**ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

**KHOA CÔNG NGHỆ PHẦN MỀM**

**PHẠM VŨ THANH PHONG**

**BÀNH LÊ VŨ THIỆN**

**KHÓA LUẬN TỐT NGHIỆP**

**XÂY DỰNG VÀ PHÁT TRIỂN HỆ THỐNG CHALLENGE GAME TRÊN NỀN TẢNG THIẾT BỊ DI ĐỘNG VỚI SPEED RUNNER**

**KỸ SƯ/ CỬ NHÂN NGÀNH KỸ THUẬT PHẦN MỀM**

**TP. HỒ CHÍ MINH, 2015**

**ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

**KHOA CÔNG NGHỆ PHẦN MỀM**

**PHẠM VŨ THANH PHONG – 11520286**

**BÀNH LÊ VŨ THIỆN – 11520378**

**KHÓA LUẬN TỐT NGHIỆP**

**XÂY DỰNG VÀ PHÁT TRIỂN HỆ THỐNG CHALLENGE GAME TRÊN NỀN TẢNG THIẾT BỊ DI ĐỘNG VỚI SPEED RUNNER**

**KỸ SƯ/ CỬ NHÂN NGÀNH KỸ THUẬT PHẦN MỀM**

**GIẢNG VIÊN HƯỚNG DẪN**

**PGS.TS . VŨ THANH NGUYÊN**

**TP. HỒ CHÍ MINH, 2015**

DANH SÁCH HỘI ĐỒNG BẢO VỆ KHÓA LUẬN

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

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

MỤC LỤC

[MỤC LỤC iv](#_Toc440837368)

[DANH MỤC HÌNH vi](#_Toc440837369)

[DANH MỤC BẢNG vii](#_Toc440837370)

[DANH MỤC TỪ VIẾT TẮT viii](#_Toc440837371)

[TÓM TẮT KHÓA LUẬN 1](#_Toc440837372)

[LỜI MỞ ĐẦU 2](#_Toc440837373)

[CHƯƠNG 1: GIỚI THIỆU VỀ ĐỀ TÀI 3](#_Toc440837374)

[1.1 Lý do chọn đề tài . 3](#_Toc440837375)

[1.2 Mục tiêu đề tài 4](#_Toc440837376)

[1.3 Ý nghĩa đề tài 4](#_Toc440837377)

[1.3.1 Đối với người chơi nói riêng và ngành công nghiệp game nói chung 4](#_Toc440837378)

[1.3.2 Đối với bản thân nhóm 4](#_Toc440837379)

[1.4 Đối tượng nghiên cứu: 5](#_Toc440837380)

[1.5 Phạm vi nghiên cứu: 5](#_Toc440837381)

[1.5.1 Ngôn ngữ C++ và engine Cocos2d-x: 5](#_Toc440837382)

[1.5.2 Ngôn ngữ Java và Jetty serverlet container: 5](#_Toc440837383)

[1.5.3 Redis 5](#_Toc440837384)

[CHƯƠNG 2: CƠ SỞ LÝ THUYẾT 6](#_Toc440837385)

[2.1 Client side 6](#_Toc440837386)

[2.1.1 Cocos2d-x. 6](#_Toc440837387)

[2.1.2 Cocos studio 24](#_Toc440837388)

[2.1.3 Quadtree 25](#_Toc440837389)

[2.2 Server side 25](#_Toc440837390)

[2.2.1 HTTP và socket 25](#_Toc440837391)

[2.2.2 Jetty 26](#_Toc440837392)

[2.2.3 Redis 28](#_Toc440837393)

[2.2.4 JSON 34](#_Toc440837394)

[2.3 Các khái niệm liên quan. 35](#_Toc440837395)

[2.3.1 Tổng quan về Mã nguồn mở. 35](#_Toc440837396)

[CHƯƠNG 3: HIỆN THỰC HÓA 40](#_Toc440837397)

[3.1 Client-side 40](#_Toc440837398)

[3.1.1 Các use case chính 40](#_Toc440837399)

[3.1.2 Các màn hình 44](#_Toc440837400)

[3.2 Server-side 55](#_Toc440837401)

[3.2.1 Kiến trúc 55](#_Toc440837402)

[3.2.2 Tầng web service 57](#_Toc440837403)

[3.2.3 Tầng business logic 58](#_Toc440837404)

[3.2.4 Tầng data access layer 58](#_Toc440837405)

[3.2.5 Tầng database 60](#_Toc440837406)

[CHƯƠNG 4: THỬ NGHIỆM 61](#_Toc440837407)

[4.1 Quy trình thử nghiệm. 61](#_Toc440837408)

[4.1.1 Các thành phần của hệ thống thử nghiệm. 61](#_Toc440837409)

[4.1.2 Tiến hành thử nghiệm. 61](#_Toc440837410)

[4.2 Một vài hình ảnh kết quả thực nghiệm 61](#_Toc440837411)

[4.2.1 Màn hình chơi game 61](#_Toc440837412)

[4.2.2 Màn hình chỉnh sửa map 63](#_Toc440837413)

[4.2.3 Màn hình bảng xếp hạng map 64](#_Toc440837414)

[CHƯƠNG 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 65](#_Toc440837415)

[5.1 Kết quả đạt được 65](#_Toc440837416)

[5.1.1 Đối với đề tài 65](#_Toc440837417)

[5.1.2 Đối với các thành viên trong nhóm 65](#_Toc440837418)

[5.2 Hạn chế 65](#_Toc440837419)

[5.3 Hướng phát triển 65](#_Toc440837420)

[5.3.1 Về mặt chức năng 65](#_Toc440837421)

[5.3.2 Về mặt kỹ thuật 66](#_Toc440837422)

[5.4 Những khó khăn và cách giải quyết 66](#_Toc440837423)

[5.5 Những bài học thu được 66](#_Toc440837424)

[TÀI LIỆU THAM KHẢO 67](#_Toc440837425)

DANH MỤC HÌNH

[Hình 2.1 Kiến trúc Cocos2d-x 9](#_Toc441345918)

[Hình 2.2 Khám phá các lớp của Cocos2d-x 10](#_Toc441345919)

[Hình 2.3 Chụp màn hình của một game Cocos2d-x 10](#_Toc441345920)

[Hình 2.4 Sơ đồ cây của một màn 11](#_Toc441345921)

[Hình 2.5 Vị trí của một nút con khi được thêm vào nút cha 12](#_Toc441345922)

[Hình 2.6 Sơ đồ các lớp kế thừa từ CCNode 13](#_Toc441345923)

[Hình 2.7 Sơ đồ lớp kế thừa từ CCAction 15](#_Toc441345924)

[Hình 2.8 Sơ đồ lớp CCActionIntervel 16](#_Toc441345925)

[Hình 2.9 Sơ đố các lớp kế thừa từ CCActionEase 17](#_Toc441345926)

[Hình 2.10 Sơ đồ các lớp kế thừa từ CCTilesGrid3Daction 18](#_Toc441345927)

[Hình 2.11 Sơ đồ các lớp kế thừa từ CCGrid3DAction 18](#_Toc441345928)

[Hình 2.12 Hiệu ứng lật trang 19](#_Toc441345929)

[Hình 2.13 Sơ đồ các lớp kế thừa từ CCActionInstant 19](#_Toc441345930)

[Hình 2.14 So sánh cách lưu trữ dữ liệu giữa NoSQL và RDBMS 30](#_Toc441345931)

[Hình 2.15 Chi phí để mở rộng hệ thống sử dụng 31](#_Toc441345932)

[Hình 2.16 Hệ thống cơ sở dữ liệu NoSQL dễ dàng 32](#_Toc441345933)

[Hình 3.1 Sơ đồ Use-case 40](#_Toc441345934)

[Hình 3.2 Sơ đồ màn hình 44](#_Toc441345935)

[Hình 3.3 Màn hình chính (1) 45](#_Toc441345936)

[Hình 3.4 Màn hình bảng xếp hạng 46](#_Toc441345937)

[Hình 3.5 Màn hình khi có sẵn map 48](#_Toc441345938)

[Hình 3.6 Màn hình thêm mới map 49](#_Toc441345939)

[Hình 3.7 Màn hình chỉnh sửa map 50](#_Toc441345940)

[Hình 3.8 Màn hình chơi game chính 52](#_Toc441345941)

[Hình 3.9 Màn hình chơi game khi tạm dừng 53](#_Toc441345942)

[Hình 3.10 Màn hình chơi game khi hoàn thành thử map 54](#_Toc441345943)

[Hình 3.11 Mô hình giản lược 56](#_Toc441345944)

[Hình 3.12 Mô hình chi tiết 57](#_Toc441345945)

[Hình 4.1 Màn hình chơi game không sử dụng quadtree 62](#_Toc441345946)

[Hình 4.2 Màn hình chơi game sử dụng quadtree 62](#_Toc441345947)

[Hình 4.3 Màn hình chỉnh sửa map 63](#_Toc441345948)

[Hình 4.4 Màn hình bảng xếp hạng map 64](#_Toc441345949)

DANH MỤC BẢNG

[Bảng 2.1 So sánh Cocos với các engine khác 23](#_Toc441345217)

[Bảng 2.2 Các tính năng của Jetty 28](#_Toc441345218)

[Bảng 2.3 So sánh các application server 28](#_Toc441345219)

[Bảng 2.4 So sánh giữa NoSQL và RDBMS [15] 29](#_Toc441345220)

[Bảng 2.5 So sánh các nosql database 34](#_Toc441345221)

[Bảng 3.1 Danh sách Actor 40](#_Toc441345222)

[Bảng 3.2 Danh sách use-case 41](#_Toc441345223)

[Bảng 3.3 Đặc tả use-case create map 41](#_Toc441345224)

[Bảng 3.4 Đặc tả use-case modify map 42](#_Toc441345225)

[Bảng 3.5 Đặc tả use-case upload map 42](#_Toc441345226)

[Bảng 3.6 Đặc tả use-case play game 43](#_Toc441345227)

[Bảng 3.7 Đặc tả use-case rate map 43](#_Toc441345228)

[Bảng 3.8 Bảng chú thích cho màn hình chính 45](#_Toc441345229)

[Bảng 3.9 Bảng chú thích cho màn hình bảng xếp hạng 47](#_Toc441345230)

[Bảng 3.10 Bảng chú thích cho màn hình khi có sẵn map 48](#_Toc441345231)

[Bảng 3.11 Bảng chú thích cho màn hình thêm mới map 49](#_Toc441345232)

[Bảng 3.12 Bảng chú thích cho màn hình chỉnh sửa map 51](#_Toc441345233)

[Bảng 3.13 Bảng chú thích cho màn hình chơi game chính 52](#_Toc441345234)

[Bảng 3.14 Bảng chú thích cho màn hình chơi game khi tạm dừng 53](#_Toc441345235)

[Bảng 3.15 Bảng chú thích cho màn hình chơi game khi hoàn thành thử map 54](#_Toc441345236)

[Bảng 3.16 So sánh các kiến trúc server 55](#_Toc441345237)

[Bảng 3.17 Bảng so sánh set, sorted set, list, vector 59](#_Toc441345238)

[Bảng 5.1 Khó khăn và cách giải quyết 66](#_Toc441345239)

DANH MỤC TỪ VIẾT TẮT

|  |  |  |
| --- | --- | --- |
| TỪ VIẾT TẮT | TỪ ĐẦY ĐỦ | ĐỊNH NGHĨA TIẾNG VIỆT |
| 2D | Two - dimensional space | Không gian hai chiều |
| GDI | Graphics Device Interface | Giao diện đồ họa thiết bị |
| GPL | General Public License | Giấy phép cộng đồng chung |
| BSD | Berkeley Software Distribution | Giấy phép phân phối phần mềm ở Berkeley |

TÓM TẮT KHÓA LUẬN

Đề tài được nguyên cứu xoay quanh trọng tâm chính là hệ thống challenge game Speed Runner. Hệ thống bao gồm 4 thành phần là:

+ Một game thể loại runner đa nền tảng

+ Map editor tool cho phép người dùng tạo map mới, upload, chia sẻ với bạn bè

+ Webserver làm nhiệm vụ tiếp nhận các request upload cũng như download map.

+ Database server để lưu trữ các map đã được upload lên.

Với SpeedRunner, người chơi có thêm lựa chọn game mới, không những thế , họ còn có thể tạo ra những map mới theo phong cách riêng của họ.

LỜI MỞ ĐẦU

Ngành công nghiệp game đã và đang phát triển không ngừng, nhờ sự tiến bộ và phát triển của khoa học công nghệ, sản phẩm công nghệ cao. Từng giây trôi qua, các studio game cạnh tranh khốc liệt nhằm tạo ra các sản phẩm chất lượng với những ý tưởng sáng tạo đột phá, dường như bất kì ý tưởng nào đều được các studio tận dụng triệt để.

Tuy nhiên, dường như các studio đã quá “tham lam” khi giành hết phần việc sáng tạo về phía mình mà không cho người chơi tham gia vào quá trình sáng tạo.

Luận văn “Xây dựng và phát triển hệ thống challenge game trên nền tảng thiết bị di động với Speed Runner”, với mong muốn cung cấp 1 hướng đi mới, giúp người chơi thỏa chí sáng tạo. Đề tài góp phần làm cho quá trình phát triển game linh hoạt hơn, phổ thông hơn, chiến lược hơn và cộng đồng hơn.

Nội dung được trình bày trong 5 chương. Cụ thể, Chương I trình bày khái quát về đề tài. Chương II đề cập đến các lý thuyết cơ sở liên quan đến đề tài như Cocos2d-x, phần mềm mã nguồn mở, nosql database, jetty. Chương III trình bày quá trình hiện thực hóa. Chương IV Kết quả thực nghiệm và đánh giá. Chương VI Kết luận - Hướng phát triển được trình bày nhằm đúc kết những thiếu sót của đề tài đối với các vấn đề hiện tại đang giải quyết, cũng như các bài học rút ra, khó khăn gặp phải khi thực hiện khóa luận.

# GIỚI THIỆU VỀ ĐỀ TÀI

## Lý do chọn đề tài

Video game đã xuất hiện từ những năm 1950, nhưng đến những năm 1960 thì ngành công nghiệp game mới đặt nền móng cho việc phát triển sau này. Dù trong giai đoạn phát triển của ngành công nghiệp non trẻ, ngành công nghiệp Game vẫn khẳng định mình trong giai đoạn giải trí của con người. Bắt đầu phát triển và trở nên thịnh vượng từ năm 1950 đến đầu những năm 1970, suy sụp đầu tiên năm 1977 và cơn khủng hoảng kinh hoàng năm 1983. Sau đó, ngành công nghiệp này đã có nhiều bước chuyển biến tích cực trên thị trường và vương lên chiếm thị phần của các ngành công nghiệp già cỗi khác.

Tính đến đầu thế kỷ 21, nhiều thể loại game và dòng game đã được ra mắt thị trường với nhiều sản phẩm nổi bật. Tuy đa dạng về thể loại game và số lượng game, tuy nhiên, đại đa số các game đều có 1 điểm chung là bó buộc người chơi theo 1 kịch bản sẵn có, điển hình nhất là dòng game platform, điều này đôi khi gây cho người chơi cảm giác ức chế, chán nản. Tuy nhiên, cũng có 1 số game cho phép người chơi sáng tạo, làm ra cái mới, có thể kể đến như:

+ Minecraft, cho phép người chơi có thể tạo ra mọi thứ trên đời từ phi thuyền không gian, tới cả 1 thành phố, thậm chí là tựa game pacman huyền thoại.

+ Warcraft, với hệ thống map editor mạnh mẽ đi cùng, người chơi đã có thể tạo ra rất nhiều custom map khác nhau, điển hình nhất là DOTA, tựa game đánh dấu sự ra đời của dòng game rất được ưa thích hiện tại MOBA.

+ Super Mario maker: hệ thống tạo và chia sẻ map của game mario trên hệ máy Wii mới ra tháng 9/2015.

Các game trên đều đạt được những thành công vượt trội, cho thấy rằng việc cho phép người chơi sáng tạo trong game là 1 điều đáng để cân nhắc.

Ngoài ra, có 1 điều đáng để lưu tâm nữa đó là: nền tảng phát triển. Trong 3 game vừa kể trên, ngoài minecraft có thể chơi trên nhiều nền tảng, thì warcraft do đặc trưng thể loại game nên hầu như chỉ có thể chơi trên PC, trong khi Super Mario maker lại chỉ chơi được trên Wii; điều này vô hình chung lại là 1 điểm trừ lớn khi mà trong những năm gần đây, các thiết bị di động( điện thoại di động, máy tính bảng và các thiết bị di động khác) đã và đang phát triển mạnh mẽ với những con số ấn tượng: 4 tỉ người sử dụng điện thoại di động, 86% người sử dụng điện thoại di động truy cập Internet trong khi đang xem TV, theo dự đoán, năm 2015, lượng người sử dụng điện thoại di động để truy cập Internet sẽ vượt qua lượng người dùng máy tính cá nhân.

Với ưu thế nhỏ gọn có thể giúp người dùng giải trí khắp mọi nợi, theo thống kê, số người sử dụng điện thoại di động để chơi game chiếm tới 61%, dẫn tới nhu cầu chơi game trên thiết bị di động cực kì lớn.

Từ những điều trên, nhận thấy việc xây dựng game đa nền tảng hỗ trợ người chơi sáng tạo, đang là 1 khía cạnh còn nhiều trống vắng, với mong muốn đóng góp cho nền công nghiệp game cũng như ứng dụng các kiến thức đã học, nhóm quyết định thực hiện đề tài này.

## Mục tiêu đề tài

Mục tiêu của đề tài là xây dựng 1 game đa nền tảng gắn kèm theo hệ thống map editor cho phép người chơi tự tạo map, lưu trữ cũng như chia sẻ, thách thức bạn bè của họ.

## Ý nghĩa đề tài

### Đối với người chơi nói riêng và ngành công nghiệp game nói chung

Tạo ra 1 game mới, làm tăng phần phong phú trong kho game hiện tại.

### Đối với bản thân nhóm

* Giúp trau dồi đồng thời nâng cao kiến thức về lập trình
* Ứng dụng những gì đã học được vào quá trình làm game
* Có được những kinh nghiệm quý báu về làm game client - server

## Đối tượng nghiên cứu:

* Một runner-game và map editor xây dựng trên ngôn ngữ C++ và nền tảng cocos2dX.
* Webserver xây dựng trên ngôn ngữ Java sử dụng Jetty.
* Database server được xây dựng bằng Redis .

## Phạm vi nghiên cứu:

### Ngôn ngữ C++ và engine Cocos2d-x:

* Tìm hiểu về cấu trúc và kỹ thuật của engine Cocos2dx-x thực thi các hành động, hiệu ứng, xử lý thông dụng và cần thiết cho game trên di động, máy vi tính.
* Tìm hiểu và ứng dụng các kỹ thuật lập trình hướng đối tượng và cấu trúc ngôn ngữ C++, cùng các bộ thư viện cơ bản.
* Nghiên cứu các cấu trúc xây dựng một game client- server thông dụng và ứng dụng của các cấu trúc framework game của các công trình đi trước
* Nghiên cứu quadtree vào việc tối ưu performance

### Ngôn ngữ Java và Jetty serverlet container:

* Tìm hiểu các Java util class như Map, Set, List, Random
* Nghiên cứu cách tạo 1 web server với Jetty
* JSON

### Redis

* Nghiên cứu cấu trúc Redis nosql database server
* Nghiên cứu và ứng dụng Jedis, redis client cho java

# CƠ SỞ LÝ THUYẾT

## Client side

### Cocos2d-x.

#### Giới thiệu

Theo ông Gustavo E Rangel, trong cuốn “Cocos2d-X 3.0: Game Development Programming for iOS and Android” xuất bản năm 2014, Cocos2d-x là một engine mã nguồn mở đa nền tảng dưới giấy phép MIT được phát triển dựa trên nền tảng engine Cocos2d. Cocos2d-x được sử dụng để xây dựng các trò chơi, và ứng dụng đa nền tảng như iOS, Andriod, Windows Phone và nhiều nền tảng khác. Người lập trình có thể sử dụng ngôn ngữ C++, JavaScript và Lua để phát triển ứng dụng. Engine đã làm việc rất hiệu quả; ít nhất 25% game di động trên thế giới được hỗ trợ bởi Cocos2d-x. Ưu điểm chính của engine là chỉ với một mã nguồn duy nhất, được chuyển sang nhiều nền tảng khác nhau mà không cần tốn bất kỳ chi phí nào khác.

Cocos2d-x được thiết kế và xây dựng trên lớp cấu trúc kế thừa chặt chẽ chạy trên nền OpenGL 2.0 và OpenGL ES 2.0. Với các thành phần tiêu biểu, Lớp CCScene tạo nên một cảnh trong game, hệ thống các cảnh tạo nên cấu trúc game. CCDirector Lớp singleton, được truy xuất toàn cục bằng cái gọi getInstance, dùng để quản lý các cảnh (CCScene) trong game. CCNode yếu tố cơ bản của Cocos2d-x, là một lớp trừu tượng và là lớp cơ sở cho tất cả các nút trong cấu trúc Cocos2d-x như: CCScene, CCLayer, CClabelTTTF, CCSprite, CCMenuItem. CCAction Lớp được sử dụng để thực hiện các thao tác di chuyển, thay đổi kích thước, xoay, làm mờ và nhiều hiệu ứng khác trên một nút. CCTransitionScene giúp chuyển đổi giữ các cảnh trở nên đẹp hơn và chuyên nghiệp hơn.

#### Giấy phép

Dựa trên tài liệu được công khai tại trang chính thức của Cocos2d-x - [http://www.cocos2d-x.org/reference/native-cpp/V3.3/index.html](http://www.cocos2d-x.org/reference/native-cpp/V3.3/index.html%20năm%202014) năm 2014: “Bản quyền © 2010-2012 cocos2d-x.org

Giấy phép này được cấp, hoàn toàn miễn phí, để bất kỳ người nào cũng có thể có được một bản sao của phần mềm này và các tài liệu liên quan (“Phần mềm”), để không bị hạn chế khi làm việc với phần mềm, bao gồm các quyền không giới hạn về việc sử dụng, sao chép, sửa đổi, sáp nhập, xuất bản, phân phối, cấp phép, và / hoặc bán các bản sao của phần mềm, và cho phép người đã trang bị phần mềm thực hiện các quyền trên, nhưng phải tuân theo các điều kiện sau đây:

Thông báo bản quyền ở trên và Giấy phép thông báo này phải được kèm trong tất cả các bản sao hoặc phần quan trọng của phần mềm.

PHẦN MỀM ĐƯỢC CUNG CẤP “NHƯ LÀ” (“AS IS”), KHÔNG ĐẢM BẢO VỚI MỌI HÌNH THỨC, RÕ RÀNG HAY NGỤ Ý, BAO GỒM KHÔNG GIỚI HẠN CÁC ĐẢM BẢO THƯƠNG MẠI, PHÙ HỢP VỚI MỘT MỤC ĐÍCH CỤ THỂ VÀ KHÔNG VI PHẠM. TRONG BẤT CỨ TRƯỜNG HỢP NÀO TÁC GIẢ HOẶC NGƯỜI GIỮ BẢN QUYỀN SẼ KHÔNG CHỊU TRÁCH NHIỆM CHO BẤT CỨ YÊU CẦU, THIỆT HẠI HOẶC TRÁCH NHIỆM KHÁC, DÙ TRONG MỘT HOẠT ĐỘNG HỢP ĐỒNG, SAI PHẠM (TORT) HOẶC NHỮNG VẤN ĐỀ KHÁC, PHÁT SINH TỪ BÊN NGOÀI HOẶC LIÊN QUAN ĐẾN PHẦN MỀM HAY SỬ DỤNG HAY NHỮNG HOẠT ĐỘNG KHÁC TRONG PHẦN MỀM.”

#### Các tính năng

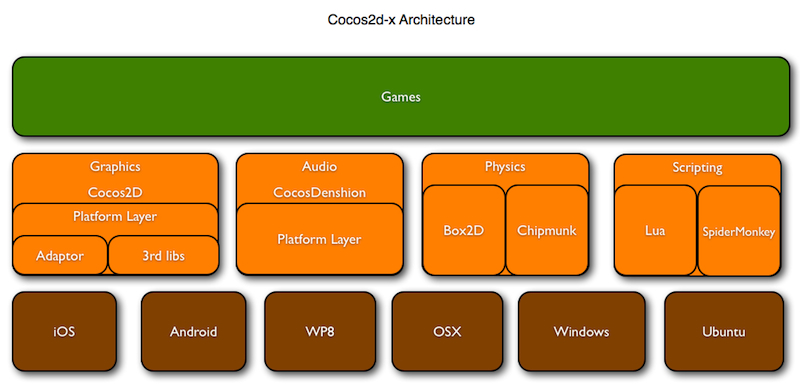
Cocos2d-x được sử dụng bởi các cá nhân và các tổ chức được biết như các công ty lớn. Ngày nay, nhiều game được viết bằng Cocos2d-x đang thống trị bảng doanh thu cao nhất của AppStore và Google Play, đặc biệt tại các quốc gia: Trung Quốc, Hàn Quốc và Nhật Bản. Những kỹ sư từ ChuKong, Google, Microsoft, ARM, Intel và BlackBerry đang hoạt động rất tích cực trên cộng đồng Cocos2d-x. Những ông chủ lớn như Zynga, Wooga, Glu, IGG, Big Fish Games, Fingersoft, Gamevil, GREE, DeNA, Konami, CJ E&M, NHN, LINE, Square Enix, Disney mobile đã sử dụng Cocos2d-x để phát triển game của mình.

Nguyên nhân chính để chọn Cocos2d-X cho luận văn bởi các yếu tố:

* API hiện đại dành cho C++.
* Đa nền tảng – cả di động và máy để bàn.
* Có thể kiểm tra và tìm lỗi của game trên máy tính để bàn trước khi thật sự đưa nó đến với các đối tượng di động hoặc máy tính để bàn hướng đến.
* Hệ thống API lớn với đầy đủ chức năng bao gồm hình cho game (sprites), hành động, diễn hình (animation), biễu diễn hạt (particles), chuyển (transitions), thời gian (timers), sự kiện (events) ( chạm, bàn phím, cảm biến gia tốc, chuột), âm thanh, Nhập xuất tập tin, chuyển động có khung xương, 3D.
* Khung xương chuyển động
* Kiểu chữ:
* Vẽ kiểu chữ nhanh sử dụng cơ chế cố định (fixed) và chiều dài cho kiểu chữ
* Hỗ trợ nhiều kiểu font chữ và định dạng .ttf
* Hỗ trợ Bản đồ theo ô (Tile Map): Tứ phương, tạo hiệu ứng 3D (Isometric) và lục phương.
* Cuộn màn hình
* Chuyển động đường sọc
* Tạo Texture
* Chạm và cảm biến gia tốc trên di động
* Chạm, Chuột, Bàn phím trên máy tính để bàn
* Engine hỗ trợ âm thanh (thư viện CocosDeshion) dựa trên OpenAL
* Tích hợp chuyển động chậm/ Nhanh về phía trước
* Nhanh và nén Texture
* PVR nén và giải nén texture
* ETC1 nén texture và nhiều hơn nữa
* Giải pháp độc lập
* Giao diện tiện ích cho người dùng (UI Widgets).
* Bố trí theo trục tung và trục hoành
* Các đối tượng chứa: ListView, PageView, ScrollView
* Tiện ích: Nút, Khung đánh dấu, Thanh tải, Thanh trượt, khung chữ
* Vật lý.
* Tích hợp Chipmunk2D bên trong
* Tích hợp Box2D bên trong
* Dễ sử dụng với lớp bọc hướng đối tượng với cơ thể, tiếp xúc, điểm nối, các khối hình
* Âm thanh.
* Hiệu ứng âm thanh
* Âm thanh nền
* Mạng.
* HTTP với SSL
* WebSocket API
* XMLHttpRequest API

#### Kiến trúc engine Cocos2d-x

Kiến trúc Cocos2d-x - www.cocos2d-x.org/wiki/Engine\_Architecture (2014):



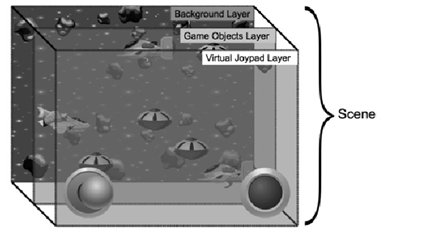
Hình . Kiến trúc Cocos2d-x

Kiến trúc trên thể hiện tính đa nền tảng của Cocos2d-x với khả năng chạy trên iOS, Android, WP8, OSX, Windows, Ubuntu nhờ các tầng adaptor và 3rd libs được bao bọc bởi các Platform Layer. Riêng về các yếu tố như vật lý và mã kịch bản, Cocos2d-x được hỗ trợ bởi Box2D và chipmunk hay lua và spiderMonkey nên thuận tiện cho việc đa nền tảng.

##### CCScene

Tất cả các đối tượng hiển thị trong game, cocos2d-x đều xem đó như là một nút và các nút đó được kế thừa bởi lớp CCNode. Để hiển thị bất cứ đối tượng nào trong game như hình (sprite) hay chữ (label) thì chúng phải được xem như là một nút và được quản lý bởi nút gốc là CCScene.

Giả sử như đây là một màn hình khi chơi game:



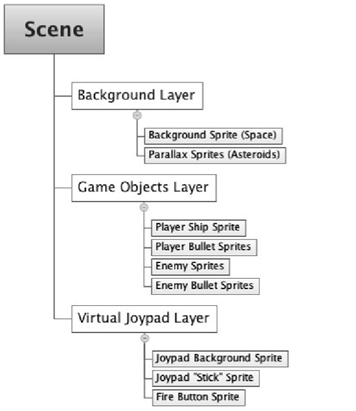
Hình . Khám phá các lớp của Cocos2d-x



Hình . Chụp màn hình của một game Cocos2d-x

Như trong hình, cảnh game này hiện tại có 3 lớp chính là Nền (Background), Đối tượng (Game Objects) và Lớp điều khiển (Virtual Joypad). Trong đó để hiển thị các thiên thạch ở lớp nền, các phi thuyền và điều khiển thì CCScene trước tiên được tạo ra và quản lý bởi CCDirector. Sau đó, CCScene sẽ thêm các nút con là Lớp nền, Lớp đối tượng, Lớp điều khiển. Các lớp này lần lượt thêm các nút con là hình thiên thạch cho lớp nền, hình phi thuyền cho lớp đối tượng và hình khung điều khiển cho lớp điều khiển.

Sơ đồ cây của lớp Scene:



Hình . Sơ đồ cây của một màn

Sử dụng nhiều lớp trong một cảnh cũng giúp quản lý các nút trở nên dễ dàng hơn. Người dùng có thể ẩn hoặc di chuyển đi toàn bộ nút trong cùng cảnh. Sắp xếp lại các lớp được tạo nên thứ tự ưu tiên khi vẽ và thậm chí khi xoay hoặc thay đổi tỉ lệ cũng sẽ ảnh hưởng đến toàn bộ các nút bị nút đó ảnh hưởng.

Cocos2d-x không quy định một cảnh (scene) phải có lớp mà có thể dùng một nút (Node) để thay cho lớp (layer) nhưng tốt nhất nên có ít nhất có 1 lớp (layer) được chứa bởi 1 cảnh (scene) và lớp này sẽ chứa các nút khác như hình, chữ, khung chọn, hộp thoại, vv.

Mỗi nút đều có thể chứa 1 nút khác và mỗi nút trong hệ thống cây, trừ nút cảnh (scene), đều có một nút cha. Các nút hoàn toàn quan hệ với nhau như một cây và không có quan hệ kế thừa đối với các nút cha, con.

Điểm khác biệt lớp nhất của CCScene với CCNode chỉ đơn giản chỉ là nút trừu tượng chỉ được dùng làm nút gốc. Và cũng như vậy, CCLayer chỉ đơn giản là có thêm các tương tác với các thành phần đầu vào như chạm, cảm biến giá tốc, ngoài ra, CCLayer không có điểm gì khác với CCNode.

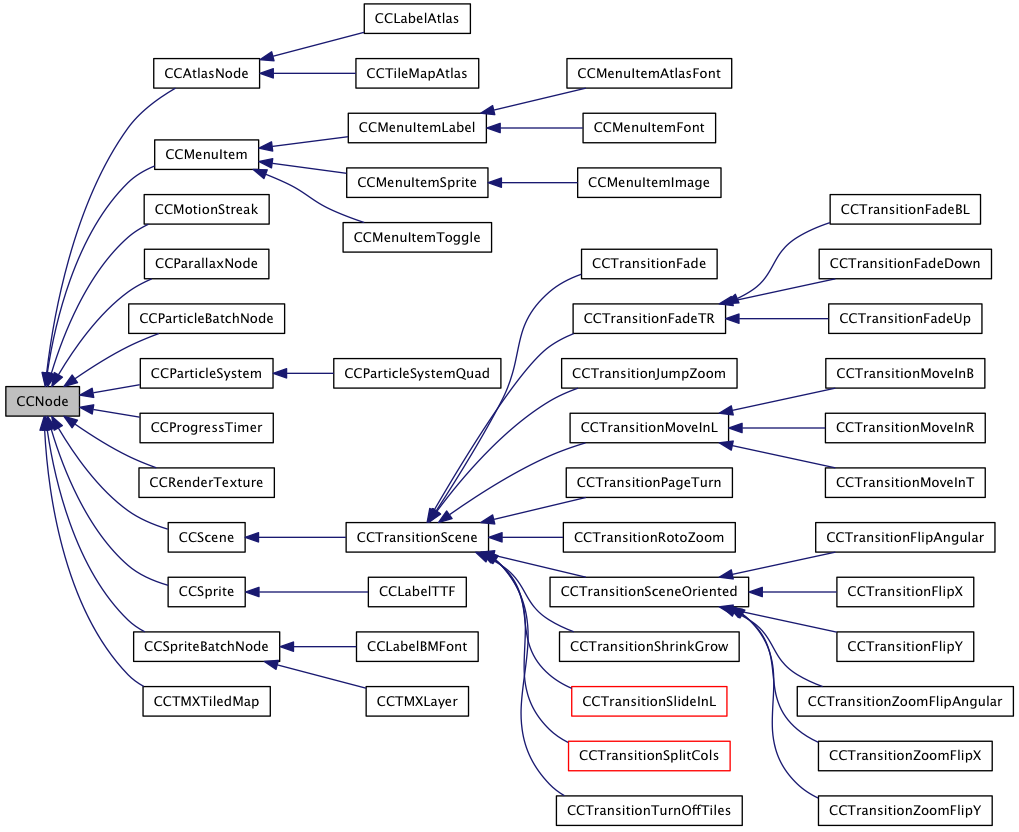
Tọa độ nút trong Cocos2d-X có liên quan đến tọa độ của nút cha, các nút con sẽ kế thừa các thuộc tính từ nút cha như tỉ lệ, độ xoay riêng màu sắc và độ hiển thị hình ảnh (Opacity, nằm trong khoảng 0 -> 255, tương ứng với alpha 0 -> 1).



Hình . Vị trí của một nút con khi được thêm vào nút cha

**2.1.1.4.2 CCNode**

Sơ đồ các lớp được kế thừa từ CCNode. Và trong đó, các lớp thường được sử dụng nhất được tô màu xám:



Hình . Sơ đồ các lớp kế thừa từ CCNode

CCNode là lớp cơ sở của tất cả các nút. CCNode là một lớp trừu tượng không chứa bất kì một hàm cụ thể nào và định nghĩa tất cả các thuộc tính và phương thức chung nhất cho tất cả các nút.

CCNode chứa tất cả các phương thức để quản lý (thêm, xóa, lấy đối tượng) cho các nút con.

**Khởi tạo một nút**:

CCNode \*childNode = CCNode::create();

**Thêm một nút**:

myNode->addChild(childNode, tag); // tag (kiểu int) dùng định danh cho nút đó gọi là “tag”.

**Lấy nút**:

myNode->getChildByTag(tag); // lấy childNode đã được định danh là tag (kiểu int)

**Xóa 1 nút bằng tag, dọn sạch toàn bộ các hành động đang chạy**:

myNode->removeChildByTag(tag, true); // true là dọn sạch

**Xóa tất cả các** **nút**:

myNode->removeAllChild();

**2.1.1.4.3 CCAction**

Một node có thể chạy các hành động. Các hành động (Action) là một lớp được sử dụng để thực hiện các thao tác di chuyển, thay đổi kích thước, xoay, làm mờ đi và nhiều hiệu ứng khác nhau trên một nút. Bởi vì lớp hành động này làm việc trên mọi nút, nên chúng ta có thể sử dụng chúng lên các hình (sprite), hay chữ (label) và thậm chí là các khung chọn (menu) hoặc toàn bộ một cảnh (scene).

**Khởi tạo một hành động**:

CCAction \*action = CCMoveBy::create(0.1f, 10); // CCMoveBy là một hành động, dùng để di chuyển

**Chạy các hành động**:

myNode->runAction(action, tag); // tag (kiểu int) để định danh cho hành động

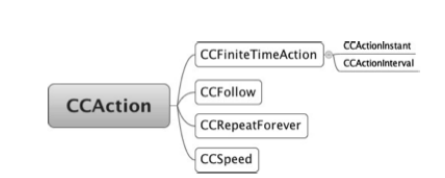
**Ngừng một hành động bằng định danh tag**:

myNode->stopActionByTag(tag);

**Ngừng toàn bộ các hành động của một nút**:

myNode->stopAllAction();

Dưới đây là sơ đồ các lớp kế thừa từ lớp CCAction.



Hình . Sơ đồ lớp kế thừa từ CCAction

Có 3 lớp Hành động kế thừa trực tiếp từ CCAction:

* CCFollow: cho phép một nút theo sau một nút khác
* CCRepeatForever: lặp lại một nút vô tận
* CCSpeed: thay đổi tần suất cập nhật của một hành động khi nó đang chạy

**CCFollow** có thể thiết lập một nút có hành động giống một nút khác. Ví dụ, đối tượng tên nhân vật (chữ - label) chạy đính vào một người chơi (player)

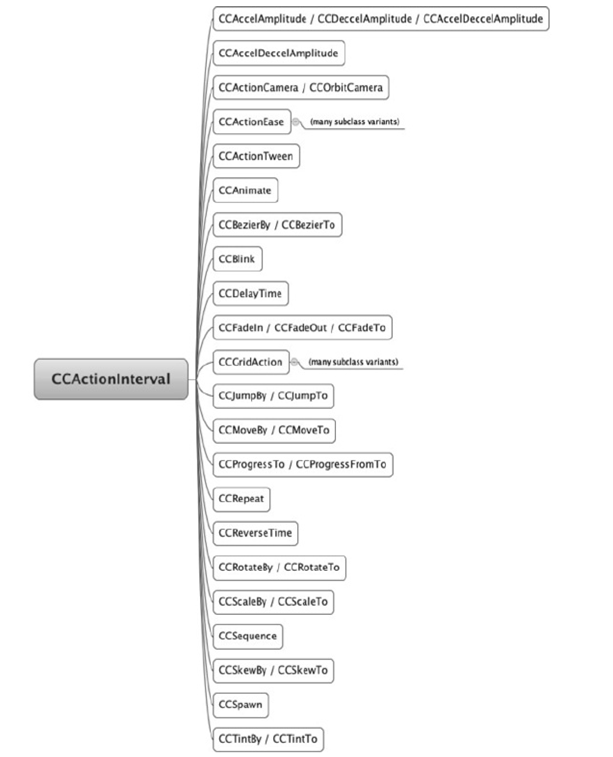
Label->runAction(CCFollow::create(player));

CCRepeatForever cho phép tạo ra chuỗi các hành động nối nhau:

CCAction \*action = CCRepeatForever::create( CCMoveBy::create(0.1f, point1), CCMoveTo::create(0.1f, point2)); //point1, point2 là một điểu kiểu CCPoint

**CCSpeed** được dùng để thay đổi tốc độ của một hành động khác chạy, dùng tương tự như các hành động khác với lệnh create và runAction. Nhưng, CCSpeed không thể thêm vào trong CCSequence.

**Interval** được tạo ra để giúp thực hiện các hành động diễn ra trong một khoảng thời gian nhất định và hành động đó không cần xóa khi thực hiện xong các hành động sẽ tự xóa.



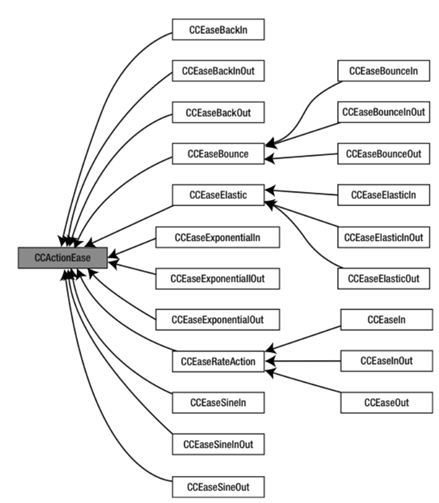
Hình . Sơ đồ lớp CCActionIntervel

**CCSequence** cho phép một hành động thực hiện sau khi một hành động khác được thực hiện xong. CCSequence cũng có thể sử dụng CCRepeatForever, CCSpeed. Lưu ý, Cocos2d-x cần thêm NULL vào sau các hành động trong CCSequence, do đó, nếu không để NULL ở cuối chuỗi hành động sẽ bị lỗi.

**Ease** cho phép thay đổi hiệu ứng của Hành động mọi lúc. Ease tạo ra các chuyển động mà thông thường tốn rất nhiều thời gian để viết một đoạn mã thực thi. Thuộc tính rate xác định mức độ hiệu ứng và phải lớn hơn 1 mới thấy được hiệu ứng.

CCMoveTo \*move = CCMoveTo::create(11, ccp(11, 1992));

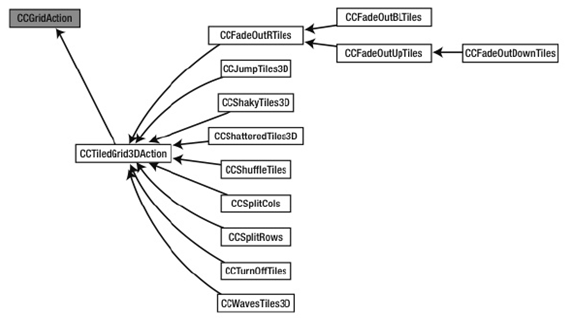
myCode->runAction(CCEaseInOut::create(move,4));



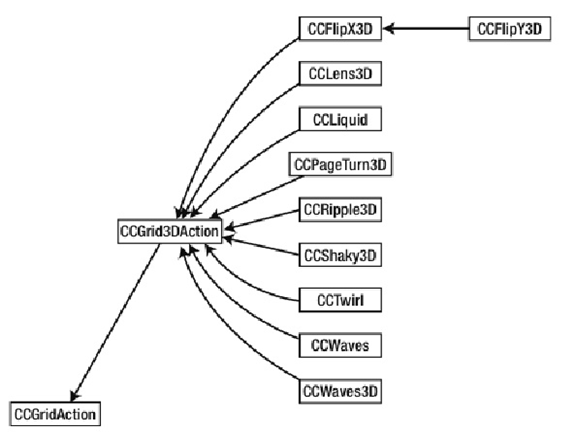
Hình . Sơ đố các lớp kế thừa từ CCActionEase

**Grid** có 2 lớp con là CCDrid3Daction và CCTiledGrid3Daction. Dưới đây là sơ đồ của CCGridAction:

Grid tạo nên các hiệu ứng 3D nhưng hiệu ứng sẽ không thực hiện được nếu không được bật depth buffering, điều này sẽ làm tốn bộ nhớ rất nhiều và chi phí xử lý.



Hình . Sơ đồ các lớp kế thừa từ CCTilesGrid3Daction



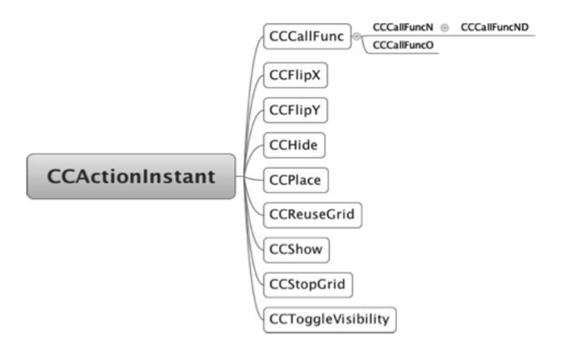
Hình . Sơ đồ các lớp kế thừa từ CCGrid3DAction

Để thiết lập depth buffering, một số đoạn mã trong lớp ủy nhiệm (Delegate) lúc khởi tạo EAGLView, mặc định thuộc tính DepthFormat là 0, để thiết lập depth buffering chúng truyền vào các đối số GL\_DEPTH\_COMPONENT\_OES cho 16-bit depth buffer hoặc GL\_DEPTH\_COMPONENT24\_OES cho 24-bit depth buffer.



Hình . Hiệu ứng lật trang

**Instant** được tạo ra để hỗ trợ cho các chuỗi hành động. Đôi khi trong chuỗi các hành động phải thay đổi một thuộc tính của nút, sau đó tiếp tục các hành động khác.



Hình . Sơ đồ các lớp kế thừa từ CCActionInstant

Điểm khác biệt giữa các lớp trên là loại chọn lọc (selector) nào sẽ được gọi và ngữ cảnh nào mà phương thức được gọi.

**2.1.1.4.4 Scheduled**

Lịch trình (scheduled) được cocos2d-x dùng để gọi thực thi các hàm. Dùng hủy lịch trình (Unschedule) để ngừng thực thi các hàm tương tự như lên lịch trình.

**ScheduleUpdate** dùng để gọi hàm Update. Update dùng để cập nhật trạng thái của nút mỗi khung (frame) tối đa nên là 60 lần 1 giây. Ví dụ:

myNode->scheduleUpdate();

Trong mã phải khai báo Cập nhật (Update) như sao:

void update(float dt) { ... }

**Schedule** dùng để gọi một hàm chọn lọc (selector) bất kỳ được khai báo giống với hàm lịch trình (Schedule Update) chỉ có điểm khác là có thời gian chờ trong các lần gọi lịch trình. Nếu truyền số vào là 0 thì nó giống như hàm lịch trình (ScheduleUpdate).

myNode->Schedule(selector, 3); // 3 là thời gian giữa 2 lần gọi schedule

void selector(float dt) { ... }

**ScheduleOnce** dùng để gọi hàm chỉ một lần không lặp dùng tương tự như hàm lịch trình (schedule) chỉ có điểm khác là số được truyền vào là thời gian tính từ lúc gọi đến lúc thực thi chọn lọc (selector).

**ScheduleUpdateWithPriority** Nếu nhiều nút cùng gọi hàm lịch trình (schedule update) thì hàm nào có độ ưu tiên càng nhỏ sẽ được thực thi trước (mặc định là 0).

**2.1.1.4.5 CCDirector**

CCDirector là một phần quan trọng của cocos2d-x. Lớp CCDirector là một lớp đơn (singleton), nghĩa là chỉ có duy nhất một thể hiện tại mọi thời điểm và CCDirector có thể được truy xuất toàn cục bằng cách gọi sharedDirector hay getInstance.

CCDirector chứa các thiết lập toàn cục cho cocos2d-x và quản lý các cảnh (scene) trong cocos2d-x. Một số chức năng của lớp CCDirector:

* Truy xuất vào cảnh hiện tại của màn hình.
* Chạy, thay thế, thêm, bớt một cảnh.
* Truy xuất vào các thiết lập của cocos2d-x.
* Truy xuất vào OpenGL view và hệ điều hành window trong cocos2d-x.
* Hỗ trợ ngừng, chạy lại và kết thúc trò chơi.
* Xác định cách mà trạng thái game được Cập nhật (Update).

Để chạy trò chơi, đầu tiên phải khởi tạo một cảnh (scene) nằm ở cuối hàm applicationDidFinishLaunching trong AppDelegate. Dùng CCDirector để bắt đầu chạy với hàm runWithScene

CCDirector::getInstance()->runWithScene(scene); // phiên bản 3.3

Thay thế bằng 1 cảnh khác (otherScene):

CCDirector::getInstance()->replaceScene(otherScene);

Dùng hiệu ứng để chuyển cảnh (changeScene):

CCSceneTransition tran = {loại chuyển màn hình}::create(2, changeScene); // 2 là số giây trước lúc chuyển

CCDirector::getInstance()->replaceScene(otherScene);

Lưu ý, khi thay thế một cảnh mới cảnh cũ sẽ bị xóa hoàn toàn khỏi RAM và các nút, lịch trình cũng bị hủy.

**Pushing, Poping** cho phép thay thế cảnh mới vào mà không phải giải phóng cảnh cũ, Giúp chuyển qua các cảnh mới nhanh hơn. Tuy nhiên, các cảnh sẽ chiếm một lượng RAM khác lớn có thể dẫn đến lỗi khi chơi.

Một số vấn đề khác, khi đẩy một cảnh (push) vào phải nhớ số lượng cảnh để có thể lấy ra (pop) một cách chính xác. Nếu không khi lấy cảnh sẽ xảy ra lỗi và gãy trò chơi.

#### Những điểm mạnh của cocos2dx:

* Cross-platform: ngày nay hầu hết các framework đều cross-platform. Nhưng vấn đề là Cocos2d-x làm được nhiều hơn hẳn, cụ thể, Cocos2d-x hỗ trợ cho: windows, mac, linux, ios, android, blackberry, tizen, winRT, window phone
* Open source: hoàn toàn open source, có thể làm bất kỳ điều gì với code của nó đồng thời cũng có thể đóng góp vào framework nữa để giúp nó phát triển hơn.
* Light weight & Fast running: sử dụng C++, Cocos2d-x có hiệu suất cao, chạy được trên cả những thiết bị cấu hình rất thấp của Android.
* APIs đơn giản và dễ học
* Lua binding, Js binding: hoàn toàn có thể sử dụng Lua, hay javascript để hiện thực sản phẩm, (không nhất thiết là C++). Điều này rất hữu ích đối với những người mới, không mạnh về C++
* Tools và Library: hiện tại có rất nhiều công cụ và thư viện hỗ trợ phát triển game bằng Cocos2dx, có thể xem tại trang hub:

<http://www.cocos2d-x.org/hub>

Nổi bật trong số đó là cocos studio

#### So sánh với các 2d game engine khác

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Cocos2dx | GameMaker | MonoGame | AndEngine | GameSalad | Love |
| Giá thành | Hoàn toàn miễn phí | Có bản miễn phí + trả phí | Chỉ miễn phí trên window | Hoàn toàn miễn phí | Trả phí/ miễn phí | Hoàn toàn miễn phí |
| Open source | Có | Không | Có | Có | Không | Có |
| Ngôn ngữ sử dụng | C++, lua, javascript | Không viết code | C# | Java | Không viết code | Lua |
| Performance | Tốt | Tương đối | Tệ | Tương đối | Tương đối | Tương đối |
| Nền tảng hỗ trợ | Mac OS X,  IOS  Android,  Window phone,  Windows,  Linux,  Blackberry  Tizen,  WinRT | HTML5, Linux, Windows, Mac, Windows Phone, Android, iOS, PS3/4/Vita | Windows, Mac, Linux, iOS,  Android | Android | Windows 8, Android, iOS, HTML5, Mac OS X | Windows, Mac OS X, Linux |
| Mức độ dễ sử dụng | Dễ sử dụng | Không đòi hỏi kĩ năng lập trình, rất dễ sử dụng | Dễ | Dễ | Không đòi hỏi kĩ năng lập trình, rất dễ sử dụng | Tương đối |
| Cộng đồng | Lớn, thân thiện | Lớn | Ít | Tương đối | Lớn | Tương đối |
| Documentation | Đa phần đã lỗi thời | Rất đầy đủ | Ít | Không có tài liệu cụ thể | Tốt | Rất đầy đủ |
| Tutorial | Khá nhiều | Tốt | Tương đối | Ít | Tốt | Tốt |

Bảng . So sánh Cocos với các engine khác

#### Thư viện vật lý Box2D

##### Nguồn gốc và lịch sử phát triển

* Phiên bản đầu tiên của Box2D có tên gọi là Box2D Lite, một engine để thực hiện việc mô phỏng các chuyển động vật lý, được phát triển bởi ErinCatto tại GDC vào năm 2006.
* Ngày 11 tháng 9 năm 2007, Box2D được phát hành dưới dạng mã nguồn mở trên Sourceforge và ngày 17 tháng 1 năm 2010 thì chuyển hosting sang Google Code.
* Ngày 6 tháng 3 năm 2008, phiên bản 2.0 được phát hành.

Box2D được viết bằng C++, vì thế có thể chạy được trên bất cứ hệ thống nào có sẵn trình biên dịch C++ (đa nền tảng). Box2D đã được sử dụng trên các máy chơi game console Nintendo DS, Wii, các thiết bị di động gồm Android, BlackBerry 10, iPhone cũng như hầu hết các hệ điều hành phổ biến.

Box2D cũng đã có phiên bản trên các ngôn ngữ khác như Java, Adobe Flash, C#, Lua, Javascript.

##### Các tính năng chính

* Collision
* Continuous collision detection
* Contact callbacks: begin, end, pre-solve, post-solve
* Convex polygons và circles
* Multiple shapes per body
* One-shot contact manifolds
* Dynamic tree broadphase
* Efficient pair management
* Fast broadphase AABB queries
* Collision groups và categories
* Continuous physics
* Persistent body-joint-contact graph
* Island solution và sleep management
* Contact, friction và restitution
* Revolute, prismatic, distance, pulley, gear, mouse joint và các loại joint khác
* Joint limits, motors, và friction
* Momentum decoupled position correction
* Fairly accurate reaction forces/impulses
* Systems
* Small block và stack allocators
* Centralized tuning parameters
* Highly portable C++ không sử dụng STL containers

### Cocos studio

Đây là công cụ nổi tiếng nhất giúp tăng tốc quá trình phát triển game với cocos2d, được phát triển bởi Chukong (nhà phát triển đồng thời cũng là chủ sở hữu cocos2dx).

Cocos studio bao gồm 3 thành phần chính là :

UI editor: dùng để tạo UI cho game, giao diện đơn giản và trực quan, dễ sử dụng, cho phép cái graphic artist sử dụng 1 cách dễ dàng mà không cần kiến thức lập trình

Animation editor: hỗ trợ animation kiểu frame by frame và skeletal, điểm nổi bật của animation editor là dễ sử dụng, trực quan sinh động, WYSIWYG, texture packing (đóng gói các file texture riêng lẻ thành 1 file để tiết kiệm bộ nhớ cũng như cải thiện performance.

Scene editor: là nơi gắn những thứ tạo ra từ UI và animation editor vào lại, cũng thể giả lập thành 1 scene trong game sử dụng bộ giả lập được tích hợp sẵn của cocos studio

### Quadtree

#### Định nghĩa

Quadtree là một cấu trúc dữ liệu phân nhánh dạng cây, được sử dụng để phân hoạch vùng không gian hai chiều hiện tại thành các vùng nhỏ hơn và dễ quản lý hơn. Khác với cây nhị phân, mỗi lần phân hoạch, quadtree sẽ chia vùng hiện tại thành 4 vùng. Do đó mỗi node trong quadtree có thể có 4 node dẫn xuất hoặc không có node dẫn xuất nào, không có ngoại lệ.

Việc sử dụng Quadtree giúp chúng ta quản lý các đối tượng trong game một cách hiệu quả hơn, bằng cách phân chia vùng cần xử lý các đối tượng thành những vùng con, các đối tượng sẽ được đưa vào các vùng tương ứng và được quản lý riêng biệt. Quá trình phân chia cứ thế tiếp diễn cho đến khi mỗi vùng chỉ chứa một số lượng đối tượng nhất định hoặc mức độ phân chia đạt mức “chấp nhận được”.

#### ứng dụng quadtree trong game

Quadtree thường được ứng dụng vào việc xét va chạm giữa các đối tượng, update và render (vẽ) các đối tượng.

Chi phí cho việc khởi tạo quadtree nhỏ hơn rất nhiều so với việc update, render, xử lý va chạm, … các đối tượng không cần thiết. Ví dụ một đối tượng nằm ngoài vùng nhìn thấy của camera trong game thì không nhất thiết phải render nó.

Khi sử dụng Quadtree, có thể dễ dàng nhận biết các đối tượng thuộc 2 node khác nhau không có khả năng va chạm với nhau (trừ một số trường hợp đặc biệt). Chi phí giảm đồng nghĩa với việc thời gian của một vòng lặp game được rút ngắn. Do đó FPS của game sẽ được cải thiện.

## Server side

### HTTP và socket

#### HTTP là gì

HTTP là chữ viết tắt của HyperText Transfer Protocol (giao thức truyền tải siêu văn bản). Đây là một giao thức ứng dụng trong bộ các giao thức TCP/IP (gồm một nhóm các giao thức nền tảng cho internet).

HTTP hoạt động dựa trên mô hình Client – Server. Trong mô hình này, các máy tính của người dùng sẽ đóng vai trò làm máy khách (Client). Sau một thao tác nào đó của người dùng, các máy khách sẽ gửi yêu cầu đến máy chủ (Server) và chờ đợi câu trả lời từ những máy chủ này. Để có thể nói chuyện được với nhau, các máy chủ và máy khách phải thực hiện việc trao đổi thông qua các giao thức. Một trong những giao thức được sử dụng thường xuyên nhất chính là HTTP.

#### Socket là gì

Socket set là một khái niệm trừ tượng, nó cho phép ứng dụng trao đổi dữ liệu với một ứng dụng khác cùng kết nối với một mạng. Thông tin được ghi vào socket bởi một ứng dụng tại một máy có thể được đọc bởi một ứng dụng nằm trên một máy khác. Khá giống HTTP

#### so sánh HTTP và socket

So với HTTP, socket có những điểm khác biệt sau:

* Persistent: connection state được giữ lại, không phải như http mỗi lần gửi đều phải tạo lại 1 connection mới
* Low latency: thời gian truyền gần như real-time bởi vì không cần phải tốn chi phí khởi tạo lại connnection mỗi khi cần gửi request như http
* Full duplex: cả server lẫn client đều có thể gửi dữ liệu qua lại với nhau, trong khi http thì request chỉ có thể gửi bởi client và response trả về từ server.

#### Tại sao sử dụng HTTP

Dễ thấy từ phần so sánh trên, ta có thể nhận ra rằng socket và HTTP được thiết kế để giải quyết những bài toán khác nhau

* Socket được thiết kế để giao tiếp đa chiều, latency cực kì quan trọng, và connection giữ lại trạng thái
* HTTP được thiết kế để theo mô hình client-server truyền thống, latency không thực sự quá cần thiết, connection phi trạng thái

Ngoài ra, cầu hình HTTP thì cũng dễ hơn so với socket

Từ đó, nhóm nhận thấy sử dụng HTTP phù hợp hơn.

### Jetty

#### Giới thiệu

Jetty là 1 open source serverlet container, nó phục vụ các nội dung web trên nền tảng java như serverlet, JSP. Jetty được viết bằng Java và API của nó được cung cấp dưới dạng 1 tập hợp các file Jar. Người dùng có thể khởi tạo jetty như là 1 object và nhanh chóng thêm khả năng kết nối mạng vào 1 ứng dụng java.

Jetty bắt đầu nổi từ năm 1998, khi được release dưới giấy phép Apache 2.0

Website: http://www.eclipse.org/jetty/

#### Các tính năng

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Version | Java Version | Protocols | Servlet Version | JSP Version |
| 9.3.x | 1.8 | HTTP/1.1, [HTTP/2](https://en.wikipedia.org/wiki/HTTP/2), [WebSocket](https://en.wikipedia.org/wiki/WebSocket) JSR356, [SPDY](https://en.wikipedia.org/wiki/SPDY) | 3.1 | 2.3 |
| 9.2.x | 1.7 | HTTP/1.1, [WebSocket](https://en.wikipedia.org/wiki/WebSocket) JSR356, [SPDY](https://en.wikipedia.org/wiki/SPDY) | 3.1 | 2.3 |
| 9.1.x | 1.7 | HTTP/1.1, [WebSocket](https://en.wikipedia.org/wiki/WebSocket) JSR356, [SPDY](https://en.wikipedia.org/wiki/SPDY) | 3.1 | 2.3 |
| 9.0.x | 1.7 | HTTP/1.1, [WebSocket](https://en.wikipedia.org/wiki/WebSocket), [SPDY](https://en.wikipedia.org/wiki/SPDY) | 3.0 (tracking 3.1 drafts) | 2.2 |
| 8.x | 1.6 | HTTP/1.1, [WebSocket](https://en.wikipedia.org/wiki/WebSocket), [SPDY](https://en.wikipedia.org/wiki/SPDY) | 3.0 | 2.1 |
| 7.x | 1.5, J2ME | HTTP/1.1, [WebSocket](https://en.wikipedia.org/wiki/WebSocket), [SPDY](https://en.wikipedia.org/wiki/SPDY) | 2.5 | 2.1 |
| 6.x | 1.4–1.5 | HTTP/1.1 | 2.5 | 2.0 |
| 5.x | 1.2–1.5 | HTTP/1.1 | 2.4 | 2.0 |
| 4.x | 1.2, J2ME | HTTP/1.1 | 2.3 | 1.2 |
| 3.x | 1.2 | HTTP/1.1 RFC2068 | 2.2 | 1.1 |
| 2.x | 1.1 | HTTP/1.0 RFC1945 | 2.1 | 1.0 |
| 1.x | 1.0 | HTTP/1.0 RFC1945 |  |  |

Bảng . Các tính năng của Jetty

#### So sánh với các server application khác

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Jetty | JBoss | Tomcat | Liberty profile | Glassfish |
| Dễ cài đặt | 10 | 10 | 8 | 8 | 7 |
| Dễ cấu hình | 8 | 8 | 8 | 10 | 6 |
| Tài liệu | 7 | 10 | 8 | 8 | 7 |
| Giá thành | Miễn phí | Giới hạn | Miễn phí | Không | Giới hạn |
| Hiệu suất | 9 | 10 | 9 | 8 | 7 |
| Mở rộng | 10 | 8 | 9 | 8 | 9 |

Bảng . So sánh các application server

### Redis

#### Tại sao chọn NoSQL?

Cùng với sự phát triển của các ứng dụng Web hiện đại, đã kéo theo sự phát triển vượt bậc của dữ liệu. Đồng thời, cũng phát sinh nhiều hạn chế của RDBMS. Thế nên, công nghệ NoSQL ra đời và không ngừng lớn mạnh. NoSQL đang được xử dụng tại các công ty internet hàng đầu như: Google, Facebook, Amazon và LinkedIn [9] mà. Chính nhờ có NoSQL mà bốn vấn đề của thời đại bao gồm: Big Users, Big Data, the Internet of Things và Cloud Computing được giải quyết một cách đơn giản nhất [13].

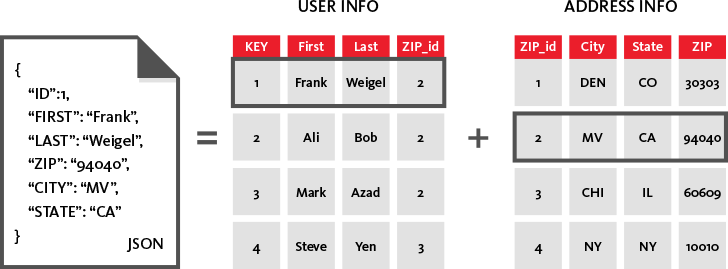
|  |  |
| --- | --- |
| NoSQL | RDBMS |
| Khả năng mở rộng hệ thống nhanh chóng lên hàng triệu người dùng | Khả năng mở rộng hệ thống lên hàng ngàn người dùng |
| Dữ liệu không cần cấu trúc, gồm một cập khóa – giá trị, giá trị có thể là JSON hay Binary | Dữ liệu phải có cấu trúc định trước và lưu trữ trong các cột của các bảng |
| Khả năng mở rộng hệ thống dễ dàng, chi phí thấp | Khả năng mở rộng rất khó, nhiều khi là không thể. Tốn nhiều chi phí |
| Không cần kiến thức về ngôn ngữ truy vấn SQL | Yêu cầu có kiến thức về ngôn ngữ truy vấn SQL |
| Thời gian truy xuất cực ngắn, được thiết kết cho các ứng dụng có khả năng chịu tải cao, nhưng không đảm bảo chính xác trạng thái dữ liệu [9]. | Thời gian truy xuất dữ liệu cao, được thiết kế cho ứng dụng có tầng suất truy xuất thấp, đảm bảo chính xác trạng thái dữ liệu. |
| Không hổ trợ phân tích, báo cáo sắp xếp dữ liệu [9] | Hổ trợ phân tích, báo cáo sắp xếp dữ liệu |
| Phù hợp với dữ liệu đơn giản [5][6] | Phù hợp với dữ liệu có ràng buộc phức tạp. |

Bảng . So sánh giữa NoSQL và RDBMS [15]

Chính vì thế, tùy vào yêu cầu và tính chất của hệ thống mà lựa chọn giữa NoSQL và RDBMS hoặc kết hợp cả hai. Còn với đề tài này, yêu cầu về cấu trúc dữ liệu khá đơn giản, không cần yêu cầu chính xác tuyệt đối, nhưng phải đảm bảo chịu tải cao, mở rộng hay thu hẹp dễ dàng và đảm bảo thường xuyên thay đổi cấu trúc dữ liệu nên NoSQL là một sự lựa chọn phù hợp.

##### Mô hình dữ liệu rất linh hoạt

Với NoSQL dữ liệu dữ liệu không cần có cấu trúc, mà chỉ đơn giản bao gồm một khóa và một giá trị, dùng khóa để truy cập vào giá trị. Khóa ở đây là một chuỗi ký tự, còn giá trị có thể là một binary, một chuỗi ký tự hay một đối tượng dạng JSON. Cập khóa – giá trị trong Couchbase thường được gọi là document. Giữa các dữ liệu không hề có bắt cứ quan hệ nào. Trong NoSQL cũng không hề có khái niệm Schema. Việc thay đổi cấu trúc dữ liệu cũng được diễn ra một cách hoàn toàn tự do.

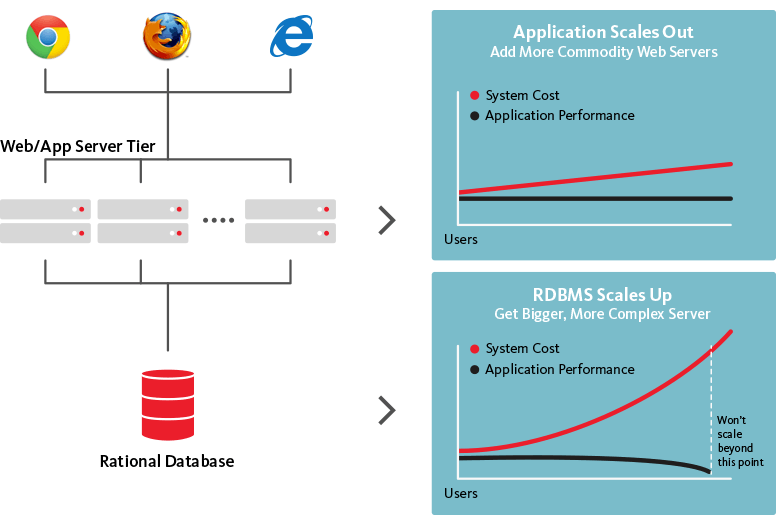


Hình . So sánh cách lưu trữ dữ liệu giữa NoSQL và RDBMS

##### Khả năng mở động

Để đáp ứng cho nhu cầu về Big Users và Big Data, bắt buộc phải mở rộng hệ thống. Việc mở rộng hệ thống có 2 lựa chọn: nâng cấp cấu hình máy chủ (scale up) và thêm mày chủ (scale out).

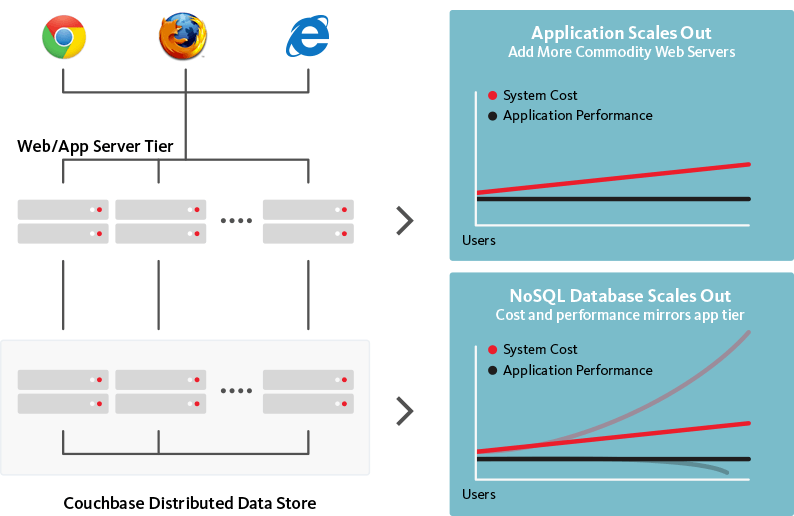
Đối với mô hình RDBMS: việc nâng cấp máy chủ gặp phải hạn chế lớn: đòi hỏi máy chủ có chất lượng rất cao, dẫn đến chi phí cao. Ngoài ra việc nâng cấp cấu hình máy chủ còn phải đối mặt với một thách thức giới là giới hạn tối đa của phần cứng. Còn đối với lựa chọn thêm máy chủ, đối với những hệ thống sử dụng kiến trúc ba lớp, cách tiếp cận thêm máy chủ đã được sử dụng hiệu quả trong thời gian qua. Tuy nhiên, ở phương pháp này việc thêm máy chủ để chia tải chỉ thực hiện được ở tầng ứng dụng. Việc thêm máy chủ ở tầng dữ liệu là gần như không thể. Nguyên nhân là do các yêu cầu khắc khe về mặt quan hệ giữa các dữ liệu.



Hình . Chi phí để mở rộng hệ thống sử dụng

RDBMS rất lớn và không hiệu quả.

Đối với mô hình NoSQL, việc thêm máy chủ vào tầng dữ liệu vô cùng dễ dàng. Khi một máy chủ mới được thêm vào, nó sẽ tham gia vào cluster với vai trò là một node. Lúc đó dữ liệu và các thao tác trên dữ liệu được bảo đảm thông suốt trong cả hệ thống rộng lớn. NoSQL đảm bảo dữ liệu được lan truyền trên khắp các node và chia tải trên các node. Nếu một thao tác thất bại trên một node sẽ được tự động chuyển sang một node khác. Một ưu điểm khác của NoSQL là chi phí máy chủ rất rẽ, do NoSQL opensource và không đòi hỏi máy chủ có cấu hình cao. Đồng thời số lượng node và user hoàn toàn tuyến tính với nhau. Cụ thể là để phục vụ cho 10.000 người dùng ban đầu cần một máy chủ. Lúc đó, khi tăng lên trong khoảng 10.000 người dùng tiếp theo, ta chỉ việc thêm mới một máy chủ vào cluster.



Hình . Hệ thống cơ sở dữ liệu NoSQL dễ dàng

mở rộng với chi phí tương đối thấp và tuyến tính

##### Hiệu năng cao

NoSQL luôn luôn đạt được hiệu năng cao hơn rất nhiều lần so với RDBMS do những nguyên nhân sau:

* Tự động chia sẻ: NoSQL đảm bảo dữ liệu được chia sẽ và sao lưu bản sao ở tất cả các node. Nên việc thêm mới hay xóa một node có thể diễn ra mà không cần phải ngừng hệ thống. Ngoài ra, còn đảm bảo tự động chia tải giữa các node.
* Khả năng truy vấn trực tiếp: dữ liệu có thể truy cập trực tiếp mà không cần phải thực hiện thao tác kết hợp các bảng và tìm kiếm như RDBMS. Ngoài ra, NoSQL còn đảm bảo tốc độ truy vấn dữ liệu tối đa cho dù cluster có đến hàng trăm node.
* Tích hợp bộ nhớ đệm: Để giảm độ trễ và tính bền vững dữ liệu. Dữ liệu sẽ được lưu trữ vào bộ nhớ Ram, sau khoảng thời gian mới cập nhật xuống đĩa cứng.

#### Giới thiệu redis

Redis là 1 open source, in-memory data structure store, được viết bằng ngôn ngữ ansi C, có thể sử dụng như 1 database, cache hoặc message broker. Redis hỗ trợ nhiều cấu trúc dữ liệu như: strings, hashes, lists, sets, sorted sets.

Ra mắt lần đầu: 10/4/2009

Giấy phép sử dụng: BSD.

Loại: nosql key-value store

Phiên bản ổn định: 3.0.6 (ra mắt ngày 18/12/2015)

#### Đặc tính redis

* Điểm mạnh nhất: cực kì nhanh.
* Protocol: telnet.
* In-memory: có hỗ trợ back-up trên disk.
* Hỗ trợ cấu trúc dữ liệu: string, list, set, hash, sorted set.
* Cấu trúc dữ liệu đơn giản nhưng có thể thực hiện các lệnh phức tạp như .ZREVRANGEBYSCORE (trả về 1 danh sách các member trong 1 sorted set, sắp xếp bằng score từ cao xuống thấp..
* Có thể script bằng lua.
* Có transaction.
* Giá trị của key có thể đặt expire (tự động xóa sau 1 khoảng thời gian).
* Đa số các lệnh đều có time complexity là O(1).

#### So sánh với các no-sql database khác

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Đặc điểm | Lưu trữ | Cấu trúc | Sử dụng khi | Mô tả |
| Redis | Cực nhanh | RAM | Key-value | Khi cần xử lý dữ liệu với tốc độ nhanh, nhưng khối lượng không thực sự lớn (do lưu trên ram) | Dữ liệu lưu trữ trên ram (có persistent), hỗ trợ các cấu trúc dữ liệu đơn giản và quen thuộc. hỗ trợ clustering + automatic failover |
| MongoDB | Khá giống sql | Disk | BSON | Truy vấn lượng lớn dữ liệu trên các field cố định | Có các tính năng quen thuộc từ sql (query, index) |
| Casandra | Dữ liệu cực lớn sql | Disk | CQL3 | Khi cần và phân tích lượng dữ liệu cực lớn | Lưu trữ và truy xuất lượng lớn với các thao tác quen thuộc của sql, tính scalability cao |
| CouchDB | Rất nhanh | Memcache | Key-value | Khi cần truy xuất lượng dữ liệu lớn, với độ trễ thấp và tốc độ cao | Dạng key-value, memcache, có hỗ trợ persistent và clustering |
| Neo4j | Graph database | Disk | Graph | Phù hợp cho việc xây dựng mạng xã hội, quản lý giao thông, bản đồ | Dữ liệu dạng đồ thị, được định nghĩa dựa trên các khái niệm quen thuộc trong lý thuyết đồ thị |

Bảng . So sánh các nosql database

### JSON

#### JSON là gì

JSON là chữ viết tắt của Javascript Object Notation, đây là một dạng dữ liệu tuân theo một quy luật nhất định mà hầu hết các ngôn ngữ lập trình hiện nay đều có thể đọc được, bạn có thể sử dụng lưu nó vào một file, một record trong CSDL rất dễ dàng. JSON có định dạng đơn giản, dễ dàng sử dụng và truy vấn hơn XML rất nhiều nên tính ứng dụng của nó hiện nay rất là phổ biến.

1 chuỗi json ví dụ:

{

"username" : "phong",

"email" : "phong3dt@gmail.com",

"website" : "daa.uit.edu.vn",

"title" : "Khóa luận tốt nghiệp"

}

Như vậy cú pháp của JSON rất đơn giản là mỗi thông tin dữ liệu sẽ có 2 phần đó là key và value, điều này tương ứng trong CSDL là tên field và giá trị của nó ở một record nào đó. Tuy nhiên nhìn qua thì đơn giản nhưng nếu ta mổ xẻ nó ra thì có một vài điều như sau:

* Chuỗi JSON được bao lại bởi dấu ngoặc nhọn {}
* Các key, value của JSON bắt buộc phải đặt trong dấu nháy kép {"}, nếu ta đặt nó trong dấu nháy đơn thì đây không phải là một chuỗi JSON đúng chuẩn. Nên trường hợp trong value của ta có chứa dấu nháy kép thì hãy dùng dấu (\) để đặt trước nó,
* Nếu có nhiều dữ liệu (nhiều cặp key => value) thì ta dùng dấu phẩy (,) để ngăn cách
* Các key của JSON nên đặt chữ cái không dấu hoặc số, dấu \_ và không có khoảng trắng., ký tự đầu tiên không nên đặt là số. Điều này rất giống với nguyên tắc đặt tên biến trong PHP.

#### Cấu trúc dữ liệu hỗ trợ

JSON định nghĩa các cấu trúc dữ liệu chính sau đây: số, chuỗi ký tự, logic (true và false), mảng (một dãy có trình tự các giá trị), đối tượng (tập hợp các cặp giá trị và khoá), và rỗng (null).

#### Tại sao sử dụng JSON

Trong việc làm game giữa client-server, cần có 1 cơ chế để giao tiếp giữa 2 bên, và JSON là 1 trong những lựa chọn hàng đầu vì những lý do sau:

* Đơn giản
* Dễ tìm hiểu
* Dễ sử dụng
* Dung lượng nhỏ (nhỏ hơn so với xml)
* Có nhiều bộ thư viện hỗ trợ
* Có tính human readable, con người nhìn vào có thể hiểu được.

Ngoài JSON ra, còn có 1 số giao thức khác như xml, google protocol buffer

## Các khái niệm liên quan.

### Tổng quan về Mã nguồn mở.

#### Giới thiệu mã nguồn mở.

Trong lĩnh vực sản xuất và phát triển, mã nguồn mở là một triết lý hoặc một phương pháp thúc đẩy tái phân phối sản phẩm tự do và truy cập tới những thiết kế và ý tưởng của sản phẩm hoặc những chi tiết triển khai sản phẩm.

Những đặc trưng của mã nguồn mở:

* Mã nguồn mở là miễn phí
* Mã nguồn mở có thể truy cập được mã nguồn
* Mã nguồn mở phát triển theo thời gian
* Mã nguồn mở thì không đóng
* Mã nguồn mở có thể sử dụng lại và thay đổi
* Mã nguồn mở có thể triển khai ở bất kỳ nơi đâu và dành cho bất kỳ ai

#### So sánh mã nguồn mở và mã nguồn đóng

Với mã nguồn mở chúng ta có thể:

* Xem hoặc sửa đổi mã nguồn của phần mềm hoặc ứng dụng
* Phần mềm mã nguồn mở được phát hành bởi cộng đồng phát triển mã nguồn mở và phải trải qua giai đoạn thứ hai của quá trình phát triển những phần mềm mã nguồn mở còn phần mềm mã nguồn đóng thì được phát triển cô lập trong một nhóm nhỏ các nhà phát triển.
* Hỗ trợ nhiều cho nhà phát triển và có cộng đồng phát triển lớn
* Mã nguồn mở an toàn hơn và các lỗi và các lỗ hổng thường được sửa lỗi

Mã nguồn mở không có nghĩa là truy cập tới mọi mã nguồn. Các điều khoản phân phối phần mềm mã nguồn mở phải tuân thủ các tiêu chuẩn sau đây:

* Tái phân phối tự do

Giấy phép mã nguồn mở không được phép hạn chế bên nào, bên bán hoặc bên mua các phần mềm, nó như một thành phần của một phân phối thành phần tổng hợp có chứa các chương trình từ các nguồn khác nhau. Giấy phép không được phép yêu cầu trả tiền bản quyền hoặc lệ phí khác để bán như vậy.

* Mã nguồn
* Các công việc được kế thừa

Giấy phép phải cho phép sửa đổi và kế thừa và phải cho phép chúng phân phối theo các điều khoản tương tự như giấy phép của phần mềm gốc.

* Tính toàn vẹn mã nguồn của tác giả

Giấy phép có thể hạn chế mã nguồn

* Không kỳ thị với người hoặc nhóm
* Không kì thị đối với lĩnh vực của Endeavor
* Phân phối giấy phép
* Giấy phép không phải riêng cho một sản phẩm
* Giấy phép không được hạn chế các phần mềm khác
* Giấy phép phải trung lập với công nghệ

#### Phần mềm mã nguồn mở

Phần mềm nguồn mở là phần mềm với mã nguồn được công bố và sử dụng một giấy phép nguồn mở. Giấy phép này cho phép bất cứ ai cũng có thể nghiên cứu, thay đổi và cải tiến phần mềm và phân phối phần mềm ở dạng chưa thay đổi hoặc đã thay đổi.

Không phải phần mềm mã nguồn mở nào cũng miễn phí. Và cũng không phải phần mềm miễn phí nào cũng là phần mềm mã nguồn mở.

Phần mềm mã nguồn mở hiểu với nghĩa rộng là một khái niệm chung được sử dụng cho tất cả các phần mềm mà mã nguồn của nó được công bố rộng rãi công khai và cho phép mọi người tiếp tục phát triển phần mềm đó. Điều này không có nghĩa là chúng có thể được sao chép, sửa chữa thoải mái hay sử dụng vào mục đích nào cũng được. Mã nguồn mở được công bố dưới rất nhiều điều kiện khác nhau (license), một số trong đó cho phép phát triển, sử dụng và bán tùy ý miễn là giữ nguyên các dòng về nguồn gốc sản phẩm (BSDL), một số bắt buộc tất các sản phẩm làm ra từ đó cũng phải là open-source (GPL), một số khác đòi hỏi phải công bố trọn vẹn mã nguồn (Mozilla), một số khác không cho phép sử dụng vào mục đích thương mại (Sun Solaris Source Code License), một số khác lại không có ràng buộc gì đáng kể (public domain, MIT X11 license) v.v. Qua đó ta thấy khái niệm open source không thể chuẩn xác mà muốn nói đến tính pháp lý của việc sử dụng các phần mềm mã nguồn mở, chúng ta phải xem xét đến điều kiện sử dụng (license) cụ thể mà dưới đó chúng được công bố. Một điều kiện hay được áp dụng nhất là GPL: GNU General Public License (<http://www.fsf.org/licenses/gpl.html>) của tổ chức Free Software Foundation.

GPL license có 2 đặc điểm phân biệt, đó là:

* Tác giả gốc giữ bản quyền về phần mềm nhưng cho phép người dùng rất nhiều quyền khác, trong đó có quyền tìm hiểu, phát triển, công bố cũng như quyền khai thác thương mại sản phẩm.
* Tác giả sử dụng luật bản quyền để bảo đảm các quyền đó không bao giờ bị vi phạm đối với tất cả mọi người, trên mọi phần mềm có sử dụng mã nguồn của mình.

Đặc biệt điểm thứ 2 thường được gọi là hiệu ứng virus (viral effect) vì nó biến tất cả các phần mềm có dùng mã nguồn GPL cũng biến thành phần mềm GPL. Trên thực tế điều này có ý nghĩa: bất kỳ tác giả nào sử dụng dù chỉ 1 phần rất nhỏ mã nguồn GPL trong chương trình của mình cũng phải công bố chương trình đó dưới điều kiện GPL.

Điều kiện này quy định ví dụ:

* Mọi phần mềm GPL đều phải công bỗ mã nguồn của mình rộng rãi công khai và phải tạo điều kiện cho mọi người truy cập được mã nguồn ấy (ví dụ qua web hoặc qua việc bán CD giá rẻ)
* Giữ nguyên mọi dòng chú thích về nguồn gốc tác giả, bản quyền của họ cũng như điều kiện được áp dụng đối với phần mềm (trong 1 file có tên LICENSE)
* Cấm việc bán mã nguồn nhưng cho phép kinh doanh chương trình được tạo ra từ mã nguồn ấy hoặc là các dịch vụ hỗ trợ liên quan.

#### Giấy phép mã nguồn mở

##### Định nghĩa

Là giấy phép bản quyền dành cho các phần mềm máy tính trong đó đặt ra các quy định buộc người sử dụng phần mềm đó phải tuân theo. Giấy phép mã nguồn mở được sử dụng cho các phần mềm mã nguồn mở.

The Open Source Definition là định nghĩa được đề xuất bởi the Open Source Initiative, được sử dụng để mô tả có giấy phép đủ điều kiện là giấy phép "mã nguồn mở".

The Open Source Initiative (OSI) cũng xác nhận giấy để chứng nhận rằng họ thuộc The Open Source Definition.

Những nguyên tắc cơ bản của bản quyền mã nguồn mở:

Giấy phép mã nguồn mở bắt buộc không độc quyền khai thác thương mại các tác phẩm được cấp phép

Mã nguồn mở phải là mã nguồn đã được viết sẵn cho các công việc.

Mã nguồn mở phải cho phép việc tạo ra các sản phẩm đã được chỉnh sửa cho riêng người lập trình.

Mã nguồn mở không chỉ có nghĩa là được tiếp cận mã nguồn. Các điều khoản cho việc phát hành phần mềm nguồn mở phải tuân thủ với các tiêu chí sau đây:

* Miễn phí phát hành lại: Giấy phép không được yêu cầu trả tiền bản quyền, lệ phí khác để bán.
* Mã nguồn: Chương trình phải bao gồm mã nguồn, và phải có giấy phép phát hành trong mã nguồn.
* Nguồn gốc tác phẩm: Giấy phép phải cho phép sửa đổi và trích dẫn công việc, và phải cho phép họ được phát hành sản phẩm theo các điều khoản tương tự như giấy phép của phần mềm ban đầu.
* Tính toàn vẹn mã nguồn của tác giả: Giấy phép có thể hạn chế mã nguồn được phát hành mà không cho chỉnh sửa chỉ khi giấy phép cho phép phát hành "các tập tin bản vá" với mã nguồn cho các mục đích của việc sửa lỗi chương trình trong thời gian phát triển.
* Không phân biệt đối xử với người nào hoặc Nhóm nào: Giấy phép không được phân biệt đối xử chống lại bất kỳ người hoặc bất kỳ nhóm người nào.
* Không phân biệt đối xử với Lĩnh vực: Giấy phép không được hạn chế bất cứ ai sử dụng chương trình trong một lĩnh vực cụ thể nào.
* Phát hành của Giấy phép: Các quyền gắn liền với chương trình phải áp dụng cho tất cả các chương trình được phát hành lại mà không cần thiết phải thực hiện của một giấy phép bổ sung bởi các bên.
* Giấy phép không được cụ thể cho một sản phẩm: Các quyền gắn liền với chương trình không được phụ thuộc vào phần chương trình của một nhà phân phối phần mềm cụ thể.
* Giấy phép không được hạn chế các phần mềm khác: giấy phép không được đặt các hạn chế đối với các phần mềm khác được phân phối cùng với phần mềm được cấp phép.
* Giấy phép phải có công nghệ trung lập: Không có điều khoản của giấy phép được xác định dựa trên bất kỳ công nghệ cá nhân hoặc phong cách của giao diện

##### Các loại giấy phép thông dụng

GNU

BSD

MIT

Aristic

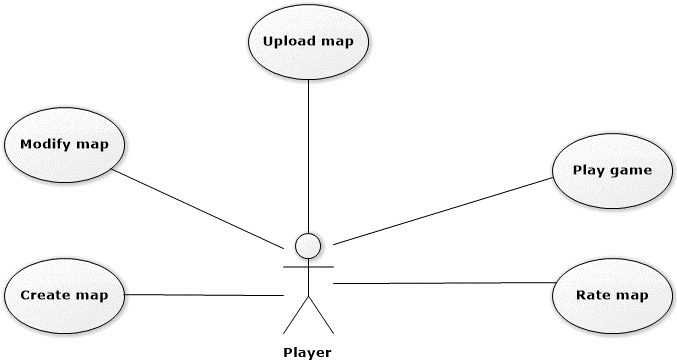
Apache

# HIỆN THỰC HÓA

## Client-side

### Các use case chính

#### Mô hình Use-case



Hình . Sơ đồ Use-case

* Danh sách Actor:

|  |  |  |
| --- | --- | --- |
| **STT** | **Tên Actor** | **Ý nghĩa/ghi chú** |
| 1 | Player | Người chơi game |

Bảng . Danh sách Actor

* Danh sách Use-case:

|  |  |  |
| --- | --- | --- |
| **STT** | **Tên Use-case** | **Ý nghĩa/ghi chú** |
| 1 | Create map | Người chơi tạo 1 map mới |
| 2 | Modify map | Người chơi chỉnh sửa map của mình |
| 3 | Upload map | Người chơi upload map lên server |
| 4 | Play game | Người chơi chơi game |
| 5 | Rate map | Người chơi đánh giá 1 map |

Bảng . Danh sách use-case

#### Đặc tả Use-case

* Create map:

|  |
| --- |
| * Tóm tắt: |
| * Tác nhân: ngươi chơi game. * Use-case này cho phép người chơi tạo thêm 1 map mới theo ý thích của mình |
| * Dòng sự kiện: |
| * Dòng sự kiện chính:  1. Người chơi bấm chọn Map editor trong main menu. 2. Người chơi bấm nút tạo map mới. 3. Người chơi thêm các đối tượng gạch, đá theo ý thích mình. 4. Người chơi nhấn nút ok để test thử map của mình. |
| * Các yêu cầu đặc biệt: |
| * Không có. |
| * Trạng thái hệ thống khi bắt đầu Use-case: |
| * Game đã được khởi động |
| * Trạng thái hệ thống khi kết thúc Use-case: |
| * Một scene mới được tạo ra để player test game |

Bảng . Đặc tả use-case create map

* Modify map:

|  |
| --- |
| * Tóm tắt: |
| * Tác nhân: ngươi chơi game. * Use-case này cho phép người chơi chỉnh sửa map đã tạo |
| * Dòng sự kiện: |
| * Dòng sự kiện chính:  1. Người chơi bấm chọn Map editor trong main menu. 2. Người chơi bấm nút chỉnh sửa map. 3. Người chơi thêm các đối tượng gạch, đá theo ý thích mình. 4. Người chơi nhấn nút ok để test thử map của mình.  * Dòng sự kiện khác:   Không có. |
| * Các yêu cầu đặc biệt: |
| * Không có. |
| * Trạng thái hệ thống khi bắt đầu Use-case: |
| * Game đã được khởi động |
| * Trạng thái hệ thống khi kết thúc Use-case: |
| * Một scene mới được tạo ra để player test game |

Bảng . Đặc tả use-case modify map

* Upload map:

|  |
| --- |
| * Tóm tắt: |
| * Tác nhân: ngươi chơi game * Use-case này cho phép người chơi upload map đã tạo lên server |
| * Dòng sự kiện: |
| * Dòng sự kiện chính:  1. Người chơi đã tạo map và đang test map 2. Người chơi test thành công (đã finish) 3. Người chơi nhấn nút upload  * Dòng sự kiện khác:   2.1: người chơi thất bại do tạo map quá khó |
| * Các yêu cầu đặc biệt: |
| * Thiết bị người chơi phải có kết nối internet |
| * Trạng thái hệ thống khi bắt đầu Use-case: |
| * Game đã được khởi động * Server đã được khởi động |
| * Trạng thái hệ thống khi kết thúc Use-case: |
| * Client hiển thị kết quả upload map * 1 map mới được lưu thêm vào database của server |

Bảng . Đặc tả use-case upload map

* Play game:

|  |
| --- |
| * Tóm tắt: |
| * Tác nhân: ngươi chơi game. * Use-case này cho phép người chơi chọn và chơi game theo map mình chọn |
| * Dòng sự kiện: |
| * Dòng sự kiện chính:  1. Người dùng nhấn nút play ở main menu 2. Người dùng chọn map được hiển thị 3. Chương trình chuyển sang scene mới cho phép người chơi map 4. Người chơi finish màn chơi 5. Client gửi kết quả lên cho server  * Dòng sự kiện khác:   4.1: người chơi thất bại không vượt màn được  5.1: client không gửi kết quả lên do người chơi thất bại |
| * Các yêu cầu đặc biệt: |
| * Thiết bị người chơi phải có kết nối internet |
| * Trạng thái hệ thống khi bắt đầu Use-case: |
| * Game đã được khởi động * Server đã được khởi động |
| * Trạng thái hệ thống khi kết thúc Use-case: |
| * Client hiển thị kết quả chơi game * Server update kết quả chơi game của map đó |

Bảng . Đặc tả use-case play game

* Rate map:

|  |
| --- |
| * Tóm tắt: |
| * Tác nhân: ngươi chơi game * Use-case này cho phép người sử dụng đánh giá 1 map nào đó |
| * Dòng sự kiện: |
| * Dòng sự kiện chính:  1. Người chơi chọn map muốn đánh giá 2. Người chơi đánh giá map 3. Client gửi request đánh giá map lên server 4. Server trả response tương ứng  * Dòng sự kiện khác:   4.1 response trả về rỗng do vấn đề kết nối mạng |
| * Các yêu cầu đặc biệt: |
| * Thiết bị người chơi phải có kết nối internet |
| * Trạng thái hệ thống khi bắt đầu Use-case: |
| * Game đã được khởi động * Server đã được khởi động |
| * Trạng thái hệ thống khi kết thúc Use-case: |
| * Client hiển thị kết quả rate map * Server update rating của map đó trong database |

Bảng . Đặc tả use-case rate map

### Các màn hình

#### Sơ đồ màn hình

Hình . Sơ đồ màn hình

#### Mô tả các màn hình

##### Màn hình chính

Giới thiệu

Là màn hình đầu tiên khi người dùng khởi động game



Hình . Màn hình chính (1)

|  |  |
| --- | --- |
| **STT** | **Ý nghĩa** |
| 1 | Nút Race: Vào màn hình bảng xếp hạng để chọn map chơi game |
| 2 | Nút Mapediter: Vào màn hình chọn tạo map, hoặc tiếp tục map đã tạo. |
| 3 | Nút Setting: xuất hiện thanh cài đặt âm thanh và thông tin về game |

Bảng . Bảng chú thích cho màn hình chính

Các layer

Hiện tại màn hình này chỉ có 1 layer duy nhất chứa background và các menu item

Cocos component sử dụng

Layer

Scene

Sprite

Menu item image

Director

Node

##### Màn hình bảng xếp hạng

Giới thiệu

Là màn hình hiển thị bảng xếp hạng hiện tại, được xếp theo thứ tự từ cao xuống thấp, hiện tại có 2 bảng là

* Bảng xếp hạng các map được rate cao nhất
* Bảng các map được upload gần nhất

****

Hình . Màn hình bảng xếp hạng

|  |  |
| --- | --- |
| **STT** | **Ý nghĩa** |
| 1 | Nút Back: Trở về màn hình chính. |
| 2 | TableViewCell: Bấm vào cell để vào chơi map mà mình muốn. |
| 3 | Nút NEW: Sang chế độ danh sách map mới nhất. |
| 4 | Nút TOP: Sang chế độ danh sách map có lượt rate và chơi nhiều nhất. |
| 5 | TableView: Lướt lên xuống để xem danh sách. |

Bảng . Bảng chú thích cho màn hình bảng xếp hạng

Các layer

Cocos component sử dụng

Layer

Scene

Sprite

Menu item image

Director

Node

HttpClient

UserDefault

##### Màn hình chọn thông tin map

Giới thiệu

Là màn hình cho phép người dùng lựa chọn tạo map mới hay edit map hiện tại

* Màn hình khi đã có sẵn map

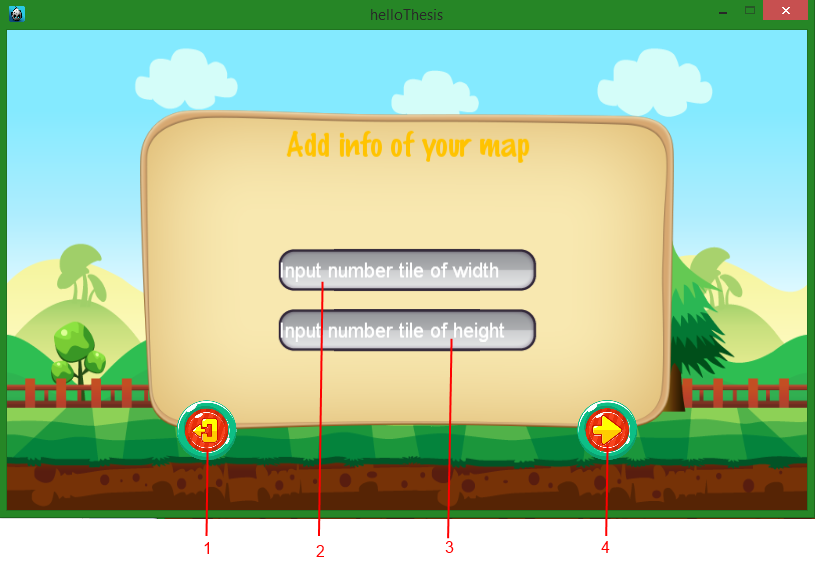
****

Hình . Màn hình khi có sẵn map

|  |  |
| --- | --- |
| **STT** | **Ý nghĩa** |
| 1 | Nút Back: Trở lại màn hình chính |
| 2 | Nút New: Tạo mới map, chuyển sang màn hình nhập thông số |
| 3 | Nút Next: Sang màn hình chỉnh sửa map với map đã lưu sẵn. |

Bảng . Bảng chú thích cho màn hình khi có sẵn map

* Màn hình thêm mới thông tin Map

****

Hình . Màn hình thêm mới map

|  |  |
| --- | --- |
| **STT** | **Ý nghĩa** |
| 1 | Nút Back: Trở về màn hình chính |
| 2 | EditBox Width: Nhập số tile chiều dài  + Chỉ được nhập số  + Số nhập vào lớn hơn 10 nhỏ hơn 500 |
| 3 | EditBox Height: Nhập số tile chiều cao  + Chỉ được nhập số  + Số nhập vào lớn hơn 10 nhỏ hơn 300 |
| 4 | Nút Tiếp tục: Chuyển sang màn hình chỉnh sửa map với thông số map vừa được tạo. |

Bảng . Bảng chú thích cho màn hình thêm mới map

Các layer

Màn hình này gồm 1 layer duy nhất

Cocos component sử dụng

Layer

Scene

Sprite

Menu item image

Director

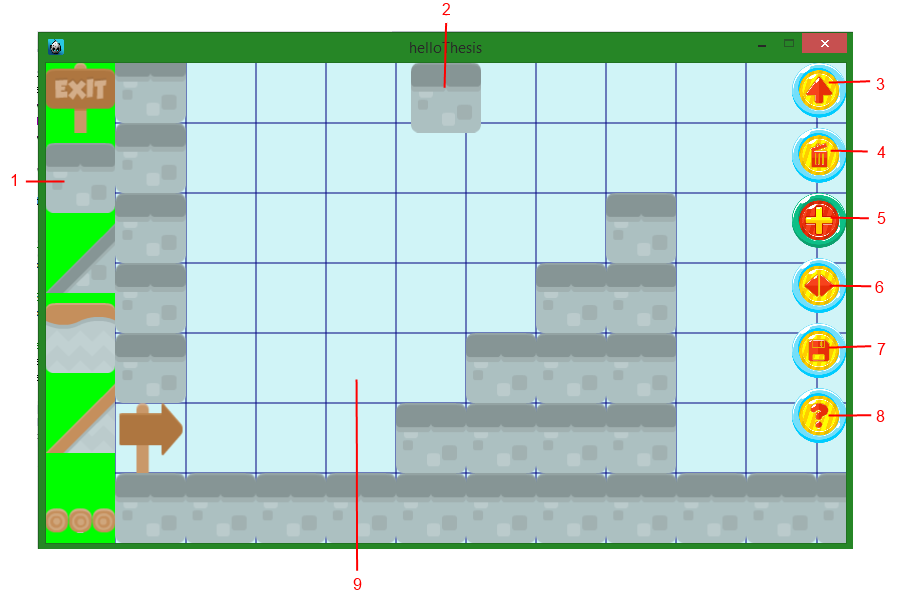
Node

UserDefault

##### Màn hình chỉnh sửa map

Giới thiệu

Là màn hình chính khi người chơi tương tác chỉnh sửa map



Hình . Màn hình chỉnh sửa map

|  |  |
| --- | --- |
| **STT** | **Ý nghĩa** |
| 1 | Hiển thị các đối tượng để thêm vào trong map. Bấm vào hình trong list view này để chọn đối tượng vẽ vào trong map. |
| 2 | Hình ảnh tượng trưng cho đối tượng đang được chọn |
| 3 | Nút Move: Chuyển về chế độ di chuyển. Trong chế độ này sẽ không thể xóa hoặc thêm đối tượng vào map, chỉ có thể vuốt để di chuyển map. |
| 4 | Nut Delete: Chuyển sang chế độ xóa. Trong chế độ này sẽ không thể di chuyển được map, hoặc thêm đối tượng vào map, chỉ có thể xóa đối tượng khi chạm vào đối tượng trên map. |
| 5 | Nút Add: Chuyển sang chế độ thêm mới. Trong chế độ này sẽ không thể di chuyển được map, hoặc xóa đối tượng trong map, chỉ có thể thêm đối tượng khi chạm vào ô bất kì trên map. |
| 6 | Nút Flip: lật đối tượng hình xiêng. |
| 7 | Nút Save: Lưu map ở bộ nhớ máy. Chuyển sang chế độ test map. |
| 8 | Nút Help: Chuyển sang hướng dẫn sử dụng map. |

Bảng . Bảng chú thích cho màn hình chỉnh sửa map

Các layer

Màn hình này chỉ có 1 layer duy nhất

Component cocos sử dụng

ScrollView: Toàn bộ map nằm trong này, lướt lên xuống để di chuyển tới vùng cần chỉnh sửa trong map.

DrawNode: Vẽ các đường thằng thông qua drawLine để tạo nên khung hình dạng lưới trong ScrollView

ListView: Chứa dánh sách các đối tượng thêm vào map, kéo lên xuống để sử dụng

Layer

Scene

Sprite

Menu item image

Director

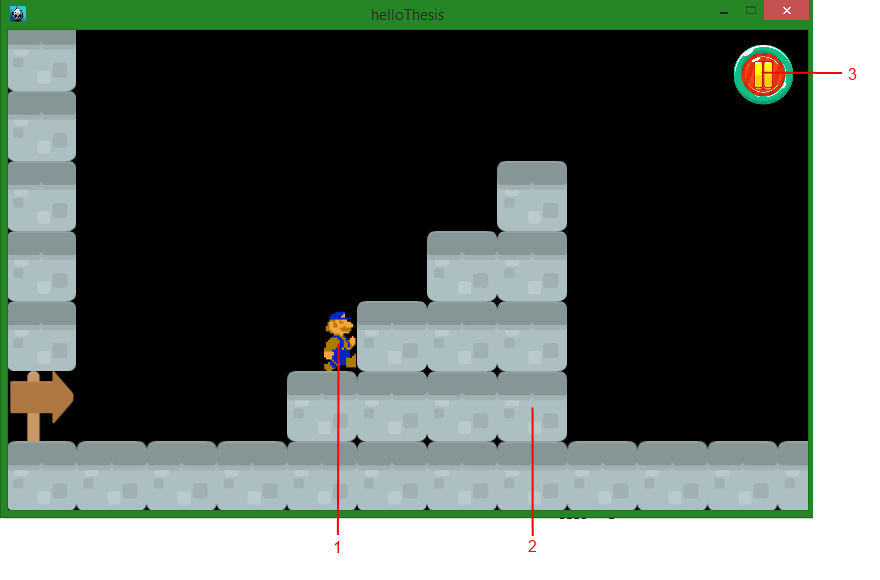
Node

##### Màn hình chơi game:

Giới thiệu

Là màn hình chính khi chơi game với thông tin map đã được load sẵn (từ server hay database local)

* Màn hình chơi game chính

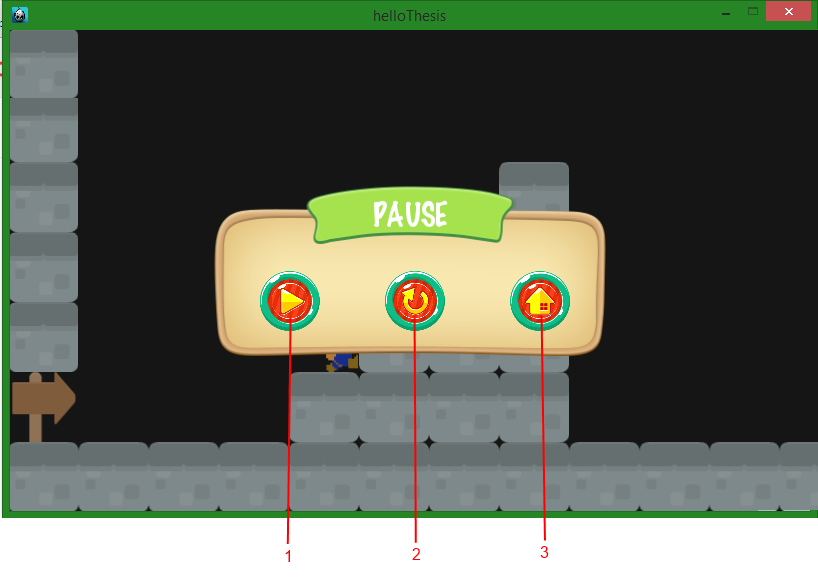
****

Hình . Màn hình chơi game chính

|  |  |
| --- | --- |
| **STT** | **Ý nghĩa** |
| 1 | Nhân vật di chuyển trong game. |
| 2 | Các đối tượng vật lý tương tác trong game với nhân vật. |
| 3 | Chuyển sang trạng thái tạm dùng chơi |

Bảng . Bảng chú thích cho màn hình chơi game chính

* Màn hình chơi game khi tạm dừng



Hình . Màn hình chơi game khi tạm dừng

|  |  |
| --- | --- |
| **STT** | **Ý nghĩa** |
| 1 | Nút Play: Tiếp tục chơi game |
| 2 | Nút Replay: Chơi lại game với map hiện tại. |
| 3 | Nút Home: Trở lại màn hình chính. |

Bảng . Bảng chú thích cho màn hình chơi game khi tạm dừng

* Màn hình chơi game khi hoàn thành thử map



Hình . Màn hình chơi game khi hoàn thành thử map

|  |  |
| --- | --- |
| **STT** | **Ý nghĩa** |
| 1 | Nút Back: trở lại màn hình chọn thông tin map |
| 2 | Nút Upload: Tải map hiện tại lên server. |

Bảng . Bảng chú thích cho màn hình chơi game khi hoàn thành thử map

Các layer

Background: chứa background của game

Setting: chứa các button setting ingame

Object: chứa các object trong game: người chơi, vật cản, v.vv

PauseLayer: hiển thị các object khi người chơi pause game

Upload map: hiển thị b

Cocos component sử dụng

Box2d

Sprite

Scene

Layer

Event listener

Action

Camera

Node

Cocos studio integration

FileUtils

Menu item, menu

Director

Label

User default

http client

## Server-side

### Kiến trúc

#### So sánh các kiến trúc

Qua khảo sát, nhóm thu được bảng so sánh các kiến trúc sau:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Dễ sử dụng | Uyển chuyển | Dễ mở rộng | Dễ bảo trì | Bảo mật |
| Client-server | ✔ |  |  | ✔ | ✔ |
| Component based |  | ✔ | ✔ |  |  |
| N-tier | ✔ |  | ✔ | ✔ | ✔ |
| Message oriented |  |  | ✔ | ✔ | ✔ |
| Object oriented |  |  | ✔ |  |  |
| Service oriented |  |  | ✔ |  |  |

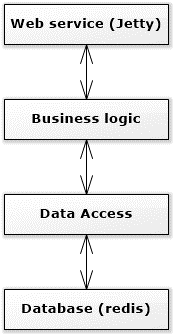
Bảng . So sánh các kiến trúc server

Qua bảng so sánh, nhóm quyết định chọn kiến trúc N-tier với những lý do sau:

* Dễ sử dụng, mở rộng, bảo trì
* Quen thuộc
* Hiện tại chưa có kinh nghiệm với các kiến trúc còn lại

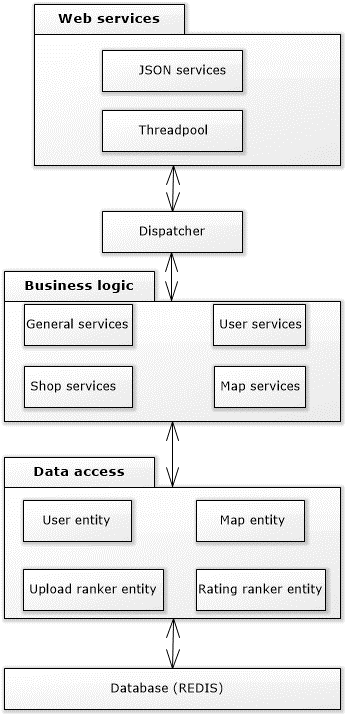
#### Mô hình kiến trúc server

##### Mô hình giản lược



Hình . Mô hình giản lược

##### Mô hình chi tiết



Hình . Mô hình chi tiết

### Tầng web service

#### Cấu hình Jetty

* Version: 9.3
* Listening port: 9999
* Context path: /runner
* Thread pool: org.eclipse.jetty.util.thread.QueuedThreadPool
* Thread size:
  + Min: 100
  + Max: 2000
* Time out: 60 sec
* Handler: org.eclipse.jetty.servlet.ServletHandler

#### Cấu hình JSON response

* Header:
  + Access-Control-Allow-Origin 🡪 \*
  + Cache-Control 🡪 no-cache, no-store, must-revalidate
  + Pragma 🡪 no-cache
  + Expires 🡪 0
* Content type: application/json;charset=UTF-8
* Encoding : UTF-8

### Tầng business logic

Tầng này được thiết kế chia ra nhiều lớp như sau:

* User services: xử lý các request liên quan tới user, ví dụ: create user, login
* Map services: xử lý các request liên quan tới map, ví dụ như: upload map, rate map, comment map.
* Shop services: xử lý các request liên quan tới mua đồ trong game
* General services: xử lý các request không thuộc nhóm trên

### Tầng data access layer

#### Hash entity

Là các entity có số lượng thuộc tính cố định, bao gồm:

* User: lưu giữ các thông tin của user ví dụ như: user id, user name, tiền, avata.
* Map: lưu trữ các thông tin của 1 map trong game, ví dụ như: map id, creator id,

Các lệnh dùng để tương tác với database Redis:

* hgetall: trả về tất cả fields và values tương ứng của hash tại key truyền vào
* hmset: set fields và values tương ứng của hash tại key truyền vào

#### List entity

Lưu trữ danh sách có thứ tự, bao gồm

* Upload ranker entity: lưu trữ có thứ tự các map id theo điều kiện xếp hạng là thời gian upload, map càng mới thì thứ hạng càng cao và ngược lại.
* Rating ranker entity: lưu trữ có thứ tự các map id theo điều kiện xếp hạng là thời gian upload, map càng mới thì thứ hạng càng cao và ngược lại.

Các lệnh dùng để tương tác với database redis

rpush: chèn thêm 1 phần tử vào bên phải của list hiện tại

rpop: xóa 1 phần tử ở bên phải của list hiện tại

lrange: lấy n phần tử trong list ra

Lý giải nguyên nhân nhóm sử dụng list thay vì sorted set của redis: về cơ bản, list và sorted set của redis rất tương đồng với vector và set container của c++ như sau:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Add | Remove | Lookup |
| set | add - O(log(n)) | erase - O(log(n)) | set.find - O(log(n)) |
| sorted set | zadd - O(log(n)) | zrem - O(log(n)) | zrange - O(log(n)) |
| list | lpush/rpush - O(1) | lpop/rpop - O(1) | lrangeO(n) |
| vector | push\_back - O(1) | pop\_back - O(1) | std::find - O(n) |

Bảng . Bảng so sánh set, sorted set, list, vector

Theo đó, ta nên dùng set/ sorted set khi yêu cầu thỏa mãn những điều kiện sau:

* Số lượng phần tử có thể tăng lên rất lớn khiến cho có sự khác biệt đáng kể giữa O(n) và O(log(n)) 🡨 thỏa
* Số lần lookups tương đương với số lần add 🡨 không chắc chắn
* Các phần tử được add vào 1 cách random, chứ không phải là theo thứ tự 🡨 không chắc chắn
* Add và remove được thực hiện xen kẽ nhau 🡨 không thỏa do bảng xếp hạng cứ sau 1 khoảng thời gian mới update 1 lần (hiện đang là 2h)

Xét thấy thỏa ½ điều kiện, tuy nhiên, có nguyên nhân chính khác như sau, nếu xài sorted set của redis, mỗi lần muốn thêm/ xóa/ lấy thông tin records thì đều phải gọi tới database, khá tốn chi phí (với trường hợp database đặt ở máy khác, ta sẽ phải chờ thêm 1 khoảng thời gian do vấn đề đường truyền mạng), trong khi với list, ta có thể tạo ra 1 arraylist trên memory rồi lấy trực tiếp trên đó, nhanh hơn rất nhiều, nên nhóm quyết định dùng list thay cho sorted set

### Tầng database

#### Cấu hình redis

* Version: 2.8.8
* Port: 6379
* Time out: 5000ms
* Max active : 10
* Max wait: 10
* Max idle: 10

#### Danh sách các “bảng”

Mặc dù redis là 1 nosql database dạng key-value, tuy nhiên, để dễ quản lý, các key được tạo ra đều tuân theo 1 pattern, ví dụ như sau: giả sử có 1 bảng User với khóa chính là Uid, với cái field là Name, Avata, Password; thì với redis, ta sẽ tạo 1 key User là 1 list trong đó chứa các Uid, đồng thời cứ với mỗi Uid, ta tạo thêm 1 key mới có dạng User:(uid), là 1 hash, có các field là Name, Avata, Password.

Như vậy, với cấu trúc như trên, thì trong database có các “bảng” chính sau:

* gr:maps:(mapID): lưu trữ thông tin của map tương ứng với mapID
* gr:phone\_id\_mapping: lưu trữ thông tin mã máy tương ứng với user id nào
* gr:u:maps: lưu trữ danh sách các map mà user tạo ra
* gr:u:(uid): lưu trữ thông tin của user

# THỬ NGHIỆM

## Quy trình thử nghiệm.

### Các thành phần của hệ thống thử nghiệm.

Các thành phần cần chuẩn bị để thử nghiệm hệ thống bao gồm:

* Về phần mềm:
  + Visual studio 2013
  + Xcode
* Về phần cứng:
  + Một máy điện thoại sử dụng hệ điều hành android
  + Một máy điện thoại sử dụng hệ điều hành iOs

### Tiến hành thử nghiệm.

* Chạy game trên android
  + Tạo map mới
  + Upload map lên
  + Chơi map vừa tạo (với danh sách map từ server gửi về)
  + Share facebook
* Lặp lại tương tự với iOs

## Một vài hình ảnh kết quả thực nghiệm

### Màn hình chơi game

Map được test có kích thước là 500x500

#### Không sử dụng quadtree



Hình . Màn hình chơi game không sử dụng quadtree

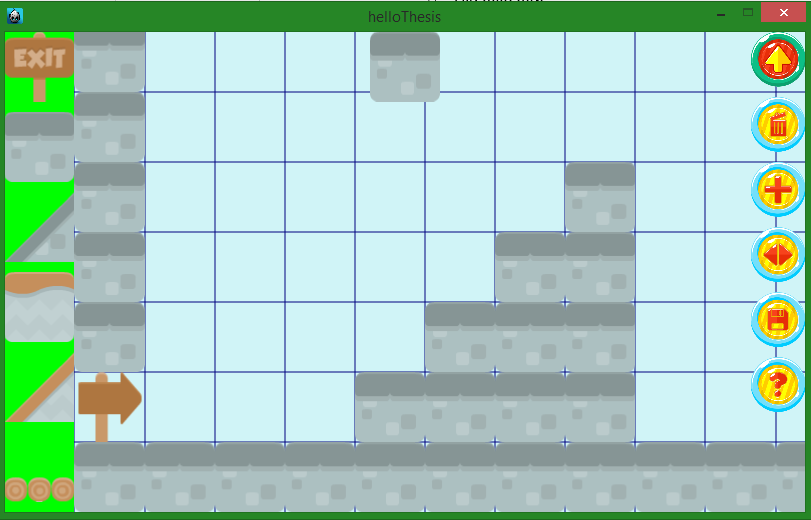
#### Sử dụng quadtree



Hình . Màn hình chơi game sử dụng quadtree

Dễ thấy, khi sử dụng quadtree, performance tốt hơn hẳn

### Màn hình chỉnh sửa map



Hình . Màn hình chỉnh sửa map

### Màn hình bảng xếp hạng map



Hình . Màn hình bảng xếp hạng map

# KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

## Kết quả đạt được

### Đối với đề tài

* Performance tốt, fps luôn ở mức 60
* Đã hoàn thành gameplay cơ bản
* Đã có thể tạo map và upload map lên
* Đã có thể download map về và chơi trên điện thoại của mình
* Đã có thể share facebook để thách thức bạn bè vào chơi

### Đối với các thành viên trong nhóm

* Hiểu được quy trình phát triển game.
* Hiểu được cấu trúc của engine Cocos2d-x.
* Biết cách xây dựng game từ engine Cocos2d-x.
* Ứng dụng được các kiến thức đã tìm hiểu được từ cocos2dx, game server vào việc thực hiện đồ án.
* Ôn luyện lại đồng thời bổ sung thêm kiến thức về ngôn ngữ C++ và Java
* Trau dồi thêm kỹ năng lập trình cũng như tư duy logic và khả năng xử lý vấn đề
* Kỹ năng làm việc nhóm của các thành viên được tăng cao, nhất là trong trường hợp 2 thành viên ở xa không thường xuyên gặp nhau được
* Khả năng quản lý, sắp xếp, phân chia công việc hợp lý được rèn luyện và hoàn thiện hơn

## Hạn chế

* Game play còn đơn giản
* Resource (hình ảnh, âm thanh) còn nhiều thiếu sót.
* Còn nhiều lỗi

## Hướng phát triển

### Về mặt chức năng

* Thêm chế độ multiplayer
* Tạo thêm hệ thống quest mới những map đã được tạo sẵn, để người chơi có thể chinh phục game từng phần, giúp người chơi gắn bó với game lâu hơn
* Thêm hệ thống shop cho phép người chơi dùng tiền (trong game hoặc ngoài) để mua thêm đồ đạc tạo map.

### Về mặt kỹ thuật

Config để có thể build ra bản cho window phone

## Những khó khăn và cách giải quyết

|  |  |
| --- | --- |
| Khó khăn | Giải quyết |
| Các thành viên chỉ là coder, không phải designer nên gặp khó khăn khi thiết kế game | Tham khảo các tool tương tự như mario maker + tham khảo ý kiến giảng viên hướng dẫn |
| Resource game không có | Cố gắng tìm các resource free trên mạng |
| Các thành viên không có điều kiện gặp nhau do ở xa + đi làm full time trên công ty | Làm việc, quản lý source code bằng github, sử dụng các công cụ để liên lạc khác như mạng xã hội, email để liên lạc bàn bạc, cố gắng gặp nhau cuối tuần |
| Chưa có kinh nghiệm làm game sever | Tìm kiếm tham khảo các project tương tự, tự customize lại cho phù hợp |

Bảng . Khó khăn và cách giải quyết

## Những bài học thu được

Qua đề tài, chúng em đã thu được những bài học quý báu, nổi bật nhất là:

* Không được chủ quan, luôn đánh giá đúng tình hình thực trạng
* Khi lên ý tưởng game, không lên quá mơ mộng xa vời dẫn tới trường hợp không thể thực hiện được trong thời gian yêu cầu 🡪 đầu voi đuôi chuột

.

TÀI LIỆU THAM KHẢO

**Tài liệu tiếng anh**

[1] Roger Engelbert (2015), Cocos2d-x by Example, Beginner’s guide

[2] Frahaan Hussain, Arutosh Gurung (2014), Cocos2d-x game development essentials

[3] Jonathan S. Harbour (2014), Beginning game programming, second edition

**Tài liệu tham khảo từ internet**

[4] Cocos reference <<http://www.cocos2d-x.org/wiki/Reference>>,

[5] Cocos document <<http://www.cocos2d-x.org/docs/README> > truy cập 10/9/2015

[6] redis command <<http://redis.io/commands> > truy cập ngày 14/10/2015

[7] Jetty document <<http://www.eclipse.org/jetty/documentation/current/> > , truy cập ngày 9/10/2015

[5] Why should not use set <<http://lafstern.org/matt/col1.pdf> >, truy cập ngày 30/10/2015

[6] The Great Java Application Server Debate with Tomcat, JBoss, GlassFish, Jetty and Liberty Profile <<http://zeroturnaround.com/rebellabs/the-great-java-application-server-debate-with-tomcat-jboss-glassfish-jetty-and-liberty-profile/> >, truy cập ngày 5/10/2015

[7] Difference between socket programming and Http programming <<http://stackoverflow.com/questions/15108139/difference-between-socket-programming-and-http-programming> > truy cập ngày 5/10/2015

[8] Introduction to redis <<http://www.slideshare.net/dvirsky/introduction-to-redis/> > truy cập ngày 14/10/2015

**Phần mềm tham khảo**

[9] Warcraft 3 world editor

[10] Mario maker Wii U