TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

──────── \* ────────



ĐỒ ÁN

**TỐT NGHIỆP ĐẠI HỌC**

NGÀNH HỆ THỐNG THÔNG TIN

XÂY DỰNG CỔNG GAME DI ĐỘNG

TRÊN NỀN TẢNG COCOS2D-X

Sinh viên thực hiện : **Nguyễn Tuấn Trí**

Lớp CNTT-TT 2.2 – K56

Giảng viên hướng dẫn: **ThS. Nguyễn Hồng Phương**

HÀ NỘI 05-2016

**PHIẾU GIAO NHIỆM VỤ ĐỒ ÁN TỐT NGHIỆP**

1. Thông tin về sinh viên

Họ và tên sinh viên: Nguyễn Tuấn Trí

Điện thoại liên lạc: 09937 09718 Email: [nguyen.tuan.tri@gmail.com](mailto:nguyen.tuan.tri@gmail.com)

Lớp: CNTT-TT 2.2 K56 Hệ đào tạo: Đại học chính quy

Đồ án tốt nghiệp được thực hiện tại: Bộ môn Hệ thống Thông tin, Viện CNTT & TT, Trường Đại học Bách Khoa Hà Nội

Thời gian làm đồ án tốt nghiệp: Từ ngày 26/01/2015 đến ngày 25/05/2015

2. Mục đích nội dung của ĐATN

* Xây dựng cổng game di động trên nền tảng cocos2dx.

3. Các nhiệm vụ cụ thể của ĐATN

* Tìm hiểu tổng quan về công nghệ làm game cocos2d-x.
* Phát triển một sản phẩm game thực tế trên nền tảng cocos2d-x.
* Nghiên cứu, đề xuất giải pháp xây dựng cổng game trên nền tảng cocos2d-x:
* Đề xuất giải pháp
* Hiện thực hóa giải pháp, để chứng minh tính khả thi của nó

4. Lời cam đoan

Tôi – *Nguyễn Tuấn Trí* – cam kết ĐATN là công trình nghiên cứu của bản thân tôi dưới sự hướng dẫn của *ThS. Nguyễn Hồng Phương*.

Các kết quả nêu trong ĐATN là trung thực, không phải là sao chép toàn văn của bất kỳ công trình nào khác.

*Hà Nội, ngày……tháng.…..năm……….*

Tác giả ĐATN

*Nguyễn Tuấn Trí*

5. Xác nhận của giảng viên hướng dẫn về mức độ hoàn thành của ĐATN, cho phép bảo vệ:

*Hà Nội, ngày……tháng.…..năm……….*

Giảng viên hướng dẫn

*ThS. Nguyễn Hồng Phương*

Tóm tắt nội dung đồ án tốt nghiệp

Hiện nay, game mobile là một trong những lĩnh vực phát triển mạnh mẽ, có doanh số lớn nhất trong toàn bộ thị trường game. Hầu hết các nhà phát triển game đều tập trung cho thị trường game mobile thay cho thị trường web game truyền thống. Trước đây, trong thời kỳ phát triển của web game, những nhà phát triển lớn đều xây dựng cho mình một cổng game trên nền web, đóng vai trò như một kênh phân phối và kênh marketing hiệu quả. Thì hiện nay, với xu hướng phát triển mạnh mẽ của game mobile, nhiều nhà phát triển game cũng dự định xây dựng cho mình một cổng game di động.

Cổng game di động là một ứng dụng di động, ứng dụng này đóng vai trò như một cửa ngõ mà từ đó, người chơi có thể trải nghiệm nhiều game khác nhau của một hoặc nhiều nhà phát triển. Có một ý tưởng khá hay cho việc xây dựng cổng game di động đó là, các game di động sẽ được đóng gói lại, theo một định dạng nào đó. Thông qua cổng game di động, người chơi có thể tải về các game này từ server. Sau đó chơi game trực tiếp trên nền ứng dụng mà không cần cài đặt game như một ứng dụng độc lập trên điện thoại.

Trong đồ án này, em đã tìm hiểu về cocos2d-x, một game engine được nhiều nhà phát triển game ở Việt Nam sử dụng. Và để thực sự hiểu về công nghệ cocos2d-x, em đã phát triển một sản phẩm game trên nền tảng này.

Tiếp đó, đồ án thực hiện nghiên cứu để đề xuất giải pháp công nghệ cho ý tưởng xây dựng cổng game di động được đưa ra ở trên. Đồ án đã đề xuất ra 2 giải pháp, so sánh ưu nhược điểm của chúng và lựa chọn được một giải pháp phù hợp. Dựa trên giải pháp này, ta hiện thực hóa nó, xây dựng các công cụ phần mềm:

* Công cụ đóng gói game để tạo ra các game pack từ một project cocos2d-x của một game cụ thể.
* Một server nơi các nhà phát triển release các game pack của họ.
* Một ứng dụng cổng game di động, download các gamepack từ server, giúp người dùng trải nghiệm game trực tiếp trên nền ứng dụng mà không cần cài đặt game như một ứng dụng độc lập.

Lời cảm ơn

Được sống, học tập và làm việc dưới mái trường Đại học Bách Khoa Hà Nội có lẽ là niềm vinh dự, niềm hạnh phúc lớn lao nhất mà cuộc đời đã hào phóng ban tặng cho em. Đặc biệt hơn nữa nhận được sự giảng dạy tận tình, tâm huyết của các Thầy, Cô giảng viên Viện Công nghệ Thông tin và Truyền thông, chúng em đã tiếp nhận được những kiến thức bổ ích. đã ngày càng trưởng thành trong suy nghĩ và nhân cách. Đó sẽ là nền tảng vững chắc trên con đường tiến tới thành công trong công việc và cuộc sống sau này.

“Luôn tự hào là sinh viên Bách Khoa Hà Nội,

sinh viên Viện Công nghệ Thông tin và Truyền thông!”

Đồ án tốt nghiệp là cơ hội tốt để chúng em áp dụng những kiến thức được học vào thực hành, giải quyết các vấn đề thực tế. Cũng là một cột mốc chúng em cần phát nỗ lực, cố gắng vượt qua một cách xuất sắc nhất để chứng minh kiến thức đã học, khả năng của bản thân. Trong suốt quá trình làm đồ án, em đã nhận được nhiều sự giúp đỡ từ bạn bè, thầy cô, đặc biệt là thầy Nguyễn Hồng Phương. Thầy đã giành nhiều thời gian quý báu để tận tình hướng dẫn, hỗ trợ trong việc tìm tài liệu, thảo luận đưa ra gợi ý giải quyết cho những vấn đề khó khăn giúp em luôn có định hướng tốt và bám sát tiến độ đồ án.

Một lần nữa, với lòng biết ơn sâu sắc em xin gửi lời cảm ơn chân thành đến người Thầy tuyệt vời Nguyễn Hồng Phương, cùng toàn thể thầy cô cũng như bạn bè đã hỗ trợ giúp đỡ em cả về kiến thức và tinh thần để hoàn thành tốt đồ án trong thời gian qua.

Em xin kính chúc Thầy, Cô luôn mạnh khỏe, hạnh phúc và thành công.

Em chân thành cảm ơn!

Mục lục

[Tóm tắt nội dung đồ án tốt nghiệp 3](#_Toc452055121)

[Lời cảm ơn 4](#_Toc452055122)

[Mục lục 5](#_Toc452055123)

[Danh mục các hình vẽ 6](#_Toc452055124)

[**Phần 1: Đặt vấn đề và định hướng giải pháp** 7](#_Toc452055125)

[**I. Lý do chọn đề tài** 7](#_Toc452055126)

[**II. Nhiệm vụ của đồ án** 8](#_Toc452055127)

[**III. Giới thiệu về cocos2d-x** 8](#_Toc452055128)

[**Phần 2: Các kết quả đạt được** 9](#_Toc452055129)

[**I. Phát triển framework test cho cocos2d-x** 9](#_Toc452055130)

[**1. Tại sao cần phát triển một framework test?** 9](#_Toc452055131)

[**2. Thiết kế framework test** 10](#_Toc452055132)

[**3. Hướng dẫn sử dụng và ví dụ minh họa** 13](#_Toc452055133)

[**4. Kết luận** 15](#_Toc452055134)

[**II. Phát triển một sản phẩm game thực tế trên nền tảng cocos2d-x** 17](#_Toc452055135)

[**1. Mô tả game** 17](#_Toc452055136)

[**2. Phân tích thiết kế** 22](#_Toc452055137)

[**3. Kết quả** 29](#_Toc452055138)

[**III. Nghiên cứu và đề xuất giải pháp xây dựng cổng game di động** 30](#_Toc452055139)

[**1. Khái niệm cổng game di động** 30](#_Toc452055140)

[**2. Mô tả bài toán** 30](#_Toc452055141)

[**3. Hướng giải quyết** 32](#_Toc452055142)

[**4. Đề xuất giải pháp deploy game trên portal** 32](#_Toc452055143)

[**5. Cài đặt thử nghiệm** 35](#_Toc452055144)

[**Phần 3: Kết luận** 42](#_Toc452055145)

[**I. Tổng kết kết quả của đồ án** 42](#_Toc452055146)

[**II. Hướng phát triển trong tương lai** 42](#_Toc452055147)

[Tài liệu tham khảo 44](#_Toc452055148)

Danh mục các hình vẽ

[Ảnh 1- Các nhiệm vụ của đồ án 8](#_Toc452094316)

[Ảnh 2- Unit test - lý tưởng và thực tế 9](#_Toc452094317)

[Ảnh 3- Các khái niệm cơ sở cho framework test 11](#_Toc452094318)

[Ảnh 4- Luồng hoạt động cơ bản của framework test 12](#_Toc452094319)

[Ảnh 5 - Cài đặt gui ctest 13](#_Toc452094320)

[Ảnh 6 - Ví dụ unit test cho lớp TimeBar 14](#_Toc452094321)

[Ảnh 7- Game tham khảo - Candy Crush Saga 17](#_Toc452094322)

[Ảnh 8 - Bảng chứa item 18](#_Toc452094323)

[Ảnh 9- Màn hình Lobby 19](#_Toc452094324)

[Ảnh 10 - Màn hình chơi game 20](#_Toc452094325)

[Ảnh 11 - Màn hình kết quả, kỷ lục 22](#_Toc452094326)

[Ảnh 12 - Màn hình kết quả bình thường 23](#_Toc452094327)

[Ảnh 13 - Tổng quan về các lớp trong game xếp hình 25](#_Toc452094328)

[Ảnh 14 - Ánh xạ từ lớp phân tích sang lớp thiết kế của ItemBoard 27](#_Toc452094329)

[Ảnh 15 - Lớp thiết kế của TimeBar 28](#_Toc452094330)

[Ảnh 16 - Phân phối một game được phát triển trên nền tảng cocos2d-x 32](#_Toc452094331)

[Ảnh 17- Luồng hoạt động cơ bản của phân phối một game qua cổng game di động 33](#_Toc452094332)

[Ảnh 18 – Biểu đồ hoạt động của usecase xem danh sách game 36](#_Toc452094333)

[Ảnh 19 - Biểu đồ hoạt động của usecase chơi game 37](#_Toc452094334)

[Ảnh 20 - Biểu đồ hoạt động của usecase chơi game 38](#_Toc452094335)

[Ảnh 21 - Kiến trúc plugin 39](#_Toc452094336)

[Ảnh 22 - Tổng quan về các lớp trong hệ thống 40](#_Toc452094337)

Phần 1: Đặt vấn đề và định hướng giải pháp

**I. Lý do chọn đề tài**

Game mobile hiện được coi là một trong những lĩnh vực phát triển mạnh mẽ nhất ở thị trường số hiện nay. Tính trên toàn bộ Đông Nam Á nói chung, dự kiến trong 3 năm từ 2015 đến 2018, quy mô của thị trường game sẽ tăng 2 lần về doanh thu, còn riêng với game mobile thì được dự đoán sẽ tăng tới 3 lần. Ở Việt Nam, vào năm 2014, phân khúc mobile game chiếm thị phần khá nhỏ, chỉ khoảng 18% doanh thu trong toàn bộ thị trường game. Nhưng dự kiến đến năm 2017, mobile game sẽ trở thành phân khúc thị trường có doanh thu lớn nhất, đạt 160 triệu đô la, chiếm hơn 50% doanh thu của toàn bộ thị trường game. [1] Trên thực tế, nhiều nhà phát triển game hiện nay cũng chuyển hướng đầu tư mạnh sang mảng game mobile, thay thế cho các game trên nền tảng web như trước kia.

Trước đây, trong thời kỳ phát triển của web game, những nhà phát triển lớn đều xây dựng cho mình một cổng game trên nền web. Ví dụ VNG có cổng game 360 và cổng game ZingPlay. VTC Game có cổng game Phát Lộc. Các cổng game này là nơi tập trung rất nhiều tựa game do nhà sản xuất đó phát triển. Chúng đem lại những lợi ích cho các nhà phát triển game. Nhưng rồi thời kỳ hưng thịnh của webgame đã qua. Giờ đây, khi các game mobile phát triển mạnh mẽ, nhiều nhà phát triển game lại dự định xây dựng một cổng game di động.

Cổng game di động là một ứng dụng di động, ứng dụng này đóng vai trò như một cửa ngõ mà từ đó, người chơi có thể trải nghiệm nhiều game khác nhau của một hoặc nhiều nhà phát triển. Có một ý tưởng khá hay cho việc xây dựng cổng game di động đó là, các game di động sẽ được đóng gói lại, theo một định dạng nào đó. Thông qua cổng game di động, người chơi có thể tải về các game này từ server. Sau đó chơi game trực tiếp trên nền ứng dụng mà không cần cài đặt game như một ứng dụng độc lập trên điện thoại.

Nếu cách làm này khả thi, nó sẽ đem lại nhiều lợi ích cho các nhà phát triển game. Một số lợi ích có thể kể ra là:

* Tạo ra một kênh phân phối riêng, giúp game của nhà phát triển đến tay người dùng nhanh hơn, việc cập nhật phiên bản diễn ra nhanh chóng, dễ dàng hơn, không cần phải thông qua khâu kiểm duyệt của các kho ứng dụng.
* Một kênh marketing hiệu quả. Bất cứ một game mới ra nào cũng có thể ngay lập tức được giới thiệu đến cộng đồng người chơi game của nhà phát triển đó theo một cách hiệu quả nhất với chi phí gần như là bằng 0.
* Có khả năng tích hợp vào cổng game nhiều tính năng đa dạng, phong phú như tin tức, diễn đàn, góp phần làm tăng trải nghiệm người dùng.

Xuất phát từ nhu cầu thực tế đó, đồ án này tiến hành tìm hiểu về cocos2d-x, một công nghệ làm game được nhiều nhà phát triển game Việt Nam lựa chọn. Sau khi có những hiểu biết nhất định về việc phát triển một game trên cocos2d-x, đồ án sẽ tiến hành nghiên cứu, đề xuất giải pháp xây dựng cổng game di động trên nền tảng cocos2d-x.

**II. Nhiệm vụ của đồ án**

Có 2 nhiệm vụ được xác định ngay từ đầu khi thực hiện dự án đó là:

* **Tìm hiểu về công nghệ làm game cocos2d-x**. Cụ thể:
* Tìm hiểu để có được cái nhìn tổng quan về cocos2d-x, để hiểu được cách thức hoạt động, vận hành của game engine này.
* Tiếp đó, sẽ phát triển một sản phẩm game thực tế trên nền tảng cocos2d-x. Bởi vì phát triển một sản phẩm cụ thể là cách tốt nhất để học về một công nghệ.
* **Nghiên cứu đề xuất giải pháp công nghệ xây dựng cổng game di động trên nền tảng cocos2dx**.
* Nghiên cứu và đưa ra đề xuất về giải pháp.
* Hiện thực hóa giải pháp để chứng minh tính khả thi của nó.

Ngoài 2 nhiệm vụ trên, trong quá trình thực hiện đồ án, một nhu cầu phát sinh là cần xây dựng một framework test để hỗ trợ việc thực hiện unit test. Đây là một công việc khá quan trọng nhưng cocos chưa có một công cụ hỗ trợ cho những việc như vậy. Cho nên, một nhiệm vụ nữa được đặt ra cho đồ án là: **Phát triển một framework test, hỗ trợ cho việc thực hiện các unit test trên nền tảng cocos2d-x**.

Như vậy, chúng ta có 3 nhiệm vụ chính. Trong đó, nhiệm vụ đầu tiên: “tìm hiểu về cocos2dx” có thể được chia thành 2 nhiệm vụ nhỏ hơn. Tổng cộng là 4 nhiệm vụ và thứ tự thực hiện được thể hiện qua sơ đồ bên dưới. Tất nhiên, thứ tự chỉ mang tính tương đối. Trong thực tế không phải cứ thực hiện xong hoàn toàn một nhiệm vụ rồi mới sang nhiệm vụ tiếp theo. Đôi khi một nhiệm vụ có thể khởi đầu tại một thời điểm và thực hiện kéo dài trong suốt thời gian thực hiện đồ án.



Ảnh 1- Các nhiệm vụ của đồ án

Phần 2: Các kết quả đạt được

Trong phần này, ta sẽ trình bày chi tiết các kết quả của đồ án. Do đã trình bày sơ lược về công nghệ làm game cocos2d-x ở phần 1 rồi nên phần này ta sẽ trình bày về các nội dung sau:

* Phát triển một framework hỗ trợ việc kiểm thử đơn vị cho cocos2d-x (gọi tắt là framework test).
* Phát triển một sản phẩm game thực tế trên nền tảng cocos2d-x.
* Nghiên cứu và thử nghiệm công nghệ làm một cổng game di động dựa trên nền tảng cocos2d-x

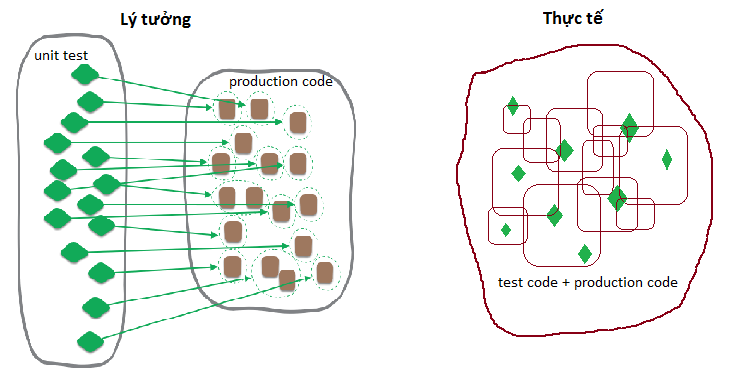
**I. Phát triển framework test cho cocos2d-x**

**1. Tại sao cần phát triển một framework test?**

Như chúng ta đã biết, kiểm thử là một công việc quan trọng để đảm bảo chất lượng sản phẩm. Unit test là một loại kiểm thử được thực hiện trên từng đơn vị mã nguồn riêng lẻ ví dụ như các lớp, các hàm...(gọi chung là các module). Việc unit test cho một module được thực hiện bởi chính developer viết ra module đó, và thường là ngay sau khi module được viết xong. Unit test đem lại rất nhiều lợi ích. Lợi ích lớn nhất là giúp chúng ta phát hiện lỗi từ rất sớm. Mà càng phát hiện lỗi sớm, thì lại càng tiết kiệm chi phí.

Đáng tiếc là cocos2d-x chưa hỗ trợ cho ta một công cụ để thực hiện unit test một cách bài bản, chuyên nghiệp. Có lẽ cũng bởi vì hầu hết các đối tượng cần test trong game là các đối tượng liên quan đến đồ họa, khó mà unit test tự động được. Thành ra việc unit test, nếu có, đều được thực hiện thủ công.

Thực ra việc thực hiện unit test thủ công cũng không phải là vấn đề. Vấn đề là chúng ta thực hiện chúng một cách phi chính thức. Chẳng hạn như, vừa viết xong một lớp nào đấy, chúng ta sẽ kiểm tra luôn, bằng cách viết thêm một số đoạn code để test vào ngay mã nguồn chính, xem nó có “chạy được” hay không. Rồi sau đó nếu chạy được thì comment cái đoạn code để test vừa viết lại. Hình vẽ dưới đây minh họa cho thực tế này:



Ảnh 2- Unit test - lý tưởng và thực tế

Cách làm này được nhiều người cho là nhanh, gọn, nhẹ, tiết kiệm thời gian. Nhưng bản chất là sự thiếu chuyên nghiệp, đánh giá thấp tầm quan trọng của unit test. Hơn nữa lại tạo ra sự lẫn lộn giữa test code với production code và đem lại nhiều rắc rồi khác về lâu dài, bởi lẽ:

* Việc test một cách phi chính thức như vậy, chỉ đảm bảo cái module chúng ta vừa test “chạy được” là được. Như vậy, không có gì đảm bảo là module đó đã được cài đặt đúng, không có lỗi. Nhỡ trong trường hợp này chạy đúng, nhưng trong trường hợp khác chạy sai thì sao? Trên nền tảng này chạy tốt nhưng trên nền tảng khác lỗi thì sao? Thành ra, nếu cứ lấy cái tiêu chuẩn tối thiểu là “chạy được” để áp dụng thì chẳng phải giống các sản phẩm “made in china” một thời được phương Tây đánh giá là: miễn ráp lại với nhau mà không bị bung ra là đạt chuẩn. Sử dụng tiêu chuẩn “chạy được”, chúng ta sẽ không thể phát hiện lỗi sớm, sản phẩm chúng ta cung cấp cho thị trường sẽ là một sản phẩm đầy bug 🡪 Thời gian fix bug kéo dài 🡪 Chi phí lớn + bòn rút thời gian mà đáng lẽ được dành cho việc phát triển các tính năng mới.
* Các code và resource cho mục đích develope và cho mục đích test sẽ bị lẫn lộn với nhau 🡪 Quản lý khó khăn. Nếu chúng ta muốn loại bỏ những phần liên quan đến test ra khỏi sản phẩm release, hay muốn chạy lại một đoạn code test nào đó, sẽ khá rắc rối và dễ xảy ra các sai sót, nhầm lẫn.
* Khi viết unit test một cách chính thức, thì những mã nguồn test đó sẽ chính là một loại tài liệu rất hữu ích. Các lập trình viên khác có thể nhìn vào và biết được cách sử dụng của từng module. Còn nếu viết một cách phi chính thức như trên thì sẽ chẳng có tài liệu hữu ích nào hết. Nếu muốn biết được cách sử dụng của module đó thì có 2 cách:
* Hoặc là ngồi đọc mã nguồn (trong nhiều trường hợp là không có một tài liệu hoặc comment gì cả).
* Hoặc là hỏi trực tiếp tác giả đã viết ra, để nhận được một sự giải thích tận tình, hoặc không.

Dựa trên quan điểm như vậy, cho nên công việc đầu tiên của đồ án này là phát triển một framework test để sử dụng trong quá trình phát triển các sản phẩm trên nền tảng cocos2d-x. Việc phát triển một framework như vậy không chỉ giúp thực hiện unit test một cách bài bản mà còn giúp cho việc kiểm thử các API của cocos2d-x, hoặc thử nghiệm và phát triển các ý tưởng. Đó đều là những công việc được thực hiện rất thường xuyên trong suốt quá trình phát triển sản phẩm.

**2. Thiết kế framework test**

***2.1-Khái niệm cơ sở***

Trước tiên, ta thấy rằng việc unit test cho các đối tượng liên quan đến đồ họa (mà chúng ta đang hướng đến) là test thủ công (manual unit test). Cho nên , sẽ không có sự khác biệt nhiều giữa unit test, hay kiểm thử API, hay các đoạn code để thử nghiệm, phát triển ý tưởng. Ta sẽ khái quát hóa chúng thành một khái niệm chung là: code for test (các đoạn code nhằm mục đích test), viết tắt là **ctest**. Phân biệt với các đoạn code liên quan đến phát triển sản phẩm: code for develope (production), viết tắt là **cdev**.

**Ctest** lại được chia làm 2 loại:

* **Console ctest**: Là loại ctest mà ta có thể quan sát kết quả của nó ở giao diện console, không cẩn đến giao diện đồ họa.

Ví dụ: đoạn code để kiểm tra xem việc download một file có thực hiện thành công hay không.

* **Gui ctest**: Là loại ctest mà ta chỉ có thể quan sát kết quả của nó ở giao diện đồ họa.

Ví dụ đoạn code để kiếm tra xem hàm làm nảy một quả bóng có thực hiện đúng hay không.

**Gui ctest** là loại ctest được quan tâm hơn. Việc hỗ trợ các developer viết các gui ctest dễ dàng, thuận tiện và hiệu quả sẽ là mục tiêu chính khi xây dựng framework test này.



Ảnh 3- Các khái niệm cơ sở cho framework test

***2.2-Ý tưởng cơ bản***

Framework test sẽ hỗ trợ cả console ctest và gui ctest, nhưng tập trung chủ yếu vào gui ctest. Như đã định nghĩa ở trên, console ctest hay gui ctest thực chất chỉ là các đoạn mã. Mục đích của chúng ta là tách biệt các đoạn mã này ra khỏi mã phát triển sản phẩm, hay tách biệt ctest ra khỏi cdev, để giúp cho việc quản lý hiệu quả hơn.

Ý tưởng là:

* Mỗi một ctest đặt vào một file riêng biệt. Các file này nằm trong một thư mục riêng, độc lập hoàn toàn với thư mục chữa cdev.
* Cung cấp 2 chế độ chạy là: chế độ bình thường và chế độ test, được điều khiển bởi một biến, chẳng hạn: enableTest:
* enableTest = false là chế độ bình thường
* enableTest = true là chế độ test.

Khi ở chế độ test, chúng ta có thể lựa chọn và chạy các gui ctest hay console ctest đã được viết trước đó, quan sát kết quả của chúng.



Ảnh 4- Luồng hoạt động cơ bản của framework test

***2.3-Thiết kế cụ thể***

*2.3.1-Console ctest*

Console ctest chẳng qua chỉ là các đoạn mã. Ta có thể đóng gói chúng trong một hàm. Mỗi console ctest đặt trong một file riêng biệt. Các file này sẽ đặt trong một thư mục chung chẳng hạn srctest/testfunc. Để tránh việc bị conflict tên hàm với các hàm có sẵn, ta có thể thêm vào trước nó một tiền tố chẳng hạn “f\_”. Ví dụ:

|  |  |  |
| --- | --- | --- |
| Tên ctest | Link file chứa ctest | Tên hàm tương ứng với ctest |
| example | srctest/testfunc/example.js | f\_example |

Khi cần chạy ctest example, framework sẽ load file srctest/testfunc/example.js vào và thực hiện việc gọi hàm f\_example.

*2.3.2-Gui ctest*

Gui ctest, thực chất là một đoạn mã có trách nhiệm chính là tạo ra một màn hình. Trong màn hình này, ta đặt đối tượng cần test lên, và tiến hành các thao tác kiểm thử trên nó. Trong cocos2d-x, để tạo ra một màn hình, chúng ta sử dụng lớp cc.Scene. Để hỗ trợ việc viết các gui ctest được tốt hơn, ta sẽ tạo ra một lớp TestScene kế thừa từ cc.Scene, cung cấp cho nó thêm một số tính năng như:

* Tạo ra menu thao tác, nhờ đó ta có thể thực hiện nhiều thao tác khác nhau với đối tượng.
* Ghi log trên màn hình, nhờ đó, ta có thể quan sát được trạng thái bên trong của các đối tượng.
* Tạo các thông báo nhanh trên màn hình. Tính năng này cũng có phần hơi giống so với tính năng ghi log trên màn hình. Nhưng khác ở chỗ, các thông báo nhanh thì sẽ hiển thị và biến mất sau một thời gian nhất định. Còn ghi log trên màn hình thì quan sát một trạng thái của một đối tượng theo thời gian thực, và trong suốt thời gian thực hiện kiểm thử.



Ảnh 5 - Cài đặt gui ctest

Hình trên thể hiện sơ đồ cài đặt của TestScene. Trong đó:

* TestScene: Lớp cơ sở để thực hiện các gui ctest, extend từ cc.Scene.
* CustomTestScene: Một gui ctest cụ thể, được extend từ TestScene.
* MenuTest: Một layer, có trách nhiệm tạo ra các menu thao tác cho TestScene.
* LogTest: Một layer, có trách nhiệm ghi log, theo dõi trạng thái của các đối tượng và tạo ra các thông báo nhanh trên màn hình.

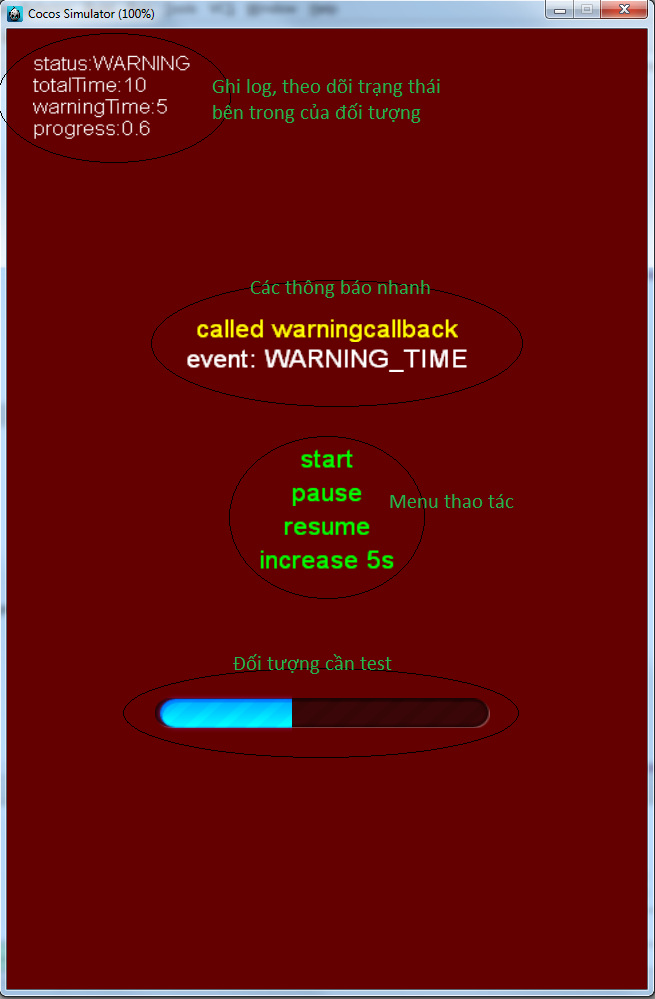
Các API cơ bản mà TestScene cung cấp:

|  |  |
| --- | --- |
| API | Mô tả |
| addCommand(name, callback) | Thêm một lệnh vào menu thao tác.   * name: Tên lệnh * callback: Hàm sẽ được thực hiện khi gọi lệnh |
| addWatch(name, callback) | Theo dõi trạng thái của một đối tượng   * name: Tên hiển thị cho trạng thái. * callback: Hàm trả về giá trị của trạng thái. |
| makeToast(text, setting) | Tạo ra một thông báo trên màn hình và mất đi sau một thời gian. Tương tự như tính năng của Toast trong android. Vì vậy, hàm này có tên là makeToast.   * text: Nội dung thông báo * setting: Một số thông số hiển thị, ví dụ setting.color: màu sắc, setting.size: kích cỡ... |

**3. Ví dụ minh họa về cách sử dụng**

Sử dụng tính năng gui ctest của framework test không chỉ cho phép ta kiểm tra được sự hoạt động của các API, mà còn cho phép ta quan sát được các trạng thái bên trong của các đối tượng, nhờ đó phát hiện ra các sai sót tốt hơn.

Dưới đây là một ví dụ về sử dụng TestScene để thực hiện unit test cho lớp TimeBar:



Ảnh 6 - Ví dụ unit test cho lớp TimeBar

Trong ví dụ này, ta thực hiện test một số hàm của lớp TimeBar:

* start
* pause
* resume
* increaseTime

Để kiểm thử tính đúng đắn của việc cài đặt các hàm trên, ngoài việc quan sát đối tượng bên ngoài, chúng ta còn quan sát cả trạng thái bên trong. Bên cạnh việc kiểm tra tính đúng đắn về mặt logic, chúng ta còn có thể kiểm tra được các yêu cầu liên quan đến hiển thị.

Một lợi ích rất đáng kể cho việc unit test theo kiểu này đó là: trên cơ sở các thao tác cơ bản được cung cấp ở menu thao tác như trên, chúng ta có thể thực hiện rất rất nhiều các test case khác nhau. Ngoài một số test case thông thường kiểu như Start🡪Pause🡪Resume, chúng ta còn có thể thực hiện rất nhiều các test case khác như:

* Thực hiện start liên tục 2 lần.
* Thực hiện pause liên tục 2 lần.
* Chưa start mà thực hiện pause luôn.
* Chưa start mà thực hiện resume luôn.
* Chưa pause nhưng vẫn thực hiện resume.
* Thực hiện increaseTime khi đang pause.
* Thực hiện increaseTime nhiều lần, vượt quá cả thời gian tối đa.
* Đã hết thời gian rồi, nhưng vẫn thực hiện increaseTime.
* Thực hiện pause, điều chỉnh đồng hồ của hệ thống chậm lại vài phút, rồi thực hiện resume.
* ....

Thực tế khi thực hiện unit test cho TimeBar lần đầu tiên, hầu hết các test case được liệt kê ở trên đều fail. Thật ra đây là một điều bình thường, bởi khi lập trình ta thường không nghĩ đến những tình huống như vậy. Tuy nhiên, thực tế sử dụng, khi TimeBar được ghép vào với các module khác, và khi sản phẩm được chạy thực tế, những tình huống như vậy có thể xảy ra. Vì vậy, thật chủ quan và nguy hiểm khi chúng ta luôn cho rằng mình đã thực hiện mọi thứ đúng đắn ngay từ đầu, không bao giờ thực hiện kiểm tra lại các công việc mình đã làm một cách bài bản, có hệ thống.

Với những bug được phát hiện, ngoài những lỗi về lập trình, chúng ta còn phát hiện ra cả những lỗi liên quan đến đặc tả và thiết kế. Có những lỗi chỉ liên quan đến TimeBar, nhưng có những lỗi lại ảnh hưởng đến toàn bộ hệ thống. Ví dụ lỗi về cách tính thời gian, chúng ta đã tham khảo và tin tưởng hoàn toàn thông tin thời gian từ client. Đó là một nguồn thông tin không thực sự đảm bảo. Người chơi game có thể dễ dàng hack bằng cách điều chỉnh đồng hồ hệ thống.

**4. Kết luận**

Từ ví dụ trên, chúng ta có thể thấy, mặc dù framework test được thiết kế khá đơn giản, nhưng hiệu quả mang lại rất lớn. Nhờ sử dụng framework test này, chúng ta có thể thực hiện rất nhiều test case phức tạp khác nhau trên đối tượng cần test, phát huy được ưu điểm của việc kiểm thử thủ công. Nhiều lỗi về lập trình và thậm chí cả lỗi về đặc tả, thiết kế cũng được phát hiện từ sớm. Ngay khi chưa tích hợp các module lại với nhau. Điều này góp phần nâng cao chất lượng của sản phẩm, giảm thiểu chi phí cho việc fix bug sau này.

Tuy nhiên, cũng cần nói thêm rằng, việc có áp dụng được vào thực tế hay không còn phụ thuộc vào một yếu tố rất quan trọng khác là: việc thiết kế phần mềm có cho phép thực hiện unit test dễ dàng hay không. Nếu các lớp không tách biệt nhau mà dính chặt với nhau thành một khối, sử dụng lớp này thì yêu cầu phải khởi tạo hàng đống đối tượng của các lớp khác thì unit test sẽ rất khó mà thực hiện được. Có một số khái niệm và nguyên tắc thiết kế giúp chúng ta giải quyết điều đó. Chẳng hạn như: khái niệm decoupling, nguyên tắc dependency inversion...Tất nhiên, để hiểu và áp dụng được những khái niệm, những nguyên tắc đó vào thực tế, cần có sự đầu tư nhất định về thời gian và công sức.

**II. Phát triển một sản phẩm game thực tế trên nền tảng cocos2d-x**

Vào những năm 350 trước công nguyên, Aristoteles – một nhà triết học, bác học, người được coi là một trong những cột trụ văn minh của Hy Lạp cổ đại đã từng viết rằng:

*“For the things we have to learn before we can do them, we learn by doing them”* [1]

Đại ý của câu này là: Có một số thứ mà chỉ học không thôi thì chưa đủ, học chỉ là để ta có thể bắt đầu làm. Và khi đi vào quá trình làm việc thực tế, đó mới là lúc ta đang thực sự học về nó.

Việc tìm hiểu một công nghệ cũng là một thứ như vậy. Chúng ta không thể học bất cứ một công nghệ nào chỉ bằng cách đọc bài viết giới thiệu về công nghệ, các tài liệu hướng dẫn hay các API reference của nó. Chúng ta chỉ có thể thực sự học được một công nghệ bằng cách phát triển một sản phẩm thực tế dựa trên công nghệ đó. Và để thực sự hiểu về cocos2d-x, không có cách nào khác là sử dụng nó để phát triển một sản phẩm game.

**1. Mô tả game**

Trong phần này, ta chỉ mô tả sơ lược về luật chơi, những tính năng cơ bản, và đưa ra một số thiết kế giao diện nhằm mục đích phác họa những hình dung ban đầu về game.

Game tham khảo ý tưởng từ loạt game cùng thể loại nổi tiếng: candy crush saga, candy crush soda saga...



Ảnh 7- Game tham khảo - Candy Crush Saga

Game thuộc thể loại ăn 3, có tính thời gian. Với không gian chơi là một bảng chứa item, kích thước 7\*7. Có tất cả 7 loại item, và một item đặc biệt tên lửa. Các item thông thường có thể ở trong trạng thái đặc biệt gọi là trạng thái cường lực. Ngoài ra còn có một số icon ngẫu nhiên sẽ xuất hiện ngẫu nhiên trên một item nào đó, và biến mất sau một khoảng thời gian nhất định.

***1.1-Sơ lược về luật chơi cho một ván chơi***



Ảnh 8 - Bảng chứa item

* Người chơi sẽ hoán đổi 2 ô cạnh nhau trong một bảng chứa các icon để tạo ra được một hàng hoặc một cột với 3 icon giống nhau nằm liền kề nhau, nhờ đó làm tăng điểm số của mình.
* Ngoài ra trong nếu tạo được các cụm 4 item, 5 item hoặc cụm item chữ L, cụm item chữ T, thì sẽ tạo ra các item đặc biệt với các tính năng đặc biệt tương ứng.

Ăn cụm 4 item: Sinh ra item cường lực. Nếu ăn được item cường lực thì sẽ kích hoạt nổ lan sang các ô xung quanh.

Ăn cụm 5 item, cụm chữ L, cụm chữ T: Sinh ra item laser. Khi kích hoạt item laser (bằng cách nhấn vào nó), thì sẽ nổ một hàng và một cột tương ứng với vị trí của item laser.

* Các icon ngẫu nhiên như: icon thời gian, icon chong chóng sẽ xuất hiện ngẫu nhiên và tồn tại trong một khoảng thời gian nhất định. Nếu như ăn được thì sẽ kích hoạt các tính năng đặc biệt.
* Icon thời gian: Tăng thời gian
* Icon chong chóng: Thổi bay một hàng
* Cách tính điểm: Một item được ăn tương ứng với một điểm.

***1.2-Các màn hình trong game***

Game được thiết kế với 3 màn hình cơ bản: là màn hình lobby, màn hình play và màn hình kết quả.

* Màn hình lobby



Ảnh 9- Màn hình Lobby

Đây là nơi người chơi bắt đầu vào game. Tại màn hình này, chúng ta thấy có 3 tính năng chính:

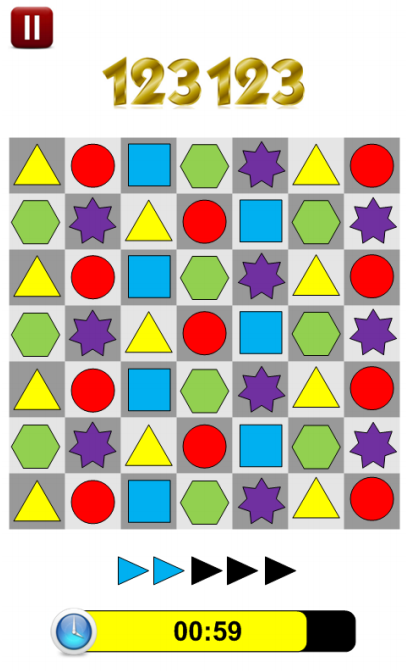
* Bắt đầu chơi game
* Bảng xếp hạng

Đây là bảng xếp hạng của người chơi và những nhân vật giả lập (chẳng hạn: siêu gà, siêu mèo, siêu chó, siêu nhân) với mức điểm tăng dần. Nó khuyến khích người chơi đạt điểm số cao hơn để đánh bại các nhân vật giả lập này.

* Thanh trái tim

Chứa 5 trái tim, mỗi trái tim tương ứng với một lần chơi. Có nghĩa là mỗi lần người chơi bắt đầu chơi game thì sẽ tiêu tốn một trái tim. Mỗi trái tim mất đi thì cần có thời gian để hồi phục lại. Nếu không còn trái tim, thì người chơi sẽ phải đợi, đến khi trái tim hồi phục thì mới được chơi tiếp.

* Màn hình play



Ảnh 10 - Màn hình chơi game

Trong màn hình play này, ta thấy có 5 thành phần chính là:

* Bảng chứa icon: Kích thước 7\*7, là nơi người chơi thao tác để chơi game.
* Thanh thời gian: Đếm ngược thời gian 60 giây cho mỗi ván chơi.
* Thanh fever: Với mỗi lần người chơi ăn được một cụm item, thanh fever sẽ tăng lên một mức. Sau một khoảng thời gian nhất định (khoảng 1.5 giây chẳng hạn), nếu người chơi không ăn được cụm item nào thì thanh fever sẽ trở về mức 0. Khi thanh fever đạt đến mức 5, game sẽ chuyển sang trạng thái fever, màn hình sáng nhấp nháy, duy trì khoảng 5 giây. Trong thời gian này, mỗi cụm item mà người chơi ăn được sẽ được tính gấp đôi.
* Thanh điểm số: Hiển thị điểm số của người chơi.
* Nút tạm dừng: Tạm dừng game.
* Màn hình kết quả

Màn hình này sẽ hiển thị sau khi người chơi kết thúc ván chơi. Tùy thuộc vào việc người chơi có phá kỷ lục hay không, mà ta có 2 kiểu hiển thị khác nhau.

* Nếu người chơi phá kỷ lục



Ảnh 11 - Màn hình kết quả, kỷ lục

* Nếu người chơi không phá kỷ lục



Ảnh 12 - Màn hình kết quả bình thường

**2. Phân tích thiết kế**

Trong phần này, ta sẽ trình bày các nội dung liên quan đến phân tích thiết kế phần mềm, cụ thể là:

* Ý tưởng thiết kế cơ bản
* Tổng quan các lớp trong hệ thống
* Một số lớp quan trọng

***2.1-Ý tưởng thiết kế cơ bản***

Ý tưởng cơ bản là Các thành phần trên giao diện là sự đóng gói của cả chức năng và giao diện

**Sự đóng gói**

Một số thành phần mà ta thấy trên giao diện như điểm số, thanh fever, thanh thời gian, ... được ánh xạ thành các lớp như ScoreBar, FeverBar, TimeBar. Điều cần nhấn mạnh ở đây là: các lớp này không chỉ có chức năng hiển thị mà còn đảm nhận luôn cả phần quản lý các chức năng, nghiệp vụ logic liên quan đến phần hiển thị đó. Vì vậy, ta nói rằng các lớp này là sự đóng gói của cả chức năng và giao diện.

**Nguyên nhân**

* Thứ nhất là do sự liên quan chặt chẽ giữa logic và hiển thị của các thành phần này (tính cohesion). Cho nên việc tách riêng logic và hiển thị có thể gây khó khăn cho việc phát triển.
* Thứ hai là tính lỏng lẻo, có thể tách rời nhau của các chức năng (tính decoupling). Có nghĩa là các chức năng không cần thiết (và không nên) tập trung vào một chỗ vì thực ra chúng chẳng liên quan đến nhau nhiều lắm. Ví dụ:
* Chức năng quản lý thời gian và quản lý điểm số chẳng liên quan gì đến nhau.
* Chức năng quản lý thời gian và quản lý fever cũng vậy.
* Chức năng chơi game (được hiểu là tương tác và chơi trên GameBoard) và quản lý điểm số thì có liên quan đến nhau nhưng cũng không quá chặt chẽ. Người ta hoàn toàn có thể chơi game mà không cần tính điểm số. Có nghĩa là chỉ điểm số phụ thuộc vào chơi game chứ chơi game không phụ thuộc vào điểm số.

**Lợi ích**

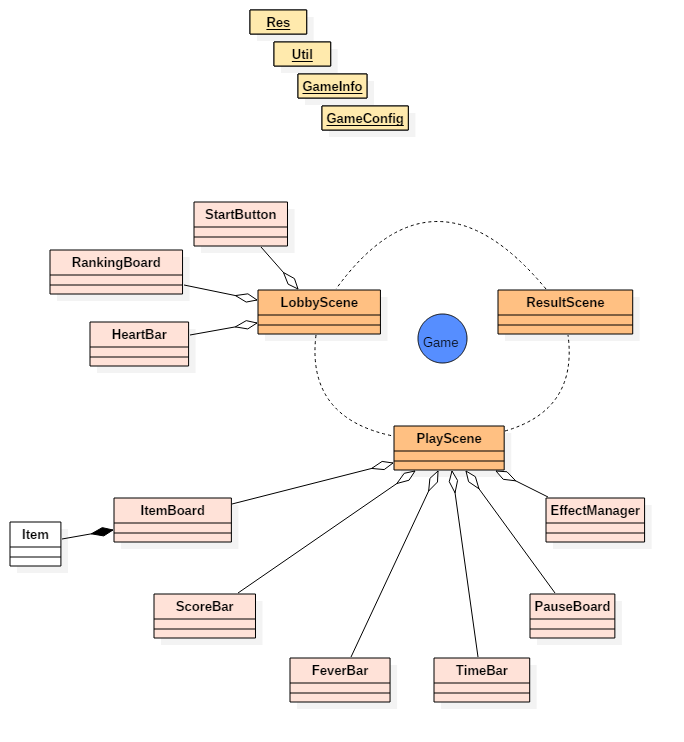
* Logic được phân tách rõ ràng, dễ quản lý
* Có thể tái sử dụng một số đối tượng cho các ứng dụng khác (ví dụ: TimeBar).

**Một cách nhìn khác rõ ràng hơn về sự đóng gói này**

* Mỗi thành phần trên giao diện sẽ được khái niệm hóa, đóng gói cả phần hiển thị và phần chức năng thành một phần tử giao diện mới hoàn chỉnh. Giống như trước đây, chúng ta có Button, có Label, có Checkbox, ProgressBar, thì bây giờ chúng ta có TimeBar, có FeverBar, có ScoreBar...
* Các phần tử giao diện mới này được phân tích, thiết kế, phát triển và kiểm thử độc lập nhau.
* Và chúng có thể được tái sử dụng trong nhiều ngữ cảnh khác nhau, nhiều ứng dụng khác nhau.

***2.2-Tổng quan về các lớp trong hệ thống***

Trong phần này, ta sẽ trình bày tổng quan về các lớp trong hệ thống. Các lớp được đưa ra ở đây chỉ là sự phân tích ban đầu, có thể chưa đầy đủ và hoàn toàn chính xác. Trong quá trình thiết kế chi tiết và cài đặt, một số lớp có thể được bổ sung thêm, một số lớp khác có thể được tách ra thành nhiều lớp nhỏ hơn. Tuy nhiên, ý tưởng thiết kế về cơ bản là không thay đổi.



Ảnh 13 - Tổng quan về các lớp trong game xếp hình

|  |  |  |
| --- | --- | --- |
| **Tên lớp** | **Mô tả** | **Các sự kiện mà nó phát sinh** |
| LobbyScene | Biểu diễn cho màn hình lobby | 0 |
| PlayScene | Biểu diễn cho màn hình bàn chơi (màn hình chính của game) | 0 |
| ResultScene | Biểu diễn cho màn hình thông báo kết quả khi người chơi chơi xong | 0 |
| StartButton | Biểu diễn cho nút bắt đầu của màn hình lobby, cần được đầu tư kỹ lưỡng về giao diện và hiệu ứng. | 0 |
| RankingBoard | Biểu diễn cho bảng xếp hạng ở màn hình lobby | 0 |
| HeartBar | Biểu diễn cho thanh chứa trái tim ở màn hình lobby | 0 |
| BackGroundLayer | Biểu diễn cho màn hình nền của GamePlayScene | 0 |
| UILayer | Là layer chính trong GamePlayScene, chứa tất cả các thành phần giao diện quan trọng của game. Và đồng thời có thể đóng vai trò như một controller, điều khiển sự phối hợp của các thành phần đó. | 0 |
| **ItemBoard** | Biểu diễn cho bảng chứa icon ở màn hình bàn chơi, nơi người chơi tương tác với game. Đây là lớp có logic phức tạp, **cần được phân tích thiết kế kỹ lưỡng**. | EatItems |
| Item | Biểu diễn cho một item cấu thành nên itemboard. Quản lý cả hiển thị, hiệu ứng (rơi) và logic của item đó. | 0 |
| ScoreBar | Biểu diễn cho thanh điểm số ở màn hình bàn chơi. Quản lý luôn logic tính điểm. | 0 |
| FeverBar | Biểu diễn cho thanh fever ở màn hình bàn chơi. Quản lý luôn logic của về chuyển trạng thái fever. | StartFever  FinishFever |
| TimeBar | Biểu diễn cho thanh thời gian ở màn hình bàn chơi. Quản lý logic về thời gian. | WarningTime  Timeout |
| PauseBoard | Biểu diễn cho màn hình tạm dừng khi người chơi nhấn nút tạm dừng. | PauseGame  ResumeGame |
| EffectManager | Có trách nhiệm quản lý các hiệu ứng trong khi chơi game. | 0 |
| GameInfo | Chứa các thông tin chung của game dùng cho tất cả các lớp như: tên người chơi, kỷ lục cũ của người chơi, danh sách các đối thủ, ... | 0 |
| GameConfig | Chứa các thông tin cấu hình cho game, ví dụ: thông tin về thời gian chơi của một ván chơi, bonus time, thời gian duy trì thanh fever, thời gian xuất hiện và tồn tại của các item ngẫu nhiên, điểm số khi ăn icon thường, điểm số khi ăn icon trong fever mode, điểm số khi ăn icon ở cú chót, thời gian phục hồi trái tim, .... | 0 |
| Res | Chứa url của các tài nguyên trong game như hình ảnh, âm thanh, hiệu ứng... | 0 |
| Util | Chứa các hàm tiện ích phục vụ cho hoạt động của các lớp khác. | 0 |

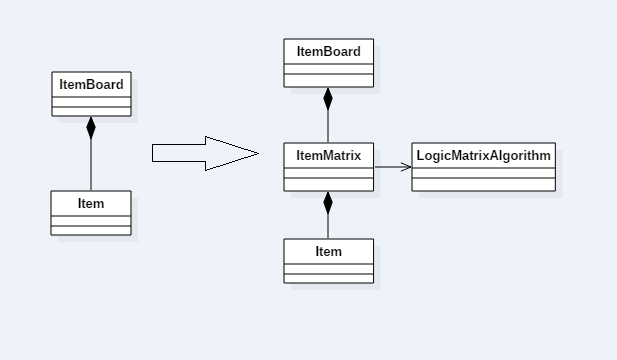
***2.3-Một số lớp quan trọng***

Trong phần này ta sẽ trình bày cách thức ánh xạ từ lớp phân tích thành lớp thiết kế của hai lớp phân tích là:

* Lớp ItemBoard: Đại diện cho bảng chứa item nơi người chơi tương tác và chơi game.
* Lớp TimeBar: Đại diện cho thanh thời gian đếm ngược. Lớp này tiêu biểu cho các phần tử giao diện đồ họa như: FeverBar, HeartBar, ScoreBar.

Lớp ItemBoard

Lớp ItemBoard đại diện cho bảng chứa item nơi người chơi tương tác và chơi game. Khi phân tích, ta chỉ có 2 lớp liên quan đến bảng chơi là ItemBoard và Item. Nhưng khi thiết kế chi tiết và lập trình do nhiều nguyên nhân khác nhau nên ta phải bổ sung thêm một số lớp mới: ItemMatrix và LogicMatrixAlgorithm. Việc tách một lớp thành nhiều lớp, san sẻ bớt trách nhiệm của lớp này cho các lớp khác, hay gộp nhiều lớp thành một lớp xảy ra rất thường xuyên trong quá trình phân tích, thiết kế hướng đối tượng.



Ảnh 14 - Ánh xạ từ lớp phân tích sang lớp thiết kế của ItemBoard

Cụ thể trong trường hợp này, ban đầu khi phân tích, ta có 2 lớp:

* Item

Đại diện cho một item cụ thể trên bảng chơi, lớp này có trách nhiệm:

* Quản lý logic dữ liệu của một item: Nó thuộc loại item gì, có phải cường lực không, có chứa icon ngẫu nhiên gì
* Quản lý logic trạng thái của item: Đang ổn định trên màn hình, đang rơi, đang swap, đã bị hủy...
* Thực hiện các hiệu ứng: Rơi xuống một vị trí nào đó trên màn hình, chuyển thành item laser, chuyển thành item cường lực, nổ, ...
* ItemBoard:

Đại diện cho bảng chứa item, lớp này sẽ chứa một mảng 7\*7 item, và có trách nhiệm

* Nhận tương tác từ người chơi
* Quản lý dữ liệu của bảng chơi
* Điều khiển luồng logic của toàn bộ bảng chơi.

Điều chỉnh khi thiết kế chi tiết và lập trình:

* Thêm lớp ItemMatrix

Điều này xuất phát từ ý tưởng: đưa vào một mảng số nguyên 7\*7 tương ứng với mảng 7\*7 các item, bởi vì các thuật toán tìm cụm ăn 3, ăn 4, ăn 5...trên mảng số nguyên sẽ dễ cài đặt hơn, và không phụ thuộc cài đặt chi tiết của các item. Tuy nhiên do cần phải có một sự đồng bộ giữa mảng số nguyên và mảng các item cho nên, nên cần tạo ra một lớp ItemMatrix để đóng gói cả 2 mảng này lại. Bất cứ theo thao tác nào liên quan đến việc thay đổi vị trí của item trong mảng, đều phải tác động thông qua lớp này. Như vậy, chức năng chính của lớp này là: Quản lý dữ liệu của bảng chơi.

* Thêm lớp LogicMatrixAlgorithm

Do các thuật toán tìm cụm ăn khá phức tạp nên ta tách chúng ra, đưa vào lớp LogicMatrixAlgorithm. Lớp này có trách nhiệm:

* Thực hiện thuật toán tìm kiếm cụm ăn dựa trên đầu vào là ma trận số nguyên.
* Cung cấp các hàm hỗ trợ duyệt qua các cụm ăn dựa trên kết quả nó trả về.
* Thay đổi vai trò của lớp ItemBoard

Lớp ItemBoard sẽ không quản lý dữ liệu của bảng chơi nữa mà chỉ:

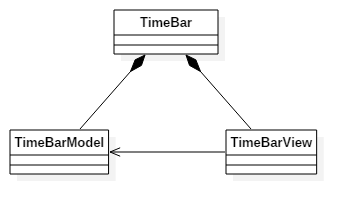
* Nhận và xử lý tương tác từ người chơi
* Điều khiển luồng logic của bảng chơi

Lớp TimeBar

Lớp TimeBar đại diện cho thanh thời gian đếm ngược. Do yêu cầu tách biệt phần logic và phần hiển thị nên ta phân bổ trách nhiệm của TimeBar sang 2 lớp khác là:

* TimeBarModel: Chịu trách nhiệm quản lý dữ liệu và logic của thanh thời gian.
* TimeBarView: Chịu trách nhiệm việc hiển thị của thanh thời gian.

Như vậy, trách nhiệm của TimeBar sẽ là điều phối hoạt động của 2 lớp trên và cung cấp giao diện lập trình cho các lớp khác sử dụng.



Ảnh 15 - Lớp thiết kế của TimeBar

TimeBarModel quản lý dữ liệu và logic, nên các thuộc tính của nó hoàn toàn là các con số thuần túy, không chút gì liên quan đến giao diện:

*/\*\*  
 \* Thời điểm bắt đầu tính giờ  
 \*/***this**.**startTime** = 0;

*/\*\*  
 \* Thời điểm của lần pause gần đây nhất  
 \*/***var** timePauseMoment;  
  
*/\*\*  
 \* Thời điểm của lần resume gần đây nhất  
 \*/***var** timeResumeMoment;  
  
*/\*\*  
 \* Tổng thời gian nằm trong trạng thái pause từ khi bắt đầu khởi động đồng hồ cho đến thời điểm hiện tại  
 \*/***var** totalTimePause = 0;  
  
*/\*\*  
 \* Thời gian được thưởng  
 \*/***var** totalBonusTime = 0;  
  
*/\*\*  
 \* Tổng thời gian đếm ngược  
 \*/***this**.**totalTime** = params.**totalTime** || DEFAULT\_TOTAL\_TIME;  
  
*/\*\*  
 \* Thời gian sẽ phát sinh sự kiện cảnh báo sắp hết giờ  
 \*/***this**.**warningTime** = params.**warningTime** || DEFAULT\_WARNING\_TIME;  
  
*/\*\*  
 \* Tiến độ thời gian hiện tại, xuất phát điểm là 0, tiến dần đến 1.  
 \*/***this**.**currentProgress** = 0;

Ngược lại với TimeBarView, những thuộc tính quan trọng đều là các thuộc tính liên quan đến hiển thị trên giao diện đồ họa:

*/\*\*   
 \* Một đối tượng giao diện kiểu cc.ProgressBar  
 \*/***var** loadingBar;  
*/\*\*  
 \* Ảnh nền của thanh thời gian  
 \*/***var** background;  
  
*/\*\*  
 \** ***@see*** *TimeBarModel  
 \*/***this**.**model** = model;

Lớp TimeBar điều phối 2 lớp trên. Ví dụ về phương thức increaseTime: tăng thời gian của TimeBar. Lớp TimeBar gọi đến increaseTime của cả model và view.

increaseTime:**function**(t) {

**if**((**this**.**state** != ***TimeBarState***.**RUNNING**)

&&(**this**.**state** != ***TimeBarState***.**WARNING**))

**return**;

**this**.**model**.increaseTime(t);  
 **this**.**view**.increaseTime(t);

**if**(**this**.**model**.timeInWarning()) {  
 **this**.**state** = ***TimeBarState***.**WARNING**;  
 } **else** {  
 **this**.**state** = ***TimeBarState***.**RUNNING**;  
 }  
}

Bản thân model và view có những xử lý riêng của nó. Model thì chỉ đơn giản là tăng giá trị của bonusTime thêm một lượng t. Còn view thì phải có những xử lý phức tạp hơn để liên quan đến tạo hiệu ứng tăng thời gian một cách liên tục (làm cho người chơi thấy rõ được một tiến trình thời gian tăng lên dần dần).

Xử lý trong model:

**this**.increaseTime = **function**(t) {  
 totalBonusTime += t;  
};

Xử lý trong view:

**this**.increaseTime = **function**(t) {  
 increasing = **true**;  
 **var** increaseTimeInterval = *setInterval*(**function**(){  
 **var** displayProgress = 1 - loadingBar.getPercent()/100;  
 **if**(displayProgress <= self.**model**.**currentProgress**) {  
 increasing = **false**;  
 *clearInterval*(increaseTimeInterval);  
 } **else** {  
 loadingBar.setPercent(loadingBar.getPercent()+1);  
 }  
 }, 5);  
};

**III. Nghiên cứu và đề xuất giải pháp xây dựng cổng game di động**

**1. Khái niệm cổng game di động**

Cổng game di động là một ứng dụng di động, ứng dụng này đóng vai trò như một cửa ngõ mà từ đó, người chơi có thể trải nghiệm nhiều game khác nhau của một hoặc nhiều nhà phát triển.

Có một ý tưởng khá hay cho việc xây dựng cổng game di động đó là, các game di động sẽ được đóng gói lại, theo một định dạng nào đó. Thông qua cổng game di động, người chơi có thể tải về các game này từ server. Sau đó chơi game trực tiếp trên nền ứng dụng mà không cần cài đặt game như một ứng dụng độc lập trên điện thoại.

Nếu cách làm này khả thi, nó sẽ đem lại nhiều lợi ích cho các nhà phát triển game. Một số lợi ích có thể kể ra là:

* Tạo ra một kênh phân phối riêng, giúp game của nhà phát triển đến tay người dùng nhanh hơn, việc cập nhật phiên bản diễn ra nhanh chóng, dễ dàng hơn, không cần phải thông qua khâu kiểm duyệt của các kho ứng dụng.
* Một kênh marketing hiệu quả. Bất cứ một game mới ra nào cũng có thể ngay lập tức được giới thiệu đến cộng đồng người chơi game của nhà phát triển đó theo một cách hiệu quả nhất với chi phí gần như là bằng 0.
* Có khả năng tích hợp vào cổng game nhiều tính năng đa dạng, phong phú như tin tức, diễn đàn, góp phần làm tăng trải nghiệm người dùng.

Đồ án có mục đích là thử nghiệm để chứng tỏ ý tưởng trên khả thi. Nó giống như một proof-of-concept hơn là một sản phẩm hoàn chỉnh. [3]

**2. Mô tả bài toán**

Trước hết, chúng ta xem xét vấn đề cần giải quyết trong ngữ cảnh rộng hơn. Đó là phân phối một game được phát triển trên nền tảng cocos2d-x. Xuất phát điểm của chúng ta là một project cocos2d-x của một game cụ thể. Đích đến của chúng ta là game đến được với người dùng. Có hai con đường:

* Con đường release game như một ứng dụng độc lập. Gồm 3 bước:
* Build project: Do cocos2d-x hỗ trợ đa nền tảng nên ta có thể build project này để tạo ra file apk cho nền tảng android, hoặc file ipa cho nền tảng iOS, hay file xap cho windows phone...
* Release game: Các file này sau đó đưa lên các kho ứng dụng như google play, app store...
* Download and install: Và được người dùng tải game về điện thoại của họ, cài đặt như một ứng dụng, cuối cùng là chơi game, trải nghiệm sản phẩm
* Con đường release game thông qua cổng game di động. Cũng gồm 3 bước:
* Đóng gói game: Đóng gói các mã thực thi và các tài nguyên trong game như hình ảnh, âm thanh, theo một định dạng quy định sẵn thành một file gọi là game pack.
* Release game: Upload game pack lên server của nhà phát triển.
* Download and deploy: Người dùng sử dụng ứng dụng cổng game di động để tải game về và chơi game trực tiếp trên nền của cổng game di động đó.



Ảnh 16 - Phân phối một game được phát triển trên nền tảng cocos2d-x

Bài toán đặt ra cho chúng ta là: Xây dựng một hệ thống gồm các công cụ, phần mềm cần thiết để release một game được phát triển trên nền tảng cocos2d-x theo con đường thứ 2, tức là thông qua cổng game di động. Và như đã đề cập ở mục 1 của chương này, chúng ta giới hạn phạm vi bài toán ở một proof-of-concept, chứng minh tính khả thi của cách làm này. Chứ không phải tạo ra một sản phẩm thực tế, hoàn thiện, đầy đủ tính năng.

Một cách trực quan, hệ thống mà chúng ta xây dựng nằm trong phần được đóng khung màu đỏ ở hình trên. Đầu vào là một project cocos2d-x của một game cụ thể. Đầu ra là người chơi trải nghiệm được game đó thông qua ứng dụng cổng game di động.

Dựa vào biểu đồ trên, các công việc mà chúng ta cần làm để giải quyết bài toán này là:

* Xây dựng một công cụ hỗ trợ đóng gói game theo định dạng xây dựng sẵn từ một project cocos2d-x.
* Tạo một server cổng game di động: để upload các game pack lên và cung cấp các API cho ứng dụng client cổng game di động.
* Xây dựng ứng dụng client cổng game di động (nếu không nhầm lẫn và để ngắn gọn ta gọi là portal): Download các game pack từ server về và deploy trên nền ứng dụng để người chơi trải nghiệm.

**3. Hướng giải quyết**

Trong bài toán trên, ta thấy luồng hoạt động cơ bản là:

Đóng gói game 🡪 Push game lên server 🡪 Download game về portal 🡪 Deploy game trên portal 🡪 Chạy game.

Nhưng các giai đoạn “push game lên server” và “download game về portal” không quá khó khăn. Push game lên server chẳng qua là upload một file lên server. Download game về portal chẳng qua là download file từ server về mà thôi. Vấn đề mấu chốt nằm ở việc đóng gói game và deploy game trên portal và chạy game.



Ảnh 17- Luồng hoạt động cơ bản của phân phối một game qua cổng game di động

Các bước: đóng gói game, deploy game và run game có liên quan chặt chẽ, không thể tách rời. Cần phải deploy game theo cách nào đó để chạy được game. Và muốn deploy game thế nào thì phải đóng gói game như thế. Cho nên ta gọi chung giải pháp để thực hiện 3 bước trên là: kỹ thuật deploy game trên portal. Đây là một kỹ thuật quan trọng, có tính chất nút thắt của bài toán.

Như vậy, hướng giải quyết của đồ án sẽ là:

* Tìm ra một kỹ thuật deploy game trên portal hợp lý.
* Dựa trên kỹ thuật này, xây dựng các công cụ, phần mềm cho hệ thống: công cụ đóng gói game, server cho cổng game và ứng dụng cổng game di động.

**4. Đề xuất giải pháp deploy game trên portal**

Quá trình chạy một game của cocos2d-x viết bằng javascript khá phức tạp, và trên mỗi nền tảng khác nhau lại có các cài đặt khác nhau. Nhưng về cơ bản ta quan tâm tới một số bước sau:

* Engine chạy file script/jsb\_boot.js (file của hệ thống)
* File script/jsb\_boot.js load vào hệ thống một vài file javascript của framework và thực hiện hàng hàng loạt các thao tác khởi tạo, chuẩn bị môi trường.
* Engine chạy file main.js (file được viết bởi developer)
* File main.js này thực hiện một số khởi tạo của game.
* File main.js gọi đến hàm cc.game.run() (Hàm do game engine cung cấp)
* Hàm này sau đó sẽ thực hiện việc load vào và chạy hàng loạt các file mã nguồn khác dựa theo thông tin về jsList được mô tả trong file project.json
* Và gọi đến hàm cc.game.onStart để chạy game (hàm này được viết bởi developer)

Do cocos2d-x là một game engine mã nguồn mở, nên để hiểu rõ hơn về quá trình chạy một game viết bằng javascript của cocos2d-x, chúng ta có thể đọc trực tiếp mã nguồn viết bằng C++ của game engine này và một số file javascript của game engine.

Dựa trên cơ chế chạy một game viết bằng javascript như mô tả ở trên, có 2 giải pháp được đưa ra để deploy game trên portal.

**Giải pháp 1**: Thay đổi đường dẫn thư mục của game và restart lại máy ảo javascript

**Ý tưởng**:

Chúng ta thấy rằng, việc load các file javascript được thực hiện trên đường dẫn tương đối, nên ta can thiệp vào code C++ để cung cấp thêm một API cho javascript, cho phép từ bên mã nguồn javascript, có thể thay đổi được đường dẫn này. Như vậy, khi chạy game, sẽ thay đổi đường dẫn thư mục đến nơi deploy game cần chạy và restart lại game engine. Khi đó các file javascript được load vào sẽ là file javascript của game cần chạy.

**Ưu điểm:**

* Game deploy trên portal sẽ chạy giống trường hợp được cài như một ứng dụng riêng lẻ, không phát sinh thêm bug. Bởi lẽ, không bị conflict tên hàm, tên biến với portal cũng như với các game khác.
* Hiệu quả sử dụng bộ nhớ tốt: Vì khi restart lại máy ảo javascript, engine sẽ release tất cả các tài nguyên, ví dụ texture cache, sprite frame cache...
* Không ảnh hưởng đến khả năng update resource.

**Nhược điểm:**

* Không quay lại được màn hình cũ.

🡪Hướng giải quyết là: Tìm cách lưu lại trạng thái của portal (tên scene hiện tại, các dữ liệu đã được khởi tạo, ...) trước khi start game và khôi phục lại trạng thái đó sau khi back game.

* Bị hạn chế khả năng giao tiếp giữa game với portal. Cách duy nhất để thực hiện giao tiếp là thông qua file system. Có nghĩa là khi game cần giao tiếp điều gì đó với portal, nó sẽ lưu ra một file. Sau khi người chơi back game, quyền điều khiển trả lại cho portal, portal sẽ đọc file này để biết được thông điệp game gửi đến cho mình. Cách thức giao tiếp này khá phức tạp và có tính chất gián đoạn.

**Giải pháp 2**: Load trực tiếp mã nguồn của game vào portal khi cần chạy game, thực hiện việc gọi hàm để thực hiện việc chuyển scene, start game bình thường.

**Ý tưởng**

Như chúng ta thấy quá trình chạy game như mô tả ở trên có 2 việc chính là:

* Load vào các file javascript được mô tả trong project.json
* Và sau khi mọi thứ đã sẵn sàng thì chạy hàm cc.game.onStart

Như vậy, chúng ta sẽ không restart máy ảo mà load trực tiếp mã nguồn của game vào portal. Sau khi load xong, thực hiện đoạn code được viết trong hàm onStart của game.

**Ưu điểm:**

* Game và portal có thể giao tiếp với nhau dễ dàng, liên tục, không bị gián đoạn.
* Sau khi back game, có thể quay lại màn hình cũ dễ dàng.

**Nhược điểm:**

* Dễ bị conflict tên hàm, tên biến, giữa game với portal và giữa các game với nhau.

🡪Giải pháp: Đóng gói toàn bộ mã nguồn của game vào một closure, giúp tránh được hoàn toàn việc conflict tên hàm tên biến. Đây cũng chính là cách mà hầu hết các thư viện javascript sử dụng. Tuy nhiên có một lưu ý khi sử dụng giải pháp này đó là: tất cả biến (kể cả khai báo lớp trong cocos2d-x) đều cần khai báo với từ khóa var. Vì nếu không khai báo với từ khóa var thì biến sẽ được coi như một biến toàn cục làm cho closure mất tác dụng.

* Nếu giải quyết được vấn đề conflict, thì ta lại gặp phải một vấn đề khác là: Ảnh hưởng đến hiệu năng của việc update resource. Bình thường, chúng ta sửa file trong mã nguồn thì khi update cũng chỉ phải update lại file đó. Còn khi đóng gói toàn bộ code vào một file lớn, thì sửa bất cứ file nào, cũng phải update toàn bộ code. Với cờ tỷ phú, toàn bộ code là khoảng hơn 800kb, khi nén code lại, chỉ còn gần 300kb, nếu tần suất update code thấp thì điều này có thể chấp nhận được.
* Một vấn đề quan trọng nữa là vấn đề về config, cụ thể là:
* Kích thước màn hình thiết kế giữa các game khác nhau
* Chiến lược giải quyết vấn đề đa màn hình giữa các game khác nhau
* Vấn đề liên quan đến đường dẫn, searchPaths, ...

🡪Giải pháp: Lưu lại các thông số cũ của portal và thay thế chúng bằng các thông số mới của game trước khi start. Khi người chơi back game thì lại khôi phục lại các thông số cũ của portal.

* Vấn đề về hiệu năng: Khi chạy nhiều game trên portal (chạy game này, rồi lại thoát ra chạy game khác), nếu quản lý không tốt thì có thể tất cả các resouce của các game đều bị load vào. Khi không sử dụng nữa cũng sẽ vẫn nằm trong bộ nhớ, không bị release đi 🡪 lãng phí.

🡪Giải pháp: Thực hiện các thao tác release tài nguyên khi back game, chẳng hạn xóa các texture không được sử dụng trong texture cache, ...

Từ thực tế cài đặt thử nghiệm chúng ta thấy giải pháp 1 tuy rất tốt nhưng lại có những nhược điểm mà hiện tại chưa tìm được cách khắc phục. Giải pháp 2 có khá nhiều nhược điểm và việc khắc phục các nhược điểm này cũng không phải hoàn toàn triệt để, nhưng các nhược điểm đều ở một mức có thể chấp nhận được. Vì vậy chúng ta chọn cách làm thứ hai để tiến hành cài đặt.

**5. Cài đặt thử nghiệm**

Các bản cài đặt gồm có:

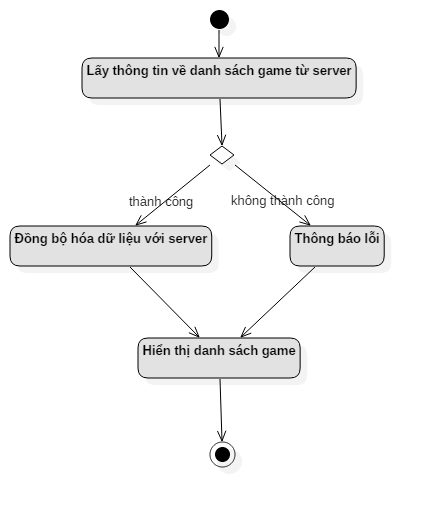
* Công cụ đóng gói game viết bằng python.
* Một server để push các gói game lên, cung cấp 1 API giả lập, trả về danh sách các gói game cho client.
* Ứng dụng cổng game di động phía client phát triển trên nền tảng cocos2d-x.

Trong phần này ta sẽ mô tả chi tiết cài đặt của cổng game di động phía client.

**Các use case cơ bản**

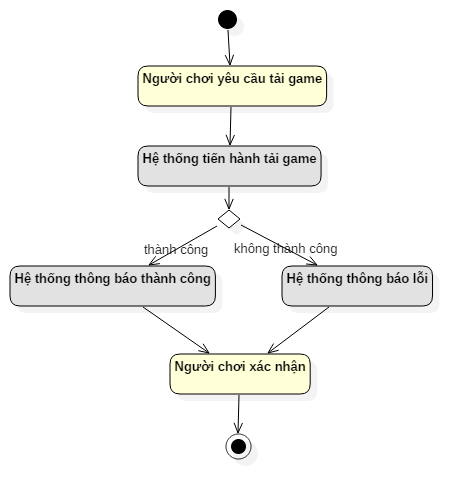
Có 3 usecase cơ bản là: Xem danh sách game, tải game và chơi game. Do các usecase này không quá phức tạp nên ta chỉ mô tả nó qua các biểu đồ hoạt động.

* Xem danh sách game:



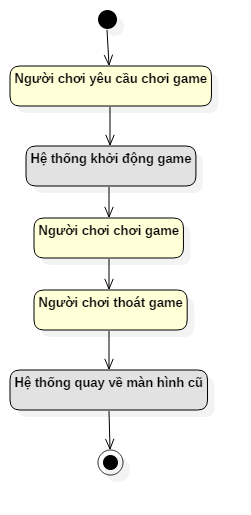
Ảnh 18 – Biểu đồ hoạt động của usecase xem danh sách game

* Tải game



Ảnh 19 - Biểu đồ hoạt động của usecase chơi game

* Chơi game



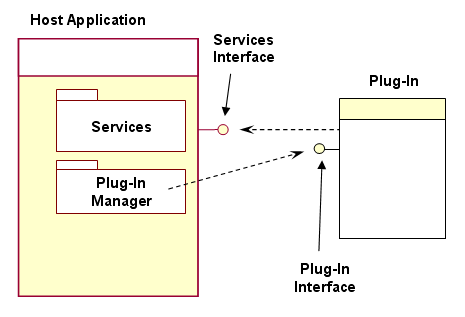
Ảnh 20 - Biểu đồ hoạt động của usecase chơi game

**Ý tưởng thiết kế**

-Thiết kế tách biệt tầng giao diện và tầng logic nghiệp vụ. Mục đích là sau này chúng ta có thể thay đổi không sử dụng giao diện cũ nhưng vẫn có thể giữ lại tầng nghiệp vụ, chừng nào flow cơ bản không thay đổi.

-Tầng logic nghiệp vụ được thiết kế dựa theo kiến trúc của các plugin

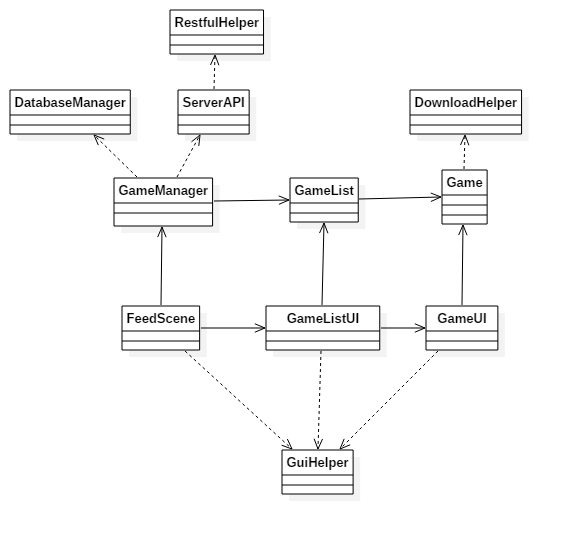
Theo đó, portal đóng vai trò là host application còn các game đóng vai trò là các plug-in.



Ảnh 21 - Kiến trúc plugin

**Sơ đồ lớp**

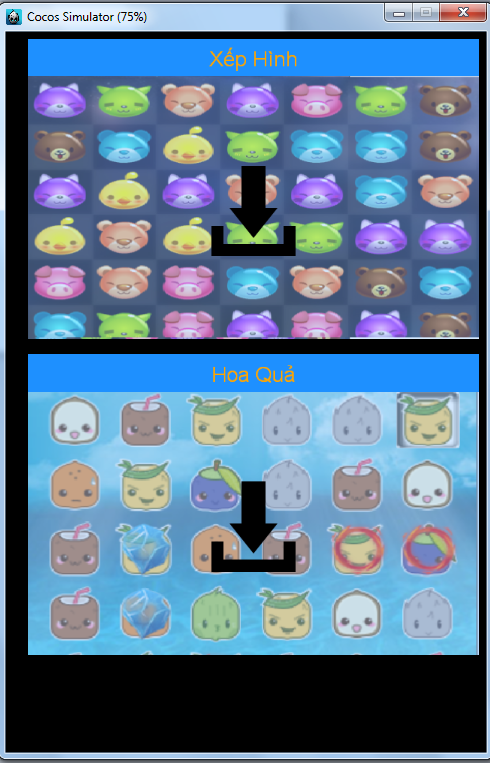
Dưới đây là sơ đồ tổng quan các lớp trong hệ thống



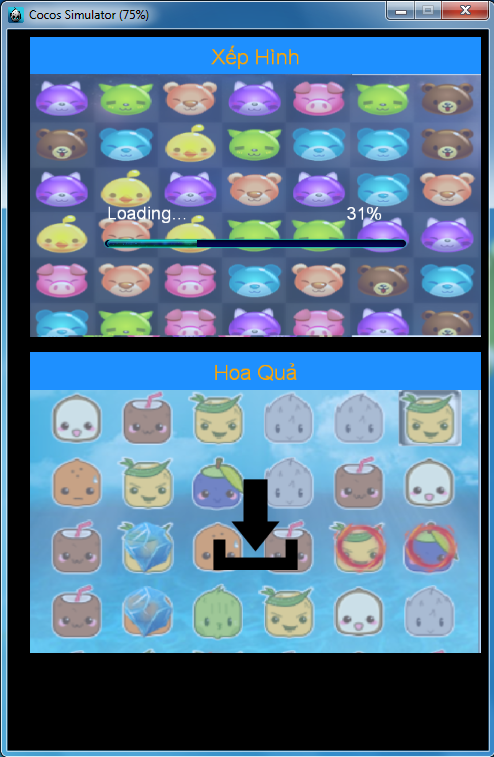
Ảnh 22 - Tổng quan về các lớp trong hệ thống

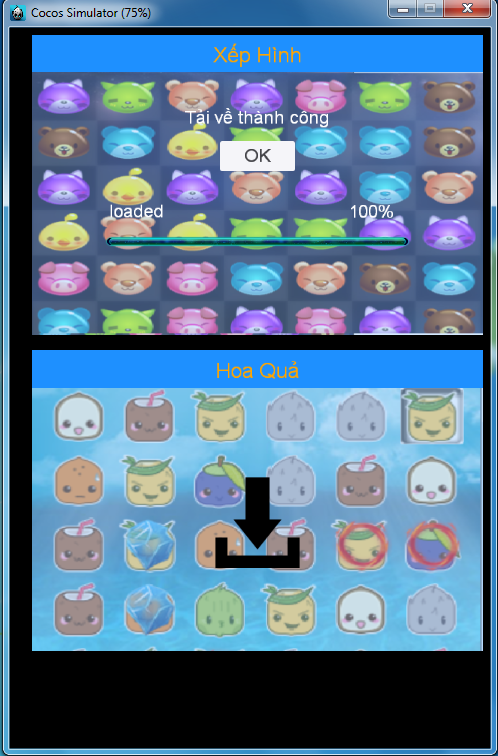
**Kết quả cài đặt**

* Hiển thị danh sách các game cho người dùng xem. Trong danh sách này, những game nào chưa tải thì hiển thị kèm nút tải về, những game nào đã tải thì hiển thị kèm nút chơi game. Lần đầu sử dụng ứng dụng thì tất cả các game đều chưa được tải về.

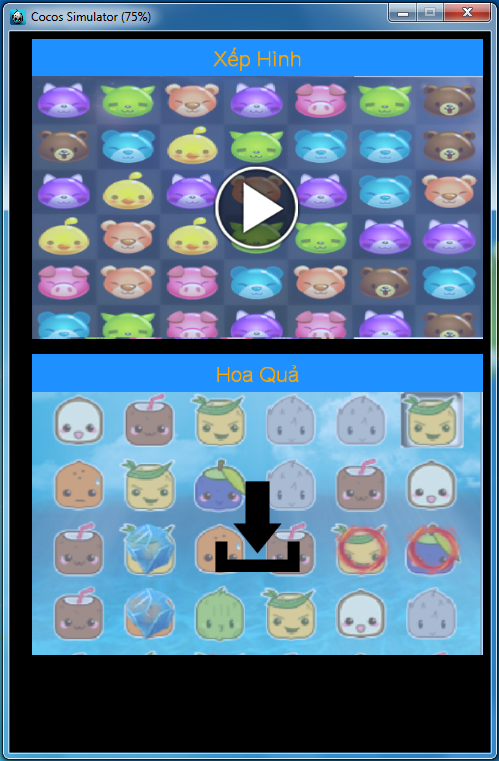


* Với những game chưa tải, người dùng có thể tải về. Trong quá trình đó, hệ thống hiển thị tiến trình tải về cho người chơi. Và thông báo cho người chơi sau khi kết thúc game.





* Với những game đã tải về, người chơi có thể chơi game. Sau khi chơi xong, người dùng thoát game và quay lại màn hình danh sách.



* Việc cài đặt portal đã hoàn thành và nhờ đó chứng minh được tính khả thi của ý tưởng. Tuy nhiên trong quá trình cài đặt, một số nguy cơ tiềm ẩn từ ý tưởng cũng đã được phát hiện. Đó là:
* Những rắc rối liên quan đến đường dẫn, chẳng hạn khi can thiệp vào searchPaths, thêm các thư mục tìm kiếm cho game engine.
* Nguy cơ từ việc sử dụng các đối tượng toàn cục hoặc những hàm có tác dụng toàn cục, ví dụ: các đối tượng quản lý các action, quản lý texture cache... Nếu bản thân trong game sử dụng không cần thận, có thể ảnh hưởng đến các tính năng bên ngoài portal.

Phần 3: Kết luận

**I. Tổng kết kết quả của đồ án**

Trong đồ án này, tất cả các nhiệm vụ được đề ra ban đầu đã hoàn thành, với các mức độ hoàn thành khác nhau.

Framework test – một gói phần mềm hỗ trợ việc manual unit test trên nền tảng cocos2d-x đã được hoàn thành, mặc dù chưa thực sự hoàn chỉnh. Chỉ có đúng 3 tính năng cơ bản:

* Tạo menu thao tác
* Tạo các thông báo nhanh
* Theo dõi trạng thái đối tượng

Nhưng bù lại nó chạy khá ổn định, tỏ ra hiệu quả trong việc unit test cho các đối tượng liên quan đến đồ họa, thực hiện được những test case phức tạp nhờ đó, khả năng phát hiện lỗi của module được kiểm thử khá tốt. Các API mà nó cung cấp đơn giản, dễ hiểu, dễ sử dụng. Ngoài mục đích thực hiện các unit test, framework test cũng thường xuyên được sử dụng để test các API của cocos2d-x, để thử nghiệm và phát triển các ý tưởng. Nó cho phép test code tách biệt với production code. Nhờ đó, việc quản lý mã nguồn hiệu quả hơn. Một test case có thể chạy nhiều lần, trên nhiều nền tảng khác nhau một cách dễ dàng. Việc loại bỏ mã kiểm thử ra khỏi sản phẩm cuối cũng được thực hiện khá đơn giản.

Về tìm hiểu cocos2d-x và phát triển một sản phẩm game thực tế cũng được hoàn thành. Sản phẩm game có thể chơi được, mặc dù có một số tính năng nhỏ chưa thực sự hoàn chỉnh. Thông qua việc phát triển sản phẩm game, em đã có những hiểu biết nhất định về phát triển một sản phẩm game, cũng như hiểu về cách thức hoạt động của game engine cocos2d-x, hiểu về các API của cocos cũng như cơ chế hoạt động bên trong của nó. Đây là những hiểu biết nền tảng để có thể nghiên cứu về các giải pháp công nghệ dựa trên cocos2d-x.

Việc nghiên cứu, đề xuất giải pháp công nghệ xây dựng cổng game di động cũng đã đạt được những kết quả khả quan. Sau khi xác định được mấu chốt vấn đề là kỹ thuật deploy game trên portal. Đồ án đã đưa ra 2 giải pháp để thực hiện kỹ thuật này, so sánh ưu nhược điểm của chúng và trên cơ sở đó lựa chọn một giải pháp hợp lý. Hiện thực hóa chúng thành một ứng dụng. Và bên cạnh việc chứng minh được tính khả thi của ý tưởng, chúng ta cũng phát hiện ra một số nguy cơ tiềm ẩn của giải pháp sử dụng.

**II. Hướng phát triển trong tương lai**

Hiện tại, giải pháp được đưa ra để xây dựng cổng game di động vẫn còn nhiều điểm chưa hoàn chỉnh. Nên trong tương lai, đồ án sẽ tiếp tục công việc nghiên cứu công nghệ xây dựng cổng game di động. Bằng cách:

* Hoàn thiện giải pháp: tìm cách giải quyết các nhược điểm và hạn chế nguy cơ của giải pháp.
* Đóng gói giải pháp: Phát triển giải pháp dưới dạng một hệ thống bao gồm các gói phần mềm, công cụ hỗ trợ, để có thể release như một sản phẩm thực tế, chứ không phải chỉ là một proof-of-concept như thời điểm hiện tại.

Và dựa trên gói giải pháp xây dựng được, ta có thể phát triển các sản phẩm cổng game di động thực tế, với đầy đủ các tính năng, cung cấp cho thị trường.

**Tài liệu tham khảo**

|  |  |
| --- | --- |
| [1] | NewZoo, "China’s & Southeast Asia’s Mobile Games Market," *https://newzoo.com/wp-content/uploads/2011/06/Newzoo\_Google\_Think-Gaming\_July\_V7\_Final.pdf,* 2015. |
| [2] | Wikipedia, "Experiential learning," *https://en.wikipedia.org/wiki/Experiential\_learning.* |
| [3] | wikipedia, "Proof of concept".  *https://en.wikipedia.org/wiki/Proof\_of\_Concept* |
| [4] | cocos2d-x, trang chính thức, *http://cocos2d-x.org/.* |