nó. Ta còn có thể dùng Joints để kết hợp các rigidbodies với nhau, tạo ra các chuyển động phức tạp hơn, chẳng hạn như một cái cửa hay một chiếc cần cẩu có nhiều đoạn nối.

Ngoài ra ta còn có thể tạo ra các phương tiện giao thông như ô tô, máy bay, hoặc những robot chuyển động như thật bằng cách sử dụng các phép Joints và áp dụng các lực tác dụng lên chúng thông qua scripts.

Rigidbodies thường được sử dụng kết hợp với primitive colliders.

* Kinematic Rigidbody

Kinematic rigidbody không chịu ảnh hưởng của các lực, trọng lực và va chạm. Kinematic rigidbody được điều khiển một cách trực tiếp bằng phép biến đổi thuộc tính về vị trí và hướng của chúng hoặc tạo các hiệu ứng chuyển động cho chúng. Kinematic rigidbody có thể tương tác với những đối tượng không phải kinematic rigidbody.

Kinematic rigidbody tạo ra ma sát với những rigidbody nằm trên nó và có khả năng “đánh thức” một cách chính xác các rigidbodies mà chúng chạm vào (khi các rigidbody không hoạt động, chúng rơi vào trạng thái “ngủ”. Ở trạng thái này, các hoạt động như xác định va chạm hoặc mô phỏng sẽ không được thực hiện nữa. Do đó, làm tăng hiệu suất làm việc của CPU).

* Static Collider

Một Static Collider là một GameObject có khả năng va chạm nhưng không phải là một Rigidbody. Static collider được sử dụng như là những đối tượng có khả năng va chạm nhưng chỉ đứng yên và không bao giờ di chuyển.

Chúng ta không nên di chuyển một Static Collider, việc di chuyển một Static Collider sẽ tạo ra một tính toán lại trong hệ thống PhysX, điều đó sẽ gây tốn kém và làm giảm hiệu suất của game. Static Collider không có khả năng “đánh thức” các rigidbody khác khi chúng va chạm và khi Static Collider di chuyển sẽ không có ma sát với rigidbody khi chúng va chạm. Nên nếu sử dụng một đối tượng va chạm có thể di chuyển thì tốt nhất nên sử dụng Kinematic Rigidbidy.

* Character Controller

Chúng ta sử dụng Character Controller khi chúng ta muốn tạo ra một đối tượng nhân vật, thường là các nhân vật chính.

Những controller này không tuân theo quy luật vật lý vì như thế sẽ không thực (ví dụ trong game Doom ta có thể chạy 90 dặm một giờ và dừng lại chỉ trong một khoảnh khắc và tiếp tục chạy cực kỳ nhanh chóng). Thay vào đó, một bộ điều khiển nhân vật sẽ xử lý và thực hiện va chạm để đảm bảo nhân vật có thể trượt dốc, đi lên xuống cầu thang…

Character Controller không bị ảnh hưởng bởi lực nhưng nó có thể áp dụng lực lên các rigidbody. Thông thường, các nhân vật hình người đều được sử dụng Character Controller.

Character Controller vốn phi vật lý, do đó nếu muốn áp dụng vật lý thực – đu trên dây thừng, đẩy những tản đá… – cho nhân vật, phải sử dụng một rigidbody, điều này cho phép sử dụng các khớp, các lực lên nhân vật. Character Controller luôn được điều khiển theo trục y, vì vậy cần sử dụng rigidbody nếu nhân vật của bạn cẩn có khả năng thay đổi hướng trong không gian. Tuy nhiên, lưu ý rằng việc điều chỉnh một rigidbody phù hợp với một nhân vật là rất khó khăn do các hành vi của nhân vật là phi vật lý – Ví dụ: các nhân vật có thể trượt trơn tru trên các con dốc có chiều cao nhất định trong khi các rigidbodies thì sẽ không.

#### Các Component vật lý trong Unity

* Rigidbody

Các thuộc tính bao gồm:

|  |  |
| --- | --- |
| Tên | Ý nghĩa |
| Mass | Khối lượng của đối tượng. |
| Drag | Thể hiện sức cản không khí đến đối tượng. |
| Angular Drag | Mức cản không khí ảnh hưởng đến đối tượng khi quay. |
| Use Gravity | Nếu được kích hoạt thì 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 bởi Transform của nó. |
| Interpolate | Chế độ làm mượt chuyển động. Gồm các tham số:   * None: Không áp dụng * Interpolate: Transform sẽ mượt mà giựa trên Transform của khung hình trước đó. * Extrapolate: Transform sẽ mượt mà giựa trên dự đoán Transform của khung hình kế tiếp. |
| Collision Detetion | Được sử dụng để ngăn chặn các đối tượng di chuyển nhanh xuyên qua các đối tượng khác dẫn đến không thể phát hiện va chạm. |
| Constraints | Cho phép giới hạn di chuyển của các Rigidbody, gồm giới hạn di chuyển và giới hạn quay theo 3 trục. |

Bảng . Các thuộc tính của Rigidbody

* Các component Collider

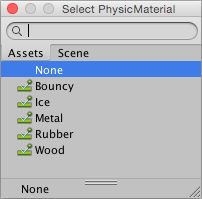
Collider là một loại component phải được thêm vào cùng với rigidbody để cho phép va chạm xảy ra. Nếu 2 rigidbody va đập vào nhau, hệ thống vật lý sẽ không tính toán va chạm trừ khi cả hai đối tượng đều có gắn Collider component. Có các loại Collider gồm:

|  |  |
| --- | --- |
| 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 hình giống con nhộng. |
| Mesh Collider | Một Collider từ một mesh (mạng lưới bao quanh object) của object, không thể va chạm với Mesh Collider khác. |
| Wheel Collider | Một loại Collider đặc biệt sử dụng cho đối tượng xe cộ. |

Bảng . Các loại Collider

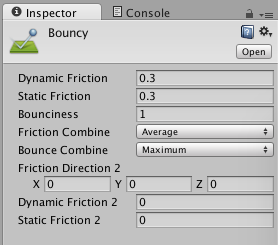
* Physic Material

Ma sát, độ đàn hồi và tính mềm dẻo được định nghĩa trong Physics Material. Standard Assets có chứa hầu hết các loại vật liệu vật lý phổ biến. Để sử dụng chúng, click vào Physics Material và xuống chọn một trong số chúng ví dụ: gỗ, sắt, băng… 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 sử dụng khi đã di chuyển. Có giá trị từ 0 đến 1. Giá trị 0 sẽ giống như băng, giá trị 1 sẽ làm cho nó dừng nhanh chóng trừ khi có rất nhiều lực hoặc trọng lực tác động vào đối tượng.
* *Static Friction*: Ma sát được sử dụng khi một đối tượng vẫn còn nằm trên một bề mặt. Có giá trị từ 0 đến 1. Giá trị 0 sẽ giống như băng, giá trị 1 sẽ làm đối tượng khó có thể di chuyển.
* *Bounciness*: Định nghĩa độ đàn hồi. Giá trị 0 là không đàn hồi, giá trị 1 sẽ đàn hồi mà không mất năng lượng.
* *Friction combine mode*: Định nghĩa 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*: Định nghĩa 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.
* Joint

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

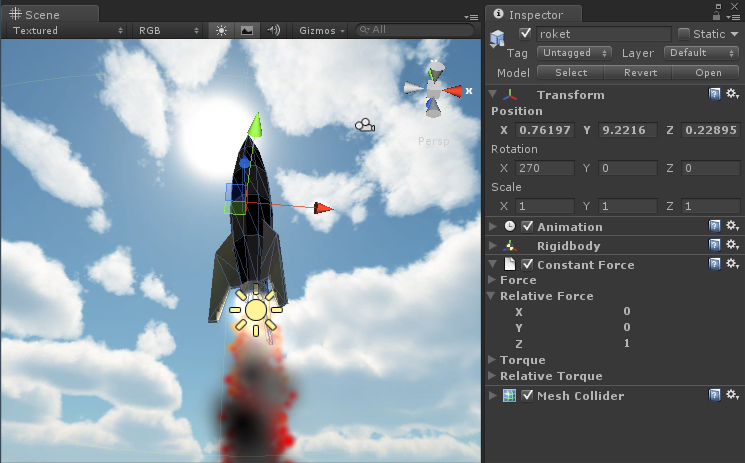
Có ba loại Joint cơ bản: Fixed Joint, Spring Joint và Hinge Joint với các mục đích sử dụng khác nhau.

|  |  |
| --- | --- |
| Tên | Miêu tả |
| FixedJoint | Gắn cứng hai đối tượng lại với nhau |
| SpringJoint | Kết nối các đối tượng dạng lò xo |
| HingeJoint | Kết nối các đối tượng dạng sợi dây |

Bảng . Các loại Joint

* Constant Force

Constant Force là một tiện ích giúp ta nhanh chóng thêm lực liên tục tác động vào Rigidbody. Điều này hoạt động tuyệt vời cho những đối tượng bắn như tên lửa…nếu ta muốn nó bắt đầu với một vận tốc lớn thay vì tăng tốc từ từ.



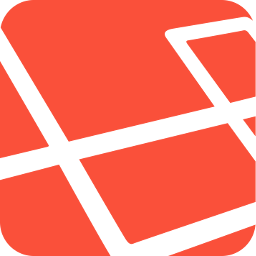
Hình . Constant Force áp dụng cho tên lửa đẩy

Thuộc tính:

* *Force*: Vector lực tác dụng trong hệ tọa độ world.
* *Relative*: Vector lực tác dụng trong hệ tọa độ của đối tượng (local space).
* *Torque*: Vector của mô-men xoắn tác dụng trong hệ tọa độ world. Đối tượng sẽ bắt đầu quay xung quanh vector này. Vector càng lớn, quay càng nhanh.
* *Relative Torque*: Vector của mô-men xoắn tác dụng vaog trong hệ tọa độ của đối tượng (local space). Đối tượng sẽ bắt đầu quay xung quanh vector này. Vector càng lớn, quay càng nhanh.

Để thực hiện một tên lửa tăng tốc về phía trước, thiết lập một Relative Force theo dọc truc Z dương. Sau đó sử dụng Drag của Rigidbody để làm nó không vượt qua một vận tốc tối đa (giá trị càng cao thì vận tôc tối đa càng thấp). Trong Rigidbody, tắt gravity để đảm bảo tên lửa sẽ luôn luôn trên đường đi của nó.

## Laravel Frameword



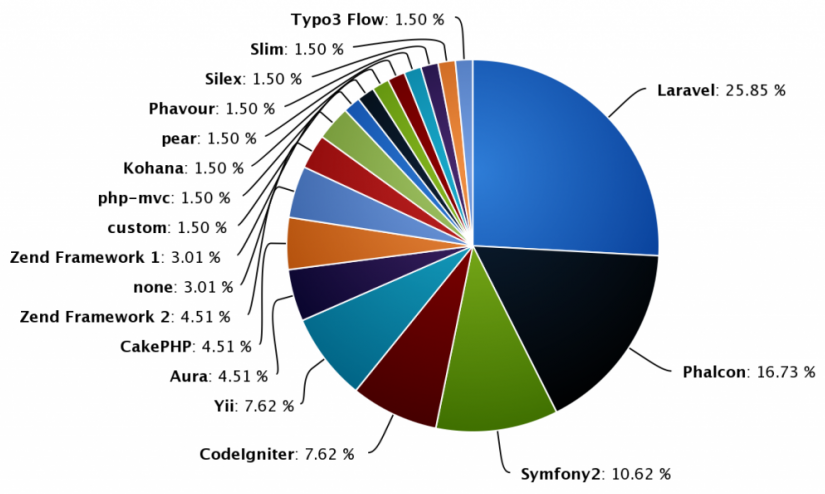
Hình . Laravel Framework

### Giới thiệu

Laravel Frameword là một framework phát triển web miễn phí, mã nguồn mở, được thiết kế theo mô hình model-view-control (MVC). Laravel được xuất bản dưới giấy phép MIT.

Laravel Framework ra đời lần đầu vào ngày 22/2/2012 bởi Taylor Otwell. Theo một khảo sát của các lập trình viên PHP, Laravel là framework PHP được sử dụng nhiều nhất trên thế giới trong năm 2013 và tiếp tục trong năm 2014, đứng sau là Phalcon, Symfony2, CodeIgniter, Yii.

Laravel Framework 4 là phiên bản mới nhất của Laravel, ra đời rất muộn vào 28/5/2013, chính vì vậy mà framework này đã kế thừa lại được những thế mạnh của framework khác. Laravel đã nhanh chóng có được một cộng đồng rất lớn trong thế giới các Framework của ngôn ngữ lập trình PHP.



Hình . Tỷ lệ sử dụng các framework PHP năm 2013

Sự thành công của Laravel trước hết là nhờ sự tinh tế nằm ở chỗ bắt kịp được xu hướng công nghệ mà điểm nhấn ở đây là các tính năng mới trong các phiên bản PHP 5.3 trở lên. Điều đó được thể hiện qua khái niệm namespace, composer, closure và rất nhiều những tiêu chuẩn trong design pattern được áp dụng trên nền tảng framework này. Đồng thời, với cách hướng dẫn đơn giản và dễ tiếp cận giống với Codeigniter Framework đã khiến người dùng thích ngay từ lần đầu tiếp xúc.

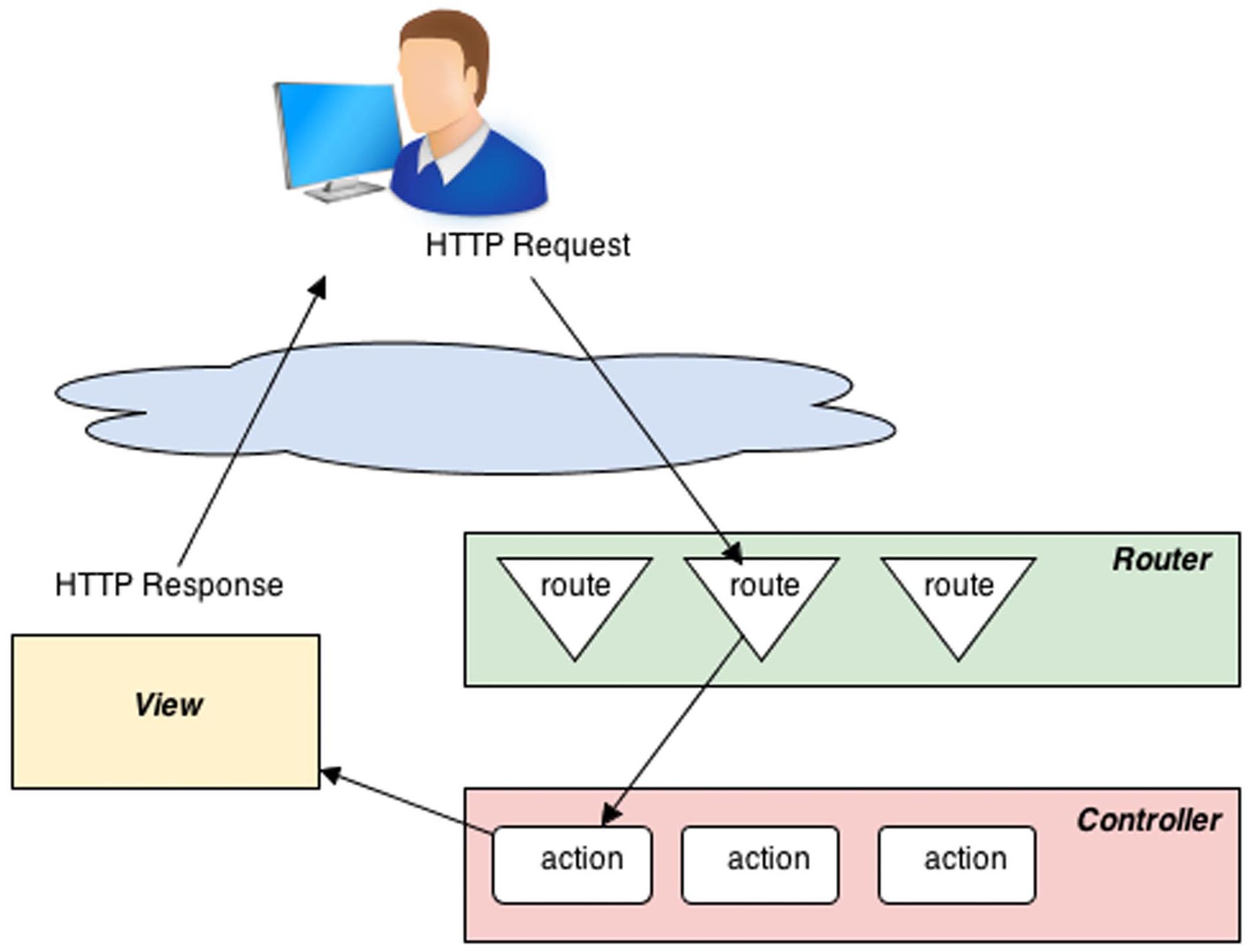
Laravel 4, vừa chỉ ra mắt vào cuối tháng 5 - 2013, cũng có sự tích hợp của một phần trong thư viện symfony và áp dụng triệt để mô hình ORM với khái niệm liên quan đến Eloquent class. Đồng thời, nó cũng giải quyết được những vấn đề mà các framework khác đang mắc phải. Chẳng hạn như master layout, mô hình xử lý với ORM, event model,....

### Một số ưu điểm

* Route trong Laravel thật sự khác biệt, mới mẻ và đầy mạnh mẽ
* Master layout được tích hợp sẵn cùng Blade template giúp code trở nên gọn gàng và tiện dụng
* Migration quản lý database thật dễ dàng khi làm việc đội nhóm
* Eloquent class đầy mạnh mẽ nổi bật khi xử lý cơ sở dữ liệu quan hệ 1–N và N–N.
* Composer quản lý và tích hợp các thư viện khác thật hay và không lo lắng khi thư viện đó bị thay đổi.
* Document dễ đọc và dễ hiểu.
* Eloquent ORM: đây là một ORM tuyệt vời với khả năng migration data và làm việc tốt với MySQL, Postgres, SQL Server và SQLite.

Và còn nhiều những chức năng khác…

### Request LifeCycle



Hình . Vòng đời một Request trong Laravel

Vòng đời tiêu chuẩn của một của một request khi gửi đến laravel như sau:

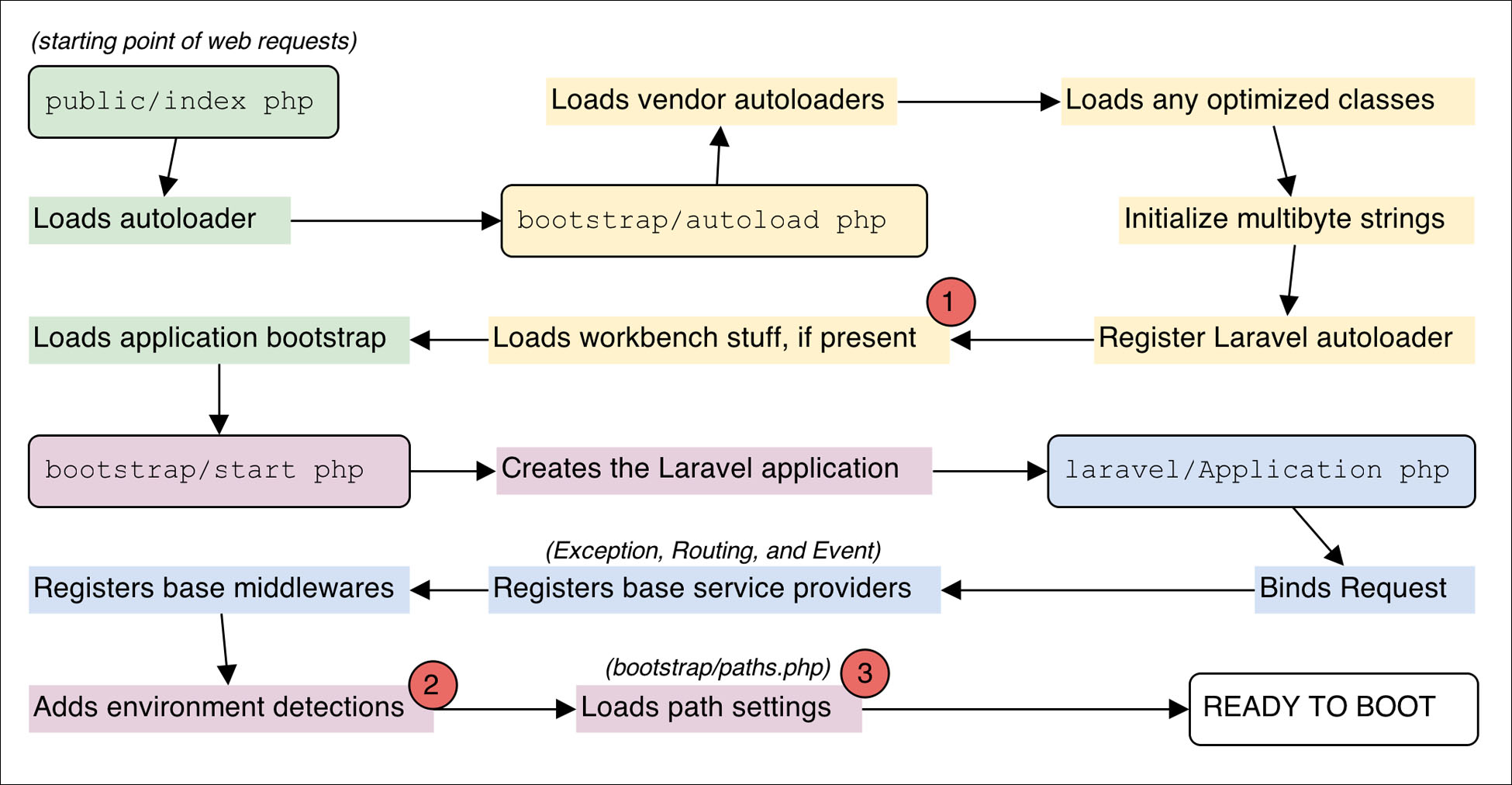
* Yêu cầu HTTP từ Routed tời một Controller
* Controller sẽ thực hiện những action và gửi kết quả tới View
* View sẽ hiện những kiểu dữ liệu phù hợp và gửi lại HTTP Response

Tuy nhiên, có rất nhiều ngoại lệ xảy ra trong vòng đời tiêu chuẩn này. VD:

* Route có thể trả về trực tiếp Views hay Responses, bỏ qua Controllers
* Filter có thể xảy ra trước hay sau route

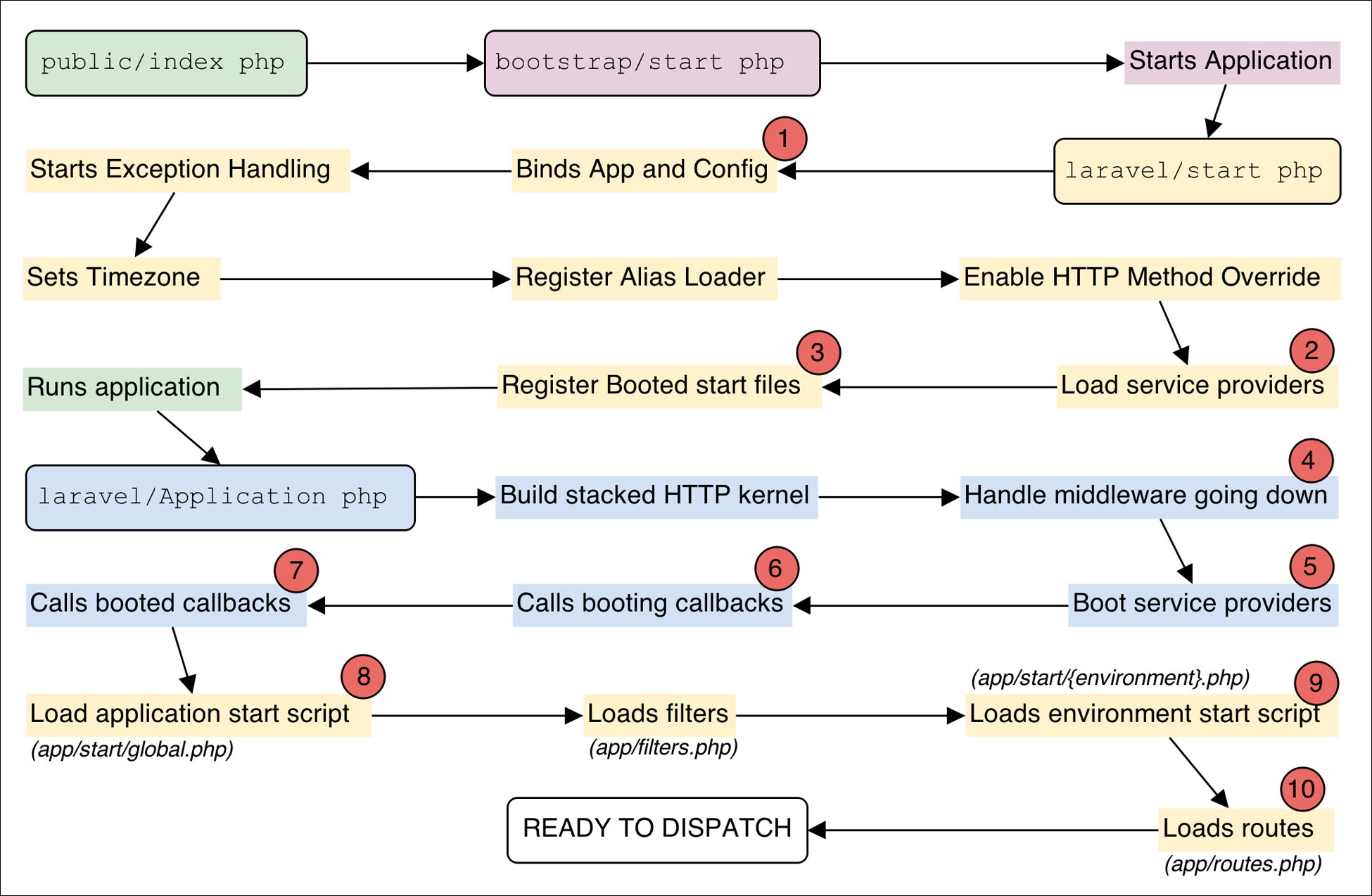
Toàn bộ vòng đời của một request có thể chia làm ba giai đoạn: Loading, Booting và Running.

* Bước Loading



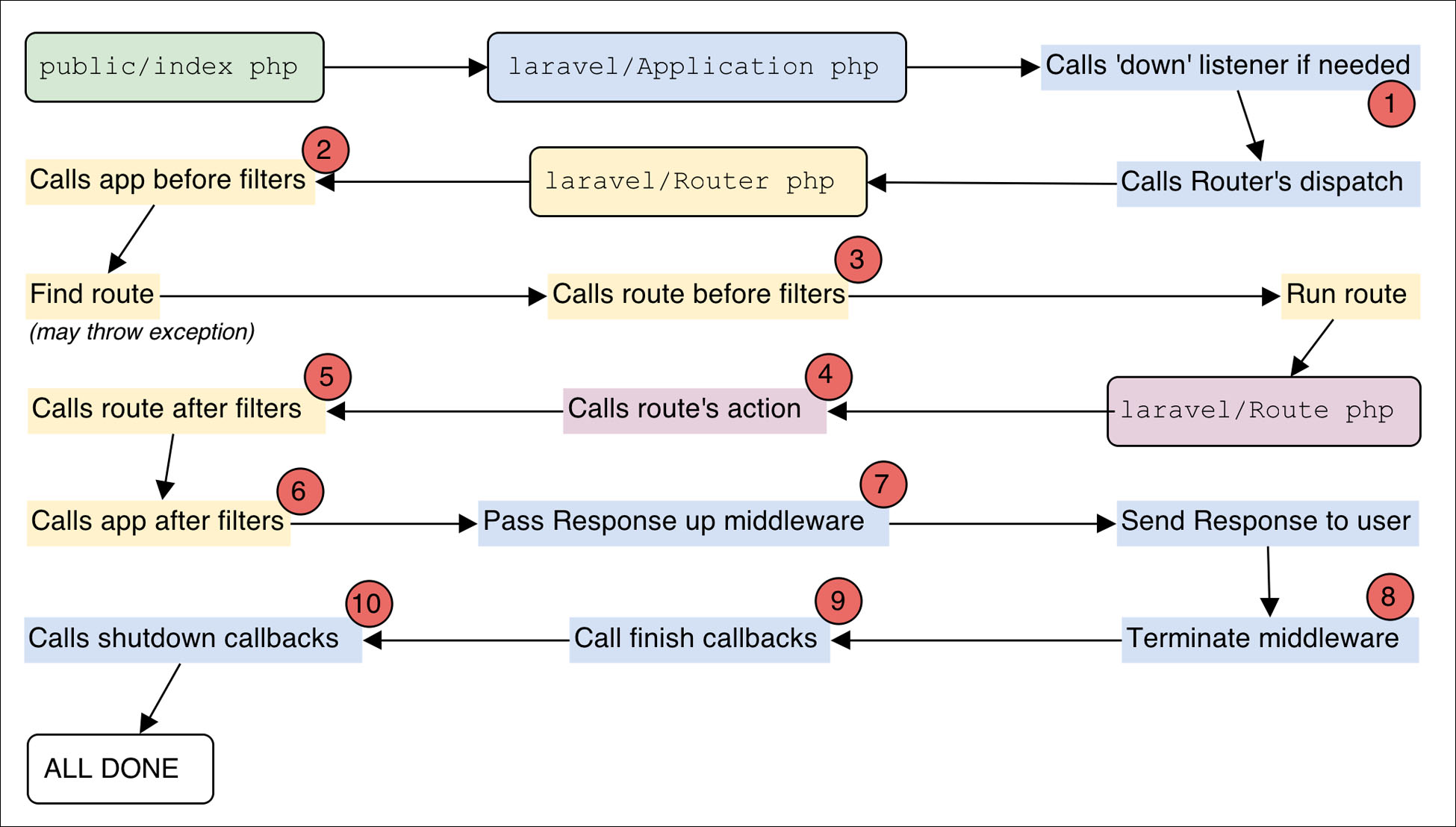
Hình . Bước Loading trong Request

* Bước Running



Hình . Bước Running trong Request

* Bước Running



Hình . Bước Running trong Request

### Router

Mục đích của router là định tuyến đến những controller cụ thể nào từ phía request của người sử dụng. Đây là một định nghĩa khá quen thuộc khi làm việc với các framework PHP. Tuy nhiên, ở Laravel, router có sự khác biệt rõ ràng. Bạn có thể không cần thiết phải tạo ra controller mà thực thi những công việc mong muốn 1 cách trực tiếp tại route dễ dàng. Bởi kết hợp theo phương pháp closure quen thuộc trong javascript để thực thi điều đó.

Để làm việc được với route trước hết ta tiếp xúc chúng với công thức cơ bản như sau:

Route::method ('Tên định danh', Tham số);

Trong đó:

* *Method*: xây dựng dựa trên Restful, về cơ bản là nó có các cơ chế: post, get, put, delete, any
  + *POST*: dành cho các thao tác lấy từ form như thêm record.
  + *GET*: dành cho các thao tác truy cập thông thường tương đương với request cơ bản trong PHP.
  + *PUT*: dành cho thao tác lấy từ form nhưng là cập nhật record.
  + *DELETE*: dành cho các thao tác thực thi hành động xóa bỏ.
  + *ANY*: là sự tổng hợp của các thao tác ở trên.
* *Tên định danh*: là đường link mong muốn trong ứng dụng.
* *Tham số*: Những thao tác mà chúng ta mong muốn với định danh trên. Tham số có thể là hàm xử lý, có thể là array() chứa các thông tin xử lý khác, có thể là sự ám chỉ cụ thể 1 controller nào đó cho định danh,...

Có rất nhiều cách truyền tham số, ta hoàn toàn có thể kết hợp để tạo ra những mong muốn tùy thích dễ dàng, trong đó ta quy về 2 cách phổ biến là:

* Truyền vào hàm closure xử lý trực tiếp, không cần qua Controller
* Truyền vào tên một action và một Controller sẽ xử lý

### View

View là các file .php trong thư mục *app/view.* Là chữ V trong mô hình MVC, View dùng để tách biệt những đoạn mã xử lý logic với những đoạn mã hiển thị. Để sử dụng view, rất đơn giản:

**return** View::make("Tên", Đối số nếu có);

Trong đó *Tên* là tên file (không tính “.php”) trong thư mục view, sau đó là phần đối số được truyền vào cho view.

Có rất nhiều cách truyền đối số vào cho view.

* Sử dụng đối số truyền trực tiếp
* Sử dụng phương thức *with*
* Sử dụng hàm compact trong PHP

### Controller

Controller là chữ C trong mô hình MVC của Laravel, Controller sẽ là phần trung gian giao tiếp với phần Model (M) và View (V).

Laravel cho phép vận hành trực tiếp trong Routing thông qua cách viết closure, nhưng khi phải xử lý các thao tác phức tạp và nhiều thì tốt hơn hết nên thao tác chúng trên Controller. Có như thế thì ứng dụng của mới trở nên linh hoạt và dễ mở rộng sau này.

Để thao tác với controller, trước hết cần tạo trong thư mục *app/controllers* một file theo cơ chế: *TênController.php*. Ví dụ: *DemoController.php*

Trong file trên ta xây dựng bởi những đoạn lệnh mở đầu như sau:

<?php

**class** DemoController **extends** BaseController{

**public** **function** index(){

**return** View::make("demo.index")->with("title","Hello");

    }

}

Mã lệnh ở trên tạo ra *DemoController* kế thừa lớp chủ đạo là *BaseController* và phương thức đầu tiên ta xây dựng là *index()*. Trong phương thức này ta thực hiện việc gọi view từ thư mục *views/demo*, nạp tập tin *index.php*. Đồng thời ta truyền sang view một biến tên title với giá trị là "Hello".

Kế tới, ta tạo ra file *index.php* trong thư mục *views/demo* với mã lệnh sau:

<?php

    echo $title;

?>

Sau cùng, ta cần khai báo trong file *routes.php* như sau:

<?php

Route::get("demo","DemoController@index");

?>

Chạy đường dẫn như sau: *localhost/laravel/public/demo* ta sẽ thấy thông báo hiển thị như mong đợi.

### Filter

Filter có thể hiểu một cách đơn giản là bộ lọc, nhiệm vụ của nó là lọc dữ liệu từ phía người dùng trước hoặc sau khi nó chạm tới Routing. Vì thế, nếu tận dụng tốt được filter sẽ tạo ra bộ ứng dụng phòng thủ chuyên nghiệp trong website. Điều này sẽ giúp xử lý dữ liệu tối ưu hơn.

Với filter của Laravel thì mọi chuyện thật dễ dàng, ta có thể làm điều đó một cách độc lập trên filter, sau đó lồng ghép một cách khéo léo vào routing mong muốn. Thì ngay khi request tới với routing sẽ bị filter chặn lại và xử lý. Nếu hợp lệ thì mới tiếp tục đi tới routing, còn nếu không thì sẽ thực thi việc gì đó dễ dàng. Với filter ta có thể tối giản code ở mức cao nhất và tái sử dụng vô cùng đơn giản.

Filter có thể đặt trước Route hoặc sau Route đều được, tuỳ vào mục đích sử dụng.



Hình . Request Filter trong Laravel

## MongoDB

### 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 là JSON và dạng dữ liệu theo kiểu key và value. NoSQL ra đời như là 1 mảnh vá cho những khuyết điểm và thiếu xót 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 các bạn sử dụng SQL server hoăc My SQL thì giữa các bảng (table) 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 No SQL thì khác Với NoSQL bạn có thể mở rộng dữ liệu mà không lo tới những việc như tạo khóa ngoại, khóa chính, kiểm tra ràng buộc, .v.v...

* Nhước điểm

Vì NoSQL không hạn chế việc mở rộng dữ liệu nên tồn tại nhiều nhược điểm như: sự phục thuộc của từng bản ghi, tính nhất quán, toàn vẹn dữ liệu,... nhưng chúng ta có thể chấp nhận những nhược điểm đó để khiến ứng dụng cải thiện 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 như 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 (bây giờ 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…

#### 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 nó có một sực khác biệt đó là nó không có ràng buộc giữa các collection như các hệ quản trị CSDL khác nên việc truy xuất rất nhanh, chính vì thế mỗi collection có thể chứa nhiều thể loại khác nhau không giống như table trong hệ quản trị mysql là các field cố định.

* 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. Document ta có thể hiểu nó giống như các record dữ liệu trong MySQL, tuy nhiên nó có 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) |

Hình . So sánh các khái niệm trong SQL và NoSQL

#### Tính năng chính

* Document-Oriented

Thay vì chia nhỏ các thuộc tính của 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 *tên sách* và *tác giả* ở hai bảng khác nhau, thì MongoDB lưu toàn bộ đối tượng *sách* vào một collect.

* Ad hoc querries

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

* Indexing

Tất cả các field trong MongoDB đều có thể được dùng để index.

* 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 tán. 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.

* Server-side JavaScript Execution

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

* Capped Collections

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

#### Sử dụng

MongoDB thật sự rất hot nhưng không phải lúc nào ta sử dụng nó cũng tốt, 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.
* 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 web bạn có đến 10 triệu records thì đó là cơn ác mộng với MySQL. Bởi vì MongoDB có khả năng tìm kiến thông tin liên quan cũng 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 SQL (sử dụng joins)

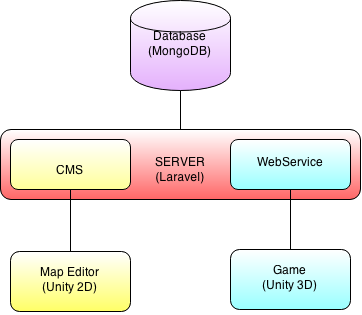
# PHÂN TÍCH THIẾT KẾ

## Tổng quan chức năng

Hệ thống gồm 3 thành phần: Game, CMS và Map Editor. Trong đó, chức năng chi tiết của từng phần như sau:

* Game
  + Đăng ký tài khoản
  + Quên mật khẩu
  + Đăng nhập / Đăng xuất
  + Chơi game
    - Kiểm tra các lỗi vi phạm
    - Xem quy định về lỗi
    - Xem mức phạt
  + Xem lịch sử kết quả
* CMS
  + Đăng nhập / Đăng xuất
  + Phân quyền
  + Quản lý tài khoản
  + Quản lý điểm
  + Quản lý nội dung game
  + Lập báo cáo
* Map Editor
  + Tạo map mới
  + Mở map cũ
  + Chỉnh sửa map với các thành phần
    - Các thành phần đường
      * Làn đường
      * Giao lộ
      * Điểm dừng xe buýt
    - Biển báo
      * Biển báo cấm
      * Biển báo hiệu lệnh
    - Phong cảnh
      * Toà nhà
      * Cây cối
    - Khác
      * Đèn đỏ
      * Điểm xuất phát / Điểm kết thúc
      * Các điểm checkpoint
      * Xe tự động

## Kiến trúc tổng quát



Hình . Kiến trúc tổng quát hệ thống

Trong đó:

Server sử dụng Laravel, làm việc trực tiếp với databse MongoDB. Server bao gồm hai thành phần là CMS và Web Service.

* CMS – Map Editor

Làm việc trực tiếp với Map Editor, khi nhà quản trị sử dụng Map Editor để tạo ra map mới hoặc chỉnh sửa map, nhà quản trị sẽ sử dụng CMS để upload dữ liệu file map và đưa vào database. Kèm theo đó là các thông tin của map, bao gồm:

* Tên map
* Level
* Thời gian ảo trong map
* Thumnail
* Web Service – Game

Cung cấp các hàm api để Game thao tác với dữ liệu trên database, bao gồm:

* Đăng ký
* Quên mật khẩu
* Đăng nhập - Đăng xuất
* Lấy thông tin user
* Lấy lịch sử chơi của user
* Lấy danh sách các map khả dụng
* Lấy dữ liệu map
* Upload kết quả chơi

## Game

### Yêu cầu

Mục đích chính của game là nhằm mô phỏng chân thực hoạt động tham gia giao thông hằng ngày của người chơi, do đó, game cần thoả mãn được những yêu cầu cơ bản sau:

* Giả lập được góc nhìn người chơi
* Giả lập đầy đủ các điều khiển cơ bản của các xe máy phổ thông như:
  + Bản hiển thị tốc độ
  + Bật / Tắt đèn
  + Đèn xa / Đèn gần
  + Đèn chuyển hướng trái / phải
  + Còi
* Xây dựng được đầy đủ các thành chính trong hệ thống cơ sở hạ tầng giao thông, bao gồm:
  + Đường, làn đường, giao lộ, vỉa hè, điểm dừng xe buýt
  + Vạch kẽ đường, dải phân cách
  + Biển báo giao thông
  + Đèn giao thông
  + Những xe tham gia giao thông cùng người chơi
* Giả lập các tình trạng giao thông thường gặp
* Hỗ trợ kiểm tra các ràng buột trên đường, bao gồm:
  + Tốc độ tối thiểu – tối đa
  + Các hướng được phép rẽ
  + Các loại xe được chạy, được dừng
* Hỗ trợ phát hiện các lỗi vi phạm của người chơi khi tham gia giao thông

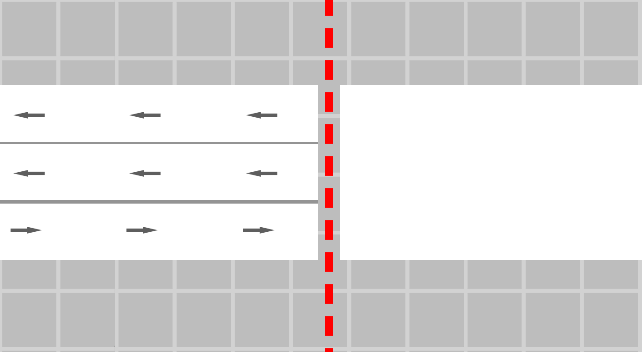
### Phân tích

Trong số các yêu cầu của game, yêu cầu về kiểm tra các lỗi vi phạm của người chơi là khó khăn nhất, đòi hỏi cần có cách thiết kế sao cho dễ dàng kiểm tra, đồng thời cho phép mở rộng về sau. Các hướng mở rộng có thể là bổ sung thêm các lỗi mới, hoặc hỗ trợ phương tiện giao thông khác cho người chơi.

* Vấn đề 1

*Đơn vị cơ sở của bản đồ là gì?*

Có hai lựa chọn cho vấn đề này: Đường và Làn đường.



Hình . Làn đường (bên trái) và Đường (bên phải)

* Đường:
  + **Ưu điểm**: Số đối tượng cần tạo ít hơn, do đó quá trình thao tác để tạo bản đồ sẽ nhanh hơn.
  + **Nhược điểm**: Cần nhiều thông số để có thể quy định các đặt điểm của các làn đường.
* Làn đường:
  + **Ưu điểm**: Cần ít thông số hơn, có thể tạo ra con đường có nhiều làn đường một cách đơn giản hơn.
  + **Nhược điểm**: Số đối tượng tạo ra nhiều hơn, thao tác nhiều hơn.

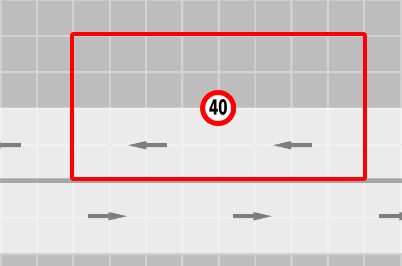
*Kết luận:* Giải pháp 2 hợp lý hơn nên được sử dụng. Tổng hợp từ vấn đề 1, khi đó, mọi thông số ràng buột của các biển báo giao thông sẽ được lưu trữ trên đối tượng Làn đường, biển báo chỉ được đặt để người chơi biết các ràng buột đang có trên Làn đường.

* Vấn đề 2

*Làm cách nào để biển báo giao thông có thể tạo ra các ràng buột trên đường?*

Trước hết, ta nhận định về biển báo giao thông, rất khó để một biển báo giao thông trở thành đối tượng lưu trữ các thông tin ràng buột trên đường. Vd: một biển báo giới hạn tốc độ tối đa 40km/h đặt tại lề của một con đường. Khi xe chạy đến gần biển báo, ta không có cách nào xác định chính xác đâu là nơi biển báo có hiệu lực và khi nào hết hiệu lực. Chưa kể khó khăn để xác định biển báo có đang có hiệu lực lên đường của xe đó chạy hay một đoạn đường khác, gần đó.

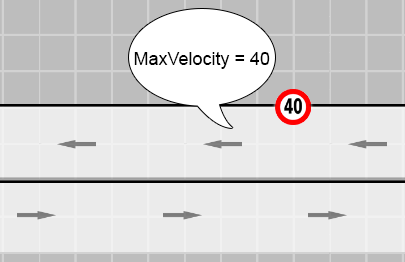
* Giải pháp 1:



Hình . Giải pháp biển báo giao thông chứa thông tin

Mỗi biển báo sẽ có một vùng tác động hình chữ nhật trong không gian 2 chiều. Khi người chơi nằm vào khu vực đó, biển báo sẽ có hiệu lực với người chơi đó. Suốt quá trình người chơi di chuyển, sẽ luôn có một vòng lặp kiểm tra người chơi thuộc khu vực biển báo nào và từ đó kiểm tra xem người chơi có vi phạm lỗi nào hay không.

* + **Ưu điểm**: Ràng buột được hiển thị trực quan thông qua vị trí biển báo và nội dung biển báo.
  + **Nhược điểm**:
    - Xây dựng Map Editor khó khăn
    - Vòng lặp kiểm tra các biển báo sẽ chạy liên tục, gây chậm.
* Giải pháp 2:



Hình . Giải pháp làn đường chứa thông tin

Đường sẽ là nơi chứa mọi thông tin ràng buột và biển báo sẽ chỉ có vai trò hiển thị cho người xem biết.

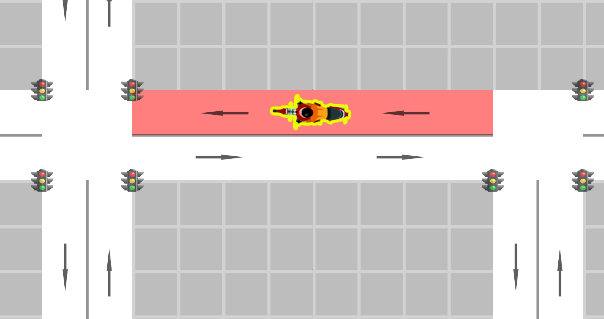
* + **Ưu điểm**:
    - Dễ dàng kiểm tra khu vực có tác dụng của ràng buột
    - Xây dựng Map Editor dễ dàng hơn
  + **Nhược điểm**: Không có

*Kết luận:* Giải pháp 2 hợp lý hơn nên được sử dụng. Mọi biển báo trên bản đồ sẽ chỉ có giá trị hiển thị, các thông số ràng buột sẽ được gán cho đối tượng “đường”.

* Vấn đề 3

*Làm cách nào để kiểm tra lỗi của người chơi?*

Khi di chuyển trên bản đồ, người chơi luôn nằm trên một Làn đường nào đó, và không bao giờ vượt ra ngoài. Đồng thời, các ràng buột đều được gắn trên Làn đường, do đó, nếu muốn kiểm tra các lỗi, thì chỉ có bằng cách xem xét thông tin trên Làn đường mà người chơi đang chạy.



Hình . Vấn đề kiểm tra lỗi người chơi

VD:

* Để kiểm tra xe vượt quá tốc độ, ta so sánh tốc độ của xe và tốc độ giới hạn của làn đường.
* Để kiểm tra xe chạy ngược chiều, ta so sánh hướng đi của xe so với hướng của làn đường.
* Vấn đề 4

*Làm cách nào để các xe tự động chạy đúng?*

Các xe tự động cần đảm bảo 2 yếu tố sau:

* Chạy đúng luật

Cơ chế chạy của xe tự động như sau. Khi xuất phát, xe sẽ được đặt trên một làn đường hợp lệ và hướng chạy hợp lệ. Xe sẽ chạy đều, khi đến các giao lộ, xe sẽ lựa chọn ngẫu nhiên 1 hướng chạy và tiếp tục chạy theo hướng đó. Nếu không tìm ra hướng chạy tiếp theo, xe sẽ ngừng lại ở đó.

* Tạo ra các tình huống giao thông dễ khiến người chơi vi phạm giao thông

### Cấu trúc dữ liệu

* Road

Đây là đối tượng cơ bản trên bản đồ, chứa tất cả thông tin ràng buột của các biển báo giao thông đặt trên đường đó. Các thông số ràng buột của Làn đường bao gồm:

* Tốc độ thiểu – tốc độ tối đa
* Được phép rẽ trái – rẽ phải – đi thẳng – quay đầu
* Danh sách loại xe được đi
* Danh sách loại xe được dừng
* Các vỉa hè (trên, dưới, trái, phải)
* Các dải phân cách (trên, dưới, trái, phải)
* Trạng thái đèn giao thông
* Player

Lớp quản lý xe của người chơi. Để có thể kiểm tra các lỗi mà người chơi mắc phải, cần có các thông số sau:

* Trạng thái mũ bảo hiểm (có/không)
* Trạng thái đèn (bật/tắt)
* Trạng thái đèn chiếu (xa/gần)
* Trạng thái đèn rẽ hướng (tắt/trái/phải)
* Hướng xe chạy (trái, phải, lên, xuống)
* Tốc độ
* Map

Đối tượng chứa tất cả thông tin của bản đồ. Map được xây dựng bằng Map Editor và được xuất ra file dưới định dạng JSON. Sau đó được upload lên server, khi game chạy sẽ load về dưới dạng text để chuyển về lại định dạng ban đầu.

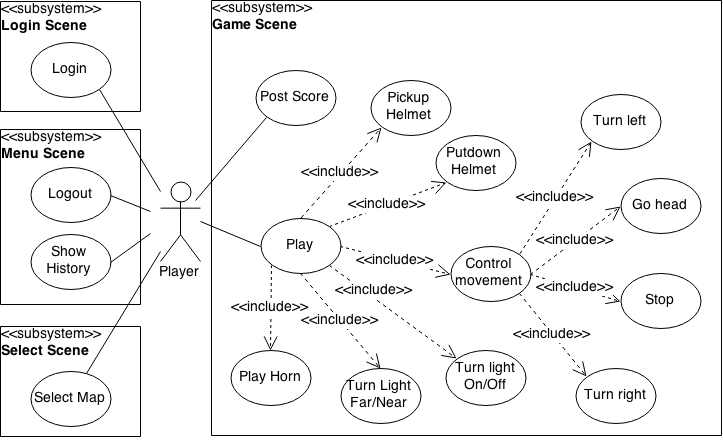
* MapManager

Lớp quản lý map, sử dụng các lớp phụ trợ để thực hiện các công việc: load và parse map (MapLoader), vẽ map (MapRenderer), vẽ mini map (MiniMap).

* ErrorManager

Lớp quản lý tất cả các lỗi của người dùng. Sử dụng lớp ConfigError để lấy các thông số lỗi từ file csv và lớp NotifierHandler để thông báo lỗi lên màn hình.

### Sơ đồ use case



Hình . Sơ đồ Use case của Game

Danh sách Actor

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

Bảng . Danh sách Actor trong sơ đồ Use case của Game

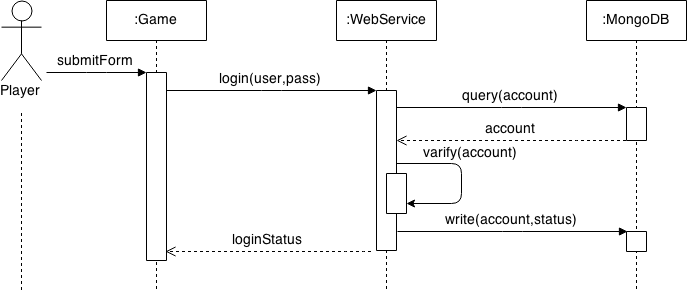
Danh sách Use case

|  |  |  |
| --- | --- | --- |
| STT | Tên Use case | Ý nghĩa |
| 1 | Login | Đăng nhập vào hệ thống |
| 2 | Logout | Đăng xuất khỏi hệ thống |
| 3 | Show history | Xem lịch sử các lần chơi trước đó |
| 4 | Select map | Chọn một map để chơi |
| 5 | Play | Các thao tác chơi trong màn hình chơi chính |
| 6 | Pickup helmet | Đội mũ bảo hiểm |
| 7 | Putdown helmet | Bỏ mũ bảo hiểm |
| 8 | Play horn | Bấm còi |
| 9 | Turn light On/Off | Bật đèn / Tắt đèn |
| 10 | Turn ligth Near/Far | Bật đèn chiếu gần / đèn chiếu xa |
| 11 | Control movement | Các thao tác điểu khiển xe |
| 12 | Turn left | Cho xe rẽ trái |
| 13 | Turn right | Cho xe rẽ phải |
| 14 | Go ahead | Cho xe chạy thẳng và tăng tốc |
| 15 | Stop | Cho xe dừng |
| 16 | Post Score | Lưu điểm lên server |

Bảng . Danh sách Use case trong Game

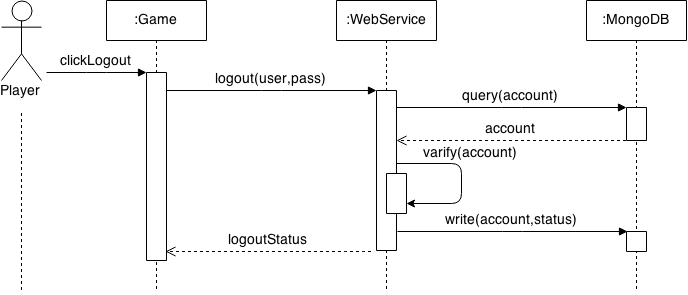
### Sơ đồ tuần tự

#### Sơ đồ tuần tự Use case Login



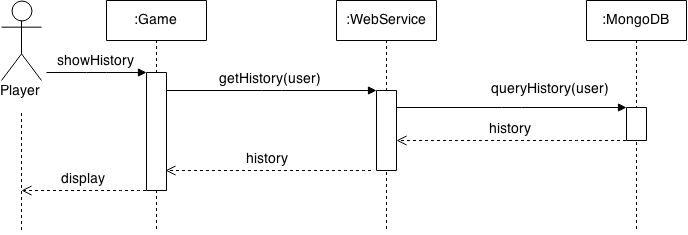
Hình . Sơ đồ tuần tự Use case Login

#### Sơ đồ tuần tự Use case Logout



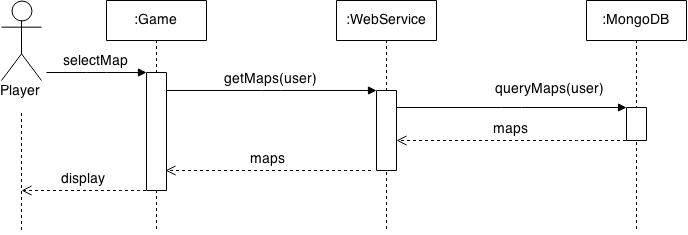
Hình . Sơ đồ tuần tự Use case Logout

#### Sơ đồ tuần tự Use case Show History



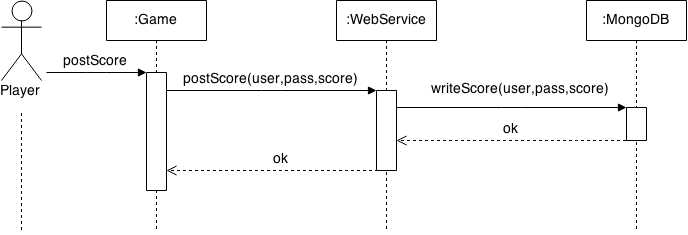
Hình . Sơ đồ tuần tự Use case Show History

#### Sơ đồ tuần tự Use case Select Map



Hình . Sơ đồ tuần tự Use case Select Map

#### Sơ đồ tuần tự Use case Post Score



Hình . Sơ đồ tuần tự Use case Post Score

### Giải thuật

Các trường hợp bắt lỗi vi phạm của người chơi đều dựa trên các thông tin của Player và Road.

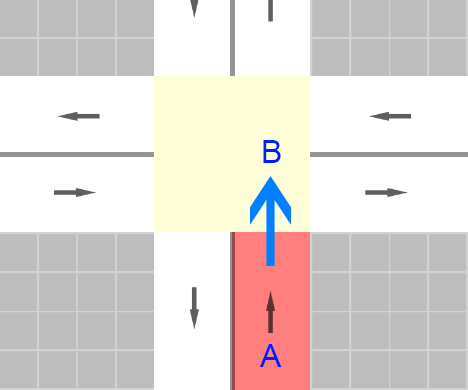
* Lỗi tốc độ

Khi tốc độ xe của người chơi lớn hơn tốc độ tối đa, hoặc bé hơn tốc độ tối thiểu của làn đường thì người chơi mắc “Lỗi tốc độ”.

* Lỗi đi ngược chiều

Khi người chơi di chuyển theo chiều ngược lại với chiều của làn đường thì người chơi mắc “Lỗi đi ngược chiều”.

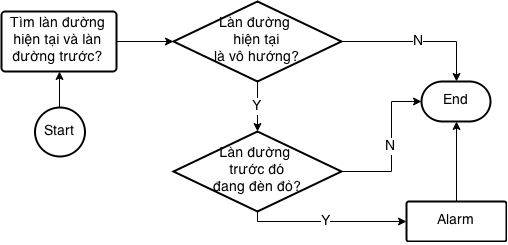
* Vượt đèn đỏ



Hình . Vượt đèn đỏ

Làn đường ở giữa giao lộ còn được gọi là làn đường vô hướng (màu vàng).

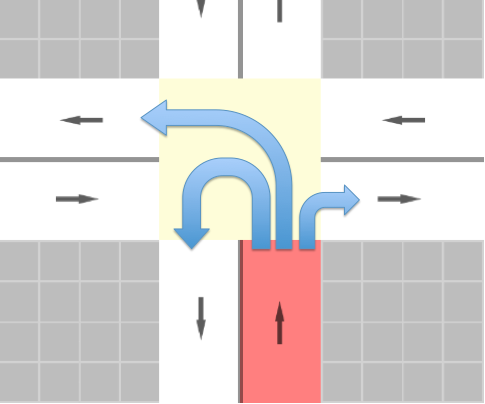
Làn đường A đang ở trạng thái đèn đỏ. Khi đó, nếu người chơi di chuyển sang làn đường vô hướng B thì người chơi đã vi phạm lỗi “Vượt đèn đỏ”.



Hình . Sơ đồ logic phát hiện lỗi Vượt đèn đỏ

* Chuyển hướng không có tín hiệu

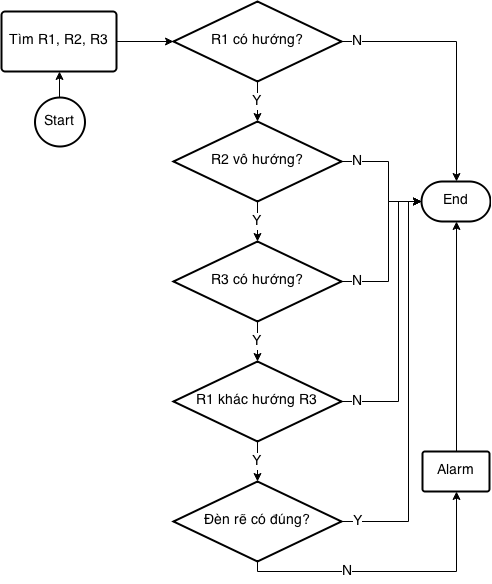
Chuyển hướng là di chuyển sang 3 hướng không cùng hướng còn lại, sau khi di chuyển vào vùng làn đường vô hướng (màu vàng).



Hình . Chuyển hướng

Khi người chơi chuyển hướng mà trạng thái đèn chuyển hướng không hợp lý thì sẽ mắc lỗi “Chuyển hướng không có tín hiệu”.

Luồng xử lý:

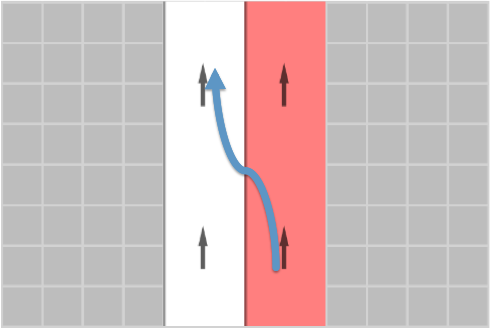


Hình . Sơ đồ logic phát hiện lỗi Chuyển hướng không tín hiệu

Trong đó:

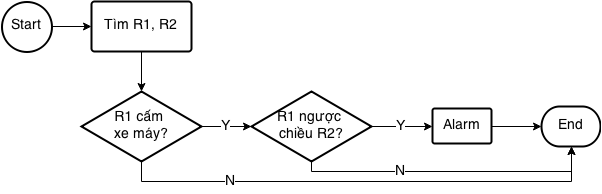
* R1, R2, R3 lần lượt là làn đường hiện tại, làn đường trước đó, làn đường trước của R2
* Lấn tuyến

“Lấn tuyến” là lỗi khi người chơi di chuyển từ làn đường hợp lệ, sang 1 làn đường khác cùng chiều, nhưng không giành cho xe máy chạy.



Hình . Lấn tuyến

Luồng xử lý:



Hình . Sơ đồ logic phát hiện lỗi Lấn tuyến

Trong đó:

* R1, R2 lần lượt là làn đường hiện tại và làn đường trước đó
* Dừng không đúng chỗ

Có các vị trí dừng không đúng bao gồm:

* Dừng trên làn đường xe chạy (không sát mép đường)
* Dừng tại trạm xe buýt
* Dừng giữa giao lộ
* Dừng trên vạch người đi bộ qua đường

## Map Editor

Map Editor là phần mềm chạy độc lập, chạy trên Mac OS, Windows hoặc trên web. Map Editor được dùng để thiết kế bản đồ thành phố, các tuyến đường giao thông, đặt các biển báo, vị trí người chơi, các xe tự động, … Sau đó, phần mềm sẽ xuất ra file theo định dạng JSON để có thể dùng trong Game.

Map Editor sẽ được nhà quản trị nội dung sử dụng để tạo bản đồ, sau đó upload file JSON lên trang CMS và được Game download về để sử dụng.

### Sơ đồ Use case



Hình . Sơ đồ Use case Map Editor

Danh sách Actor:

|  |  |  |
| --- | --- | --- |
| STT | Tên Actor | Ý nghĩa |
| 1 | User | Người dùng |

Bảng . Danh sách Actor trong Map Editor

Danh sách Use case:

|  |  |  |
| --- | --- | --- |
| STT | Tên Use case | Ý nghĩa |
| 1 | New Map | Tạo map mới |
| 2 | Open Map | Mở file map đã lưu |
| 3 | Save Map | Lưu map thành file |
| 4 | Exit | Thoát ứng dụng |
| 5 | Edit Map | Chỉnh sửa map |
| 6 | Edit Map Info | Chỉnh sửa thông tin map |
| 7 | New Tile | Tạo một đối tượng mới trên map |
| 8 | Delete Tile | Xoá một đối tượng trên map |
| 9 | Edit Tile | Chỉnh sửa đối tượng trên map |
| 10 | Change Position | Thay đổi vị trí đối tượng trên map |
| 11 | Change Tile Info | Chỉnh sửa thông tin của đối tượng trên map |

Bảng . Danh sách Use case của Map Editor

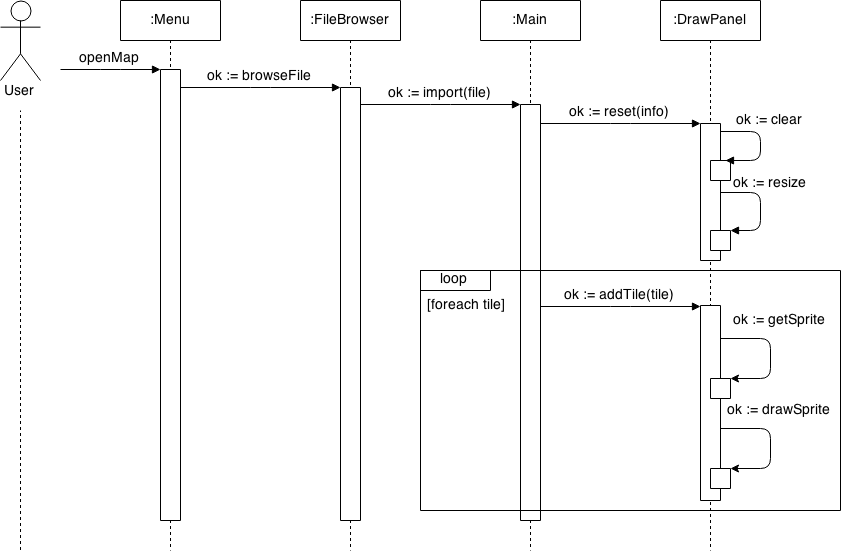
### Sơ đồ tuần tự

#### Sơ đồ tuần tự Use case New Map



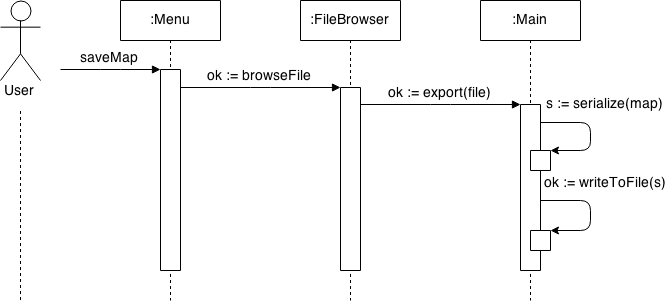
Hình . Sơ đồ tuần tự Use case New Map

#### Sơ đồ tuần tự Use case Open Map



Hình . Sơ đồ Use case Open Map

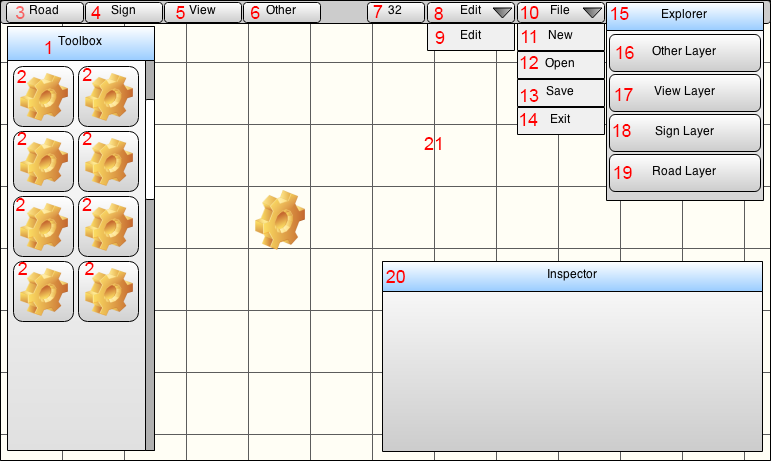
#### Sơ đồ tuần tự Use case Save Map



### Thiết kế giao diện

#### Màn hình chính

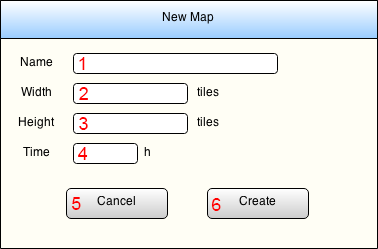
Trang chính là nơi thực hiện tất cả các thao tác thiết kế, xây dựng bản đồ.



Chú giải giao diện:

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Tên | Chức năng | Ghi chú |
| 1 | Toolbox | Hiển thị những đối tượng được dùng để tạo bản đồ | Có thanh cuộn |
| 2 | Tile | Các đối tượng được dùng để tạo bản đồ |  |
| 3 | RoadTab | Tab hiển thị các đối tượng thuộc kiểu “road” |  |
| 4 | SignTab | Tab hiển thị các đối tượng thuộc kiểu “sign” |  |
| 5 | ViewTab | Tab hiển thị các đối tượng thuộc kiểu “view” |  |
| 6 | OtherTab | Tab hiển thị các đối tượng thuộc kiểu “other” |  |
| 7 | Step | Chỉnh sửa giá trị step, dùng trong quá trình di chuyển các đối tượng trên bản đồ |  |
| 8 | MenuEdit | Menu |  |
| 9 | Edit | Chỉnh sửa thông số bản đồ |  |
| 10 | File | Menu |  |
| 11 | New | Tạo bản đồ mới |  |
| 12 | Open | Mở bản đồ đã tạo |  |
| 13 | Save | Lưu bản đồ thành file JSON |  |
| 14 | Exit | Thoát ứng dụng |  |
| 15 | Explorer | Chuyển qua lại giữa các layer đối tượng trên bản đồ |  |
| 16 | Other Layer | Chuyển sang layer Other |  |
| 17 | View Layer | Chuyển sang layer View |  |
| 18 | Sign Layer | Chuyển sang layer Sign |  |
| 19 | Road Layer | Chuyển sang layer Road |  |
| 20 | Inspector | Thông tin chi tiết của đối tượng được chọn |  |
| 21 | Grid | Khu vực tạo bản đồ |  |

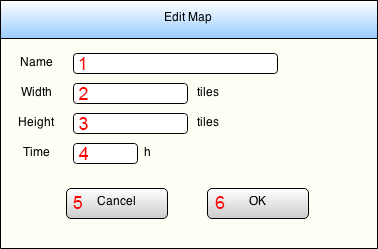
#### Màn hình tạo bản đồ mới



Chú giải giao diện:

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Tên | Chức năng | Ghi chú |
| 1 | Name | Tên bản đồ |  |
| 2 | Width | Chiều rộng bản đồ theo chiều ngang | Đơn vị tile |
| 3 | Height | Chiều dài bản đồ theo chiều dọc | Đơn vị tile |
| 4 | Time | Thời gian ảo trong hệ thống | Đơn vị giờ |
| 5 | Cancel | Huỷ |  |
| 6 | Create | Tạo bản đồ mới |  |

#### Màn hình chỉnh sửa bản đồ



Chú giải giao diện:

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Tên | Chức năng | Ghi chú |
| 1 | Name | Tên bản đồ |  |
| 2 | Width | Chiều rộng bản đồ theo chiều ngang | Đơn vị tile |
| 3 | Height | Chiều dài bản đồ theo chiều dọc | Đơn vị tile |
| 4 | Time | Thời gian ảo trong hệ thống | Đơn vị giờ |
| 5 | Cancel | Huỷ |  |
| 6 | OK | Hoàn tất chỉnh sửa |  |

### Cấu trúc dữ liệu

Map sau khi thiết kế sẽ được lưu lại dưới dạng file JSON. File chứa đầy đủ thông tin về thiết kế, trạng thái sửa đổi của file.

Cấu trúc JSON:

{

    "info": {

//thông tin chung của file

    },

    "layer": {

//dữ liệu bản đồ

    },

    "state": {

//trạng thái sửa đổi

    }

}

Trong đó

#### Info

Phần info chứa thông tin chung về bản đồ, bao gồm:

{

    "name": "Map", //tên bản đồ

    "width": 120, //chiều rộng, đơn vị tile

    "height": 120, //chiều cao, đơn vị tile

    "tile": 32, //kích thước 1 tile

    "simulateTime": 8 //thời gian giả lập

}

#### State

Phần state chứa thông tin trạng thái sửa đổi của bản đồ, dùng để lưu các trạng thái của lần chỉnh sửa trước đó, ví dụ: vị trí sửa, layer được chọn, độ zoom màn hình, … Mục đích là để việc chỉnh sửa các bản đồ lớn đơn giản hơn.

{

    "x": 0, //vị trí

    "y": 0,

    "scale": 1, //độ zoom

    "step": 32, //step để di chuyển đối tượng trong lưới

    "current\_layer": 1, //layer đang được chọn

    "cam\_x": 0, //vị trí camera

    "cam\_y": 0,

    "cam\_z": 0,

    "cam\_size": 2.100 //kích thước camera

}

#### Layer

Phần layer chứa dữ liệu của bản đồ, các đối tượng trong bản đồ được lưu ở đây. Phần Layer chứa trực tiếp các layer có trong bản đồ.

{

    "1": {

        "id": 1, //id của layer

        "name": "Other 1", //tên layer

        "type": "Other", //loại layer

        "tile": { //danh sách các đối tượng

        }

    }

//, ...

}

Trong đó, các đối tượng tile chứa bên trong 1 layer sẽ có cấu trúc sau:

"4": {

    "objId": 4, //id của đối tượng

    "typeId": 310, //loại đối tượng

    "layerType": "Other", //đối tượng thuộc loại layer nào

    "x": -96, //toạ độ

    "y": -192,

    "w": 32, //kích thước

    "h": 46,

    "properties": { //tất cả các thuộc tính của đối tượng

        "key": "value" //lưu dạng key-value

    }

}

### Chủ đề cấp độ 3

Nội dung

### Chủ đề cấp độ 3

#### Chủ đề cấp độ 4

Nội dung

Hình 3.21: Tên hình 1

Bảng 3.5: Tên bảng 1

# TÊN CHƯƠNG 2

## Chủ đề cấp độ 2

### Chủ đề cấp độ 3

#### Chủ đề cấp độ 4

Bảng 4.1: Tên bảng 1

## Chủ đề cấp độ 2

### Chủ đề cấp độ 3

# TÊN CHƯƠNG 3

## Chủ đề cấp độ 2

Nội dung …………………

Nội dung………………….

### Chủ đề cấp độ 3

#### Chủ đề cấp độ 4

## Chủ đề cấp độ 2

TÀI LIỆU THAM KHẢO

Theo chuẩn IEEE