ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

ĐỖ QUANG DƯƠNG

LẬP TRÌNH GAME TRÊN ĐIỆN THOẠI DI ĐỘNG BẰNG NỀN TẢNG COCOS2D-IPHONE

LUẬN VĂN THẠC SĨ CÔNG NGHỆ THÔNG TIN

Hà Nội, 2016

ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

ĐỖ QUANG DƯƠNG

LẬP TRÌNH GAME TRÊN ĐIỆN THOẠI DI ĐỘNG BẰNG NỀN TẢNG COCOS2D-IPHONE

Ngành: Công nghệ thông tin
Chuyên ngành: Truyền dữ liệu và Mạng máy tính

LUẬN VĂN THẠC SĨ CÔNG NGHỆ THÔNG TIN
NGƯỜI HƯỚNG DẪN KHOA HỌC: TS. TRẦN THỊ MINH CHÂU

Hà Nội, 2016

LỜI CAM ĐOAN

Tôi xin cam đoan đây là công trình nghiên cứu của riêng tôi, tôi không sao chép của ai. Các số liệu, kết quả nêu trong Luận văn là trung thực và chưa từng được ai công bố trong bất kỳ công trình nào khác.

Tôi xin cam đoan rằng các trích dẫn trong Luận văn đã được chỉ rõ nguồn gốc.

Học viên thực hiện Luận văn

Đỗ Quang Dương

MỤC LỤC

DANH MỤC CÁC BẢNG

DANH MỤC CÁC HÌNH VẼ

CHƯƠNG 1. GIỚI THIỆU BÀI TOÁN

1.1. Hoàn cảnh bài toán:

Năm 2013, game di động Flappy Bird của tác giả Nguyễn Hà Đông đã gây một tiếng vang rất lớn cho giới lập trình viên tại Việt Nam và toàn Thế giới. Vào tháng 1 năm 2014, game đã đứng đầu trong danh sách download nhiều nhất trên Apple Store, với hơn hàng chục triệu download. Game đã nhanh chóng làm cho tác giả Nguyễn Hà Đông trở lên nổi tiếng toàn Thế giới, và kiếm được số tiền theo dự đoán khoảng 50.000 USD/ngày. Doanh thu này giúp Nguyễn Hà Đông trong một thời gian ngắn trở thành tỷ phú.

Trong khi đó, với góc nhìn của các lập trình viên, game Flappy Bird là một game có đồ hoạ đơn giản, kỹ thuật chơi đơn giản và không hề khó về mặt kỹ thuật để tạo ra một game tương tự. Game đã được nhân bản, clone với rất nhiều phiên bản ăn theo. Tất cả các forum về lập trình game, các trang hướng dẫn lập trình game đều có ít nhất một bài viết hướng dẫn tạo game tương tự như Flappy Birds, bằng nhiều nền tảng khác nhau Cocos2d, Unity, Game Box, … Thông thường với một lập trình viên mới học, cũng chỉ mất khoảng 2 đến 7 ngày để tạo ra một game tương tự. Điều này làm cho các lập trình viên toàn thế giới đổ xô vào học lập trình game di động với hi vọng đạt được kết quả như Flappy Bird.

Game Flappy Bird cho thấy một lập trình viên đơn lẻ, với chi phí nhỏ, không cần thuê hạ tầng máy chủ, không cần có đội ngũ đồ hoạ hỗ trợ, cũng có thể tạo ra một game có giá trị hàng tỷ đồng Việt Nam, qua đó thúc đẩy các lập trình viên trên toàn Việt Nam học và xây dựng game di động. Với học viên, cũng không nằm ngoại lệ, đã có nhu cầu tìm và học một nền tảng lập trình game cho di động để tận dụng thời gian rảnh rỗi và với hi vọng có thể kiếm được nguồn thu nhập thụ động từ viết game.

1.2. Tình hình công nghệ thế giới:

Hiện nay có rất nhiều nền tảng lập trình game di động, trong đó đại đa số là miễn phí. Vậy nền tảng lập trình game di động là gì: đó là một bộ công cụ hỗ trợ lập trình viên lập trình game di động và public cho người dùng nhanh chóng, gồm:

- Bộ thư viện lập trình: là tập hợp các class hỗ trợ việc lập trình game di động, gồm các class quản lý đối tượng trong game, quản lý action (hành động), mô phỏng các hiện tượng vật lý như va chạm, lực quán tính, ma sát, trọng lượng, mô phỏng âm thanh, ánh sáng, và các hiệu ứng đặc biệt như mưa, lửa, khói, ...
- Bộ công cụ đồ hoạ giúp việc lập trình game di động nhanh chóng, export game thành file chạy, tương thích với từng nền tảng như iOS hay Android, Windows Phone, ...

Vậy với một lập trình viên đơn lẻ, không biết bắt đầu từ đâu để lập trình game di động, anh ta cần tìm một framwork lập trình game thoả mãn các yêu cầu:

- Framework lập trình game di động, có thể tạo ra game cho các nền tàng iOS, Android.
- Frameword phải có các công cụ hỗ trợ kiếm tiền từ việc bán đồ hay quảng cáo trong game.
- Framework phải miễn phí
- Frameword hỗ trợ lập trình game 2D (vì với game 3D, đòi hỏi kiến thức rất nhiều và phải có đội ngũ hỗ trợ đồ hoạ lớn, không phù hợp với lập trình viên cá nhân)
- Framwork đó phải có cộng đồng phát triển lớn, để có thể tra cứu, hỏi nhằm sửa lỗi
- Framework phải có công cụ hỗ trợ đồ hoạ, giúp việc tạo ra game dễ dàng, nhanh chóng hơn.

Học viên cũng là một trong các lập trình viên nêu trên, và trong quá trình tìm hiểu, học viên đã tìm được một nền tảng thoả mãn các yêu cầu nên trên. Đó là nền tảng Cocos2d-iPhone.

1.3. Phát biểu bài toán

Với một lập trình viên bình thường, khi bắt tay vào học lập trình game cho thiết bị di động thông minh cần tìm hiểu những công cụ gì? Cần kiến thức gì và cách làm như thế nào? Đây là các câu hỏi mà học viên cũng đã từng gặp khi bắt đầu học lập trình game. Dưới đây học viên xin tóm tắt lại các điểm chính của việc lập trình game di động:

a. Tìm hiểu nền tảng lập trình game di động

Hiện có rất nhiều nền tảng lập trình game 2D trên thiết bị di động, đại đa số chúng đều miễn phí. Có một số framework có phí như Unity và Unreal, trong đó chi phí cho các framwork này khá đắt (như Unity có phí lên đến 1500$/hệ điều hành/1bản)

Tại sao lại là game 2D: bởi việc xây dựng game 2D thường đỡ tốn công sức hơn rất nhiều so với game 3D. Game 3D thường không phải dành cho những lập trình viên nghiệp dư hoặc không chuyên về làm game. Để làm game 3D thường lập trình viên phải có hiểu biết sâu sắc về việc dựng hình 3D, các thuật toán tạo hình khối, ... và các công cụ lập trình game 3D thường khá phức tạp, đòi hỏi nhiều thời gian để học làm chủ công cụ.

Với game 2D, một lập trình viên có thể phát triển game mà không cần nhiều đội ngũ hỗ trợ như các nhân viên đồ hoạ, nhân viên âm thanh, ... Game 2D phù hợp hơn với các lập trình viên nghiệp dư.

b. Cách thức tìm kiếm doanh thu từ Game di động

Có 3 cách để tìm kiếm doanh thu từ Game di động:

- Bán game di động: các Công ty hoặc lập trình viên lập trình game và bán trên các chợ ứng dụng thông dụng như App Store của Apple hoặc Play Store của Google, các Game này thường có giá từ 1$ trở lên. Việc kiếm tiền từ cách này rất khó, đòi hỏi game xây dựng phải hay và thường công ty phát hành game phải có tiếng tăm trên thị trường.
- Bán đồ trong game: đây là cách thức khá phổ biến của các Công ty làm game di động, thường họ tạo ra các game có chất lượng rất tốt, có thể kích thích người sử dụng mua các vật phẩm trong game, ví dụ các loại vũ khí, hiệu ứng đặc biệt, ... Cách này thường phù hợp với các game có chất lượng tốt, phức tạp và có khả năng giữ người chơi trong một thời gian dài.
- Quảng cáo trong game: đây là cách thức phổ biến của các game di động thuộc loại đơn giản về nội dung. Với cách này, lập trình viên thường sử dụng các thư viện API của các công ty quảng cáo trên Internet như Admob, Unity Ads, Apple Ads để gắn vào game, giúp hiển thị quảng cáo trong game. Lập trình viên sẽ được trả tiền cho các hiển thị quảng cáo trong game và các click vào quảng cáo của người sử dụng.

c. Tìm hiểu công cụ hỗ trợ

Hầu hết các nền tảng miễn phí thường có ít công cụ hỗ trợ, đòi hỏi lập trình viên phải tìm hiểu nhiều để tìm được các công cụ hỗ trợ thông dụng, dễ sử dụng, giúp việc lập trình được thuận lợi, nhanh chóng. Các công cụ ở đây gồm:

- IDE: công cụ lập trình, cho phép biên dịch code thành mã nhị phân, thực thi được trên thiết bị di động. Các IDE còn có thể hỗ trợ việc export ra file nhị phân tương thích với từng nền tảng di động, và hỗ trợ việc upload file lên các chợ ứng dụng.
- Công cụ hỗ trợ việc tạo các chuyển động phức tạp: thông thường các chuyển động phức tạp không được các nhân viên đồ hoạ tạo ra ngay từ đầu, họ tạo ra các thành phần của đối tượng (như chân, tay, đầu, mình, đuôi, cánh, ...) riêng. Sau đó tuỳ theo chuyển động để tạo ra các hình ảnh động tương ứng, từ việc kết hợp vị trí của các bộ phận này. Công cụ hỗ trợ việc tạo các chuyển động phức tạp sẽ giúp việc tạo các chuyển động này nhanh chóng hơn, đồng thời cho phép kết xuất ra file theo định dạng mà nền tảng lập trình game di động hỗ trợ.
- Công cụ hỗ trợ tạo màn chơi hay bản đồ: với các game có đồ hoạ phức tạp, việc tạo ra một màn chơi hay bản đồ hoàn toàn bằng code là vô cùng phức tạp và mất rất nhiều công sức, vì vậy, các công cụ tạo bản đồ (map) ra đời, hỗ trợ rất nhiều cho các lập trình viên. Công cụ này sẽ tạo ra các bản đồ bằng cách kéo

thả các thành phần của bản đồ, cho phép lập trình viên nhìn một cách trực quan bản đồ này, giúp việc lập trình nhanh chóng và hiệu quả.

- Công cụ hỗ trợ việc tạo các hiệu ứng đặc biệt như lửa, khói, mưa, tuyết. Thường mỗi hiệu ứng đặc biệt có khoảng 30 thuộc tính, việc lập trình tạo ra một hiệu ứng rất khó và mất nhiều thời gian để tinh chỉnh. Công cụ hỗ trợ tạo hiệu ứng đặc biệt sẽ giúp lập trình viên nhìn thấy ngay hiệu quả của mỗi khi thay đổi các tham số.

Qua các phân tích nêu trên, cho thấy việc tạo được một game di động 2D hay tốn khá nhiều công sức của lập trình viên, đòi hỏi phải tìm hiểu một khối lượng công việc không hề nhỏ. Đề tài tìm hiểu tổng quan về việc xây dựng game cho thiết bị di động thông minh, khảo sát một số framework hỗ trợ, và cách thức tìm kiến doanh thu từ game di động, từ đó đề tài áp dụng cho việc xây dựng game di động để tạo thu nhập từ các sản phẩm đó.

Nội dung của đề tài gồm các nội dung chính sau đây:

- Giới thiệu tổng quan về các framework lập trình game di động và lý do vì sao chọn Cocos2d-iPhone để tìm hiểu.
- Cocos2d-iPhone và các module chính của framework, cũng như các công cụ hỗ trợ lập trình viên.
- Tổng quan về cách thức tìm kiếm thu nhập trong lập trình game di động
- Một số ứng dụng mà học viên đã xây dựng bằng Cocos2d-iPhone.

Trên đây là bài toán mà học viên cần phải giải quyết, nội dung chi tiết sẽ có trong các chương tiếp theo.

CHƯƠNG 2. MỘT SỐ FRAMEWORK LẬP TRÌNH GAME DI ĐỘNG

Chương này mô tả một cách tổng quan về các framework lập trình game di động và so sánh giữa các frawework, và lý do vì sao học viên lựa chọn frawework Cocos2d-iPhone.

Như đã nêu ở trên, với một lập trình viên game nghiệp dư, việc tìm kiếm một nền tảng cho phép lập trình ra các game 2D nhanh chóng, hiệu quả cần thoả mãn các điều kiện sau:

- Framework lập trình game di động, có thể tạo ra game cho các nền tàng iOS, Android.
- Framework phải có các công cụ hỗ trợ kiếm tiền từ việc bán đồ hay quảng cáo trong game.
- Framework phải miễn phí
- Frameword hỗ trợ lập trình game 2D (vì với game 3D, đòi hỏi kiến thức rất nhiều và phải có đội ngũ hỗ trợ đồ hoạ lớn, không phù hợp với lập trình viên cá nhân)
- Framwork đó phải có cộng đồng phát triển lớn, để có thể tra cứu, hỏi nhằm sửa lỗi
- Framework phải có công cụ hỗ trợ đồ hoạ, giúp việc tạo ra game dễ dàng, nhanh chóng hơn.

Hiện nay trên thế giới có rất nhiều nền tảng lập trình game di động, trong đó phải kể đến các nền tảng sau:

2.1. Cocos2d

Cocos2d [1] là một framework lập trình game 2d mã nguồn mở, được bắt đầu xây dựng từ ngày 29/2/2008. Tại làng Los Cocos, gần Córdoba in Argentina , lập trình viên Ricardo Quesada đã cùng với một số người bạn của anh, tạo nên một framework lập trình game, được đặt tên là “Los Cocos”. Sau khi phiên bản 1.0 được ra đời, thì được đổi tên thành “Cocos2d”. Mới đầu nền tảng này được xây dựng nhằm phát triển các game cho máy tính PC, trên nền OS là Windows hoặc Linux. Tuy nhiên hiện nay đã có khá nhiều biến thể của cocos2d, được sử dụng cho các nền tảng khác nhau, trong đó phổ biến nhất là các nhánh:

- Cocos2d-x: là framework dựa trên nguồn gốc Cocos2d, do một nhóm các nhà phát triển game của Trung Quốc xây dựng, được xây dựng để phát triển game 2d bằng ngôn ngữ C++, hiện nay đây là nhánh phát triển nhất của Cocos2d, được rất nhiều nhà phát triển game sử dụng. Đây là framework có thể dùng để phát triển game cho điện thoại di động dùng hệ điều hành iOS, hoặc Android. Người sáng

lập ra Cocos2d, hiện cũng đã đầu quân cho nhóm phát triển Cocos2d-x, vì vậy đây là nhánh lớn nhất của Cocos2d.

- Cocos2d-iPhone (hiện đã đổi tên thành Cocos2d-ObjC: là framework dùng ngôn ngữ Object-C của Apple để phát triển game 2d cho hệ điều hành iOS, dùng cho các dòng thiết bị iPhone, iPod, iPad của Apple. Đã có những công cụ giúp chuyển mã nguồn Cocos2d-iPhone sang chạy trên hệ điều hành Android, tuy nhiên vẫn còn nhiều lỗi và chưa phổ biến.

- Cocos2d-html5: là nhánh của cocos2d được sử dụng để phát triển các game trên nền tảng html5. Hiện nay nhóm phát triển cho nền tảng này cũng chính là nhóm phát triển của Cocos2-x.

- Cocos2d-xna: là nhánh của cocos2d được sử dụng để phát triển các game cho Windows Phone và Xbox360, sử dụng ngôn ngữ lập trình C#.

- Hiện nay nhóm phát triển Cocos2d-x do Công ty Chukong Technologies Inc tài trợ cũng đã bắt tay vào xây dựng framework Cocos3d để phát triển các game 3D

a. Cocos2d-iPhone

Vào tháng 3 năm 2008, và sau đó là tháng 6 năm 2010, hệ điều hành iOS của Apple ra đời, cùng với nó là sản phẩm điện thoại thông minh iPhone, nhận thấy tiền năng to lớn của hệ điều hành và sản phẩm iPhone, rất nhiều lập trình viên trên toàn thế giới đã bắt tay xây dựng phần mềm cho nó. Không nằm ngoài xu thế đó, tác giả Ricardo Quesada cũng nhanh chóng chuyển Cocos2d sang phát triển phầm mềm cho iPhone, chính vì thế Cocos2d-iPhone [1] ra đời.

Cocos2d-iPhone là framework viết bằng Object-C và sử dụng chính IDE lập trình xCode của Apple làm công cụ phát triển chính. Để tạo thuận lợi cho lập trình viên, đã có hàng loạt công cụ ra đời, giúp cho việc xây dựng game 2d được dễ dàng hơn, trong đó phải kể đến các công cụ:

- SpriteBuilder: là công cụ đồ hoạ, giúp cho việc tạo các màn chơi (scene), quản lý tài nguyên cho lập trình viên dễ dàng hơn. Đây là công cụ chính để phát triển game 2d dùng Cocos2d-iPhone
- Particle Designer: là công cụ giúp tạo ra các hiệu ứng đặc biệt như lửa cháy, tuyết rơi, khói, ... để cho vào các màn chơi
- TexturePacker: là công cụ giúp gộp các ảnh, tài nguyên của game vào 1 file nhằm tăng tốc quá trình load game.

- Sprite: là công cụ giúp tạo ra các hình chuyển động, dựa trên các mảnh ghép của cơ thể nhân vật.

b. Cocos2d-x

Vào tháng 10 năm 2010, một lập trình viên người Trung Quốc tên là Zhe Wang đã xây dựng một nhánh khác của Cocos2d, và đặt tên là Cocos2d-x [1]. Cocos2d-x. Hiện nay đây là nhánh phát triển nhất của Cocos2d, sáng lập viên của Cocos2d cũng đã join vào nhóm phát triển Cocos2d-x. Cocos2d-x sử dụng ngôn ngữ lập trình C++ để phát triển game, do đó nó có thể được sử dụng để phát triển trên nhiều hệ điều hành khác nhau, phổ biến được dùng cho iOS và Android.

Cocos2d-x cũng có công cụ để tạo màn chơi bằng đồ hoạ gọi là Cocos Code IDE.

Giữa hai nền tảng này có sự tương đồng rất lớn, hầu như các hàm API của chúng đều giống nhau, khác biệt chính và lớn nhất là Cocos2d-x sử dụng C++ để lập trình còn Cocos2d-iPhone sử dụng Object-C. Tuy nhiên hiện nay Cocos2d-x được sử dụng nhiều hơn và có lượng lập trình viên rất lớn (có lẽ bởi số lượng lập trình viên Trung Quốc phát triển game 2d sử dụng Cocos2d-x).

2.2. Unity

Unity [2] là nền tảng phát triển game 3d, nhưng có thể dùng để phát triển game 2d, do Công ty Unity Technologies phát triển. Nó có thể được sử dụng để phát triển game trên hầu hết các nền tảng phần cứng (PC, Mac, iPhone, Android, Windows Phone, ...). Đây là một trong các nền tảng lập trình game lớn nhất hiện nay, được hàng triệu lập trình viên trên thế giới sử dụng. Unity có một bộ công cụ lập trình và hỗ trợ lập trình viên rất phong phú, gần như không phải sử dụng các công cụ bên ngoài để viết game. Unity sử dụng C# để lập trình. Nền tảng này đạt được rất nhiều phần thưởng cho nền tảng phát triển tốt nhất và có rất nhiều game có chất lượng cao được phát triển từ nền tảng này.

2.3. SpriteKit

SpriteKit [3] là một game engine khác do chính Apple tạo ra năm 2013. Được sử dụng để phát triển các game2d cho hệ điều hành iOS và xOS. Sử dụng ngôn ngữ Object – C hoặc Switf của Apple. Hiện nay nền tảng này cũng vẫn chưa được sử dụng nhiều, do nó chỉ chủ yếu để phát triển cho iOS và xOS (máy Mac).

Do Apple phát triển nền SpriteKit sử dụng chính IDE xCode của Apple để phát triển, tuy nhiên xCode không có công cụ đồ hoạ để tạo màn chơi, chính vì vậy để tạo màn chơi, lập trình viên có thể sử dụng các công cụ như là SpriteBuilder hoặc Tiled để tạo màn chơi.

Công cụ này về cơ bản có nhiều nén giống với Cocos2d-iPhone, và nên sử dụng để tạo các game có đồ hoạ và cách chơi đơn giản.

2.4. Các nền tảng khác

Hiện nay trên thế giới có hàng chục game engine khác nhau, từ những công cụ mất phí, rất mạnh và phức tạp như UnReal, Unity cho đến các công cụ mà thậm chí không phải lập trình để có thể tạo được game như Game Maker, Build Box, ... Bảng dưới đây là các so sánh giữa chúng:

Bảng 2.1:so so sánh các framework lập trình game di động

Framework	iOS/ Android	Kiếm tiền	Miễn phí	2D/3D	Cộng đồng lớn	Công cụ hỗ trợ	Ngôn ngữ sử dụng
Cocos2D-iPhone	iOS/Android	X	X	2D	X	Nhiều	Object C
Cocos2D-x	iOS/Android	X	X	2D	X	Nhiều	C++
Unity	iOS/Android	X		2D/3D	X	Nhiều	C#
Game Maker[4]	iOS/Android	X		2D		Ngay trong tool	Unknow
SpriteKit[3]	iOS	X	X	2D			Object C
Construct2[5]	Html5	X		2D		Ngay trong tool	Javascript
Build Box[6]	iOS	X		2D		Ngay trong tool	Unknow
Unreal[7]	iOS/Android	X	5% doanh thu mỗi game	2D/3D	Chủ yếu là các lập trình game chuyên nghiệp	Nhiều	C++

Nhìn từ bảng so sánh nêu trên, có thể thấy các nền tảng Cocos2D-iPhone và Cocos2D-x là các nền tảng thoả mãn nhiều điều kiện nhất. Tuy nhiên Cocos2D-x dùng C++ để lập trình. Trong C++, việc quản lý bộ nhớ khá phức tạp và mất nhiều công sức của lập trình viên, đòi hỏi lập trình viên phải có kinh nghiệm lập trình C++. Chính vì thế, với học viên, nền tảng Cocos2D-iPhone là một lựa chọn tốt, thoả mãn nhiều tiêu chí.

CHƯƠNG 3. COCOS2D-IPHONE

Chương này,dùng để mô tả tổng quan về một chương trình viết theo framework Cocos2d-iPhone thường có, các module quan trọng của thư viện lập trình Cocos2d-iPhone [8].

3.1. Cấu trúc một chương trình viết bằng Cocos2d-iPhone

Cấu trúc một chương trình Cocos2d-iPhone khá đơn giản:

a. Lớp AppDelegate do chương trình tự sinh có hàm (CCScene *)startScene(), trong hàm này sẽ gọi ra scene (cảnh) đầu tiên của game, thường có thể là game menu, trong game menu có các đường dẫn đến các cảnh khác của game.

Ví dụ:

```
-(CCScene *)startScene
{
// This method should return the very first scene to be run when your app starts.
        return [IntroScene scene];
}
```

Trong mỗi một scene (cảnh) có hàm static dùng để khởi tạo scene, được cài đặt theo mẫu:

```
+ (IntroScene *)scene{
        return [[self alloc] init];
}

- (id)init{
        if ((self = [super init])) {
        // Khởi tạo các thành phần của scene ở đây, ví dụ các nút bấm, các nhân vật trong game…
        }
        return self;
}
```

b. Trong một scene hàm quan trọng nhất là hàm update(), mô tả như dưới đây:

```
- (void)update:(CCTime)dt
```

```
{
//các xử lý mỗi detla thời gian diễn ra
}
```

Hàm update sẽ được chương trình tự động gọi mỗi một khoảng thời gian delta diễn ra, đây chính là lúc lập trình viên tính toán và hiển thị lại các thành phần trong game. Khoảng thời gian detla được tính toán thường là 1/60 giây.

3.2. Các module chính trong thư viện lập trình Cocos2d-iPhone [8]

Trong thư viện lập trình game Cocos2d-iPhone, có thể chia thành các module chính sau đây:

- Quản lý đối tượng game: Các class dùng để quản lý đối tượng trong game, gồm các thông tin như là tọa độ, hình đại diện, các đối tượng con thuộc nó, toạ độ, …Đối tượng game gồm nhiều loại: các item, các scene, các hiệu ứng, …
- Quản lý hành động: Các class dùng để mô tả hành động trong game, hành động gồm rất nhiều loại, ví dụ xoay tròn, ẩn, hiện, lặp lại, di chuyển, thực hiện hoạt hình, biến mất, … Có khoảng 60 loại hoạt động (action) khác nhau đã được lập trình sẵn.
- Hiệu ứng vật lý: Các class dùng để mô phỏng các hiện tượng vật lý như là va chạm, lò xo, lực hút, …
- Hiệu ứng đặc biệt: Các class dùng để mô phỏng các hiệu ứng đặc biệt như lửa cháy, khói, mưa, tuyết rơi, …
- Hiệu ứng âm thanh: Các class dùng để thực hiện các hiệu ứng âm thanh và thực hiện các âm thanh trong game.

Dưới đây là mô tả cụ thể các module trong game.

3.3. Quản lý đối tượng game

Các đối tượng trong game gồm nhiều loại, có thể kể đến như sau:

a. Các scene: các cảnh trong game, cần quản lý được scene hiện tại hiển thị trên màn hình, cách thức chuyển đổi các scene. Cocos2d-iPhone có các hình thức chuyển đổi các scene là:
   - Mờ dần trong một khoảng thời gian nhất định.
   - Mờ dần trong một khoảng thời gian nhất định với màu cho trước
   - Di chuyển cảnh cũ đến một vị trí nhất định của cảnh mới.
   - Trượt cảnh mới vào vị trí của cảnh cũ, cảnh cũ được giải phóng sau đó.
   - Cảnh mới bật ra chèn lên cảnh cũ, cảnh cũ được giải phóng sau đó.

b. Các đối tượng windown form: là các đối tượng cơ bản của một giao diện đồ hoạ như: button, text, label, listview, font chữ, …

c. Các đối tượng trong game: nhân vật, kẻ thù, tiền, đá, …được sử dụng để mô tả game. Một đối tượng có nhiều thuộc tính, trong đó có các thuộc tính sau là quan trọng nhất:
   - Anchor point - điểm neo - được sử dụng để xác định toạ độ của đối tượng trong toạ độ 2 chiều (x, y);
   - Z order: đối tượng có Z order càng dương thì càng hiển thị lên phía trước màn hình.
   - Góc quay: để biết đối tượng đang được quay bao nhiêu độ
   - ScaleX, Y: tỷ lệ kéo dãn của đối tượng
   - Childs: các đối tượng con của nó. Ví dụ để tạo được 1 người thì có thể lấy mình làm đối tượng chính, các đối tượng con là đầu, chân, tay.
   - PhysicBody: hình để thực hiện các mô phỏng vật lý.

     …

3.4. Quản lý hành động

Đây là module quản lý các hành động của một đối tượng trong game. Có khoảng 60 hành động được lập trình sẵn, được bắt đầu bởi chữ CCAction, có thể chia làm 3 loại:

- Hành động đơn lẻ: là các hành động đơn lẻ như di chuyển đến 1 điểm, xoay tròn, mờ dần, tự xoá nó đi,…
- Hành động gộp: là class sẽ gộp nhiều hành động đơn lẻ hoặc hành động gộp khác để thực hiện các hành động phức tạp như là: vừa di chuyển vừa xoay tròn, vừa di chuyển vừa nhấp nháy, di chuyển đến điểm A và thực hiện xoay tròn sau đó… Có các hành động gộp được lập trình sẵn gồm:
  - Hành động nối tiếp: các hành động con của nó sẽ được thực hiện tuần tự theo thứ tự được thêm vào.
  - Hành động song song: các hành động con của nó sẽ được thực hiện đồng thời với nhau.
  - Hành động lặp: các hành động con của nó sẽ được lặp đi lặp lại một số lần hoặc lặp đi lắp lại mãi cho đến khi nó được Stop hoặc Remove.
- Hành động để mô phỏng hoạt hình: thông thường các nhân vật trong game sẽ chuyển động theo dạng hoạt hình, để thực hiện được các hoạt hình này, cần một tập ảnh mô tả liên tiếp của đối tượng, sau đó sẽ được đọc ảnh và khai báo trong class CCAnimation. Để thực hiện các hành động phức tạp, hoạ sĩ thường vẽ các bộ phận của đối tượng như chân, tay, mình, ... rồi dùng các phần mềm hỗ trợ để tạo ra ảnh chuyển động của nhân vật. Phần mềm hỗ trợ ở đây được sử dụng nhiều nhất là **Spine[9]**, đây là công cụ có phí.

Hình 3.1:Giao diện phần mềm Spine[1]

3.5. Hiệu ứng vật lý

Là module dùng để mô phỏng các hiện tượng vật lý như là lực hút trái đất, va chạm, lò xo, nổ bom hay lực hút nam châm, …

Đây là module phức tạp và rất quan trọng trong Coos2d-iPhone. Gồm 2 thư viện chính là:

- Chipmunk: thư viện được viết bằng C, được thêm vào trong thư viện Cocos2d-iPhone nhằm mô phỏng các hiệu ứng vật lý.
- Box2d: thư viện được viết bằng C++, chạy chậm hơn Chipmunk nhưng dễ sử dụng hơn. Đồng thời nó có một số tính năng mới như giải quyết được vấn đề khi một vật di chuyển với tốc độ cao thì các hiệu ứng ma sát, .. biến mất.
- Hiện Cocos2d-iPhone chỉ có các Class hỗ trợ việc mô phỏng được hiện tượng trọng lực và va chạm, các hiện tượng khác bắt buộc phải dùng các thư viện bên ngoài, ví dụ như sử dụng một trong hai thư viện mô phỏng nêu trên

Mỗi đối tượng được cấu hình sử dụng hiệu ứng vật lý cần được gán một rigit body, nó có thể là dạng hình chữ nhật, hình tròn hoặc đa giác. Có 2 loại rigit body là loại dynamic (có thể chuyển động) và loại static (không thể chuyển động), loại static thường được dùng để mô phỏng các bức tường, nền đất, … mặc dù vẫn có các hiệu ứng vật lý, nhưng hầu như không thay đổi khi tương tác.

[1] Ảnh được lấy tại: http://esotericsoftware.com/

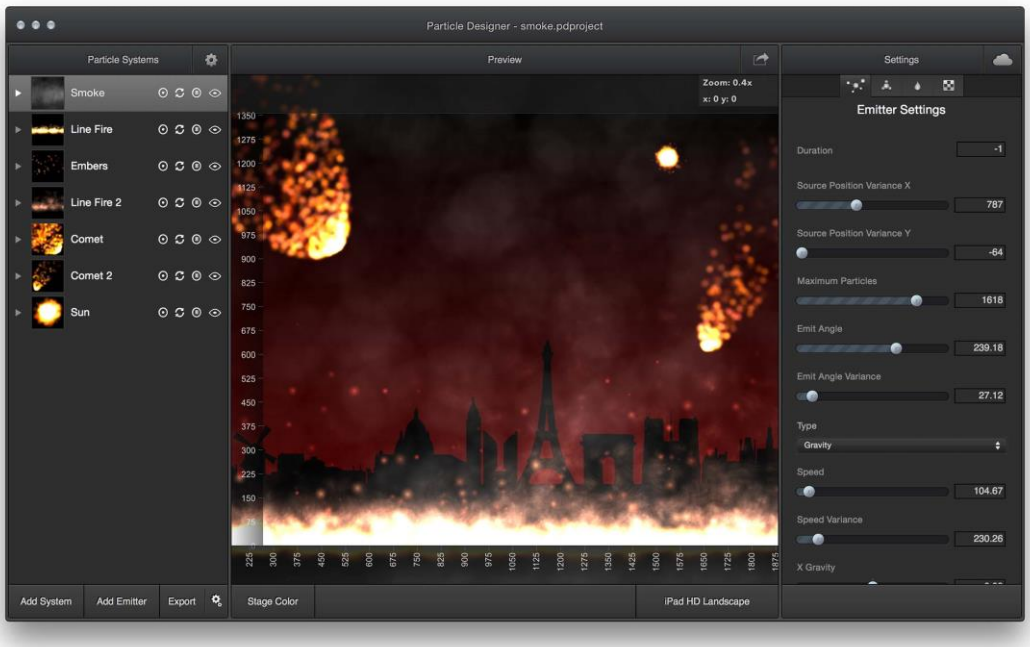
Một dynamic rigit body có 3 thuộc tính quan trọng là: khối lượng, ma sát và đàn hồi. Và các đối tượng vật lý được nối với nhau bởi các khớp nối. Nhờ có các thuộc tính này, lập trình viên có thể mô phỏng được các vật thể khó như: lò xo, cơ cấu cánh cửa, lực hút, lực đẩy, …

Sau khi thiết lập các tham số vật lý, việc các đối tượng tương tác với nhau sẽ do thư viện mô phỏng vật lý tự động tính toán và mô phỏng, chúng ta không nên tác động vào quá trình này.

Việc thiết lập tham số, chạy mô phỏng để kiểm tra các thiết lập tham số đã đạt yêu cầu hay chưa cần có công cụ đồ hoạ để thử nghiệm, chính vì thế, theo học viên, không nên dùng Cocos2d-iPhone để lập trình các game có mức độ mô phỏng vật lý phức tạp, bởi lẽ công cụ đồ hoạ hỗ trợ làm game chưa đủ mạnh, dẫn đến việc tính toán, thử nghiệm các giá trị tham số đòi hỏi nhiều thời gian để thực hiện. Chúng ta chỉ nên dùng Cocos2d-iPhone cho các game có đồ hoạ đơn giản và ít sử dụng các hiệu ứng vật lý.

3.6. Hiệu ứng đặc biệt

Các hiệu ứng đặc biệt như lửa, khói, tuyết rơi, mưa rơi, … trong game được tạo ra bởi Particle System. Một hiệu năng đặc biệt thường rất phức tạp, gồm khoảng 30 thuộc tính. Để mô phỏng được một hiệu ứng, nếu chỉ dùng code thì việc thử - sai sẽ mất rất nhiều thời gian. Do đó, để hiệu quả, cần phải có các công cụ hỗ trợ. Một trong các công cụ được sử dụng là Particle Designer [10]. Công cụ này cho phép thay đổi tham số các Particle và chạy ngay trên giao diện đồ hoạ để xem được hiệu ứng. Sau đó export hiệu ứng ra thành file tương thích với Cocos2d-iPhone và gọi trực tiếp trong chương trình.



Hình 3.2: Giao diện công cụ Particle Designer[2]

Trước đây Cocos2d-iPhone có xây dựng trước một số hiệu ứng đặc biệt để lập trình viên có thể sử dụng trong chương trình, tuy nhiên đến phiên bản mà học viên sử dụng, các class đó đã bị loại bỏ, toàn bộ việc thiết kế ra các hiệu ứng sẽ được các công cụ bên thứ 3 (ví dụ công cụ Particle Designer nêu trên) tạo ra và export dạng file tương thích với Cocos2d-iPhone (là file Plist) để sử dụng trong game.
Đoạn code mẫu như sau:

```
CCParticleSystemQuad *emitter;
-(void) particleShow
{
    emitter.position = ccp(screenSize.width/2, screenSize.width/2);
   emitter = [CCParticleSystemQuad particleWithFile: @"file.plist"];
  // emitter.life =0.3;
  // emitter.duration = 0.50;
   [self addChild:emitter];
  // emitter.autoRemoveOnFinish = YES;
   [emitter release];
}
```

[2] Ảnh được lấy tại https://71squared.com/particledesigner

3.7. Hiệu ứng âm thanh

Trong Cocos2d-iPhone, hiệu ứng âm thanh của thư viện này rất đơn giản, hầu như không có tính năng gì đặc biệt. Nó đơn thuần là play một file âm thanh tại mỗi thời điểm thích hợp. Để play một file âm thanh, đơn giản ta chỉ cần thực hiện lệnh:

```
[[OALSimpleAudio sharedInstance] playEffect: ”sound.file”];
```

Trên đây là giới thiệu toàn bộ các tính năng và các công cụ hỗ trợ của Cocos2d-iPhone, cho thấy framework này khá nhỏ, chạy rất nhanh và dễ tiếp cận đối với các lập trình viên. Framework đã giúp cho công sức của lập trình viên giảm đi rất nhiều khi muốn xây dựng ứng dụng game trên thiết bị di động thông minh.

Tuy nhiên cũng cho thấy nhược điểm của framework này, đó là: do là một công cụ mã nguồn mở, miễn phí, nên framework chưa có một công cụ đồ hoạ nào đủ mạnh để hỗ trợ lập trình viên trong tất cả mọi công việc. Đối với lập trình game, coding chỉ là một phần rất nhỏ của việc tạo ra game. Các hiệu ứng âm thanh, hình ảnh rất quan trọng. Chính vì thế nhóm triển khai một dự án game cần phải sử dụng thêm công cụ hỗ trợ của hãng thứ 3 để giúp cho việc xây dựng ứng dụng được nhanh hơn và có thể triển khai được các ứng dụng game phức tạp.

CHƯƠNG 4. GẮN QUẢNG CÁO TRONG GAME DI ĐỘNG

Việc làm game trên thiết bị di động cũng như mọi ứng dụng khác trên thiết bị di động thì 90% là nhằm để tìm kiếm thu nhập. Chính vì vậy, việc nghiên cứu các phương án để có thể kiếm được tiền từ các ứng dụng trên thiết bị di động thông minh rất cần thiết. Trong luận văn này, học viên xin trình bày tổng hợp về các phương án tạo thu nhập từ game di động bằng cách gắn quảng cáo.

4.1. Tổng quan

Để thực hiện kiếm thu nhập bằng quảng cáo trên ứng dụng di động ta cần các bước sau:

- Tìm kiếm nhà cung cấp dịch vụ quảng cáo trên di động uy tín
- Đăng ký thành viên
- Sử dụng hàm API của nhà quảng cáo đó, gắn vào ứng dụng của mình, public ứng dụng và chờ khi người sử dụng xem hoặc click vào các quảng cáo của ứng dụng
- Chờ đến khi số tiền kiếm được đặt mức ngưỡng của nhà cung cấp dịch vụ quảng cáo thì nhà cung cấp dịch vụ quảng cáo sẽ chuyển khoản cho lập trình viên.

4.2. Các nhà cung cấp quảng cáo trên di động

Hiện nay có rất nhiều nhà cung cấp dịch vụ quảng cáo trên di động, dưới đây là một số nhà cung cấp trong số họ:

Hình 4.1:Một số nhà cung cấp dịch vụ quảng cáo trên Thiết bị di động

Trong đó các nhà cung cấp dịch vụ quảng cáo trên di động được nhiều lập trình viên Việt Nam tin dùng là: Google Admob, Unity Ads và Opera Mediaworks. Trước đây

lập trình viên Nguyễn Hà Đông cũng sử dụng Google admob trong game Flappy Birds. Tại sao các nhà cung cáp này lại được tin dùng, có một số nguyên nhân chính:

- Họ được nhiều lập trình viên sử dụng làm kênh quảng cáo ứng dụng của họ, vì vậy các ứng dụng quảng cáo trên các kênh này thường phong phú, có chất lượng tốt, qua đó gián tiếp đã kích thích người sử dụng clcik vào các quảng cáo, và mang lại doanh thu tốt cho lập trình viên.
- Uy tín: tình trạng huỷ tài khoản, không trả tiền cho lập trình viên rất ít xảy ra. Một số đơn vị khi lập trình viên kiếm được vài nghìn, hoặc vài chục nghìn $/tháng thì tìm cách huỷ tài khoản của họ, từ chối trả tiền mà không rõ lý do. Quá trình helpdesk sau đó rất phức tạp, thường lập trình viên phải chịu mất doanh thu do công sức của mình tạo ra. Đơn cử như việc Nguyễn Hà Đông trong một tháng có thể đã kiếm được cả triệu USD, nếu nhà cung cấp không uy tín, họ có thể từ chối trả tiền cho Nguyễn Hà Đông, với lý do là một số hình ảnh của game Flappy Birds lấy từ game Mario, nhưng Google Admob đã không làm thế, tạo điều kiện để Hà Đông trở thành triệu phú.
- Tích hợp đơn giản. Hầu như các nhà cung cấp dịch vụ quảng cáo đều cho phép lập trình viên có thể tích hợp thư viện API quảng cáo trong ứng dụng của mình với một vài dòng lệnh đơn giản, tạo nhiều thuận lợi cho lập trình viên.
- Thư viện quảng cáo được tối ưu, chạy nhanh và mượt trong ứng dụng: đây cũng là điều kiện rất quan trọng, bởi lẽ game là ứng dụng sử dụng nhiều bộ nhớ và năng lực xử lý CPU, vì vậy đói hỏi các tiến trình download quảng cáo chạy ngầm phía dưới phải được tối ưu để chạy rất nhanh trong ứng dụng.

Trong quá trình tìm hiểu, học viên đã tích hợp một số ứng dụng của mình với các thư viện quảng cáo của Google Admob và Unity Ads. Dưới đây là một số giới thiệu tổng quan về hai thư viện này:

a. Google Admob [11]:

Admob là một công ty cung cấp giải pháp quảng cáo trên di động, được Google mua lại và trở thành một trong các giải pháp hiệu quả nhất được Google mua. Chính nhờ uy tín là công ty con của Google, đã tạo sự tin cậy rất lớn cho lập trình viên.

Trang web chính thức giới thiệu về giải pháp và hướng dẫn sử dụng tại địa chỉ: https://www.google.com/admob/

Hiện nay thư viện này đang hỗ trợ các nền tảng iOS, Adroid và các platform lập trình game là Unity và Cocos2d, qua đó cũng cho thấy Cocos2d là một thư viện lập trình game rất nổi tiếng.

Để tích hợp thư viện Admob, chúng ta cần đăng ký thành viên của Admob, sử dụng địa chỉ email Google để đăng ký là tiện nhất. Việc đăng ký hoàn toàn miễn phí. Sau

khi đăng ký, người dùng đăng ký một ID cho ứng dụng, và sử dụng nó để tích hợp vào game.

Admob hỗ trợ các hình thức quảng cáo:

- Quảng cáo banner
- Quảng cáo dạng cửa sổ bật ra
- Quảng cáo cho các In-app purchase item trong game của người chơi

Cách thức quảng cáo tương ứng, có thể xem trong các chương phía trước hoặc trong các hướng dẫn của Admob trên trang web chính thức của họ.

b. Unity Ads [12]:

Sau khi công cụ lập trình game Unity ra đời, số lượng lập trình viên sử dụng công cụ này để lập trình game tăng rất nhanh, số lượng ứng dụng game viết bằng unity cũng không ngừng tăng lên. Chính vì thế, nhà phát triển ứng dụng đã tạo ra công cụ Unity Ads để cho phép lập trình viên quảng cáo chính ứng dụng game của họ trên đó, tạo thành một hệ sinh thái khá toàn diện trong Unity:

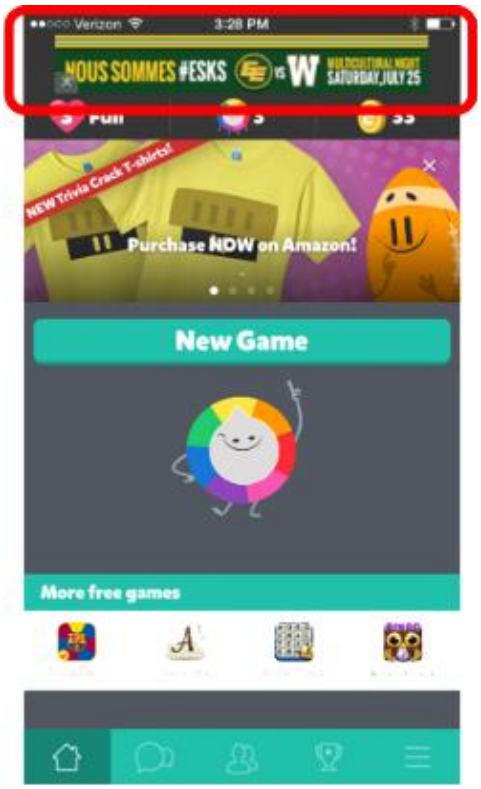
- Lập trình viên có thể lập trình game.
- Bán game trong chợ ứng dụng của Unity
- Bán các công cụ, plugin cho công cụ Unity, hình ảnh, nhân vật, hiệu ứng, thư viện bổ trợ, … trong chợ Asset Store của Unity
- Quảng cáo ứng dụng của mình trên Unit Ads
- Tích hợp ứng dụng của mình với Unit Ads để kiếm tiền từ quảng cáo.

Đây là một hệ sinh thái khá toàn diện cho lập trình viên. Việc tích hợp ứng dụng game được viết bằng Unity với Unity Ads rất đơn giản, chi vài dòng lệnh là đã thực hiện được. Bên cạnh các hình thức quảng cáo mà Admob hỗ trợ, Unity còn hỗ trợ thêm hình thức quảng cáo xem video nhận phần thưởng. Đây là cách mà hiện nay rất nhiều game sử dụng. Lập trình viên có thể yêu cầu người sử dụng xem film (thực chất là các đoạn video quảng cáo cho một ứng dụng khác) để nhận được phần thưởng như đồ vật, thêm lượt chơi, … Khi đó mỗi lần xem, lập trình viên sẽ được một khoản tiền nhỏ.
Học viên đã thử tích hợp Unity Ads với game viết bằng Unity, và nhận thấy việc tích hợp rất đơn giản, dễ hiểu. Tuy nhiên chưa thử tích hợp game viết bằng Cocos2d-iPhone với Unity Ads.

4.3. Các hình thức tích hợp quảng cáo trên di động

a. Banner Ads:

Đây là hình thức đơn giản nhất, trong ứng dụng ta gắn các banner quảng cáo ở header hoặc footer của các màn chơi, hoặc của cửa sổ thông báo.



Hình 4.2:Quảng cáo dạng Banner Ads[3]

Nhà cung cấp dịch vụ quảng cáo sẽ tính tiền cho mỗi lần người sủ dụng view quảng cáo, hoặc click vào quảng cáo. Giá trị của mỗi lần click có thể cao gấp hàng chục, hàng trăm lần mỗi lần view. Giá trị này sẽ thay đổi theo từng khu vực. Ví dụ: 1 click của người dùng ở USA sẽ có giá cao hơn rất nhiều 1 click của người dùng ở Việt Nam.

b. Interstitial Ads:

Đây là hình thức mà quảng cáo được bật ra dưới dạng một của sổ popup mỗi khi người cho kết thúc một lần chơi. Cửa sổ quảng cáo này thường đẹp hơn banner ads, có nhiều thông tin hơn, có thể kèm theo video giới thiệu ứng dụng, chính vì thế nó thường có xác xuất click vào cao hơn. Tuy nhiên đổi lại là nó có thể làm game chạy chậm, giật, nếu không được bố trí thời điểm download quảng cáo thích hợp.

[3] Ảnh được lấy tại: https://firebase.google.com/docs/admob/android/banner

Hình 4.3: Quảng cáo dạng Interstitial Ads[4]

c. In-App Purchase Ads

Là hình thức cho phép lập trình viên quảng cáo chính các item bán trong game. Ví dụ trong game có bán các loại vũ khí, mỗi loại vũ khí có sát thương cao có thể bán với giá 2$, loại khác bán 1$. Thì hình thức In-App Purcharse Ads này cho phép lập trình viên quảng cáo chính các item này.

In-app Purchase có rất nhiều loại, có thể là các item bán trong game, có thể là các nhân vật, có thể là quyền chơi game (với các game đòi hỏi người chơi phải mua quyền chơi mỗi tháng, …) hoặc cũng có thể là hình thức bán quyền chơi bản full của game (ví dụ game cho người chơi bản free với ít chức năng, và có thêm in-app purchase để người chơi có thể chơi bản full đầy đủ chức năng), …

Đây là hình thức Ads hay, nhiều game lớn sử dụng, bởi game của có đủ độ hay, lôi kéo được người sử dụng trả phí cho game.

[4] Ảnh được lấy tại: https://firebase.google.com/docs/admob/android/interstitial

Hình 4.4:Quảng cáo dạng In-app Purchase[5]

d. Reward Video:

Là hình thức là cho phép người sử dụng xem video để nhận phần thưởng. Đây là phương thức quảng cáo thân thiện với người sử dụng, hiện được nhiều lập trình viên sử dụng.

Như chúng ta thấy ở trên, 3 hình thức quảng cáo nêu trên thường khiến người dùng khó chịu bởi nó bật ra trong quá trình chơi hoặc bật ra khi mà không được sự đồng ý của người dùng. Còn Reward Video thường được cài đặt dưới dạng nút bấm, khi người sử dụng click vào nút bấm để nhận phần thưởng như thêm lượt chơi, thêm máu, được đồ vật, thì game sẽ bật ra cửa sổ chạy quảng cáo, người dùng click vào video thì cũng được tính như click vào quảng cáo thông thường.

Reward Vidoe chỉ mới có một số thư viện lập trình game hỗ trợ nhu Unity Ads. Admob hiện vẫn chưa hỗ trợ hình thức này.

[5] Ảnh được lấy tại: https://firebase.google.com/docs/admob/android/iap

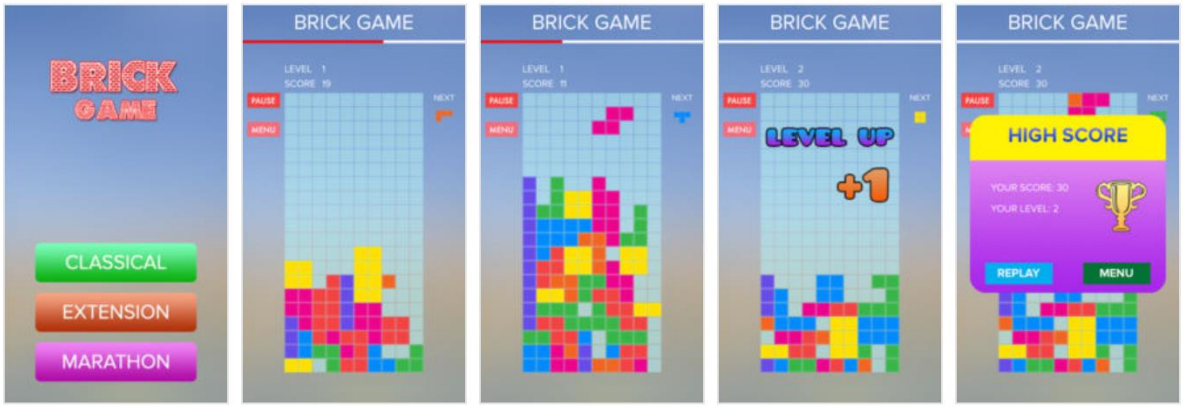
CHƯƠNG 5. SỬ DỤNG FRAMEWORK COCOS2D-IPHONE

Chương này giới thiệu về các game mà học viên đã xây dựng từ việc áp dụng kiến thức thu thập được khi tìm hiểu nền tảng Cocos2d-iPhone. Có tất cả 05 game, đều được public trên AppStore.

5.1. Các game đã được xây dựng từ Cocos2d-iPhone.

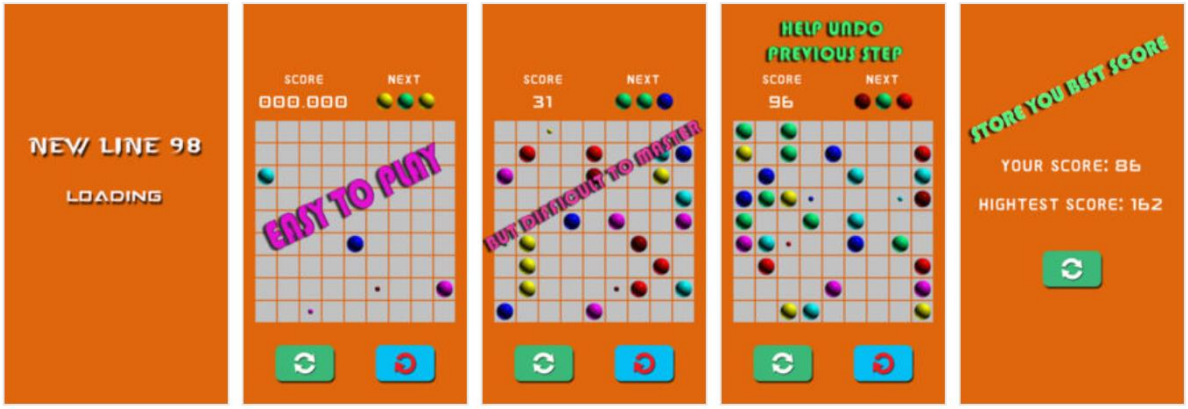
Trong thời gian tìm hiểu lập trình game bằng Cocos2d-iPhone, học viên đã lập trình được 5 game, gồm:

a. Game xếp hình (Tetrix): xây dựng lại một game đã rất quen thuộc với người dùng thế hệ 7x, 8x.



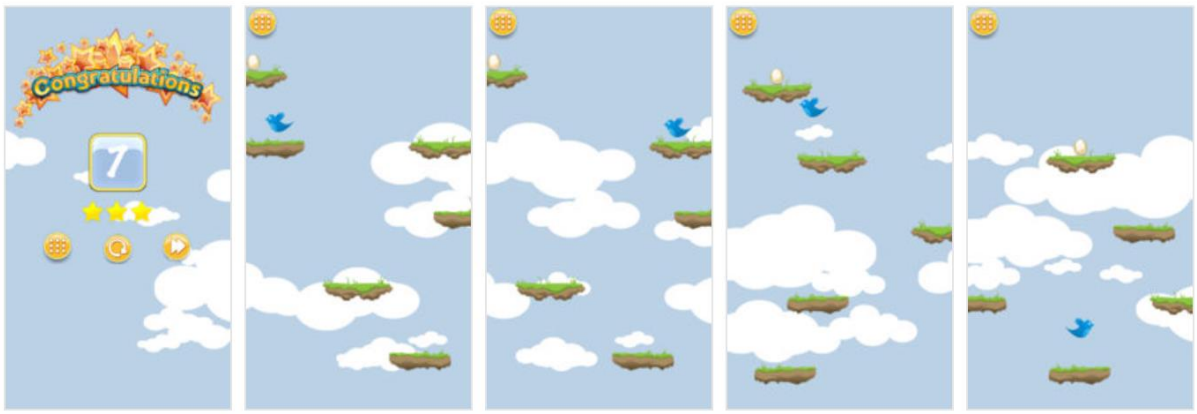
Hình 5.1: Game Tetrix

b. Game Line 98: cũng là một game quen thuộc của thế hệ người dùng 7x, 8x.



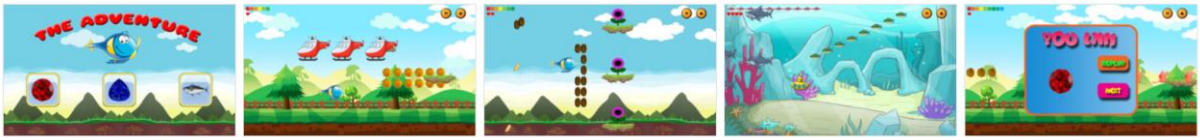
Hình 5.2: Game Line 98

c. Game Recuse Egg: người dùng đóng vai một chú chim, di chuyển lên trên, vượt qua các chướng ngại vật bằng cách lắc tay để tìm quả trứng bị mất.



Hình 5.3: Game Recuse Egg

d. Game Plane: có 3 màn chơi, người dùng đóng vai chiếc máy bay hoặc tàu ngầm, vượt qua các chướng ngại vật để tìm kiếm vật phẩm của từng màn chơi.



Hình 5.4: Game Plane

e. KidGame: game nhỏ, dành cho các em bé từ 4 tuổi trở lên, kéo thả các hình vẽ vào vị trí phù hợp.

Hình 5.5: Game KidGame

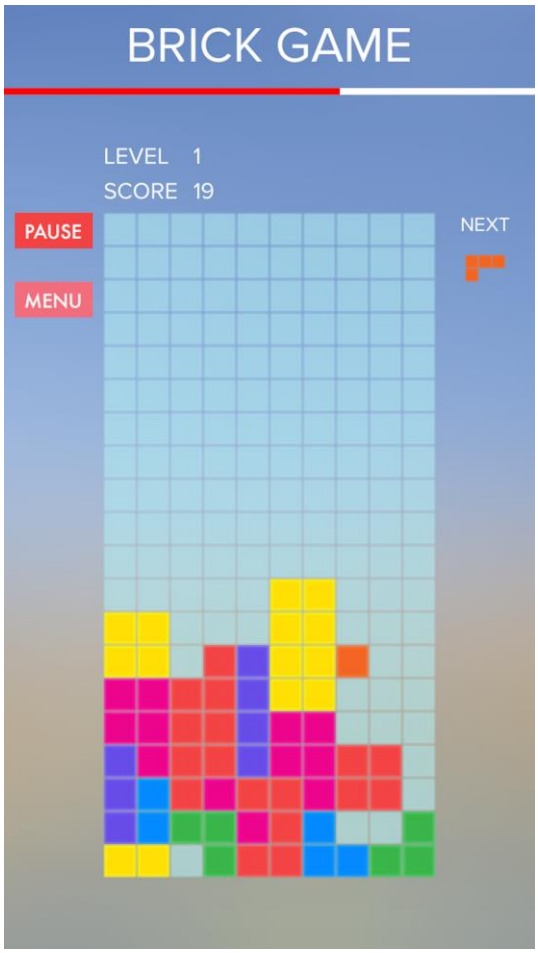
Trong đề tài này, học viên chỉ nêu nội dung của 02 game, đó là game Tetrix và game Plane, vì hai game này có các kỹ thuật cao hơn các game khác như kỹ thuật tạo chuyển động của nhân vật, quảng cáo dạng cửa sổ, nhận sự kiện lắc tay của người sử dụng, áp dụng các chuyển động phức tạp: đồng thời, lặp lại vĩnh viễn, …

5.2. Game 1: Xếp hình (Tetrix)

5.2.1. Các yêu cầu

a. Luật chơi:

- Có một số hình cho trước, được tạo bởi các ô vuông: hình chữ I, chữ L, chữ E, ... được tạo ngẫu nhiên.
- Người chơi sử dụng ngòn tay, vuốt trái, phải, xuống dưới để di chuyển hình vẽ, chạm trên màn hình để xoay hình.
- Người chơi sắp xếp sao cho mỗi hàng không có ô trống, khi đó hàng đó được xoá đi và điểm số tăng lên 1.
- Mỗi khi điểm số tăng lên 30 điểm thì lên 1 level, khi đó tốc độ rơi của hình vẽ sẽ tăng lên.
- Người chơi kết thúc màn chơi khi các hình vẽ chạm trần.



Hình 5.6: Giao diện game Tetrix

5.2.2. Yêu cầu thêm

- Có thêm 5 loại hình khác nhau, trong đó có hình chữ nhật nhỏ (1 ô vuông), hình này có khả năng đi xuyên qua các ô vuông khác để đến ô

vuông trống cuối cùng của hàng, hình này có dấu nhấp nháy trong quá trình chuyển động

- Có thêm hình thức chơi marathon: cố định tốc độ rơi của hình. Người chơi chơi đến lúc game over thì thôi, sau đó có thể so sánh kết quả với các người chơi khác.

5.2.3. Thực hiện yêu cầu

a. Tạo các hình khối khác nhau:

- Được tạo ra bằng cách tạo ra các đối tượng, tuỳ loại đối tượng mà có số lượng đối tượng con khác nhau, và sắp sếp các đối tượng con khác nhau.
- Hình cơ bản nhất là hình chữ nhật, có đơn vị là 1.
- Tiếp theo là các hình khối của game Tetrix thông thường:

Hình 5.7: Các hình khối thông thường

- Để thú vị hơn, học viên đã tạo thêm các hình khối sau:

Hình 5.8: các hình khối mở rộng

- Trong đó hình vuông nhỏ, có khả năng đi xuyên qua tất cả các ô trống trên hàng dọc, đến ô trống cuối cùng. Trong lúc di chuyển, nó sẽ không ngừng nhấp nháy, để nhắc nhở người dùng về tính năng đặc biệt của nó.

b. Tương tác với người dùng:

- Chương trình nhận các hành động vuốt sang trái, phải, đi xuống để di chuyển hình hiện thời sang trái, phải, đi xuống tương ứng.
- Nhận hành động chạm vào màn hình để xoay hình hiện tại.
- Để thực hiện được việc đó, sử dụng các hàm bắt sự kiện của cocos2d-iphone:

- Bắt sự kiện khi chạm vào màn hình:

```
(void)touchBegan:(UITouch *)touch withEvent:(UIEvent *)event;
```

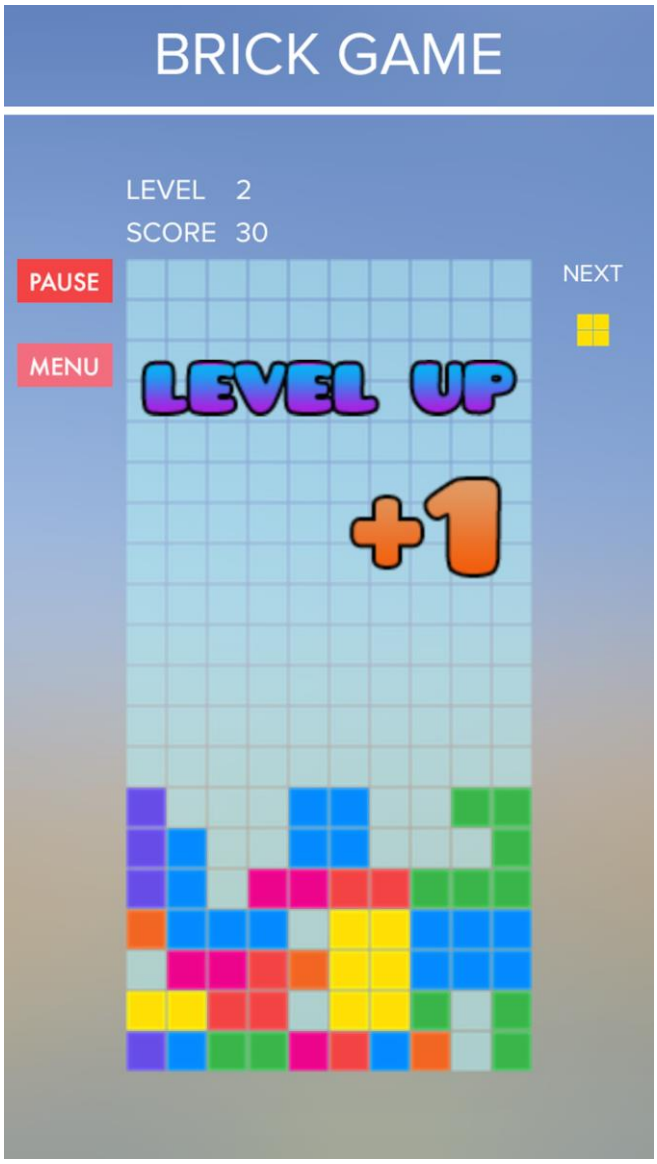
- Bắt sự kiện khi ngòn tay rời khỏi màn hình:

```
(void)touchEnded:(UITouch *)touch withEvent:(UIEvent *)event;
```

- Trong 2 sự kiện trên, ta sẽ lưu trạng thái của điểm tiếp xúc khi bắt đầu chạm màn hình và điểm tiếp xúc cuối cùng trước khi ngón tay rời màn hình. Tính độ lớn của khoảng cách để xác định hành động đó là vuốt sang trái, phải, xuống dưới hay là 1 chạm để có hành động tương ứng với mong muốn của người sử dụng.

c. Thực hiện luật chơi:

- Luật 1: khi một hàng ngang được lấp đầy bởi các ô vuông thì xoá các ô vuông ở hàng đó, các ô vuông ở hàng trên sẽ rơi xuống 1 ô tương ứng:
  Để thực hiện Luật 1, ta cần lưu mảng các trạng thái của các ô vuống trên màn hình. Trong hàm update(), mỗi khi đến thời gian chuyển động của hình thì check mảng này, nếu tồn tại một chỉ số i, mà ở đó mọi giá trị mang[i][j] đều bằng 1 (với mọi j) thì hàng đó đã kín, lấy các ô vuống ở hàng đó và thực hiện lênh remove, cộng điểm và hiển thị hoạt hình cộng điểm cho người dùng biết.
- Luật 2: mỗi 30 điểm ăn được thì lên level: mỗi khi ăn điểm, cộng vào tổng điểm và kiểm tra chia hết cho 30. Nếu chia hết, thì lên level và giảm chu kỳ thời gian chuyển động của hình, hình sẽ rơi nhanh hơn .
- Luật 3: mỗi khi ăn điểm hoặc lên level cần hiển thị điểm ăn được hoặc level lên được, cảnh báo hiện thị phía trên và traong thời gian ngắn sẽ tự mờ đi.



Hình 5.9: Ăn điểm và lên Level

Để làm được như vậy, các đối tượng như số 1 hay chữ "Level Up" cần được gán các Action tương ứng, ở đây gồm các Action:

- CCActionMoveBy: di chuyển đối tượng game đến 1 vị trí xác định trên màn hình
- CCActionFadeOut: mờ dần
- CCActionSpawn: các hành động diễn ra liên tiếp, đầu tiên là di chuyển hình vẽ (số 1 hay chữ Level) nêu trên lên phía đầu của giao diện, sau đó mờ dần nó đi.

- Luật 4: với hình vẽ là đối tượng hình vuông nhỏ, thì cần phải hiện thị nhấp nháy trong khi nó di chuyển. Để làm được việc này cần gán cho đối tượng hình vuông nhỏ đó các chuyển động:

- CCActionBlink: đối tượng sẽ nhấp nháy khi nhận được action này.
- CCActionRepeatForever: lặp lại hành động được gán trong nó mãi mãi (cho đến khi hành đông CCActionRepeatForever bị stop hay remove).

Ngoài các luật nêu trên, trò chơi còn có một số yêu cầu khác, tuy nhiên các yêu cầu này thường dễ thực hiện vì vậy học viên không nêu trong luận văn này, nếu cần tham khảo, xin xem source code đính kèm.

5.2.3. Gắn quảng cáo

Mặc dù Tetrix là một game nổi tiếng, nhiều người biết chơi, nhưng do chưa có sáng tạo nào đáng kể. Vì vậy, việc kiếm doanh thu bằng cách bán game hay bán vật phẩm rất khó, chỉ có thể gắn quảng cáo cho game.

Để thực hiện gắn quảng cáo, học viên lựa chọn cách gắn banner ở đầu của cửa sổ chơi chính. Thư viện quảng cáo được sử dụng là dùng Admob, cách gán banner khá đơn giản:

```
//Ham application này do chương trình tự sinh ra
-(BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
        //Các khai báo khác của chương trình



        //Khoi tao google admob:
    [self createAdmobAds];
}

////////GOOGLE ADMOB:
-(void)createAdmobAds
{
    mBannerType = BANNER_TYPE;

    if(mBannerType <= kBanner_Portrait_Bottom)
        mBannerView = [[GADBannerView alloc] initWithAdSize:kGADAdSizeSmartBannerPortrait];
    else
```

```
    mBannerView = [[GADBannerView alloc]
initWithAdSize:kGADAdSizeSmartBannerLandscape];

    // Specify the ad's "unit identifier." This is your AdMob Publisher ID.

    mBannerView.adUnitID = ADMOB_BANNER_UNIT_ID;



    // Let the runtime know which UIViewController to restore after taking
    // the user wherever the ad goes and add it to the view hierarchy.

    mBannerView.rootViewController = self.navController;
    [self.navController.view addSubview:mBannerView];

    //#ifdef DEBUG
    //    GADRequest *request = [GADRequest request];
    //    request.testDevices = [NSArray arrayWithObjects:GAD_SIMULATOR_ID, nil];
    //#endif



    // Initiate a generic request to load it with an ad.
    [mBannerView loadRequest:[GADRequest request]];

    CGSize s = [[CCDirector sharedDirector] viewSize];

    CGRect frame = mBannerView.frame;

    off_x = 0.0f;
    on_x = 0.0f;

    switch (mBannerType)
    {
        case kBanner_Portrait_Top:
```

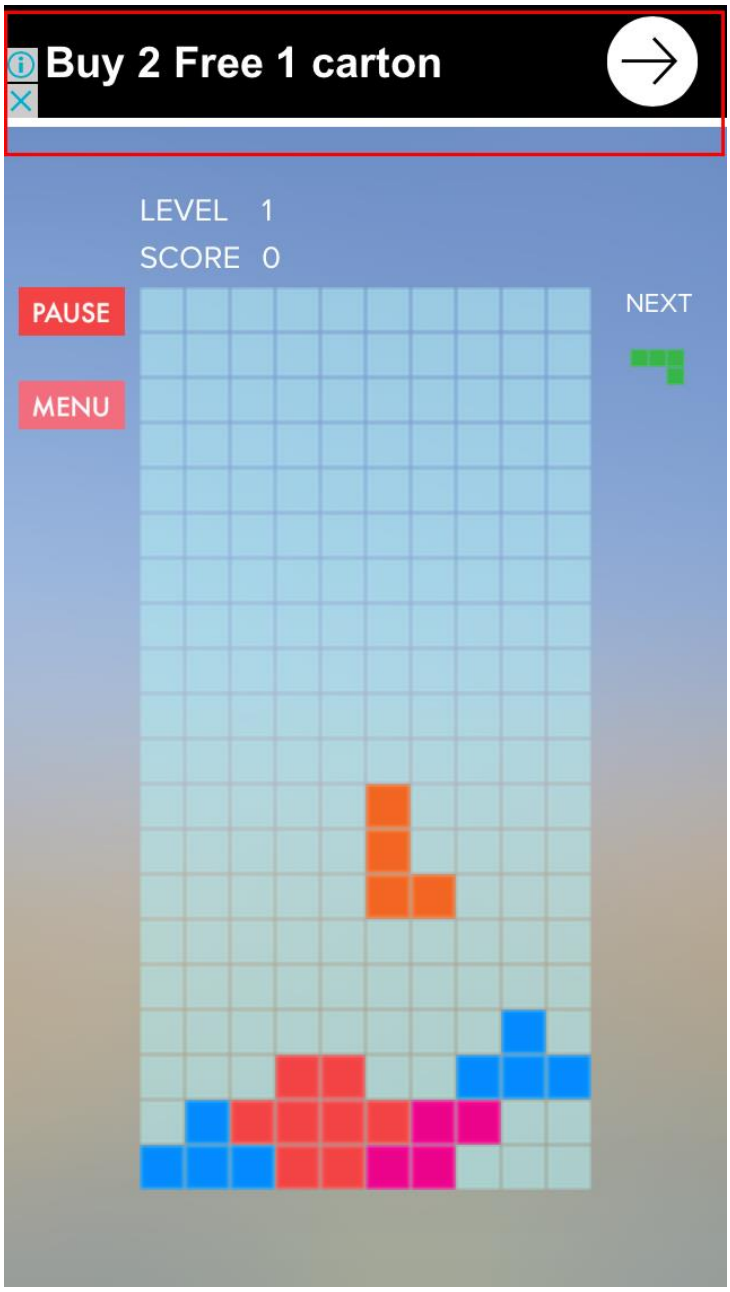
```
    {
        off_y = -frame.size.height;
        on_y = 0.0f;
    }
        break;
    case kBanner_Portrait_Bottom:
    {
        off_y = s.height;
        on_y = s.height-frame.size.height;
    }
        break;
    case kBanner_Landscape_Top:
    {
        off_y = -frame.size.height;
        on_y = 0.0f;
    }
        break;
    case kBanner_Landscape_Bottom:
    {
        off_y = s.height;
        on_y = s.height-frame.size.height;
    }
        break;

    default:
        break;
}

frame.origin.y = off_y;
frame.origin.x = off_x;

mBannerView.frame = frame;
```

```
    [UIView beginAnimations:nil context:nil];
    [UIView setAnimationDuration:0.5];
    [UIView setAnimationCurve:UIViewAnimationCurveEaseOut];

    frame = mBannerView.frame;
    frame.origin.x = on_x;
    frame.origin.y = on_y;


    mBannerView.frame = frame;
    [UIView commitAnimations];
}


-(void)showBannerView
{
    if (mBannerView)
    {
        //banner on bottom
        {
            CGRect frame = mBannerView.frame;
            frame.origin.y = off_y;
            frame.origin.x = on_x;
            mBannerView.frame = frame;


            [UIView animateWithDuration:0.5
                                  delay:0.1
                                options: UIViewAnimationCurveEaseOut
                             animations:^
             {
                CGRect frame = mBannerView.frame;
                frame.origin.y = on_y;
```

```
            frame.origin.x = on_x;

            mBannerView.frame = frame;
        }
                completion:^(BOOL finished)
        {
        }];
    }
  }
}
-(void)hideBannerView
{
  if (mBannerView)
  {
    [UIView animateWithDuration:0.5
                  delay:0.1
                options: UIViewAnimationCurveEaseOut
              animations:^
     {
        CGRect frame = mBannerView.frame;
        frame.origin.y = off_y;
        frame.origin.x = off_x;
        mBannerView.frame = frame;
     }
                completion:^(BOOL finished)
     {

     }];
  }
}
-(void)dismissAdView
{
  if (mBannerView)
```

```
    {
        [UIView animateWithDuration:0.5
                              delay:0.1
                            options: UIViewAnimationCurveEaseOut
                         animations:^
        {
            CGSize s = [[CCDirector sharedDirector] viewSize];

            CGRect frame = mBannerView.frame;
            frame.origin.y = frame.origin.y + frame.size.height ;
            frame.origin.x = (s.width/2.0f - frame.size.width/2.0f);
            mBannerView.frame = frame;
        }
                         completion:^(BOOL finished)
        {
            [mBannerView setDelegate:nil];
            [mBannerView removeFromSuperview];
            mBannerView = nil;

        }];
    }
}
```

Kết quả thu được như hình vẽ dưới đây:



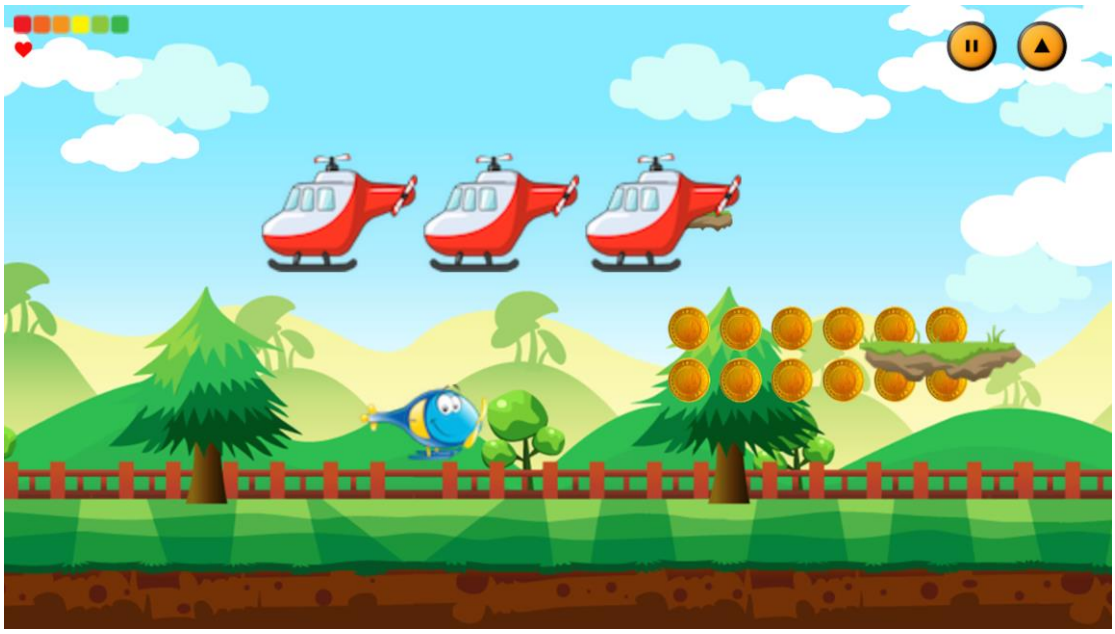
Hình 5.10: Gắn quảng cáo dạng Banner

5.2.4. Nhận xét

Với thư viện lập trình game Cocos2d-iPhone, việc lập trình game Tetrix khá đơn giản, chương trình chỉ có khoảng 5000 dòng code. Nếu không sử dụng thư viện này, mà thực hiện lập trình từ đầu, thì rất phức tạp, chỉ với việc thực hiện các chuyển động hay hiệu ứng mờ dần của đối tượng cũng đòi hỏi lập trình viên phải tạo nhiều thread để điều khiển. Việc quản lý multi thread trong chương trình thường rất khó, mất nhiều công sức và kinh nghiệm để thực hiện. Qua đó cho thấy tác dụng lớn của việc sử dụng các thư viện lập trình game.

Trong chương trình tetrix, học viên chưa áp dụng các module phức tạp của Cocos2d-iPhone như hiệu ứng vật lý, hay hiệu ứng đặc biệt, nhưng đã có thể hoàn thành 1 game nổi tiếng. Nếu nghiên cứu, áp dụng thêm các hiệu ứng đặc biệt sẽ làm game hấp dẫn hơn.

5.3. Game 2: Plane



Hình 5.11: Giao diện game Plane

5.3.1. Các yêu cầu

- Có 3 màn chơi, trong đó người sử dụng điều khiển máy bay tránh các vật cản (máy bay khác, các con chim, đạn bắn từ các bông hoa, cá mập, ...) và ăn các đồng tiền dọc đường đi để có đủ năng lượng chạy đến cuối màn chơi.
- Người sử dụng điều khiển nhân vật (máy bay hay tàu ngầm) bằng cách chạm vào màn hình, di chuyển ngón tay để nhân vật chạy theo, hoặc dùng cách lắc điện thoại, để nhân vật di chuyển theo.
- Các đối tượng trong game sẽ có các hoạt hình tương ứng trong khi di chuyển, ví dụ máy bay chạy thì cánh quạt quay, chim bay, đồng tiền (coin) xoay.
- Có kiểm tra các va chạm, mỗi khi nhân vật va chạm vào kẻ thù (máy bay khác, chim, cá mập hay đạn) thì máu giảm một giọt. Khi hết máu thì game over.
- Gắn quảng cáo dạng cửa sổ bật ra mỗi khi người chơi kết thúc màn chơi hoặc bị game over.

5.3.2. Thực hiện yêu cầu

a. Nhận các sự kiện tác động của người dùng:

- Với sự kiện chạm và di chuyển ngón tay của người dùng trên màn hình: tương tự như game Tetrix ta thực hiện bắt trên các hàm TouchBegin và TouchEnd.
- Với sự kiện lắc điện thoại, nghiên để điều khiển nhân vật:

////////////////

```
//Xu ly su kien khi nguoi dung lac tay:
- (void)accelerometer:(UIAccelerometer*)accelerometer
didAccelerate:(UIAcceleration*)acceleration {
}
//Hai su kien xu ly su kien lac tay:
- (void)onEnter
{
    [super onEnter];
    [_motionManager startAccelerometerUpdates];
}
- (void)onExit
{
    [super onExit];
    [_motionManager stopAccelerometerUpdates];
}
```

b. Tạo các hình ảnh động cho các đối tượng trong game

- Để tạo được hình ảnh động cho các đối tượng trong game, cách đơn giản nhất là có được một tập các ảnh liên tiếp nhau, mô tả hành động trong game, sau đó khai báo và gắn chúng với nhau trong một Animation Frame, đoạn code như sau:

```
- (id) init {
//Tạo hình ảnh đầu tiên:

        sprite = [CCSprite spriteWithImageNamed:@"h0.png"];

// Tạo thêm các hình ảnh khác, và gắn vào Frame
    NSMutableArray* spriteframes = [NSMutableArray array];
    for(int i = 0; i < 14; i++){
        NSString *str = [NSString stringWithFormat:@"h%d.png", i];
        CCSpriteFrame *frame = [CCSpriteFrame frameWithImageNamed:str];
        [spriteframes addObject:frame];
    }
```

```
    //Tao animatin:
    CCAnimation *animation = [CCAnimation animationWithSpriteFrames:spriteframes delay:0.01f];
    CCActionAnimate *animationAction = [CCActionAnimate actionWithAnimation:animation];
    CCActionRepeatForever *action = [CCActionRepeatForever actionWithAction: animationAction];
    [action setTag:100];
    [sprite runAction:action];
    return self;
}
```

Hình 5.12: Giao diện game Plane

c. Va chạm

Để kiểm tra va chạm ta có thể cài đặt theo 2 cách:

- Cách 1, dùng các thư viện mô phỏng hiện tượng vật lý của Cocos2D, như trong Chương II. Phương án này có nhược điểm là hiệu năng kém, vì kiểm tra va chạm và mô phỏng khá khó.
- Cách 2, đơn giản chỉ cần kiểm tra khoảng cách giữa hai anchor point, nếu khoảng cách đó mà nhỏ hơn một nửa tổng chiều dài thì đó là va chạm.

Học viên lựa chọn cách thứ 2. Với cách này, mã code sẽ đơn giản như sau:

```
- (BOOL) checkVacham:(CCSprite*) sprite{
```

```
    CGPoint postHero = heroSprite.position;
    CGSize sizeHero = CGSizeMake(heroSprite.contentSize.width *fabs(heroSprite.scaleX)/2, heroSprite.contentSize.height*fabs(heroSprite.scaleY)/2);
    CGRect rectHero = CGRectMake(postHero.x - sizeHero.width, postHero.y - sizeHero.height, 2*sizeHero.width, sizeHero.height);

    CGPoint postSprite = sprite.position;
    CGSize sizeSprite = CGSizeMake(sprite.contentSize.width * fabsf(sprite.scaleX)/2, sprite.contentSize.height * fabsf(sprite.scaleY)/2);
    CGRect rectSprite = CGRectMake(postSprite.x - sizeSprite.width, postSprite.y - sizeSprite.height, 2 * sizeSprite.width, 2 * sizeSprite.height);

    if(!sprite.visible) return false; //Dang bien mat
    return CGRectIntersectsRect(rectHero, rectSprite);
}
```

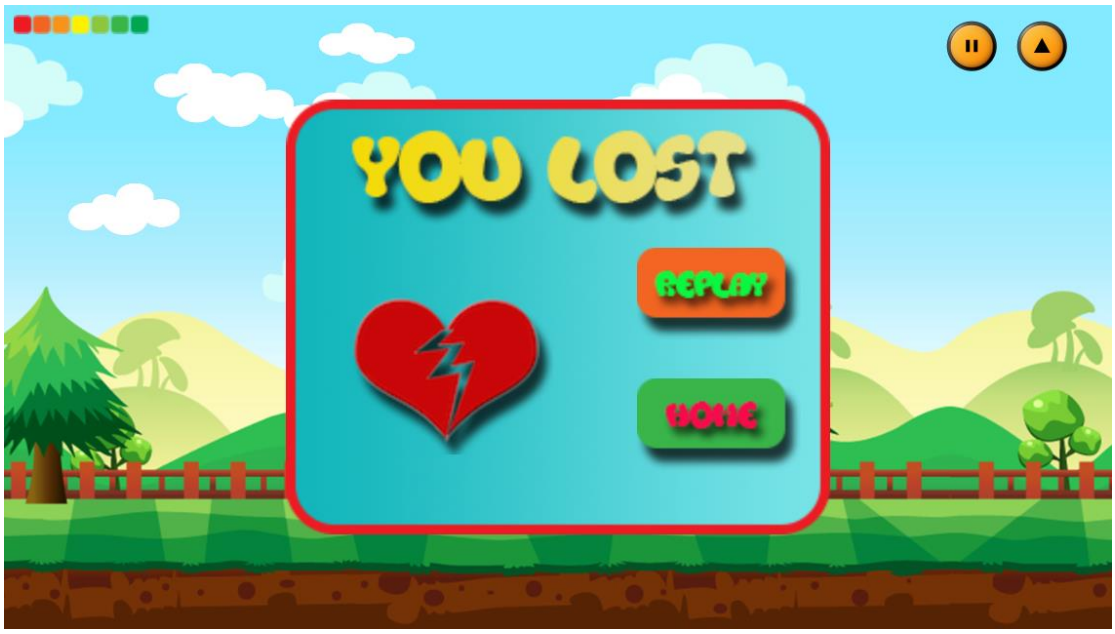
5.3.3. Gắn quảng cáo

Học viên lựa chọn phương án gắn quảng cáo là sử dụng thư viện API của Google Admob và cách gắn quảng cáo là dùng cửa số bật ra mỗi khi game over hoặc khi người chơi hoàn thành một màn chơi. Để làm cách này, ta sử dụng class GADInterstitial của thư viện Admob. Cách thực hiện như sau:

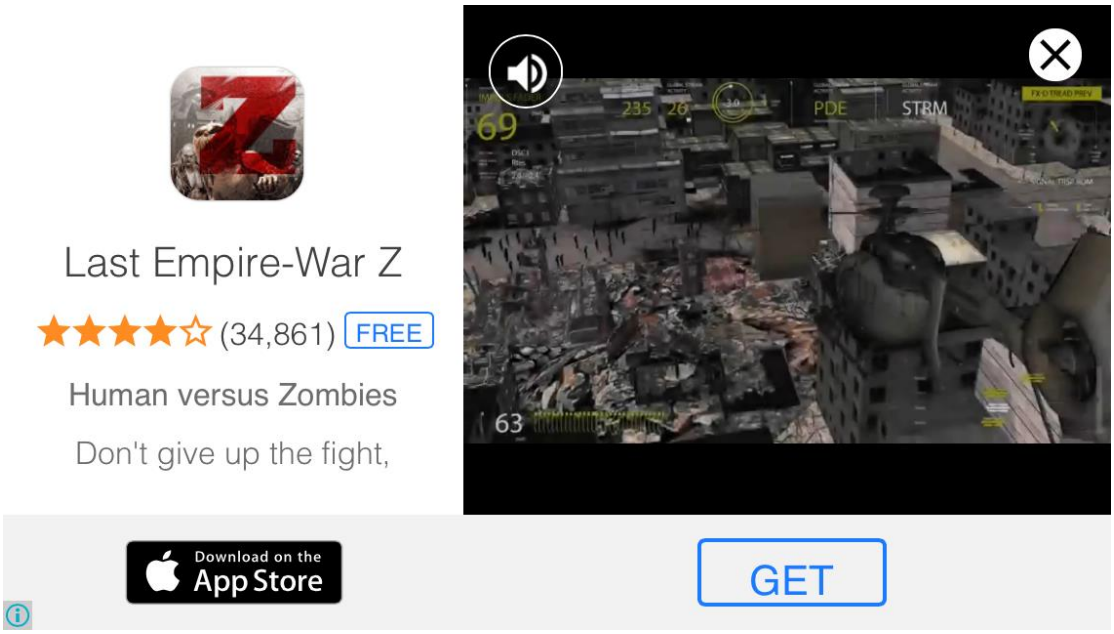
```
//Tao Admob
- (void)preLoadInterstitial {
    //Call this method as soon as you can - loadRequest will run in the background and your interstitial will be ready when you need to show it
    GADRequest *request = [GADRequest request];
    interstitial_ = [[GADInterstitial alloc] initWithAdUnitID:@"ca-app-pub-1903138713698660/3686046XXX"];
    interstitial_.delegate = self;
    [interstitial_ loadRequest:request];
}

- (void)interstitialDidDismissScreen:(GADInterstitial *)ad
{
```

```
    //An interstitial object can only be used once - so it's useful to automatically load a new one when the current one is dismissed
    [self preLoadInterstitial];
}

- (void)interstitial:(GADInterstitial *)ad didFailToReceiveAdWithError:(GADRequestError *)error
{
    //If an error occurs and the interstitial is not received you might want to retry automatically after a certain interval
    [NSTimer scheduledTimerWithTimeInterval:2.0f target:self selector:@selector(preLoadInterstitial) userInfo:nil repeats:NO];
}

- (void) showInterstitial
{
    //Call this method when you want to show the interstitial - the method should double check that the interstitial has not been used before trying to present it
    if (!interstitial_.hasBeenUsed) [interstitial_ presentFromRootViewController:[CCDirector sharedDirector].navigationController];
}
```

Mỗi khi cần hiện thị quảng cáo thì gọi hàm showInterstitial. Kết quả thu được như hình dưới:



Hình 5.13: Giao diện trước khi bật cửa sổ quảng cáo



Hình 5.14: Giao diện khi bật quảng cáo

5.3.4. Nhận xét

Trong game thứ hai, học viên đã sử dụng được các kỹ thuật:

- Bắt sự kiện tương tác với người dùng qua hai phương án: chạm màn hình và lắc màn hình.
- Tạo hoạt hình cho các nhân vật từ một tập ảnh cho trước.
- Tạo map bằng cách sinh ngẫu nhiên các đối tượng trong game (các đối tượng backgroud, máy bay, hoa, cá mập, cá, ...)
- Xử lý va chạm
- Quảng cáo dạng cửa sổ bật ra khi người chơi kết thúc một phiên chơi.

CHƯƠNG 6. KẾT LUẬN

Đề tài này đã tóm tắt lại các kiến thức mà học viên đã thu thập được trong quá trình nghiên cứu làm game cho di động, qua đó giúp người đọc có cái nhìn tổng quát về quá trình làm game, cách thức tìm kiếm thu nhập và một vài so sánh giữa các framwork lập trình game phổ biến hiện nay. Sử dụng framework Cocos2d-iPhone, học viên đã xây dựng được một số ứng dụng và public lên kho ứng dụng AppStore của Apple, qua đó giúp học viên kiếm được một số thu nhập nhất định. Việc sử dụng framework Cocos2d-iPhone giúp việc lập trình các game cho điện thoại di động thông minh nền iOS trở lên dễ dàng hơn rất nhiều, không đòi hỏi lập trình viên phải tìm hiểu các kỹ thuật phức tạp như multithread hay hiển thị đồ họa phức tạp, rút ngắn thời gian phát triển ứng dụng game.

Các framework lập trình game ngày càng phát triển, tạo rất nhiều thuận lợi cho lập trình viên. Tuy nhiên nó cũng dẫn đến số lượng ứng dụng game ngày càng nhiều, làm tăng mức độ cạnh tranh của các ứng dụng với nhau, các ứng dụng muốn thành công cần phải được chau chuốt hơn, chỉnh chu hơn, có nội dung hay và đồ họa đẹp hơn. Như vậy, cơ hội cho các lập trình viên đơn lẻ cũng càng ngày càng hẹp hơn, những cá nhân như Nguyễn Hà Đông ngày càng ít. Để tạo được một game di động thành công, đòi hỏi phải có một nhóm làm việc, mà trong đó ít nhất phải có các cá nhân có kỹ năng đồ họa, âm thanh, xây dựng kịch bản game, …, và lập trình viên cũng không còn đóng vai trò quyết định trong việc xây dựng ứng dụng game nữa.

Với số lượng ứng dụng trên các kho ứng dụng AppStore của Apple, PlayStore của Google đã lên đến con số hàng triệu. Việc một ứng dụng muốn thành công, ngoài bản thân nội dung của ứng dụng, còn có rất nhiều yếu tố ảnh hưởng đến nó, trong đó có một yếu tố quan trọng đó là tối ưu từ khóa trong việc đặt tên ứng dụng, mô tả ứng dụng để tăng cơ hội người dùng tìm thấy ứng dụng của mình trên các chợ ứng dụng. Đây chính là hướng nghiên cứu của học viên trong thời gian tới.

TÀI LIỆU THAM KHẢO

1. Tổng quan về Cocos2d: http://cocos2d.org
2. Unity: https://unity3d.com/
3. SpriteKit: https://developer.apple.com/spritekit/
4. GameMaker: http://www.yoyogames.com/gamemaker
5. Contruct2: https://www.scirra.com/construct2
6. Buildbox: https://www.buildbox.com/
7. Unreal: https://www.unrealengine.com/what-is-unreal-engine-4
8. Cocos2d-iPhone SDK
9. Spine: http://esotericsoftware.com/
10. Particle Designer: https://71squared.com/particledesigner
11. Admob: https://www.google.com/admob/
12. Unity Ads: https://unity3d.com/services/ads