

TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP THỰC PHẨM

TP HỒ CHÍ MINH

KHOA CÔNG NGHỆ THÔNG TIN



BÁO CÁO THỰC TẬP TỐT NGHIỆP

NGÀNH : Công Nghệ Thông Tin

CÔNG TY : GAMELOFT VIETNAM

SINH VIÊN THỰC HIỆN : Trần Trung Hiếu

TP.HCM, Ngày 01 Tháng 04 Năm 2015

TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP THỰC PHẨM

TP HỒ CHÍ MINH

KHOA CÔNG NGHỆ THÔNG TIN



BÁO CÁO THỰC TẬP TỐT NGHIỆP

ĐỀ TÀI: **GAME CROSS PLATFORM**

WITH ENGINE COCOS2D-X

Giảng Viên Hướng Dẫn : Trần Thanh Phước

Người hướng dẫn tại công ty : Lê Minh Tài

Trần Duy Khánh

La Tâm

Sinh Viên Thực Hiện : Trần Trung Hiếu

Lớp : 02DHTH

Mã Số Sinh Viên : 2001110117

TP.HCM, Ngày 01 Tháng 04 Năm 2015

LỜI CẢM ƠN

Để hoàn thành khóa thực tập tại đầu tiên em xin gửi lời cảm ơn đến quý thầy cô trong khoa Công Nghệ Thông Tin trường Đại Học Công Nghiệp Thực Phẩm TP HCM đã tận tình dạy dỗ và truyền đạt nhưng kiến thức căn bản nhất trong suốt 4 năm đại học. Và em gửi lời cảm ơn đến Thầy **Trần Thanh Phước** đã dùi cát Em trong những năm học qua.

Đặc biệt em xin cảm ơn các Anh/Chị trong team Training của công ty GameLoft Việt Nam đã tận tình chỉ dạy và giúp đỡ em trong suốt quá trình thực tập. Và Em chân thành cảm ơn Anh **Trần Duy Khánh** – người trực tiếp hướng dẫn em tại team Training.

Trong hai tháng thực tập, em đã học hỏi được rất nhiều kiến thức mới, công nghệ mới và đã củng cố lại được nhiều kiến thức em được học tại trường, tuy nhiên không thể không còn thiếu sót. Mong các Anh/Chị trong team training thông cảm.

Em xin chân thành cảm ơn!

DANH MỤC BẢNG

Bảng 1: Một số khái niệm cơ bản của SVN	18
Bảng 2: Môi trường lập trình C++.....	24
Bảng 3: Kiểu dữ liệu cơ bản trong C++	26
Bảng 4: Toán tử cơ bản trong C++.....	27
Bảng 5: Ngôn ngữ lập trình hỗ trợ lập trình Cocos2d-x.....	35
Bảng 6: Công cụ hỗ trợ lập trình game với Engine Cocos2d-x	35
Bảng 7: Các file cần thiết để cấu hình cài đặt Cocos2d-x trên Windows và link download	36
Bảng 8: Các biến môi trường cần thiết trong quá trình cài đặt	40
Bảng 9: Các Action cơ bản trong Cocos2d-x	46
Bảng 10: Công cụ hỗ trợ dùng trong làm game “Internship AH”	68
Bảng 11: Phân công công việc làm game Internship AH!	71
Bảng 12: Chỉ số của file thông tin bản đồ game.....	86
Bảng 13: Chỉ số các loại quái(Enemi) trong game.....	87
Bảng 14: Chỉ số các loại trụ (Tower) trong game	89

DANH MỤC HÌNH ẢNH

Hình 1: Cơ chế lưu trữ của Tortoise SVN.....	17
Hình 2: Tạo Repository cho server.....	19
Hình 3: Checkout không gian làm việc từ server về client	20
Hình 4: Cấu hình đường dẫn khi Checkout	20
Hình 5: So sánh sự khác nhau bằng "Show Differences"	22
Hình 6: Cấu trúc của một chương trình C++.....	25
Hình 7: Game TicTacToe	30
Hình 8: Kiểm tra tính đúng đắn dữ liệu đầu vào game TicTacToe.....	31
Hình 9: Thêm Project thư viện dạng lib	31
Hình 11: Thêm thư viện OpenGL(.lib) ở Input - Dependencies	33
Hình 10: Thêm thư viện OpenGL (.h, .lib) ở Directories.....	33
Hình 12: Thư mục cài đặt JDK.....	36
Hình 13: Thư mục cài đặt Python.....	37
Hình 14: Thêm file qtjava.zip để build Cocos2d-x	37
Hình 15: Thư mục chuẩn bị đầy đủ cài đặt Cocos2d-x	38
Hình 16: Thêm biến môi trường cho Windows.....	38
Hình 17: Thẻ Environment Variables để thêm biến môi trường	39
Hình 18: Tạo project game cocos2d-x với lệnh CMD	40
Hình 19: Thư mục project game coos2d-x sau khi tạo (template của cocos2d-x)	41
Hình 20: Project Game Cocos2d-x "Hello World"	41
Hình 21: Bố trí các Layer trong màn hình game	42
Hình 22: Quản lý các màn hình trong game (Scene)	42
Hình 23: Tạo project với TexturePacker	52
Hình 24: Giao diện Tool TexturePacker để tạo Sprite Sheet	52
Hình 25: Public resource Sprite Sheet để tạo Animation cho nhân vật	53
Hình 26: Lưu Resource Sprite Sheet (.plish).....	53
Hình 27: Hoàn thành Publish Sprite Sheet	54
Hình 28: Thêm sprite sheet vào resource của project game	54
Hình 29: Thêm "Tiled "để tạo map	56
Hình 30: Khởi tạo mới 1 bản đồ bằng Tiled Map Editor	57
Hình 31: Chọn Tile để vẽ cho bản đồ.....	58
Hình 32: Thêm Tiled để vẽ bản đồ	58
Hình 33: Thiết kế bản đồ cho game trên Tiled Map Editor	59
Hình 34: Lưu bản đồ dưới định dạng .tmx	59
Hình 35: Thêm layer Backgound cho bản đồ	60
Hình 36: Đánh dấu vật cản trong bản đồ để kiểm tra va chạm	61

Hình 37: Đối tượng đánh dấu vị trí xuất hiện của nhân vật trong bản đồ	61
Hình 38: Hoàn thành thiết kế bản đồ.....	62
Hình 39: Giao diện game khi tải bản đồ lên màn hình	63
Hình 40: Giới thiệu game Internship AH!	68
Hình 41: Flow Chart của game Internship AH!	73
Hình 42: Phân tích hướng đối tượng trong game Internship AH!.....	74
Hình 43: Đối tượng Game Object	75
Hình 44: Đối tượng quái trong game Internship AH! (Enemi)	77
Hình 45: Đối tượng quái Bandit trong game Internship AH!	77
Hình 46: Đối tượng quái Goblin trong game Internship AH!	77
Hình 47: Đối tượng quái Orc trong game Internship AH!	78
Hình 48: Đối tượng trụ trong game Internship AH! (Tower)	79
Hình 49: Đối tượng trụ Archer trong game Internship AH!	80
Hình 50: Đối tượng trụ BomBand trong game Internship AH!	80
Hình 51: Đối tượng trụ XXX trong game Internship AH!	81
Hình 52: Đối tượng quản lý tài nguyên trong game Internship AH!	81
Hình 53: Đối tượng quản lý các bản đồ trong game Internship AH!	82
Hình 54: Các lớp tùy chỉnh Sprite dùng trong game Internship AH!	83
Hình 55:Đối tượng lưu trữ thông tin người chơi game Internship AH!	83
Hình 56: Các màn hình trong game.....	85
Hình 57: Các Enum đánh dấu các Sprite cho các đối tượng	85
Hình 58: File map.txt.....	86
Hình 59: File "enemies.txt"	88
Hình 60: File "towers.txt"	90
Hình 61: Tạo bản đồ cho game Internship AH! bằng Tiled Map Editor.....	91
Hình 62: Đánh dấu trên bản đồ vị trí xây trụ và đường đi của quái	92
Hình 63: Lưu đối tượng các đường của quái	93
Hình 64: Lưu đối tượng vị trí bắt đầu xuất hiện của quái trên bản đồ	93
Hình 65: Lưu vị trí xây các trụ trên bản đồ	94

MỤC LỤC

LỜI CẢM ƠN.....	1
DANH MỤC BẢNG	2
DANH MỤC HÌNH ẢNH.....	3
MỤC LỤC	5
PHẦN 1: TỔNG QUAN.....	11
Chương 1: GIỚI THIỆU CHUNG	11
Chương 2: GIỚI THIỆU CÔNG TY NƠI THỰC TẬP.....	13
Chương 3: NỘI DUNG CÔNG VIỆC VÀ KẾT QUẢ ĐẠT ĐƯỢC	15
Chương 4: TRẢI NGHIỆM BẢN THÂN.....	16
Chương 5: ĐỊNH HƯỚNG NGHỀ NGHIỆP	16
PHẦN 2: QUẢN LÝ PHIÊN BẢN VỚI SVN	17
Chương 1: QUẢN LÝ PHIÊN BẢN.....	17
1.1. Thế nào là quản lý phiên bản?	17
1.2. Tại sao phải sử dụng?	17
Chương 2: PHẦN MỀM “TORTOISE SVN”	18
2.1. Tortoise SVN là gì?	18
2.2. Cách sử dụng “Tortoise SVN”	18
2.2.1. Một số khái niệm cơ bản của SVN.....	18
2.2.2. Các tạo Server	18
2.2.3. Đối với Client	19
2.2.4. Update	21
2.2.4. Commit	21
2.2.5. Switch revision	21
2.2.6. Revert	21
2.2.7. Merge.....	21
2.2.8. Conflict.....	21
2.2.9. Show differences	21
2.2.9. Clean up.....	22

PHẦN 3: NGÔN NGỮ LẬP TRÌNH C++.....	23
Chương 1: NGÔN NGỮ LẬP TRÌNH	23
1.1. Biên dịch là gì?	23
1.2. Thế nào là ngôn ngữ lập trình.....	23
Chương 2: BẮT ĐẦU VỚI C/C++	24
2.1. Môi trường lập trình.....	24
2.2. Hướng dẫn ngôn ngữ lập trình C++.....	24
2.2.1. Giới thiệu	24
2.2.2. Cơ bản của ngôn ngữ lập trình C++	25
i. Cấu trúc chương trình	25
ii. Biến và kiểu dữ liệu	25
iii. Hằng số	26
iv. Toán tử	26
v. Cơ bản Nhập/Xuất dữ liệu trong C++.....	27
2.2.3. Cấu trúc chương trình.....	28
i. Cấu trúc điều khiển	28
ii. Vòng lặp.....	28
iii. Lệnh ngắn	28
iv. Lệnh nhảy.....	28
v. Hàm	29
vi. Ghi đè hàm (Overloaded)	29
2.2.4. Kiểu dữ liệu phức tạp	29
i. Mảng	29
ii. Chuỗi ký.....	29
iii. Con trỏ	29
iv. Kiểu dữ liệu khác	29
2.2.5. Hướng đối tượng	29
2.2.6. Tính năng của ngôn ngữ lập trình C++	29
i. Ép kiểu (Implicit – Explicit)	29

ii. Type Casting	29
ii. Preprocessor.....	29
Chương 3: BÀI TẬP THỰC HÀNH.....	30
3.1. Bài 1: Learn how to build/use a static Library (.Lib). Example game TicTacToe	30
3.2. Bài 2: Physic Engine Framework	33
PHẦN 4: TÌM HIỂU LÀM GAME VỚI ENGINE COCOS2D-X.....	34
Chương 1: TÌM HIỂU ENGINE COCOS2D-X	34
1.1. Cocos2d-x là gì?	34
1.2. Tính năng chính	34
1.3. Build Requirements	34
1.4. Runtime Requirements	34
1.5. Ngôn ngữ lập trình	35
1.6. Công cụ hỗ trợ	35
Chương 2: BẮT ĐẦU VỚI ENGINE COCOS2D-X	36
2.1. Download và chuẩn bị file cần thiết	36
2.2. Cài đặt.....	36
2.3. Cơ bản một số lệnh sử dụng trong “cmd’ của Windows	40
2.4. Game đầu tiên “Hello World”	40
Chương 3: MODULES	42
3.1. Các thành phần chính trong Cocos2d-x	42
3.1.1. Director.....	42
3.1.2. Scene	43
3.1.2. Layer.....	43
3.1.3. Scene Graph	43
3.1.4. Action	43
3.1.5. Sprite	43
3.2. Giao diện trong Cocos2d-x (UI)	43
3.2.1. Label	43
3.1.2. <i>Label Effects</i>	43

3.1.3. Menu và MenuItem	43
3.1.4. Button	44
3.1.5. CheckBox	44
3.1.6. LoadingBar	44
3.1.7. Slider	44
3.1.8. TextField	44
3.3. Di chuyển giữa các màn hình (Transition Scene).....	44
3.3. Các loại lắng nghe sự kiện.....	44
3.4. Actions	44
3.4.1. Basic Actions.....	44
3.4.2. Sequences and how to run them	47
3.5. Animation	47
3.6. Touch	47
3.6.1. Single – touch events.....	47
3.6.2. Multi – touch events	48
3.7. Audio	48
3.7.1. Effect	48
3.7.2. Music	49
Chương 4: TÍNH NĂNG NÂNG CAO.....	50
4.1. Physics Engine.....	50
4.1. TileMap.....	50
Chương 5: CÔNG CỤ HỖ TRỢ	51
5.1. TexturePacker	51
5.1.1. SpriteSheet là gì?.....	51
5.1.2. Download and cài đặt phần mềm hỗ trợ.....	51
5.1.3. Tạo Spritesheet cho Cocos2d-x sử dụng TexturePacker.....	52
5.1.3.1. Tạo Sprite Sheet với TexturePacker	52
5.1.3.2. Sử dụng Sprite Sheet cho Cocos2d-x.....	54
5.2. Tile Map Editor.....	56

5.2.1. Tiled Map Editor là gì?	56
5.2.2. Download and cài đặt	56
5.2.3. Tạo bản đồ cho game với Tiled Map Editor	56
5.2.3.1. Tạo bản đồ với MapTiled with Tiled	56
5.2.3.2. Tải map lên game Cocos2d-x	60
PHẦN 5:LÀM GAME “INTERNSHIP AH!” VỚI COCOS2D-X.....	68
Chương 1: GIỚI THIỆU “Game Internship AH!”	68
1.1. Giới thiệu game Internship AH!	68
1.2. Công cụ hỗ trợ	68
1.3. Thông tin nhóm thực hiện.....	69
1.4. Task.....	69
Chương 2: BẮT ĐẦU THỰC HIỆN	72
2.1. Lên ý tưởng và xây dựng GamePlay	72
2.2. Flow Chart	73
2.3. Phân tích hướng đối tượng.....	73
2.4. Chuẩn bị tài nguyên cho game.....	86
2.3.1. Resource Tile Map	90
2.3.2. Resourc Image	91
2.3.3. Resourc Sprite Sheet	91
2.3.4. Resourc Audio.....	91
2.5. Thiết kế bản đồ cho game	91
2.6. Thiết kế Sprite Sheet.....	94
2.6.1. Sprite Sheet cho quái (Enemi).....	94
2.6.2. Sprite Sheet cho trụ (Tower)	94
2.7. Viết Code	94
2.1.1. Tạo các màn hình game theo Flow Chart.....	94
2.1.2. Xây dựng trụ	97
2.1.3. Thêm vật lý vào cho game.....	100
2.1.4. Kiểm tra va chạm giữa quái và trụ.....	101

2.1.5. Xây dựng hệ thống nâng tru.....	102
2.1.6. Xây dựng tính điểm cho người chơi và chuyển bản đồ khi thắng game.	102
TÀI LIỆU THAM KHẢO	105
Tài liệu training của GameLoft	105
Ebook Tham khảo	105
Websites tham khảo	105
KẾT LUẬN	106

PHẦN 1: TÔNG QUAN

Chương 1: GIỚI THIỆU CHUNG

Họ tên : **TRẦN TRUNG HIẾU**

Email : trunghieu4121993@gmail.com Số điện thoại : 0902.401.367

Lớp : 02DHTH Khoa : CNTT

MSSV : 2001110117

Công ty thực tập : GameLoft Việt Nam

Địa chỉ : Tầng 7, tòa nhà E-Town 2, 364 Cộng Hòa, TP HCM

Vị trí tại nơi thực tập : Thực tập sinh.

Chương trình thực tập bao gồm những nội dung chính sau:

- ✓ Đào tạo về các kỹ thuật phát triển game trên di động.
- ✓ Tìm hiểu quy trình phát triển game chuyên nghiệp trên điện thoại di động.
- ✓ Thực hành phát triển game trên điện thoại di động.
- ✓ Rèn luyện các kỹ năng làm việc nhóm, thực hiện dự án, kỹ năng giao tiếp, trình bày, báo cáo.

Quá trình thực tập tại GameLoft, Em trải qua quy trình sau:

- Em bắt đầu bằng cách **nộp hồ sơ** vào GameLoft với vị trí thực tập sinh (Internship). Khi được các Chị bên team nhân sự gọi lên làm bài test.
- **Làm bài test** (C++/Java) và đợi kết quả. Khi kết quả đạt sẽ được nhận lịch lên GameLoft để phỏng vấn.
- **Phỏng vấn** với 1 Anh/Chị thuộc team training. Và đợi kết quả. Khi kết quả đạt sẽ nhận được lịch lên Thực tập.

- Trong quá trình **thực tập** Em được truyền đạt 3 phần chính: SVN, C++, Cocos2d-x. Và kết thúc bằng 1 project Game làm theo nhóm.
- Cuối cùng là buổi **thuyết trình báo cáo** những gì đã học được trong suốt quá trình thực tập tại GameLoft.

Chương 2: GIỚI THIỆU CÔNG TY NƠI THỰC TẬP

GameLoft là công ty làm về lĩnh vực công nghệ thông tin, được thành lập năm 1993 với trụ sở chính đặt ở Pháp. GameLoft hiện có 28 studio đặt trên khắp các quốc gia, trong đó Việt Nam có 4 studio 2 cái ở HCM, 1 ở Hà Nội, 1 ở Đà Nẵng. Trong đó studio ở toàn nhà E.town 2 là lớn nhất với khoản 500 nhân viên và chiếm trọn 1 tầng của tòa nhà.



GameLoft có khoảng 5.000 nhân viên trong đó nhân viên ở Việt Nam chiếm tới 1.400 người. Phần lớn các nhân viên đều rất trẻ và nhân viên được làm việc trong môi trường chuyên nghiệp và rất thoải mái.

Studio của GameLoft nằm trọn trên tầng 7 của tòa nhà Etow 2, có khoảng 500 nhân viên trong đó có nhiều bạn là người nước ngoài. Không gian tại studio rất gọn gàng, ngăn nắp, sạch sẽ và đặc biệt mặc dù có hàng trăm người làm việc. Tại Studio nhân viên được phân bổ theo dạng từng nhóm, mỗi nhóm có số lượng khác nhau, làm một chuyên môn khác nhau như: Game 2d, Game 3d, Test Game, Viết Tool, bộ phận IT, Artist, ...

Văn hóa làm việc ở GameLoft rất thoải mái, có thể mặc bất cứ trang phục nào bạn thích. Thời gian làm việc từ 8-8h30 sáng đến 12h nghỉ trưa và 1h30 làm tiếp tới 5h30. Tuy nhiên mỗi nhóm có thể linh hoạt thời gian, miễn đảm bảo tiến độ công việc.

Tại GameLoft nhân viên có cơ hội làm việc các dự án trong môi trường toàn cầu với các đội ngũ khác nhau trên thế giới tại Mỹ, Nam Mỹ, Châu Âu, New York, ...

Đặc biệt ở GameLoft Việt Nam sẽ có cơ hội làm việc với nhiều công nghệ khác nhau (iOS, Android, WinPhone 8, ...) và các thiết bị chạy nền tảng Java đến những nền tảng phức tạp như tablet (Samsum galaxy Tab, Ipad, Blackbrry Playbook, ...)

Chương 3: NỘI DUNG CÔNG VIỆC VÀ KẾT QUẢ ĐẠT ĐƯỢC

CÔNG VIỆC	ĐÁNH GIÁ
Training Tortoise SVN	95%
Training C++ Language	80%
Training Engine Cocos2d-x	90%
Game Internship AH!	90%

Kiến thức học được:

- ✚ Học hỏi được nhiều kỹ năng làm việc như:
 - Cách làm việc cá nhân.
 - Cách làm việc nhóm.
 - Cách lên kế hoạch, viết báo cáo công việc.
 - Cách quản lý thời gian.

✚ Quy trình làm game:

1. Lên ý tưởng cho game
2. Hoàn thiện game play
3. Chuẩn bị Resource hình ảnh, Sprite Sheet để làm Animation . . .
4. Thiết kế OOP cho game.
5. Coding

✚ Kỹ thuật sử dụng trong làm game:

1. Thiết kế hình ảnh để làm tài nguyên cho game (Designer): Thiết kế giao diện màn hình trong game, các button, sprite trên màn hình.
2. Quản lý tài nguyên trong game.
3. Cách tạo Sprite Sheet để làm Animation cho các nhân vật trong game.
4. Sử dụng Tiled Map Editor để vẽ bản đồ trong game và đánh dấu điểm cần thiết trong game.

- Tổ chức lưu trữ các thông số của các đối tượng trong game xuống dạng file để dễ dàng nâng cấp, sửa đổi để điều chỉnh lại game. (Enemi, Tower, Map, ...)

Chương 4: TRẢI NGHIỆM BẢN THÂN

Có cơ hội tiếp xúc với môi trường thực tế tại một công ty làm game lớn hàng đầu Việt Nam, được tham gia vào khóa training của GameLoft. Gameloft một trong những số ít công ty có team training để đào tạo cho nhân viên. Được làm quen với làm việc với phong cách chuyên nghiệp.

Kết thúc khóa thực tập tại GameLoft, Em được rèn luyện, củng cố kiến thức ngôn ngữ lập trình C++, đặc biệt được biết thêm nhiều mảng kiến thức của ngôn ngữ C++ (Casting, PreProcessor, ...). Được tiếp cận với nhiều công nghệ mới đặc biệt những công nghệ làm game. Có cơ hội tiếp xúc với nhiều Anh/Chị có nhiều năm kinh nghiệm trong lĩnh làm game, đào tạo lập trình viên.

Chương 5: ĐỊNH HƯỚNG NGHỀ NGHIỆP

Hiểu được quy trình làm Game, cách kiếm lợi nhuận, kinh doanh từ game. Từ đó có thể hiểu được phần nào ngành “Công nghiệp làm game”. Vận dụng các tính năng ưu việt của các tính năng hỗ trợ trong các Engine khi làm game, đặc biệt làm game 2D với Cocos2d-x.

Định hướng được một team làm game không thể thiếu các nhân tố sau:

- Thiết kế, lên ý tưởng và xây dựng cách hoạt động của game (Game designer).
- Thiết kế và vẽ đồ họa cho game (Artist).
- Coder.
- Bộ phận marketing để quản bá game.

Trở thành một lập trình viên phát triển game đa nền tảng.

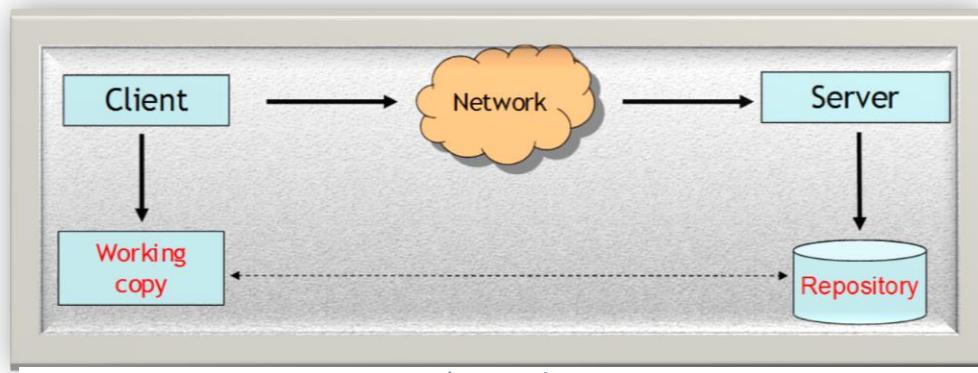
PHẦN 2: QUẢN LÝ PHIÊN BẢN VỚI SVN

Chương 1: QUẢN LÝ PHIÊN BẢN

1.1. Thế nào là quản lý phiên bản?

“Quản lý phiên bản” là khái niệm quản lý nhiều phiên bản của 1 dự án (Ứng dụng). Giúp giữ được nhiều phiên bản của 1 tập hợp thông tin. Dễ dàng lấy lại được những phiên bản trước đó ("Roll back").

Giảm sự chồng chéo trong công việc và cập nhật liên tục tình hình trong nhóm. Cơ chế



Hình 1: Cơ chế lưu trữ của Tortoise SVN

2.2. Tại sao phải sử dụng?

- Quản lý việc chia sẻ file, tài liệu: ngăn chặn việc xoá, sửa file , . . .
- Giữ lại các phiên bản của file, tài liệu trước đó.

Chương 2: PHẦN MỀM “TORTOISE SVN”

2.1. Tortoise SVN là gì?

Tortoise SVN công cụ giúp quản lý phiên bản dạng "Revision Control".

2.2. Cách sử dụng “Tortoise SVN”

2.2.1. Một số khái niệm cơ bản của SVN

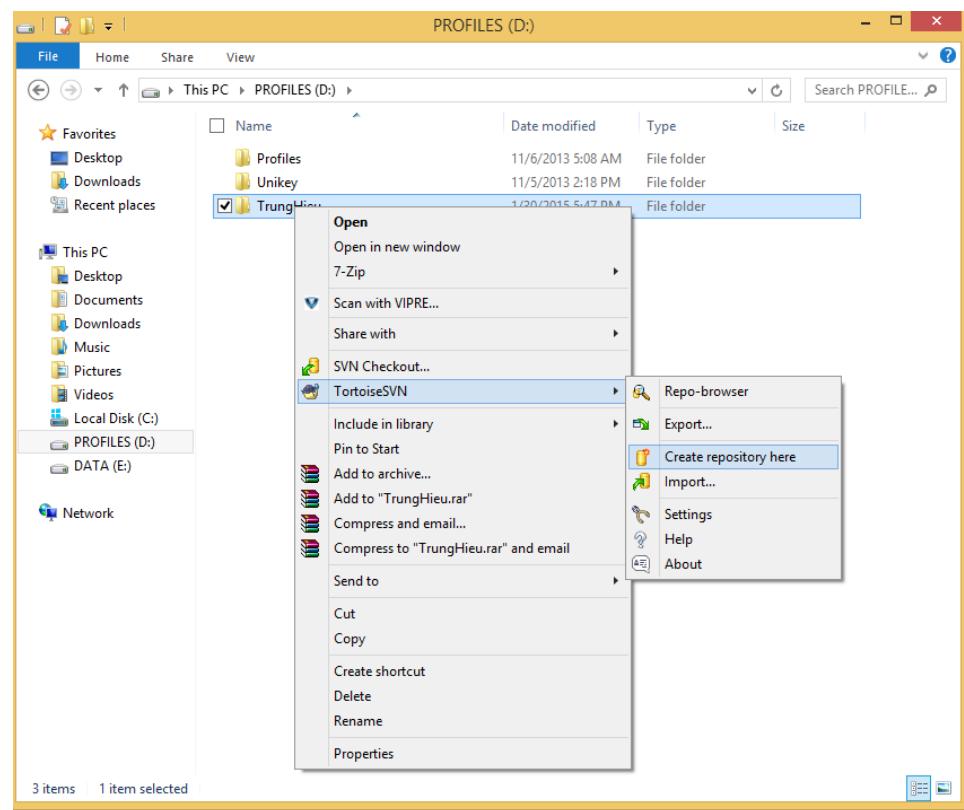
Khái niệm	Chức năng
Repository	Vùng lưu trữ dữ liệu trên server.
Working Copy	Vùng làm việc của Client tại máy.
HEAD	Phiên bản mới nhất trên
Revision	Phiên bản cũ

Bảng 1: Một số khái niệm cơ bản của SVN

2.2.2. Cách tạo Server

- Chọn thư mục để làm server lưu trữ.
- Create repository here.

Chọn 1 thư mục để chứa dữ liệu cho server



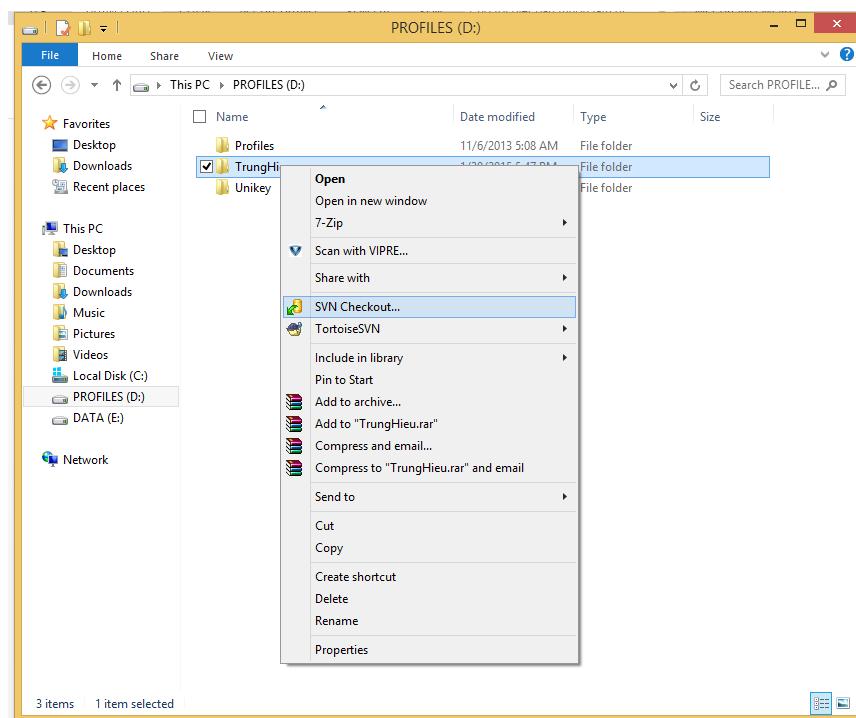
Hình 2: Tạo Repository cho server

2.2.3. Đối với Client

Để sử dụng file, tải liệu từ server (Check Out)

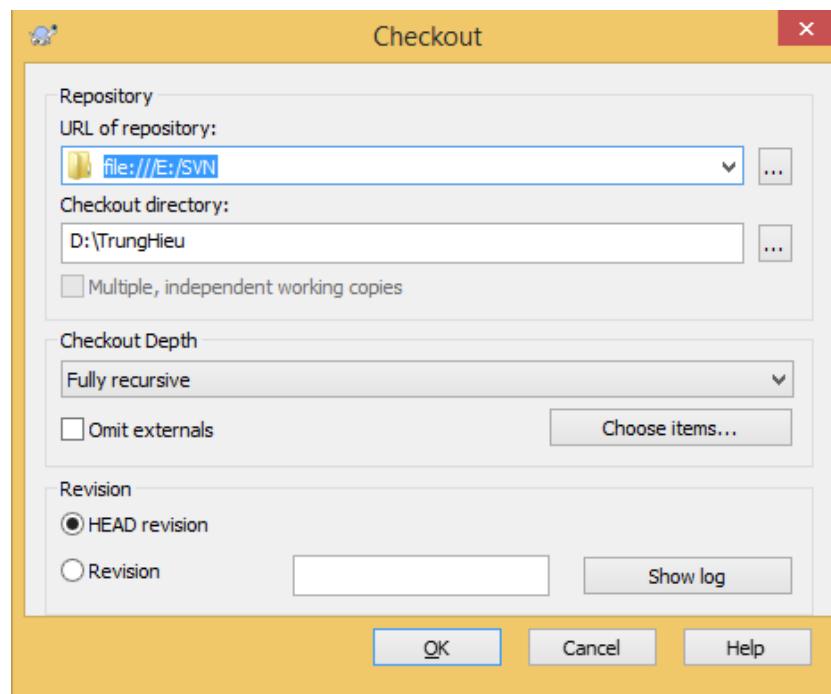
- Chọn thư mục lưu trữ.
- SVN Check out.

Chọn 1 thư mục để lưu dữ liệu check out từ server xuống:



Hình 3: Checkout không gian làm việc từ server về client

Cấu hình lại nơi lưu:



Hình 4: Cấu hình đường dẫn khi Checkout

2.2.4. Update

Lấy dữ liệu mới nhất từ Repository của server về không gian làm việc của máy Client. Thao tác:

Tortoise SVN → Update

2.2.4. Commit

Cập nhật dữ liệu từ không gian làm việc hiện tại lên Repository của server. Thao tác:

Tortoise SVN → Commit

2.2.5. Switch revision

Chọn một phiên bản cũ hơn HEAD để lưu về không gian làm việc hiện tại (Working copy). Thao tác:

Tortoise SVN → Switch...

2.2.6. Revert

Quay lui về một phiên bản cũ hơn để làm việc cho Repository của server. (Tức chọn 1 phiên bản cũ nào đó làm phiên bản mới nhất cho Repository trên server). Thao tác:

Tortoise SVN → Show Log

→ Chọn 1 phiên bản trong danh sách Log

→ Revert to this revision

2.2.7. Merge

Gôm dữ liệu các phiên bản lại thành phiên bản mới nhất cho Repository của Server. Thao tác:

Tortoise SVN → Merge

2.2.8. Conflict

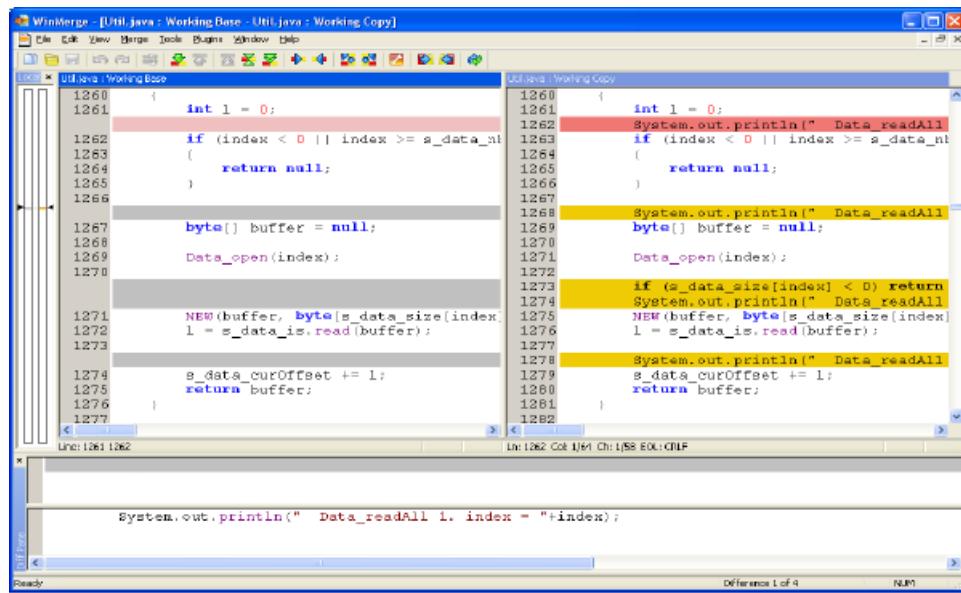
Xung đột dữ liệu giữa Working copy và Repository trên server (Do dữ liệu của người commit chưa phải là phiên bản mới nhất).

2.2.9. Show differences

Xem sự khác nhau giữa phiên bản hiện tại và phiên bản trước đó trên server. Thao tác:

Chọn file → Tortoise SVN → Show differences

Click chuột phải vào file đã chỉnh sửa: Tortoise SVN → Diff



Hình 5: So sánh sự khác nhau bằng "Show Differences"

2.2.9. Clean up

Xoá các file bị xung đột với server, Lấy bản mới nhất của Reposion của server về Working copy. Thao tác:

Tortoise SVN → Clean up...

PHẦN 3: NGÔN NGỮ LẬP TRÌNH C++

Chương 1: NGÔN NGỮ LẬP TRÌNH

1.1. Biên dịch là gì?

Máy tính chỉ hiểu một ngôn ngữ gồm thông tin bao gồm nhưng con số 0 và 1. Ngôn ngữ này được gọi là “Ngôn ngữ máy” chỉ gồm các bit 0 1 (Với tương ứng 0 là tín hiệu điện đóng và 1 là tín hiệu điện mở). Với “ngôn ngữ máy” chỉ bao gồm 0 và 1 thì quá trình làm việc với máy tính sẽ rất khó khăn và gây rất nhiều bất tiện.

Để khắc phục việc phải làm việc với “Ngôn ngữ máy” con người đã đưa ra tập hợp các cú pháp lệnh để thể hiện ngôn ngữ máy một cách dễ dàng hơn và thân thiện với người sử dụng hơn.

Để máy tính có thể hiểu được những tập hợp các cú pháp lệnh này, thì cần có quá trình chuyển đổi các cú pháp lệnh này sang “Ngôn ngữ máy” để máy tính có thể hiểu được.

1.2. Thế nào là ngôn ngữ lập trình

Ngôn ngữ lập trình là tập hợp các cú pháp lệnh giúp lập trình viên giao tiếp với tính một cách thuận tiện nhất nhờ các cú pháp của ngôn ngữ đó mà không cần phải dùng trực tiếp ngôn ngữ máy để làm việc với máy tính.

Khi lập trình viên sử dụng ngôn ngữ lập trình, để máy tính có thể hiểu được đều cần quá trình biên dịch để chuyển sang “Ngôn ngữ máy”.

Chương 2: BẮT ĐẦU VỚI C/C++

 *Lưu ý:* Hướng dẫn cài đặt trên môi trường Windows (Win 8.1 - 64bit)
Và sử dụng IDE Visual Studio Community 2013.

2.1. Môi trường lập trình

IDE	Platforms
Code::blocks	Windows
	MAC OS
	Linux
Visual Studio Express 2013	Windows

Bảng 2: Môi trường lập trình C++

2.2. Hướng dẫn ngôn ngữ lập trình C++

2.2.1. Giới thiệu

C++ là ngôn ngữ lập trình cấp cao được một nhà khoa học máy tính người Đan Mạch tên “Bjarne Stroustrup” xây dựng trong quá trình làm luận án tiến sĩ của ông.

C++ được chuẩn hóa bởi Tổ chức tiêu chuẩn hóa quốc tế (International Organization for Standardization – ISO) phiên bản tiêu chuẩn mới nhất được phê duyệt và công bố bởi ISO và ISO/IEC 14882:1998 vào tháng 9 năm 2011 (Còn gọi là C++ 11)

2.2.2. Cơ bản của ngôn ngữ lập trình C++

i. Cấu trúc chương trình

```

1 #include <iostream>
2
3 int main(int argc, const char * argv[]) {
4     // insert code here...
5     std::cout << "Hello, World!\n";
6     return 0;
7 }
8

```

Hình 6: Cấu trúc của một chương trình C++

- Chú thích trong chương trình (Comment)

// Chú thích code	Chú thích cho 1 dòng lệnh
/* Chú thích code dòng 1 Chú thích code dòng 2 , . . . */	Chú thích cho khối lệnh (nhiều dòng lệnh)

Sử dụng namespace STD: std là một “namespace” được hỗ trợ trong thư viện chuẩn của C++. Để sử dụng ta có thể dùng tên namespace để lấy các hàm đã được định nghĩa trong namespace:

“std::cout” hàm xuất ra màn hình console 1 chuỗi.

“std::cin” hàm nhận dữ liệu từ màn hình console.

Để tiện sử dụng các hàm mà k cần phải lặp lại tên namespace ta có thể khai báo sử dụng namespace theo cú pháp:

“using namespace std;”

ii. Biến và kiểu dữ liệu

- Cú pháp:

<Kiểu dữ liệu> <Tên biến> ;

- Gán giá trị cho biến:

`<Tên biến> = <Giá trị>;`

Kiểu dữ liệu cơ bản trong C++

Kiểu dữ liệu	Kích thước – bytes (32 bit)	Miền giá trị
char	1	Kí tự, số nguyên
bool	1	true or false
int	4	-32767 - 32767
unsigned int	4	0 - 66335
float	4	
double	8	

Bảng 3: Kiểu dữ liệu cơ bản trong C++

iii. Hằng số

- Cú pháp

`const <kiểu dữ liệu> <tên hằng> = <giá trị>`

- Cú pháp #define

`#define <Tên> <Giá trị>`

iv. Toán tử

Các loại toán tử

Loại toán tử	Các toán tử
Toán tử số học	+ - * / %
Toán tử so sánh	< , =, >, >=, <=, ==, !=
Toán tử logic	a

Toán tử trên bit	<code>^, >>, <<, , &. ~</code>
Toán tử gán	<code>=, +=, -=, *=, /=,</code>
Toán tử hỗn hợp	<code>Sizeof, ->, .. ::, new, delete, ..</code> .

Bảng 4: Toán tử cơ bản trong C++

v. Cơ bản Nhập/Xuất dữ liệu trong C++

- Xuất dữ liệu: sử dụng hàm “cout” trong namespace std của thư viện chuẩn `<iostream>`

Bước 1: Thêm thư viện

```
#include <iostream>
```

Bước 2:

Thêm khai báo sử dụng namespace

```
using namespace std;
```

Bước 3: Xuất dữ liệu của biến

```
int a = 2;  
  
cout << "a = " << a;
```

➔ Kết quả chương trình: a = 2

- Nhập dữ liệu

Bước 1: Thêm thư viện

```
#include <iostream>
```

Bước 2: Thêm khai báo sử dụng namespace

```
using namespace std;
```

Bước 3: Nhận dữ liệu cho biến

```
int a;  
  
cout << "Nhập a = ";  
  
cin >> a;
```

→ Màn hình sẽ hiện: Nhập a =

Và đợi ta nhập dữ liệu từ bàn phím vào

- Đối với chuỗi

```
#include <string.h>
```

```
string s;  
getline(cin, s); // nhập dữ liệu cho biến s
```

2.2.3. Cấu trúc chương trình

i. Cấu trúc điều khiển

- + if
- + if else
- + switch case

ii. Vòng lặp

- + for
- + while
- + do while

iii. Lệnh ngắn

- + break
- + return

+ continue

iv. Lệnh nhảy

- + goto

v. Hàm

vi. Ghi đè hàm (Overloaded)

2.2.4. Kiểu dữ liệu phức tạp

i. Mảng

+ Mảng 1 chiều .

+ Mảng nhiều chiều.

ii. Chuỗi ký

iii. Con trỏ

+ Cấp phát động (Toán tử new)

iv. Kiểu dữ liệu khác

+ typedef

+ Unions

+ enum

2.2.5. Hướng đối tượng

+ Lớp – đối tượng (class)

+ Tính chất của hướng đối tượng (Đóng gói, trừu tượng, Đa hình, kế thừa)

+ Mẫu design pattern (Singleton Design Pattern, . . .)

2.2.6. Tính năng của ngôn ngữ lập trình C++

i. Ép kiểu (Implicit – Explicit)

ii. Type Casting

+ dynamic_cast

+ static_cast

+ reinterpret_cast

+ const_cast

ii. Preprocessor

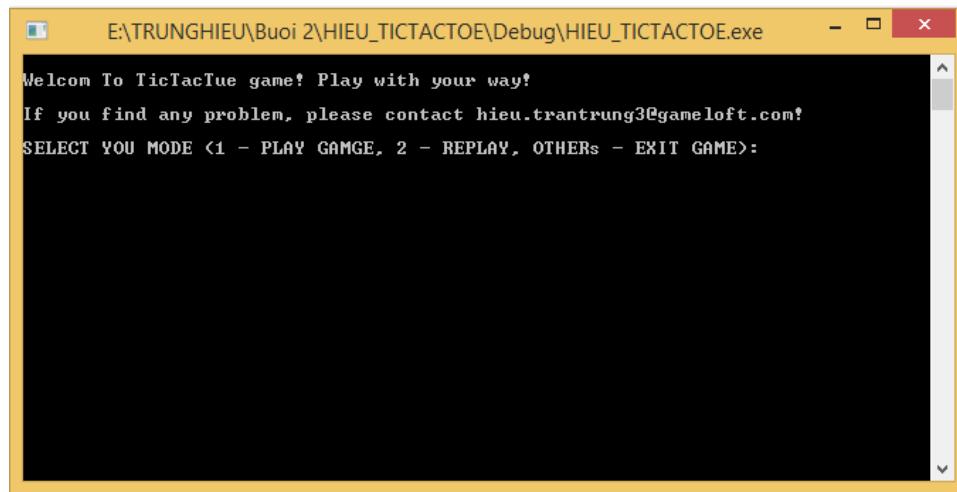
+ Khai báo macro (#define, #undef)

+ Thêm thư viện (#include <>, #include "")

- + Giải quyết xung đột thêm thư viện (#ifndef, #ifndef, #if, #endif, #else and #elif)

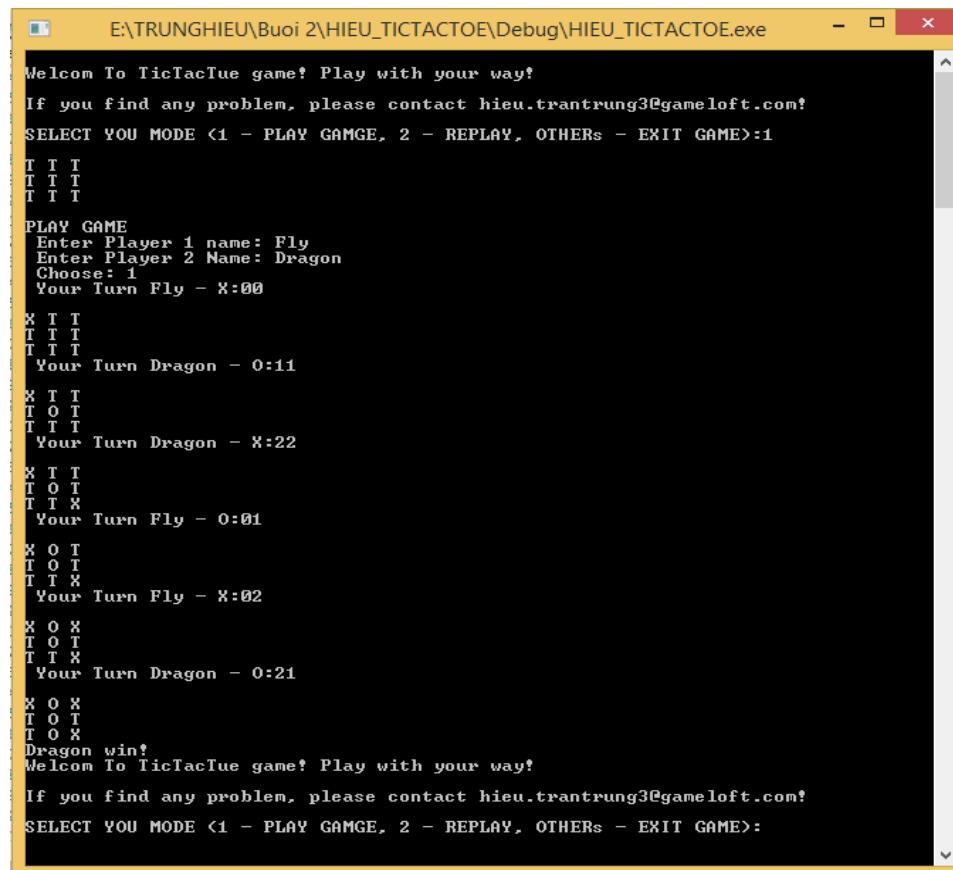
Chương 3: BÀI TẬP THỰC HÀNH

3.1. Bài 1: Learn how to build/use a static Library (.Lib). Example game TicTacToe



Hình 7: Game TicTacToe

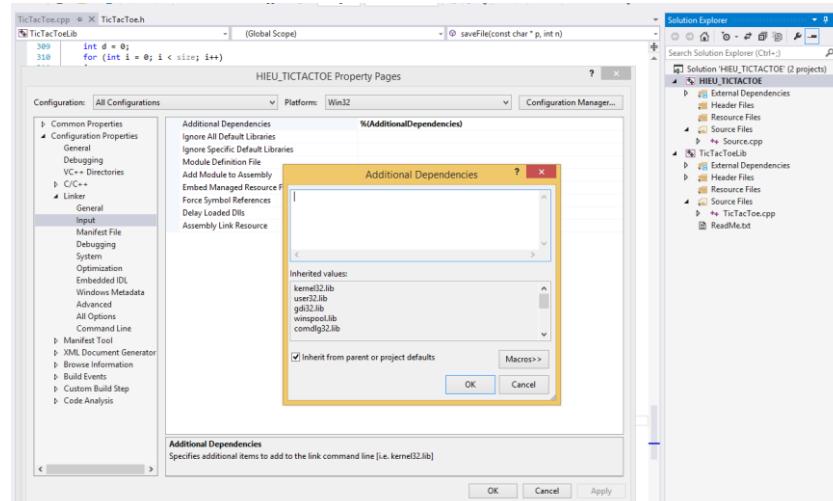
- **Task 1:** Tạo project TicTacToe và xây dựng những chức năng cho game.



Hình 8: Kiểm tra tính đúng đắn dữ liệu đầu vào game TicTacToe

Kiểm tra tính đúng đắn và bắt lỗi ngoại lệ do người dùng nhập vào.

- **Task 2:** Tạo thêm project làm thư viện (Vào Configuration Type -> chọn Static Library (.lib)). Chuyển tất cả source code qua project thư viện. Cấu



Hình 9: Thêm Project thư viện dạng lib

hình cho project chính sử dụng project thư viện vừa tạo:

Ở project chính chỉ có code gọi hàm.

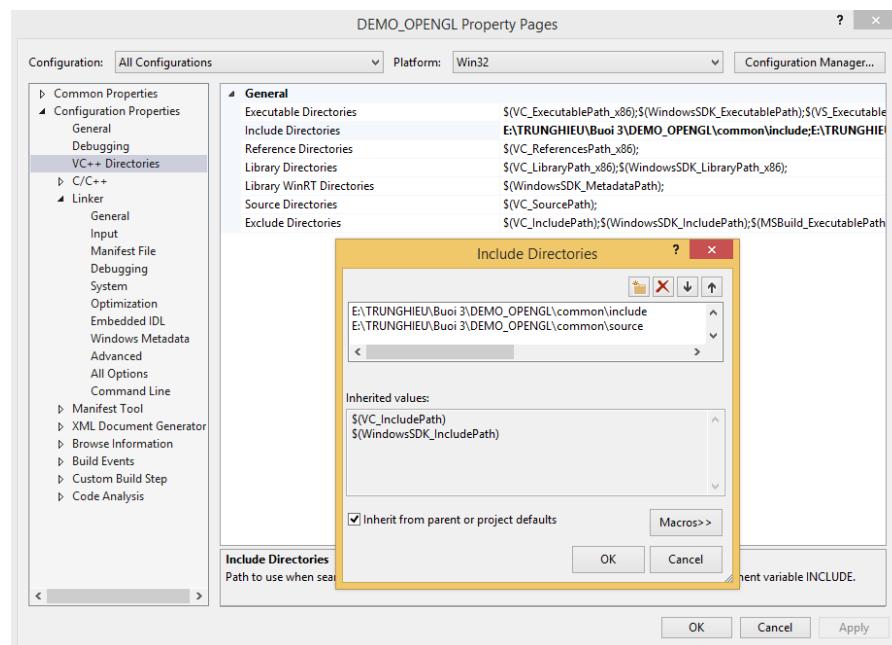
- **Task 3:** Thêm tính năng cho game TicTacToe

Ghi lại lịch sử lượt chơi của người chơi trước khi thắng game. (lưu vào file save.bin). Thêm chức năng xem lại game chơi trước đó. Commit project lên SVN.

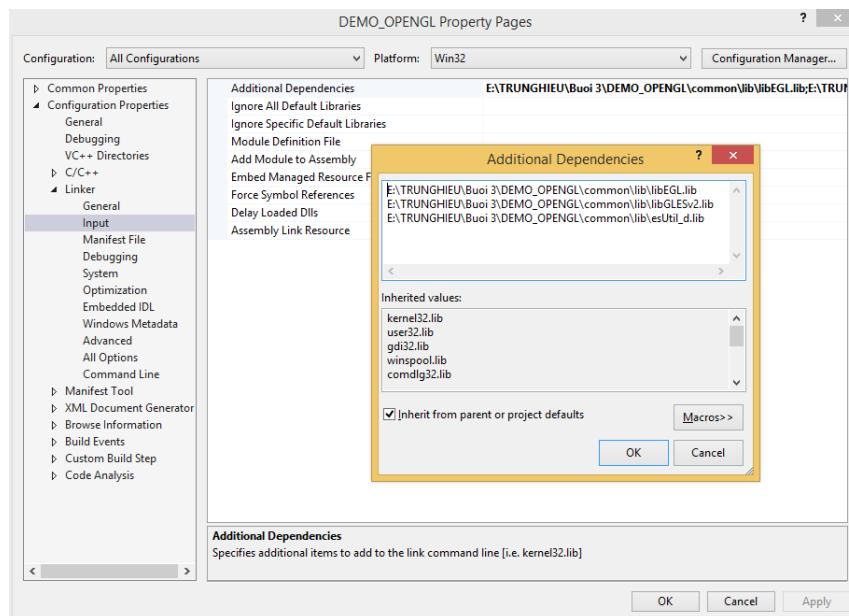
3.2. Bài 2: Physic Engine Framework

- **Task 0:** Cấu hình project thư viện <gl.h> để sử dụng OPENGL

Add thư viện gồm thư mục chứa .lib và .h:



Hình 11: Thêm thư viện OpenGL (.h, .lib) ở Directories



Hình 10: Thêm thư viện OpenGL(.lib) ở Input - Dependencies

PHẦN 4: TÌM HIỂU LÀM GAME VỚI ENGINE COCOS2D-X

Chương 1: TÌM HIỂU ENGINE COCOS2D-X

1.1. Cocos2d-x là gì?

Cocos2d-x là Framework mã nguồn mở viết bằng C++ được cấp giấy phép bởi MIT. Cocos2d-x tối ưu đồ họa 2D sử dụng OPENGL.

Cocos2d-x cho phép nhà phát triển có thể sử dụng ngôn ngữ lập trình C++, Lua và Javascript để triển khai trên các nền tảng iOS, Android, Windows Phone, Mac OS X, Windows Desktop và Linux.

1.2. Tính năng chính

- Scene Management
- 2D Graphics
- UI Widgets
- Physics
- Audio
- Network

1.3. Build Requirements

- Mac OS X 10.7+, Xcode 5.1+
- or Ubuntu 12.10, CMake 2.6+
- or Windows 7+, VS 2012+
- Python 2.7.5
- NDK r9d build Android games

1.4. Runtime Requirements

- iOS 5.0+ cho game trên iPhone / iPad.
- Android 2.3+ cho game trên Android.
- Windows Phone 8+ cho game trên Windows Phone.
- OS X v10.6+ cho game trên Mac.
- Windows 7+ cho game trên Win.

1.5. Ngôn ngữ lập trình

	Platforms	C++	Lua	JavaScript
Mobile	iOS	X	X	X
	Android	X	X	X
	WP8	X	X	
Desktop	Win32	X	X	X
	Mac OSX	X	X	X

Bảng 5: Ngôn ngữ lập trình hỗ trợ lập trình Cocos2d-x

1.6. Công cụ hỗ trợ

Features	Tool	Windows	Mac OS X	Linux
IDE	CocosStudio	X	X	X
Bimap Font Tools	Glyph Designer			
Particle Editing Tools	Particle Designer			
Texture Atlas Tools	Texture Packer			
	Zwoptext			
Tile map Editing Tools	Tiled Map Editor	X	X	X
Animation Editor	Spine	X	X	X
	Dragon Bones			

Bảng 6: Công cụ hỗ trợ lập trình game với Engine Cocos2d-x

Chương 2: BẮT ĐẦU VỚI ENGINE COCOS2D-X

Lưu ý: Hướng dẫn cài đặt trên môi trường Windows (Win 8.1 - 64bit) Và sử dụng IDE Visual Studio Community 2013.

2.1. Download và chuẩn bị file cần thiết

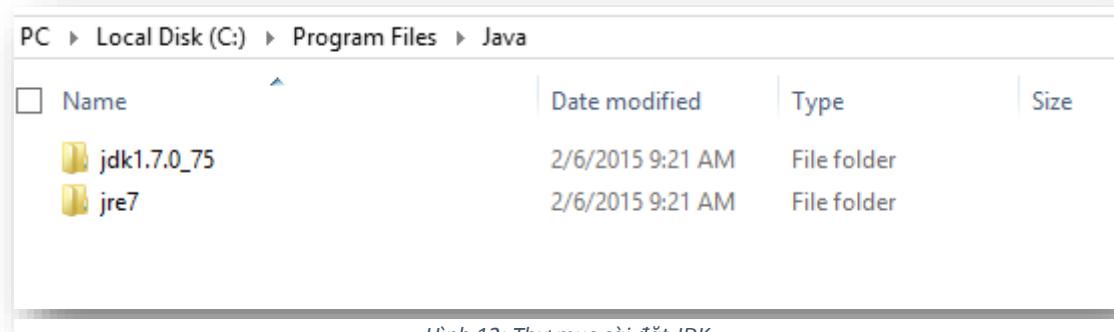
Download các file

	Link	Install
Visual Studio Community 2013	http://www.visualstudio.com/en-us/news/vs2013-community-vs.aspx	X
JDK	http://www.oracle.com/technetwork/java/javase/downloads/jdk7-downloads-1880260.html	X
Python	https://www.python.org/downloads/	X
SDK	https://developer.android.com/sdk/index.html#Other	
ANT	http://ant.apache.org/bindownload.cgi	
NDK	https://developer.android.com/tools/sdk/ndk/index.html	
Cocos2d-x	http://www.cocos2d-x.org/download	
File <i>qtjava.zip</i>	http://www.aldrizzagno.com/jdk-7-download-zip-file	

Bảng 7: Các file cần thiết để cấu hình cài đặt Cocos2d-x trên Windows và link download

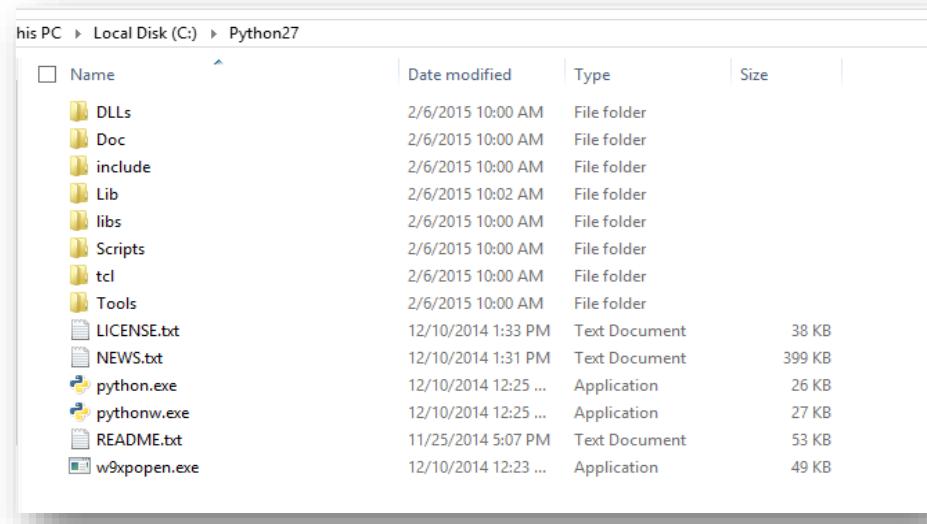
2.2. Cài đặt

- Cài đặt JDK



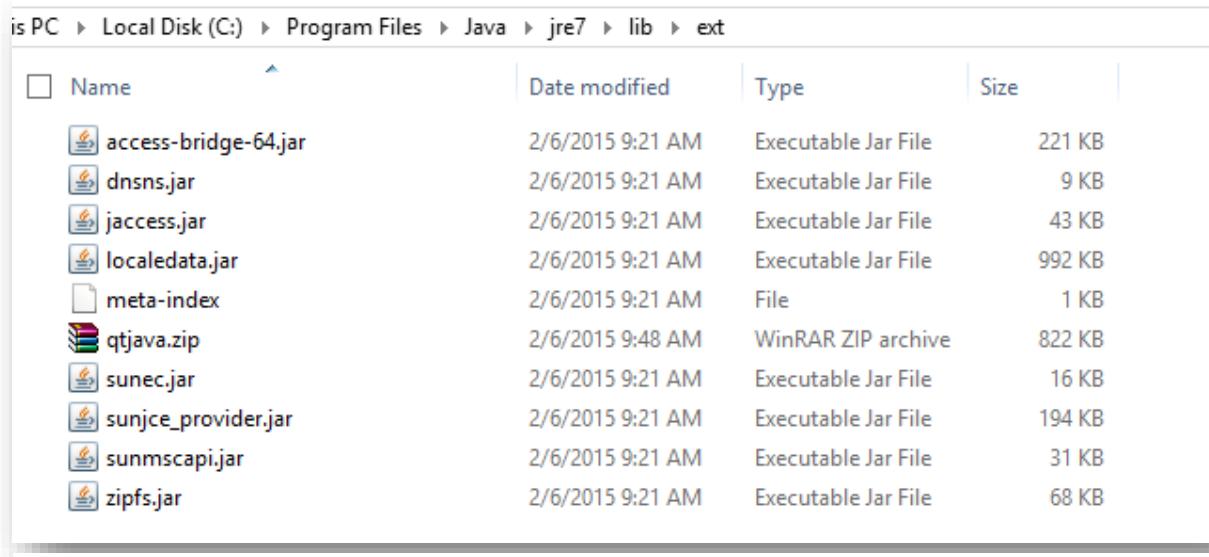
Hình 12: Thư mục cài đặt JDK

- Cài đặt Python



Hình 13: Thư mục cài đặt Python

Sau khi cài đặt JDK và python vào máy cần thêm file qtjava.zip vào thư mục ext theo đường dẫn [jre7 → lib → ext] như hình:



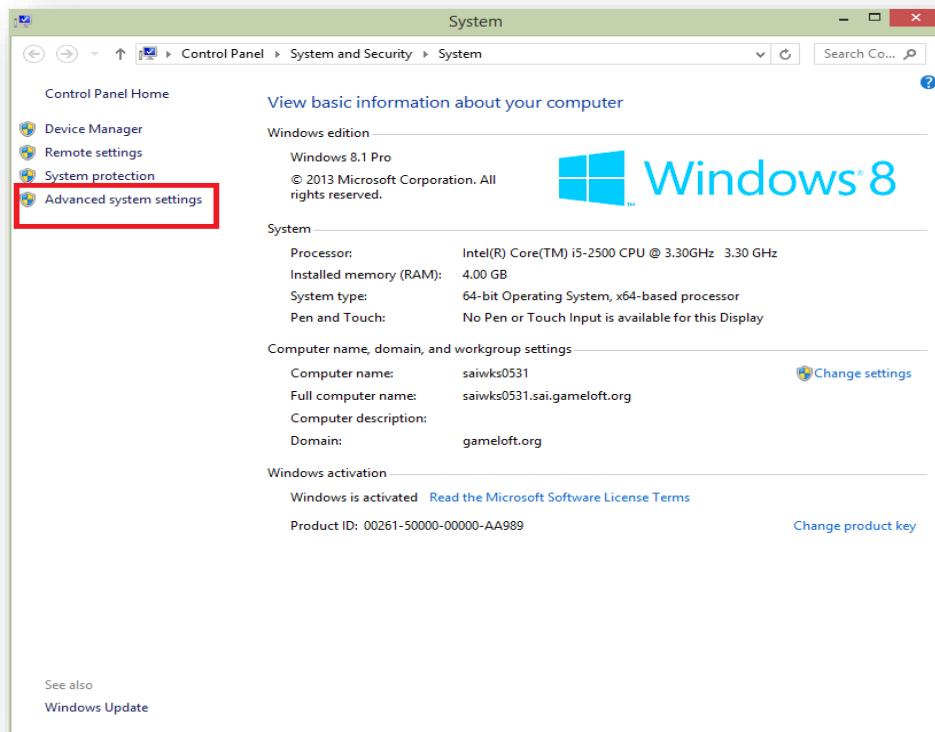
Hình 14: Thêm file qtjava.zip để build Cocos2d-x

- Các file còn lại chỉ cần giải nén để sử dụng (SDK, NDK, ANT, COCOS2D-X)

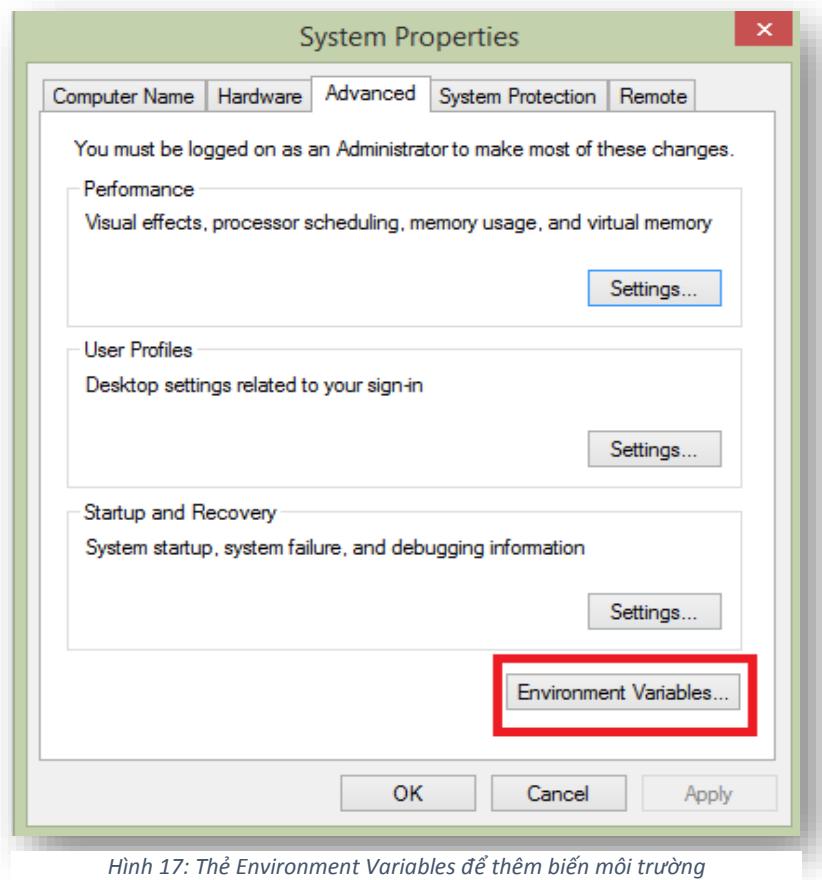
Name	Date modified	Type	Size
android-ndk-r10d	12/3/2014 10:50 AM	File folder	
apache-ant-1.9.4-bin	2/6/2015 9:27 AM	File folder	
cocos2d	2/5/2015 10:18 AM	File folder	
code	2/6/2015 11:18 AM	File folder	
sdk	2/6/2015 9:46 AM	File folder	
android-ndk-r10d-windows-x86_64.exe	2/6/2015 9:27 AM	Application	461,537 KB
apache-ant-1.9.4-bin.zip	2/6/2015 9:26 AM	WinRAR ZIP archive	8,115 KB
jdk-7u75-windows-x64.exe	2/6/2015 9:19 AM	Application	132,633 KB

Hình 15: Thư mục chuẩn bị đầy đủ cài đặt Cocos2d-x

- Tạo biến môi trường: Vào Advance system settings -> Environment



Hình 16: Thêm biến môi trường cho Windows



Hình 17: Thẻ Environment Variables để thêm biến môi trường

Variables

	Name	Path
User	ANDROID_SDK_ROOT	
	COCOS_CONSOLE_ROOT	
	PATH	
System	JAVA_HOME	
	ANT_HOME	
	ANT_ROOT	

	NDK_ROOT	
	PATH	
	CLASSPATH	

Bảng 8: Các biến môi trường cần thiết trong quá trình cài đặt

2.3. Cơ bản một số lệnh sử dụng trong “cmd” của Windows

- Tạo project :

```
cocos new [name project] -p [packet] -l cpp -d [path save project]
```

- Build project:

```
cocos compile -s [path save project] -p win32
```

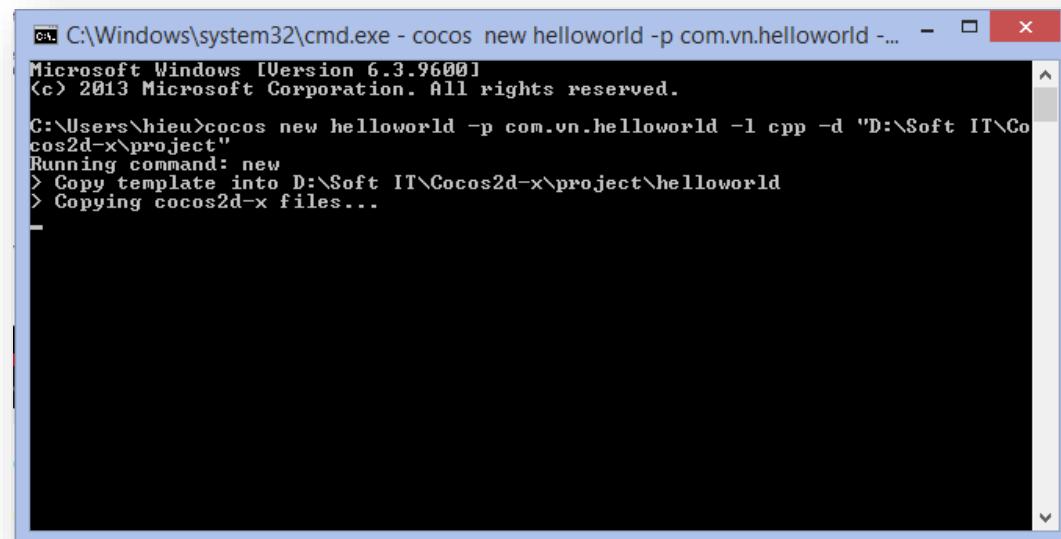
- Run project:

```
cocos run -s [path save project] -p win32
```

2.4. Game đầu tiên “Hello World”

- Tạo project “Hello world”

Vào Start -> run -> CMD. Gõ lệnh để tạo project “Hello world”



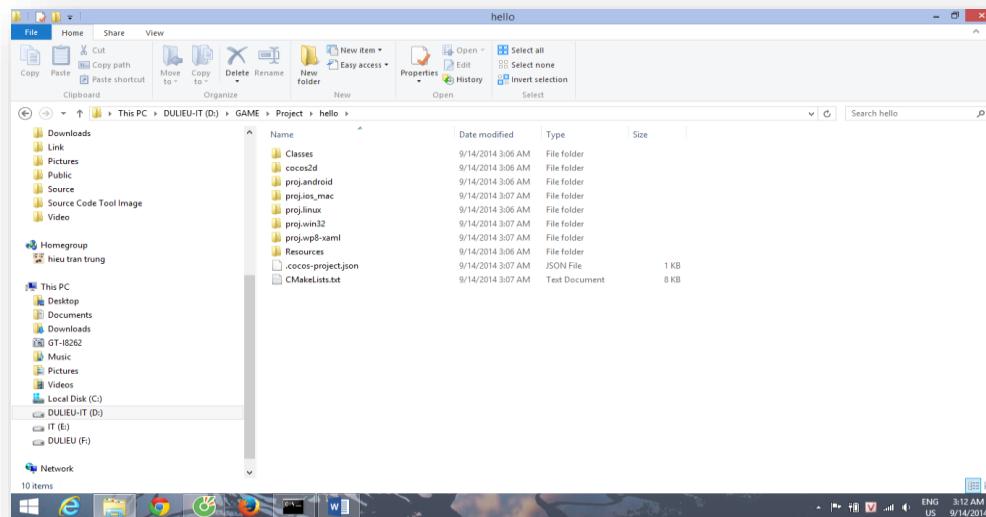
```
C:\Windows\system32\cmd.exe - cocos new helloworld -p com.vn.helloworld -l cpp -d "D:\Soft IT\Cocos2d-x\project"
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Users\hieu>cocos new helloworld -p com.vn.helloworld -l cpp -d "D:\Soft IT\Cocos2d-x\project"
Running command: new
> Copy template into D:\Soft IT\Cocos2d-x\project\helloworld
> Copying cocos2d-x files...
```

Hình 18: Tạo project game cocos2d-x với lệnh CMD

Cocos new hello -p com.studycoding.hello -l cpp -d D:\GAME\Project

Sau khi chạy xong dòng lệnh ta được thư mục như sau



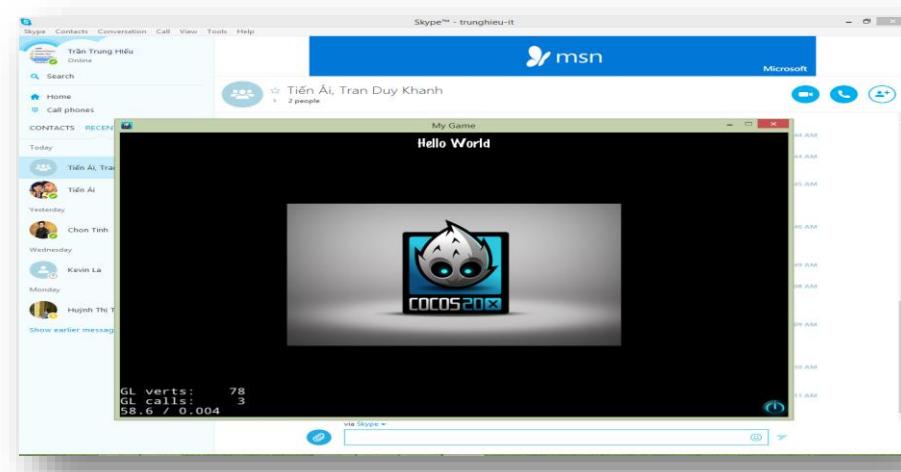
Hình 19: Thư mục project game coos2d-x sau khi tạo (template của cocos2d-x)

- Build project

Cocos compile -s *D:\GAME\Project\hello* -p win32

- Run project

Cocos run -s *D:\GAME\Project\hello* -p win32



Hình 20: Project Game Cocos2d-x "Hello World"

Chương 3: MODULES

3.1. Các thành phần chính trong Cocos2d-x

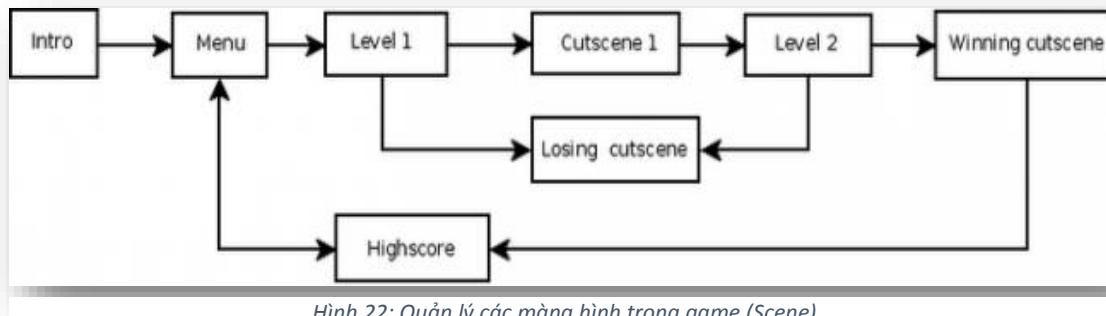
Cocos2d-x là một engine đa nền tảng dành cho cả di động và máy tính. Một Engine làm game luôn luôn hỗ trợ các thành phần như: Render, 2d/3d Graphics, Collision Detection, Physics Engine, Sound, Controller support, Animations, ... Engine hỗ trợ nhiều nền tảng vì vậy làm cho việc xây dựng 1 game không hề khó.

Các thành phần của Cocos2d-x: Scene, Sprite, Action, Transition, Menu, Sound, Effect, ...

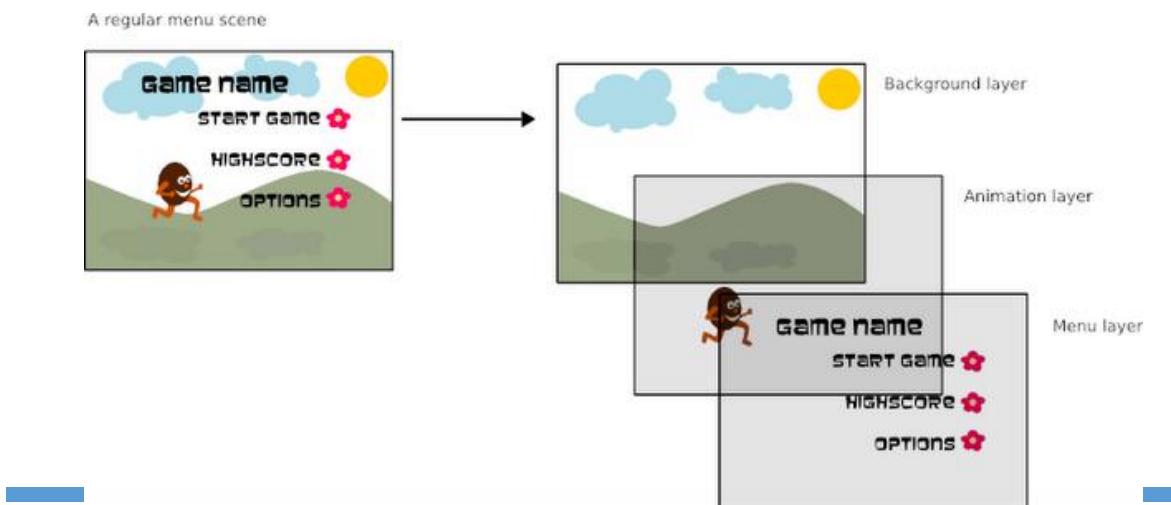
3.1.1. Director

Director là đối tượng cho phép chia sẻ các đối tượng giữa các cảnh (Scene) trong game (Sử dụng mẫu Singleton Design Pattern). Director quản lý các “Scene” trong game, cho biết “Scene” nào đang hoạt động và cho phép thay đổi “Scene” theo cơ chế Stack.

Ví dụ: Quản lý các Scene của một game:



Hình 22: Quản lý các màn hình trong game (Scene)



Hình 21: Bố trí các Layer trong màn hình game

3.1.2. Scene

Là một thành phần được hiển thị trên thiết bị, cho phép thêm các đối tượng đồ họa trên đó.

3.1.2. Layer

Layer là các tầng (lớp) của Scene nhằm phân định thứ tự hiển thị các đối tượng đồ họa trên màn hình đó.

3.1.3. Scene Graph

Scene Graph là cấu trúc dữ liệu được sắp xếp theo dạng cây. Dùng quản lý các đối tượng đồ họa trên giao diện màn hình. Hỗ trợ các phương thức duyệt các đối tượng đồ họa trên cây một cách dễ dàng.

3.1.4. Action

Mô tả một sự thay đổi của một thuộc tính của đối tượng. Cho phép tương tác và thay đổi các thuộc tính của đối tượng đồ họa.

3.1.5. Sprite

Là đối tượng đồ họa vẽ lên các Scene, Layer. Hỗ trợ các tính năng: di chuyển (move), hiệu ứng hoạt cảnh (Animation) và bắt va chạm.

Thông qua Sprite có thể mô phỏng nhân vật.

- Tạo Sprite

```
// tạo từ file hình trong resource
auto sprite = Sprite::create("football.jpg");
// xét tọa độ
sprite->setPosition(Point(visibleSize.width/2,visibleSize.height/2));
// thêm vào màn hình
this->addChild(sprite);
```

3.2. Giao diện trong Cocos2d-x (UI)

3.2.1. Label

3.2.2. Label Effects

3.2.3. Menu và MenuItem

- Tạo menu

Khai báo bên file .h:

```
void menuPlay(Ref* pSender);
```

Viết code trong hàm *init()* của file .cpp

```
auto automenuItem_1 = MenuItemFont::create("PLAY", CC_CALLBACK_1(HelloWorld::menuPlay, this));
auto closeItem = MenuItemImage::create( "CloseNormal.png",
                                         "CloseSelected.png",
                                         CC_CALLBACK_1(HelloWorld::menuCloseCallback, this));

    auto menu = Menu::create(automenuItem_1,closeItem, NULL);
    menu->setPosition(visibleSize.width / 2, visibleSize.height / 2);
    menu->alignItemsVertically();
    this->addChild(menu);

void HelloWorld::menuPlay(Ref* pSender) {
    // code xử lý khi nhấn vào
}
```

3.1.4. Button

3.1.5. CheckBox

3.1.6. LoadingBar

3.1.7. Slider

3.1.8. TextField

3.3. Di chuyển giữa các màn hình (Transition Scene)

3.3. Các loại lắng nghe sự kiện

3.4. Actions

3.4.1. Basic Actions

Action	Description
Scale Phóng to	<pre>/* duration: thời gian phóng to sprite s: số lần phóng to lên so với kích thước hiện tại */ auto action = ScaleBy::create(duration,s); sprite->runAction(action);</pre>

	<pre> /* duration: thời gian phóng to sprite s: số lần phóng to lên so với kích thước ban đầu */ auto action = ScaleTo::create(duration,s); sprite->runAction(action); </pre>
Rotate Xoay	<pre> /* duration: thời gian xoay sprite deltaAngel: góc quay so với sprite hiện tại */ auto action = RotateBy::create(duration, deltaAngel); sprite->runAction(action); </pre>
	<pre> /* duration: thời gian xoay sprite deltaAngel: góc quay so với sprite ban đầu */ auto action = RotateBy::create(duration, deltaAngel); sprite->runAction(action); </pre>
Visibility	<pre> /**/ hiện ra auto action = Show::create(); sprite->runAction(action); // ẩn đi auto action = Hide::create(); sprite->runAction(action); // ẩn <-> hiện auto action = ToggleVisibility::create(); sprite->runAction(action); </pre>
Opacity 0-255	<pre> // giảm độ mờ cho sprite khi khởi tạo sprite->setOpacity(100); // trong 3 giây sáng lên nhất có thể auto action = FadeIn::create(3); sprite->runAction(action); // trong 3 giây mờ dần -> mất đi auto action = FadeOut::create(3); sprite->runAction(action); // trong 3 giây ẩn hiện 5 lần auto action = Blink::create(3,5); sprite->runAction(action); </pre>
Move	<pre> // di chuyển 3 giây với đồ dài x:y -> 200;0 auto action = MoveBy::create(3, Point(200,0)); // bắt đầu </pre>

<i>Di chuyển</i>	<pre>sprite->runAction(action); // di chuyển 3 giây tới tọa độ 200;0 auto actionto = MoveTo::create(3, Point(200, 100)); // bắt đầu sprite->runAction(action);</pre>
Jump	<pre>// nhảy với độ dài x:y ->200;-100 với độ cao = 150 và số bước nhảy là 4 auto actionJum = JumpBy::create(5, Point(200, -100), 150, 4); sprite->runAction(actionJum); // nhảy tới tọa độ 200;-100 với độ cao = 150 và số bước nhảy là 4 auto actionJum = JumpTo::create(5, Point(200, -100), 150, 4); sprite->runAction(actionJum);</pre>
<i>Đường cong</i>	<pre>// tạo biến cấu hình cho đường cong di chuyển ccBezierConfig config; config.controlPoint_1 = Point(300, 0); config.controlPoint_2 = Point(0, 300); config.endPosition = Point(-300, 0); // xét thời gian và cấu hình auto action = BezierBy::create(5, config); sprite->runAction(action); // tạo biến cấu hình cho đường cong di chuyển ccBezierConfig config; config.controlPoint_1 = Point(300, 0); // tâm config.controlPoint_2 = Point(0, 300); config.endPosition = Point(300, 200); // điểm kết thúc // xét thời gian và cấu hình auto action = BezierTo::create(5, config); sprite->runAction(action);</pre>
<i>Place</i>	<pre>// di chuyển tức thời auto action = Place::create(Point(50, 50)); sprite->runAction(action); // trong 3 giây chuyển sang màu 100,0,0 auto action = TintBy::create(3,100,0,0); sprite->runAction(action);</pre>

Bảng 9: Các Action cơ bản trong Cocos2d-x

3.4.2. Sequences and how to run them

3.5. Animation

3.6. Touch

Trong Cocos2d-x hỗ trợ 2 loại tương tác “Touch events”

- Single – touch: Phương thức bắt sự kiện chạm vào 1 điểm trên màn hình tại 1 thời điểm.
- Multi – touch: Phương thức bắt sự kiện chạm vào nhiều điểm trên màn hình tại 1 thời điểm.

Quá trình cài đặt sự kiện “Touch” (single or multi – touch) như sau:

1. Khai báo hàm “Touch”
2. Khai báo lắng nghe sự kiện cho “Touch events”
3. Gán các hàm “Touch” tương ứng với các sự kiện (Begun – Moved – Ended)
4. Triển khai chức năng cho các hàm “Touch”
5. Xử lý code trong các hàm “Touch” tương ứng.

3.6.1. Single – touch events

- Khai báo function ở file .h:

```
bool onTouchBegan(cocos2d::Touch *touch, cocos2d::Event * event);

void onTouchMoved(cocos2d::Touch *touch, cocos2d::Event * event);

void onTouchEnded(cocos2d::Touch *touch, cocos2d::Event * event);

void onTouchCancelled(cocos2d::Touch *touch, cocos2d::Event * event);

bool isTouching;

float touchPosition;
```

- Thêm sự kiện cho Scene ở hàm *init()*:

```
//add sự kiện cho this
```

```

auto listener = EventListenerTouchOneByOne::create();

listener->setSwallowTouches(true);

listener->onTouchBegan = CC_CALLBACK_2(PlayerScene::onTouchBegan, this);

listener->onTouchMoved = CC_CALLBACK_2(PlayerScene::onTouchMoved, this);

listener->onTouchEnded = CC_CALLBACK_2(PlayerScene::onTouchEnded, this);

listener->onTouchCancelled = CC_CALLBACK_2(PlayerScene::onTouchCancelled, this);

this->getEventDispatcher() ->addEventListenerWithSceneGraphPriority(listener, this);

isTouching = false;

touchPosition = 0;

```

Viết code xử lý cho hàm đã khai báo

```

void PlayerScene::onTouchMoved(cocos2d::Touch *touch, cocos2d::Event * event) {
    // not used for this game
}

void PlayerScene::onTouchEnded(cocos2d::Touch *touch, cocos2d::Event * event) {
    isTouching = false;
}

void PlayerScene::onTouchCancelled(cocos2d::Touch *touch, cocos2d::Event * event) {
    onTouchEnded(touch, event);
}

```

3.6.2. Multi – touch events

3.7. Audio

3.7.1. Effect

- Thêm thư viện: `#include "SimpleAudioEngine.h"`
- Sử dụng

```

CocosDenshion::SimpleAudioEngine::getInstance()->preloadEffect("myEffect.mp3");
// thuộc tính cho phép lai lại khi kết thúc

```

```

idNhac = CocosDenshion::SimpleAudioEngine::getInstance()->playEffect("myEffect.mp3",true);

// xét âm lượng cho âm thanh
CocosDenshion::SimpleAudioEngine::getInstance()->setEffectsVolume(0.1);
//-----
// tắt tất cả âm thanh
CocosDenshion::SimpleAudioEngine::getInstance()->stopAllEffects();

// pause
CocosDenshion::SimpleAudioEngine::getInstance()->pauseEffect(idNhac);
//resume
CocosDenshion::SimpleAudioEngine::getInstance()->resumeEffect(idNhac);
// tắt nhạc theo ID của nó
CocosDenshion::SimpleAudioEngine::getInstance()->stopEffect(idNhac);

```

3.7.2. Music

- Thêm thư viện: `#include "SimpleAudioEngine.h"`
- Sử dụng

```

CocosDenshion::SimpleAudioEngine::getInstance()->preloadBackgroundMusic("myMusic.mp3");
CocosDenshion::SimpleAudioEngine::getInstance()->playBackgroundMusic("myMusic.mp3",true);
// pause
CocosDenshion::SimpleAudioEngine::getInstance()->pauseBackgroundMusic();
//resume
CocosDenshion::SimpleAudioEngine::getInstance()->resumeBackgroundMusic();

```

Chương 4: TÍNH NĂNG NÂNG CAO

4.1. Physics Engine

4.1. TileMap

Chương 5: CÔNG CỤ HỖ TRỢ

5.1. TexturePacker

5.1.1. *SpriteSheet là gì?*

SpriteSheet là anh bao gồm nhiều ảnh đơn ghép lại với nhau và có đi kèm 1 file lưu thông số của từng ảnh. Những ảnh đơn tạo thành từng nhóm ảnh của một Animation.

Để tạo 1 SpriteSheet ta cần tạo 2 file (dạng text để lưu thông số các ảnh đơn và 1 hình ảnh lớn gồm các ảnh đơn lại 1 tấm).

TexturePacker là công cụ giúp tạo SpriteSheet. File lưu thông số định dạng **.plist** và hình ảnh định dạng **.png**

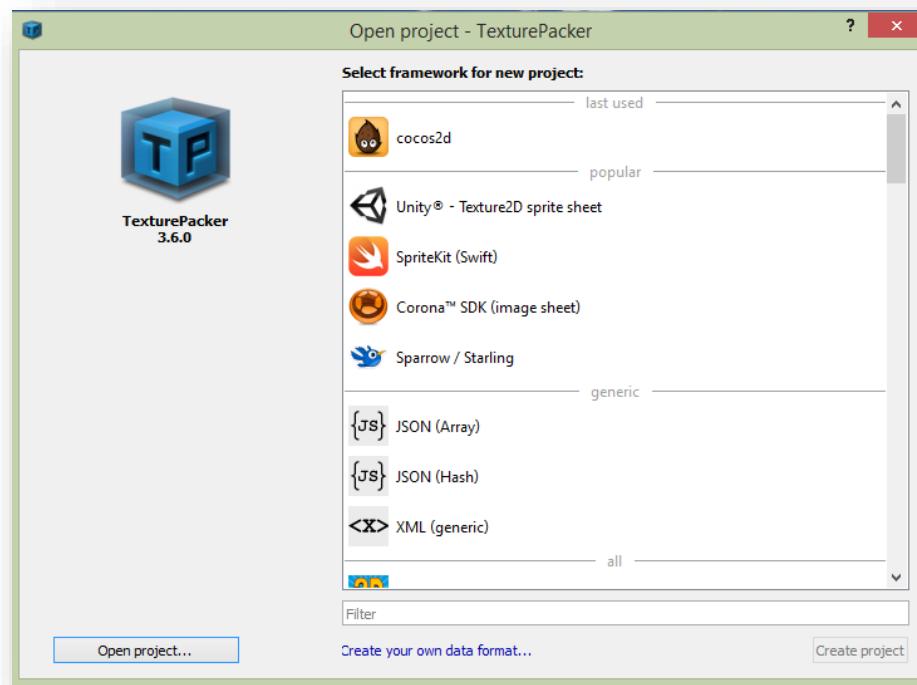
5.1.2. *Download and cài đặt phần mềm hỗ trợ*

Download tại trang: <https://www.codeandweb.com/texturepacker>

Đọc thêm: <https://www.codeandweb.com/texturepacker/tutorials#cocos2d-x>

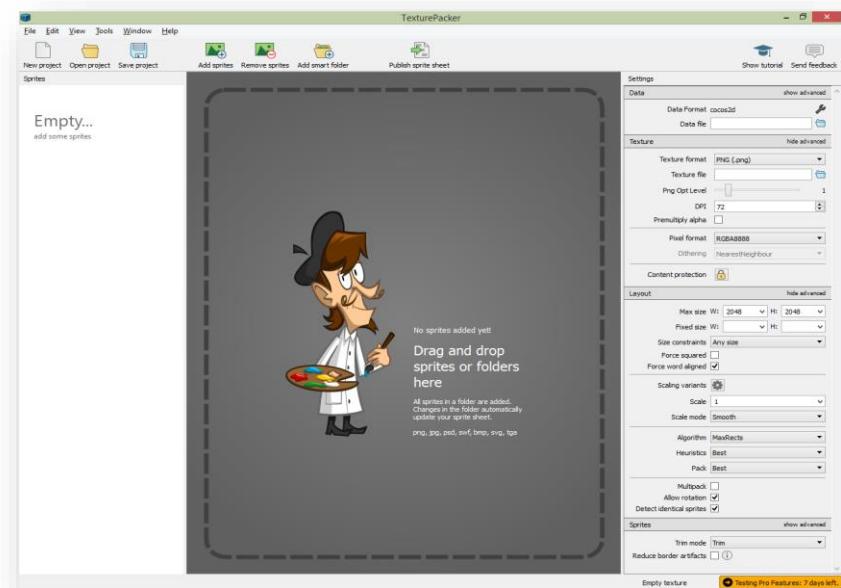
5.1.3. Tạo Spritesheet cho Cocos2d-x sử dụng TexturePacker

5.1.3.1. Tạo Sprite Sheet với TexturePacker



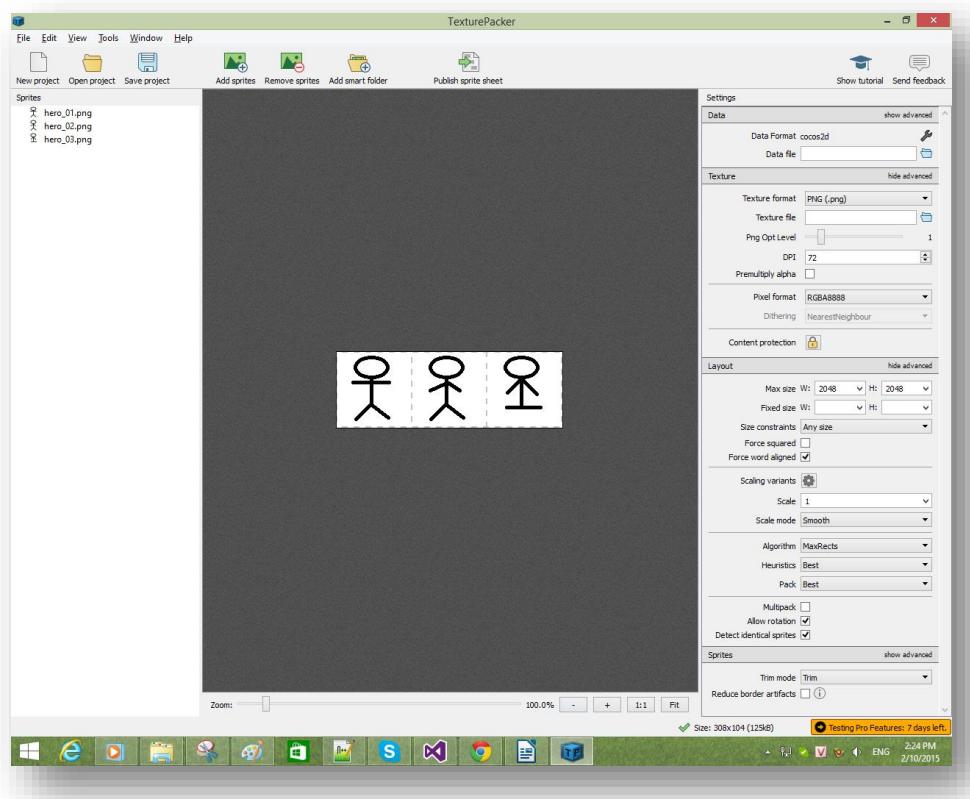
Hình 23: Tạo project với TexturePacker

Sau khi chọn cocos2d ta được giao diện chính của chương trình



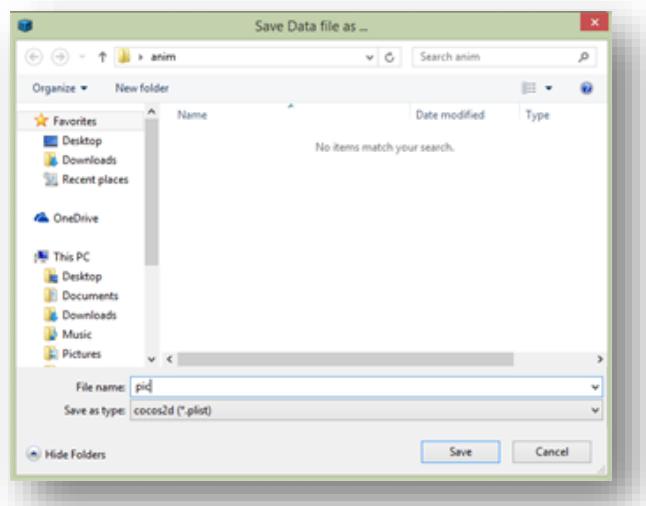
Hình 24: Giao diện Tool TexturePacker để tạo Sprite Sheet

Sau khi kéo thả các hình vào box “Sprites”, và tùy chỉnh các thông số, Chọn “Public sprite sheet”

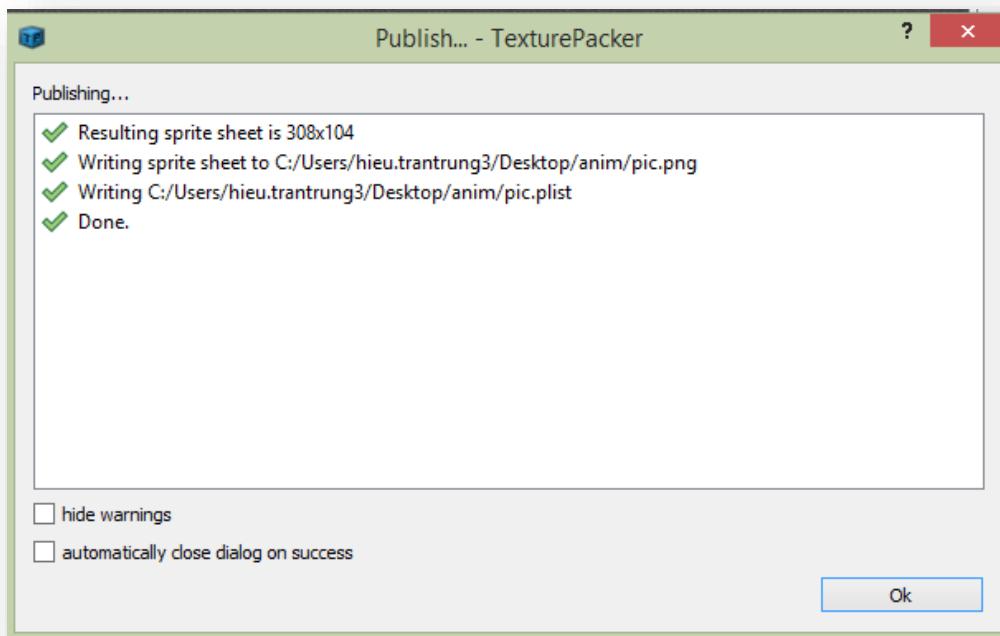


Hình 25: Public resource Sprite Sheet để tạo Animation cho nhân vật

Đã hoàn tất và đặt tên “Sprite sheet” chọn nơi lưu (lưu file .plist và .png)



Hình 26: Lưu Resource Sprite Sheet (.plish)



Hình 27: Hoàn thành Publish Sprite Sheet

Sau khi được 2 file .plist và .png ta copy vào thư mục resource của project cocos2d-

C > Local Disk (C:) > code > demo > Resources > images					
Name	Date	Type	Size	Tags	
pic.plist	2/10/2015 10:28 AM	PLIST File	3 KB		
pic.png	2/10/2015 10:28 AM	PNG image	1 KB		

Hình 28: Thêm sprite sheet vào resource của project game

5.1.3.2. Sử dụng Sprite Sheet cho Cocos2d-x

- Vào Appdelege.cpp load file .plist và .png để tạo Sprite

```
//search path

std::vector<std::string> searchPaths;

searchPaths.push_back("images");

 FileUtils::getInstance()->setSearchPaths(searchPaths);

//add sprite frame

SpriteFrameCache::getInstance()->addSpriteFramesWithFile("pic.plist", "pic.png");
```

- Tạo class Hero để thể hiện Sprite

Hero.h	Hero.cpp
<pre>#ifndef __HERO_SCENE_H__ #define __HERO_SCENE_H__ #include "cocos2d.h" USING_NS_CC; class Hero : public cocos2d::Sprite { public: Hero(); ~Hero(); private: cocos2d::RepeatForever *moving(); }; #endif // __HERO_SCENE_H__</pre>	<pre>#include "Hero.h" Hero::Hero() { initWithSpriteFrameName("a1.png"); this->runAction(moving()); } Hero::~Hero() { } cocos2d::RepeatForever* Hero::moving() { int numFrame = 3; cocos2d::Vector<cocos2d::SpriteFrame *> frames; cocos2d::SpriteFrameCache *frameCache = cocos2d::SpriteFrameCache::getInstance(); char file[100] = { 0 }; for (int i = 0; i < numFrame; i++) { sprintf(file, "a%d.png", i + 1); cocos2d::SpriteFrame *frame = frameCache- >getSpriteFrameByName(file); frames.pushBack(frame); } cocos2d::Animation *animation = cocos2d::Animation::createWithSpriteFrames(frames, 1); cocos2d::Animate *animate = cocos2d::Animate::create(animation); cocos2d::RepeatForever *repeat = cocos2d::RepeatForever::create(animate); return repeat; }</pre>

- Gọi “Sprite” ở Scene

```
Hero *hero = new Hero();

hero->setPosition(Point(visibleSize.width / 2, visibleSize.height / 2));

this->addChild(hero);
```

5.2. Tile Map Editor

5.2.1. *Tiled Map Editor* là gì?

Tile Map Editor là phần mềm mã nguồn mở hỗ trợ việc thiết kế bảng đồ cho game một cách tiện lợi và nhanh nhất. Tile Map Editor là công cụ giúp tạo map, config và render trong game.

Khi tạo 1 map sẽ được tài nguyên gồm:

- 1 file .tmx lưu thông tin cái đối tượng trong map.
 - 1 file .png lưu lại hình ảnh trong map
-  **Ưu điểm của Tile Map Editor:**
- Miễn phí.
 - Dễ sử dụng.
 - Hỗ trợ nhiều định dạng: XML, JSON, BIN.

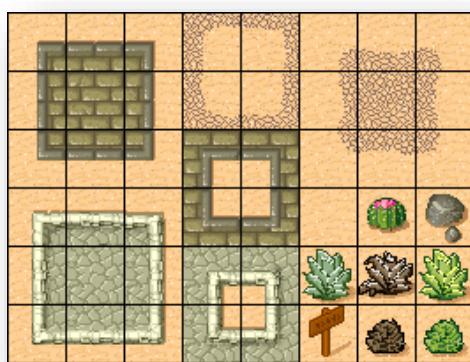
5.2.2. Download and cài đặt

Link Download: <http://www.mapeditor.org/download.html>

5.2.3. Tạo bản đồ cho game với Tiled Map Editor

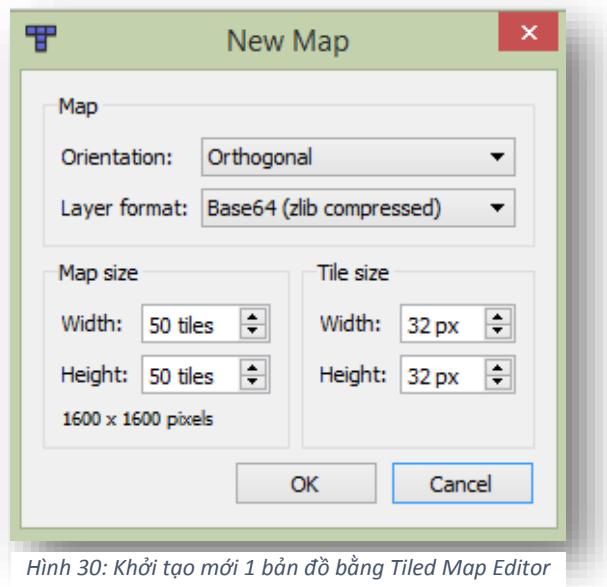
5.2.3.1. Tạo bản đồ với MapTiled with Tiled

Bước 1: Chuẩn bị file ảnh “Tile” để tạo map:



Hình 29: Thêm "Tiled" để tạo map

Bước 2: Mở chương trình Tiled vào File -> new



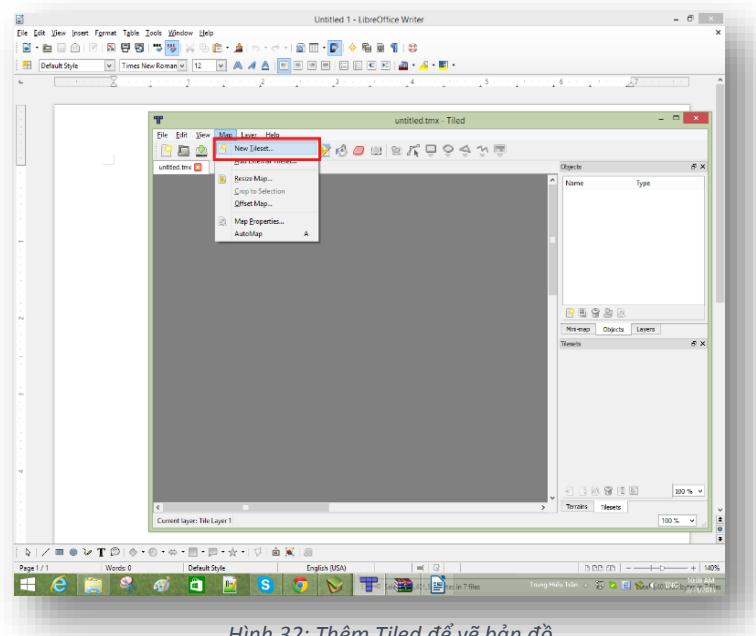
Hình 30: Khởi tạo mới 1 bản đồ bằng Tiled Map Editor

Chọn Width, Height cho Map. (*Lưu ý đơn vị được tính bằng tiles*)

Trong đó:

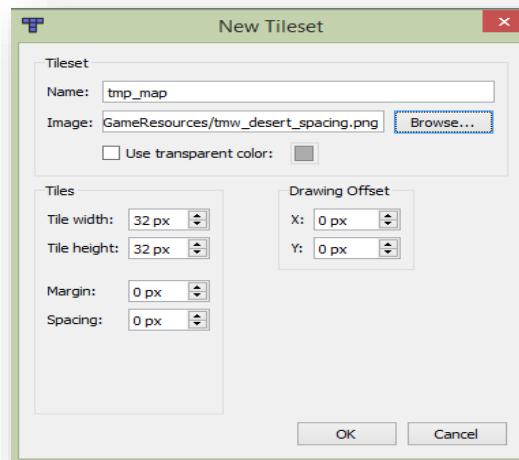
- Map – Orientation: chọn Orthogonal
- Layer – format: Base64(zlib compressed)
- Map size: Dài và rộng của Map (đơn vị tính bằng “Tile”)
- Tile size Width – Height: Chiều rộng và chiều dài của 1 đơn vị “Tile”

Bước 3: Tiếp theo, thêm “Tile” cho map bằng cách vào Map -> New Tileset.



Hình 32: Thêm Tiled để vẽ bản đồ

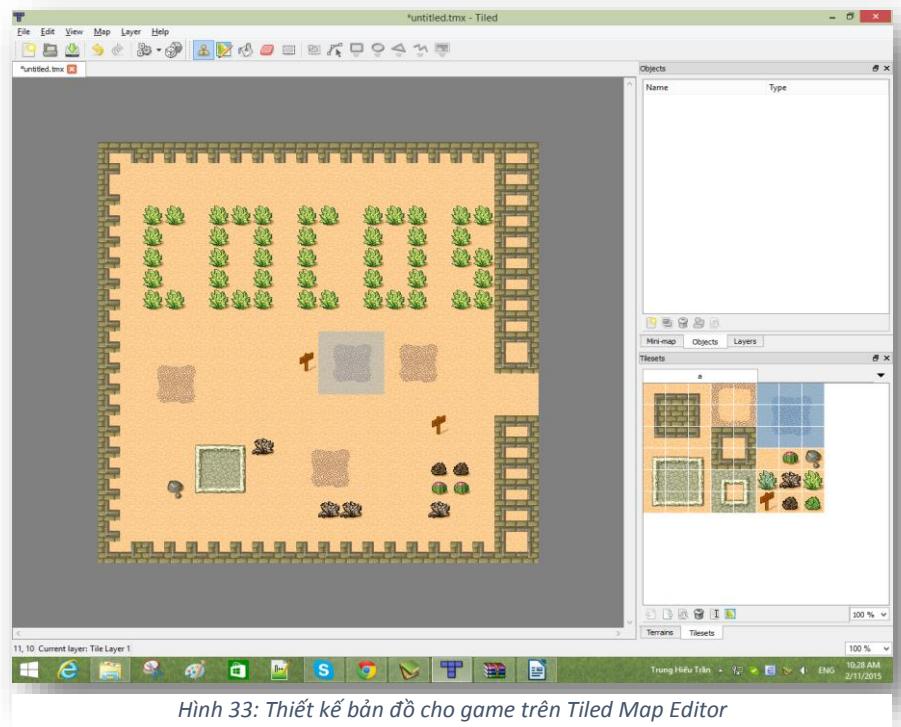
Chọn tới file hình: ảnh “Tile” lúc đầu chuẩn bị:



Hình 31: Chọn Tile để vẽ cho bản đồ

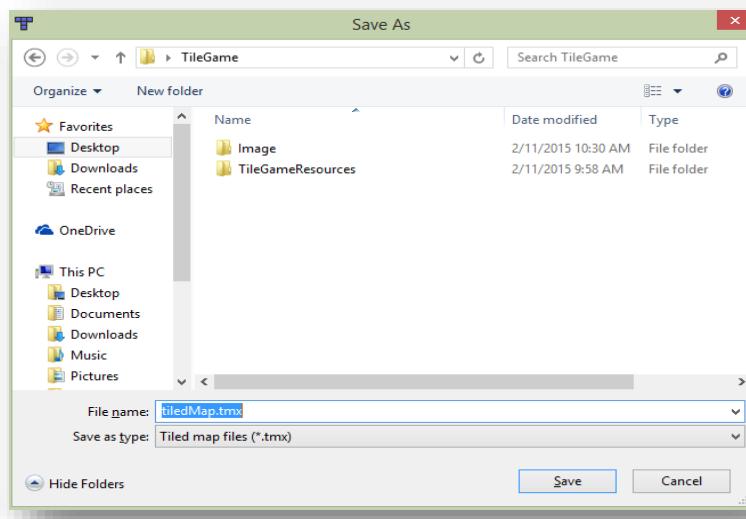
⚠ **Lưu ý:** cần lưu lại file hình này để làm resource khi load vào Cocos2d-x (.png)

Bước 4: Bắt đầu thiết kế bản đồ cho game bằng cách kéo thả từng “Tile” để vẽ vào Map



Hình 33: Thiết kế bản đồ cho game trên Tiled Map Editor

Bước 5: Sau khi thiết kế xong vào File -> Save để lưu map (định dạng .tmx)

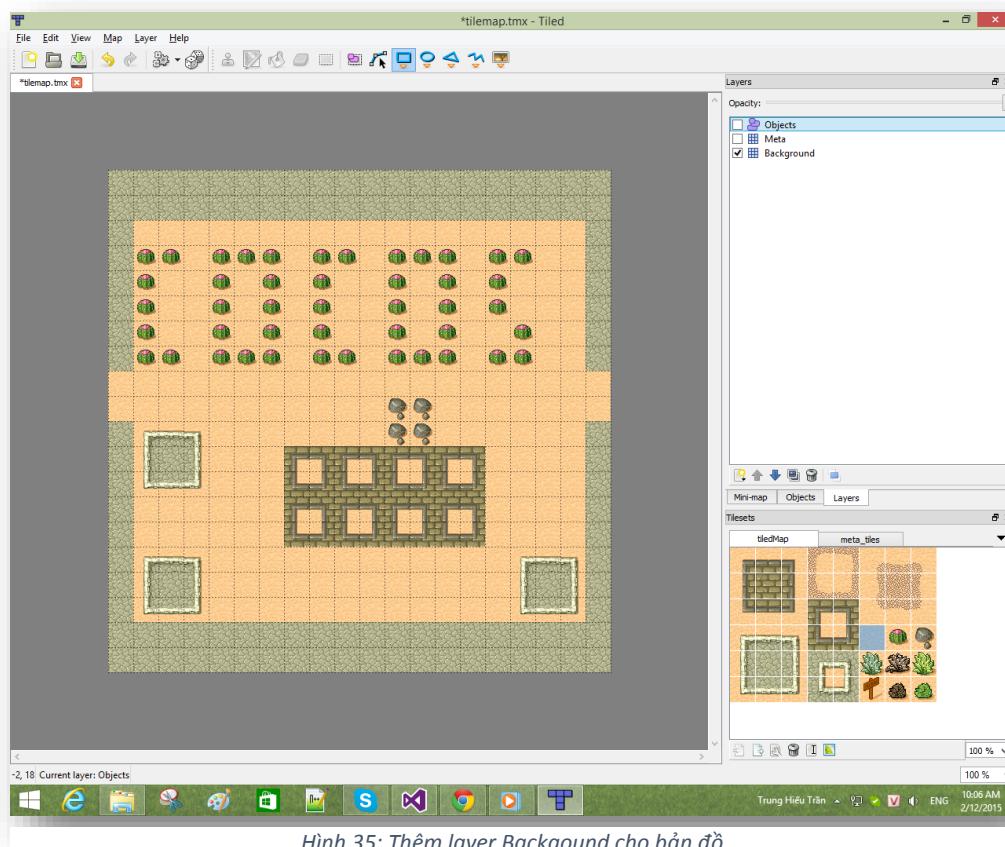


Hình 34: Lưu bản đồ dưới định dạng .tmx

 **Lưu ý:** cần lưu lại file hình này để làm resource khi load vào Cocos2d-x (.tmx)

5.2.3.2. Tải map lên game Cocos2d-x

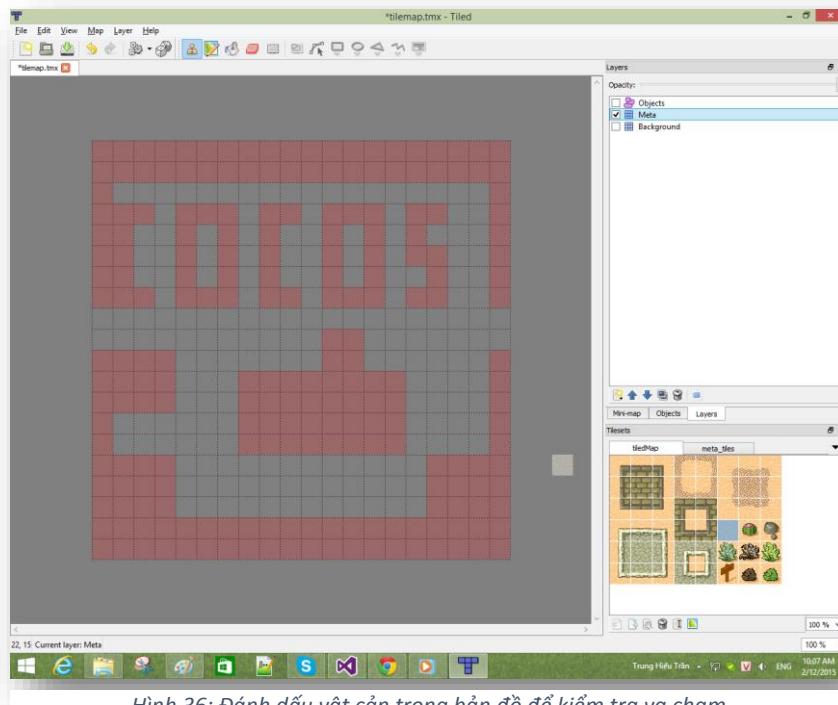
Bước 1: Tạo resource: Vào Tile để tạo map và vẽ thêm các đối tượng: Layer, object



Hình 35: Thêm layer Background cho bản đồ

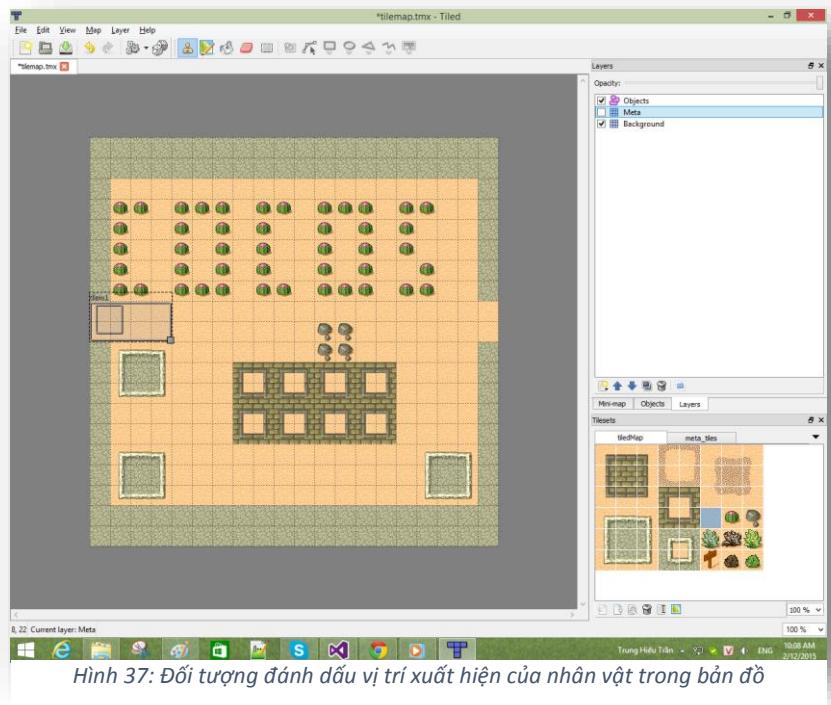
- Vẽ Background

- Vẽ Collision cho map để nhận biết va chạm với vật cản

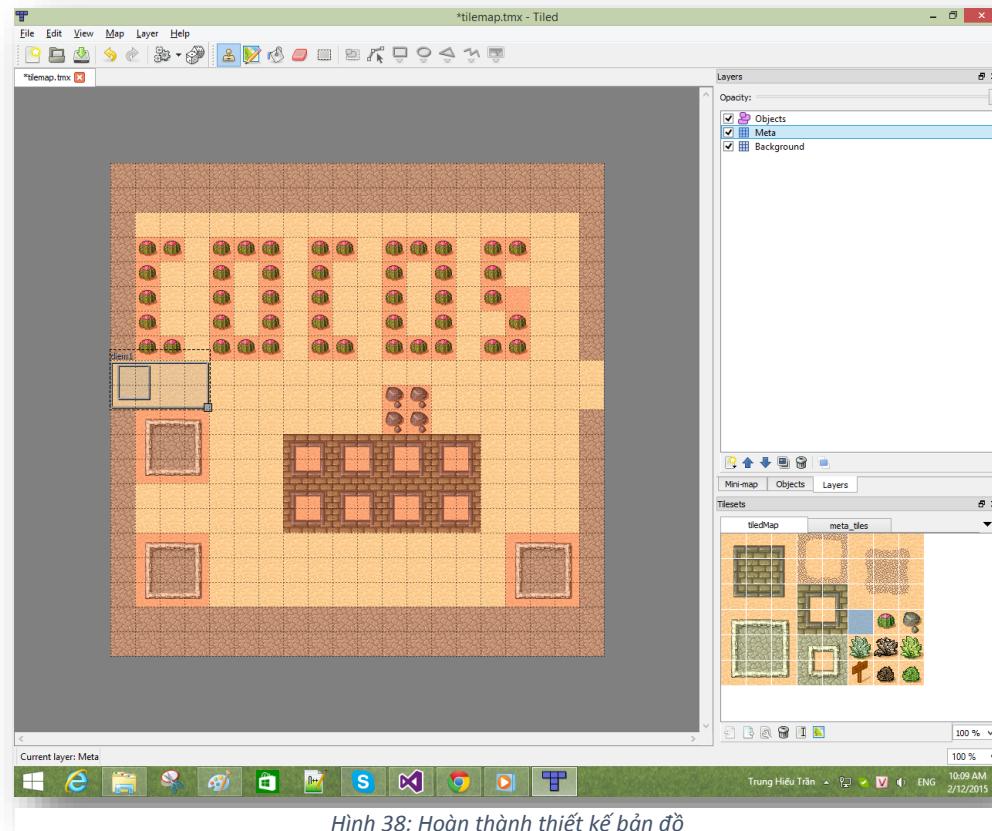


Hình 36: Đánh dấu vật cản trong bản đồ để kiểm tra va chạm

- Vẽ thêm đối tượng vào map để nhận biết nơi hiển thị cho Sprite



Hình 37: Đối tượng đánh dấu vị trí xuất hiện của nhân vật trong bản đồ



- Ta được map như sau: (lưu lấy file .tmx và file .png sử dụng làm Tiles để làm resource)

Bước 2: Vào hàm Init() để viết code load map vào Scene

```
Load Map
```

```
auto map = TMXTiledMap::create("map3/tilemap.tmx");
this->addChild(map, -1, 1);
```



Hình 39: Giao diện game khi tải bản đồ lên màn hình

Kết quả load được map vào game:

Bước 3: Vẽ đối tượng kiểu Sprite vào Map và kiểm tra va chạm trong map

- Vào file .h để khai báo 1 số hàm bắt event touch và khai báo tạo Layer, Sprite

Thêm vào public

```
// Event touch

bool _onTouchBegan(Touch *touch, Event *event);

void _onTouchMoved(Touch *touch, Event *event);

void _onTouchEnded(Touch *touch, Event *event);

void _onTouchCancelled(Touch *touch, Event *event);

// In the public section

void setViewPointCenter(Point position);
```

Thêm vào private

```
private:
```

```
cocos2d::TMXTiledMap *_tileMap;
cocos2d::TMLayer *_background;
void setPlayerPosition(Point position);
Sprite *_player;
TMLayer *_meta;
Point tileCoordForPosition(Point position);
```

- Vào file .cpp để viết code cho các hàm

Thêm vào trong hàm *init()*

```
// In Event Touch
auto _listener = EventListenerTouchOneByOne::create();
_listener->setSwallowTouches(true);
_listener->onTouchBegan = CC_CALLBACK_2(HelloWorld::_onTouchBegan, this);
_listener->onTouchEnded = CC_CALLBACK_2(HelloWorld::_onTouchEnded, this);
_listener->onTouchMoved = CC_CALLBACK_2(HelloWorld::_onTouchMoved, this);
_listener->onTouchCancelled = CC_CALLBACK_2(HelloWorld::_onTouchCancelled, this);
this->getEventDispatcher()->addEventListenWithSceneGraphPriority(_listener, this);

// In Load Map
_tileMap = TMXTiledMap::create("map3/tilemap.tmx");
_background = _tileMap->layerNamed("Background");
this->addChild(_tileMap, -1, 1);
_meta = _tileMap->layerNamed("Meta");
_meta->setVisible(false);
TMXObjectGroup *objectGroup = _tileMap->objectGroupNamed("Objects");
if (objectGroup == NULL) {
    return false;
}
```

```

auto spawnPoint = objectGroup->getObject("diem1");

int i = spawnPoint.size();

int x = spawnPoint["x"].asInt();

int y = spawnPoint["y"].asInt();

_player = Sprite::create("Player.png");

_player->setPosition(ccp(x, y));

this->addChild(_player);

this->setViewPointCenter(_player->getPosition());

```

- Viết code cho các hàm đã khai báo trong file .h

```

Point HelloWorld::tileCoordForPosition(Point position) {

    int x = position.x / _tileMap->getTileSize().width;

    int y = (_tileMap->getMapSize().height * _tileMap->getTileSize().height) - position.y / _tileMap-
>getTileSize().height;

    return ccp(x, y);
}

void HelloWorld::setViewPointCenter(Point position) {

    Size winSize = Director::sharedDirector()->getWinSize();

    int x = MAX(position.x, winSize.width / 2);

    int y = MAX(position.y, winSize.height / 2);

    x = MIN(x, (_tileMap->getMapSize().width * this->_tileMap->getTileSize().width) - winSize.width / 2);

    y = MIN(y, (_tileMap->getMapSize().height * _tileMap->getTileSize().height) - winSize.height / 2);

    Point actualPosition = ccp(x, y);

    Point centerOfView = ccp(winSize.width / 2, winSize.height / 2);

    Point viewPoint = ccpSub(centerOfView, actualPosition);

    this->setPosition(viewPoint);

}

```

```
bool HelloWorld::_onTouchBegan(Touch *touch, Event *event) {  
  
    auto action = RotateBy::create(1, 30);  
  
    _player->runAction(action);  
  
    return true;  
}  
  
void HelloWorld::_onTouchEnded(Touch *touch, Event *event) {  
  
    Point touchLocation = touch->getLocationInView();  
  
    touchLocation = Director::sharedDirector()->convertToGL(touchLocation);  
  
    touchLocation = this->convertToNodeSpace(touchLocation);  
  
    Point playerPos = _player->getPosition();  
  
    Point diff = ccpSub(touchLocation, playerPos);  
  
    if (abs(diff.x) > abs(diff.y)) {  
  
        if (diff.x > 0) {  
  
            playerPos.x += _tileMap->getTileSize().width;  
        }  
  
        else {  
  
            playerPos.x -= _tileMap->getTileSize().width;  
        }  
    }  
  
    else {  
  
        if (diff.y > 0) {  
  
            playerPos.y += _tileMap->getTileSize().height;  
        } else {  
  
            playerPos.y -= _tileMap->getTileSize().height;  
        }  
    }  
  
    if (playerPos.x <= (_tileMap->getMapSize().width * _tileMap->getTileSize().width) && playerPos.y <=  
(_tileMap->getMapSize().height * _tileMap->getTileSize().height)
```

```
&& playerPos.y >= 0 && playerPos.x >= 0) {  
  
    this->setPlayerPosition(playerPos);  
  
}  
  
this->setViewPointCenter(_player->getPosition());  
  
}  
  
void HelloWorld::_onTouchCancelled(Touch *touch, Event *event) {  
  
}  
  
void HelloWorld::_onTouchMoved(Touch *touch, Event *event){  
  
}
```

Hàm kiểm tra va chạm của đối tượng

```
void HelloWorld::setPlayerPosition(Point position) {  
  
    Point tileCoord = this->tileCoordForPosition(position);  
  
    int tileGid = _meta->tileGIDAt(tileCoord);  
  
    if (tileGid) {  
  
        return;  
  
    }  
  
    _player->setPosition(position);  
}
```

PHẦN 5:LÀM GAME “INTERNSHIP AH!” VỚI COCOS2D-X

Chương 1: GIỚI THIỆU “Game Internship AH!”

1.1. Giới thiệu game Internship AH!

“Internship AH!” là thể loại game chiến thuật. Được viết bởi “Team AH!” trong thời gian thực tập tại công ty Gameloft. Được phát hành phiên đầu tiên vào ngày 01 tháng 04 năm 2015.



Hình 40: Giới thiệu game Internship AH!

Game phát hành phiên bản dành cho máy tính hỗ trợ hệ điều hành Windows 7, Windows 8 trên cả 32 bit và 64 bit.

1.2. Công cụ hỗ trợ

Chức năng	Tool	Download	Platform
Tạo Flow Chart Diagram	https://www.draw.io	https://www.draw.io	Web
Tạo Map	Tiled Map Editor	http://www.mapeditor.org/	Windows/Mac OS/ Linux
Thiết kế	Photoshop CS6	www.adobe.com/	Windows/Mac OS

Bảng 10: Công cụ hỗ trợ dùng trong làm game “Internship AH”

1.3. Thông tin nhóm thực hiện

Người hướng dẫn: Trần Duy Khánh

Thành viên nhóm thực hiện:

- Trần Trung Hiếu
- Nguyễn Tiến Ái

1.4. Task

Ngày thực hiện	Chức năng	Người làm	Chú thích
2015 - 03 - 02	Tạo Khung sườn – bộ cục cho Game	Hiếu	
	Tạo Flow Chart	Hiếu	
	Tạo Map	Ái	
	Tạo và load Sprite	Ái	
2015 - 03 - 04	Thiết kế Sprite Sheet	Ái	
	Thiết kế Map, vẽ đối tượng (trụ đánh quái vật và vẽ đường đi cho quái vật)	Hiếu	
2015 - 03 - 05	Code xây dựng trụ (Xây dựng, kiểm tra duy nhất)	Hiếu	
	Code hướng đi và di chuyển quái theo đường đi trên map	Ái	
2015 - 03 - 06	Kiểm tra va chạm giữa quái và trụ khi quái di chuyển trên đường đi	Hiếu	
	Thêm các thuộc tính, thêm thanh HP cho quái, giảm máu khi bị trụ bắn	Ái	
2015 - 03 - 09	Xây dựng hướng đối tượng cho quái vật	Ái	

	Phân tích hướng đối tượng trụ	Hiếu	
2015 - 03 - 10	Resourch manager (đọc từ file txt các thông tin về quái cũng như thông tin của trụ và bản đồ)	Ái	
	Thêm vật lý vào game	Hiếu	
2015 - 03 - 11	Resource manager (đọc từ file tx các thông tin về quái cũng nhu thông tin của trụ và bản đồ) (TT)	Ái	
	Xây dựng mũi tên và va chạm	Hiếu	
2015 - 03 - 12	Xây dựng các đối tượng truyền dữ liệu, cấu hình game.		
	Xây dựng các hàm chức năng dùng chung trong game		
2015 - 03 - 13	Định hướng cho mũi tên và xử lý di chuyển cho mũi tên khi tấn công quái	Hiếu	
	Xây dựng class ResourceManager để quản lý tài nguyên của game	Ái	
2015 - 03 - 16 đến 2015 - 03 - 18	Thiết kế hình ảnh scene start, upgrade, skill, scene pause, scene game over.	Ái	
	Thiết kế hình ảnh cho scene choose map, đạn, hiệu ứng	Hiếu	
2015 - 03 - 19 đến 2015 - 03 - 24	Hoàn thiện game play	Hiếu	

		Ái	
2015 – 03 – 25	Xây dựng thông tin người chơi cho game (Lưu Map đã hoàn thành, tiền của người chơi)	Hiếu	
	Đạn của trụ		
2015 – 03 – 26	Resource manager cho trụ	Hiếu	
	Xây dựng hệ thống Upgrade cho Tower		
2015 – 03 – 27	Ảnh hưởng của kill đến quái	Hiếu	
	Tạo Skill cho Game (Tạo Sprite, Load vào Game, Bắt tương tác Touch kéo thả để sử dụng Skill)		
2015 – 03 – 30	Xây dựng Scene Pause, Scene Game Win, Game Over	Hiếu	
	Tạo Skill cho Game (Mất máu cho Enemi)		Ái
2015 – 03 – 31 đến 2015 – 04 – 02	Tổng hợp các tính năng của Game và viết báo cáo.	Hiếu	
		Ái	

Bảng 11: Phân công công việc làm game Internship AH!

Chương 2: BẮT ĐẦU THỰC HIỆN

 **Lưu ý:** Hướng dẫn cài đặt trên môi trường Windows (Win 8.1 - 64bit) Và sử dụng IDE Visual Studio Community 2013.

2.1. Lên ý tưởng và xây dựng GamePlay

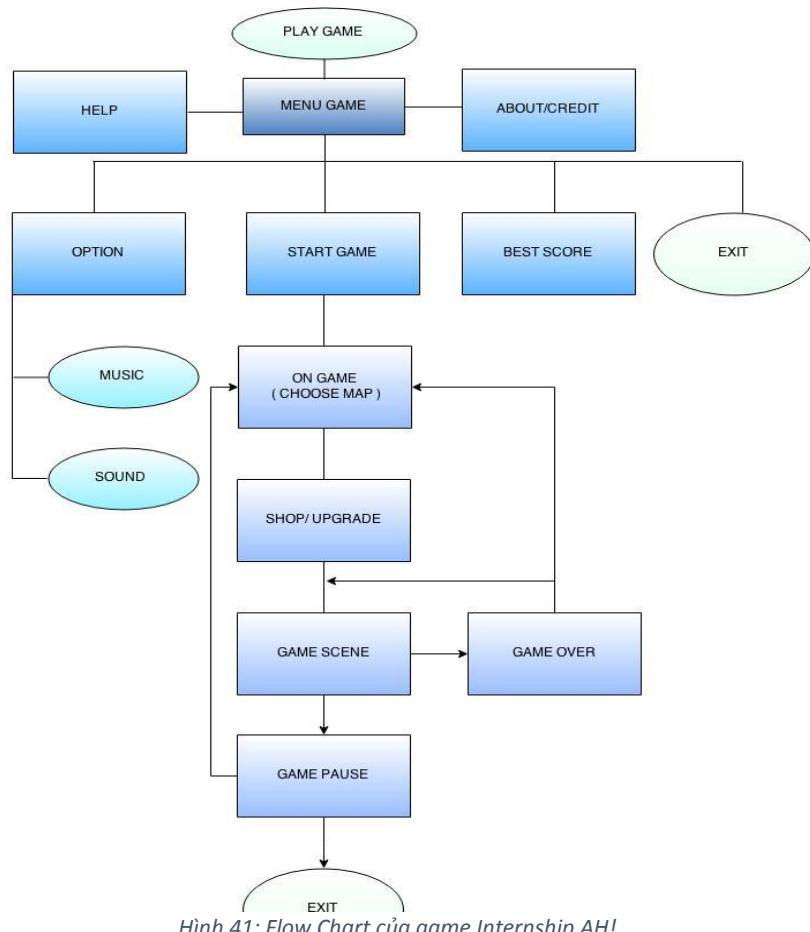
Game “Internship AH!” thuộc thể loại “Game Defense”. Game phát sinh tự động các loại quái di chuyển trên đường đi trong bản đồ, người chơi tạo mỗi bản đồ sẽ được 1 số vàng cụ thể để xây dựng các trụ tại các vị trí nhất định được cho phép trong bản đồ.

 Các chơi:

- Người chơi sẽ xây dựng các Tower (trụ đánh quái) để đánh các Enemi (quái game tự động sinh ra và di chuyển trên đường đi trong map).
- Ngoài ra người chơi được hỗ trợ 3 Skill để tương tác với quái.

Tại mỗi bản đồ sẽ có số lượng vòng nhất định, mỗi vòng sẽ có nhiều loại quái khác nhau thuộc các loại khác nhau.

2.2. Flow Chart



Hình 41: Flow Chart của game Internship AH!

2.3. Phân tích hướng đối tượng

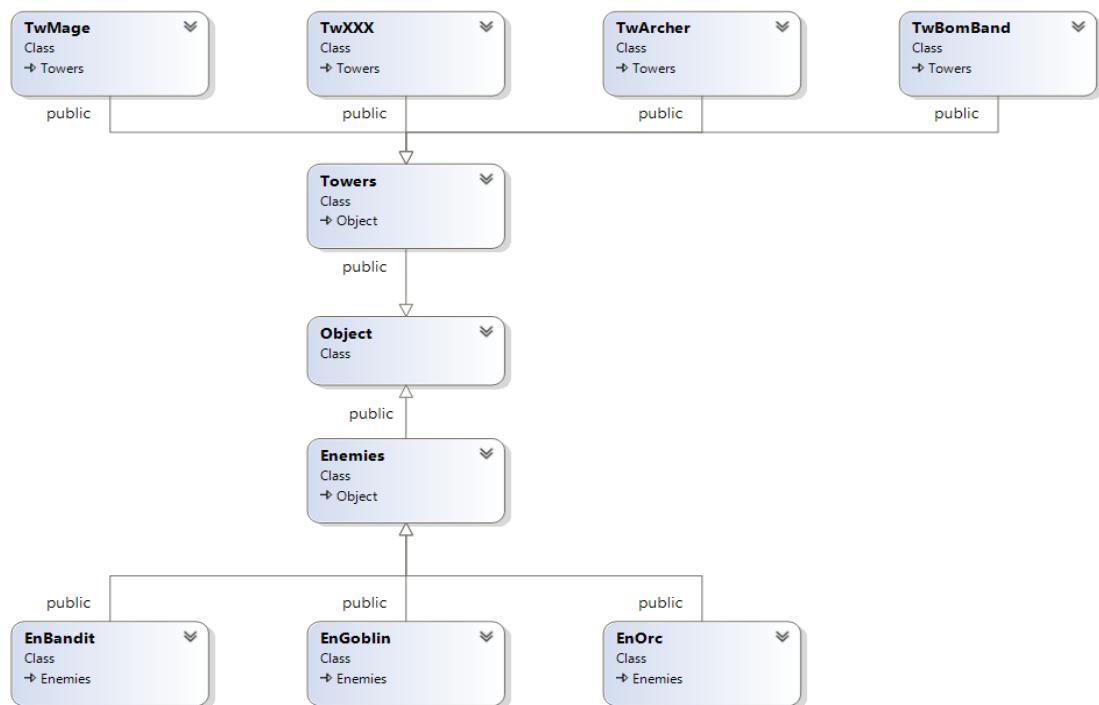
Game có 2 đối tượng chính là Tower và Enemi, Enemi có nhiều loại khác nhau, mỗi loại Enemi tương ứng có loại có đối tượng “boss”. Tower có nhiều loại khác nhau, mỗi loại Tower có hệ thống nâng cấp khác nhau. Khi Tower được nâng cấp, mỗi loại Tower sẽ có sự thay đổi về các thông số khác nhau.

Để xây dựng các đối tượng sử dụng trong game ta xây dựng đối tượng game Object gồm có các thuộc tính cơ bản của 1 đối tượng trong game.

Game gồm có hệ thống các loại quái khác nhau (tùy chỉnh thuộc tính) và hệ thống các loại trụ khác nhau (tùy chỉnh thuộc tính), các trụ có hệ thống nâng cấp khác nhau (tùy chỉnh thông số của thuộc tính)

Ngoài ra cần xây dựng các lớp chức năng để sử dụng trong game:

- Lớp lưu thông tin người chơi.
- Các lớp mảng hình game tương ứng trên mô hình Flow Chart.
- Các lớp Sprite để tùy chỉnh Sprite cho các nhân vật trong game.
- Lớp cấu hình các thông số cơ bản cho game (các đối tượng để đóng gói 1 số thuộc tính cần thiết để lưu trữ dữ liệu cho các đối tượng).
- Lớp quản lý tất cả tài nguyên cho game.

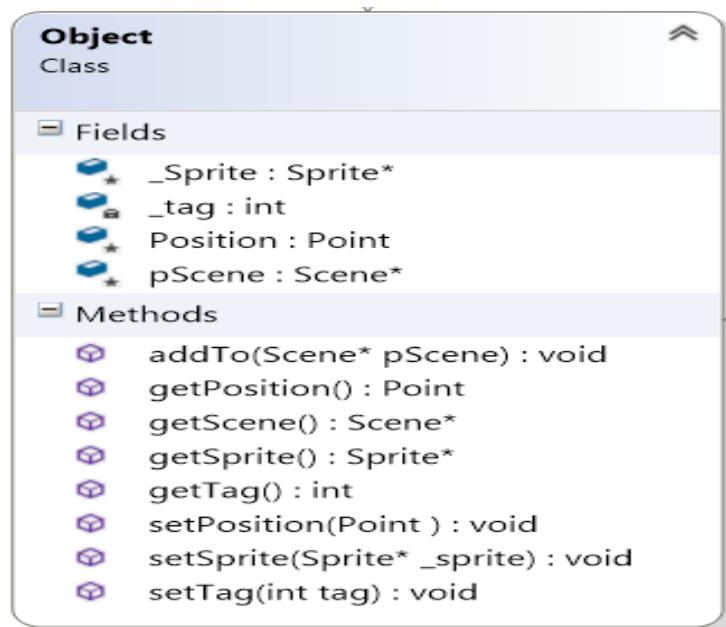


Chi tiết các đối tượng trong game (Nhân vật quái “Enemi” và trụ “Tower”,

Hình 42: Phân tích hướng đối tượng trong game Internship AH!

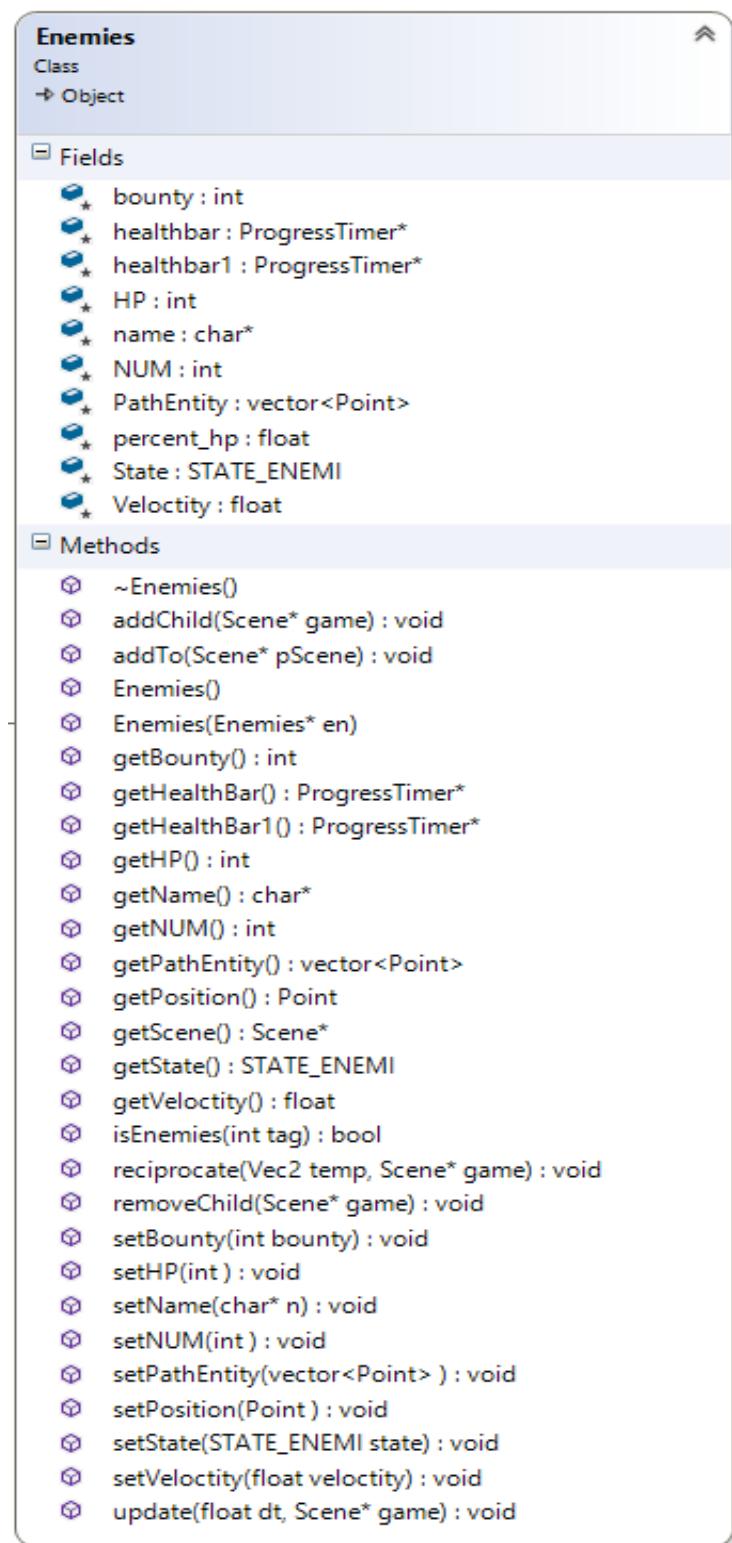
đối tượng game object, Lớp quản lý tài nguyên, lớp lưu thông tin người chơi, . . .):

- Đối tượng Game Object



Hình 43: Đối tượng Game Object

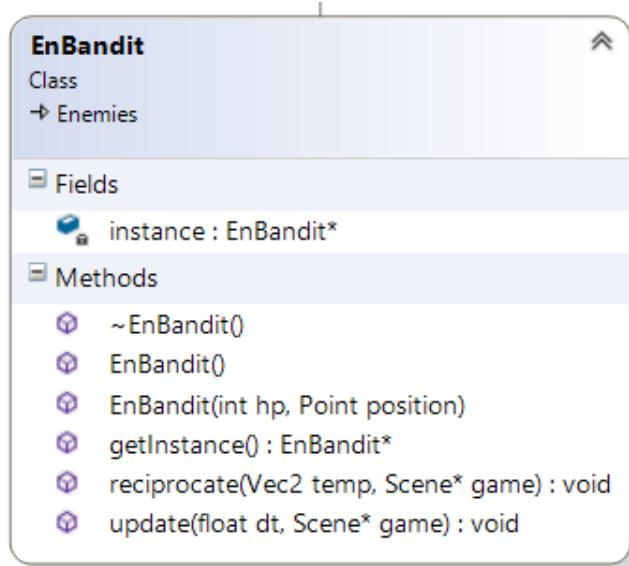
- Enemi



-

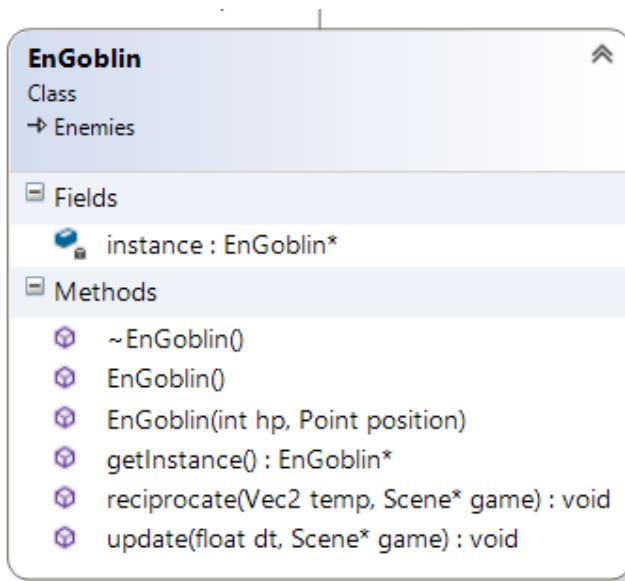
Các loại Enemi có trong game

- EnBandit



Hình 45: Đối tượng quái Bandit trong game Internship AH!

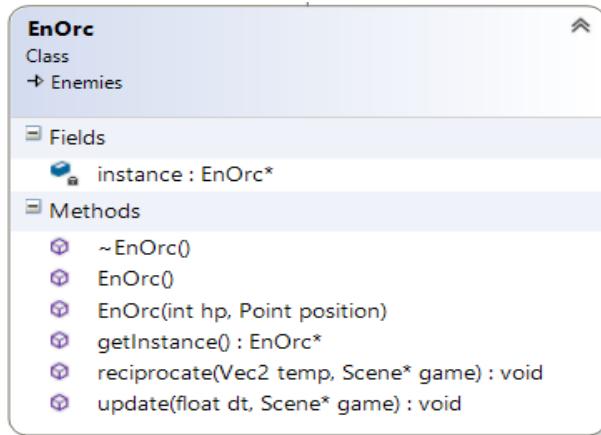
- EnGoblin



Hình 46: Đối tượng quái Goblin trong game Internship AH!

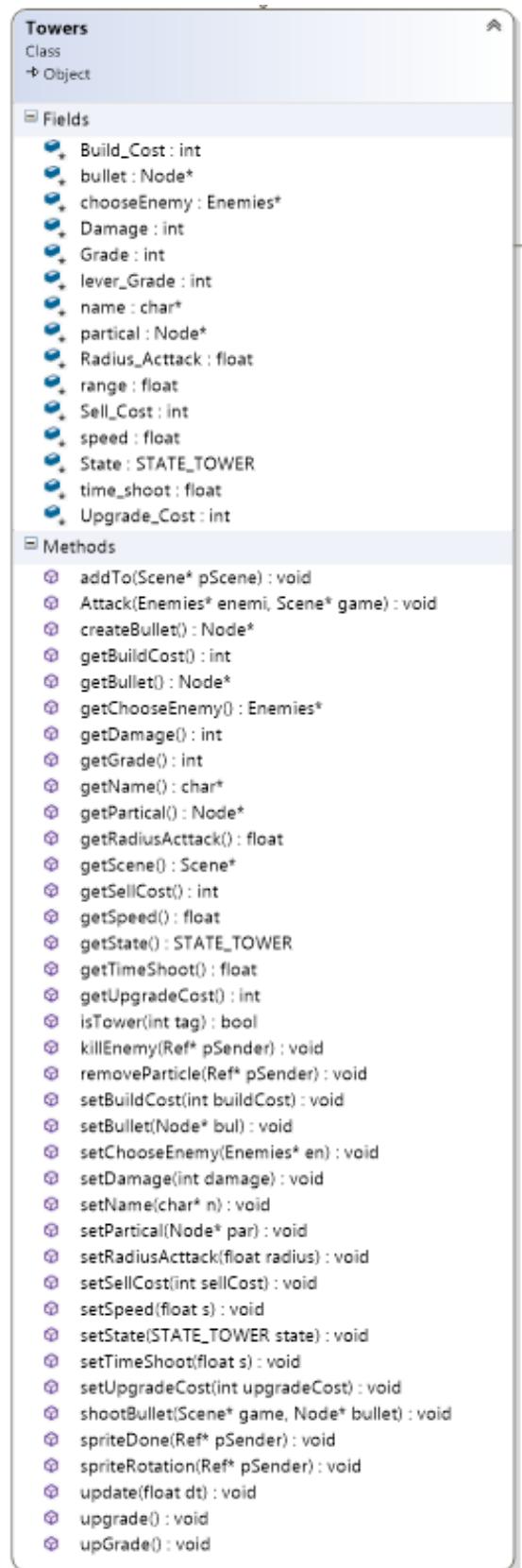
Hình 44: Đối tượng quái trong game Internship AH! (Enemi)

- EnOrc



Hình 47: Đối tượng quái Orc trong game Internship AH!

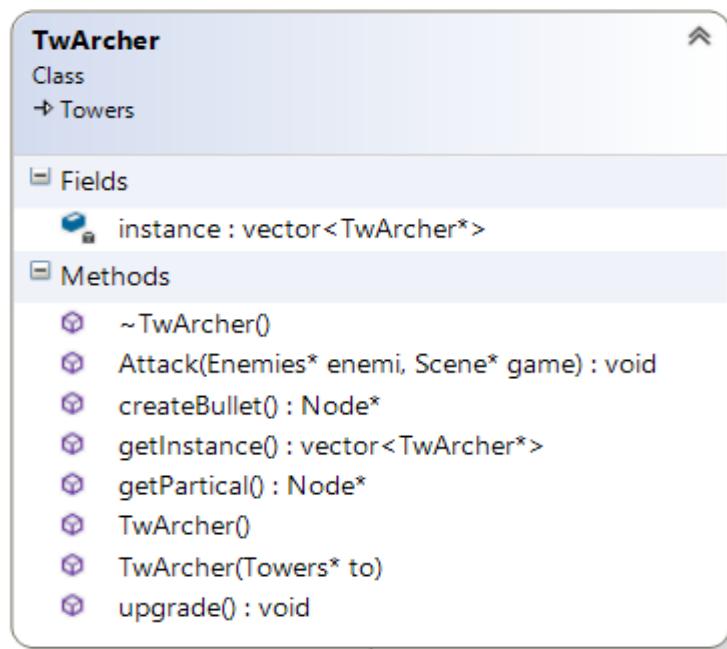
- Tower



Hình 48: Đối tượng trụ trong game Internship AH! (Tower)

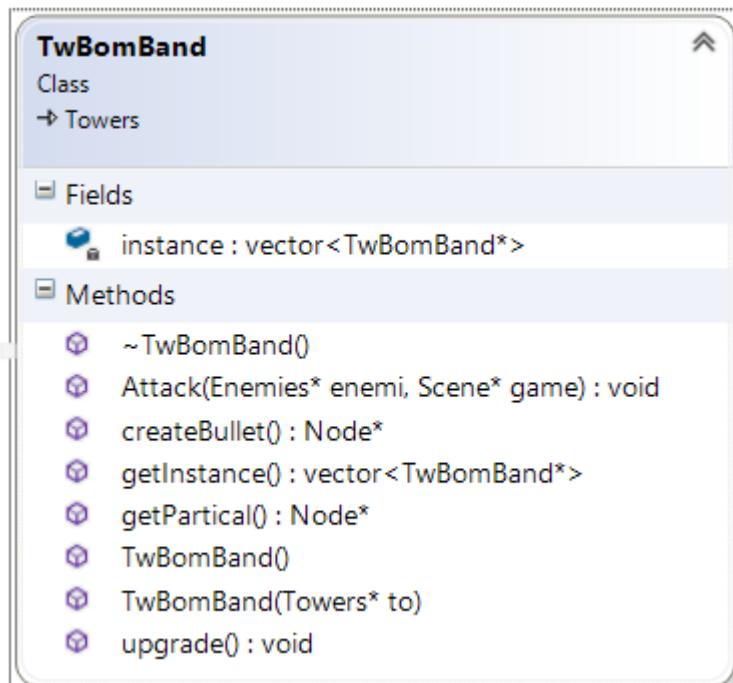
Các loại Tower trong game

- TwArcher



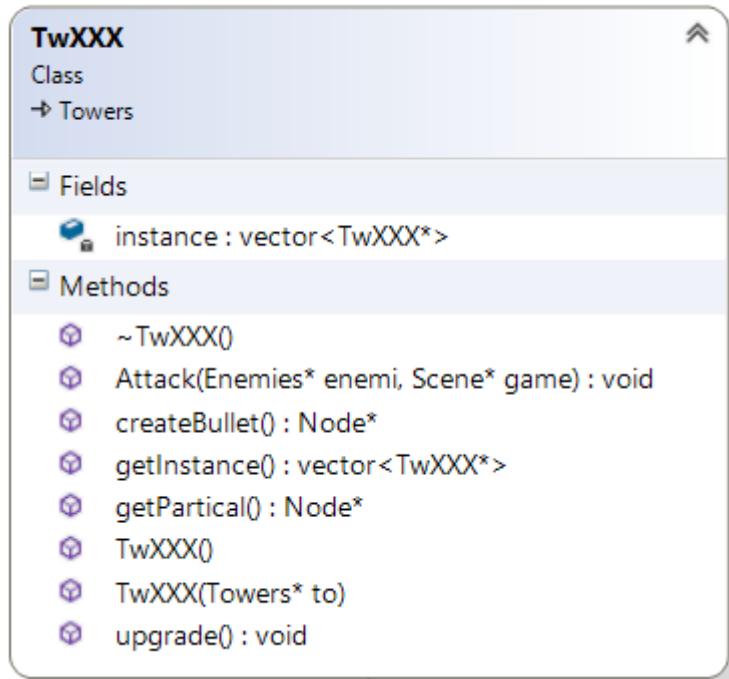
Hình 49: Đối tượng trụ Archer trong game Internship AH!

- TwBomBand



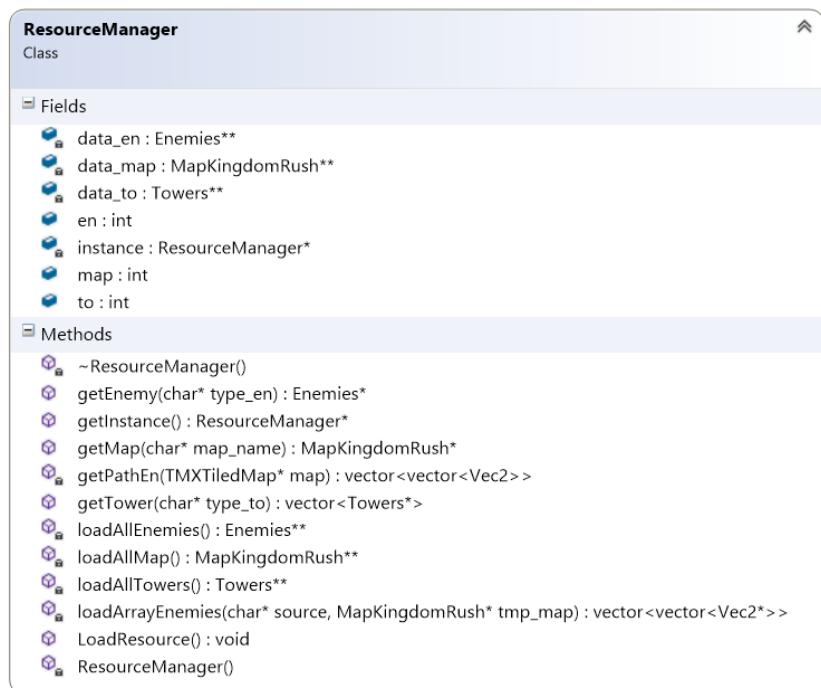
Hình 50: Đối tượng trụ BomBand trong game Internship AH!

- TwXXX



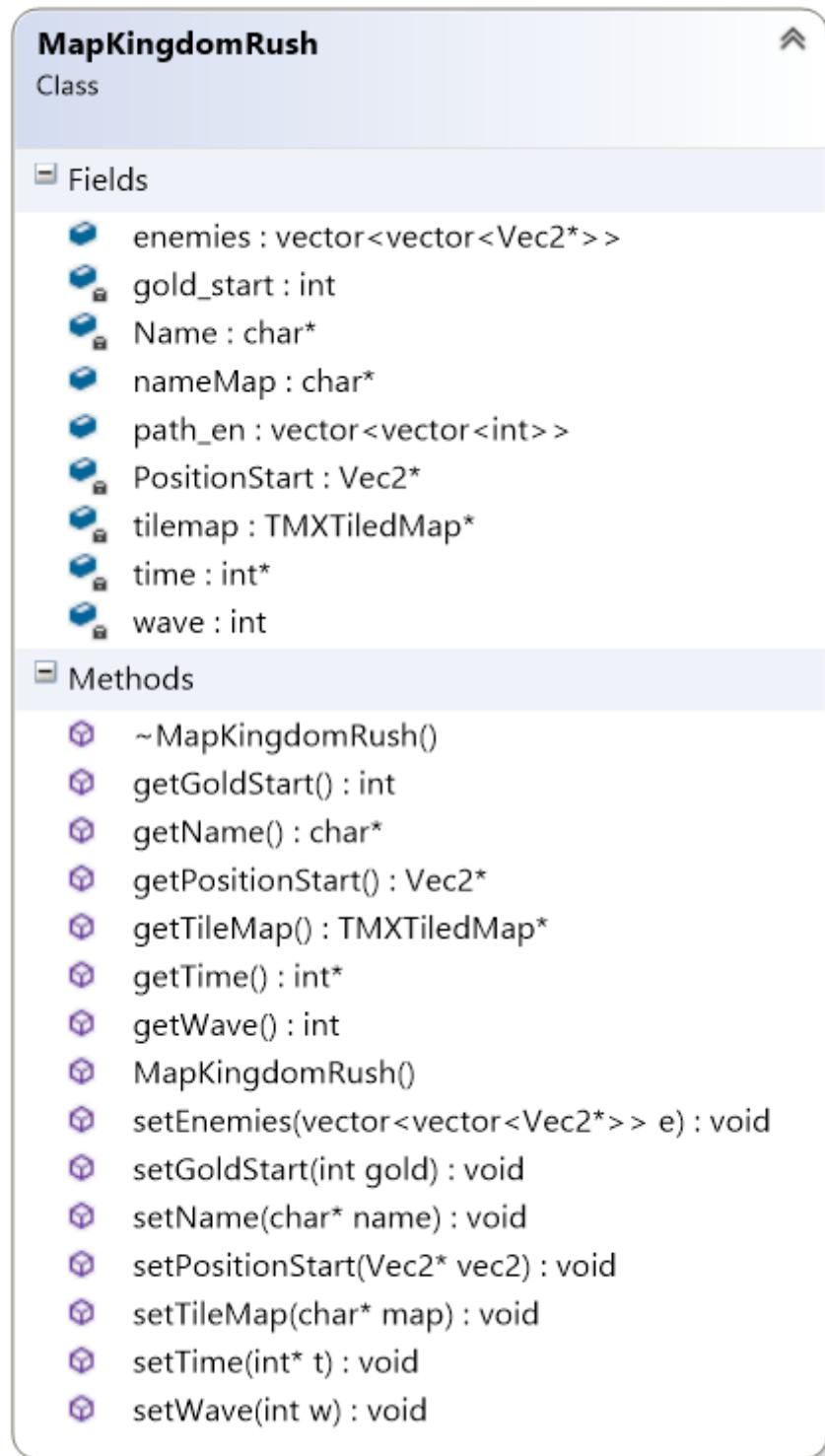
Hình 51: Đối tượng trụ XXX trong game Internship AH!

- Lớp quản lý tất cả tài nguyên sử dụng trong game: ResourceManager



Hình 52: Đối tượng quản lý tài nguyên trong game Internship AH!

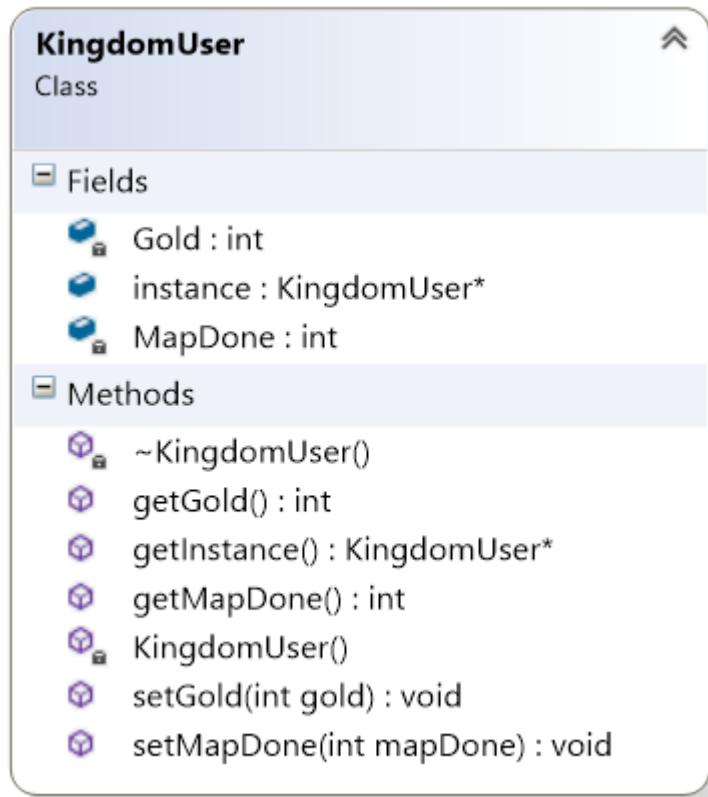
- Lớp lưu trữ dữ liệu các bản đồ trong game và kịch bản của mỗi bản đồ (Máy vòng, mỗi vòng có những loại quái nào, mỗi loại quái có với số lượng nhiêu,



Hình 53: Đối tượng quản lý các bản đồ trong game Internship AH!

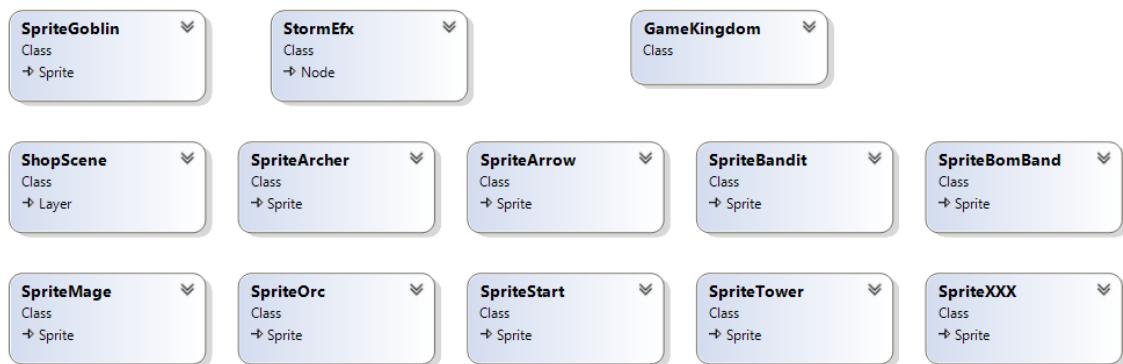
...)

- Lớp lưu trữ thông tin người chơi trong game



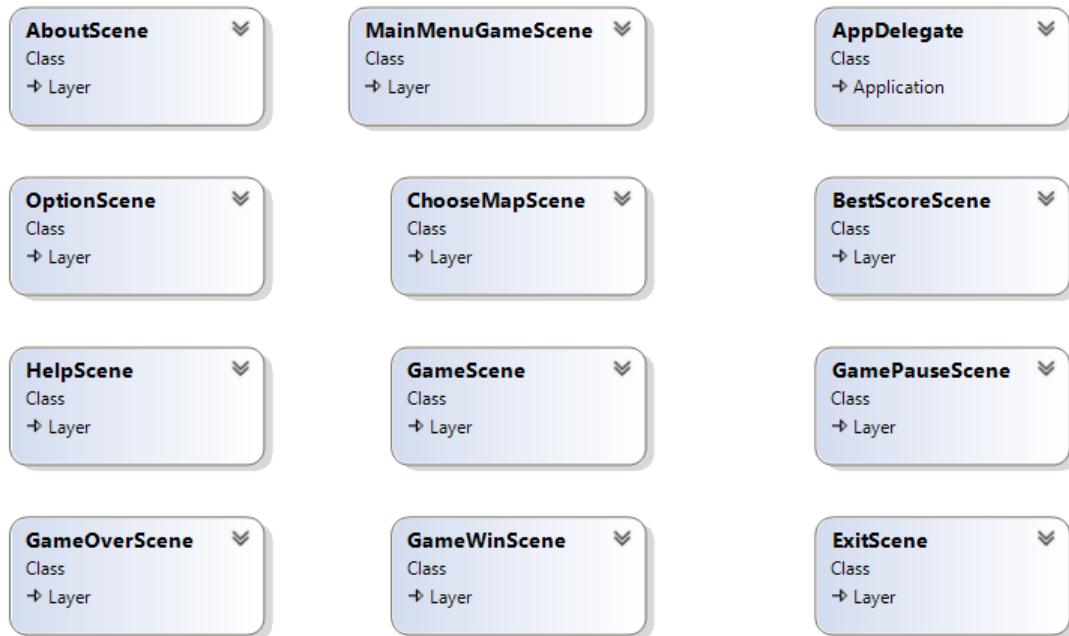
Hình 55: Đối tượng lưu trữ thông tin người chơi game Internship AH!

- Các lớp tùy chỉnh Sprite để vẽ cho các đối tượng trong game



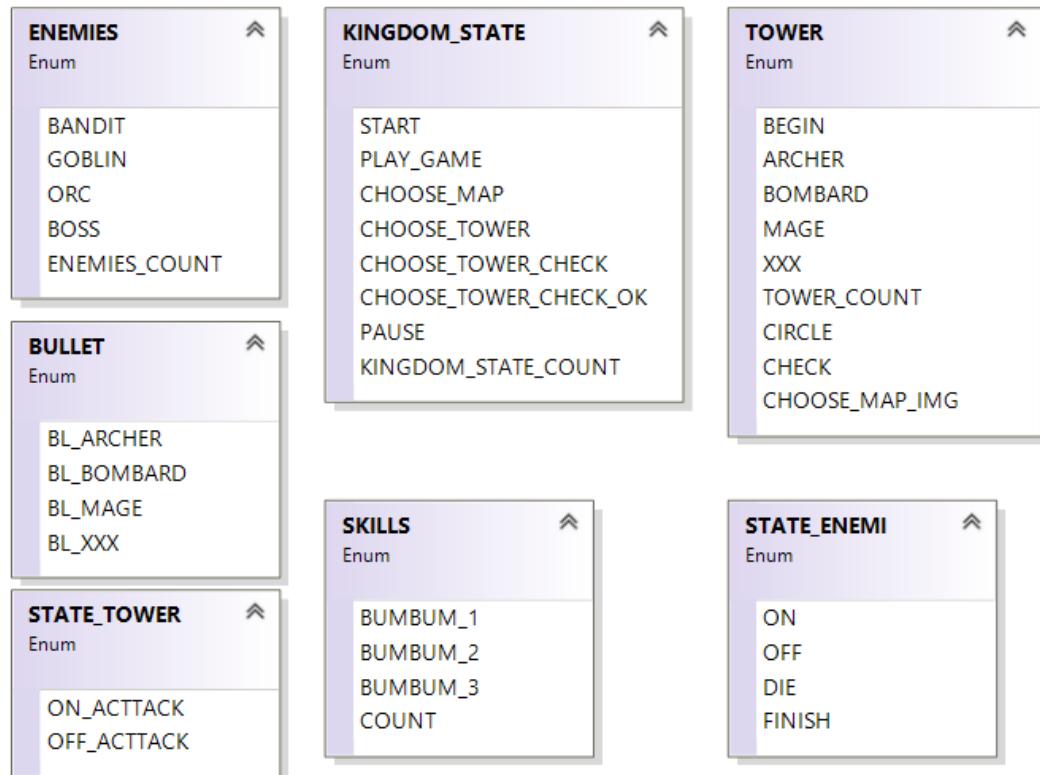
Hình 54: Các lớp tùy chỉnh Sprite dùng trong game Internship AH!

- Các màn hình game



Hình 56: Các màn hình trong game

- Các Enum để đánh dấu các đối tượng trong game



Hình 57: Các Enum đánh dấu các Sprite cho các đối tượng

2.4. Chuẩn bị tài nguyên cho game

Tổ chức lưu trữ thông số cho các loại trụ (Tower), quái (Enemi) và bản đồ (Map) dưới dạng file .txt

- File “map.txt”: lưu tất cả các bản đồ (Map) có trong game, mỗi bản đồ sẽ lưu gồm 4 dòng và bắt đầu bằng “#” (đường dẫn file .tmx, tên bản đồ, số tiền của bản đồ đó và kịch bản của bản đồ đó)

Chỉ số	Mô tả	Bắt buộc
FILE_PATH	Tên đường dẫn lưu file bản đồ .tmx	Có
NAME	Tên bản đồ.	Có
GOLD	Số vàng của bản đồ	Có
ENEMIES	Kịch bản của bản đồ.	Có

Bảng 12: Chỉ số của file thông tin bản đồ game

Các bản đồ có trong game

```
#FILE_PATH TildeMap/map1.tmx
NAME Map1
GOLD 265
ENEMIES @+1x3:3+0x3:1+2x3:2-0@+0x6:3-25@+0x9:3-25@+0x4:2+2x1:4-25
#FILE_PATH TildeMap/map2.tmx
NAME Map2
GOLD 269
ENEMIES @+1x5:3+0x4:1+2x3:2-0@+0x10:3-25@+0x9:3-25@+0x4:2+2x1:4-25
#FILE_PATH TildeMap/map3.tmx
NAME Map3
GOLD 369
ENEMIES @+1x5:3+0x4:1+2x3:2-0@+0x10:3-25@+0x9:3-25@+0x4:2+2x1:4-25
```

Hình 58: File map.txt

- File “Enemi.txt”: lưu thông số của các loại quái có trong game, Mỗi quái bắt đầu bằng “#” gồm các thông tin:

Chỉ số	Mô tả	Bắt buộc
NAME	Tên của quái	Có

HP	Chỉ số máu	Có
DAMAGE	Chỉ số sức mạnh	Có
ARMOR	Chỉ số giáp	Không
MAGIC	Chỉ số phép thuật	Không
SPEED	Chỉ số tốc độ	Có
LIVE	Chỉ số máu mất cho người chơi	Có
BOUNTY	Chỉ số vàng khi quái chết	Có
ABILITIES	Chỉ số kỹ năng của quái	Có

Bảng 13: Chỉ số các loại quái(Enemi) trong game

```

#NAME Bandit
HP 70
DAMAGE 20
ARMOR 0
MAGIC 0
SPEED 0.8
LIVE 1
BOUNTY 8
ABILITIES Can dodge melee attacks
#NAME Goblin
HP 20
DAMAGE 2
ARMOR 0
MAGIC 0
SPEED 0.9
LIVE 1
BOUNTY 8
ABILITIES Can dodge melee attacks
#NAME Orc
HP 80
DAMAGE 6
ARMOR 0
MAGIC 0
SPEED 1.0
LIVE 1
BOUNTY 8
ABILITIES Can dodge melee attacks
#NAME Shaman
HP 100
DAMAGE 4
ARMOR 0
MAGIC 0
SPEED 0.9
LIVE 1
BOUNTY 8
ABILITIES Can dodge melee attacks

```

Hình 59: File "enemies.txt"

Các loại quái (Enemi) trong game:

- File “Towers.txt”: lưu thông số các trụ trong game bao gồm cả trụ cơ bản và các trụ sau khi nâng cấp (upgrade)

Mô tả	Chỉ số	Bắt buộc
NAME	Tên của trụ.	Có
LEVEL	Chỉ số cấp của trụ.	Có
BUILD	Chỉ số vàng cần khi xây trụ.	Có
UPGRADE	Chỉ số vàng cần khi nâng cấp.	Có

MAGAGE	Chỉ số sức mạnh của trụ khi đánh quái.	Có
SPEED	Chỉ số tốc độ đánh của trụ.	Có
RANGE	Chỉ số bán kính đánh quái của trụ	Có
TO	Tên trụ sau khi nâng cấp lên	Có
TYPE	Loại trụ	Có

Bảng 14: Chỉ số các loại trụ (Tower) trong game

```

#NAME TwArcher
LEVEL 1
BUILD 60
UPGRADE 100
DAMAGE 5
SPEED 2
RANGE 100
TO TwArcher2
TYPE 0
#NAME TwArcher2
LEVEL 2
BUILD 100
UPGRADE 150
DAMAGE 9
SPEED 1.2
RANGE 110
TO TwArcher3
TYPE 0
#NAME TwArcher3
LEVEL 3
BUILD 150
UPGRADE 200
DAMAGE 18
SPEED 0.5
RANGE 100
TO
TYPE 0
#NAME TwMage
LEVEL 1
BUILD 70
UPGRADE 110
DAMAGE 8
SPEED 2.5
RANGE 100
TO TwMage2
TYPE 2

```

Hình 60: File "towers.txt"

Các loại trụ trong game Mở rộng thêm các loại Enemi, Tower, Map

2.3.1. Resource Tile Map

Dùng phần mềm Tiled Map Editor để tạo đồ. Resource của mỗi bản đồ gồm:

- File “.tmx” lưu các thông số của bản đồ được xuất ra từ Tiled Map Editor
- File ảnh nguồn lúc tạo bản đồ khi thêm “Tilesets”

2.3.2. Resource Image

Dùng photoshop để thiết kế hình ảnh dùng trong game gồm:

- Các ảnh nền của mỗi màn hình và các “Sprite” tại mỗi màn hình. (Các “Sprite” ảnh xuất ra có 2 trạng thái: Hiển thị Sprite và Nhấn vào Sprite)
- Các Sprite cho các trụ (Tower) và các loại quái (Enemi) cho trong game.

2.3.3. Resource Sprite Sheet

Mỗi nhân vật có để tạo Animation cần có 1 file cấu hình .plist và 1 file ảnh nguồn .png được xuất ra từ phần mềm TexturePacker

2.3.4. Resource Audio

Audio gồm có âm thanh nền trong game, và các âm thanh hiệu ứng chỉ có tương tác:

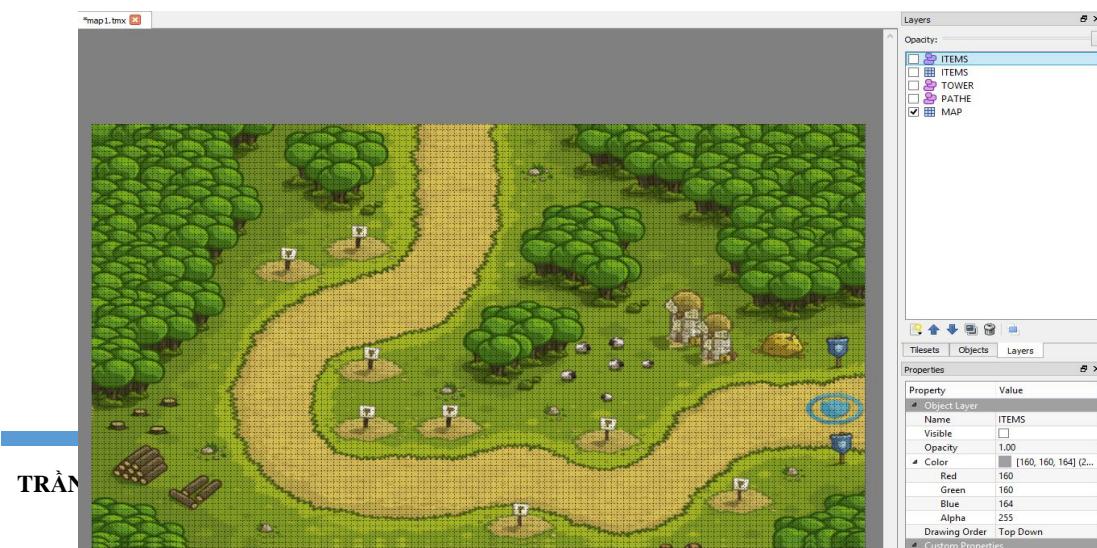
- Quái chết.
- Trụ đánh quái.
- Quái vây tới đích.
- Thắng game.
- Thua game.

2.5. Thiết kế bản đồ cho game

Tạo map với kích thước 128x77 và Tile = 10px x 10px

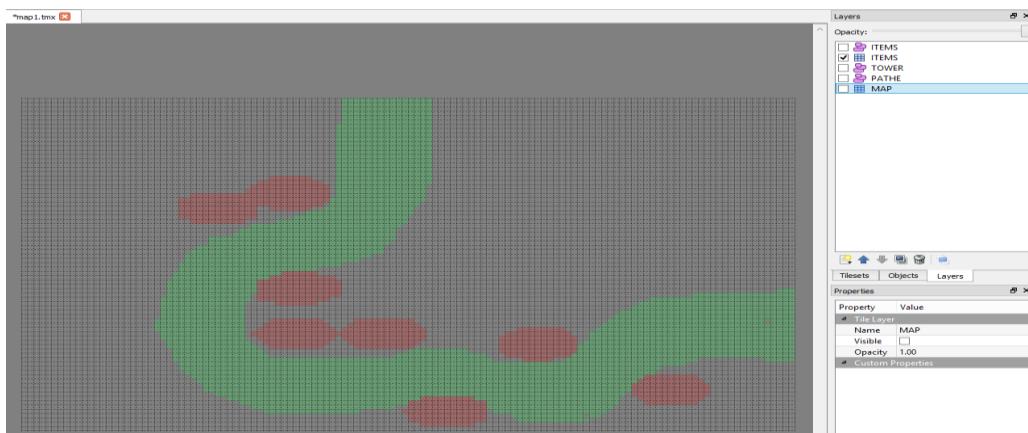
Thêm Objectlayer Tower và object để đánh dấu vị trí xây dựng trụ và đánh dấu đường đi cho quái trong map.

Bước 1: Tạo Tile Layer BackGround cho map



Hình 61: Tạo bản đồ cho game Internship AH! bằng Tiled Map Editor

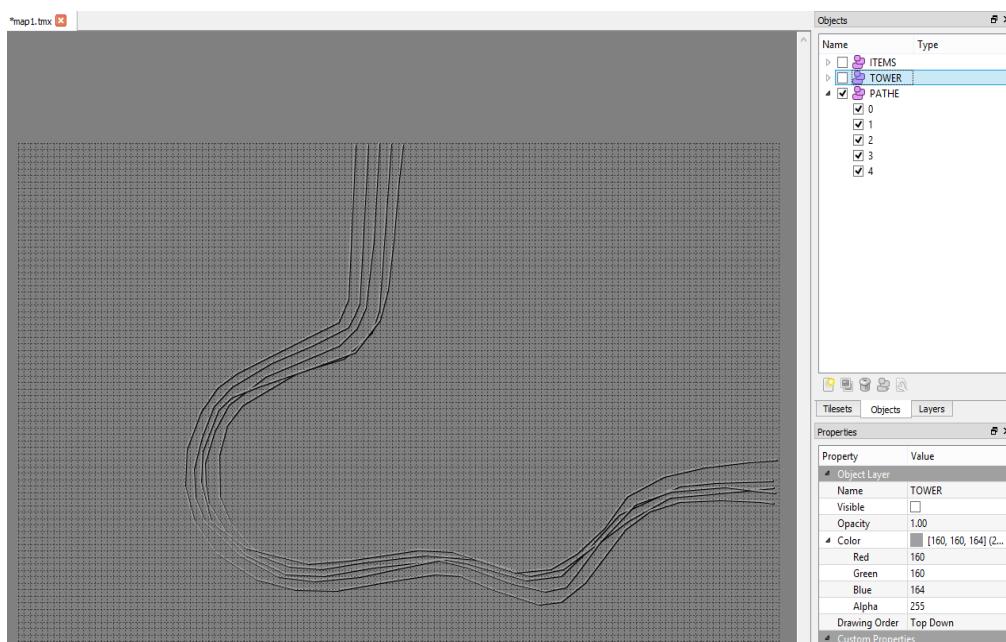
Bước 2: Tạo Tile Layer đánh dấu vị trí xây dựng Tower và phạm vi đường đi của Enemi trên map



Hình 62: Đánh dấu trên bản đồ vị trí xây trụ và đường đi của quái

Bước 3: Tạo Object Layer “PATHE” đánh dấu đường đi của Enemi trên map

Các đường đi được đánh số từ 0 → hết số lượng tùy map.



Hình 63: Lưu trữ tượng các đường của quái

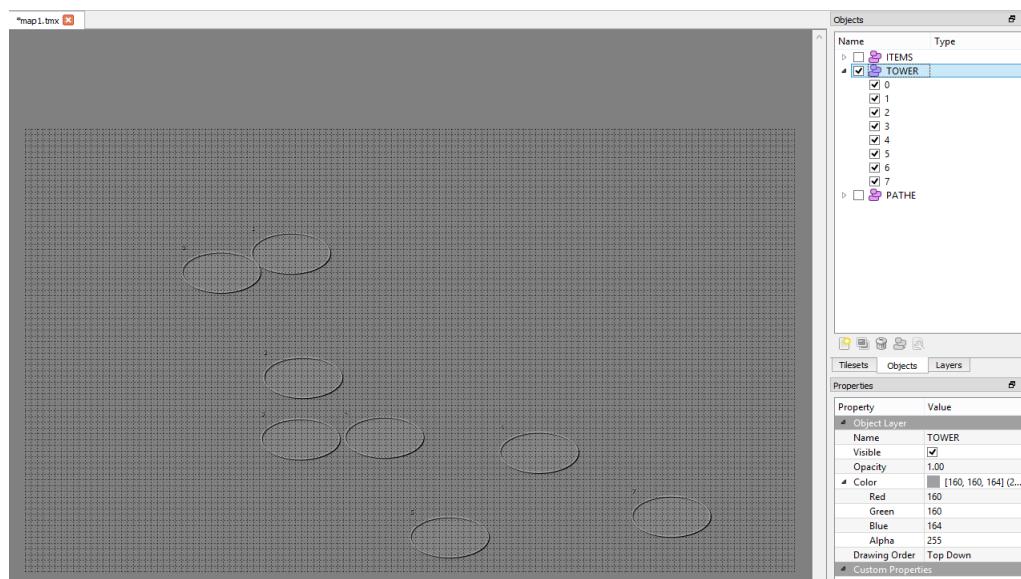
Bước 4: Tạo Object Layer “ITEM” đánh dấu vị trí Enemi sẽ bắt đầu xuất hiện trong map



Hình 64: Lưu trữ tượng vị trí bắt đầu xuất hiện của quái trên bản đồ

Bước 5: Tạo Object Layer “TOWER” lưu vị trí xây các Tower trên map

Các Tower được đánh số từ bắt đầu từ 0 đến hết số lượng theo tùy map



Hình 65: Lưu vị trí xây các trụ trên bản đồ

2.6. Thiết kế Sprite Sheet

2.6.1. *Sprite Sheet cho quái (Enemy)*

2.6.2. *Sprite Sheet cho trụ (Tower)*

2.7. Viết Code

2.7.1. *Tạo các mảng hình game theo Flow Chart.*

Thêm các mảng hình theo Flow Chart bao gồm:

- AboutScene
- BestScoreScene
- ChooseMapScene
- ExitScene
- GameOverScene
- GamePauseScene
- GameScene
- GameWinScene
- HelpScene
- MainMenuGameScene
- OptionScene
- ShopScene

Mỗi màn hình thêm 2 file .h, .cpp để viết code cho lớp đó.

Khi game bắt đầu “AppDelegate” sẽ bắt đầu màn hình “MainMenuGameScene”, tại đây màn hình sẽ tải hình nền của game và nút “Start” để bắt đầu vào chọn bản đồ để chơi game. (Ngoài ra còn có 1 số nút phụ để chuyển qua các màn hình như thông tin tác giả game, thông tin cài đặt của game, nút bắc tắt âm thanh trong game, ...)

- Viết code cho hình chính khi vào game

Thêm vào 2 file MainMenuGameScene.h và MainMenuGameScene.cpp và viết code như sau:

Ở file MainMenuGameScene.h:

```
#ifndef __MAIN_MENU_GAME_SCENE_H__
#define __MAIN_MENU_GAME_SCENE_H__

#include "cocos2d.h"

USING_NS_CC;

class MainMenuGameScene
    : public cocos2d::Layer {
private:
    void DoneSprite(cocos2d::Node* pSender);
public:
    // there's no 'id' in cpp, so we recommend returning the class instance pointer
    static cocos2d::Scene* createScene();

    // Here's a difference. Method 'init' in cocos2d-x returns bool, instead of returning 'id' in cocos2d-iphone
    virtual bool init();
        CREATE_FUNC(MainMenuGameScene);

    // Menu
    void PlayGame(cocos2d::Ref *pSender);
    void ExitGame(cocos2d::Ref *pSender);
    void AboutGame(cocos2d::Ref *pSender);
    void OptionGame(cocos2d::Ref *pSender);
    void BestScoreGame(cocos2d::Ref *pSender);
    void HelpGame(cocos2d::Ref *pSender);

    float scale = 2.0;
};

#endif // __MAIN_MENU_GAME_SCENE_H__
```

Ở file MainMenuGameScene.cpp:

```
#include "MainMenuGameScene.h"
#include "GameScene.h"
```

```

#include "ChooseMapScene.h"
#include "HelpScene.h"
#include "OptionScene.h"
#include "ShopScene.h"
#include "AboutScene.h"
#include "BestScore.h"
#include <cocosstudio/CocoStudio.h>
using namespace cocostudio;

USING_NS_CC;

Scene* MainMenuGameScene::createScene(){
    auto scene = Scene::create();
    auto layer = MainMenuGameScene::create();
    scene->addChild(layer);
    return scene;
}
bool MainMenuGameScene::init(){
    //////////////////////////////
    // 1. super init first
    if ( !Layer::init() )
    {
        return false;
    }

    Size visibleSize = Director::getInstance()->getVisibleSize();
    Vec2 origin = Director::getInstance()->getVisibleOrigin();

    auto sprite = cocos2d::Sprite::create("scene/background_kingdom.png");
    float w = visibleSize.width;
    float h = visibleSize.height;
    sprite->setPosition(Vec2(w / 2, h / 2));
    this->addChild(sprite,-2);
    float b_scale = scale;
    auto sp = cocos2d::Sprite::create("scene/logo_kingdom.png");
    sp->setPosition(Vec2(w / 2, h));
    this->addChild(sp, -2);
    sp->setScale(scale);
    sp->setTag(CONTROLS::LOGO);
    auto actionMove = cocos2d::MoveTo::create(0.15, Vec2(w / 2, h - h / 4));
    auto actionDone = CallFuncN::create(CC_CALLBACK_1(MainMenuGameScene::DoneSprite, this));
    auto action = Sequence::create(actionMove, actionDone, NULL);
    sp->runAction(action);

    return true;
}
void MainMenuGameScene::PlayGame(Ref *pSender) {
    auto scene = ChooseMapScene::createScene();
    Director::getInstance()->replaceScene(TransitionMoveInL::create(0.3, scene));
}
void MainMenuGameScene::ExitGame(Ref *pSender) {
#if (CC_TARGET_PLATFORM == CC_PLATFORM_WP8) || (CC_TARGET_PLATFORM == CC_PLATFORM_WINRT)
    MessageBox("You pressed the close button. Windows Store Apps do not implement a close
button.", "Alert");
    return;
#endif
    Director::getInstance()->end();
}
#if (CC_TARGET_PLATFORM == CC_PLATFORM_IOS)
    exit(0);
#endif
}

```

```

void MainMenuGameScene::HelpGame(Ref *pSender)
{
    auto scene = HelpScene::createScene();
    Director::getInstance()->pushScene(scene);
}

void MainMenuGameScene::OptionGame(Ref *pSender) {
    //CCLOG("First menu tapped");
    auto scene = OptionScene::createScene();
    Director::getInstance()->pushScene(scene);
}

void MainMenuGameScene::BestScoreGame(Ref *pSender) {
    auto scene = BestScoreScene::createScene();
    Director::getInstance()->pushScene(scene);
}

void MainMenuGameScene::AboutGame(Ref *pSender) {
    auto scene = AboutScene::createScene();
    scene->setContentSize(cocos2d::Size(300, 300));
    Director::getInstance()->pushScene(scene);
}

```

- Viết code cho màn hình chọn bản đồ:

Thêm vào 2 file ChooseMapScene.h và ChooseMapScene.cpp và viết code như sau: Khi khởi tạo, cần lấy lên giá trị bản đồ người chơi đã chơi qua (Max Map Done).

Khi vẽ Sprite cho các vị trí các bản đồ nếu bản đồ nào người chơi chưa chơi tới cần làm mờ đi và k không người chơi chọn bản đồ đó.

Ở file ChooseMapScene.h:

Ở file ChooseMapScene.cpp:

2.1.2. Xây dựng trụ.

Để xây dựng trụ ta cần chuẩn bị các Item của mỗi trụ và tương ứng giá của mỗi trụ.



Ở hàm init của GameScene sau khi lấy bản đồ từ ResourceManager, lấy dữ liệu bản đồ và lưu vào biến TMXTiledMap, Lưu lại Layer MAP đánh dấu bên file Tiled Map lúc thiết kế bản đồ, để nhận biết được những điểm nào trên bản đồ được phép xây trụ.

Khai báo các biến ở file GameScene.h để tải bản đồ và lưu các Layer đã đánh dấu bên file Tiled Map và khai báo vector để lưu :

- vector các Object Eclipse lưu vị trí có thể xây trụ trên bản đồ.
- vector các Tower đã được xây trên bản đồ.

Code như sau:

```
cocos2d::TMXTiledMap *_tiledMap;
cocos2d::TMXLayer *_map;
cocos2d::TMXLayer *_items;
std::vector<gamekingdom::Eclipse> vec_tower;
std::vector<gamekingdom::Object*> vec_Tower;
```

Đồng thời, lưu lại mảng các object Eclipse được vẽ bên TiledMap

Code như sau:

```
this->_Map = gamekingdom::ResourceManager::getInstance()->getMap(src);
_tiledMap = _Map->getTileMap();
// Load Tiled Map
this->addChild(_tiledMap);
_items = _tiledMap->layerNamed("ITEMS");
_items->setVisible(false);
_map = _tiledMap->layerNamed("MAP");
_map->setVisible(true);
 addButton();

// Get Eclipse of Tower
TMXObjectGroup *obj_Tower = _tiledMap->getObjectGroup("TOWER");
int n = 0;
if (obj_Tower != nullptr) {
    n = obj_Tower->getObjects().size();
    for (int i = 0; i < n; i++) {
        auto ob = obj_Tower->objectNamed(std::to_string(i));
        if (!ob.empty()) {
            Size size = Size(ob["width"].asFloat(), ob["height"].asFloat());
            float x = ob["x"].asFloat() + size.width/2;
            float y = ob["y"].asFloat() + size.height/2;
            Vec2 v = Vec2(x, y);
            vec_tower.push_back(Eclipse(v, size));
        }
    }
}
```

Vào hàm “onTouchEnded” để bắt sự kiện người chơi chạm vào màn hình để xây trụ, khi toạ độ người chơi chạm đúng vào những điểm có đánh dấu thuộc Layer MAP và có Properties COLOR = RED thì xem xét vị trí đó có xây trụ không.

```
Point touchLocation = this->tileCoordForPosition(touch->getLocationInView());
int tiled_Item = _items->tileGIDAt(touchLocation);
if (tiled_Item) {
    auto properties = _tiledMap->getPropertiesForGID(tiled_Item).asValueMap();
    if (!properties.empty()) {
        auto collectable = properties["COLOR"].asString();
        if ("RED" == collectable) {
            // kiểm tra vị trí có được xây trụ không!
        }
    }
}
```

Khi người chơi đã chọn đúng vào 1 trong các điểm được đánh dấu trong layer MAP, tìm trong mảng các Object Eclipse điểm có khoảng cách gần nhất với điểm người chơi đang chạm để xây trụ.

```
int vt = -1;
if (vec_tower.size() == 0) {
    return;
}

float distance = vec_tower[0].Position.getDistance(Vec2(x,y));
for (int i = 0; i < vec_tower.size(); i++) {
    if (vec_tower[i].STATE == false) {
        continue;
    }
    float dis = vec_tower[i].Position.getDistance(Vec2(x, y));
    if (dis <= distance) {
        vt = i;
        distance = dis;
    }
}
```

Kiểm tra trong khi mảng vị trí Object Eclipse còn trống thì cho phép người chơi xây trụ. Xét bán kính của Eclipse để xem xét khi điểm người chơi chạm vào có thuộc Eclipse không và tài Sprite “choose_map” và gọi hàm xây dựng Sprite Tower:

```
if (vec_tower.size() != 0) {
    if (distance <= vec_tower[0].Size.height/2 ||
        distance <= vec_tower[0].Size.width/2) {
        this->CountTouch = this->CountTouch + 1;
        switch (this->CountTouch) {
            case 1:
            {
                GAME_STATE = KINGDOM_STATE::CHOOSE_TOWER;
                break;
            }
            case 2:
            {
                break;
            }
            case 3:
            {
                break;
            }
        default:
    }}
```

```
        break;
    }

    float x = vec_tower[vt].Position.x;
    float y = vec_tower[vt].Position.y;
    // Choose Tower
    this->GAME_STATE = KINGDOM_STATE::CHOOSE_TOWER;
    auto chooseMap = cocos2d::Sprite::create("Sprite/choose/choose_map.png");
    chooseMap->setPosition(x, y);
    chooseMap->setTag(TOWER::CHOOSE_MAP_IMG);

    this->addChild(chooseMap);
    this->POSITION_CHOOSE_MAP = vt;
    chooseMap->setScale(0);

    auto actionScale = cocos2d::ScaleTo::create(0.2, 0.9);

    auto actionFin = CallFuncN::create(CC_CALLBACK_1(GameScene::spriteChooseTower, this));

    auto action = Sequence::create(actionScale, actionFin, NULL);
    chooseMap->runAction(action);
}
```

Code hàm xây dựng Sprite:

```
void GameScene::spriteChooseTower(cocos2d::Node *pSender){  
    cocos2d::Sprite *p = dynamic_cast<cocos2d::Sprite*>((cocos2d::Sprite*)pSender);  
    if (p != NULL) {  
        // Xây dựng các Sprite Tower và giá tiền tương ứng  
    }  
}
```

2.1.3. Thêm vật lý vào cho game.

Để thêm vật lý vào game vào hàm createScene thay đổi phương thức createScene thành createSceneWithPhysics, và thiết lập gravity Vect(0.0f,0.0f) để các đối tượng k bị rơi do tác động của gia tốc vật lý.

Code như sau:

```
Scene* GameScene::createScene(){
    auto scene = Scene::createWithPhysics();
    scene->getPhysicsWorld()->setGravity(Vect(0.0f,0.0f));
    auto layer = GameScene::create();
    scene->addChild(layer);
    return scene;
}
```

Tiếp theo vào hàm init để thêm sự kiện lắng nghe vật lý cho màn hình game

```
// Add Physics
auto contactListener = EventListenerPhysicsContact::create();
contactListener->onContactBegin = CC_CALLBACK_1(GameScene::onContactBegin, this);
eventDispatcher->addEventListenerWithSceneGraphPriority(contactListener, this);
```

Thêm định nghĩa hàm cho hàm onContactBegin ở file GameScene.h

```
bool onContactBegin(const cocos2d::PhysicsContact &contact);
```

Và định nghĩa cụ thể ở GameScene.cpp

```
bool GameScene::onContactBegin(const cocos2d::PhysicsContact &contact){
    // code xử lý va chạm vật lý
}
```

Để 1 đối tượng khi thêm vào màn hình của game có chịu tác động của vật lý, cần xét thuộc tính body Physics cho đối tượng đó, ví dụ:

```
void Towers::addTo(cocos2d::Scene* pscene){
    this->pScene = pscene;
    // bán kính chịu tác động vật lý của đối tượng
    float radius = this->getRadiusAttack
    auto tower_body = cocos2d::PhysicsBody::createCircle(radius);
    tower_body->setContactTestBitmask(0x1);
    this->getSprite()->setPhysicsBody(tower_body);
    pscene->addChild(this->getSprite());
}
```

2.1.4. Kiểm tra va chạm giữa quái và trụ.

Để kiểm tra va chạm khi quái(Enemi) di chuyển gần lại trụ (Tower) chúng ta cần:

- Khi khởi tạo các Sprite của quái, trụ cần xét tag tương ứng cho Sprite để nhận biết đối tượng.
- Vào hàm onContactBegin để lấy 2 đối tượng va chạm và nhận biết nhờ tag của chúng.

Hàm onContactBegin xét va chạm cho tất cả các đối tượng, chỉ cần xét sự va chạm giữa quái (Enemi) và trụ (Tower) còn các va chạm giữa quái và quái hay trụ với trụ cần trả về false để 2 đối tượng không chịu tác động của vật lý.

Khi khởi tạo 1 đối tượng cần xét tag cho Sprite như sau:

```
TwBomBand::TwBomBand(Towers* to){
    this->setSprite(new gamekingdom::SpriteBomBand());
    this->getSprite()->setTag(TOWER::BOMBARD);
}
```

Nhận biết đối tượng va chạm ở hàm onContactBegin như sau:

```
bool GameScene::onContactBegin(const cocos2d::PhysicsContact &contact){
    auto target = (Sprite*)contact.getShapeA()->getBody()->getNode();
    int tag = target->getTag();
    auto target1 = (Sprite*)contact.getShapeB()->getBody()->getNode();
    int tag1 = target1->getTag();

    gamekingdom::Object *tower = nullptr;
    int pos_En = -1;
    if (Enemies::isEnemies(tag)) {
        pos_En = getObj(target, vec_Enemies);
        if (Towers::isTower(tag1)) {
            return false;
        }
        else {
            if (tag1 == BULLET::BL_BOMBARD) {
                goto KILL_ENEMY;
            }
            else {
                if (this->Skill_OK) {
                    // xử lý va chạm
                }
            }
        }
    }
}
```

```

        }
    }
else {
    if (Enemies::isEnemies(tag1)) {
        pos_En = getObj(target1, vec_Enemies);
        if (Towers::isTower(tag)) {
            return false;
        }
    else {
        if (tag == BULLET::BL_BOMBARD) {
            goto KILL_ENEMY;
        }
    else {
        if (this->Skill_OK) {
            // xử lý va chạm
        }
    }
}
else {
    return false;
}
}
return false;
KILL_ENEMY:
((gamekingdom::Enemies*)vec_Enemies[pos_En])->setHP(((gamekingdom::Enemies*)vec_Enemies[pos_En])->getHP() - 10);
return false;
}
}

```

2.1.5. Xây dựng hệ thống nâng tru.

2.1.6. Xây dựng tính điểm cho người chơi và chuyển bản đồ khi thắng game.

Thông tin người chơi cần lưu lại số bản đồ người chơi đã chơi qua, số tiền người chơi đạt được. Tuân thủ mẫu “**Singleton design Pattern**” để đảm bảo thông tin người chơi được load vào game 1 lần duy nhất khi chạy game, lưu trữ lại khi người chơi kết thúc game và cập nhật khi người chơi chơi thắng tại mỗi bản đồ

Thêm vào 2 file **GameKingdomUser.h** và **GameKingdomUser.cpp**. Và bắt đầu viết code:

Ở file GameKingdomUser.h:

```

#ifndef __gamekingdom_KingdomUser__
#define __gamekingdom_KingdomUser__

#include "cfGameKingdom.h"

NS_GAMEKINGDOM_BEGIN

class KingdomUser{
public:
    static KingdomUser* instance;
    static KingdomUser* getInstance();
    void setMapDone(int mapDone);
}

```

```
void setGold(int gold);
int getMapDone();
int getGold();

private:
    KingdomUser();
    ~KingdomUser();
    int MapDone = 0, Gold = 0;
};

NS_GAMEKINGDOM_END
#endif /* defined(__gamekingdom_KingdomUser__) */
```

Ở file GameKingdomUser.cpp

```
#include "KingdomUser.h"

NS_GAMEKINGDOM_BEGIN

KingdomUser* KingdomUser::instance = nullptr;

KingdomUser::KingdomUser() {
    this->MapDone = 0;
    this->Gold = 0;
}

KingdomUser::~KingdomUser() { }

KingdomUser* KingdomUser::getInstance() {
    if (KingdomUser::instance == nullptr) {
        KingdomUser::instance = new KingdomUser();
    }
    return KingdomUser::instance;
}

void KingdomUser::setMapDone(int mapDone) {
    this->MapDone = mapDone;
}

int KingdomUser::getMapDone() {
    return this->MapDone;
}

void KingdomUser::setGold(int gold) {
    this->Gold = gold;
}

int KingdomUser::getGold() {
    return this->Gold;
}

NS_GAMEKINGDOM_END
```

TÀI LIỆU THAM KHẢO

Tài liệu training của GameLoft

- Training CC++
- TortoiseSVN

Ebook Tham khảo

- **Cocos2d-x by Example Beginner's Guide**, Roger Engelbert
- **Cocos2d-x Game Development Essentials**, Frahaan Hussain – Arutosh Gurung – Gareth Jones.

Websites tham khảo

- <http://cocos2d-x.org/programmersguide/>
- <http://www.cplusplus.com/doc/tutorial/>
- <http://bfxr.net/>

KẾT LUẬN

Với khoảng thời gian không nhiều trong khóa thực tại Em đã một lần nữa củng cố lại kiến thức lập trình C++, biết cách sử dụng Engine Cocos2d-x để làm game đa nền tảng. Bên cạnh đó đã học hỏi được quy trình làm game và nhiều kỹ thuật sử dụng trong game như tải và load tài nguyên cho game.

Bên cạnh đó vẫn không tránh khỏi nhiều thiếu sót, và đặc biệt còn nhiều mảng kiến thức cần tìm hiểu kỹ hơn và cần rèn luyện nhiều kỹ năng như: quản lý thời gian, trình bày báo cáo, kỹ năng thuyết trình. Và cần phải làm nhiều để đúc kết ra nhiều kinh nghiệm hơn.