**NGHIÊN CỨU VÀ SỬ DỤNG ANDENGINE ĐỂ LẬP TRÌNH TRÒ CHƠI CHO CÁC THIẾT BỊ DI ĐỘNG ANDROID**

Đào Trọng Nghĩa

*Khóa QH-2009-I/CC, ngành Công nghệ thông tin*

**Tóm tắt Khóa luận tốt nghiệp:**

Hiện nay, các thiết bị di động ngày càng thông minh và ưu việt hơn. Việc phát triển bùng nổ của các thiết bị di động thời gian gần đây kéo theo sự phát triển mạnh mẽ của ngành công nghiệp nội dung số nói chung và ngành công nghiệp phát triển trò chơi nói riêng. AndEngine là một Game Engine hoàn toàn miễn phí giúp cho việc phát triển các trò chơi cho các thiết bị di động Android nhanh hơn và dễ dàng hơn, đồng thời nó là một Game Engine phù hợp với những ai bắt đầu nghiên cứu về Game Engine dành cho Android, tạo tư duy và nền tảng để nghiên cứu những Game Engine khác có thu phí và mạnh mẽ hơn. Khóa luận này tập trung nghiên cứu các thành phần trong AndEngine và cách sử dụng AndEngine để lập trình trò chơi xếp tranh.

***Từ khóa:***andengine, lập trình trò chơi sử dụng andengine, game andengine gles1.

**MỤC LỤC**

[MỞ ĐẦU 3](#_Toc355869298)

[CHƯƠNG 1 4](#_Toc355869299)

[TỔNG QUAN VỀ TRÒ CHƠI DÀNH CHO ĐIỆN THOẠI DI ĐỘNG 4](#_Toc355869300)

[1.1. Thị trường trò chơi dành cho thiết bị di động 4](#_Toc355869301)

[1.2. Phân loại sơ lược các thể loại loại trò chơi trên thiết bị di động 5](#_Toc355869302)

[1.3. Các Game Engine thông dụng hiện nay cho Android 7](#_Toc355869303)

[CHƯƠNG 2 8](#_Toc355869304)

[TỔNG QUAN VỀ ANDENGINE 8](#_Toc355869305)

[2.1. Khái quát về AndEngine 8](#_Toc355869306)

[2.1.1. Khái niệm AndEngine 8](#_Toc355869307)

[2.1.2. Ưu nhược điểm của AndEngine 8](#_Toc355869308)

[2.1.3. Cài đặt AndEngine 9](#_Toc355869309)

[2.1.4. Các khái niệm căn bản trong AndEngine 9](#_Toc355869310)

[2.1.5. Chu kỳ sống của một trò chơi trong AndEngine 10](#_Toc355869311)

[2.2. Các thành phần trong AndEngine 12](#_Toc355869312)

[2.2.1. Entity 12](#_Toc355869313)

[2.2.2. Layers 15](#_Toc355869314)

[2.2.3. Scenes 15](#_Toc355869315)

[2.2.4. Modifier 16](#_Toc355869316)

[2.2.5. Vẽ đường thẳng và hình chữ nhật trong AndEngine 22](#_Toc355869317)

[2.2.6. Sprite 22](#_Toc355869318)

[2.2.7. Text 26](#_Toc355869319)

[2.2.8. Âm thanh 27](#_Toc355869320)

[CHƯƠNG 3 29](#_Toc355869321)

[XÂY DỰNG TRÒ CHƠI XẾP TRANH SỬ DỤNG ANDENGINE 29](#_Toc355869322)

[3.1. Mô tả trò chơi 29](#_Toc355869323)

[3.2. Mô tả giao diện trò chơi 30](#_Toc355869324)

[3.3. Công cụ phát triển trò chơi 31](#_Toc355869325)

[3.4. Lập trình trò chơi 31](#_Toc355869326)

[3.4.1. Các layout xml 31](#_Toc355869327)

[3.4.2. Dữ liệu của các màn chơi 31](#_Toc355869328)

[3.4.3. Dữ liệu hình ảnh và âm thanh 32](#_Toc355869329)

[3.4.4. Sơ đồ các package và các class trong trò chơi 32](#_Toc355869330)

[3.6. Kết quả và định hướng phát triển trò chơi trong tương lai 39](#_Toc355869331)

[3.6.1. Kết quả lập trình trò chơi 39](#_Toc355869332)

[3.6.2. Định hướng phát triển trò chơi trong tương lai 41](#_Toc355869333)

[KẾT LUẬN 41](#_Toc355869334)

[TÀI LIỆU THAM KHẢO 42](#_Toc355869335)

# 

# MỞ ĐẦU

Với sự phát triển của khoa học công nghệ, thiết bị di động đang ngày trở nên phổ biến và trở thành một vật dụng không thể thiếu trong đời sống của mỗi cá nhân. Thiết bị di động ngày càng thông minh và ưu việt hơn, người dùng có thể điện thoại, gửi tin nhắn, lướt web, thực hiện các giao dịch mua bán, các dịch vụ giá trị gia tăng, chơi trò chơi trên điện thoại di động. Theo hãng nghiên cứu BI Intelligence khảo sát đầu năm 2013 về hành vi sử dụng các thiết bị di động của người dùng, việc chơi trò chơi chiếm tới 39% các hoạt động của người dùng trên điện thoại thông minh (smartphones) và tới 67% các loại máy tính bảng (Tablets). Điều này cho thấy rằng ngành công nghiệp sản xuất trò chơi sẽ ngày càng phát triển là ngành công nghiệp lớn nhất trong lĩnh vực cung cấp nội dung số. Các Game Engine giúp cho việc phát triển các trò chơi được nhanh hơn và dễ dàng hơn, việc nắm bắt nội dung của những Game Engine lớn là một điều không dễ dàng đối với các nhà phát triển. Khóa luận này sẽ tập trung nghiên cứu AndEngine – một Game Engine hoàn toàn miễn phí để phát triển các trò chơi cho các thiết bị di động Android, giúp các nhà phát triển có thể tạo ra các trò chơi một cách nhanh chóng và dễ dàng hơn; đồng thời. AndEngine phù hợp với những đối tượng bắt đầu làm quen với Game Engine cho Android, tạo nền tảng kiến thức chung về một Game Engine, giúp cho việc nắm bắt những Game Engine lớn có phí và mạnh mẽ một cách dễ dàng hơn. Nội dung của Khóa luận được chia làm 3 chương.

**Chương 1: Tổng quan về phát triển trò chơi dành cho thiết bị di động**

Chương này khái quát các vấn đề về việc phát triển trò chơi dành cho các thiết bị di động hiện nay bao gồm thị trường trò chơi, các thể loại trò chơi, việc sử dụng Game Engine trong phát triển trò chơi

**Chương 2: Nội dung AndEngine**

Chương này tập trung vào các vấn đề lý thuyết trong AndEngine, ưu điểm và nhược nhược điểm khi sử dụng AndEngine để phát triển trò chơi, vấn đề sử dụng AndEngine để phát triển trò chơi hiện nay.

**Chương 3: Sử dụng AndEngine để lập trình trò chơi xếp tranh**

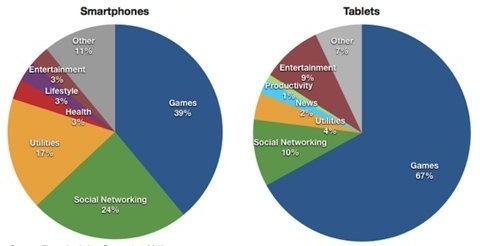
Chương này sẽ sử dụng AndEngine để tạo 1 trò chơi đơn giản (trò chơi xếp tranh).

# CHƯƠNG 1

# TỔNG QUAN VỀ TRÒ CHƠI DÀNH CHO ĐIỆN THOẠI DI ĐỘNG

## Thị trường trò chơi dành cho thiết bị di động

Theo bản báo cáo mang tên “The Mobile Economy 2013” tạm dịch là “Nền kinh tế di động 2013” của hiệp hội di động toàn cầu GSMA (Global Systems for Mobile communications Association) hiện đang có khoảng 3,2 tỷ người dùng thiết bị di động trên toàn cầu và sẽ tăng lên khoảng 4 tỷ người vào năm 2018. Điều này là cơ hội cho sự phát triển mạnh mẽ của các dịch vụ nội dung số nói chung và các trò chơi nói riêng cho thiết bị di động. Theo hãng nghiên cứu BI Intelligence khảo sát đầu năm 2013 về hành vi sử dụng các thiết bị di động của người dùng, việc chơi trò chơi chiếm tới 39% các hoạt động của người dùng trên điện thoại thông minh (smartphones) và tới 67% các loại máy tính bảng (Tablets).



*Hình 1.1.1: Biểu đồ các hoạt động của người dùng trên Smartphones và Tablets*

Các trò chơi sau khi được xây dựng xong có thể được tiêu thụ thông qua các kho ứng dụng, như vậy trò chơi được xuất bản mang tính toàn cầu, không bị thu hẹp trong phạm vi lãnh thổ, hay nói cách khác, cơ hội cho những nhà sản xuất trò chơi kết nối được với người dùng là như nhau. Các nhà sản xuất và phân phối trò chơi không những có thể thu lợi thông qua việc bán các quảng cáo trong trò chơi mà còn có thể thu lợi từ việc bán các trò chơi của mình thông qua kho ứng dụng, bán các phiên bản nâng cấp trò chơi, các tính năng bổ xung cho trò chơi hoặc thu phí của người chơi…

Như vậy, thị trường trò chơi dành cho các thiết bị di động đã, đang và sẽ phát triển một cách bùng nổ trong tương lai gần.

* 1. **Phân loại sơ lược các thể loại loại trò chơi trên thiết bị di động**

Một trò chơi thông thường sẽ có các thuộc tính cơ bản sau:

* Mục tiêu của trò chơi (A goal): Trò chơi cần phải xác định rõ ràng một mục tiêu để cho người chơi đạt được. Mục tiêu là một thách thức nhưng có thể đạt được, mục tiêu làm cho người chơi cảm giác được mục đích chơi trò chơi
* Các quy tắc (Rules): Trò chơi có những quy định mà tất cả người chơi cần phải đồng ý tuân theo, các quy tắc thường làm cho việc đạt được mục tiêu của trò chơi sẽ khó khăn hơn, do đó sẽ khuyến khích được trí tò mò và tính sáng tạo của người chơi
* Thông tin phản hồi (Feedback): Trò chơi sẽ có những phản hồi thích hợp cho người người chơi mỗi khi hành động của người chơi bắt đầu, đang diễn ra hoặc kết thúc
* Tham gia tự nguyện (Voluntary participation): Tất cả người chơi chỉ chơi trò chơi khi thực sự họ muốn chơi, không có một sự ép buộc người chơi “phải” chơi trò chơi, nhưng khi tham gia trò chơi thì người chơi tình nguyện chấp hành mục tiêu của trò chơi, các quy tắc và các thông tin phản hồi.

Các nhà phát triển trò chơi luôn liên tục sáng tạo ra các thể loại trò chơi mới, tinh vi hơn và độc đáo hơn, đồng thời cũng có hướng lai tạp các thể loại trò chơi lại với nhau. Do đó, các trò chơi có thể được phân loại một cách tương đối như sau:

***Trò chơi hành động (Action Game):*** Trò chơi đề cao khả năng chiến đấu của nhân vật trong từng khu vực rộng lớn; ít đề cập đến những câu đố, thông điệp phức tạp, hay những trận đánh tay đô. Người chơi “nhìn” bằng mắt của chính nhân vật. Dạng này yêu cầu bạn phải có khả năng phản xạ nhanh, phối hợp tay - mắt tốt, am tường và hiểu rõ các loại vũ khí có trong game;

***Trò chơi nhập vai ( Role – Playing Game):*** người chơi sẻ vào vai một nhân vật hoặc một nhóm để tham gia các sự kiện, giao tiếp với các nhân vật phụ, bước vào những cuộc chiến (theo lượt hoặc thời gian thực) với một quá trình xây dựng nhân vật thông qua việc tăng cấp độ, phát triển các kỹ năng, tính cách...

***Trò chơi phiêu lưu (Adventure Game):*** Nét cơ bản của của thể loại này là khám phá các vùng đất, tìm các vật dụng và giải đố; có rất ít hoặc gần như là không có các cảnh mang tính “hành động”. Thể loại trò chơi này không chú trọng vào sự phát triển nhân vật, cũng như tăng cấp độ. Trò chơi phiêu lưu có cốt truyện rất chặt chẽ và rất dài, đòi hỏi người chơi phải vận dụng trí tuệ, lòng nhẫn nại và sự quyết đoán để chọn hướng đi tốt nhất.

***Trò chơi chiến thuật (Strategy Game):*** Thể loại này chú trọng cách quản lý nguồn tài nguyên và quân đội trước những đối thủ do máy hoặc người chơi khác điều khiển, hầu hết đều “dính dáng” chiến tranh. Nhìn chung, thể loại này được chia làm 2 dạng phân biệt khác nhau. Thứ nhất là chiến thuật thời gian thực (RTS), thứ 2 là chiến thuật theo lượt(TBS). RTS có đặc trưng là hành động (diễn biến) xảy ra liên tục, không ngừng do danh sách các lệnh được đưa ra xuyên suốt trong trò chơi, đòi hỏi người chơi phải có cách “điều binh khiển tướng” hợp lý. TBS đòi hỏi người chơi phải biết cách tính toán “nặng nhẹ” cho mỗi nước đi của mình

***Trò chơi kinh dị:*** Gần giống Trò chơi hành động – Trò chơi phiêu lưu nhưng luôn tạo ra cảm giác sợ hãi và hồi hộp cho người chơi. Yếu tố kinh dị trong game thường được tạo ra từ ma quỷ (demon), linh hồn (ghost), thây ma (zombie).

***Trò chơi mô phỏng (Simulation Game):*** Yêu cầu duy nhất của thể loại này là phải mô phỏng làm sao cho càng giống thế giới thực càng tốt

***Trò chơi thể thao (Sport Game):*** Được xem là một nhánh rẽ của trò chơi mô phỏng, trò chơi thể thao mô tả một môn thể thao của thế giới thực diễn ra trong môi trường 3D. Game thường được tổ chức theo cơ cấu mùa giải, và cho phép người chơi đảm nhận các vị trí huấn luyện viên, ông bầu... bên cạnh các tính năng trao đổi, chuyển nhượng cầu thủ, quản lý ngân sách

***Trò chơi giải đố (Puzzle Game):*** Đây là lĩnh vực dành cho những người yêu thích các câu đố cực... khó, hơn hẳn trò chơi phiêu lưu. Game đòi hỏi người chơi phải quyết định nhanh chóng (và chính xác) các vấn đề đã được đặt ra theo hướng tư duy, suy luận logic. Đôi khi các vấn đề này lại được lồng ghép, xâu chuỗi để tăng độ khó.

***Trò chơi âm nhạc (Rhythm Game):*** Trò chơi tập trung vào âm nhạc và thời gian, điều khiển trong trò chơi chủ yếu bằng cách nhấn nút trên một bộ điều chỉnh thông thường

***Trò chơi đối kháng (Fighting Game):*** Đặc trưng của thể loại này là những trận so tài tay đôi giữa các nhân vật.

***Trò chơi giải trí (Entertaiment Game):*** Là dạng game ngắn, đơn giản, được thiết kế cho nhiều người chơi và... dễ chơi

***Trò chơi nhập vai trực tuyến ( Massively -multiplayer online Game):*** Đặc trưng của thể loại này là có sự tham gia của một số lượng lớn người vào một thế giới liên tục, thông qua giao tiếp cộng đồng trực tuyến với những người chơi khác

* 1. **Các Game Engine thông dụng hiện nay cho Android**

- AndEngine

- Cocos2d

- Cocos2d-x

- Age

- Age

- Candroidengine

- Catcake

- Cloak

- Forget3d

- JmonkeyEngine

- ReplicaIsland

- Rokon

# 

# CHƯƠNG 2

# TỔNG QUAN VỀ ANDENGINE

## 2.1. Khái quát về AndEngine

### 2.1.1. Khái niệm AndEngine

AndEngine là một Game engine mã nguồn mở hoàn toàn miễn phí dựa trên openGL ES, giúp các nhà phát triển trò chơi cho Android nhanh chóng và dễ dàng hơn. AndEngine đượng xây dựng bởi Nicolas Glamlich, về cơ bản AndEngine là một gói các lớp của java được viết sẵn và đóng gói.

AndEngine Tương thích với môi trường Eclipse

### 2.1.2. Ưu nhược điểm của AndEngine

**Ưu điểm**

* Engine viết hoàn toàn bằng Java nên người lập trình chỉ cần biết về Java và Android là có thể học được, Engine tương đối dễ sử dụng hơn so với các Engine khác, thuận lợi cho những người bắt đầu nghiên cứu về Game Engine để nghiên cứu những Game Engine có phí lớn hơn và mạnh hơn.
* Hỗ trợ nhiều tiện ích khi xây dựng trò chơi
* Cộng đồng lập trình đông đảo
* Hoàn toàn miễn phí

**Nhược điểm**

* Hiệu năng không cao
* Không tận dụng được các mã chương trình C hoặc C++ đã có sẵn (khác biệt với Engine Cocos2d-x).
* Một trò chơi khi viết bằng AndEngine chỉ có thể chạy được trên các thiết bị di động chạy hệ điều hành Android, muốn trò chơi chạy trên nền tảng khác như iOS hoặc Window Phone thì cần xây dựng lại từ đầu (Điều này hạn chế so với Cocos2d-x và HTML5)

*Phiên bản AndEngine GLES2 đang được phát triển để khắc phục các nhược điểm trên của AndEngine.*

### 2.1.3. Cài đặt AndEngine

- Download file andengine.jar (hoặc file andenginesvgtextureextension.jar nếu trong game có sử dụng ảnh vector) tại www.andengine.org

- Thêm file vừa download vào thư mục libs của dự án Android

- Click chuột phải vào file -> Build path -> Add to build path

### 2.1.4. Các khái niệm căn bản trong AndEngine

- Camera: là 1 lớp sẽ định nghĩa 1 khung chữ nhật của quang cảnh (Scene) mà nội dung sẽ hiển thị lên màn hình bên trong khung chữ nhật này.

- Scene: là 1 lớp sẽ chứa tất cả những đối tượng thực thể (entity) cần hiển thị trên màn hình

- Layer: Scene bao gồm các lớp đồ họa, các lớp chồng lên nhau

- Sprite: Là những hình ảnh mô tả các nhân vật trong trò chơi, kết cấu Sprite thường được nạp từ một tập hợp lớn các hình ảnh họa tiết cùng nằm trên một tờ họa tiết (Sprite sheet)

- Entity: Là bất cứ thứ gì có thể được thể hiện ra màn hình, tất cả các thực thể đều có các thuộc tính như màu sắc, sự di chuyển, quy mô, vị trí. Chúng có thể được thay đổi bằng cách thay đổi các bộ điều chỉnh (Modifiers)

- Modifier: Dùng để thay đổi các thuộc tính của một thực thể

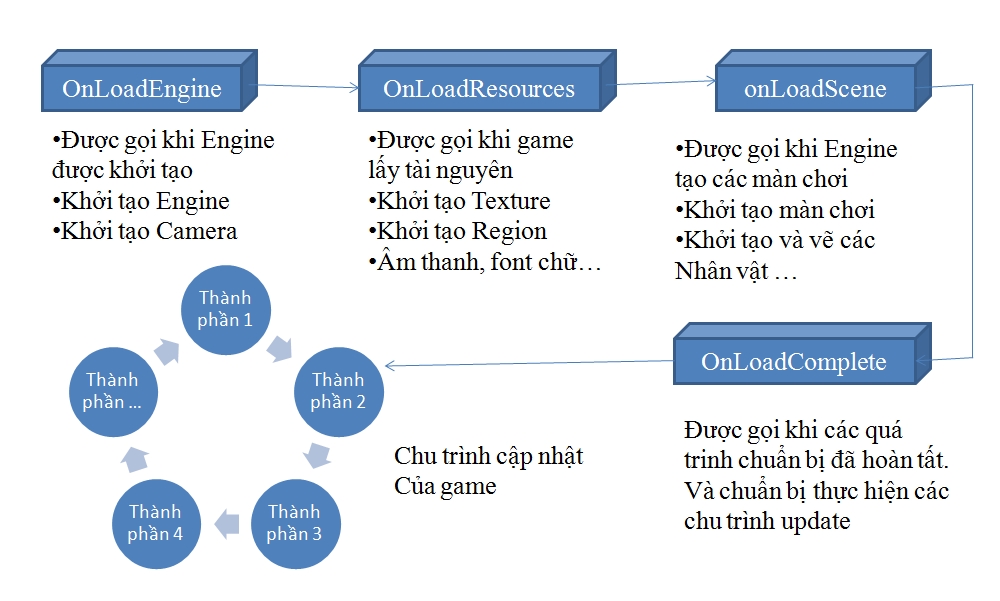
- Texture: là 1 lớp được xem như là bộ nhớ của hình ảnh. Nó ko trực tiếp lưu ảnh, nó chỉ xem như bộ nhớ để chứa ảnh đó. Ví dụ ảnh có kích thước: 45 x 126, thì cần phải tạo texture với kích thước X to hơn kích thước ảnh thật sự ( X phải là kết quả của mũ 2)

- TextureRegion: nạp ảnh vào texture và cắt ảnh theo 1 khung chữ nhật với kích thước tự chọn

- Engine: Dùng để quản lý tiến độ của trò chơi, mỗi engine chạy một scene. Nếu trò chơi là một bộ phim thì engine giống như đạo diễn hoặc nhà sản xuất khi sản xuất bộ phim đó, có nhiệm vụ nói với các mọi người những việc họ cần phải làm.

- BaseGameActivity: Đây là lớp kế thừa lớp Activity của Android, nó làm tất cả các công việc chung cho tất cả các Scene và thiết lập game engine

### 2.1.5. Chu kỳ sống của một trò chơi trong AndEngine



*Hình 2.6.1.1: Chu kỳ sống của một trò chơi trong AndEngine*

**Phương thức khởi tạo Engine**

@Override

public Engine onLoadEngine() {

this.mCamera = new Camera(0, 0, CAMERA\_WIDTH, CAMERA\_HEIGHT);

return new Engine(new EngineOptions(true, ScreenOrientation.LANDSCAPE,

new RatioResolutionPolicy(CAMERA\_WIDTH, CAMERA\_HEIGHT), this.mCamera));

}

Engine sẽ làm một số công việc cho các nhà phát triển trò chơi

* Chỉ đạo việc khởi tạo và bảo trì liên tục của OpenGL SurfaceView mà chúng ta sử dụng để vẽ trên màn hình.
* Quản lý người dùng nhập vào từ bàn phím và màn hình cảm ứng.
* Quản lý đầu vào cảm biến từ gia tốc và định hướng (hướng la bàn), cảm biến.
* Quản lý thư viện của phông chữ văn bản.
* Tạo ra và quản lý việc nạp và chơi hiệu ứng âm thanh và âm nhạc.
* Cập nhật phần còn lại của AndEngine định kỳ để thúc đẩy trạng thái của trò chơi

đang được chơi.

Hàm khởi tạo mặc định của đối tượng Engine là

public Engine(final EngineOptions pEngineOptions)

trong đó phương thức EngineOptions có các thuộc tính khởi tạo như sau

public EngineOptions(

final boolean pFullscreen,

final ScreenOrientation pScreenOrientation,

final IResolutionPolicy pResolutionPolicy,

final Camera pCamera

)

Các biến truyền vào EngineOptions() cho chúng ta biết chúng ta muốn cài đặt Engine cho trò chơi này như thế nào

* pFullScreen: Nếu là giá trị true có nghĩa rằng chúng ta muốn trò chơi được thể hiện trên toàn diện tích màn hình
* ScreenOrientaion: có 2 giá trị: Nếu là LANSCAPE tưc là yêu cầu màn hình hiển thị nằm ngang, nếu là PORTRAIT tức là màn hình hiển thị theo chiều dọc
* pResolutionPolicy: cho Engine biết cách thức chúng ta muốn hiển thị trò chơi đối với các loại màn hình khác nhau

+ RatioResolutionPolicy: duy trì tỉ lệ ban đầu của đồ họa và sẽ trải rộng màn hình tối đa có thể

+ FillResolutionPolicy: việc hiển thị sẽ lấp đầy màn hình, bỏ qua tỉ lệ.

+ FixedResolutionPolicy: sử dụng kích thước cố định ban đầu, không mở rộng đồ họa để phù hợp với kích cỡ màn hình.

+ RelativeResolutionPolicy: Cho phép tùy chỉnh theo từng kích thước màn hình cụ thể

* pCamera: Là đối tượng Camera, định nghĩa view mà người chơi sẽ nhìn thấy các Scene, hàm khởi tạo của Camera là

public Camera(final float pX, final float pY, final float pWidth, final

float pHeight)

trong đó pX và pY là tọa độ gốc, pWidth và pHeight là chiều rộng và chiều cao của vùng scene có thể nhìn thấy, tất cả các đơn vị đều tính bằng pixels

**Phương thức onLoadResource()**

* Được gọi khi trò chơi lấy tài nguyên
* Khởi tạo Texture
* Khởi tạo TextureRegion
* Khởi tạo âm thanh, font chữ…

**Phương thức onLoadScene()**

* Được gọi khi Engine tạo các màn chơi
* Khởi tạo màn chơi
* Khởi tạ và vẽ các nhân vật trong trò chơi

**Phương thức onLoadComplete()**

* Được gọi đến khi tất cả các quá trình chuẩn bị đã hoàn tất và chuẩn bị thực hiện các chu trình cập nhật

## 2.2. Các thành phần trong AndEngine

### 2.2.1. Entity

Trong AndEngine, hầu như tất cả các phần tử có thể nhìn thấy được (bao gồm cả scene) trong trò chơi đều thuộc lớp con của lớp Entity. Entity chứa các thuộc tính chung của những thứ có thể thấy được, các thuộc tính đó có thể được truy cập thông qua thuộc tính getter và setter và chúng có thể được tùy biến thông qua các lớp Modifier. Các thuộc tính đó như sau:

|  |  |  |
| --- | --- | --- |
| 1 | float mX, mY | vị trí hiện thời của Entity |
| 2 | float mInitialX, mInitialY | vị trí khởi tạo của Entity |
| 3 | boolean mVisible | xét xem các Entity có được nhìn thấy hay không |
| 4 | boolean mIgnoreUpdate | Các Entity có cần chú ý đến việc cập nhật Entity hay không |
| 5 | int mZindex | Nơi Entity xuất hiện trong ngăn xếp của các Entity được hiển thị |
| 6 | IEntity mParent | Xác định cha của một Entity |
| 7 | SmartList<IEntity> mChildren | Tập hợp các con của một Entity |
| 8 | EntityModifierList mEntityModifiers | danh sách các Modifier để áp dụng cho một thực thể |
| 9 | UpdateHandlerList mUpdateHandlers | danh sách các Update Handlers để áp dụng cho một Entity |
| 10 | float mRed, mGreen, mBlue | Định nghĩa màu sắc của Entity |
| 11 | float mAlpha | Xác định tính trong suốt của Entity |
| 12 | float mRotation | Xác định vòng xoay hiện tại của thực thể Entity |
| 13 | float mRotationCenterX, mRotationCenterY | Điểm về mà các thực thể quay |
| 14 | float mScale | xác định việc mở rộng kích thước thực thể tại thời điểm hiện tại |
| 15 | float mScaleCenterX, mScaleCenterY | điểm về mà các thực thể mở rộng |

**Khởi tạo Entity**

*Cách 1:* Entity()

*Cách 2*: Entity(final float pX, final float pY) trong đó pX và pY là vị trí hiện tại của Entity

**Vị trí của entity**

Vị trí của entity được tính theo tọa độ trung tâm của entity

|  |  |  |
| --- | --- | --- |
| 1 | float getX()  float getY() | Trả về các thành phần hiện tại của Entity |
| 2 | float getInitialX()  float getInitialY() | Trả về các thành phần ban đầu của Entity |
| 3 | void setPosition(final float pX, final float pY) | Đặt vị trí hiện tại cho Entity |
| 4 | setPosition(final IEntity pOtherEntity) | Đặt vị trí hiện tại của Entity giống với vị trí của một Entity khác |
| 5 | void setInitialPosition() | Đặt vị trí hiện thời của Entity trùng với vị trí khởi tạo Entity |

**Scale**

|  |  |  |
| --- | --- | --- |
| 1 | boolean isScaled() | Trả true nếu quy mô hiện tại của entity khác 1.0 |
| 2 | float getScaleX()  float getScaleY() | Trả về tỉ lệ scale theo các chiều x và y |
| 3 | void setScaleX(final float pScaleX)  void setScaleY(final float pScaleY) | Đặt giá trị scale theo chiều x và y |
| 4 | void setScale(final float pScale) | Đặt giá trị scale chung cho entity |
| 5 | void setScale(final float pScaleX,  final float pScaleY) | Đặt giá trị scale theo chiều x và y tại cùng một thời điểm |
| 6 | float getScaleCenterX()  float getScaleCenterY() | Trả về những vị trí điểm trung tâm của scale |
| 7 | void setScaleCenterX(final float pScaleCenterX)  void setScaleCenterY(final float pScaleCenterY) | Đặt giá trị vị trí điểm trung tâm của scale |
| 8 | void setScaleCenter(final float pScaleCenterX,  final float pScaleCenterY) | Đặt giá trị vị trí điểm trung tâm của scale tại cùng một thời điểm |

**Màu sắc entity**

float getRed()

float getGreen()

float getBlue()

float getAlpha()

void setColor(final float pRed, final float pGreen, final float pBlue)

void setColor(final float pRed, final float pGreen, final float pBlue, final float

pAlpha)

**Sự xoay entity (Rotation)**

float getRotation() trả về giá trị xoay hiện tại

setRotation(final float pRotation) đặt giá trị xoay hiện tại cho entity

float getRotationCenterX()

float getRotationCenterY() trả về vị trí trung tâm xoay hiện tại

void setRotationCenterX(final float pRotationCenterX)

void setRotationCenterY(final float pRotationCenterY) đặt các thành phần cho các vị trí trung tâm xoay

void setRotationCenter(final float pRotationCenterX, final float pRotationCenterY) đặt các thành phần cho các vị trí trung tâm xoay tại cùng một thời điểm

**Quản lý các con của entity**

IEntity IEntity getFirstChild()

IEntity getLastChild() lấy con đầu tiên và con cuối cùng của entity

void attachChild(final IEntity pEntity) thêm một con vào entity (được thêm vào sau con cuối cùng của entity)

void detachChildren()

boolean detachChild(final IEntity pEntity) gỡ toàn bộ các con của entity ra

int getChildCount() trả về số lượng các con của entity

IEntity getChild(final int pIndex) trả về con thứ n của entity

void sortChildren() sắp xếp các con của entity

**Quản lý các bộ điều chỉnh (Modidfier)**

void registerEntityModifier(final IEntityModifier pEntityModifier) đăng ký mộ bộ điều chỉnh với entity, nó sẽ được áp dụng khi entity được hiển thị

boolean unregisterEntityModifier(final IEntityModifier pEntityModifier) loại bỏ một bộ điều chỉnh khỏi entity, hàm sẽ trả về giá trị true nếu việc loại bỏ thành công

void clearEntityModifiers() loại bỏ toàn bộ các bộ điêu chỉnh khỏi entity

### 2.2.2. Layers

Một layer trong AndEngine là một lớp con của lớp entity, nó cho phép chúng ta tạo ra các lớp đồ họa để gắn trong scene. Có 2 phương thức khởi tạo layer

Layer()

Layer(final float pX, final float pY)

Nếu pX và pY đều bằng 0 thì hai phương thức khởi tạo là tương đương nhau

### 2.2.3. Scenes

AndEngine cung cấp một lớp Scene để chúng ta có thể tạo ra mỗi cảnh trong trò chơi. Các Scene có các thuộc tính sau

* Scene mParentScene:Tất cả các Scene có thể có một Scene cha tùy chọn
* Scene mChildScene: Tất cả các Scene có thể có một Scene con tùy chọn
* SmartList<ITouchArea> mTouchAreas: Mỗi Scene có một danh sách các khu vực chạm và biết giải quyết như thế nào khi người dùng chạm vào các khu vực đó
* IOnSceneTouchListener mOnSceneTouchListener: Một sự lắng nghe được đăng ký sẽ được gọi tới khi Scene được chạm
* IOnAreaTouchListener mOnAreaTouchListener: Một sự lắng nghe riêng biệt được đăng ký khi một khu vực cảm ứng được chạm
* IBackground mBackground: Mỗi Scene sẽ có màu nền, mặc định là màu đen
* boolean mOnAreaTouchTraversalBackToFront: Cờ chỉ định thứ tự sự va chạm sẽ được xử lý
* Các lớp đồ họa (layer) có thể được gắn vào Scene như là các con của Scene

**Khởi tạo Scene**

Để khởi tạo Scene xử dụng phương thức

Scene(final int pLayerCount)

pLayerCount là số lớp đồ họa trọng Scene

**Quản lý nền Scene**

Theo mặc định, nền của Scene sẽ là màu đen (đặc), chúng ta có thể xử dụng một số phương thức để tùy chỉnh và truy cập tới nền của Scene

IBackground getBackground()

void setBackground(final IBackground pBackground)

void setBackgroundEnabled(final boolean pEnabled)

boolean isBackgroundEnabled()

**Quản lý Scene con**

Các Scene có thể có một Scene con không phải là Layer, Scene con này có vài phương thức để kết hợp với Scene cha.

Để gắn Scene con vào Scene cha có thể xử dụng các phương thức

void setChildScene(final Scene pChildScene)

void setChildScene(final Scene pChildScene, final boolean pModalDraw,

final boolean pModalUpdate, final boolean pModalTouch)

void setChildSceneModal(final Scene pChildScene)

Scene getChildScene() trả về Scene con

boolean hasChildScene() Xác định xem Scene có Scene con hay không

void clearChildScene() loại bỏ Scene con ra khỏi Scene cha

**Quản lý Layer**

Chỉ có 1 phương thức quản lý Layer được định nghĩa trong Scene là

void sortLayers()

Nó sắp xếp các Layer (là con của Scene) theo chiều z

**Quản lý Scene cha**

Các Scene có thể có 1 Scene cha, thứ tự cha con sẽ được đặt một cách tự động khi mà một Scene được thêm vào một Scene khác, để thêm Scene cha cho một Scene chúng ta xử dụng phương thức

void setParentScene(final Scene pParentScene)

### 2.2.4. Modifier

Đối với mỗi Entity, chúng ta có thể đặt vào bộ điều chỉnh để kiểm soát các thông số như vị trí, scale, màu sắc, …

Để đăng ký bộ điều chỉnh với thực thể xử dụng phương thức

entity.registerEntityModifier(new XxxModifier(p1, p2, …);

Các phương thức chung

XxxModifier clone() trả về một bản sao của bộ điều chỉnh

boolean isFinished() trả về true nếu bộ điều chỉnh đã kết thúc hoạt động

void setRemoveWhenFinished(final boolean pRemoveWhenFinished)

boolean isRemoveWhenFinished()

Dùng trong việc có loại bỏ bộ điều chỉnh khỏi thực thể sau khi đã kết thúc hay không

void setModifierListener(final IModifierListener<T> pModifierListener)

IModifierListener<T> getModifierListener()

Sets hoặc gets hành trình lắng nghe (được gọi đến khi bộ điều chỉnh kết thúc), chỉ một ModifierListener được đăng ký

**Điều chỉnh vị trí**

Các bộ điều chỉnh sẽ thay đổi vị trí hiện thời của Entity

MoveModifier(final float pDuration, final float pFromX, final float pToX,

final float pFromY, final float pToY, final IEntityModifierListener

pEntityModifierListener, final IEaseFunction

pEaseFunction)

MoveModifier(final float pDuration, final float pFromX,

final float pToX,

final float pFromY, final float pToY, final IEntityModifierListener

pEntityModifierListener, final IEaseFunction pEaseFunction)

MoveModifier(final float pDuration, final float pFromX, final float pToX,

final float pFromY, final float pToY, final IEntityModifierListener

pEntityModifierListener, final IEaseFunction pEaseFunction)

MoveModifier(final float pDuration, final float pFromX, final float pToX,

final float pFromY, final float pToY, final IEntityModifierListener

pEntityModifierListener, final IEaseFunction pEaseFunction)

Tất cả các phương thức trên làm cho Entity di chuyển theo cả hai hướng X và Y cùng một lúc, pDuration là độ lơn của thời gian tính bằng giây

MoveXModifier(final float pDuration, final float pFromX, final float pToX)

MoveXModifier(final float pDuration, final float pFromX, final float pToX, final

IEaseFunction pEaseFunction)

MoveXModifier(final float pDuration, final float pFromX, final float pToX, final

IEntityModifierListener pEntityModifierListener)

MoveXModifier(final float pDuration, final float pFromX, final float pToX, final

IEntityModifierListener pEntityModifierListener, final IEaseFunction

pEaseFunction)

MoveXModifier(final MoveXModifier pMoveXModifier)

MoveYModifier(final float pDuration, final float pFromY, final float pToY)

MoveYModifier(final float pDuration, final float pFromY, final float pToY, final

IEaseFunction pEaseFunction)

MoveYModifier(final float pDuration, final float pFromY, final float pToY, final

IEntityModifierListener pEntityModifierListener)

MoveYModifier(final float pDuration, final float pFromY, final float pToY, final

IEntityModifierListener pEntityModifierListener, final IEaseFunction

pEaseFunction)

MoveYModifier(final MoveYModifier pMoveYModifier)

Tất cả các bộ điều chỉnh trên sẽ di chuyển Entity theo riêng một hướng X, Y

**Điều chỉnh đường dẫn (PathModifier)**

Di chuyển Entity từ một điểm này tới một điểm khác sẽ có nhiều đường dẫn để làm việc đó. Có 3 cách để tạo đường dẫn

Path(final int pLength)

Path(final float[] pCoordinatesX, final float[] pCoordinatesY)

Path(final Path pPath)

Phương thức thứ nhất chỉ ra rằng đường dẫn sẽ chứa pLength điểm, sau đó đường dẫn được xây dựng bằng cách xử dụng các phương thức để thêm vào từng phân khúc

to(final float pX, final float pY)

Phương thức thứ 2 xử dụng một mảng các tọa độ X và Y để tạo đường dẫn và phương thức thứ 3 đơn giản nhất, xử dụng một đường dẫn mà đã tồn tại sẵn

***Để khởi tạo các bộ điều chỉnh đường dẫn có thể xử dụng các phương thức sau***

PathModifier(final float pDuration, final Path pPath)

PathModifier(final float pDuration, final Path pPath,

final IEaseFunction pEaseFunction)

PathModifier(final float pDuration, final Path pPath,

final IEntityModifierListener pEntityModiferListener)

PathModifier(final float pDuration, final Path pPath,

final IEntityModifierListener pEntityModiferListener,

final IEaseFunction pEaseFunction)

PathModifier(final float pDuration, final Path pPath,

final IEntityModifierListener pEntityModiferListener,

final IPathModifierListener pPathModifierListener)

PathModifier(final float pDuration, final Path pPath,

final IEntityModifierListener pEntityModiferListener,

final IPathModifierListener pPathModifierListener,

final IEaseFunction pEaseFunction)

***Ví dụ:***

/\*\*

final Path path = new Path(5).to(10, 10).to(10, 50).to(50, 50).to(50, 10).to(10, 10);

entity.registerEntityModifier(new LoopEntityModifier(new PathModifier(30,

path, null, new IPathModifierListener() {

@Override

public void onWaypointPassed(final PathModifier pPathModifier, final IEntity pEntity, final int pWaypointIndex) {

switch(pWaypointIndex) {

case 0:

entity.setColor(1.0f, 0.0f, 0.0f);

break;

case 1:

entity.setColor(0.0f, 1.0f, 0.0f);

break;

case 2:

entity.setColor(0.0f, 0.0f, 1.0f);

break;

case 3:

entity.setColor(1.0f, 0.0f, 0.0f);

break;

}

}

));

scene.getLastChild().attachChild(entity);

\*\*/

**Điều chỉnh Scale**

Các phương thức khởi tạo bộ điều chỉnh Scale

ScaleModifier(final float pDuration, final float pFromScale, final float pToScale)

ScaleModifier(final float pDuration, final float pFromScale, final float pToScale,

final IEaseFunction pEaseFunction)

ScaleModifier(final float pDuration, final float pFromScale, final float pToScale,

final IEntityModifierListener pEntityModifierListener)

ScaleModifier(final float pDuration, final float pFromScale, final float pToScale,

final IEntityModifierListener pEntityModifierListener, final

IEaseFunction pEaseFunction)

(Scale = 1.0f tức là kích thước của Entity chính là kích thước của Entity gốc)

ScaleModifier(final float pDuration, final float pFromScaleX, final float

pToScaleX, final float pFromScaleY, final float pToScaleY)

Các bộ điều chỉnh Scale theo hai hướng X và Y khác nhau

ScaleAtModifier(final float pDuration, final float pFromScale,

final float pToScale, final float pScaleCenterX, final float pScaleCenterY)

ScaleAtModifier(final float pDuration, final float pFromScale,

final float pToScale, final float pScaleCenterX, final float pScaleCenterY,

final IEaseFunction pEaseFunction)

ScaleAtModifier(final float pDuration, final float pFromScale,

final float pToScale, final float pScaleCenterX, final float pScaleCenterY,

final IEntityModifierListener pEntityModifierListener)

ScaleAtModifier(final float pDuration, final float pFromScale, final

float pToScale, final float pScaleCenterX, final float pScaleCenterY,

final IEntityModifierListener pEntityModifierListener,

final IEaseFunction pEaseFunction)

ScaleAtModifier(final float pDuration, final float pFromScaleX,

final float pToScaleX, final float pFromScaleY, final float pToScaleY,

final float pScaleCenterX, final float pScaleCenterY)

ScaleAtModifier(final float pDuration, final float pFromScaleX,

final float pToScaleX, final float pFromScaleY, final float pToScaleY,

final float pScaleCenterX, final float pScaleCenterY,

final IEaseFunction pEaseFunction)

ScaleAtModifier(final float pDuration, final float pFromScaleX,

final float pToScaleX, final float pFromScaleY, final float pToScaleY,

final float pScaleCenterX, final float pScaleCenterY,

final IEntityModifierListener pEntityModifierListener)

ScaleAtModifier(final float pDuration, final float pFromScaleX,

final float pToScaleX, final float pFromScaleY, final float pToScaleY,

final float pScaleCenterX, final float pScaleCenterY,

final IEntityModifierListener pEntityModifierListener,

final IEaseFunction pEaseFunction)

**Điều chỉnh màu sắc**

ColorModifier(final float pDuration, final float pFromRed, final float pToRed,

final float pFromGreen, final float pToGreen, final float pFromBlue,

final float pToBlue)

ColorModifier(final float pDuration, final float pFromRed, final float pToRed,

final float pFromGreen, final float pToGreen, final float pFromBlue,

final float pToBlue, final IEaseFunction pEaseFunction)

ColorModifier(final float pDuration, final float pFromRed, final float pToRed,

final float pFromGreen, final float pToGreen, final float pFromBlue,

final float pToBlue, final IEntityModifierListener

pEntityModifierListener, final IEaseFunction pEaseFunction)

ColorModifier(final float pDuration, final float pFromRed, final float pToRed,

final float pFromGreen, final float pToGreen, final float pFromBlue,

final float pToBlue, final IEntityModifierListener

pEntityModifierListener, final IEaseFunction pEaseFunction)

Các giá trị màu sắc có giá trị từ 0.0f tới 1.0f, tất cả các phương thức trên làm cho màu của Entity thay đổi theo thời gian, phương thức EaseFunction và EntityModifierListener hoạt động giống như các bộ điều chỉnh khác

**Điều chỉnh xoay Entity**

RotationModifier(final float pDuration, final float pFromRotation,

final float pToRotation)

RotationModifier(final float pDuration, final float pFromRotation,

final float pToRotation, final IEntityModifierListener

pEntityModifierListener)

RotationModifier(final float pDuration, final float pFromRotation,

final float pToRotation final IEaseFunction pEaseFunction)

RotationModifier(final float pDuration, final float pFromRotation,

final float pToRotation, final IEntityModifierListener

pEntityModifierListener, final IEaseFunction pEaseFunction)

RotationAtModifier(final float pDuration, final float pFromRotation,

final float pToRotation, final float pRotationCenterX,

final float pRotationCenterY)

RotationAtModifier(final float pDuration, final float pFromRotation,

final float pToRotation, final float pRotationCenterX,

final float pRotationCenterY, final IEntityModifierListener

pEntityModifierListener)

RotationAtModifier(final float pDuration, final float pFromRotation,

final float pToRotation, final float pRotationCenterX,

final float pRotationCenterY, final IEaseFunction pEaseFunction)

RotationAtModifier(final float pDuration, final float pFromRotation,

final float pToRotation, final float pRotationCenterX,

final float pRotationCenterY, final IEntityModifierListener

pEntityModifierListener, final IEaseFunction pEaseFunction)

RotationByModifier(final float pDuration, final float pRotation)

RotationByModifier(final RotationByModifier pRotationByModifier)

**Các bộ điều chỉnh điều khiển sự trong suốt**

Sự trong suốt của một Entity được định nghĩa thông qua giá trị Alpha

AlphaModifier(final float pDuration, final float pFromAlpha, final float

pToAlpha)

AlphaModifier(final float pDuration, final float pFromAlpha, final float

pToAlpha, final IEntityModifierListener)

AlphaModifier(final float pDuration, final float pFromAlpha, final float

pToAlpha, final IEaseFunction pEaseFunction)

AlphaModifier(final float pDuration, final float pFromAlpha, final float

pToAlpha, final IEntityModifierListener pEntityModifierListener,

final IEaseFunction pEaseFunction)

Giá trị Alpha thay đổi từ 0.0f tới 1.0f

Các bộ điều chỉnh làm cho Entity nhạt đi hoặc đậm lên

FadeInModifier(final float pDuration)

FadeInModifier(final float pDuration, final IEaseFunction pEaseFunction)

FadeInModifier(final float pDuration, final IEntityModifierListener

pEntityModifierListener)

FadeInModifier(final float pDuration, final IEntityModifierListener

pEntityModifierListener, final IEaseFunction pEaseFunction)

### 2.2.5. Vẽ đường thẳng và hình chữ nhật trong AndEngine

**Vẽ đường thẳng**

Line(final float pX1, final float pY1, final float pX2, final float pY2)

Line(final float pX1, final float pY1, final float pX2, final float pY2, final float pLineWidth)

Nếu xử dụng phương thức khởi tạo thứ nhất thì độ dày của đường thẳng là 1 pixel

**Vẽ hình chữ nhật**

Rectangle(final float pX, final float pY, final float pWidth, final float pHeight)

Rectangle(final float pX, fina float pY, final float pWidth, final float pHeight,

final RectangleVertexBuffer pRectangleVertexBuffer)

### 2.2.6. Sprite

AndEngine cung cấp 3 loại prite khác nhau

* Một Sprite xử dụng một Texture đơn, lấy ra từ một TextureRegion
* Một Tiled Sprite chon môt Texture, lấy từ một mảng Texture trong một TiledTextureRegion
* Một AnimatedSprite là lớp con của TiledSprite để hiển thị hoạt ảnh

**Texture**

AndEngine lưu trữ tập hợp các Texture trong bộ nhớ như là một trường hợp của lớp Texture, có 1 lớp Texture và 1 singleton TextureManager quản lý tất cả các Texture trong trò chơi. Mỗi Texture có thể có vài TextureRegion bên trong nó định nghĩa các ảnh bitmap của Texture, AndEngine có 2 kiểu TextureRegion cơ bản

* TextureRegion: luôn chứa đựng 1 ảnh với chiều rộng và chiều cao duy nhất
* TiledTextureRegion: luôn chứa đựng nhiều hơn 1 ảnh, mỗi ảnh có cùng chiều rộng và chiều cao, các ảnh sẽ được tổ chức thành các mảng và được truy xuất đến thông qua vị trí trong mảng

Để tạo một Texture mới có thể xử dụng các phương thức sau

Texture(final int pWidth, final int pHeight)

Texture(final int pWidth, final int pHeight,

final ITextureStateListener pTextureStateListener)

Texture(final int pWidth, final int pHeight, final TextureOptions pTextureOptions)

Texture(final int pWidth, final int pHeight, final TextureOptions pTextureOptions,

final ITextureStateListener pTextureStateListener)

Tất cả chúng sẽ tạo ra một không gian trống với chiều dài và chiều rộng là lũy thừa của 2 có tác dụng nạp các Texture vào trong Sprite của trò chơi. Nếu chiều dài và chiều rộng không là lũy thừa của 2 thì AndEngine sẽ ném một ngoại lệ là IllegalArgumentException

**TextureRegionFactory**

Để nạp các ảnh vào trong Texture chúng ta sử dụng TextureRegionFactory, TextureRegionFactory tạo TextureRegion và TiledTextureRegion từ 2 kiểu tài nguyên

* Assets: Các file ảnh được lưu tại thư mục Assets của dự án Android
* Resources: Các file ảnh được lưu tại thư mục drawable bên trong thư mục res trong dự án Android

Các phương thức căn bản

TextureRegion createFromAsset(final Texture pTexture,

final Context pContext, final String pAssetPath,

final int pTexturePositionX, final int pTexturePositionY)

TiledTextureRegion createTiledFromAsset(final Texture pTexture,

final Context pContext, final String pAssetPath,

final int pTexturePositionX, final int pTexturePositionY,

final int pTileColumns, final int pTileRows)

TextureRegion createFromResource(final Texture pTexture,

final Context pContext, final int pDrawableResourceID,

final int pTexturePositionX, final int pTexturePositionY)

TiledTextureRegion createTiledFromResource(final Texture pTexture,

final Context pContext, final int pDrawableResourceID,

final int pTexturePositionX, final int pTexturePositionY,

final int pTileColumns, final int pTileRows)

TextureRegion createFromSource(final Texture pTexture,

final ITextureSource pTextureSource, final int pTexturePositionX,

final int pTexturePositionY)

TiledTextureRegion createTiledFromSource(final Texture pTexture,

final ITextureSource pTextureSource, final int pTexturePositionX,

final int pTexturePositionY, final int pTileColumns, final int pTileRows)

Các tham số trong các phương thức trên được định nghĩa như sau

* Texture pTexture: Texture đang muốn nạp vào
* Context pContext: Ngữ cảnh Activity cho Activity hiện tại
* String pAssetPath: Tên tệp tin trong thư mục Assets
* pDrawableResourceID: Một số nguyên được gán cho tài nguyên trong tệp tin R,java, thường được xem như là R.drawable.<tên tài nguyên>
* int pTexturePositionX, int pTexturePositionY: Vị trí trên Texture mà hình ảnh sẽ được nạp
* int pTileColumns, int pTileRows: (dành cho TiledTextureRegion) số lượng cột và hang trong ảnh lát (tiled image)

TextureRegionFatory cũng chứa một phương thức cho thiết lập đường dẫn từ thư mục assets:

void setAssetBasePath(final String pAssetBasePath)

Phương thức này hữu dụng khi trong thư muc assets chứa đựng nhiều thư mục ảnh khác nhau, tham số pAssetBasePath là tên đường dẫn kết thúc bằng “/”

**Lớp Sprite**

Giả sử rằng chúng ta đã có các TextureRegion với các ảnh chúng ta muốn sử dụng cho các Sprite trong trò chơi, chúng ta có thể sử dụng các phương thức khởi tạo sau

Sprite(final float pX, final float pY, final TextureRegion pTextureRegion)

Sprite(final float pX, final float pY, final float pWidth, final float pHeight,

final TextureRegion pTextureRegion)

Sprite(final float pX, final float pY, final TextureRegion pTextureRegion,

final RectangleVertexBuffer pRectangleVertexBuffer)

Sprite(final float pX, final float pY, final float pWidth, final float pHeight,

final TextureRegion pTextureRegion,

final RectangleVertexBuffer pRectangleVertexBuffer)

Tất cả các phương thức khởi tạo trên đều cần khởi tạo vị trí của Sprite thông qua pX, pY, đồng thời cần một TextureRegion để cho AndEngine và OpenGL biết rằng Texture nào sẽ được vẽ vào Sprite. Bởi vì Sprite là lớp con của lớp Entity nên tất cả các phương thức của Entity đều có thể sử dụng đối với Sprite

Phương thức

TextureRegion getTextureRegion()

Trả về TextureRegion của Sprite

**TiledSprites**

Chúng ta có thể sử dụng TiledSprite để tạo ra Sprite kết hợp với một mảng texture lưu trữ trong một TiledTextureRegion. Một TiledTextureRegion luôn luôn có hơn 1 Texture, tất cả chúng đều cùng kích thước và được lưu trữ trong một mảng lơn

Để khởi tạo TileSprite chúng ta làm như sau

TiledSprite(final float pX, final float pY, final TiledTextureRegion

pTiledTextureRegion)

TiledSprite(final float pX, final float pY, final float pTileWidth,

final float pTileHeight, final TiledTextureRegion pTiledTextureRegion)

TiledSprite(final float pX, final float pY, final TiledTextureRegion

pTiledTextureRegion, final RectangleVertexBuffer

pRectangleVertexBuffer)

TiledSprite(final float pX, final float pY, final float pTileWidth,

final float pTileHeight, final TiledTextureRegion pTiledTextureRegion,

final RectangleVertexBuffer pRectangleVertexBuffer)

*TileSprite có thêm 4 phương thức mới*

int getCurrentTileIndex()

void setCurrentTileIndex(final int pTileIndex)

void setCurrentTileIndex(final int pTileColumn, final int pTileRow)

void nextTile()

**AnimatedSprite**

Sử dụng đối với các Sprite hoạt ảnh, AnimateSprite là một lớp con của lớp TiledSprite, việc khởi tại AnimatedSprite có thể theo các phương thức khởi tạo sau

AnimatedSprite(final float pX, final float pY, final TiledTextureRegion

pTiledTextureRegion)

AnimatedSprite(final float pX, final float pY, final float pTileWidth,

final float pTileHeight, final TiledTextureRegion pTiledTextureRegion)

AnimatedSprite(final float pX, final float pY, final TiledTextureRegion

pTiledTextureRegion, final RectangleVertexBuffer

pRectangleVertexBuffer)

AnimatedSprite(final float pX, final float pY, final float pTileWidth,

final float pTileHeight, final TiledTextureRegion pTiledTextureRegion,

final RectangleVertexBuffer pRectangleVertexBuffer)

Các tham số trong các phương thức trên có ý nghĩa như sau

* float pX and float pY: Vị trí của Sprite
* TileTextureRegion pTiledTextureRegion: TextureRegion nắm giữ các khung hoạt ảnh của Sprite
* RectangleVertexBuffer pRectangleVertexBuffer: Đây là một cấu trúc của OpenGL được sử dụng để điều chỉnh Sprite khi nó Sprite được hiển thị
* float pTileWidth and float pTileHeight: chiều rộng và chiều cao của mỗi ô trong TiledTextureRegion

**Các phương thức Animation**

AnimatedSprite thêm một số phương thức để điều khiển sự hoạt động của Sprite

AnimatedSprite animate(final long pFrameDurationEach)

AnimatedSprite animate(final long pFrameDurationEach, final boolean pLoop)

AnimatedSprite animate(final long pFrameDurationEach, final int pLoopCount)

AnimatedSprite animate(final long pFrameDurationEach, final boolean pLoop,

final IAnimationListener pAnimationListener)

AnimatedSprite animate(final long pFrameDurationEach, final int pLoopCount,

final IAnimationListener pAnimationListener)

Các tham số truyền vào có ý nghĩa như sau

* long pFrameDurationEach: Định nghĩa khoảng thời gian (theo mili giây) mà mỗi khung hình sẽ được hiển thị
* boolean pLoop: Nếu giá trị pLoop là true thì hoạt ảnh sẽ được lặp lại sau khi kết thúc, nếu pLoop là false thì hoạt ảnh sẽ diễn ra 1 lần
* int pLoopCount: Định nghĩa số lần hoạt ảnh lặp lại

### 2.2.7. Text

**Typeface**

Chúng ta sử dụng Android Typeface để truy cập đến các font Android chuẩn

Typeface.create(Typeface family, int style)

Tham số family có thể là

Typeface.DEFAULT

Typeface.DEFAULT\_BOLD

Typeface.MONOSPACE

Typeface.SANS\_SERIF

Typeface.SERIF

Tham số style có thể là

Typeface.NORMAL

Typeface.BOLD

Typeface.ITALIC

Typeface.BOLD\_ITALIC

**Font**

Trong AndEngine, lớp Font là một phần của mã OpenGL, để khởi tạo Font sử dụng phương thức

Font(final Texture pTexture, final Typeface pTypeface,

final float pSize, final boolean pAntiAlias, final int pColor)

Ý nghĩa của các tham số như sau

Texture pTexture: Texture mà ảnh font được lưu trữ

Typeface pTypeface: là Android Typeface như đã nói ở trên

float pSize: Chiều cao của font theo pixels

boolean pAntiAlias: nếu pAntiAlias là true thì font sẽ có dạng răng cưa khi hiển thị

int pColor: Màu của font hiển thị

**FontFactory**

Font createFromAsset(final Texture pTexture, final Context pContext,

final String pAssetPath, final float pSize, final boolean pAntiAlias,

final int pColor)

**Text trong AndEngine**

Các phương thức khởi tạo

Text(final float pX, final float pY, final Font pFont, final String pText)

Text(final float pX, final float pY, final Font pFont, final String pText,

final HorizontalAlign pHorizontalAlign)

Text(final float pX, final float pY, final Font pFont, final String pText,

final HorizontalAlign pHorizontalAlign, final int pCharactersMaximum)

ý nghĩa các tham số như sau

* float pX, pY: Vị trí của text trên màn hình
* String pText: Nội dung đoạn text
* HorizontalAlign pHorizontalAlign: Có 3 sự lựa chọn, mặc định là bên trái

+ HorizontalAlign.LEFT

+ HorizontalAlign.CENTER

+ HorizontalAlign.RIGH

* int pCharactersMaximum: Số kí tự tối đa trong đoạn text

### 2.2.8. Âm thanh

Trong AndEngine có 2 khái niệm về âm thanh đó là

* Âm nhạc (Music): một phần của bản nhạc có độ dài trên 5 giây
* Tiếng động (Sound): những âm thanh có độ dài dưới 5 giây

**Lớp Music (Music class)**

Các phương thức chạy Music

void play()

void stop()

void pause()

void resume()

void release()

Các phương thức điều khiển việc chơi Music

void setLooping(final boolean pLooping)

void setVolume(final float pLeftVolume, final float pRightVolume)

void seekTo(final int pMilliseconds)

boolean isPlaying()

**Lớp Sound (Sound class)**

Các phương thức chạy Sound

void play()

void stop()

void pause()

void resume()

void release()

Các phương thức điều khiển việc chạy Sound

void setLooping(final boolean pLooping)

void setVolume(final float pLeftVolume, final float pRightVolume)

**MusicFactory**

Lớp này cung cấp một nhóm các phương thức để tải Music

Music createMusicFromFile(final MusicManager pMusicManager,

final Context pContext, final File pFile)

Music createMusicFromAsset(final MusicManager pMusicManager,

final Context pContext, final String pAssetPath)

Music createMusicFromResource(final MusicManager pMusicManager, final

Context

pContext, final int pMusicResID)

**SoundFactory**

Lớp này cung cấp một nhóm các phương thức để tải Sound

Sound createSoundFromPath(final SoundManager pSoundManager,

final Context pContext, final String pPath)

Sound createSoundFromAsset(final SoundManager pSoundManager,

final Context pContext, final String pAssetPath)

Sound createSoundFromResource(final SoundManager pSoundManager,

final Context pContext, final int pSoundResID)

Sound createSoundFromFileDescriptor(final SoundManager pSoundManager,

final FileDescriptor pFileDescriptor, final long pOffset,

final long pLength)

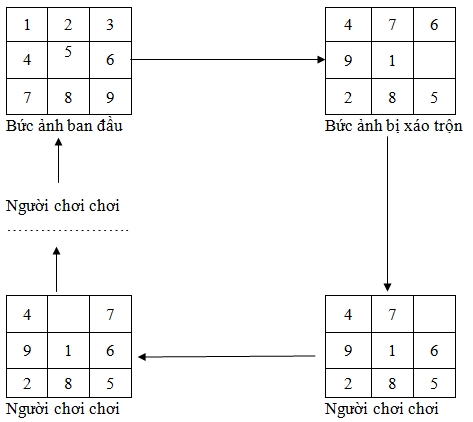
# 

# CHƯƠNG 3

# XÂY DỰNG TRÒ CHƠI XẾP TRANH SỬ DỤNG ANDENGINE

## 3.1. Mô tả trò chơi

Có 1 bức ảnh được chia thành NxN phần bằng nhau, các phần được xáo trộn để phá vỡ bức ảnh thành lộn xộn, trong các phần lộn xộn đó, có 1 phần sẽ bị che khuất, nhiệm vụ của người chơi là sắp xếp các phần lộn xộn đó thành bức tranh ban đầu dựa vào việc di chuyển ngang hoặc dọc các phần lộn xộn theo phần bị che khuất.



*Hình 3.1.1: Mô tả trò chơi*

- Có 8 màn chơi cho phép người chơi lựa chọn tùy ý

+ 2 màn chơi ảnh bị cắt thành 2x2

+ 2 màn chơi ảnh bị cắt thành 3x3

+ 2 màn chơi ảnh bị cắt thành 4x4

+ 2 màn chơi ảnh bị cắt thành 5x5

- Với mỗi màn chơi, người chơi có thể đạt được một trong 3 huy chương: vàng, bạc hoặc đồng tùy vào thời gian hoặc số lần xếp hình

- Người chơi có thể tùy chọn âm thanh bật hoặc tắt

- Với mỗi màn chơi sau sau khi người chơi hoàn thành sẽ thông báo kết quả chi tiết về huy chương, thời gian, số lần xếp mà người chơi đạt được

## 3.2. Mô tả giao diện trò chơi

Giao diện 1: Bắt đầu vào trò chơi

* Hiển thị hình ảnh nền
* Hiển thị hướng dẫn (chạm vào màn hình để bắt đầu)
* Hiển thị button Link, khi người chơi chọn Link (chạm vào button Link) thì sẽ lien kết tới website giới thiệu về trò chơi và các thống kê về trò chơi (Do chưa xây dựng website nên tạm thời nếu người chơi chọn Link thì sẽ liên kết tới website google.com.vn)

Giao diện 2: Sau khi người chơi chạm vào màn hình (sau Giao diện 1)

* Hiển thị hình ảnh nền
* Hiển thị button “Bắt đầu” để nguời chơi vào lựa chọn các màn chơi
* Người chơi có thể lựa chọn bật hoặc tắt âm thanh bằng cách chọn button “Âm thanh”
* Hiển thị button Link như trong Giao diện 1

Giao diện 3: Người chơi lựa chọn “Bắt đầu”

* Hiển thị danh sách các màn chơi
* Hiển thị thông tin chi tiết về các màn chơi bao gồm: NxN phần mà ảnh sẽ bị cắt, thời gian cần đạt được để đạt mỗi loại huy chương, số lần xếp cần đạt được để đạt mỗi loại huy chương.
* Nếu người chơi đã hoàn thành 1 màn chơi trước đó thì hiển thị huy chương, thời gian, số lần xếp hình mà người chơi đã đạt được
* Hiển thị button “Quay lại” cho phép người chơi trở về màn hình chính (Giao diện 2)

Giao diện 4: Khi người chơi lựa chọn màn chơi

* Hiển thị hướng dẫn
* Hiển thị button tải lại trò chơi cho phép người chơi tải lại trò chơi
* Hiển thị lần xếp
* Hiển thị thời gian
* Hiển thị button thoát
* Hiển thị hình nền và hình sẽ bị cắt

Giao diện 5: Giao diện chính của trò chơi

* Hiển thị hình ảnh bị xáo trộn
* Hiển thị số lần xếp
* Hiển thị thời gian
* Hiển thị hình nền
* Hiển thị button “tải lại”
* Hiển thị button “thoát”

Giao diện 6: Người chơi hoàn thành màn chơi

* Hiển thị hộp thoại thông báo kết quả gồm huy chương, thời gian, lượt xếp mà người chơi đã đạt được
* Hiển thị button “Đóng” để người chơi đóng hộp thoại

## 3.3. Công cụ phát triển trò chơi

- Máy tính tốc độ 2.1GHz, Ram 2GB

- Hệ điều hành windows 7 - 32bit

- Java Development Kit (JDK) 7

- Android Development Tool (ADT) Bundle

- AndEngine GLES1

- Common-primitives-1.0

- Eval-0.5

## 3.4. Lập trình trò chơi

### 3.4.1. Các layout xml

Là các layout được định nghĩa bằng cấu trúc xml, nằm bên trong thư mục layout trong thư mục res của dự án Android

* Layout stagerow.xml: hiển thị thông tin chi tiết về từng màn chơi trong giao diện 3
* Layout title.xml: hiển thị giao diện 3
* Layout resultdlg.xml: dùng trong việc định nghĩa hộp thoại hiển thị kết quả người chơi trong giao diện 6
* Layout selector.xml: một layout tương tự như layout title.xml, hiển thị giao diện 3 nhưng các kết quả của người chơi đã đạt được trong từng màn chơi đã được tải và thêm vào layout stagerow.xml

### 3.4.2. Dữ liệu của các màn chơi

Trong lập trình Android, để lưu dữ liệu, có nhiều cách lưu khác nhau. Trong trò chơi này, dữ liệu chi tiết về các màn chơi được lưu bằng định dạng xml, thông tin được lưu trong file stage.xml nằm trong thư mục raw bên trong thư mục res. Ví dụ, thông tin về màn chơi thứ nhất sẽ có dạng như sau:

<stage order="1"> *//Thứ tự mà chơi*

<lock value="0" /> *//Đặt trạng thái khóa hoặc mở khóa màn chơi*

<name value="Màn 1" /> *//Tên màn chơi*

<innername value="e0" /> *//Sử dụng để tải ảnh e0… cho màn chơi*

<stageimage value="e0" /> *//Tải ảnh e0… cho màn chơi*

<size value="2" /> *//Bức ảnh được cắt thành 2x2*

<background value="" /> *//Nền của trò chơi*

<qualifyTime value="5,10,-1" /> *//Thời gian để đạt các huy chương*

<qualifyStep value="10,20,-1" /> *//Lần xếp để đạt các huy chương*

<title value="Màn 1" /> *//Tiêu đề màn chơi*

<description></description> *//Mô tả màn chơi*

</stage>

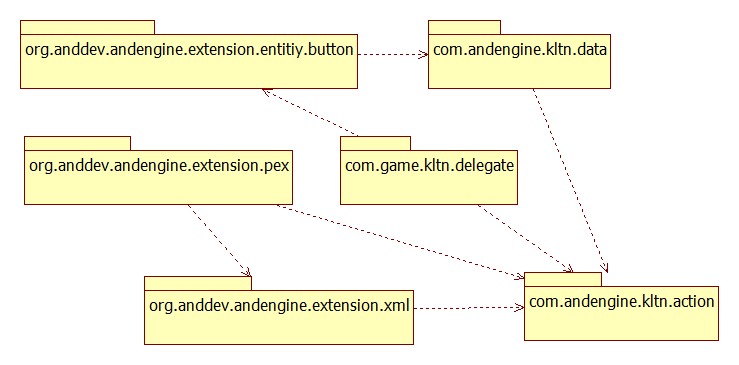
### 3.4.3. Dữ liệu hình ảnh và âm thanh

Hình ảnh được lưu tại thư mục gfx bên trong thư mục assets và trong thư mục drawable bên trong thư mục res. Trong trò chơi có sử dụng 3 loại ảnh khác nhau đó là ảnh jpg, ảnh png và ảnh vector svg

Âm thanh trò chơi được lưu tại thư mục mfx bên trong thư mục assets, có thể sử dùng nhiều loại tệp tin âm thanh khác nhau, trò chơi này sử dụng âm thanh .mp3

### 3.4.4. Sơ đồ các package và các class trong trò chơi

Chương trình gồm 6 package



*Hình 3.4.4.1: Sơ đồ package*

Package com.andengine.kltn.data

**Lớp StageData**: lớp này gồm các phương thức getter và setter các thông tin chi tiết cho từng màn chơi. Các thông tin của màn chơi gồm có:

+ Tên màn chơi

+ Ảnh icon của màn chơi

+ Ảnh sẽ bị cắt trong màn chơi

+ Ảnh nền của màn chơi

+ Kích thước màn chơi

+ Thời gian hoàn thành màn chơi để đạt được các loại huy chương vàng, bạc, đồng

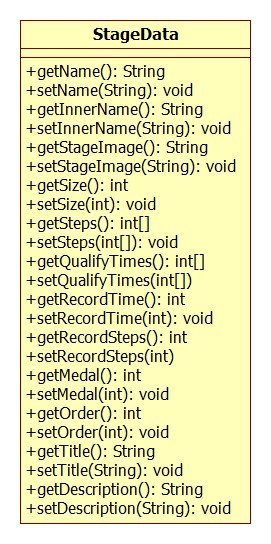
+ Số lần xếp màn chơi để đạt được huy chương vàng, bạc, đồng

+ Huy chương mà người chơi đã đạt được

+ Thứ tự của màn chơi

+ Tiêu đề màn chơi

+ Mô tả về màn chơi

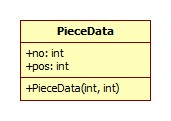


*Hình 3.4.4.2: Lớp StageData*

**Lớp PieceData:** mỗi bức ảnh bị cắt thành NxN phần bằng nhau, mỗi phần nhỏ bị cắt ra được gọi là 1 piece, lớp này định nghĩa các đối tượng piece. Mỗi đối tượng piece gồm 2 thành phần

+ Thành phần no: Lưu vị trí của piece trong ảnh ban đầu khi chưa bị cắt

+ Thàn phần pos: Lưu vị trí của piece khi ảnh bị cắt



*Hình 3.4.4.3: Lớp PieceData*

Package com.game.kltn.delegate

**Lớp Delegate:** giúp cho việc cài đặt âm thanh bật hoặc tắt chạy trong suốt quá trình chơi trò chơi

public class Delegate {

public interface Simple {

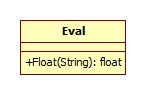
public void invoke();

}

}

Package org.anddev.andengine.extension.pex

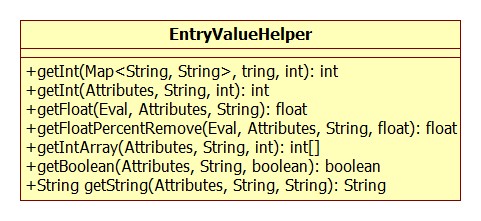
**Lớp Eval:** Chuyển đổi dữ liệu từ kiểu chuỗi sang kiểu số thực. Các thông tin về từng màn chơi được lưu trong file stage.xml được lưu dưới dạng chuỗi, nhưng kích thước màn chơi sẽ phải được chuyển sang kiểu số thực để tính toán.



*Hình 3.4.4.4: Lớp Eval*

Package org.addev.andengine.extension.xml

**Lớp EntryValueHelper:** chứa các phương thức get dữ liệu hỗ trợ việc get dữ liệu từ tệp tin stage.xml



*Hình 3.4.4.5: Lớp EntryValueHelper*

Package org.anddev.andengine.extension.entity.button

**Lớp MyButton:** là lớp thừa kế từ lớp Rectangle của AndEngine, định nghĩa các button xuất hiện trong trò chơi bao gồm các phương thức sau

+ Khởi tạo Button là một Entity trong AndEngine, giống như một hình chữ nhật gồm có 2 tọa độ đầu và 2 toạn độ cuối, một giá trị Boolean cho phép đặt button ở trạng thái lắng nghe sự kiện click hoặc chạm vào màn hình hay không

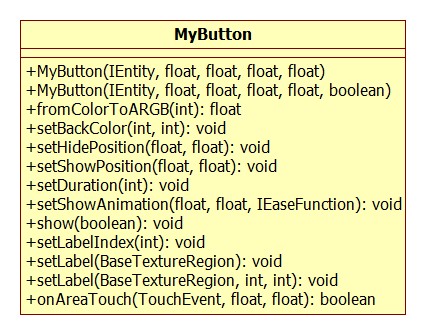
+ Tạo màu cho button

+ Tạo vị trí nhìn thấy (hoặc không nhìn thấy) cho button

+ Đặt trạng thái nhìn thấy (hoặc không nhìn thấy) cho button

+ Đặt nhãn cho button

+ Vị trí mà người chơi có thể chạm vào button



*Hình 3.4.4.6: Lớp MyButton*

**Lớp ButtonData**: chứa các dữ liệu của 1 button

public class ButtonData {

public ButtonData(){}

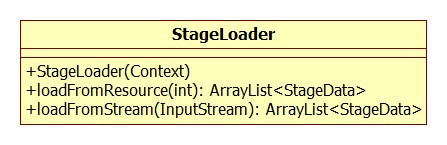
public Boolean pressing=false;

public Object userarea;

}

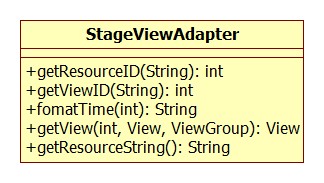
Package com.andengine.kltn.action

**Lớp StageLoader:** tải các dữ liệu từ file stage.xml



*Hình 3.4.4.7: Lớp StageLoader*

**Lớp StageViewAdapter:** set các dữ liệu tải được từ file stage.xml vào các vị trí các view trong từng màn chơi



*Hình 3.4.4.8: Lớp StageViewAdapter*

**Lớp Title:** là lớp thừa kế từ lớp BaseGameActivity của AndEngine, khởi chạy khi người chơi bật trò chơi, cho phép người chơi cài đặt âm thanh, lựa chọn các màn chơi. Do là lớp kế thừa từ lớp BaseGameActivity của AndEngine, nên các phương thức căn bản cần phải có là

+ onLoadEngine(): Khởi động Engine với việc khởi tạo các Camera

+ onLoadResource(): tải các tài nguyên hình ảnh, âm thanh, định nghĩa các Texture và TextureRegion cần sử dụng đến

+ onLoadScene(): Khởi tạo và gắn các đối tượng trong trò chơi vào Scene trong mỗi Layer

+ onLoadComplete(): Chạy khi Engine kết thúc

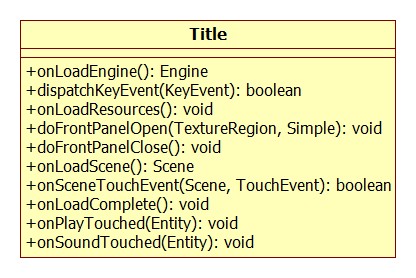
Ngoài ra lớp Title gồm các phương thức sau:

+ dispatchKeyEvent(KeyEvent): sử lý sự kiện bàn phím

+ onScenTouchEvent(Scene, TouchEvent): Sử lý sự kiện chạm vào Scene

+ onPlayTouched(Entity): sử lý sự kiện khi chạm vào button “Bắt đầu” để bắt đầu vào trò chơi

+ onSoundToched(Entity): sử lý sự kiện chạm vào button “Âm thanh” để cài đặt trạng thái âm thanh (bật/tắt)



*Hình 3.4.4.9: Lớp Title*

**Lớp GameAction:** là lớp con của lớp BaseGameActivity của AndEngine, được gọi tới sau khi người chơi chọn các màn chơi

Ngoài các phương thức căn bản của một BaseGameActivity trong AndEngine, lớp này còn chứa các phương thức sau

+ countSteps(): đếm số lần xếp mà người chơi đã xếp

+ congrats(): thực hiện các công việc sau khi người chơi hoàn thành màn chơi như thông báo kết quả chi tiết (thời gian, huy chương, lần xếp) mà người chơi đạt được

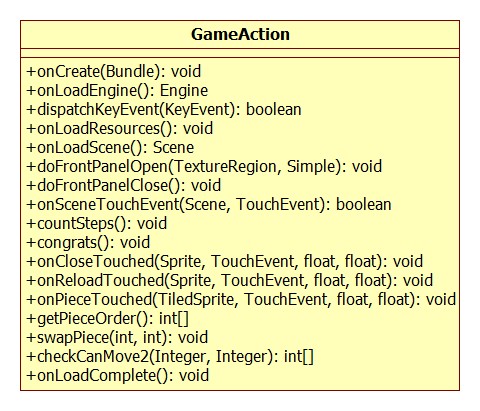
+ onCloseTouched(Sprite, TouchEvent,float, float): bắt sự kiện chạm vào button “đóng” để thoát trò chơi

+ onReloadTouched(Sprite, TouchEvent, float, float): bắt sự kiện chạm vào button “Reload” để tải lại trò chơi

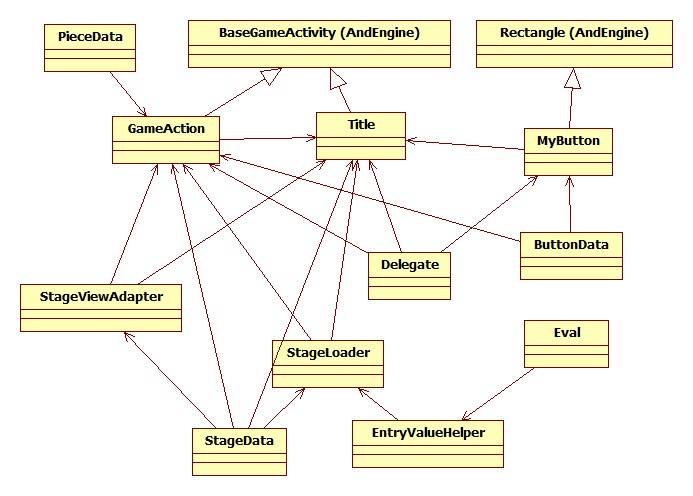
+ onPieceTouch(TiledSprite, TouchEvent, float, float): bắt sự kiện chạm vào mỗi piece

+ getPieceOrder(): Lấy thứ tự của các piece

+swapPiece(): thực hiện đổi chỗ các piece



*Hình 3.4.4.10: Lớp GameAction*



*Hình 3.4.4.11: Sơ đồ lớp*

## 3.6. Kết quả và định hướng phát triển trò chơi trong tương lai

### 3.6.1. Kết quả lập trình trò chơi

Trò chơi chạy tốt trên nên Android 2.2 trở lên. Sau đây là các giao diện màn hình chạy trên Android 2.3

|  |  |
| --- | --- |
| o1.jpg  Giao diện 1 | 02.jpg  Giao diện 2 |

|  |  |
| --- | --- |
| gd3.jpg  Giao diện 3 | gd4.jpg  Giao diện 4 |

|  |  |
| --- | --- |
| gd5.jpg  Giao diện 5 | gd6.jpg  Giao diện 6 |

*Hình 3.6.1: Giao diện các màn chơi*

### 3.6.2. Định hướng phát triển trò chơi trong tương lai

Có thể thêm nhiều chức năng vào trò chơi như thêm quảng cáo, gửi điểm lên facebook, twitter, thách đấu, mời bạn bè cùng chơi, cho phép người chơi chọn mà chơi từ chính các ảnh trong thẻ nhớ điện thoại hoặc được chụp từ camera, cho phép người chơi tự động cắt ảnh theo ý mình, xây dựng website chuyên thống kê các số lượng về trò chơi và trợ giúp trò chơi…

# 

# KẾT LUẬN

Sau khi thực hiện Khóa luận tốt nghiệp này, em đã tiếp thu được những kiến thức nền tảng của của một Game Engine nói chung và của AndEngine nói riêng, có khả năng tự phát triển các trò chơi đơn giản dành cho các thiết bị di động Android, tạo tiền đề nghiên cứu sử dụng những Game Engine lớn, tham gia vào các đội phát triển các trò chơi có quy mô lớn. Ngoài ra em còn phát triển thêm một số kiến thức và kỹ năng như

* Lý thuyết trò chơi
* Quy trình xây dựng trò chơi
* Phát triển kịch bản trò chơi
* Kỹ năng tìm kiếm thông tin và nghiên cứu tài liệu
* ...

Một lần nữa em xin chân thành cảm ơn các Thầy cô giáo, đặc biệt là giảng viên TS. Trương Anh Hoàng đã tận tình hướng dẫn em trong quá trình thực hiện khóa luận tốt nghiệp này.

# 

# TÀI LIỆU THAM KHẢO

**Tiếng Việt**

[1] AVG Team, Lập trình Game Android sử dụng AndEngine, công ty cổ phần

truyền thông VTC, 2011.

[2] Y Lam, “Người dùng làm gì smartphone và tablet?”, báo vietnamnet,

27/03/2013

**Tiếng Anh**

[3] Jayme Schroeder, Brian Broyles, AndEngine for Android Game Development

Cookbook, [Packt Publishing](http://it-ebooks.info/publisher/14/), America, 2013.

[4] Jeff Friesen, Learn Java for Android Development, Apress, America, 2010.

[5] Mario Zechner, Beginning Android Games, Apress, America, 2013

[6] Mark L. Murphy, Android Programming Tutorials, CommonsWare, America,

2011.

[7] Mark L. Murphy, Beginning Android 2, Apress, America, 2010.

[8] Rick Rogers, Learning Android Game Programming, Addison-Wesley,

America, 2012.

[9] Sayed Y. Hashimi, Satya Komatineni, and Dave MacLean, Pro Android 2,

Apress, 2010.

[10] Vladimir Silva, Pro Android Games, Apress, America, 2009.