**Lời Cảm Ơn**

Lời đầu tiên tôi xin bày tỏ lòng biết ơn chân thành tới các thầy cô giáo Trường Đại Học Bách Khoa – Đại Học Đà Nẵng nói chung và các thầy cô bộ môn công nghệ phần mềm nói riêng. Trong suốt thời gian tôi học tại trường, các thầy cô đã luôn tận tình dạy dỗ, chỉ bảo để tôi có được kết quả như ngày hôm nay.

Đặc biệt, tôi xin chân thành cảm ơn giảng viên, tiến sĩ Lê Thị Mỹ Hạnh. Cảm ơn cô vì những định hướng, nhận xét quý báu và động viên kịp thời để tôi hoàn thành khóa luận này. Tôi cũng cảm ơn Gameloft – Đà Nẵng đã tạo điều kiện cho tôi có môi trường làm việc trong quá trình hoàn thành khóa luận này.

Cuối cùng tôi xin bày tỏ lòng biết ơn sâu sắc tới gia đình, bạn bè, những người đã luôn động viên, giúp đỡ tôi cả về vật chất lẫn tinh thần trong suốt những năm tháng qua.

Đà nẵng, ngày …. Tháng … năm…

Sinh viên

**Tóm tắt**

Sự phát triển của công nghệ mô phỏng và giải trí, ngày càng có nhiều ứng dụng đồ họa và game 3D được xậy dựng nhằm phục vụ mục đích nghiên cứu và mô phỏng và giải trí… Cùng với đó, công nghê phần cứng cũng phát triển rất nhanh, đặc biệt là các phần cứng xử lý đồ họa có thể lập trình được và các ngôn ngữ lập trình trên các phần cứng này.

Nói đến game thì chắc phần lớn mọi người đều đồng ý đó là một loại hình giải trí có sức thu hút rất lơn, nhất là với giới trẻ hiện nay. HIện nay game được phát triển mạnh trên các thiết bị di động.

Xuất phát từ thực tế đó, nội dung của đồ án tập trung nghiên cứu xây dựng một hệ thống game engine để build game trên các thiết bị di động sử dụng Opengl ES 2.0. Khi phát triển game 3D, người lập trình không cần phải quan tâm đến quá trình quản lý tài nguyên, cũng như xủ lý đồ họa ở mức thấp bên dưới. Thay vào đó, họ chỉ cần tập trung vào quản lý ở mức logic các thành phần của game, hoặc thêm mới các thành phần cơ sở do hệ thống cung cấp.

**CHƯƠNG 1 – CHƯƠNG MỞ ĐẦU**

* 1. **Lý do chọn đề tài**

Ngành công nghiệp phát triển game trên thế giới đang phát triển như vũ bão. Cách đây 20 năm, game rất đơn giản và chỉ một người hoặc một nhóm nhỏ người phát triển nhưng ngày nay, các game hiện đại với đồ họa và khả năng tương tác ấn tượng thường được phát triển bởi một đội ngũ đông đảo người thiết kế game, lập trình viên, nghệ sĩ… trong thời gian ròng rã từ một đến ba năm. Chính việc game ngày càng trở nên phức tạp như vậy nên hiện nay, các game thương mại không còn phát triển từ nguyên thủy nữa mà được phát triển lên từ game engine. Game engine hỗ trợ việc xây dựng game một cách nhanh chóng, đơn giản hơn đồng thời cung cấp khả năng tái sử dụng code cao do đó có thể phát triển nhiều game từ một game engine.

Cùng với sự bùng nỗ của các thiết bị di động, game mobile hiện nay đang trở nên hot hơn bao giờ hết, bởi sự giải trí tiện lợi nhưng không kém phần hấp dẫn. Số game được phát hành mỗi ngày lên đến con số hàng tram.

Tôi chọn đề tài tốt nghiệp này không phải với một tham vọng quả lớn, chỉ là muốn đi những bước chân chập chững đầu tiên vào thế giới phát triển game rộng lơn, để thu lượm kiến thức về lĩnh vực khó khăn nhưng đầy thú vị này và hy vọng ở một tương lai không xa, tôi có thể góp một phần sức lực vào ngành công nghiệp game của nước nhà.

* 1. **Mục Tiêu**
* Tìm hiều lập trình OpenGL ES 2.0
* Tìm hiểu cấu trúc 1 game
* Thiết kế mô hình hệ thống game
* Xây dựng game engine
* Xậy dựng 1 game 3D và chuyển sang di động ( Android )
* Thiết kế game Kill The Rat mang tính giải trí đơn thuần
  1. **Đối tượng nghiên cứu và phạm vi hệ thống**

**Đối tượng:**

* Ngôn ngữ : C++ , Java
* Công cụ : Visual Studio 2013, Android Studio
* Hệ điều hành window và tài nguyên hệ thống Android

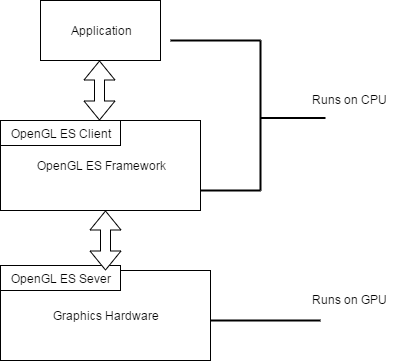
**Phạm vi:** Đề tài chỉ dừng lại tại phần xây dựng render game engine cho game và build game Kill The Rat trên nền tảng window và Android Mobile

* 1. **Phương pháp tiếp cận**
* Khảo sát các game có lối chơi gần giống trên mạng internet để rút ra tính đột phá cho game
* Thu thập các thông tin cần thiết
* Phân tích các thông tin và yêu cầu người dung và đề ra các yêu cầu chức năng cho game.
* Tìm hiểu công cụ hỗ trợ phát triển game Visual Studio và Android Studio

**Chương 1: Tổng quan về Game Engine**

1. **Khái niệm Game Engine**

Game Engine là một công cụ hỗ trợ, một lớp trung gian ở giữa các ứng dụng game và nền tảng bên dưới , các thư viện lập trình cấp thấp. Game Enigine giúp phát triển ứng dụng game một cách nhanh chóng và đơn giản, đồng thời cung cấp khả năng tái sử dụng mã nguồn cao do có thể phát triển nhiều ứng dụng game khác nhau từ một game Engine.



Hình 1: Kiến trúc tổng quan của Game Engine Opengl ES

1. **Mô hình Game Engine**

Game Engine hỗ trợ đa nền (Window PC và Mobile Android). Và nhắm đến thị trường Game Mobile đang rất phổ biến trên các thiết bị di động.

Đầu tiên là khả năng chạy đa nền. Để đạt được điều này, Engine sử dụng các bản build khác nhau trên các nền tảng khác nhau (chứ không phải sử dụng thông dịch). Để giải quyết vấn đề này, đầu tiên em chọn build game bằng Visual Studio 2013 trên nền tảng window sử dụng C++. Sau đó sử dụng NDK trên android để sử dụng code C++ và build game ra Mobilde Android.

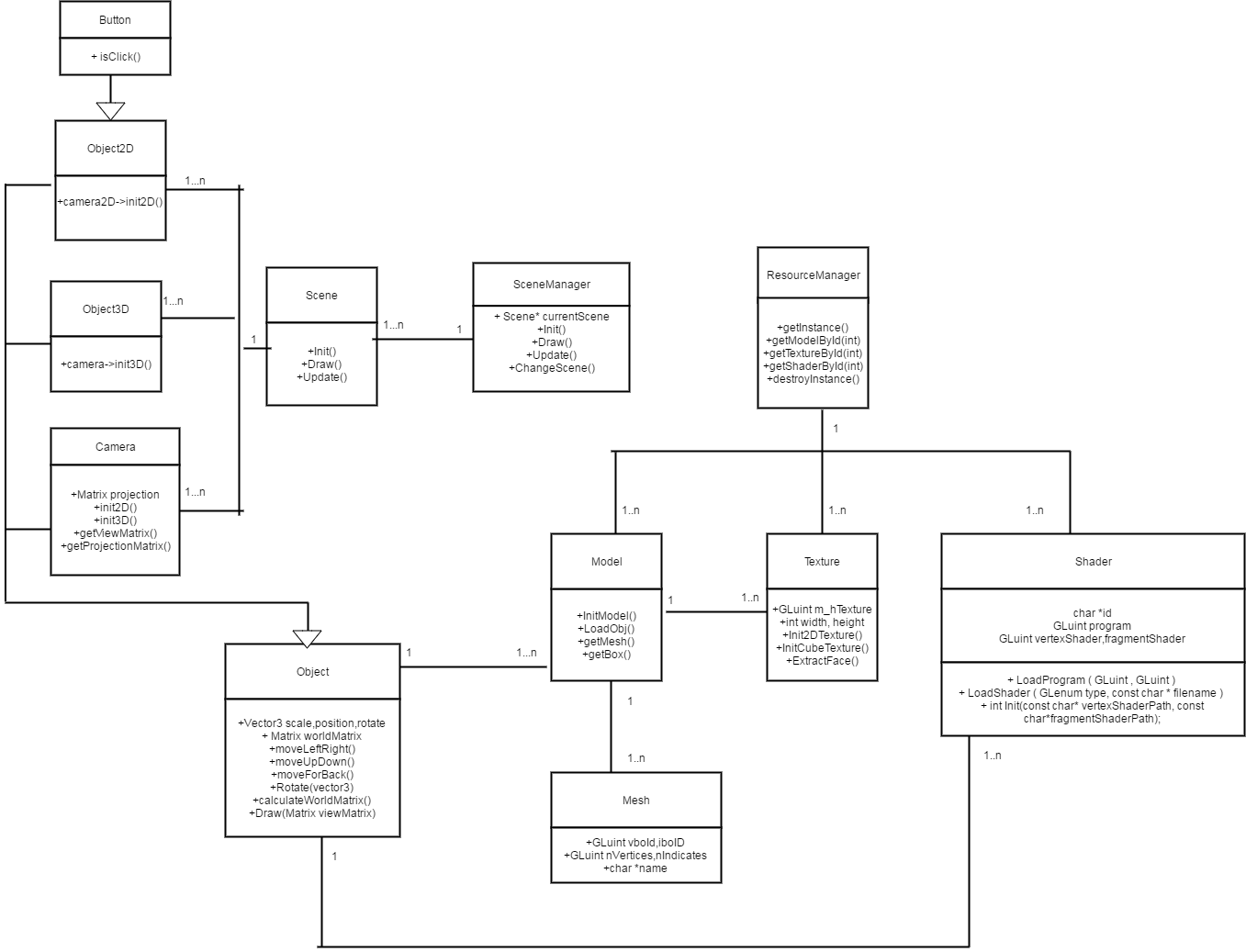
Engine được thiết kế hướng đối tượng và yêu cầu hiệu năng chạy cao nên việc sử dụng ngôn ngữ C++ ngôn ngữ đáp ứng điều kiện trên.

1. **Các thành phần chính của Engine**

Engine hỗ trợ người dùng các công việc thiết yếu để tao cảnh 3D và các mô hình 3D. Người dùng sẽ không cần biết nhiều đến những công việc tầng thấp như quá trình đọc file tài nguyên, sử dụng các API đồ họa 3D, tạo các hiệu ứng… mà quản lý logic các đối tượng bằng các giao diện do Engine cung cấp.

Engine giao tiếp với phần cứng đồ họa thông qua thư viện đồ họa 3D cấp thấp, cụ thể ở đây là OpenGL .

Sau đây là biểu đồ lớp của engine



Hình 1.2 : OpenGL ES 2.0 Game Engine Class Diagram

* Resource Manager là mô đun quản lý tài nguyên cần thiết để xây dựng nên các Object và cảnh 3D như texture 2D , cubemap… Mô đun này giúp cho việc dễ dàng quản lí tài nguyên , tăng tốc độ game và tránh việc sử dụng các model , texture trùng nhau
* Scene Manager là mô đun quản lý các màn chơi trong game (Scene), bao gồm các hàm chuyển màn chơi, pause ..
* Scene là một cảnh game level, chẳng hạn như các cảnh trong 1 game thường có chia làm Start Scene, Play Scene , GameOver Scene. Việc chia làm các Scene và thông quan Scene Manager giúp cho việc quản lý game trở nên dễ dàng hơn, dễ control hơn trong quá trình phát triển game.
* Object chia làm 2 loại là Object 2D và Object3D. Việc mô phỏng chuyển động của các Object này sẽ thông qua một ma trận worldMatrix , dựa vào các vector postion, scale, rotate mà ta sẽ mô phỏng chuyển động cho các Object này. Chúng ta sẽ bàn kỹ hơn về vấn đề này ở chương sau.
* Camera trong engine giống như góc nhìn ngoài đời thật. Ở các vị trí khác nhau sẽ cho ra các kết quả khác nhau khi nhìn các Object. Camera cũng là một object trong engine , và bằng vào việc di chuyển camera trong môi trường 3D sẽ cho ra các kết quả khác nhau khi nhìn vào một cảnh game.
* Một Object sẽ bao gồm nhiều model và texture và mỗi model sẽ được tạo ra từ nhiều mesh.
* Shader sẽ giúp chúng ta load các file shader tạo hiệu ứng màu sắc cho object. Bằng việc này chúng ta chỉ việc thao tác với các file shader để tạo hiệu ứng cho object một cách đơn giản hơn.

Chương 3: Cơ sở lý thuyết

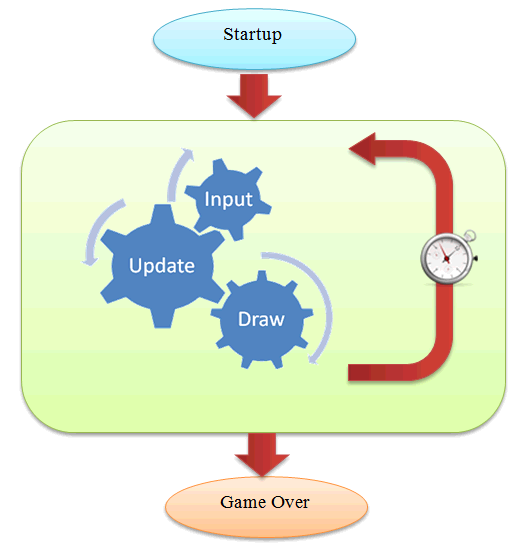
3.1 Vòng lặp game

Phần cốt lõi của hầu hết các game chính là vòng lặp được dùng để cập nhật và hiển thị trạng thái của game.

Một ví dụ đơn giản:

|  |  |
| --- | --- |
|  | while(gameRunning)  {      processInput(); // keyboard, mouse,...      updateGame();      draw();      // checkGameOver();  } |

Minh họa:



* 1. Tổng quan về Opengl

OpenGL (Open Graphics Library) là một tiêu chuẩn kỹ thuật đồ họa được hình thành với mục đich định ra một giao diện lầm trình ứng dụng API theo không gian 3 chiều (Theo wikipedia.com). Tuy nhiên, tự bản thân OpenGL không phải là một API, mà chỉ là một đặc tả được xây dựng và duy trì bởi Khrnos Group.

Các đặc tả OpenGL xác định chính xác kết quả đầu ra của mỗi chức năng và cách nó phải thực hiện. Sau đó, các lập trình viên sẽ thực hiện đặc tả này để tìm ra giải pháp về cách hoạt động của chức năng này. Do đặc tả OpenGL không cung cấp cho chúng ta các chi tiết thực hiện nên các phiên bản thực tế của OpenGL được phép có các cài đặt khác nhau, miễn là kết quả của chúng tuân thủ đặc tả.

Những developer phát triển các thư viện OpenGL thực tế thường là các nhà sãn xuất card đồ họa. Mỗi card đồ họa mà bạn mua hỗ trợ các phiên bản cụ thể của OpenGL là các phiên bản của OpenGL được phát triển riêng cho các card này một cách hang loạt. Chẳng hạn, khi sử dụng hệ thống của Apple, thư viên OpenGL được duy trì bởi chính Apple và dưới Linux có tồn tại sự kết hợp của các phiên bản của các nhà cung cấp đồ họa và sự thích nghi của những người thích thú với các thư viện này. Điều này cũng có nghĩ là khi OpenGL hiển thị các hành vi lạ, điều này có thể là lỗi của nhà sản xuất card đồ họa (hoặc bất cứ ai phát triển/ duy trì thư viện ).

Trước đây, sử dụng OpenGL có nghĩa là phát triển trong chế độ tức thời ( immediate mode) là một phương pháp dễ sử dụng để vẽ đồ họa. Hầu hết các chức năng của OpenGL đã được ẩn trong thư viện và các developer không được tự do xác định cách mà OpenGL hoạt động.

Các developer đã kiểm soat nhiều hơn đồ họa của họ. Chế độ tức thời thực sự dễ sử dụng và dễ hiểu, nhưng nó không đạt được hiệu quả cao. Vì lý do đó, các đặc tả bắt đầu không dung chức năng chế đọ tức thời từ phiên bản 3.2 và bắt đầu thúc đẩy các developer trong chế độ cấu hình lõi của OpenGL, một bộ phận đặc tả OpenGL đã loại bỏ tất cả các chức năng cũ.

Khi sử dụng mô hình cốt lõi của OpenGL, OpenGL buộc chúng ta phải sử dụng các phương pháp hiện đại. Bất cứ khi nào chúng ta cố gắng sư dụng một trong những chức năng không dung nữa của OpenGL, OpenGL sẽ gây ra lỗi và ngừng vẽ. Ưu điểm của việc học cách tiếp cận hiện đại là nó rất linh hoạt và hiệu quả, nhưng tiếc là cũng khó học hơn. Chế độ tức thời (immediate mode) trừu tượng khá nhiều từ các hoạt động thực tế OpenGL thực hiện và trong khi nó được dễ dàng tìm hiểu, thật khó để hiểu OpenGL hoạt động như thế nào. Cách tiếp cận hiện đại đòi hỏi developer thực sự hiểu được OpenGL và lập trình đồ họa và bên cạnh có một chút khó khăn trong việc tìm hiểu, mặc khác nó cho phép tính toán linh hoạt hơn, hiệu quả hơn và điều quan trọng nhất là hiểu rõ hơn về lập trình đồ họa.