Proof Of Concept

Game Battle City

Version 1.0

Revison History

|  |  |  |  |
| --- | --- | --- | --- |
| Created Date | Version | Description | Author |
| 08/10/2014 | 1.0 | Khởi tạo tài liệu | Bạch Đình Nhân Nguyễn Hữu Hoàng |
|  |  |  |  |
|  |  |  |  |

Contents

[1. Giới thiệu 4](#_Toc400623630)

[1.1 Mục đích tài liệu 4](#_Toc400623631)

[1.2 Đối tượng tài liệu 4](#_Toc400623632)

[1.3 Tác giả 4](#_Toc400623633)

[1.4 Thuật ngữ sử dụng 4](#_Toc400623634)

[2. Đặt vấn đề 4](#_Toc400623635)

[3. Giải pháp 4](#_Toc400623636)

[2. Prototype 4](#_Toc400623637)

[3. Thuật toán 6](#_Toc400623638)

[3.1 Phát hiện va chạm 6](#_Toc400623639)

[3.2 Điều khiển xe tăng computer 6](#_Toc400623640)

[3.2.1.Bài toán: 6](#_Toc400623641)

[3.2.2.Ý tưởng: 6](#_Toc400623642)

[3.2.3. Mô tả thuật toán 6](#_Toc400623643)

[3.2.4. Độ phức tạp của thuật toán: 6](#_Toc400623644)

[3.2.5. Cài đặt 7](#_Toc400623645)

[4. Cấu trúc AndEngine 9](#_Toc400623646)

[4.1 Sơ đồ cấu trúc của AndEngine 9](#_Toc400623647)

[4.2 Mô tả 9](#_Toc400623648)

# 1. Giới thiệu

## 1.1 Mục đích tài liệu

Tài liệu này nhằm chứng minh tính khả thi và khả năng hoàn thành dự án Game Battle City. Giới thiệu các thuật toán, nền tảng mà nhóm phát triển sẽ ứng dụng.

## 1.2 Đối tượng tài liệu

Đối tượng đọc tài liệu này gồm có:

* Khách hàng (Thầy Ngô Huy Biên và Thầy Trương Phước Lộc).
* Đội ngũ thực hiện dự án (Nhóm 7 môn Quản Lý Dự Án Phần Mềm).

## 1.3 Tác giả

Toàn bộ thành viên nhóm 7

## 1.4 Thuật ngữ sử dụng

Không có

# 2. Đặt vấn đề

Dự án Game Battle City là dự án về lĩnh vực giải trí dành cho thiết bị di động. Dự án này đã từng được hãng Nintendo phát triển rất thành công trên thiết bị console. Sau khi khảo sát và phân tích nhóm phát triển nhận thấy một số vấn đề chính cần giải quyết của dự án Game Battle City như sau:

* Thời gian thực hiện dự án ngắn.
* Cách phát hiện va chạm xảy ra trong Game.
* Cách điều khiển xe tăng computer một cách thông minh.

# 3. Giải pháp

Do dự án chỉ được phép thực hiện trong khoảng thời gian 3 tháng nên nhóm chúng tôi sẽ chọn nền tảng phát triển game có sẵn là AndEngine.

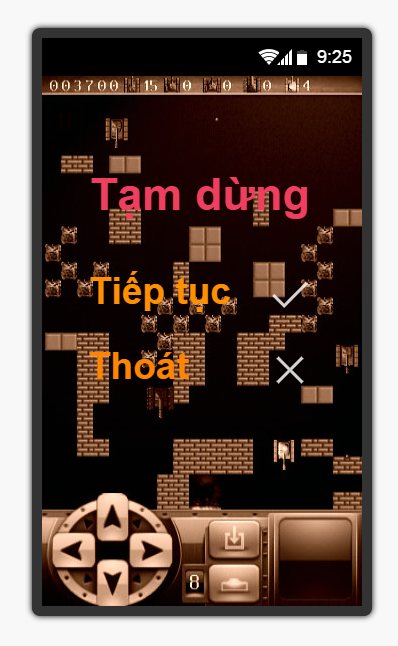
Để phát hiện được các va chạm xảy ra trong game chúng tôi sẽ sử dụng 1 kỹ thuật chính là: phát hiện va chạm bằng pixel.

Để điều khiển xe tăng một cách thông minh nhằm tạo hứng thú cho người chơi chúng tôi sẽ sử dụng 2 chiến thuật chính là: chọn bước đi tiếp theo cho xe tăng sao cho tiến đến gần người chơi nhất, hoặc chọn bước đi tiếp theo cho xe tăng sao cho tiến gần đến Đại bàng nhất.

# 2. Prototype

Prototype trên giao diện mobile

<https://www.fluidui.com/editor/live/preview/p_VubtbWJAWlABblrTEawMKzzn7bHkLgoC.1412750016567>



# 3. Thuật toán

## 3.1 Phát hiện va chạm

Mỗi một đối tượng trong game (xe tăng, tường, gạch, player,…) đều được thể hiện bằng một hình vẽ gồm nhiều pixel. Chỉ cần 1 pixel nào đó ở đối tượng A chồng lấn lên miền pixel của đối tượng B thì sự kiện va chạm xảy ra.

Trong AndEngine có sẵn một extension cho phép phát hiện va chạm theo mô tả như trên.

Extension: <https://github.com/MakersF/AndEngineCollisionsExtension>

Demo : <https://www.youtube.com/watch?v=abbXURuDaTo>

## 3.2 Điều khiển xe tăng computer

### 3.2.1.Bài toán:

Có 1 bản đồ và tank computer đang ở 1 ô cho trước, tìm đường di chuyển sao cho tank computer di chuyển về phía đại bàng

### 3.2.2.Ý tưởng:

Liệt kê tất cả những ô tank có thể đi vào, tính khoảng cách đường chim bay của mỗi ô đó tới đại bàng (đường thẳng nối trực tiếp), chọn ô có khoảng cách ngắn nhất (random nếu bằng nhau), di chuyển vào ô đó và lặp lại vòng lặp cho đến khi thỏa điều kiện nhìn thấy ô gạch bao xung quanh đại bàng

### 3.2.3. Mô tả thuật toán

Input: tọa độ hiện tại của tank computer.

Output: tọa độ của ô kế tiếp mà tank nên di chuyển

FindCell(currentLocation)

{

<List> list;

FindAvailableCell(currentLocation , listcell); //hàm tìm ô có thể đi và add vào list

return FindBestCell(currentLocation, listcell); // hà m trả về ô ngắn nhất

}

### 3.2.4. Độ phức tạp của thuật toán:

a) trường hợp tốt nhất: chỉ có 1 ô tank có thể đi

Ttốt(n) = O(1)

b) trường hợp xấu nhất: có 4 ô xung quanh tank có thể đi

T(n) = O(4)

### 3.2.5. Cài đặt

Cell FindCell(currentCell)

{

<List> list;

FindAvailableCell(currentCell, list); //hàm tìm ô có thể đi và add vào list

return FindBestCell(currentCell, list); // hàm trả về ô ngắn nhất

}

Hàm tìm ô có thể di chuyển:

list FindAvailableCell(list)

{

Foreach (cells in currentCell.getAroundCell())

{

If (cells.isMoveAble) list.Add(cells)

}

Return list;

}

Hàm tìm ô tốt nhất:

Cell FindBestCell(currentCell,list)

{

Cell cellmin = new Cell(left,top);

Foreach(cells in list)

{

If (cellmin.getdis() < cells.getdis())

cellmin = cells;

}

return cellmin;

}

Hàm lấy khoảng cách:

Int Cell::getdis()

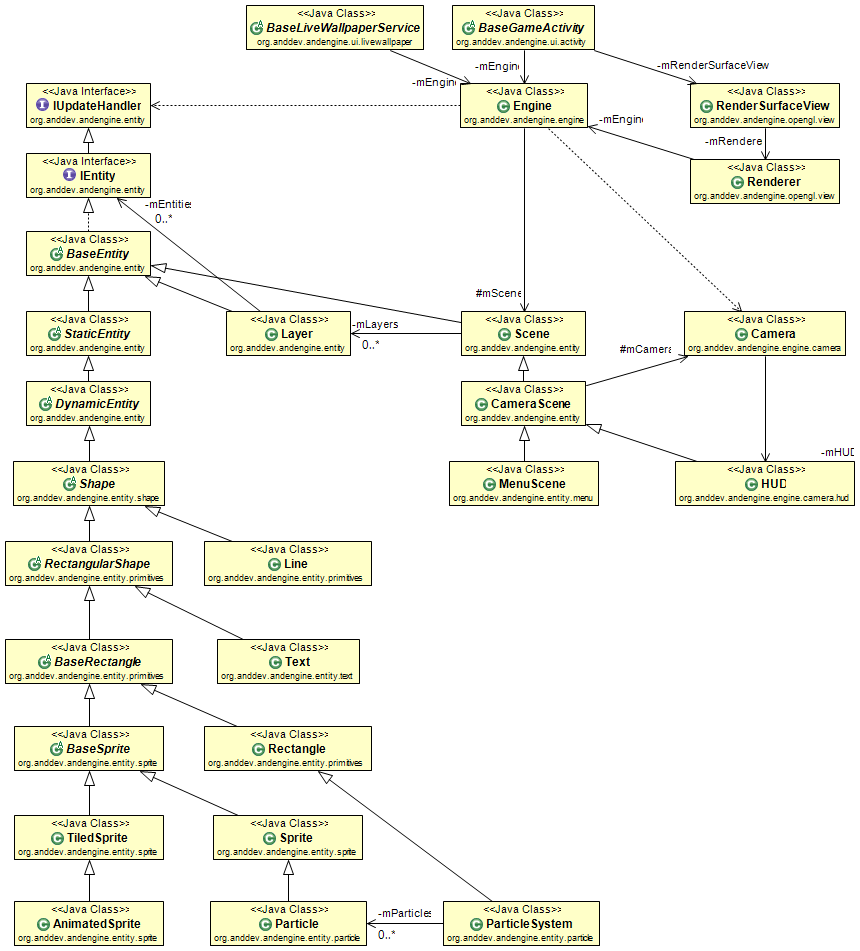
{

return distant = Math.sqrt(Math.pow(cells.x – FALCON\_X, 2) + Math.pow(cell.y – FALCON\_Y))

}

# 4. Cấu trúc AndEngine

## 4.1 Sơ đồ cấu trúc của AndEngine



## 4.2 Mô tả

Sau đây là bảng mô tả các lớp và Interface trọng tâm

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Tên class/interface | Mô tả | Chức năng chính |
| 1 | IupdateHandler | Interface này quy đình các phương thức cập nhật game | BaseEntity cài đặt Interface này để nhận được sự kiện update game |
| 2 | IEntity | Interface quy định các phương thức của một thực thể game |  |
| 3 | Scene | Lớp này quản lý các thực thể trong game |  |
| 4 | Camera | Lớp này liên quan đến thuật ngữ Camera trong game |  |
| 5 | BaseGameActivity | Đây là lớp cơ sơ quy định sẵn các hàm callback | Ứng dụng game sẽ kế thừa lớp này để làm Activity cho game |
| 6 | Engine | Lớp này khởi tạo các đối tượng quản lý, thiết lập options, thiết lập camera, | Điều phối luồng chạy của game |
| 7 | HUD | Lớp này chưa các thực thể mà không bị ảnh hưởng khi camera thay đổi |  |
| 8 | Sprite | Lớp này chứa các thông tin về hiển thị của một đối tượng |  |