**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC CẦN THƠ**

**KHOA CÔNG NGHỆ THÔNG TIN & TRUYỀN THÔNG**

**LUẬN VĂN TỐT NGHIỆP ĐẠI HỌC**

**NGÀNH TRUYỀN THÔNG & MẠNG MÁY TÍNH**

**Đề tài**

**LẬP TRÌNH GAME TRÊN ANDROID**

**VỚI CORONA SDK**

**GAME: MONSTER RUN**

**Sinh viên: Tô Trí Khương**

**Mã số: B1203935**

**Khóa: K38**

**Cần Thơ, 11/2016**

**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC CẦN THƠ**

**KHOA CÔNG NGHỆ THÔNG TIN & TRUYỀN THÔNG**

**BỘ MÔN MẠNG MÁY TÍNH & TRUYỀN THÔNG**

**LUẬN VĂN TỐT NGHIỆP ĐẠI HỌC**

**NGÀNH TRUYỀN THÔNG & MẠNG MÁY TÍNH**

**Đề tài**

**LẬP TRÌNH GAME TRÊN ANDROID**

**VỚI CORONA SDK**

**GAME: MONSTER RUN**

**Người hướng dẫn Sinh viên thực hiện**

**TS. Trần Công Án Tô Trí Khương**

**Mã số: B1203935**

**Khóa: K38**

***Cần Thơ, 11/2016***

# LỜI CẢM ƠN

Lời đầu tiên, tôi xin chân thành gửi lời cảm ơn và sự tri ân sâu sắc đến các thầy cô trường Đại học Cần Thơ. Đặc biệt là các thầy cô khoa Công Nghệ Thông Tin và Truyền Thông đã tạo điều kiện cho tôi học tập rèn luyện và hoàn thành khoá học, cũng như luận văn tốt nghiệp này.

Bên cạnh đó, tôi cũng xin gửi lời cảm ơn chân thành đến thầy Trần Công Án đã nhiệt tình hướng dẫn, chỉ bảo những thiếu sót, giúp tôi hoàn thành luận văn tốt nghiệp trong thời gian qua.

Đồng thời tôi xin cảm ơn tất cả bạn bè người thân đã động viên, tạo điều kiện cho tôi hoàn thành tốt luận văn này.

Trong thời gian học tập, cũng như trong quá trình làm luận văn, mặc dù đã cố gắng hết sức nhưng có thể vẫn còn sai sót, vì vậy rất mong nhận được sự đóng góp ý kiến quý báu của quý Thầy Cô và các bạn.

Tôi xin chân thành cảm ơn!

Cần Thơ ngày tháng 11 năm 2016

Sinh viên

**Tô Trí Khương**

# MỤC LỤC

[LỜI CẢM ƠN 3](#_Toc450563989)

[MỤC LỤC 4](#_Toc450563990)

[DANH MỤC HÌNH 6](#_Toc450563991)

[DANH MỤC BẢNG 7](#_Toc450563992)

[TÓM TẮT 8](#_Toc450563993)

[ABSTRACT 9](#_Toc450563994)

[PHẦN GIỚI THIỆU 1](#_Toc450563995)

[1. ĐẶT VẤN ĐỀ 10](#_Toc450563996)

[2. TÓM TẮT LỊCH SỬ GIẢI QUYẾT VẤN ĐỀ 11](#_Toc450563997)

[3. MỤC TIÊU ĐỀ TÀI 11](#_Toc450563998)

[4. ĐỐI TƯỢNG VÀ PHẠM VI NGHIÊN CỨU 11](#_Toc450563999)

[5. NỘI DUNG NGHIÊN CỨU 11](#_Toc450564000)

[6. NHỮNG ĐÓNG GÓP CHÍNH CỦA ĐỀ TÀI 12](#_Toc450564001)

[7. BỐ CỤC CỦA QUYỂN LUẬN VĂN 13](#_Toc450564002)

[PHẦN NỘI DUNG 14](#_Toc450564003)

[Chương 1 - MÔ TẢ BÀI TOÁN 14](#_Toc450564004)

[1. MÔ TẢ CHI TIẾT BÀI TOÁN 14](#_Toc450564005)

[2. PHÂN TÍCH ĐÁNH GIÁ CÁC GIẢI PHÁP CÓ LIÊN QUAN ĐẾN BÀI TOÁN 15](#_Toc450564006)

[3. TIẾP CẬN GIẢI QUYẾT VẤN ĐỀ 15](#_Toc450564007)

[Chương 2 - THIẾT KẾ GIẢI PHÁP 17](#_Toc450564008)

[1. THIẾT KẾ KIẾN TRÚC 17](#_Toc450564009)

[2. THIẾT KẾ DỮ LIỆU 19](#_Toc450564010)

[3. THIẾT KẾ GIAO DIỆN 20](#_Toc450564011)

[4. THIẾT KẾ THUẬT TOÁN 23](#_Toc450564012)

[Chương 3 - CÀI ĐẶT GIẢI PHÁP 30](#_Toc450564013)

[1. CÀI ĐẶT KIẾN TRÚC 30](#_Toc450564014)

[2. CÀI ĐẶT DỮ LIỆU 30](#_Toc450564015)

[3. XÂY DỰNG GIAO DIỆN 35](#_Toc450564016)

[Chương 4 - KIỂM THỬ VÀ ĐÁNH GIÁ 40](#_Toc450564017)

[1. MỤC TIÊU KIỂM THỬ 40](#_Toc450564018)

[2. KỊCH BẢN KIỂM THỬ 40](#_Toc450564019)

[3. KẾT QUẢ KIỂM THỬ 41](#_Toc450564020)

[PHẦN KẾT LUẬN 42](#_Toc450564021)

[1. KẾT QUẢ ĐẠT ĐƯỢC 42](#_Toc450564022)

[2. HẠN CHẾ 42](#_Toc450564023)

[3. HƯỚNG PHÁT TRIỂN 42](#_Toc450564024)

[TÀI LIỆU THAM KHẢO 43](#_Toc450564025)

[PHỤ LỤC 44](#_Toc450564026)

[1. HƯỚNG DẪN CÀI ĐẶT CORONA SDK TRÊN WINDOWS 44](#_Toc450564027)

[2. HƯỚNG DẪN SỬ DỤNG 44](#_Toc450564028)

[3. SƠ LƯỢC VỀ NGÔN NGỮ LUA 47](#_Toc450564029)

[4. SƠ LƯỢC VỀ CORONA SDK 48](#_Toc450564030)

# DANH MỤC HÌNH

Hình 1.1: Sơ đồ Game flow cơ bản. 16

Hình 2.1: Hệ thống phân cấp project cơ bản 17

Hình 2.2: Sơ đồ phân cấp các chức năng của phần mềm. 18

Hình 2.5: Sơ đồ phân cấp dữ liệu dùng trong game 19

Hình 2.6: Thiết kế giao diện chính của trò chơi. 20

Hình 2.7: Thiết kế giao diện trợ giúp người chơi. 20

Hình 2.8: Thiết kế giao diện Chọn màn chơi. 21

Hình 2.9: Thiết kế giao diện đang trong trò chơi. 21

Hình 2.10: Thiết kế giao diện Kết thúc màn chơi. 22

Hình 2.11: Thiết kế giao diện Hoàn thành màn chơi. 22

Hình 2.12: Thuật toán tạo một địa hình cho màn chơi. 24

Hình 2.13: Thuật toán tạo cạm bẫy và quái vật ngẫu nhiên trong màn chơi. 25

Hình 2.14: Thuật toán “cuộn” địa hình trong khi chơi. 26

Hình 2.15: Thuật toán kiểm tra va chạm trong trò chơi. 27

Hình 2.16: Thuật toán sự kiện chạm button trong trò chơi. 28

Hình 2.17: Thuật toán sự kiện kết thúc trò chơi. 29

Hình 2.18 : Loạt hình ảnh tạo nên chuyển động chạy của nhân vật chính sau khi xuất ra tập tin hình ảnh PNG. 35

Hình 2.19: Hình ảnh quái vật chuyển động 35

Hình 2.20: Hình ảnh tạo ra các quái vật gai nhọn. 36

Hình 2.21: Hình ảnh cách tạo ra nhiều quái vật gai nhọn. 36

Hình 2.22: Hình ảnh các mảnh vuông tạo nên địa hình trong game. 36

Hình 2.23: Giao diện Menu. 37

Hình 2.24: Giao diện Help. 37

Hình 2.25:Giao diện Settings. 37

Hình 2.26: Giao diện Chọn màn chơi. 38

Hình 2.27: Giao diện trong trò chơi 38

Hình 2.28: Giao diện Game Over. 39

Hình 2.29: Giao diện Game Win. 39

# DANH MỤC BẢNG

Bảng 2.2: Bảng diễn giải các dữ liệu dùng trong Game 19

Bảng 2.3: Bảng danh sách tên các thuật toán có trong trò chơi. 23

Bảng 3.1: Danh sách các đối tượng trong game "MonsterRun". 31

Bảng 4.1: Kết quả kiểm thử theo từng chức năng. 41

# TÓM TẮT

Ngày nay, với tốc độ phát triển nhanh đến chóng mặt của các thiết bị thông minh trong vài năm trở lại đây, một môi trường làm việc đầy tiềm năng đã ra đời và ngày càng lôi cuốn nhiều người tham gia, đó chính là môi trường ứng dụng di động. Môi trường này có tốc độ phát triển nhanh, mạnh mẽ và đặc biệt là trong lĩnh vực lập trình game trên di động, và đó cũng chính là vùng đất màu mỡ cho các nhà lập trình viên đam mê về game trên di động. Luận văn tốt nghiệp là công trình nghiên cứu về Game Engine - Một bộ công cụ cung cấp môi trường phát triển tích hợp sử dụng các thư viện hổ trợ phát triển game trên các thiết bị di động, đặc biệt phải nói đến đó chính là công cụ Corona SDK, một nền tảng phần mềm linh hoạt và dễ sử dụng lại mới mọi chức năng cốt lỗi cần thiết ngay bên trong nó. Sản phẩm của luận văn là một game (Monster Run) chạy được trên các thiết bị di động, máy tính bảng chạy hệ điều hành Android. Game sử dụng ngôn ngữ LUA và công cụ phát triển Corona SDK, theo cách lập trình hướng dữ liệu, game bao gồm 5 màn chơi được xếp từ mức độ dễ đến khó, giao diện gần gũi, đối tượng đồ hoạ chân thật, hiệu ứng vui nhộn, âm thanh tương tác trực quan, lôi cuốn. Tuy ban đầu game chỉ chạy được trên nền tảng Android, nhưng trong tương lai, game sẽ tiếp tục phát triển và có mặt trên cả những hệ điều hành thông dụng hiện nay là iOS và Windows Phone.

# ABSTRACT

Nowaday, with the rapid growth of the dizzying smart devices in the past few years, a potential market was born and increasingly attracted many people take park in , it is the mobile app market . This market growth is fast, powerful network and especially in the field of mobile game programming, and it is also a fertile ground for developers passionate about games on mobile. Thesis is the study of the Game Engine - A toolkit provides integrated development environment using the library supports game development on mobile devices, especially to say to that is Corona SDK tool, a flexible software and ease of use to all the core functionality needed right inside it. Products of the thesis is a game (Monster Run) run on mobile devices, tablets running the Android operating system. The game uses language LUA and Corona SDK development tools, follow programming in a data-driven manner, the game includes 5 levels are ranked from easy to difficult , user friendly interface, graphical objects honest, funny effect, audio visual, interactive appealing. But the original game only run on the Android platform, but in the future, the game will continue to grow and can run on all operating systems now commonly used is iOS and Windows Phone.

# PHẦN GIỚI THIỆU

### ĐẶT VẤN ĐỀ

Trong sự phát triển vượt bật của smartphone, sự ra đời và lên ngôi của game di động là điều tất yếu. Tình trạng này diễn ra rất rõ tại Việt Nam khi mà tổng số lượng thiết bị smartphone chiếm tới hơn 50% tổng dân số Việt Nam trong năm 2016. Các tựa game hấp đẫn trên các thiết bị di động ra đời với số lượng ngày càng nhiều và phong phú về nội dung cũng như thể loại. Trong xu thế đó lập trình game di động phát triển một cách vượt bật, mở ra một thị trường làm việc hấp dẫn thu hút đối với các lập trình viên. Việc tiếp cận và lập trình game di động cũng trở nên dễ dàng hơn nhờ những bộ công cụ phát triển ứng dụng – game di dộng như Unity3D, Cocos2d-x, Game Salad, Corona SDK,…

Corona SDK là một công cụ phát triển game đa nền tảng, phát triển ứng dụng nhanh, thế nên nó thường được sử dụng để phát triển game 2D trên các thiết bị di động. Ngoài ra, nó còn cung cấp hệ thống thư viện phong phú, hổ trợ đầy đủ và được thiết kế rất tốt. Do đó các nhà phát triển phần mềm có thể dễ dàng xây dựng các thao tác (hành động) trong game chỉ với một vài dòng code đơn giản. Với công cụ Corona SDK, chúng ta chỉ cần viết code một lần, sau đó ta có thể nhanh chóng biên dịch dễ dàng ứng dụng sang các nền tảng khác nhau như Android, iOS, Kindle Fire, Nook và cả Windows Phone. Không chỉ vậy, Corona SDK còn hỗ trợ chuyển đổi với các thiết bị có tỉ lệ, kích cỡ và độ phân giải khác nhau một cách dễ dàng.

Từ những nội dung trên, tác giả đã đề xuất đề tài **"PHÁT TRIỂN GAME TRÊN THIẾT BỊ DI ĐỘNG SỬ DỤNG FRAMEWORK CORONA SDK".** Game cụ thể sẽ được phát triển dựa trên ý tưởng được trình bày ở trên (Truy đuổi quái vật) có tên là ***"MonsterRun".***

Game *“Monster Run”* lấy ý tưởng từ thể loại game huyền thoại - trò chơi điện tử thể loại "chạy không ngừng – Mario" để vượt qua các chướng ngại vật và tiêu diệt các quái vật trên đường nhằm giải cứu công chúa. Trong trò chơi này, người chơi sẽ hoá thân thành một “người ngoài hành tinh” đi tìm chiếc phi thuyền của mình. Người chơi sẽ chạy trên đường, tránh các vật cản và tiêu diệt các con quái vật để tìm kiếm phi thuyền. Game được phân thành nhiều màn chơi và có những địa hình, cạm bẫy khác nhau để tăng tính hấp dẫn cho trò chơi.

### TÓM TẮT LỊCH SỬ GIẢI QUYẾT VẤN ĐỀ

Việc phát triển game chạy đa nền tảng là một vấn đề không phải là mới, bởi vì hiện nay đã có rất nhiều game sử dụng nhiều bộ Game Engine khác nhau ( Unity, Cocos2D, Cocos2D - X, V-Play,...) được đăng tải liên tục lên các chợ ứng dụng như Google Play hay App Store.

Tựa game huyền thoại “Mario” đã được cả thế giới chào đón nồng nhiệt, mặc dù có đồ họa không cao nhưng trò chơi này có sức thút rất lớn vì lối chơi vui nhộn, dễ dàng tiếp cận các đối tượng người chơi khác nhau. Tác giả có sử dụng một số thuật toán tương tự để lập trình cho trò chơi này. Mặc dù game "MonsterRun" dựa trên ý tưởng của game “Mario”, nhưng áp dụng trên các đối tượng khác nhau, số lượng đối tượng cũng khác nhau, có sự pha trộn thể loại game bắn súng và cách giải quyết vấn đề do bản thân tác giả đề xuất.

### MỤC TIÊU ĐỀ TÀI

Ứng dụng bộ công cụ phát triển ứng dụng Corona SDK và các thư viện sẵn có (OpenGL, OpenAL, Box2D, JSON ...) để xây dựng game *"MonsterRun"* với các vấn đề trọng tâm như:

* Bố trí các màn chơi một cách logic và theo mức độ từ dễ đến khó.
* Quản lý (bật/ tắt) âm thanh (hiệu ứng/ nhạc nền) thông qua giao diện menu.
* Thực hiện cú nhảy cho nhân vật một cách chân thực.
* Phát hiện nhân vật bắn trúng mục tiêu.
* Xử lý va chạm và hiệu ứng xảy ra khi nhân vật va phải các vật cản.
* Lưu các dữ liệu trong game (thiết lập âm thanh, thành tích,... ).

### ĐỐI TƯỢNG VÀ PHẠM VI NGHIÊN CỨU

Đối tượng được nghiên cứu là các đối tượng đồ hoạ và các thuật toán được sử dụng trong game. Mặc dù công cụ được sử dụng để phát triển game đa nền tảng. Tuy nhiên, trong thời gian có hạn và phạm vi luận văn nghiên cứu chỉ tập trung vào nền tảng Android.

### NỘI DUNG NGHIÊN CỨU

Tác giả sẽ tập trung nghiên cứu các nội dung chính như sau:

* Cách thức xây dựng trò chơi đa nền tảng sử dụng công cụ Corona SDK và ngôn ngữ lập trình LUA.
* Bộ thư viện quản lý trang (composer) và thư viện vật lý (physics) và một số thư viện hổ trợ khác trong Corona SDK. Bộ thư viện quản lý trang hỗ trợ chuyển cảnh trong quá trình phát triển game. Bộ thư viện vật lý (Physics) hỗ trợ cách tạo ra môi trường vật lý cho game, cho đến cách phát hiện va chạm các thành phần vật lý với nhau, tuỳ biến sản phẩm cuối cùng để có thể biên dịch chạy được trên đa nền tảng. Hiểu được một cách tổng quan, sơ bộ về phương thức hoạt động của một số trò chơi đơn giản, những thành phần cơ bản có trong một trò chơi. Tìm hiểu cách thêm, sửa, xoá, tạo mới các kết cấu, cấu tạo, xếp đặt màu sắc lên các đối tượng hiện thị có trong Corona SDK.
* Ngôn ngữ LUA, ngôn ngữ lập trình thông dịch với đặc điểm nhỏ gọn, đa nền tảng. Tìm hiểu cách lập trình, soạn thảo mã kịch bản cho trương trình, cách đặt ra quy luật và giải quyết vấn đề trong trò chơi.
* Tham khảo các runner game, shoot game trên Internet, tìm kiếm các ý tưởng để tạo ra các màn chơi cũng như các hiệu ứng cho game thêm sinh động.
* Xây dựng game "MonsterRun" thông qua các bước phân tích, thiết kế, cài đặt và kiểm thử ứng dụng.

### NHỮNG ĐÓNG GÓP CHÍNH CỦA ĐỀ TÀI

Đề tài đã đóng góp thêm vào tài liệu nghiên cứu cho sinh viên Đại Học Cần Thơ nói chung, và khoa Công nghệ thông tin và Truyền thông nói riêng, trong lĩnh vực lập trình game trên các thiết bị di động. Hơn nữa, đề tài cũng là tài liệu tham khảo cho các sinh viên khoá tiếp theo khi nghiên cứu về Game Engine Corona SDK và ngôn ngữ lập trình LUA.

Game "MonsterRun" đã bổ sung thêm vào kho trò chơi phong phú và đa dạng trên chợ ứng dụng Google Play. Với cách tiếp cận dễ dàng, gameplay quen thuộc của trò chơi, người chơi có thể giải trí nhẹ sau những thời gian làm việc căng thẳng, mệt mỏi, nhưng cũng không kém phần lôi cuốn, hấp dẫn.

### BỐ CỤC CỦA QUYỂN LUẬN VĂN

Quyển luận văn gồm năm nội dung chính:

Phần đầu gồm có: Lời cảm ơn, Mục lục, Danh mục đồ thị, Biểu bảng và hình ảnh, Tóm tắt.

Phần giới thiệu gồm: mô tả sơ lược về bài toán (truy đuổi quái vật), đưa ra vấn đề dựa trên bài toán đó, lợi ích và vì sao chọn Corona SDK; tóm tắt lịch sử giải quyết vấn đề; nêu mục tiêu đề tài với những vấn đề trọng tâm cần giải quyết; nêu đối tượng và pham vi của đề tài; liệt kê toàn bộ nội dung mà đề tài nghiên cứu và nêu những đóng góp chính của đề tài.

Phần nội dung gồm: ba chương.

Chương 1 - Mô tả chi tiết bài toán, đặt ra các yêu cầu chức năng, yêu cầu phi chức năng, cách giải quyết; phân tích đánh giá các giải pháp có liên quan đến bài toán, đưa ra các hướng giải quyết và chọn lựa hướng giải quyết; tiếp cận giải quyết vấn đề đặt ra.

Chương 2 - Trình bày các thiết kế kiến trúc; thiết kế dữ liệu; thiết kế đối tượng - giao diện; thiết kế thuật toán; trình bày cách cài đặt giải pháp dựa vào các thiết kế. Chương 3 - Mô tả mục tiêu kiểm thử, kịch bản kiểm thử và kết quả kiểm thử.

Phần kết luận trình bày về các kết quả đạt được, hạn chế và hướng phát triển của đề tài.

Phần cuối là các tài liệu tham khảo và phụ lục bao gồm: hướng dẫn cài đặt, hướng dẫn sử dụng.

# PHẦN NỘI DUNG

## MÔ TẢ BÀI TOÁN

### MÔ TẢ CHI TIẾT BÀI TOÁN

Vấn đề hiện tại đặt ra là làm sau để có thể tạo ra hình ảnh nhân vật đang chạy và thực hiện các cú nhảy với các độ cao khác nhau để có thể tránh các vật cản, tiêu diệt các quái vật. Bài toán đặt ra có 5 màn chơi, trong từng màn chơi tuỳ vào mức độ phức tạp mà người lập trình đã đặt ra: như độ dài và mức phức tạp của địa hình, các khối vật cản cao thấp, số lượng các quái vật và số lượng các vật cản trên đường đi.

Các yêu cầu đặt ra với bài toán này được chia thành hai nhóm: yêu cầu chức năng và yêu cầu phi chức năng.

**Các yêu cầu chức năng:**

* Chọn màn chơi - cho phép người chơi lựa chọn màn để chơi. Khi người chơi chọn vào màn cụ thể thì màn chơi sẽ có địa hình cụ thể, nhân vật chính, các quái vật, vật cản.
* Hướng dẫn cách chơi - người dùng có thể hiểu thêm về luật chơi và điều khiển nhân vật theo ý muốn.
* Bật/tắt âm thanh (hiệu ứng/ nhạc nền).

**Các yêu cầu phi chức năng:**

* Giao diện:
* Đẹp, bắt mắt.
* Các tông màu phù hợp với các đối tượng.
* Thời gian thực hiện
* Thời gian trễ không quá 3s.
* Game vận hành mượt mà không phát sinh lỗi.
* Thiết bị hỗ trợ
* Hệ điều hành Android 2.3.3 hoặc cao hơn (ARMv7). Ứng dụng được xây dựng trong phiên bản này sẽ không cài đặt được trên các thiết bị ARMv6.
* Bộ nhớ điện thoại trống 30M.
* Độ phân giải màn hình đề xuất 960x640 hoặc cao hơn
* Ghi chú: Ứng dụng được biên dịch cho hệ điều hành Android, việc chuyển sang hệ điều hành iOS và Windows không tốn quá nhiều thời gian cấu hình.

### PHÂN TÍCH ĐÁNH GIÁ CÁC GIẢI PHÁP CÓ LIÊN QUAN ĐẾN BÀI TOÁN

Để có thể giải quyết vấn đề đã tiếp nhận ở trên, có nhiều hướng giải quyết và dưới đây là các hướng có thể áp dụng giải quyết.

Về công nghệ, Adobe PhotoShop là một phần mềm biên tập chỉnh sửa ảnh đồ hoạ, nó có các tính năng vẽ các đối tượng ảnh phong phú, dễ dàng thay đổi kích thước ảnh mà không làm cho đối tượng ảnh bị vỡ, nhoè, dễ sử dụng, trực quan và kết xuất hình ảnh cũng rất chất lượng. Vì thế, tác giả đã sử dụng bộ công cụ này để tạo hình ảnh cho game *"MonsterRun".*

**Hướng giải quyết:** Sử dụng công nghệ Corona SDK và ngôn ngữ LUA.

Công nghệ Corona SDK và LUA có nhiều ưu điểm nổi trội, máy ảo trong Corona SDK chạy rất nhanh, dễ dàng điều chỉnh các độ phân giải, tỉ lệ màn hình khác nhau. Đặc biệt CoronaSDK cho phép tạo âm thanh/yếu tố trực quan chỉ với một hoặc hai dòng code đơn giản. Lua được phát triển từ C và các hệ thống API theo hướng đơn giản hóa. Cài đặt ngôn ngữ LUA, vì có các tính năng: nhỏ gọn, nhanh, kết hợp tốt, đa nền tảng. Toàn bộ phần mềm thông dịch Lua cùng mã nguồn, tài liệu hướng dẩn và các ví dụ chỉ gói gọn trong dung lượng 860 KB (phiên bản 5.1.4, không nén). Tốc độ của Lua thuộc vào loại nhanh nhất trong số các ngôn ngữ thông dịch. Kết hợp tốt với các ngôn ngữ khác, chẳng hạn C. Chạy trên Windows, Unix, Linux, Mac OS X cùng nhiều hệ điều hành khác. Sử dụng ngôn ngữ Lua để cài đặt các thiết kiến trúc, thiết kế dữ liệu và thiết kế thuật toán trong game. Vậy nên tác giả đã chọn hướng giải quyết này.

### TIẾP CẬN GIẢI QUYẾT VẤN ĐỀ

Về cơ bản, một ứng dụng game cần có một luồng trình tự nhất định, nó giúp ta hiểu rõ hơn về cách mà game hoạt động. Với Corona SDK, ứng dụng sẽ thực hiện đầu tiên là xử lý trực tiếp tập tin **main**. Để có thể dễ dàng trong việc quản lý và sửa lỗi ta cần phân loại game ra thành những thành phần khác nhau. Dưới đây là sơ đồ game flow cơ bản:

Main.lua

Menu.lua

Help.lua

Levelselect.lua

Game.lua

gameOver.lua

GameWon.lua

gameOverTotally.lua

Exitgame.lua

Hình 1.1: Sơ đồ Game flow cơ bản.

## THIẾT KẾ GIẢI PHÁP

### THIẾT KẾ KIẾN TRÚC

Cấu trúc tổ chức project được phân cấp như sau

**Project\_Folder**

**images**

**scripts**

**videos**

main.lua

config.lua

**sounds**

build.settings

Hình 2.1: Hệ thống phân cấp project cơ bản

Dưới đây là sơ đồ phân cấp chức năng của ứng dụng:

Game "MonsterRun"

Chọn màn chơi

Hướng dẫn

Thiết lập âm thanh

Màn 1

Màn ...

Màn 5

Hiển thị màn chơi

Xử lý va chạm (đang chơi)

Hiển thị thua cuộc

Hiển thị chiến thắng

Hiển thị điều hướng

Hình 2.2: Sơ đồ phân cấp các chức năng của phần mềm.

### THIẾT KẾ DỮ LIỆU

Sơ đồ phân cấp dữ liệu trong game

**myData**

**maxLevels**

**settings**

**unlockedLevels**

**levels**

**score**

**level**

Hình 2.5: Sơ đồ phân cấp dữ liệu dùng trong game

Bảng diễn giải các dữ liệu dùng trong game

Bảng 2.2: Bảng diễn giải các dữ liệu dùng trong Game

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tên dữ liệu** | | | | **Diễn giải** |
| myData | maxLevels | | | Số màn chơi tối đa |
| settings | unlockedLevels | | Số màn chơi đã mở khoá |
| levels | level | Thông tin màn chơi |
| score | Số điểm khi chơi |

### THIẾT KẾ GIAO DIỆN

**Thiết kế giao diện trò chơi**



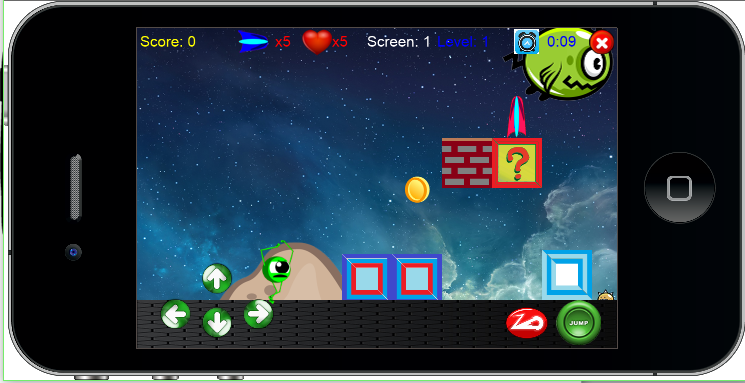
Hình 2.6: Thiết kế giao diện chính của trò chơi.



Hình 2.7: Thiết kế giao diện trợ giúp người chơi.



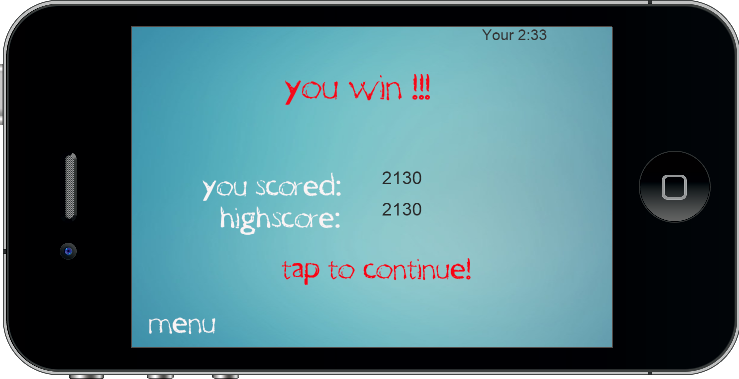
Hình 2.8: Thiết kế giao diện Chọn màn chơi.



Hình 2.9: Thiết kế giao diện đang trong trò chơi.



Hình 2.10: Thiết kế giao diện Kết thúc màn chơi.



Hình 2.11: Thiết kế giao diện Hoàn thành màn chơi.

### THIẾT KẾ THUẬT TOÁN

Danh sách thuật toán cần thiết trong trò chơi.

Bảng 2.3: Bảng danh sách tên các thuật toán có trong trò chơi.

|  |  |  |
| --- | --- | --- |
| **STT** | **Tên thuật toán** | **Ghi chú** |
| 1 | Tạo địa hình. | Tạo địa hình cho từng màn chơi từ việc load một file đã định nghĩa về địa hình cho màn chơi đó. |
| 2 | Tạo các cạm bẫy và quái vật | Thực hiện tạo các cạm bẫy và quái vật ngẫu nhiên trên các vị trí và khoảng các phù hợp. |
| 3 | "Cuộn" màn hình. | Cuộn màn hình dần sang trái và tăng theo tốc độ nhầm tạo cảm giác nhân vật đang chạy sang phải. |
| 4 | Kiểm tra va chạm. | Kiểm tra va chạm giữa nhân vật với các đối tượng khác trong trò chơi. |
| 5 | Bắt sự kiện chạm các button. | Sự kiện xảy ra khi người dùng chạm vào các button. Button phải, trái, lên trên, xuống dưới di chuyển nhân vật theo các hướng trái phải, lên xuống. Button nhảy nhằm thực hiện cú nhảy của nhân vật. Không cho phép nhân vật nhảy trên không trung và cho phép nhân vật nhảy với độ cao tuỳ vào động khác chạm của người dùng.  Button shoot, nhân vật thực hiện bắn viên đạn theo hướng chỉ định. |
| 6 | Bắt sự kiện kết thúc. | Khi hoàn thành màn chơi hoặc khi nhân vật va chạm với các đối tượng như là vật cản hoặc các đối tượng khác như: quái vật, phi thuyền. Thực hiện chuyển trang sang NextLevel hoặc Game Over. |

#### Tạo địa hình cho một màn chơi:

Hình 2.12: Thuật toán tạo một địa hình cho màn chơi.

Bắt đầu

mapTable = yêu cầu

dữ liệu địa hình level

Tạo nhóm đối tượng hiển thị tile

nextDistance = 0

Biến vòng lặp i =0

Tạo đối tượng địa hình tile[i]

tile[i].x = nextDistance

tile[i].objType = “ground”

Physics.addBody(tile[i],{…})

nextDistance += tile[i].width+

math.radom()

i++

Trả về nhóm đối tượng hiển thị tile

i < số lượng đối tượng

địa hình mapTable

Đúng

Sai

Kết thúc

#### Tạo các cạm bẫy và quái vật:

Hình 2.13: Thuật toán tạo cạm bẫy và quái vật ngẫu nhiên trong màn chơi.

Bắt đầu

Nhận vào nhóm đối tượng địa hình tile

Khởi tạo mảng Array các quái vật/cạm bẫy

Biến vòng lặp i = độ dài tile + 1

Local start= khoảng cách ban đầu

Local endpoint = điểm cuối địa hình

Lấy các quái vật/cạm bẫy Obj trong mảng Array

Tạo đối tượng tile[i] = Obj

tile[i].x = giá trị cụ thể

Thêm sự kiện vật lý cho tile[i]

Cập nhật start

i++

Trả về nhóm đối tượng hiển thị tile

start < endPoint

Đúng

Sai

Kết thúc

#### "Cuộn" địa hình:

Hình 2.14: Thuật toán “cuộn” địa hình trong khi chơi.

Bắt đầu

Obj = đối tượng

Speed = tốc độ cuộn

endPoint = điểm cuối màn chơi

Obj = Obj.x - speed

Nếu Obj là đối tượng cuối màn chơi và endPoint == Obj.x+ Obj.width

Đúng

Sai

Kết thúc

.

#### Kiểm tra va chạm:

Hình 2.15: Thuật toán kiểm tra va chạm trong trò chơi.

Đúng

Đúng

Bắt đầu

char = nhân vật

Obj = đối tượng va chạm

NumRescue = biến đếm

Nhân vật có thể nhảy

char.canJump++

Nhân vật không thể nhảy

char.canJump--

Phase == begin

Sai

Kết thúc

Nếu char va chạm với obj.objType == “ground”

Nếu char va chạm với obj.objType == “enemy”

Nếu char va chạm với obj.objType == “spikes”

Nếu char va chạm với obj.objType == “coin”

Bắt sự kiện Game over

NumRescue++

Obj.rescued = true

Đúng

Sai

Sai

Sai

Đúng

Sai

Đúng

#### Bắt sự kiện chạm button:

Hình 2.16: Thuật toán sự kiện chạm button trong trò chơi.

Nhân vật rơi xuống

Đúng

Đúng

Bắt đầu

Phase = trạng thái sự kiện touch screen

Cho phép nhân vật nhảy lên một khoảng rồi rơi xuống

Phase == begin

Sai

Kết thúc

Nếu nhân vật đang xảy ra va chạm với “ground” và có thể nhảy

Phase == ended

Sai

#### Bắt sự kiện kết thúc:

Hình 2.17: Thuật toán sự kiện kết thúc trò chơi.

Thực hiện chuyển cảnh kết thúc

Đúng

Bắt đầu

Thực hiện cuộn địa hình

Kiểm tra các va chạm của nhân vật

Thực hiện chuyển cảnh Hoàn thành

Nếu cuộn đến điểm cuối địa hình

Sai

Kết thúc

Nếu nhân vật va chạm với các đối tượng là quái vật/spikes

Sai

Đúng

## CÀI ĐẶT GIẢI PHÁP

### CÀI ĐẶT KIẾN TRÚC

Cấu trúc dự án được cài đặt dựa trên cấu trúc đã thiết kế:

+**Project:**

+ sounds (folder): bao gồm các tập tin âm thanh và nhạc nền.

+ images (folder):

+ gaming (folder): những hình ảnh sẽ có trong khi chơi.

+ ammon(folder): những hình ảnh về các viên đạn.

+ block(folder): những hình ảnh về các khối chướng ngại vật.

+ coin(folder): những hình ảnh về các khối điểm số.

+ flap(folder): những hình ảnh về các phi thuyền.

+ footter (folder): những hình ảnh về địa hình.

+ header (folder):: những hình ảnh về màn chơi.

+ monster (folder): những hình ảnh về các quái vật.

+ trap (folder): những hình ảnh về các quái vật cạm bẫy.

+ button: những hình ảnh về các button dùng trong game.

+ charater: những hình ảnh về nhân vật dùng trong game.

+ scene: những hình ảnh về hình nền, các nút điều hướng...

+ script (folder): bao gồm những tập tin mã nguồn của game.

+ level (folder): chứa các file menu trò chơi.

+ map (folder): chứa các file về địa hình của từng màn chơi.

+ scene (folder): chứa các file script trang điều hướng trong trò chơi.

+ main.lua: ứng dụng sẽ chạy tập tin này đầu tiên.

+ config.lua: chứa các thông tin thiết lập cấu hình cho ứng dụng.

+ build.settings: Để kiểm soát định hướng ứng dụng trên các thiết bị thực tế.

### CÀI ĐẶT DỮ LIỆU

Danh sách các đối tượng có trong game "Monster Run" được mô tả trong bảng dưới đây. Bao gồm tên của đối tượng, hình ảnh và vai trò trong trò chơi.

Bảng 3.1: Danh sách các đối tượng trong game *"MonsterRun".*

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Tên đối tượng** | **Hình ảnh** | **Vai trò** |
| 1 | Nhân vật chính |  | Nhân vật chính trong game, có vai trò tiêu diệt các quái vật, vượt qua địa hình hiểm trở và các cạm bẫy. |
| 2 | Quái vật |  | Các con quái vật ngăn cản nhân vật. |
| 13 | Quái vật cạm bẫy |  | Các con quái vật có gai nhọn cản trở nhân vật. |
| 14 | Hòm đạn |  | Tiếp đạn cho nhân vật. |
| 15 | Viên đạn |  | Đạn do nhân vật bắn ra nhằm tiêu diệt các quái vật. |
| 16 | Khối gạch 1 |  | Khối gạch tạo chướng ngại di chuyển cho nhân vật, có thể phá hủy |
| 17 | Khối gạch 2 |  | Khối gạch tạo chướng ngại di chuyển cho nhân vật, không thể phá hủy |
| 18 | Khối gạch 3 |  | Khối gạch tạo chướng ngại di chuyển cho nhân vật, có thể di chuyển, không thể phá hủy |
| 19 | Khối bí ẩn |  | Khối gạch tạo chướng ngại di chuyển cho nhân vật, không thể phá hủy, chạm vào từ dưới lên sẽ xuất hiện các ngôi sao mang điểm số cao. |
| 20 | Phiến đất |  | Một trong những thành phần cấu tạo nên địa hình bản đồ. |
| 22 | Coin |  | Tích điểm số trong trò chơi |
| 23 | Đĩa bay 1 |  | Một trong những thành phần cấu tạo nên địa hình bản đồ. |
| 24 | Đĩa bay 2 |  | Một trong những thành phần cấu tạo nên địa hình bản đồ. |
| 25 | Đĩa bay 3 |  | Khi nhân vật chạm vào game sẽ kết thúc và người chơi chiến thắng. |
| 26 | Cái thang |  | Một trong những thành phần cấu tạo nên địa hình bản đồ. |
| 27 | Nút Play |  | Khi người chơi nhấn vào sẽ chuyển đến trang Chọn màn chơi. |
| 28 | Nút Settings |  | Khi người chơi nhấn vào sẽ chuyển đến trang thiết lập để bật tắt âm cho trò chơi. |
| 29 | Nút Help |  | Khi người chơi nhấn vào sẽ chuyển đến trang giúp đỡ, hiển thị luật chơi. |
| 30 | Nút Menu, Back, Tap to go back menu |  | Khi người chơi nhấn vào sẽ kết thúc trang và quay trở về giao diện chọn màn chơi. |
| 31 | Nút Nextlevel |  | Khi người chơi nhấn vào sẽ kết thúc trang và khởi động màn chơi tiếp theo. |
| 32 | Nút Restart |  | Khi người chơi nhấn vào sẽ khởi động lại trò chơi. |
| 33 | Sinh mệnh của nhân vật |  | Thể hiện số lượt chơi của nhân vật. |
| 34 | Thời gian chơi |  | Thể hiện thời gian thực hiện trò chơi của người chơi. |
| 35 | Nút exit |  | Khi người chơi nhấn vào sẽ chuyển đến trang menu chính. |
| 36 | Nút shoot |  | Khi người chơi nhấn vào sẽ bắn ra các viên đạn. |
| 37 | Nút nhảy |  | Khi người chơi nhấn vào nhân vật sẽ nhảy lên cao. |
| 38 | Nút sang trái |  | Khi người chơi nhấn vào nhân vật sẽ di chuyển sang trái. |
| 39 | Nút sang phải |  | Khi người chơi nhấn vào nhân vật sẽ di chuyển sang phải. |
| 40 | Nút lên trên |  | Khi người chơi nhấn vào nhân vật sẽ di chuyển lên trên. |
| 41 | Nút xuống dưới |  | Khi người chơi nhấn vào nhân vật sẽ di chuyển xuống dưới. |

### XÂY DỰNG GIAO DIỆN

#### Xây dựng hình ảnh nhân vật.

Để có thể tạo hoạt ảnh nhân vật chạy một cách mượt mà, tác giả đã sử dụng **Spriter**. Một trình ứng dụng giúp tạo ảnh động thông qua việc "gắn xương" cho nhân vật. Lợi thế của Spriter là có thể xuất ra nhiều frame ảnh định dạng PNG:

Sau khi hoàn tất công đoạ tạo hoạt ảnh động cho nhân vật, ta có thể Export ra file PNG:



Hình 2.18 : Loạt hình ảnh tạo nên chuyển động chạy của nhân vật chính sau khi xuất ra tập tin hình ảnh PNG.

#### Xây dựng hình ảnh các quái vật

**

Hình 2.19: Hình ảnh quái vật chuyển động

#### Xây dựng hình ảnh các cạm bẫy

**

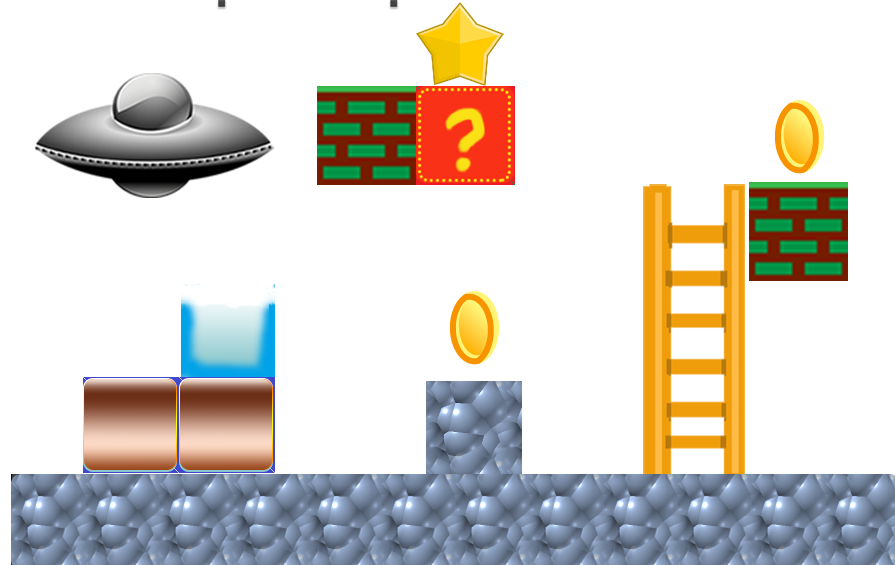
Hình 2.20: Hình ảnh tạo ra các quái vật gai nhọn.

**

Hình 2.21: Hình ảnh cách tạo ra nhiều quái vật gai nhọn.

#### Xây dựng địa hình từ những mảnh địa hình có sẵn:

Để có thể tạo một địa hình hoàn chỉnh cho một màn chơi, ta sử dụng các mảnh địa hình nhỏ, sắp xếp chúng theo ý muốn và kết hợp với các đối tượng hiển thị khác.

**

Hình 2.22: Hình ảnh các mảnh vuông tạo nên địa hình trong game.

#### Thiết kế giao diện các trang (scene) điều hướng trong trò chơi:

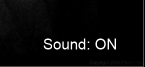
**

Hình 2.23: Giao diện Menu.

**

Hình 2.24: Giao diện Help.

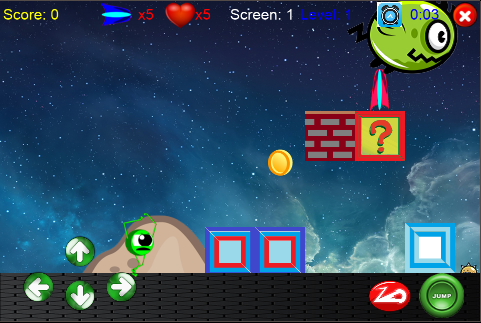
.

**

Hình 2.25:Giao diện Settings.

**

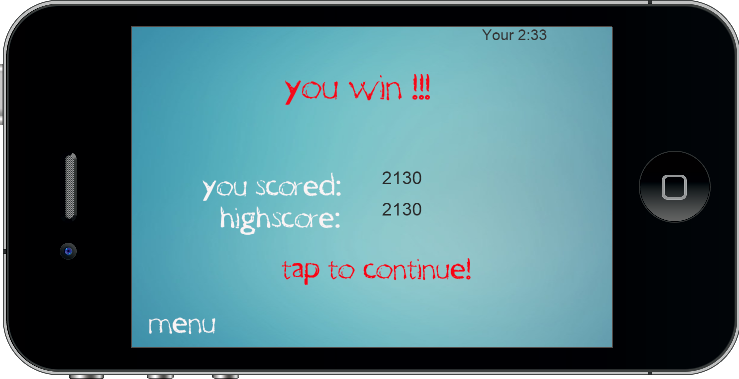
Hình 2.26: Giao diện Chọn màn chơi.

**

Hình 2.27: Giao diện trong trò chơi



Hình 2.28: Giao diện Game Over.



Hình 2.29: Giao diện Game Win.

## KIỂM THỬ VÀ ĐÁNH GIÁ

### MỤC TIÊU KIỂM THỬ

Nhằm xác định được sản phẩm đã hoạt động bình thường hay chưa, có còn xuất hiện lỗi chương trình hay không, quy hoạch lại các lỗi không mong muốn và khắc phục các lỗi đó.

Bên cạnh đó, ta cần kiểm tra lại tính chính xác của từng chức năng có trong chương trình của từng đối tượng. Xem xét lại những chức năng thực hiện có đúng như yêu cầu đề ra hay không.

### KỊCH BẢN KIỂM THỬ

* **Kịch bản 1:** Vào chơi ở màn 1, thực hiện cú nhảy, khi nhân vật chưa rơi xuống, liên tiếp chạm màn hình để thực hiện thêm cú nhảy khác. Thực hiện chạm ngắn và chạm lâu hơn trên màn hình.
* **Kịch bản 2:** Khi kết thúc màn 1, thực hiện các thao tác như: Chơi lại, chơi tiếp, và quay về giao diện chính.
* **Kịch bản 3:** Vào một màn chơi bất kỳ, thực hiện va chạm giữa nhân vật và các chướng ngại vật, quái vật, xem kết quả khi va chạm. Thực hiện các thao tác như: Chơi lại, chọn màn chơi khác và quay về giao diện chính.
* **Kịch bản 4:** Thực hiện các thao tác bật tắt âm, nhạc nền, hoàn thành vài màn chơi và thoát khỏi game. Sau đó thực hiện mở lại game.

Ghi chú: Môi trường kiểm thử:

* Thiết bị: Philips S389
* Cấu hình: Ram 1Gb, bộ nhớ trong 8Gb. CPU 1GHz. Android: 4.4.4
* Kết quả: Hoạt động ổn định.
* Thiết bị: Obi S507
* Cấu hình: Ram 2Gb, bộ nhớ trong 16Gb. CPU 1.5GHz. Android: 5.0
* Kết quả: Hoạt động ổn định, mượt mà.

### KẾT QUẢ KIỂM THỬ

Sau đây là bảng danh sách kết quả kiểm thử cho trò chơi

Bảng 4.1: Kết quả kiểm thử theo từng chức năng.

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Mục đích | Kết quả sản phẩm | Kết luận |
| 1 | Kiểm tra xem, nhân vật có thực hiện được một cú nhảy hay không. Khi nhân vật đang nhảy, thì ta có thể thực hiện nhảy liên tiếp khi chưa chạm đất hay không. Kiểm tra xem nhân vật có thể nhảy thấp, và nhảy cao được hay không. | Nhân vật chỉ có thể nhảy lên khi đang chạm đất. Nhân vật sẽ nhảy cao hơn nếu ta chạm liên tục vào nút nhảy. | Thành công |
| 2 | Kiểm tra số liệu khi chuyển từ trang đang chơi sang trang hoàn thành. Kiểm tra khi hoàn thành màn chơi thì có chuyển cảnh sang Level Completed hay không. Kiểm tra xem các phím chức năng sau khi chuyển cảnh. | Chuyển cảnh sang Level Completed khi hết màn chơi. Các phím chức năng hoạt động đúng theo yêu cầu. | Thành công |
| 3 | Kiểm tra va chạm giữa nhân vật với các đối tượng khác trong trò chơi. Kiểm tra xem khi va chạm với các vật cản thì có chuyển cảnh sang Game Over hay không. Kiểm tra các phím chức năng sau khi chuyển cảnh. | Nhân vật va chạm được với các đối tượng khác trong trò chơi. Khi va chạm với vật cản thì xuất hiện hiệu ứng và tự động chuyển sang trang Game Over sau một khoảng thời gian ngắn. | Thành công |
| 4 | Kiểm tra xem các thiết lập của người dùng đối với game. Kiểm tra lưu trữ thành tích các màn đã chơi trong game. | Các thiết lập vẫn giữ nguyên. Thành tích các màn đã chơi trong game hiển thị đúng. | Thành công |

# PHẦN KẾT LUẬN

### KẾT QUẢ ĐẠT ĐƯỢC

Về lý thuyết, tác giả đã tìm hiểu và tự học được một ngôn ngữ lập trình mới, bổ sung vốn kiến thức lập trình cho bản thân. Bên cạnh đó, tác giả còn học hỏi được rất nhiều những tiện ích khi sử dụng Corona SDK để có thể phát triển một ứng dụng hoạt động trên các thiết bị Android và các thiết bị chạy hệ điều hành khác. Từ đó, bản thân tác giả có thể tự tạo dự án và phát triển các ứng dụng hoặc game một cách nhanh chóng.

Về chương trình, hoàn thành khoá luận, tác giả đã hoàn thành được một trò chơi bao gồm 5 màn chơi, được sắp xếp theo thứ tự từ dễ đến khó, với những chức năng đáp ứng được yêu cầu đề ra. Sản phẩm vận hành tốt trên nền tảng android. Khi thực hiện kiểm tra trên thiết bị thật, game hoạt động mượt mà, không xuất hiện tình trạng giật lag, âm thanh phát ra chính xác, không bị ngắt hoãn, giao diện thân thiện, dễ sử dụng, dễ thao tác. Sản phầm còn góp phần làm phong phú hơn kho tàng trò chơi giải trí.

### HẠN CHẾ

Do hạn chế trong việc thiết kế đồ họa, không có đầy đủ các thiết bị (điện thoại di động) để kiểm thử, thời gian làm luận văn bị hạn chế, tìm hiểu công nghệ mới và ngôn ngữ lập trình mà tác giả chưa từng tìm hiểu qua, cùng với việc không kịp phát triển, thêm mới những hình ảnh, hiệu ứng cần thiết cho trò chơi, nên hiện tại trò chơi còn một số nhược điểm về tính chân thật, hình ảnh thiếu chi tiết.

Trò chơi theo hướng các trò chơi truyền thống, mang tính giải trí nhẹ, chưa thực sự hấp dẫn số lượng người chơi.

### HƯỚNG PHÁT TRIỂN

Tác giả sẽ tiếp tục phát triển game, bổ sung các hình ảnh chi tiết, các hiệu ứng âm thanh ,thêm các phần tiếp theo, bổ sung nội dung câu truyện nhằm tăng tính hấp dẫn cho game. Bên cạnh đó, tác giả sẽ thêm các tiện ích như tạm dừng game và lưu lại điểm tạm dừng để người chơi có thể chơi lại tại lúc. Game sẽ tiếp tục phát triển trên các nền tảng khác như iOS và Windows Phone.

Mã nguồn của trò chơi sẽ được công bố lên cộng đồng mạng. Để những ai có đam mê với nghề lập trình game có thể tiếp tục phát triển hơn.

# TÀI LIỆU THAM KHẢO

[1] **TS. Huỳnh Xuân Hiệp, Th.S Phan Phương Lan**. GIÁO TRÌNH Nhập môn CÔNG NGHỆ PHẦN MỀM. Nhà xuất bản Đại học Cần Thơ 2011.

[2] Corona Việt. Game Studio Cổng thông tin cho nhà phát triển Game – Diễn đàn trao đổi kiến thức về Corona. Website: www.coronaviet.com/forum

[3] Corona Việt. Hướng dẫn: Cho phép nhân vật thực hiện cú nhảy trong Corona SDK. http://coronaviet.com/forum/index.php?threads/huong-dan-cho-phep-nhan-vat-thuc-hien-mot-cu-nhay-trong-corona-sdk.496/

[4] **Coronalabs – Documentation – Getting Started**. Website: https://docs.coronalabs.com/guide/programming/01/index.html

# [5] Tyler King, (Jan 2012). Corona SDK: Build an Endless Runner Game From Scratch! Website:

https://code.tutsplus.com/series/corona-sdk-build-an-endless-runner-game-from-scratch--mobile-22913

[6] **Androi Việt. Bài tổng hợp lập trình với Corona.** http://android.vn/threads/bai-tong-hop-lap-trinh-voi-corona.13797/

# PHỤ LỤC

### HƯỚNG DẪN CÀI ĐẶT CORONA SDK TRÊN WINDOWS

#### Cài đặt Java Development Kit

Cài đặt **Java Development Kit** để có thể thử nghiệm một ứng dụng trên một thiết bị Android giải lập.

Truy cập trang Web để có thể tải về cài đặt: http://www.oracle.com/technetwork/java/javase/archive-139210.html.

#### Cài đặt Corona SDK

Đăng ký một tài khoản Corona SDK để có thể Download và sử dụng. Đăng ký tại: https://developer.coronalabs.com/user/login

Sau khi tải về và cài đặt, mở Corona Simulator. Đăng nhập tài khoản đã đăng ký để kích hoạt.

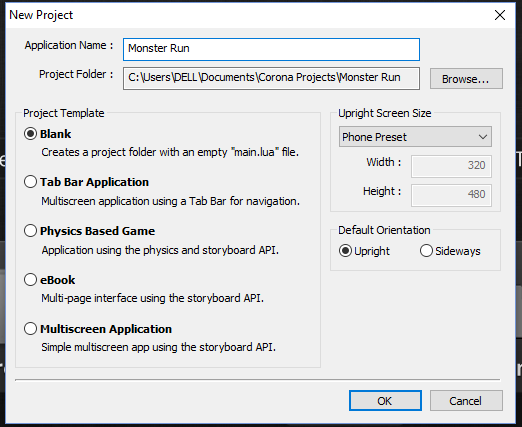
### HƯỚNG DẪN SỬ DỤNG

#### Tạo một Project từ Corona SDK

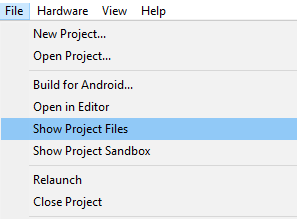
Mở ứng dụng Corona Simulator. Nhấn vào nút "New Project" để tạo mới một dự án.



Điền tên ứng dụng, chọn mẫu dự án (mặc định là Blank). Chọn độ kích thước màn hình và hướng mặc định cho ứng dụng. Sau khi hoàn tất, nhấn OK.



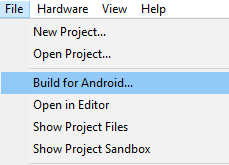
Sau khi tạo Project, từ thanh công cụ, chọn File > Show Project File.



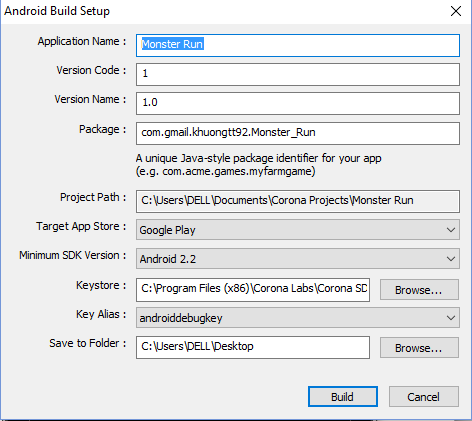


Ở đây có sẵn các tập tin biểu tượng phụ thuộc vào độ phân giải, cũng như tỉ lệ màn hình của nhiều thiết bị. Cần chú ý tập tin main.lua, config.lua và build.settings. Tập tin main.lua là tập tin sẽ thực thi đầu tiên sau khi chương trình hoạt động. Bạn Có thể viết một vài dòng code để hiển thị đối tượng hình ảnh ở đây.

#### Cách build project ( xuất sản phẩm ra file .apk)

Từ thanh công cụ, chọn File > Chọn Build for Android.

Trong bảng Android Build Setup, điền các thông tin về: tên ứng dụng, mã phiên bản, tên phiên bản, gói,... chọn đường dẫn lưu trữ và cuối cùng là nhấn Build.



### SƠ LƯỢC VỀ NGÔN NGỮ LUA

#### Giới thiệu

Giới thiệu ngôn ngữ Lua: LUA (tiếng Bồ Đào Nha: Mặt Trăng) là một ngôn ngữ lập trình thông dịch với đặc điểm nhỏ gọn, đa nền tảng. Lua được phát triển từ C và hệ thống các API theo hướng đơn giản hóa.

**Tính năng:**

– Nhỏ gọn: Toàn bộ phần mềm thông dịch Lua cùng mã nguồn, tài liệu hướng dẩn và các ví dụ chỉ gói gọn trong dung lượng 860 KB (phiên bản 5.1.4, không nén).

– Nhanh: Tốc độ của Lua thuộc vào loại nhanh nhất trong số các ngôn ngữ thông dịch.

– Kết hợp tốt với các ngôn ngữ khác, chẳng hạn C.

– Đa nền tảng: Chạy trên Windows, Unix, Linux, Mac OS X cùng nhiều hệ điều hành khác.

– Thư viện chương trình có một số tính năng tốt: xử lý chuỗi, coroutine.

– Là một trong những ngôn ngữ đang phát triển mạnh

**Đặc điểm kĩ thuật**

– Lua là ngôn ngữ kiểu động. Các biến được sử dụng mà không cần khai báo trước. Biến trong Lua được mặc định là biến toàn cục (global).

– Lua là ngôn ngữ theo thiên hướng lập trình thủ tục. Tuy nhiên trong Lua hàm được coi là kiểu biến cơ bản (first-class variable), giống như lập trình hàm. Ngoài ra Lua còn có một cách khéo léo mô phỏng lập trình hướng đối tượng.

– Lua có cấu trúc dữ liệu rất gọn. Chỉ một kiểu số duy nhất (không phân biệt số nguyên, số thực). Kiểu dữ liệu phức hợp gồm một dạng là table, tức là hash-table, associative array hoặc dictionary theo các cách gọi khác nhau.

– Từ một kiểu dữ liệu table có thể sử dụng linh hoạt để biểu diễn các kiểu dữ liệu khác. Nếu như kiểu table có dạng tổng quát là {key1: value1, ke2y: value2,... } thì khi đặt key1, key2 bằng các số tự nhiên thì ta có kiểu mảng một chiều (kiểu dãy). Ngoài ra, với việc sử dụng metatable ta có thể hình thành các phương thức bằng việc đưa tên hàm vào đóng vai trò của các khóa.

### SƠ LƯỢC VỀ CORONA SDK

Corona được coi là một trong những nền tảng phát triển game 2D tốt nhất hiện nay. Corona hỗ trợ đa nền tảng, điều đó có nghĩa là bạn chỉ cần phát triển game một lần duy nhất và dễ dàng biên dịch sang các nền tảng di động khác nhau.

Dưới đây là những đặc chưng của Corona SDK :

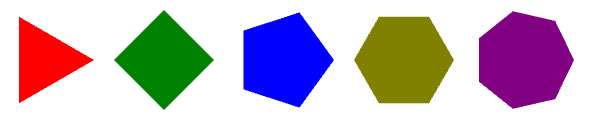
- Corona SDK giúp phát triển ứng dụng nhanh hơn 10 lần. Với hệ thống thư viện hỗ trợ phong phú và được thiết kế rất tốt, lập trình viên có thể dễ dàng xây dựng các thao tác (hành động ) trong game chỉ bằng một vài dòng code.

- Corona hỗ trợ đa nền tảng. Bạn chỉ cần phát triển game một lần, sau đó có thể dễ dàng biên dịch ứng dụng sang các nền tảng khác như Android, iOS, Kindle Fire, Nook và Windows Phone một cách dễ dàng. Không những vậy Corona còn hỗ trợ chuyển đổi với những thiết bị có kích cỡ và độ phân giải khác nhau một cách dễ dàng.

- Đạt các tiêu chuẩn của ngành công nghiệp game.

#### Các đối tượng hiển thị

Các đối tượng hiển thị trên màn hình thiết bị là một thể hiển của một Display Object như hình chữ nhật, hình thoi, hình tam giác, hình tròn, văn bản, hình ảnh,...



Một số thuộc tính của các đối tượng hiển thị (Obj)

– Obj.alpha: chỉ độ trong suốt của đối tượng giá trị từ 0 đến 1, mặc định là 1.

– Obj.x: tọa độ x

– Obj.y: tọa độ y

– Obj.isVisible: có thể nhìn thấy hay không thấy của đối tượng, mặc định là true.

– Obj.rotation: góc quay hiện tại của đối tượng (đơn vị là độ)

– Obj.contentHeight: chiều cao trong tọa độ màn hình

– Obj.contentWidth: chiều rộng trong tọa độ màn hình

– Obj.xScale: giá trị tỉ lệ theo chiều rộng của đối tượng

– Obj.yScale: giá trị tỉ lệ theo chiều dài của đối tượng

##### Image Sheet

**Xác định trình tự Animation**

**local** sequenceData = {  
  { name = "normalRun",  *--tên trình tự*  
    start = 1,  *--chỉ mục của frame bắt đầu*  
    count = 8,  *--tổng số lượng frame*  
    time = 800,  *--milliseconds; nếu không cung cấp, sprites sẽ dựa trên khung cơ bản*  
    loopCount = 0,  *--0 (default) lập lại mãi mãi; một số nguyên sẽ chỉ định số lần lập lại*  
    loopDirection = "forward"  *--hoặc "forward" (mặc định) hoặc "bounce" sẽ play khung hình tới hay lui trong trình tự*  
  }  *--nếu xác định nhiều trình tự hơn, bạn đặt một dấu phẩy ở đây và tiến hành phụ bảng của trình tự tiếp theo*  
}

**Khai báo các đối tượng Sprite**

* display.newSprite( [parent,] imageSheet, sequenceData )

+ parent = Các nhóm hiển thị parent để chèn Sprites (option).  
+ imageSheet = image sheet mà sprite sẽ sử dụng.  
+ sequenceData = Các mảng của trình tự animation mà bạn thiết lập.

* **local** animation = display.newSprite( mySheet, sequenceData )

**Phương pháp điều khiển Sprites**

- animation:play() :Bắt đầu play animation.  
- animation:pause() :Tạm dừng các animation.  
- animation:setFrame( frame ) :Lập tức thiết lập hoặc bỏ qua các chỉ số frame chỉ định.   
- animation:setSequence( sequence )  : Thiết lập các Sprites với một trình tự cụ thể.

Ví dụ:

**local** sheetData = { width=512, height=256, numFrames=8, sheetContentWidt h=1024, sheetContentHeight=1024 }  
 **local** mySheet = graphics.newImageSheet( "runningcat.png", sheetData )  
 **local** sequenceData = {  
    { name = "normalRun", start=1, count=8, time=800 },  
    { name = "fastRun", frames={ 1,2,4,5,6,7 }, time=250 }  
  }  
**local** animation = display.newSprite( mySheet, sequenceData )  
animation.x = display.contentWidth/2  
animation.y = display.contentHeight/2  
  
animation:play()

##### Text

– display.newText( string , x , y , font , size )

+ string: chuỗi hiển thị

+ x, y: tọa độ x, tọa độ y

+ font: tên font

+ size: kích thước

– Thuộc tính thường dùng

+ Tô màu text: Obj:setTextColor(R, G, B)

+ Thay đổi text: Obj.text = "text"

##### Group

– Dùng để nhóm các đối tượng lại thành 1 nhóm

– Khai báo: local g = display.newGroup()

– Thuộc tính:

+ g.insert( index , obj1 ) *--Thêm obj1 vào trong group g tại vị trí index*

+ g.remove( index\_of\_child) *-- Xoá 1 đối tượng tại index ra khỏi g*

+ g.numChildren *--Trả về số lượng các phần tử trong group*

#### Sự kiện trên các đối tượng hiển thị ( tap, touch)

**Thêm sự kiện vào đối tượng hiển thị**

Obj.addEventListener( "touch" , functouch)

**Xoá sự kiện khỏi đối tượng hiển thị**

Obj.removeEventListener( "touch" , functouch)

##### Phát hiện Tap

Tap event là hình thức cơ bản nhất của sự tương tác với màn hình. Về cơ bản, một Tap được hiểu là khi người sử dụng chạm vào màn hình và nhất tay ra khỏi màn hình tại vị trí đó. Tap event này được coi là thành công chỉ khi người sử dụng chạm và rời khỏi màn hinh ngay tại điểm đó.

Lắng nghe các sự kiện tap bằng cách đăng ký một "tap" event listener trên hầu hết các đối tượng hiển thị:

Ví dụ:

**local** menuButton = display.newRect(100, 100, 200, 50) *--tạo một nút*

**local** **function** tapListener(event)

*-- Code thực thi khi nút được tap*

print( "Button tapped = " ..toString(event.target))

**return** true

**end**

menuButton:addEventListener( "tap" , myTapListener) *-- thêm một sự kiện tap*

Các thuộc tính event trả về từ một tap bao gồm:

**+ event.target** - tham chiếu đến đối tượng đó đã được tap.

**+ event.name** – string của tap

**+ event.numTaps** – number của tap trên màn hình. Delay mặc định với tap tiếp theo trong dãy là 0, nhưng thời gian này có thể được điều chỉnh với các hàm system.setTapDelay().

**+ event.x/event.y** - vị trí x và y của tap, trong tọa độ nội dung.

Không giống như touch events, tap events không có thuộc tính phase - tap là một hành động đơn lẻ của touch và nhấc tay ra, do đó bạn không cần phải xử lý các phase trong bất kỳ cách thức đặt biệt nào.

##### Phát hiện touch

Touch event cung cấp một mức độ lớn hơn nhiều so với tap trong việc tương tác với màn hình. Sử dụng touch event, để có thể phát hiện khi người sử dụng lần đầu tiên chạm vào màn hình và khi họ nhấc tay ra khỏi màn hình. Hặc cũng có thể theo dõi chuyển động của các touch khi người dùng chạm và di chuyển xung quanh màn hình. Để thực hiện điều này, Corona cung cấp các thuộc tính event.phase ở một trong bốn trạng thái sau:

+ "**began"** - chỉ ra một touch đã bắt đầu trên màn hình.

**+ "moved"** - chỉ ra một touch đã di chuyển trên màn hình.

**+ "ended"** - chỉ ra một touch đã kết thúc, người dùng đã nhấc tay ra khỏi màn hình.

**+ "cancelled"** - chỉ ra rằng hệ thống hủy theo dõi các touch.

Ví dụ:

**local** **function** touchListener(event)

**if** (event.phase == "began") **then**

*-- khi bắt đầu touch*

print("object touched = "..toString(event.target))

**elseif** (event.phase == "moved" ) **then**

*-- khi di chuyển touch trên đối tượng*

print( "touch location in content coordinates = "..event.x..","..event.y )

**elseif** (event.phase == "ended") **then**

*-- khi kết thúc touch*

print( "touch ended on object "..tostring(event.target) )

**end**

**return** true *-- chặn việc truyền các touch vào các đối tượng bên dưới*

**end**

**Các thuộc tính event trả về từ một touch bao gồm:**

**+ event.id** - một định danh duy nhất để phân biệt qua những touch event khác nhau.

**+ event.target** - tham chiếu đến các đối tượng đã được touch.

**+ event.name** – giá trị tên của touch

**+ event.phase** - phase của touch như mô tả ở trên.

**+ event.time** - thời gian trong mili giây kể từ khi bắt đầu của ứng dụng, truy cập từ bên trong hàm touch listener.

**+ event.x/event.y** - vị trí x và y của các touch, trong tọa độ nội dung.

**+ event.xStart/event.yStart** - vị trí x và y của touch từ "begin" phase của trình tự touch, trong tọa độ nội dung.

#### Thêm sự kiện vật lý cho các đối tượng (physics)

**Cú pháp:**

physics.addBody(obj[,bodyType] [, proerties])

Trong đó:

– **obj** là đối tượng hiển thị mà bạn muốn thêm sự kiện vật ký.

– **bodyType** là trạng thái của một cơ thể vật lý, gồm các giá trị dynamic, static or kinematic.

+ static: đối tượng sẽ ở vị trí đặt ban đầu.

+ dynamic: Anh hưởng bởi trọng lực và va chạm với các đối tượng khác làm chuyển động bản thân đối tượng (mặc định).

+ kinematic: Bị ảnh hưởng bởi một lực tác động

– **Properties** bao gồm một số tham số cơ bản sau:

+ density: là khối lượng của một cơ thể vật lý. Giá trị mặc định là 1.0

+ friction: là độ ma sát của một cơ thể vật lý. Giá trị này không âm. Giá trị 0

có nghĩa là không có ma sát và 1.0 có nghĩa là ma sát khá mạnh. Giá trị mặc định là 0.3.

+ bounce: là độ nhảy của một cơ thể vật lý khi gặp va chạm. Giá trị mặc định là 0.2.

+ radius: là tham số bán kính của một cơ thể vật lý nếu cơ thể đó là hình tròn. + shape: là tham số hình dạng dùng để tạo ra cơ thể vật lý hình đa giác. Đây là một table Lua của một cặp x và y, mỗi cặp xác định một điểm đỉnh cho các hình dạng. Các tọa độ được quy định liên quan đến các trung tâm của đối tượng hiển thị - do đó, với (0,0) tương ứng với các trung tâm của đối tượng, một điểm đỉnh của (-20, -10) định nghĩa một điểm 20 điểm ảnh bên trái của trung tâm và 10 điểm ảnh trên trung tâm.

+ isSensor: là tham số xác định cơ thể vật lý này có bị chuyển động khi bị vật khác đụng phải hay không, tuy nhiên vẫn có thể phát hiện được va chạm thông qua sự kiện collision trong khi tính năng này được tắt. Giá trị mặc định là true.

Ví dụ:

local m\_physics = require("physics")

m\_physics.start()

local m\_rect = display.newRect( 0,420,800,60 )

m\_physics.addBody( m\_rect, "static", {desity = 1.0, friction = 0.5, bounce = 0.3 }) local m\_circle = display.newCircle( 100,100,30 )

m\_physics.addBody( m\_circle, {desity = 3.0, friction = 1, bounce = 0.3, radius = 30})

#### Quản lý cảnh (scene) với Composer

Composer là một API dùng để quản lý các cảnh (scene) trong game. Cho phép tạo ra các tập tin có phần mở rộng là .lua đại diện cho một cảnh (scene) trong ứng dụng/ game. Khi người dùng điều hướng đến những cảnh khác nhau, Composer sẽ xử lý quá trình chuyển cảnh và giúp người lập trình dễ dàng kiểm soát các đối tượng (hoặc biến) không cần thiết. Nó là một API rất hữu ích và có thể làm cho việc tạo và quản lý các trang trong game dễ dàng hơn. Composer cung cấp một cấu trúc đơn giản và có khả năng mở rộng hơn cho việc quản lý các cảnh trong trò chơi.

Thư viện Composer cung cấp cho các nhà phát triển cách để tạo ra và chuyển đổi giữa các cảnh cho game một cách dễ dàng thông qua các hàm sau:

– composer.getScene() *--trả về đối tượng scene chỉ định*

– composer.getSceneName() *--trả về tên scene hiện tại*

– composer.getVariable() *--trả về giá trị của một biến*

– composer.gotoScene() *--chuyển đổi một scene cụ thể*

– composer.hideOverlay() *--ẩn/loại bỏ các overlay scene hiện tại*

– composer.loadScene() *--tải scene xác định*

– composer.newScene() *--tạo ra các đối tượng scene mới*

– composer.removeHidden() *--loại bỏ các scene trừ các scene đang hoạt động*

– composer.removeScene() *--loại bỏ scene nào đó được chỉ định*

– composer.setVariable() *--đặt giá trị cho một biến*

– composer.showOverlay() *--tải scene overlay trên scene đang hoạt động*

Ví dụ sau đây được sử dụng để tạo nên các đối tượng scene mới. Lưu ý rằng ví dụ này bao gồm tất cả các hàm sự kiện trong scene, có thể chỉ cần sử dụng các sự kiện khi muốn xử lý.

*--Require composer*

**local** composer = require("composer") *--Tạo mới một composer*

**local** scene = composer.newScene()

*--Hàm sự kiện create*

**function** scene:create(event)

**local** sceneGroup = self.view *--Tạo ra các đối tượng hiển thị*

*-- Tạo ra các đối tượng cần thiết cho cảnh ở đây*

**end**

*--Hàm sự kiện show*

**function** scene:show(event)

**local** sceneGroup=self.view

**local** phase=event.phase  *-- Khời tạo biến trạng thái sự kiên*

**if** ( phase == "will" ) **then**

*-- Những câu lệnh dưới đây sẽ thực thi trước khi scene hiển thị*

**elseif** ( phase == "did" ) **then**

*-- Những câu lệnh dưới đây sẽ thực thi sau khi scene hiển thị*

**end**

**end**

*--Hàm sự kiện ẩn*

**function** scene:hide(event)

**local** sceneGroup = self.view

**local** phase = event.phase

**if** (phase == "will" ) **then**

*-- Những câu lệnh dưới đây sẽ thực thi sau khi scene ẩn*

**elseif** (phase == "did" ) **then**

*-- Những câu lệnh dưới đây sẽ thực thi sau khi scene ẩn*

**end**

**end**

*--Hàm sự kiện phá huỷ̉*

**function** scene:destroy(event)

**local** sceneGroup = self.view

*-- Những câu lệnh dưới đây sẽ thực thi khi scene bị huỷ bỏ.*

*-- Bao gồm các việc dừng timer hay xoá các nhóm đối tượng hiển thị*

**end**

#### Đối tượng lớp widget

Trong lớp widget có một số hàm cụ thể như: newButton(), newPickerWheel(), newProgressView(), newScrollView(), newSegmentedControl(), newSlider(), newSpinner(), newStepper(), newSwitch(), newTabBar(), newTableView(), setTheme()

#### Thao tác với File

**Read và write các file** - dữ liệu dưới dạng text đơn giản, JSON, XML, dữ liệu local SQLite, vv - là cần thiết để phát triển ứng dụng. Có vô số trường hợp mà ứng dụng của bạn có thể cần phải read và/hoặc write một file, nhưng đây là một vài trường hợp phổ biến :

– Save/Load thiết lập người dùng hoặc các thống kê khác.

– Save "app state" vào một file và read nó trong một session sau này, vì vậy người dùng có thể quay trở lại chính xác nơi mà họ rời đi, ngay cả khi ứng dụng đã thoát.

– Tạo ra một file local HTML và load nó trong một native Web view.

– Load dữ liệu từ một file đã được download từ một máy chủ và sử dụng dữ liệu đó trong ứng dụng.

**Lấy đường dẫn file**

– Các hàm system.pathForFile() là nền tảng của tất cả các thao tác với filetrong Corona. Lua đòi hỏi toàn bộ đường dẫn của một file khi read và write, nhưng hệ điều hành di động như iOS đã “che giấu” file hệ thống thông qua "sandboxing". Điều này gây khó khăn khi xác định chính xác nơi mà file đang ở. May mắn thay, Corona đã đơn giản hóa điều này với các hàm system.pathForFile(). API này trả về một đường dẫn mà nó tương thích với file I/O.

**Lưu ý:** Tham số baseDirectory là một hằng số tùy chọn tương ứng với thư mục cơ sở mà file được đặt. Nếu không chỉ định, mặc định là system.ResourceDirectory. Xem phần System Directories dưới đây để biết thêm thông tin.

**Writing Files** (Ghi tập tin):

– system.pathForFile() - trả về đường dẫn cho các file để write.

– io.open() - Hàm này sẽ mở các file để write (hoặc read). Nó trả về một file xử lý mới, thiết lập vào file. Tham số thứ hai, "w", tương ứng với "mode" tin sẽ được mở ra trong đó. Mệnh lệnh này là những gì bạn sẽ thực hiện với các file. Trong trường hợp này, "w" chỉ write và nói với Corona tạo (write) một file mới, hoặc write đè lên file nếu nó đã tồn tại.

– file: write() - ghi vào file chỉ định

– io.close() - đóng file xử lý và kết thúc quá trình I/O.

Ví dụ

**local** saveData = "My app state data"

**local** path = system.pathForFile ( "myfile.txt", system.DocumentsDirectory )

**local** file= io.open( path, "w" )

file:write(saveData)

io.close(file)

file = **nil**

**Reading Files** (Đọc tập tin):

– io.open() - Hàm này sẽ mở ra các file và trả về các file xử lý, thiết lập thành file trong ví dụ này. Lần này, tham số thứ hai phải được thiết lập là "r" tương ứng với chế độ read.

– file: read() - giả sử các file được mở trong chế độ read tương ứng, phương thức file:read() sẽ read nội dung của file và đặt nó vào biến savedData.

– io.close() - đóng file xử lý và kết thúc quá trình I/O.

– Kiểu lấy nội dung ra: có 3 trường hợp

+ \*l: lấy từng dòng.

+ \*a: lấy toàn bộ nội dung.

+ \*n: đọc từng dòng cho đến khi gặp chuỗi ký tự thì kết thúc.

Ví dụ:

**local** path = system.pathForFile( "myfile.txt" , system.DocumentsDirectory)

**local** file = io.open(path, "r" )

**local** saveData = file:read("\*a")

io.close(file)

file = nil

**Reading Lines** (Đọc từng dòng)

Một cách phổ biến khi read dữ liệu là xử lý mỗi dòng của một tập tin và sử dụng nó cho một số mục đích cụ thể. Ví dụ, nếu có một tập tin văn bản, thể hiện một danh sách các sản phẩm cùng với một mức giá cho mỗi thứ, có thể tạo vòng lặp cho tập tin theo dòng và xuất ra hoặc parse mỗi dòng riêng biệt.

Hàm file:lines() kết hợp với một vòng lặp for sẽ thực hiện việc này:

Ví dụ:

**local** path = system.pathForFile( "myfile.txt" , system.DocumentsDirectory)

**local** file = io.open(path, "r" )

**for** line in file:lines() **do**

print(line)

**end**

io.close(file)

file = nil