|  |
| --- |
| TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN  KHOA CÔNG NGHỆ THÔNG TIN  LỚP CỬ NHÂN TÀI NĂNG  NGUYỄN HOÀI NAM – NGUYỄN THANH TẤN  GAME ĐỐI KHÁNG 3D DỰA TRÊN  ADRENO SDK    KHÓA LUẬN TỐT NGHIỆP CỬ NHÂN CNTT    TP.HCM, 2015 |

|  |
| --- |
| TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN  KHOA CÔNG NGHỆ THÔNG TIN  LỚP CỬ NHÂN TÀI NĂNG  NGUYỄN HOÀI NAM 1112192  NGUYỄN THANH TẤN 1112279  GAME ĐỐI KHÁNG 3D DỰA TRÊN  ADRENO SDK  **KHÓA LUẬN TỐT NGHIỆP CỬ NHÂN CNTT**    GIÁO VIÊN HƯỚNG DẪN  ThS. NGUYỄN HUY KHÁNH  NIÊN KHÓA 2011-2015 |

|  |
| --- |
| **NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN**  ……………………………………………………………………………………………………  ……………………………………………………………………………………………………  …………………………………………………………………………………………………………………………………………………………………………………………………………  …………………………………………………………………………………………………………………………………………………………………………………………………………  …………………………………………………………………………………………………………………………………………………………………………………………………………  …………………………………………………………………………………………………………………………………………………………………………………………………………  …………………………………………………………………………………………………………………………………………………………………………………………………………  …………………………………………………………………………………………………………………………………………………………………………………………………………  …………………………………………………………………………………………………………………………………………………………………………………………………………  TpHCM, ngày ….. tháng …… năm 2015  Giáo viên hướng dẫn  [Ký và ghi rõ họ tên] |

|  |
| --- |
| **NHẬN XÉT CỦA GIÁO VIÊN PHẢN BIỆN**  ……………………………………………………………………………………………………  ……………………………………………………………………………………………………  …………………………………………………………………………………………………………………………………………………………………………………………………………  …………………………………………………………………………………………………………………………………………………………………………………………………………  …………………………………………………………………………………………………………………………………………………………………………………………………………  …………………………………………………………………………………………………………………………………………………………………………………………………………  …………………………………………………………………………………………………………………………………………………………………………………………………………  …………………………………………………………………………………………………………………………………………………………………………………………………………  ……………………………………………………………………………………………………  TpHCM, ngày ….. tháng …… năm 2015  Giáo viên phản biện  [Ký và ghi rõ họ tên] |

**LỜI CẢM ƠN**

Chúng em xin chân thành cảm ơn các thầy cô trong Khoa Công Nghệ Thông Tin, trường Đại Học Khoa Học Tự Nhiên, Tp.HCM đã tạo điều kiện tốt cho chúng em thực hiện đề tài tốt nghiệp này.

Chúng em xin chân thành cảm ơn thầy Nguyễn Huy Khánh người đã tận tình hướng dẫn, chỉ bảo chúng em trong suốt thời gian thực hiện đề tài.

Chúng em cũng xin gửi lời cảm ơn sâu sắc đến quý Thầy Cô đã tận tình giảng dạy, trang bị cho chúng em những kiến thức quý báu trong suốt những năm học vừa qua.

Chúng em xin gửi lòng biết ơn sâu sắc đến ba, mẹ, các anh chị và bạn bè đã ủng hộ, giúp đỡ và động viên chúng em trong những lúc khó khăn cũng như trong suốt thời gian học tập và nghiên cứu.

Mặc dù chúng em đã cố gắng hoàn thành luận văn trong phạm vi và khả năng cho phép, nhưng chắc chắn sẽ không tránh khỏi những thiếu sót, kính mong sự cảm thông và tận tình chỉ bảo của quý Thầy Cô và các bạn.

Nhóm thực hiện

Nguyễn Hoài Nam – Nguyễn Thanh Tấn

MỤC LỤC

[Chương 1. Mở đầu 4](#_Toc423031948)

[1.1. Giới thiệu chung 4](#_Toc423031949)

[1.1.1. Game 3D trên thiết bị di động 4](#_Toc423031950)

[1.1.2. Khảo sát một số Engine làm game trên thiết bị di động 6](#_Toc423031951)

[1.2. Mục tiêu đề tài 7](#_Toc423031952)

[1.3. Nội dung luận văn 8](#_Toc423031953)

[Chương 2. Tổng quan về OpenGL ES 10](#_Toc423031954)

[2.1. OpenGL ES là gì? 10](#_Toc423031955)

[2.2. Sơ lược lịch sử hình thành và phát triển của OpenGL ES 10](#_Toc423031956)

[2.3. Tính năng của OpenGL ES 10](#_Toc423031957)

[2.4. Các khái niệm cơ bản trong OpenGL ES 10](#_Toc423031958)

[2.4.1. Texture 10](#_Toc423031959)

[2.4.2. Shader 10](#_Toc423031960)

[2.4.3. Mesh 10](#_Toc423031961)

[2.5. Cơ chế hoạt động của OpenGL ES 10](#_Toc423031962)

[2.6. Kết luận 10](#_Toc423031963)

[Chương 3. Tổng quan về Adreno SDK Framework 10](#_Toc423031964)

[3.1. Adreno SDK là gì? 10](#_Toc423031965)

[3.2. Tính năng của Adreno SDK 10](#_Toc423031966)

[3.3. Tại sao chọn Adreno SDK 10](#_Toc423031967)

[3.4. Kết luận 10](#_Toc423031968)

[Chương 4. Một số vấn đề và giải pháp khi xây dựng game 3D với Adreno SDK 10](#_Toc423031969)

[4.1. Load mô hình 3D 10](#_Toc423031970)

[4.2. Animation cho mô hình 3D 10](#_Toc423031971)

[4.3. Tạo địa hình trong game 11](#_Toc423031972)

[4.4. Chiếu sáng cảnh vật 11](#_Toc423031973)

[4.5. Xử lí di chuyển trong bản đồ 11](#_Toc423031974)

[4.6. Tạo hiệu ứng Billboard 11](#_Toc423031975)

[4.7. Xây dựng GUI 11](#_Toc423031976)

[4.8. Âm thanh trong game 11](#_Toc423031977)

[4.9. Game Service 11](#_Toc423031978)

[Chương 5. Ứng dụng game 3D phát triển trên Adreno 11](#_Toc423031979)

[5.1. Giới thiệu game 11](#_Toc423031980)

[5.2. Các quy luật chơi chính 11](#_Toc423031981)

[5.2.1. Di chuyển 11](#_Toc423031982)

[5.2.2. Tấn công 11](#_Toc423031983)

[5.2.3. Cứu chữa 11](#_Toc423031984)

[5.2.4. Nâng cấp kĩ năng 11](#_Toc423031985)

[5.2.5. Tài nguyên 11](#_Toc423031986)

[5.3. Các khái niệm trong game 11](#_Toc423031987)

[5.3.1. Bản đồ thu nhỏ 11](#_Toc423031988)

[5.3.2. Người chơi 11](#_Toc423031989)

[5.3.3. Công trình 11](#_Toc423031990)

[5.3.4. Quân lính 11](#_Toc423031991)

[5.3.5. Kĩ năng 12](#_Toc423031992)

[5.3.6. Vật phẩm 12](#_Toc423031993)

[5.4. Kiến trúc trong game 12](#_Toc423031994)

[5.4.1. Kiến trúc tổng thể 12](#_Toc423031995)

[5.4.2. Kiến trúc xử lí, phát sinh đối tượng 12](#_Toc423031996)

[5.4.3. Kiến trúc quản lí chuyển động của mô hình 12](#_Toc423031997)

[5.4.4. Kiến trúc quản lí sự kiện thay đổi màn hình 12](#_Toc423031998)

[5.4.5. Kiến trúc quản lí AI 12](#_Toc423031999)

[5.5. Kết quả thực nghiệm 12](#_Toc423032000)

[Chương 6. Kết luận và hướng phát triển 12](#_Toc423032001)

[6.1. Các kết quả đạt được 12](#_Toc423032002)

[6.2. Hướng phát triển 12](#_Toc423032003)

TÓM TẮT KHÓA LUẬN

Ngày nay, với sự phát triển mạnh mẽ của công nghệ hiện đại, điện thoại di động đã trở thành phương tiện không thể thiếu trong cuộc sống chúng ta. Điện thoại không còn đơn thuần là dành cho những cuộc gọi hay nhắn tin nữa, nhu cầu sử dụng điện thoại bây giờ rất đa dạng và phong phú, trong đó không thể bỏ qua nhu cầu giải trí. Game trên trên điện thoại di động đã trở thành thú vui số 1 của giới trẻ.

Tuy game trên điện thoại di động chơi không sướng như trên máy tính nhưng tính tiện lợi thì rất rõ, có thể chơi mọi lúc mọi nơi. Trước kia các ứng dụng game trên di động hầu hết là đồ họa 2D, nhưng với sự bùng nổ công nghệ hiện nay dẫn đến các thiết bị di động có đủ sức mạnh để chạy được các ứng dụng đòi hỏi cấu hình cao, các ứng dụng đồ họa phức tạp trong không gian 3 chiều.

Chính vì vậy mà nhu cầu chơi game 3D trên di động trở nên tăng cao, các hãng sản xuất liên tục cho ra đời những thể loại game 3D hình ảnh đẹp mắt, âm thanh sống động chạy trên các thiết bị di động. Song song đó, các game Engine 3D cũng không ngừng phát triển và hỗ trợ làm game cho các thiết bị di động với nhiều hệ điều hành khác nhau như Android, iOS.

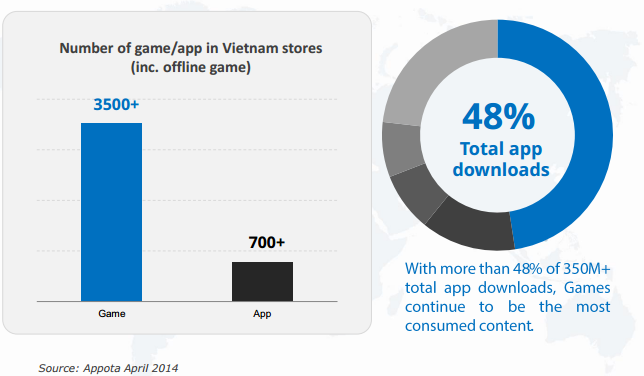
Nội dung luận văn chúng em thực hiện bao gồm việc tìm hiểu Framework Adreno SDK và sử dụng Adreno SDK để xây dựng game đối kháng 3D cho điện thoại di động.

# Chương 1. Mở đầu

## Giới thiệu chung

### Game 3D trên thiết bị di động

Game di động (game mobile) hiện đang là một thị trường rộng lớn và đầy tiềm năng trong những năm gần đây. Với xu hướng ngày một phát triển của thiết bị cầm tay cũng như cuộc cách mạng về công nghệ đã giúp game mobile ngày càng có chất lượng hơn, hình ảnh cũng như âm thanh sắc nét hơn, chân thật hơn và được nhiều người chơi đón nhận. Thêm vào đó tốc độ gia tăng chóng mặt của thị trường thiết bị di động cũng ảnh hưởng trực tiếp đến sự phát triển của game mobile, nhà nhà người người sử dụng smartphone trong công việc và để giải trí. Theo thống kê của công ty Appota, game là mảng được quan tâm bậc nhất trong số các ứng dụng trên mobile, số lượng các ứng dụng game nhiều gấp 5 lần các ứng dụng khác. Điều này tương đối dễ hiểu khi con người luôn có xu hướng giải trí và giảm stress sau những giờ học tập, lao động mệt mỏi.



Hình 1.1 Số lượng ứng dụng game đang vượt trội trên kho tải Việt Nam

Cũng theo một tờ báo mới đây, Việt Nam là quốc gia đứng thứ 17 trên thế giới và đứng thứ 4 tại châu Á về mức độ sử dụng mobile internet. Đây là những con số miêu tả rõ nhất tiềm năng và độ lớn của thị trường game mobile.

Không thể phủ nhận rằng, game mobile đang ngày càng đa dạng và phong phú. Đặc biệt hơn nhiều tựa game lớn đang dần được chuyển sang game mobile, mới có cũ có. Chúng ta có thể nhắc đến một số cái tên như GTA SanAndreas, Need for Speed: Most Wanted hay Bio Shock, những tựa game kinh điển này từng làm mưa làm gió trên các cỗ máy PC ngày xưa giờ đã được các nhà phát hành chuyển sang mobile. Điều này đã cho thấy sự quan tâm và dần chuyển hướng của họ sang thị trường di động đầy tiềm năng.

Các sản phẩm trên internet thường có tính toàn cầu hóa rất cao và game mobile cũng không phải là ngoại lệ. Một số sản phẩm game Việt đã có mặt trên thị trường quốc tế có thể kể đến như: Flappy Bird, Freaking Math, Ninja Revenge, School Cheater…

Một vài game 3D tiêu biểu trên thiết bị di động:

|  |  |
| --- | --- |
| 1. Game Asphalt 8   *(Nguồn:* [www.genk.vn](http://genk.vn/apps-games/top-7-game-3d-hap-dan-nhat-tren-android-20131008155203708.chn)*)* | (b) Game Riptide GP2  *(Nguồn:* [www.genk.vn](http://genk.vn/apps-games/top-7-game-3d-hap-dan-nhat-tren-android-20131008155203708.chn)*)* |
| (c) Game Dead Trigger 2  (Nguồn: [www.mobay.vn](http://mobay.vn/game-mobile-hay/5-game-android-so-huu-do-hoa-tuyet-dinh-705986.html)) | (d) Game Assassin’s Creed Pirates  (Nguồn: [www.mobay.vn](http://mobay.vn/game-mobile-hay/5-game-android-so-huu-do-hoa-tuyet-dinh-705986.html)) |

Hình . Một số hình ảnh về game 3D trên Android

### Khảo sát một số Engine làm game trên thiết bị di động

* **LibGDX:** là một thư viện mã nguồn mở, đa nền tảng và được thiết kế để tạo những game sử dụng ngôn ngữ Java. Bên cạnh đó, LibGDX cũng sử dụng ngôn ngữ C/C++ cho những tác vụ quan trọng thông qua NDK nên cho tốc độ xử lí nhanh. Một ưu điểm của engine này là cho phép chúng ta viết game và test hoàn toàn ngay trên nên desktop. Tuy nhiên, LibGDX được phát triển lúc đầu chủ yếu dành cho nền 2D nên cho đến thời điểm hiện tại vẫn chưa hỗ trợ nhiều trên nền 3D.
* **Unity:** là một trong những engine thương mại khá phổ biến hiện nay, có khả năng phát triển game đa nền tảng. Unity tạo ra được nhiều loại game 3D đa dạng, hỗ trợ nhiều loại định dạng mô hình khác nhau, cũng như hỗ trợ tạo mô hình trực tiếp. Lượng tài liệu hướng dẫn nhiều, cộng đồng lớn với diễn đàn riêng. Unity có 2 phiên bản là Unity Pro có tính phí và Unity Free để người dùng dễ dàng lựa chọn
* **Cocos2D-x:** là một engine hỗ trợ lập trình game 2D đa nền tảng, hơn nữa còn được cung cấp dưới dạng mã nguồn mở tạo sự linh động cho người sử dụng. Cocos2D-x hỗ trợ chủ yếu 3 ngôn ngữ: C++, Lua, Javascript. Với cộng đồng người dùng lớn, lượng tài liệu và ví dụ mẫu khá nhiều, Cocos2D-x hiện đang được nhiều lập trình viên sử dụng.
* **Unreal Engine:** là game engine được phát triển bởi Epic Games, ban đầu được dùng chủ yếu để phát triển game bắn súng góc nhìn thứ nhất, tuy nhiên hiện nay nó đã được sử dụng thành công trong nhiều thể loại game khác. Được viết bằng C++, Unreal Engine có tính linh động cao, hỗ trợ chạy đa nền tảng và có chất lượng đồ họa tốt nhất hiện nay. Hiện tại engine được cung cấp miễn phí và được rất nhiều các nhà phát triển game sử dụng.
* **Adreno SDK:** là một framework hoàn toàn miễn phí hỗ trợ làm game 3D đa nền tảng, sử dụng ngôn ngữ C++. Adreno SDK cho tốc độ render nhanh và nhiều đối tượng tại một thời điểm, cho chất lượng đồ họa tốt nhất có thể. Một trong những điểm nổi bật của Adreno SDK là khả năng chạy và gỡ lỗi (debug) code ngay trên desktop như một ứng dụng thực sự, điều này làm giảm đáng kể thời gian code cũng như tìm và sửa lỗi một cách nhanh chóng. Một điểm quan trọng nữa, Adreno SDK là một framework mã nguồn mở chứ không phải là một công cụ để tạo game. Điều này có vẻ như là một nhược điểm lúc bắt đầu nhưng thực sự nó lại là một lợi thế cho phép chúng ta tự do định nghĩa, linh động thay đổi theo cách của riêng mình cho từng dự án.

Với một game đối kháng 3D tương đối lớn và phải đảm bảo chạy tốt, ổn định trên thiết bị di động, chúng em cần tìm được một game engine 3D đủ tốt để có thể xây dựng game. Tiêu chí lựa chọn của chúng em là engine đó tốc độ nhanh, xử lí được nhiều mô hình cùng lúc trên màn hình, chất lượng đồ họa tốt nhất. Ngoài ra, engine cần hỗ trợ game đa nền tảng và thay đổi linh động được trong quá trình phát triển. Chính vì những tiêu chí này, chúng em quyết định chọn engine Adreno SDK cho luận văn của mình.

## Mục tiêu đề tài

Đề tài này thuộc hướng tìm hiểu công nghệ và xây dựng ứng dụng. Mục tiêu của đề tài là tìm hiểu về thư viện đồ họa OpenGL ES và Adreno SDK Framework (framework hỗ trợ làm game với thư viện đồ họa OpenGL ES), từ đó sử dụng Adreno SDK xây dựng game đối kháng 3D thời gian thực (real-time) chạy đa nền tảng, hiện tại nhóm đã thử nghiệm thành công trên môi trường Android và Windows.

Đề tài bao gồm các phần chính sau:

* Giới thiệu tổng quan về game 3D trên thiết bị di động.
* Tìm hiểu tổng quan về thư viện đồ họa OpenGL ES.
* Tìm hiểu tổng quan về kiến trúc của Adreno SDK Framework, cách tạo các ứng dụng và một số ứng dụng mẫu trong Adreno SDK.
* Tìm hiểu các vấn đề phát sinh khi làm game 3D với Adreno SDK: load mô hình 3D, tạo địa hình, xử lí di chuyển trên bản đồ, các hiệu ứng trong game…, tìm hiểu các thư viện hỗ trợ xử lí âm thanh và một số game service. Ứng với mỗi vấn đề cần đưa ra hướng giải quyết và phương pháp thực hiện.
* Xậy dựng và phát triển ứng dụng game đối kháng 3D thời gian thực bằng Adreno SDK.

## Nội dung luận văn

Nội dung luận văn được trình bày trong 6 chương:

**Chương 1:** Giới thiệu tổng quan về sự phát triển của game 3D trên thiết bị di động, các engine hỗ trợ làm game 3D trên thiết bị di động.

**Chương 2:** Giới thiệu tổng quan về thư viện đồ họa OpenGL ES, trình bày lịch sử phát triển, các tính năng chính, một số khái niệm cơ bản và cơ chế hoạt động của OpenGL ES.

**Chương 3:** Giới thiệu tổng quan về Adreno SDK Framework, trình bày các khái niệm cơ bản, các tính năng nổi bật, đặc biệt là kiến trúc của framework.

**Chương 4:** Trình bày các vấn đề và giải pháp khi xây dựng ứng dụng game 3D trên Adreno SDK, bên cạnh đó tìm hiểu một số thư viện hỗ trợ xử lí âm thanh trong game và các game service cần thiết.

**Chương 5:** Trình bày một cách cụ thể về ứng dụng game 3D đã xây dựng được, các kiến trúc sử dụng trong game và kết quả thực nghiệm đã đạt được.

**Chương 6:** Kết luận và đưa ra hướng phát triển cho ứng dụng.

# Chương 2. Tổng quan về OpenGL ES

## 2.1. OpenGL ES là gì?

## 2.2. Sơ lược lịch sử hình thành và phát triển của OpenGL ES

## 2.3. Tính năng của OpenGL ES

## 2.4. Các khái niệm cơ bản trong OpenGL ES

### 2.4.1. Texture

### 2.4.2. Shader

### 2.4.3. Mesh

## 2.5. Cơ chế hoạt động của OpenGL ES

## 2.6. Kết luận

# Chương 3. Tổng quan về Adreno SDK Framework

## 3.1. Adreno SDK là gì?

## 3.2. Tính năng của Adreno SDK

## 3.3. Tại sao chọn Adreno SDK

## 3.4. Kết luận

# Chương 4. Một số vấn đề và giải pháp khi xây dựng game 3D với Adreno SDK

## 4.1. Load mô hình 3D

## 4.2. Animation cho mô hình 3D

## 4.3. Tạo địa hình trong game

## 4.4. Chiếu sáng cảnh vật

## 4.5. Xử lí di chuyển trong bản đồ

## 4.6. Tạo hiệu ứng Billboard

## 4.7. Xây dựng GUI

## 4.8. Âm thanh trong game

## 4.9. Game Service

# Chương 5. Ứng dụng game 3D phát triển trên Adreno

## 5.1. Giới thiệu game

Ứng dụng game đối kháng 3D chúng em xây dựng thuộc thể loại game đánh quái với phong cách hành động đầy kịch tính, mang lại sự giải trí cao và sức hấp dẫn đối với người chơi. Giống như các game đối kháng khác, game chia bản đồ thành 2 bên, mỗi bên sở hữu một đội quân gồm xe, lính và anh hùng (hero). Nhiệm vụ của người chơi để có thể giành chiến thắng là tiêu diệt căn cứ của đối phương. Người chơi thua cuộc khi để cho đối phương tiêu diệt căn cứ của mình. Người chơi cùng đội quân của mình có thể tấn công đối phương theo các lối đi chính, phá hủy các tháp canh, cướp phá và lên cấp từ từ.

Người chơi sẽ hóa thân vào một nhân vật anh hùng (hero) được lựa chọn trước khi bắt đầu game. Mỗi hero có các chỉ số sức mạnh (strength), nhanh nhẹn (agility), thông minh (intelligence) và bộ kĩ năng đặc biệt (skill) khác nhau. Hai bên sẽ phải đối đầu với nhau trong một cuộc chiến toàn diện bao gồm việc giết quái, tiêu diệt hero đối phương lấy vàng, thu nhận kinh nghiệm, kiểm soát bản đồ, giao tranh, tấn công trụ, phòng thủ nhà... Người chơi có thể gia tăng sức mạnh cho hero bằng việc mua các trang bị bằng lượng vàng có được. Sự đa dạng về hero, skill và trang bị sẽ buộc người chơi phải lựa chọn và đưa ra những quyết định khác nhau có ảnh hưởng đến kết quả trận đấu vào từng thời điểm.

## 5.2. Các quy luật chơi chính

### 5.2.1. Di chuyển

Các nhân vật hỗ trợ bao gồm lính và xe là các nhân vật AI, được tự động sinh ra và tự động di chuyển theo các lối đi chính, sẵn sàng tấn công kẻ thù trong tầm nhìn của mình bất cứ lúc nào.

Hero được người chơi điều khiển, có khả năng di chuyển khắp mọi nơi trên bản đồ trừ nhưng nơi có vật cản. Người chơi chạm (touch) trên màn hình để di chuyển hero đến vị trí mong muốn.

### 5.2.2. Tấn công

Tùy theo khoảng cách đánh của nhân vật mà ta chia ra 2 loại tấn công: đánh gần và đánh xa. Lính, xe và tháp canh là các đối tượng AI, tấn công tự động quân và các công trình của đối phương xuất hiện trong tầm đánh của mình, mỗi phát đánh gây ra một lượng sát thương đúng bằng chỉ số sát thương (damage) của nó trong thời điểm hiện tại, trong đó lính đánh gần, xe và trụ đánh xa.

Hero được người chơi điều khiển và có 2 loại đòn đánh: đòn tấn công cơ bản (basic attack) và các kĩ năng đặc biệt (special skills). Đòn tấn công cơ bản được thực hiện một cách tự động khi có đối tượng kẻ thù nằm trong tầm đánh của hero, các kĩ năng thì được gắn vào những phím ảo ở màn hình bên phải. Đòn tấn công cơ bản gây lượng sát thương bằng với chỉ số sát thương của hero, còn mỗi kĩ năng đều có chỉ số sát thương riêng. Người chơi hoàn toàn có thể chỉ định một mục tiêu cụ thể cho hero của mình bằng cách chỉ thẳng vào mục tiêu đó. Khi phát đánh cuối cùng (last hit) khiến cho một đối tượng kẻ thù bị tiêu diệt, người chơi sẽ nhận được lượng vàng và kinh nghiệm ứng với đối tượng đó.

### 5.2.3. Hồi phục

Có 3 cách để hồi phục máu (health) và năng lượng (mana) cho hero: hồi phục cơ bản, hồi phục tại căn cứ, hồi phục dựa vào vật phẩm. Hồi phục cơ bản là lượng máu và năng lượng nhỏ được công thêm cho hero mỗi giây. Hồi phục tại căn cứ cho phép tăng lượng máu và năng lượng lên mức tối đa trong khoảng thời gian rất ngắn. Hồi phục dựa vào vật phẩm cho phép hero tăng tức thời lượng máu và năng lượng mà vật phẩm cung cấp.

### 5.2.4. Hồi sinh

Nếu căn cứ của phe mình chưa bị tiêu diệt, hero sẽ có khả năng sống lại (hồi sinh) sau khi chết. Thời gian đợi hồi sinh được tính từ lúc bị tiêu diệt và kéo dài phụ thuộc vào cấp độ và vật phẩm của hero hiện tại. Hero có cấp càng cao và vật phẩm càng mạng thì thời gian hồi sinh càng lâu và ngược lại.

### 5.2.5. Nâng cấp kĩ năng

Mỗi hero đều được cung cấp các kĩ năng (skill) khác nhau. Khi hero tiêu diệt một đối tượng của đối phương (lính, xe, các công trình, hero…) sẽ nhận được điểm kinh nghiệm (exp) tương ứng với đối tượng đó. Điểm kinh nghiệm này được cộng dồn vào điểm kinh nghiệm hiện tại cho đến khi đạt đến ngưỡng lên cấp. Khi đó tùy vào sự lựa chọn của người chơi, có thể nâng 1 điểm cho các chỉ số tổng quát (damage, health, mana…) hay một điểm cho các kĩ năng đặc biệt (skill) để kĩ năng này trở lên mạnh hơn, linh động hơn.

### 5.2.6. Tài nguyên

Tài nguyên được sử dụng trong game là vàng. Vàng được dùng để trang bị thêm các vật phẩm cho hero để tăng cường sức mạnh. Số vàng của mỗi hero có được từ số vàng tăng tự động mỗi giây và số vàng nhận được khi tiêu diệt các đối tượng của đối phương.

## 5.3. Các khái niệm trong game

### 5.3.1. Bản đồ thu nhỏ

Bản đồ thu nhỏ (minimap) là thể hiện thu nhỏ của toàn bộ địa hình game và được đặt ở góc phải trên màn hình. Bản đồ thu nhỏ cho phép người chơi theo dõi và dự đoán được thế cục của toàn trận đấu. Đọc minimap là một kĩ năng cần thiết cho người chơi, từ minimap chúng ta có thể xem được vị trí của hero, quân lính… của đối phương, biến mất khi nào và xuất hiện khi nào rồi từ đó suy đoán chiến thuật cho hợp lí.

### 5.3.2. Người chơi

Mỗi người chơi sẽ có tương ứng cho mình một hero được lựa chọn khi bắt đầu game. Nhiệm vụ của người chơi là điều khiển hero giết quái, tiêu diệt hero đối phương lấy vàng, thu nhận kinh nghiệm, kiểm soát bản đồ, giao tranh, tấn công trụ, phòng thủ nhà... Số lượng hero trong game được xây dựng rất đa dạng bao gồm nhiều loại khác nhau. Có những loại thiên về phép thuật, hỗ trợ và cũng có loại thiên về sức mạnh… Người chơi tùy vào sở thích và tình huống game cụ thể để có thể đưa ra sự lựa chọn hero hợp lí cho mình.

### 5.3.3. Công trình

Công trình trong game bao gồm nhà chính và các tháp canh được đặt trên các lối đi chính, ngoài ra còn có nơi mua bán vật phẩm (shop) cho hero. Các công trình này được xây dựng khác nhau với mỗi bên, giúp cho người chơi dễ dàng nhận biết. Tuy nhiên chúng được cân bằng về lượng máu, sức mạnh, vị trí và số lượng để bảo đảm tính hài hòa của game.

### 5.3.4. Quân lính

Quân lính trong game là các nhân vật AI bao gồm lính đánh gần và xe đánh xa. Quân lính được sinh ra tự động với số lượng cố định và theo chu kì tại căn cứ, sau đó sẽ di chuyển theo các lối đi chính, chúng sẽ tấn công tất cả các đối tượng của đối phương trên đường đi của mình. Cũng giống như các công trình trong game, quân lính cũng được phân biệt rõ ràng giữa 2 bên và đảm bảo tính công bằng, đây là lực lượng hỗ trợ rất đắc lực cho người chơi trong nhưng pha đẩy trụ, bảo vệ trụ hay là nguồn cung cấp vàng và kinh nghiệm khi bị tiêu diệt.

### 5.3.5. Kĩ năng

Hệ thống kĩ năng (skill) trong game được xây dựng rất phong phú và đa dạng. Các skill đó có thể skill về sức mạnh, phép thuật, hay các skill thiên về hỗ trợ, bảo vệ chính mình… Các skill này được tổ hợp một cách hài hòa, hợp lí để tạo nên một bộ skill cho các hero, mỗi hero có một bộ skill khác nhau với những thế mạnh và cách chơi riêng. Người chơi có thể tăng cường sức mạnh cho skill bằng cách tăng 1 điểm cho skill đó sau mỗi lần lên cấp hay mua thêm các vật phẩm phù hợp khác.

### 5.3.6. Vật phẩm

Vật phẩm game được bán trong cửa hàng, người chơi có thể trang bị vật phẩm cho hero từ lượng vàng có được. Vật phẩm hỗ trợ rất nhiều cho sức mạnh của hero và là nhân tố quan trọng để đánh giá thực lực của một hero. Sự đa dạng về số lượng vật phẩm sẽ buộc người chơi phải lựa chọn và đưa ra những quyết định khác nhau có ảnh hưởng trực tiếp đến kết quả trận đấu.

## 5.4. Kiến trúc trong game

### 5.4.1. Kiến trúc tổng thể

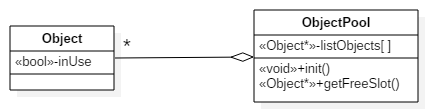
|  |
| --- |
| Adreno SDK  My Framework:  OpenGL ES  Khởi tạo và quản lí đối tượng  Quản lí thay đổi màn hình  Quản lí trạng thái nhân vật  Bắt sự kiện  Lớp cơ sở  Quản lí tài nguyên |

Hình 5.1 Kiến trúc tổng thể của game

Hình 5.1 thể hiện kiến trúc tổng thể của ứng dụng game đối kháng 3D chúng em xây dựng dựa trên Adreno SDK . Các thành phần chính trong kiến trúc này bao gồm:

* **Khởi tạo và quản lí đối tượng**: kiến trúc này giúp cho việc tạo và quản lí các đối tượng trong game trở nên dễ dàng hơn, các đối tượng này có thể là hero, quân lính, trụ hay một hiệu ứng phép thuật… (xem chi tiết về kiến trúc này trong phần 5.4.2)
* **Quản lí thay đổi màn hình**: đây là kiến trúc hỗ trợ việc thay đổi các màn hình của game, từ các màn hình menu game đến các màn hình chơi game, kết thúc game… (xem chi tiết về kiến trúc này trong phần 5.4.3)
* **Quản lí trạng thái nhân vật**: một nhân vật trong game có thể có rất nhiều trạng thái chuyển động: đứng yên, di chuyển, tấn công, đuổi bắt kẻ thù… kiến trúc này rất hữu ích cho việc thay đổi và cập nhật các trạng thái chuyển động này (xem chi tiết về kiến trúc này trong phần 5.4.4)
* **Bắt sự kiện**: có nhiều sự kiện trong game xảy ra mà không biết trước được thời gian hay vị trí cụ thể (click một button, mua một món đồ trong cửa hàng, phá hủy một tháp canh của đối phương…), kiến trúc giúp ta hoàn toàn kiểm soát được những vấn đề đó (xem chi tiết về kiến trúc này trong phần 5.4.5)
* **Lớp cơ sở**: các đối tượng có cùng đặc điểm, tính chất nào đó sẽ cùng được kế thừa từ các lớp cơ sở đã được dựng sẵn, các lớp cơ sở này tạo gắn kết chặt chẽ giữa các đối tượng cũng như hỗ trợ việc trao đổi, cập nhật thông tin của các đối tượng đó (xem chi tiết về kiến trúc này trong phần 5.4.6)
* **Quản lí tài nguyên**: các tài nguyên trong game sẽ được load lên cùng một lúc khi bắt đầu game và quản lí chung tại một khu vực, điều này giúp chúng ta dễ dàng quản lí và tránh thất thoát bộ nhớ.

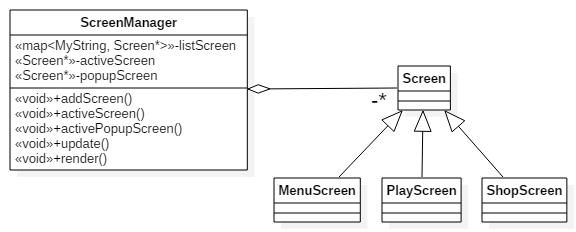
### 5.4.2. Kiến trúc khởi tạo và quản lí đối tượng



Hình 5.2 Sơ đồ lớp khởi tạo và quản lí đối tượng

Hầu hết các đối tượng trong game (hero, quân lính, tower, hiệu ứng phép…) đều được quản lí theo kiến trúc này. Mỗi Object sẽ có tương ứng với mình lớp ObjectPool như sau: Hero tương ứng với HeroPool, Tower tương ứng TowerPool… Lớp ObjectPool chứa một mảng các phần tử Object đã cố định trước số lượng, số lượng này được lựa chọn sao cho đảm bảo số thể hiện của lớp Object tại thời điểm bất kì trong suốt quá trình chơi game không được vượt quá. Mảng Object này sẽ được tạo sẵn trong hàm init( ) của lớp ObjectPool và để ở trạng thái chưa được sử dụng. Trong lớp Object có chứa một thuộc tính inUse, thuộc tính này cho biết đối tượng hiện tại có đang được sử dụng hay không. Điều đó có nghĩa là các phần tử của mảng Object trong lớp ObjectPool không hề bị hủy đi khi không dùng tới, chúng bị đánh dấu là đang không được sử dụng (inUse = false). Và khi có yêu cầu cần tạo ra một thể hiện lớp Object, lớp ObjectPool chỉ cần duyệt qua listObjects và trả về một phần tử hiện không được sử dụng thông qua phương thức getFreeSlot( ). Điều này giúp chúng ta tiết kiệm và quản lí tốt được bộ nhớ, vì tất cả các đối tượng chỉ được khởi tạo một lần lúc bắt đầu game và được tái sử dụng lại khi cần thiết. Ngoài ra, kiến trúc này tránh việc phải xóa đi các phần tử nằm ở vị trí bất kì trong một mảng khi chúng bị tiêu diệt hay không cần dùng đến, điều này rất dễ gây xung đột đối với các game xử lí thời gian thực.

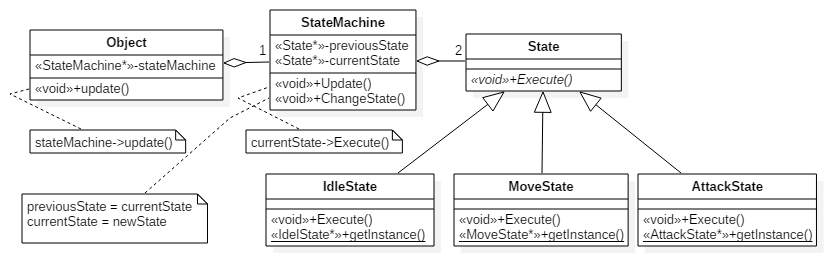
### 5.4.3. Kiến trúc quản lí sự kiện thay đổi màn hình



5.3 Sơ đồ lớp quản lí sự kiện thay đổi màn hình

Lớp ScreenManager chứa một danh sách các đối tượng Screen có kiểu dữ liệu là **map<MyString, Screen\*>**, kiểu dữ liệu này cho phép chúng ta dễ dàng quản lí các phần tử của nó thông qua tên truyền vào. Ngoài ra, hai thuộc tính activeScreen và popupScreen giữ thông tin của Screen hiện tại, cũng như Screen Popup hiện chồng lên Screen Active nếu có. Khi bắt đầu vào game, đối tượng screenManager sẽ được khởi tạo, tất cả các screen của game sẽ được thêm vào thuộc tính listScreen thông qua phương thức addScreen( ). Và dĩ nhiên 2 phương thức update( ), render( ) của lớp ScreenManager chỉ thực hiện cho Screen Active và Screen Popup được xác định thông qua 2 thuộc tính activeScreen và popupScreen. Chính vì thế, để thay đổi Screen Active cũng như Screen Popup ta chỉ cần thay đổi giá trị 2 thuộc tính activeScreen và popupScreen thông qua 2 phương thức activeScreen( MyString ) và activePopupScreen( MyString ). Với kiến trúc này, việc quản lí thay đổi màn hình trở nên linh động hơn, tạo được mối liên hệ giữa các màn hình với nhau trong game.

### 5.4.4. Kiến trúc quản lí trạng thái của đối tượng



Hình 5.4 Sơ đồ lớp quản lí trạng thái của nhân vật

Mỗi đối tượng (Object) sẽ chứa một thuộc tính kiểu StateMachine. Tất cả các vấn đề về trạng thái của đối tượng được giao hết cho thuộc tính stateMachine này.

Lớp StateMachine chứa 2 thuộc tính kiểu State là previousState và currentState, previousState lưu trữ trạng thái trước đó của đối tượng, currentState lưu trữ trạng thái hiện tại của đối tượng. Điều này giúp chúng ta dễ dàng quản lí trạng thái hiện tại, cũng như quay về trạng thái trước đó khi cần thiết hay thay đổi trạng thái của đối tượng thông qua phương thức ChangeState( ):

void ChangeState(State\* newState)

{

previousState = currentState;

currentState = newState;

}

Lớp State chứa phương thức thuần ảo *Execute( )*, phương thức này sẽ được override lại trong các lớp con IdelState, MoveState, AttackState… Tất cả các công việc của một trạng thái sẽ được cài đặt trong phương thức này. Hơn thế nữa, các lớp con kế thừa từ lớp State được cài đặt theo mẫu Singleton để đảm bảo chỉ có một trạng thái loại này ứng với một đối tượng cụ thể:

class State\_Idle : public State

{

private:

State\_Idle(){}

State\_Idle(const State\_Idle&);

public:

static State\_Idle\* instance() { static State\_Idle ins; return &ins; }

virtual void Execute(Pawn\* pawn);

};

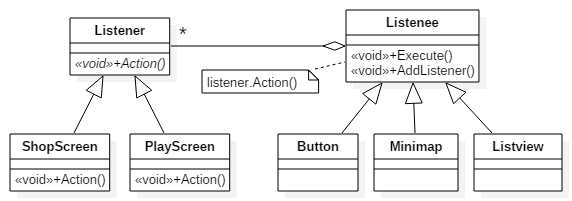
Kiến trúc trên được phát triển từ mẫu State, theo kiến trúc này thì quá trình cập nhật trạng thái của một đối tượng diễn ra như sau:

object->update( )

stateMachine->Update( )

currentState->Execute( )

### 5.4.5. Kiến trúc bắt sự kiện



Kiến trúc này được thiết kế dựa trên mẫu Command. Lớp Listenee đại diện cho các đối tượng phát sinh ra sự kiện, các đối tượng này có thể là Button, Minimap, Listview... và các sự kiện có thể là OnButtonClick, OnMiniMapClick, OnListviewScroll... Lớp Listener đóng vai trò là người nhận sự kiện từ các đối tượng Listenee. Chẳng hạn, ta có ShopScreen là đối tượng Listener và ShopButton là đối tượng Listenee, khi người chơi click vào ShopButton thì sự kiện này sẽ được gửi đến cho ShopScreen, lúc này ShopScreen sẽ được hiển thị lên cho phép người chơi mua các vật phẩm trong game.

Một đối tượng Listenee sẽ bao gồm một mảng các đối tượng Listener, là tất cả các đối tượng có liên quan và cần bắt sự kiện mà đối tượng Listenee phát ra. Chúng ra có thể dễ dàng thêm một Listener vào mảng này thông qua phương thức AddListener( ). Các lớp con kế thừa từ lớp cha Listener sẽ override lại phương thức Action( ), đây là phương thức chứa những công việc sẽ làm khi nhận được sự kiện từ đối tượng Listenee, theo như ví dụ trên thì việc hiện thị ShopScreen sẽ được cài đặt trong phương thức này. Đối tượng Listenee gọi hàm Execute( ) để gửi sự kiện cho các Listener của mình:

void Execute()

{

foreach (Listener listener in listListener)

listener.Action();

}

### 5.4.6. Kiến trúc lớp cơ sở

Trong game, các đối tượng có cùng những đặc điểm, tính chất nổi bật nào đó sẽ cùng được kế thừa từ một lớp cơ sở đặc trưng cho đặc điểm, tính chất đó. Việc làm này trước hết là giúp rút ngọn mã nguồn chương trình, tránh lặp đi lặp lại những đoạn code thừa không cần thiết. Ngoài ra, kiến trúc này còn giúp tạo sự gắn kết giữa các đối tượng, giúp chúng ta dễ dàng gom nhóm các đối tượng khi cần truy xuất. Chẳng hạn chúng ta có thể gom nhóm các đối tượng có thể di chuyển được hay các đối tượng sống, có máu…

**Lớp BaseEntity**: đây là lớp đối tượng cơ sở nhất trong game, tất cả các đối tượng khác đều được kế thừa từ lớp này. Lớp lưu trữ một số thông tin cơ bản của các đối tượng như loại đối tượng (hero, quân, trụ…), đối tượng này thuộc team nào (team mình, team đối phương, team trung lập…)…

class BaseEntity

{

public:

BaseEntity();

virtual ~BaseEntity();

ENTITY\_TYPE getEntityType()const;

void setEntityType(ENTITY\_TYPE entityType);

TEAM\_TYPE getTeamType()const;

void setTeamType(TEAM\_TYPE teamType);

protected:

ENTITY\_TYPE m\_entityType;

TEAM\_TYPE m\_teamType;

};

**Lớp LivingEntity**: lớp này quản lí tất cả các đối tượng sống, có máu, có thể tấn công được trong game. Trong game thường có rất nhiều loại đối tượng khác nhau, nếu không gom nhóm các đối tượng sẽ rất khó khăn cho chúng ta khi phải duyệt tìm và cập nhật một đối tượng nào đó. Chẳng hạn, để tìm một đối tượng để tấn công thay vì phải duyệt tìm tất cả các đối tượng game, sau đó loại bỏ đi những đối tượng không phù hợp chúng ta chỉ cần chọn một đối tượng trong mảng các đối tượng LivingEntity đã được tạo trước đó. Lớp LivingEntity cho biết thông tin về lượng máu hiện tại, lượng máu tối đa, chỉ số sát thương, tầm đánh, đối tượng bị tấn công…

class LivingEntity : public BaseEntity

{

public:

LivingEntity();

virtual ~LivingEntity();

protected:

int m\_maxHealth;

int m\_health;

int m\_damage;

float m\_atkRange;

LivingEntity\* m\_atkTarget;

};

**Lớp MovingEntity**: lớp này giúp thực hiện tất cả các công việc liên quan đến di chuyển của đối tượng. Vì đa số các đối tượng trong game có cách thức di chuyển tương đối giống nhau, chỉ khác nhau về một số chỉ số tốc độ, góc quay… nên việc xây dựng lớp giúp tránh trùng lặp các đoạn code thừa và giúp việc quản lí di chuyển cho đối tượng trở nên dễ dàng hơn. Lớp chứa rất nhiều chỉ số liên quan đến di chuyển: tốc độ, tốc độ quay, vị trí hiện tại, vị trí đích đến, đường đi… và các phương thức xoay đối tượng, di chuyển đến điểm đích, đi theo đường mô tả trước…

class MovingEntity

{

public:

MovingEntity();

virtual ~MovingEntity();

void turnTo(const MyVec3& target);

void disFollowPath();

void reFollowPath();

protected:

MyVec3 m\_position;

MyVec3 m\_target;

float m\_currOrientation;

float m\_speed;

float m\_turnSpeed;

std::vector<MyVec3> m\_path;

int m\_pathPivot;

};

### 5.4.7. Kiến trúc quản lí tài nguyên

Lượng tài nguyên trong game rất lớn, vì vậy cần phải quản lí tập trung để dễ dàng kiểm soát và tránh thất thoát bộ nhớ. Tất cả các tài nguyên game đều được load một lần duy nhất khi bắt đầu game, các đối tượng muốn sử dụng tài nguyên nào sẽ giữ một biến con trỏ đến tài nguyên đó, tránh sự sao chép làm tốn bộ nhớ.

Các tài nguyên cùng loại (texture, shader, sprite…) sẽ được load lên cùng một mảng tài nguyên và được truy xuất, gọi tên thông qua biến Enum thay vì bộ chỉ số mảng thông thường (0, 1, 2…). Điều này tránh sự nhập nhằng, nhầm lẫn khi sử dụng tài nguyên, giúp cho việc viết mã nguồn trở nên dễ dàng hơn.

class Layer\_World

{

private:

enum

{

SHADER\_TERRAIN,

SHADER\_MESH,

SHADER\_BILLBOARD,

NUM\_SHADERS,

};

enum

{

TEXTURE\_TERRAIN\_BLEND,

TEXTURE\_BLOODBAR\_GREEN,

TEXTURE\_BLOODBAR\_RED,

NUM\_TEXTURES,

};

enum

{

SPRITE\_SHEET\_ENERGY\_BALL,

SPRITE\_SHEET\_BULLET,

NUM\_SPRITE\_SHEETS,

};

public:

Layer\_World();

~Layer\_World();

private:

Shader m\_shaders[NUM\_SHADERS];

Texture m\_textures[NUM\_TEXTURES];

SpriteSheet m\_spriteSheets[NUM\_SPRITE\_SHEETS];

};

## 5.5. Kết quả thực nghiệm

# Chương 6. Kết luận và hướng phát triển

## 6.1. Các kết quả đạt được

Sau khi tìm hiểu các phương pháp xây dựng game 3D với framework Adreno SDK chúng em đã hoàn thành ứng dụng game đối kháng thời gian thực chạy trên Android và Windows. Trong quá trình xây dựng game, chúng em đã đưa ra các vấn đề gặp phải và trình bày giản pháp để giải quyết các vấn đề đó.

Các màn hình giao diện chính của ứng dụng game mà chúng em đã xây dựng trong luận văn này:

## 6.2. Hướng phát triển

* Tối ưu hóa các xử lí và bộ nhớ sử dụng giúp game chạy nhanh hơn, mượt hơn.
* Hỗ trợ chế độ chơi nhiều người (multiplayer) và chơi online thông qua Bluetooth hoặc Internet, sử dụng Google Game Service.
* Tăng thêm số lượng màn chơi để người chơi có thêm nhiều lựa chọn.
* Xây dựng thêm nhiều Hero, Skill và Item để game ngày càng trở nên hấp dẫn hơn.
* Cải thiện trí thông minh nhân tạo của máy giúp người chơi có cảm giác thật hơn, hợp lí hơn khi đấu với máy.