8/5/2013﷽﷽﷽﷽﷽﷽﷽n ﷽﷽﷽à 1**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 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

TÓM TẮT LUẬN VĂN 1

MỞ ĐẦU 3

Chương 1. TỔNG QUAN 5

1.1. Bối cảng nghiên cứu 5

1.2. Động lực nghiên cứu 6

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

2.1.2. Một số Engine trên thị trường 9

2.2. Unity Engine 12

2.2.1. Giới thiệu 12

2.2.2. Các tính năng tiêu biểu 14

2.2.3. Kiến trúc Component 16

2.2.4. Những component trong Unity 18

2.3. Laravel Frameword 22

2.3.1. Giới thiệu 23

2.3.2. Một số ưu điểm 24

2.3.3. Router 25

2.3.4. View 26

2.3.5. Controller 27

2.3.6. Filter 29

**DANH MỤC HÌNH ẢNH**

# 

**DANH MỤC BẢNG**

# TÓM TẮT LUẬN VĂN

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 lần một phần mềm standalone, 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.

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

Trong chín tháng đầu năm 2014, cả nước đã xảy ra 18.697 vụ, 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 cảng 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 39 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 2.3 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 VN 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 |
| 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.

1. 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.

1. Mostly-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.

1. 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 Engine trên thị trường

1. 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.

1. 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.

1. 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.

1. 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 built 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 của “tiền bối” 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.

Unity hay Unity 3D là một engine game đa nền tảng được phát triển bởi Unity Technologies. Engine này được phát triển bằng C/C++ và có khả năng hỗ trợ mã viết bằng C#, JavaScript hoặc Boo. Đây là một trong 4 engine game phổ biến nhất thế giới, hiện đã phát hành bản Unity 3D 4.5.5

### Các tính năng tiêu biểu

Được viết dựa trên ngôn ngữ C# và JavaScript: 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 dụng 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.

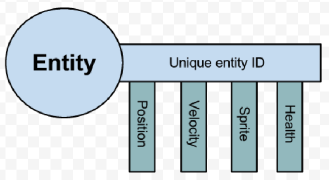
• Screen Space Ambient Occlusin (SSAO): Unity pro sử dụng công nghệ SSAO như là một công cụ tạo hiệu ứng ảnh chuẩn. Đơn giản chỉ cần thêm thành phần này vào bất kỳ góc nhìn nào và chúng ta sẽ đạt được hiệu quả tuyệt vời.

• 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.

### 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 add thêm các component. Mỗi thực thể sẽ mang nhưng tính năng tính chất của component và behavior mà được thêm vào.

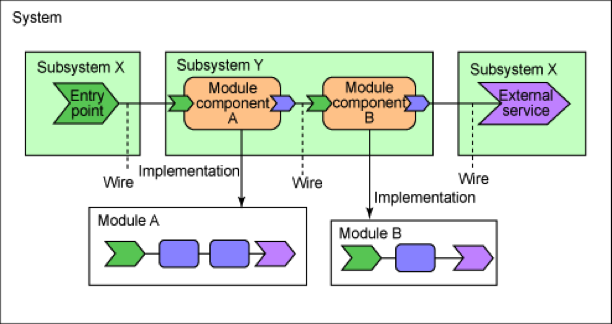


Mỗi component sẽ độc lập với nhau chỉ ảnh hướng tới nhau theo từng liên kết nhất định. Mỗi component và behavior sẽ có một hệ thống xử lý riêng. Chạy trên những nhánh độc lập với nhau.

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



Có thể sử dụng các đặc điểm của OOP nhưng sẽ không phù hợp với kiến trúc của 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.

1. 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.

Hầu như không có cấp độ thừa kế theo cây giống cấu trúc của OOP.

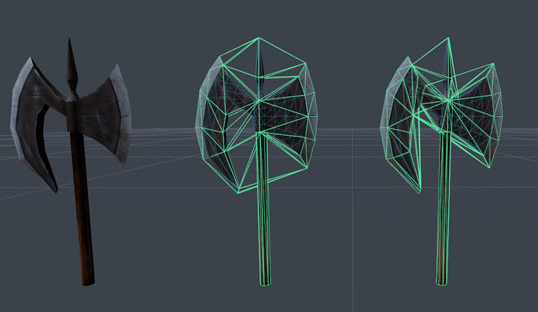
1. Nhược điểm

Không thích hợp với các project nhỏ.

### Những component trong Unity

1. 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**

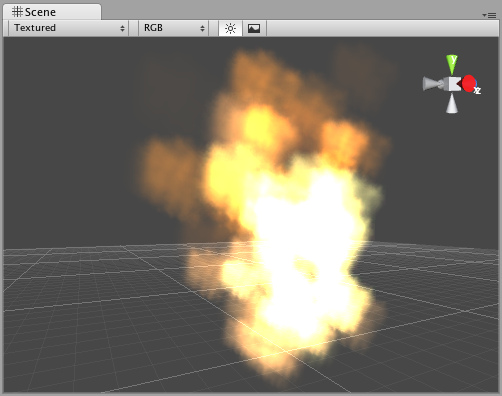
Hiển thị mesh lên không gian 3d

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

1. Nhóm Effect

Tạo ra các effect trong game như: khói, hơi nước, hiệu ứng khí quyển.



Bao gồm:

* **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**

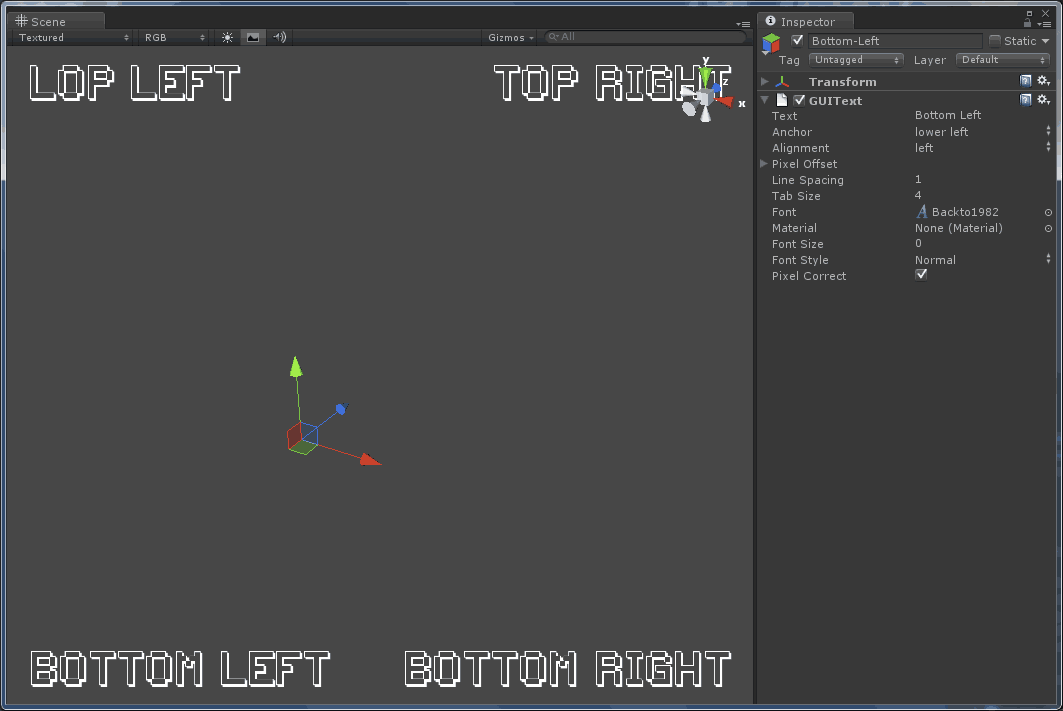
Tạo hiệu ứng vầng sáng xung quanh một đối tượng GameObject. Halo được dùng chủ yếu cho hiệu ứng của một điểm sáng (Point Light).

* **Projector**

Tạo ra hiệu ứng đổ bóng lên một mặt phẳng.

1. Nhóm Renderring

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



Bao gồm:

* **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**

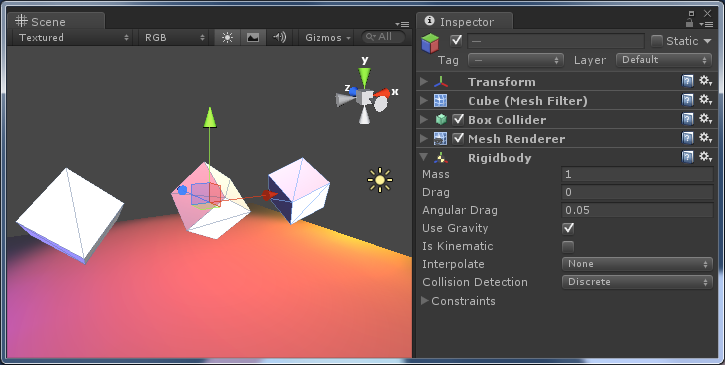
Tạo ra một nguồn sáng, chiếu sáng các thành phần hoặc toàn bộ các GameObject có trong scene.

* **Skybox**

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

1. 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

1. Nhóm Audio

Âm thanh trong game

Bao gồm:

* **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**

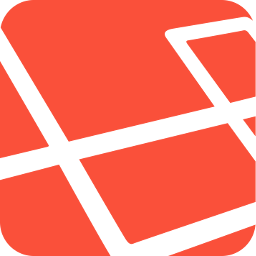
Đại diện cho một đối tượng phát ra âm thanh trong không gian. Trong môi trường 3D, nếu một Audio Source có vị trí càng gần Audio Listener thì âm thanh sẽ càng to hơn.

1. Nhóm Terrain



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

## Laravel Frameword

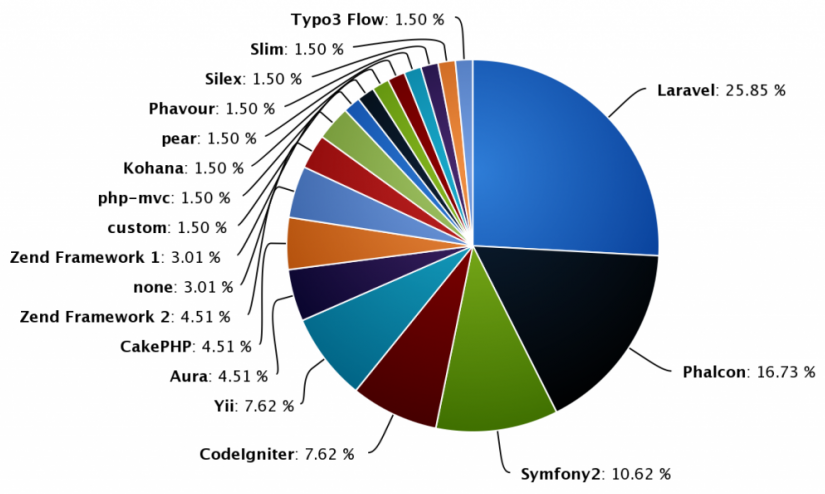


### Giới thiệu

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.



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…

### 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);

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.

Tuy là có 5 method nhưng chúng ta chỉ quan tâm đến 2 method chính là get và post mà thôi.

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.

Một số ví dụ khi người dùng truy cập vào: [http://*localhost/laravel/abc*](http://localhost/laravel/abc)

**Cách 1:**

Route::get("abc",**function**(){

**return** "Hello Laravel";

});

Nội dung hiển thị là:

“Hello Laravel”

**Cách 2:**

Route::get("abc","HelloController@demo");

Action “demo” trong Controller “Hello” sẽ được gọi.

**Cách 3:**

Route::get("abc/{id}",**function**($id){

**return** "Hello $id";

});

Khi người dùng truy cập vào: <http://localhost/laravel/abc/john> thì tham số “john” sẽ được truyền vào hàm xử lý và kết quả sẽ là:

“Hello john”

**Cách 4:**

Route::get("abc/{id?}",**function**($id = null){

**if**($id != null){

**return** "Hello $id";

    }**else**{

**return** "Hello everybody";

    }

})->where(**array**("id"=>"[0-9]+"));

Giá trị mặc định được gán cho biến *id*, đồng thời có thêm phần kiểm soát dữ liệu thông qua hàm *where*.

### View

View là các file .php trong thư mục *app/view.*

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.

Có rất nhiều cách truyền đối ố vào cho view.

**Cách 1:** Sử dụng đối số

Route::get("abc",**function**(){

    $data['id']= "9999";

**return** View::make("abcd",$data);

});

Gọi tới view tên *abcd*, trong abcd, có thể truy cập trực tiếp biến *$id*

**Cách 2:** Truyền qua phương thức *with*

Route::get("abc",**function**(){

**return** View::make("abcd")->with(**array**("id"=>"9999"));

});

Nhược điểm là truyền nhiều đối số không được.

**Cách 3:** Dùng hàm *compact* trong PHP

Route::get("abc",**function**(){

    $id=9999;

**return** View::make("abcd",compact("id"));

});

Để gọi vào view nằm trong thư mục con, ta có thể gọi như sau:

Route::get("abc",**function**(){

**return** View::make("demo.abcd");

});

File *abcd.php* nằm trong thư mục *app/view/demo.*

### Controller

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.

Trong trường hợp ta muốn truyền tham số vào cho Controller từ Route, ta thực hiện như sau:

Trước hết, ta cần khai báo trong Route đối số ta mong muốn:

Route::get("demo/content/{id}","DemoController@getContent");

Sau đó ta tạo thêm phương thức *getContent()* với đối số là $id như sau:

<?php

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

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

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

    }

**public** **function** getContent($id){

**return** "Your ID: $id";

    }

}

### 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.

Để bắt tay vào filter ta thiết lập filter như sau trong file *filters.php*

Route::filter('tên',hàm xử lý);

Ví dụ:

Route::filter('check\_time',**function**($route,$request){

**if**(date('A') == "AM"){

**return** "Morning";

    }

});

Đoạn filter này thực hiện công việc kiểm tra thời gian hiện tại có phải là ban ngày hay không. Hàm *date('A')* với đối số là A nghĩa là thời gian sẽ trả về AM hoặc PM. Dựa vào đoạn code ở trên, nếu thời gian từ hệ thống là AM, nó sẽ chặn yêu cầu tới routing và xuất thông báo "Ban ngày".

Giờ ta tiến hành gắn filter vào routing nào. Mở file *routes.php* thêm vào cú pháp sau:

Route::get("demo/content/{id}",**array**("as"=>"content","before"=>"check\_time","uses"=>"DemoController@getContent"));

Từ khóa *before* chỉ tới *check\_time* chính là filter ta đã thiết lập trước đó, và đẩy nó sang controller demo với action là getContent.

Tạo tiếp file *DemoController.php* với nội dung sau:

<?php

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

**public** **function** getContent($id){

**return** "Your ID: $id ";

    }

}

Và chạy đường link *localhost/laravel/public/demo/content/999* sẽ thấy kết quả hiển thị phù thuộc theo khung thời gian của máy.

Qua điều này, ta thấy được ý nghĩa và sức mạnh của filter trong thực tế. Với filter ta có thể tối giản code ở mức cao nhất và tái sử dụng vô cùng đơn giản.