

**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH**  
**ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**  
**KHOA CÔNG NGHỆ PHẦN MỀM**



## **BÁO CÁO ĐỒ ÁN CUỐI KỲ**

**MÔN: NHẬP MÔN PHÁT TRIỂN GAME**

**Đề tài: Xây dựng game bắn xe tank tương tự Battle City**

**Giảng viên hướng dẫn: ThS. Nguyễn Phương Anh**

**Lớp: SE102.G13**

**Sinh viên thực hiện:**

- |                         |          |
|-------------------------|----------|
| ➤ Văn Trương Quốc Thắng | 13520776 |
| ➤ Nguyễn Quang Nghĩa    | 13520540 |
| ➤ Nguyễn Văn Nguyễn     | 13520567 |
| ➤ Tô Thành Thương       | 13520862 |

***TP. Hồ Chí Minh, ngày 3 tháng 1 năm 2016***

## **LỜI CẢM ƠN**

Đầu tiên, nhóm xin chân thành gửi lời cảm ơn đến tập thể quý Thầy, Cô Trường Đại Học Công nghệ thông tin – Đại Học Quốc Gia Thành phố Hồ Chí Minh.

Đặc biệt, nhóm xin gửi lời cảm ơn và lòng biết ơn với thầy Nguyễn Phương Anh (Giảng viên Lý thuyết và Thực hành môn Nhập môn Phát triển Game). Thầy đã trực tiếp hướng dẫn, giải đáp các thắc mắc của nhóm trong quá trình thực hiện đồ án.

Trong thời gian một học kỳ thực hiện đề tài, nhóm đã vận dụng những kiến thức nền tảng đã tích lũy và phối hợp với việc tự học hỏi, nghiên cứu những kiến thức mới. Từ đó, nhóm đã hoàn thành được đồ án. Trong quá trình thực hiện, nhóm không thể tránh khỏi những thiếu sót. Chính vì vậy, nhóm mong được những ý kiến đóng góp từ phía Giảng viên nhằm hoàn thiện những thiếu sót để hoàn thiện cho hành trang của nhóm trong tương lai.

Xin chân thành cảm ơn!

## This image shows a full page of white paper with horizontal dotted lines. The lines are evenly spaced and run across the width of the page, providing a guide for handwriting practice. There are no margins, text, or other markings on the page.

## Mục lục

<b>I. GIỚI THIỆU SƠ LƯỢC VỀ GAME</b> .....	5
1. Game Concept: .....	5
2. Hướng dẫn chơi: .....	6
<b>II. THIẾT KẾ GAME</b> .....	6
1. Hệ thống các nhân vật trong game: .....	6
2. Hệ thống item trong game .....	7
3. Các trạng thái game .....	8
4. Các màn chơi .....	12
<b>III. HIỆN THỰC GAME</b> .....	14
1. Các lớp đối tượng tương tác chính .....	14
2. Các lớp quản lý các thành phần khác trong game .....	15
3. Chi tiết về các lớp trong Game: .....	16
<b>IV. CÁC KỸ THUẬT ÁP DỤNG TRONG GAME</b> .....	54
1. Thiết kế các lớp đối tượng theo singleton pattern: .....	54
2. Vẽ các đối tượng động có animation: .....	54
3. Xét va chạm trong game .....	55
4. Thiết kế AI cho các xe tank địch: .....	58
5. Xử lý âm thanh trong game .....	59
6. Chuyển đổi file map tạo được từ phần mềm Tiled .....	62
<b>V. ĐÁNH GIÁ MỨC ĐỘ HOÀN THIỆN CỦA GAME:</b> .....	64
<b>VI. KẾT LUẬN CHUNG</b> .....	65

## I. GIỚI THIỆU SƠ LƯỢC VỀ GAME

### 1. Game Concept:



- Game có nội dung tương tự:
  - Tank Battalion
  - Tank Force
- Game thuộc thể loại arcade
- Người chơi điều khiển một chiếc xe tank có sức mạnh tiềm ẩn với khả năng nâng cấp mở rộng với số mạng nhất định.
- Khi mới vào màn chơi xe tank người chơi sẽ xuất hiện ở bên trái căn cứ
- Quân địch sẽ xuất hiện ở ba vị trí góc trái-trên, giữa-trên và phải-trên của chiến trường.
- Người chơi có nhiệm vụ bảo vệ căn cứ khỏi sự tấn công của xe tank địch bằng cách tiêu diệt hết tất cả xe tank địch
- Người chơi có thể điều khiển xe tank di chuyển và bắn theo 4 hướng.
- Khi xe tank người chơi bị trúng đạn sẽ giảm số mạng còn lại. Nếu hết mạng game sẽ tự động kết thúc.
- Đạn có thể phá hủy gạch và tường thép( nếu đạn đạt cấp độ cao nhất).
- Quân đoàn xe tank địch gồm có 20 xe tank mỗi đợt tiến công địch sẽ tung ra 4 tank trên chiến trường. Xe tank địch gồm 4 chủng loại với các đặc điểm sức mạnh khác nhau.

- Sẽ có 3 xe tank địch có mang các item đặc biệt giúp ích cho người chơi trong mỗi trận chiến.
- Xe tank của người chơi khi thu thập được item sẽ giúp nâng cấp xe tank mạnh hơn, tạo lá chắn bảo vệ, làm kẻ địch bị đứng yên hay tăng thêm mạng.
- Người chơi sẽ được cộng điểm khi tiêu diệt tank địch tùy theo loại tank địch là gì và khi thu thập được item.
- Khi người chơi vượt qua tất cả màn chơi sẽ có thông báo và ghi nhận lại số điểm của người chơi.

## **2. Hướng dẫn chơi:**

- Người chơi dùng 4 phím mũi tên để di chuyển và phím space để bắn.
- Cố gắng tiêu diệt hết tank địch đồng thời bảo vệ căn cứ không để bị phá hủy. Nếu vượt qua hết 3 màn chơi, người chơi sẽ hoàn thành nhiệm vụ. Nếu hết mạng hoặc căn cứ bị phá hủy, người chơi sẽ thua cuộc.
- Lợi dụng các vật cản để tránh đạn của xe tank địch đồng thời thu thập các item hỗ trợ để được tăng điểm và các hỗ trợ tương ứng với từng loại item

## **II. THIẾT KẾ GAME**

### **1. Hệ thống các nhân vật trong game:**





#### **1.1 Xe tank do người chơi điều khiển (PlayerTank):**



- Người chơi được điều khiển một chiếc xe tank có sức mạnh tầm ẩn có khả năng nâng cấp mở rộng, khi người chơi thu thập các item thì xe tank sẽ được nâng cấp hay có các trang bị đặc biệt hỗ trợ người chơi.

#### **1.2 Các loại tank địch (Enemy tank):**

- Trong game người chơi sẽ đối đầu với 4 loại xe tank địch và chúng có các đặc điểm riêng:




- Medium tank:  Loại xe tank địch có sức mạnh yếu nhất game, với tốc độ di chuyển và khả năng tấn công hạn chế, người chơi có thể dễ dàng tiêu diệt chúng mà không tốn nhiều công sức.
- Light tank:  Loại xe tank địch có tốc độ di chuyển nhanh nhất trong game chính vì tốc độ di chuyển nhanh nên sẽ gây khó khăn cho người chơi khi muốn tiêu diệt chúng
- Heavy tank:  Loại xe tank địch có tốc độ di chuyển chậm nhưng chúng có khả năng tấn công tốt nhất game vì có tốc độ đạn nhanh người chơi sẽ khó khăn khi tấn công mặt đối mặt với loại tank này.
- Super Heavy Tank:  Loại tank địch nguy hiểm nhất game, chúng có lớp giáp cực tốt để hạ được tank này người chơi phải cần đến 3 phát đạn, với lớp giáp dày của nó người chơi không nên đối mặt trực tiếp khi xe tank chưa được nâng cấp hay có lá chắn bảo vệ.


## 2. Hệ thống item trong game

### 2.1 Item star:



- Khi người chơi thu thập được item này thì xe tank của người chơi sẽ được nâng cấp sức mạnh, với khởi đầu game người chơi ở level 1 sau khi thu thập item này sẽ có khả năng nâng cấp như sau:

- Level 1:  tốc độ bay của đạn và tốc độ bắn của tank ở mức cơ bản
- Level 2:  tốc độ bay của đạn nhanh hơn và tốc độ bắn của tank ở mức cơ bản
- Level 3:  Tốc độ bay của đạn như level 2 và có khả năng bắn 2 viên đạn vào cùng 1 loạt bắn

- Level 4:  Tốc độ bay của đạn như level 3, có khả năng bắn 2 viên đạn vào cùng 1 loạt bắn và lúc này đạn có khả năng xuyên phá tường thép.

## 2.2 Item shield:



- Khi người chơi thu thập được item này thì xe tank của người chơi sẽ có một lá chắn bảo vệ tồn tại trong 1 khoảng thời gian ngắn, xe tank người chơi khi bị trúng đạn của xe tank địch sẽ không bị phá hủy trong khi lá chắn còn tồn tại.

## 2.3 Item freeze:



- Khi người chơi thu thập item này toàn bộ xe tank địch xuất hiện trên bản đồ sẽ bị ngừng hoạt động trong một khoảng thời gian, người chơi có thể thoải mái tiêu diệt chúng mà không lo bị tấn công.

## 2.4 Item grenade:



- Khi người chơi thu thập được item này thì tất cả xe tank địch xuất hiện trên bản đồ lúc này sẽ bị phá hủy hoàn toàn.

## 2.5 Item tank:



- Khi người chơi thu thập được item này thì sẽ được tăng thêm 1 lượt hồi sinh.

# 3. Các trạng thái game

## 3.1 Menu chính

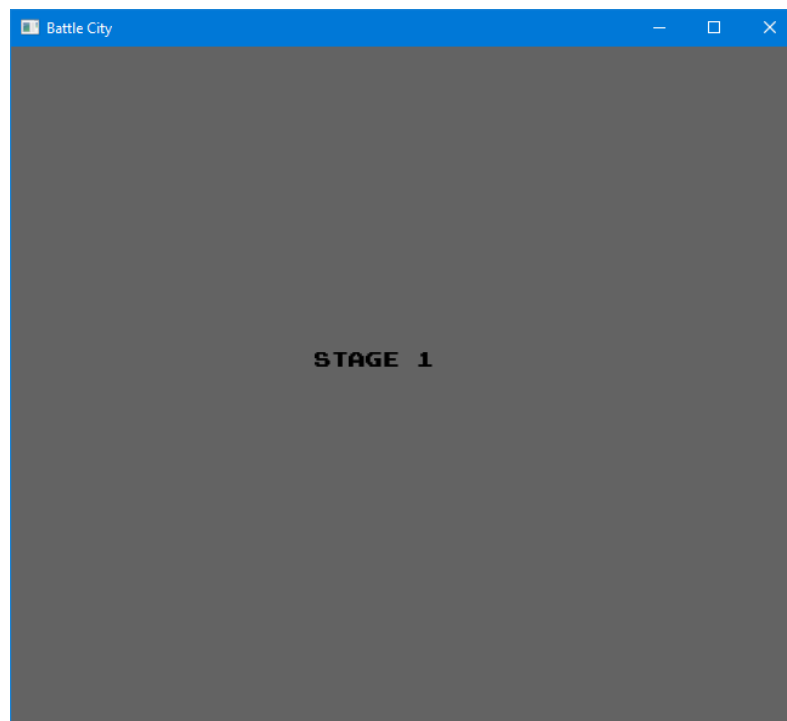




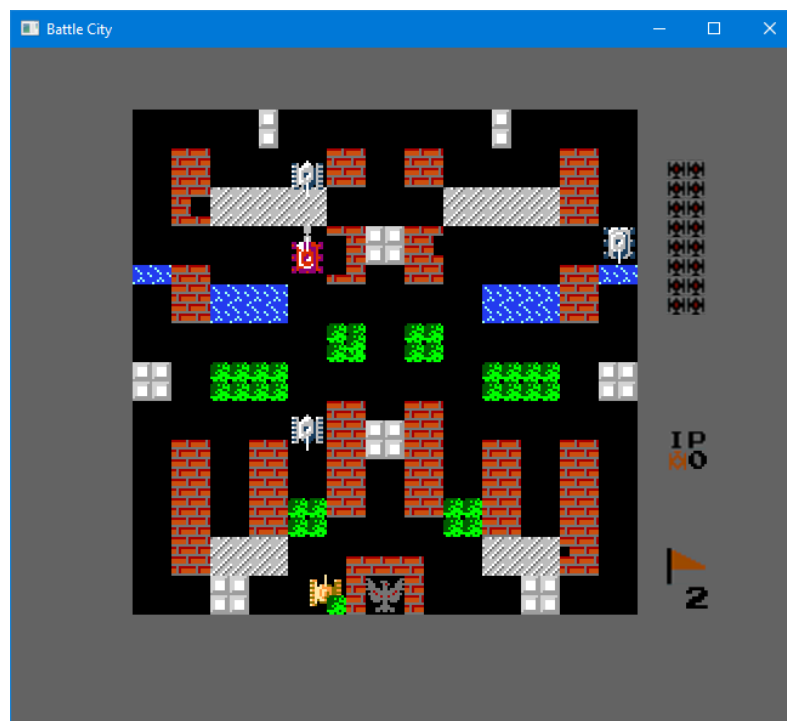
### 3.2 Phần hướng dẫn chơi



### 3.3 Chuẩn bị vào phần chơi chính



### 3.4 Trạng thái chính: điều khiển tank bảo vệ căn cứ



### 3.5 Tính điểm



### 3.6 Thua cuộc



### 3.7 Hoàn thành nhiệm vụ

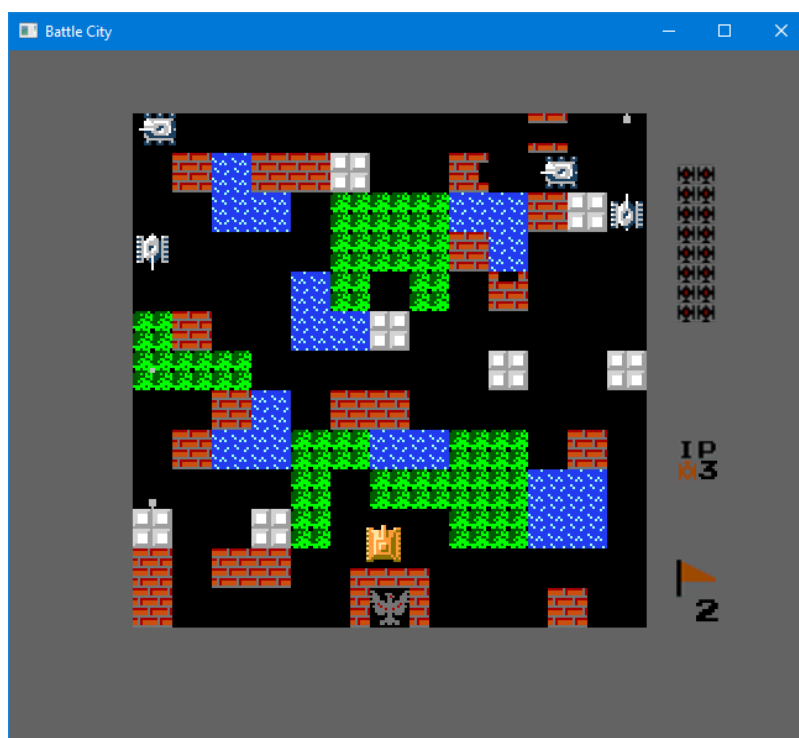


## 4. Các màn chơi

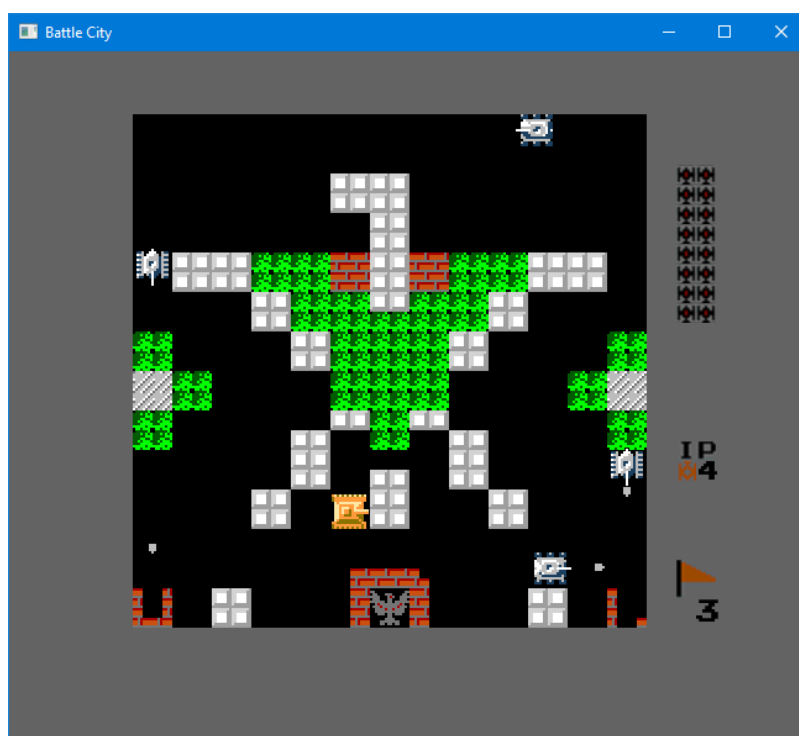
### 4.1 Màn 1



## 4.2 Màn 2

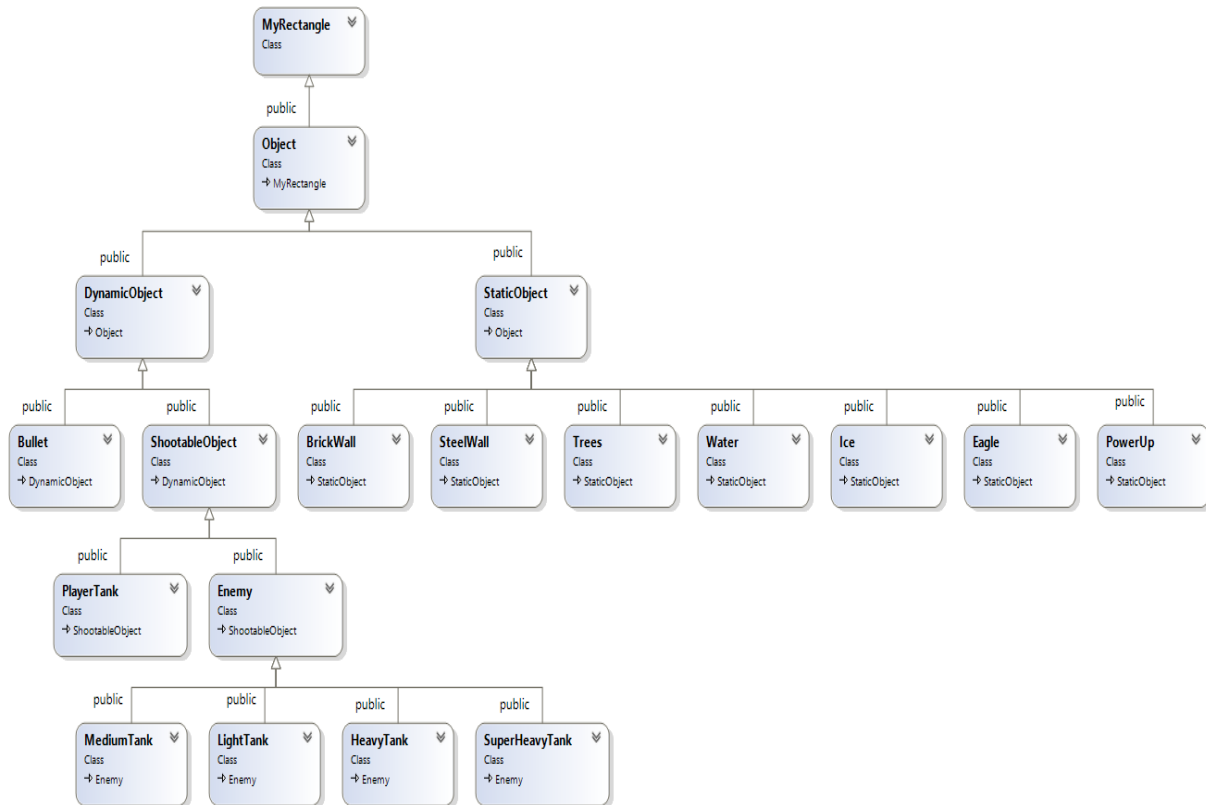


## 4.3 Màn 3



### III. HIỆN THỰC GAME

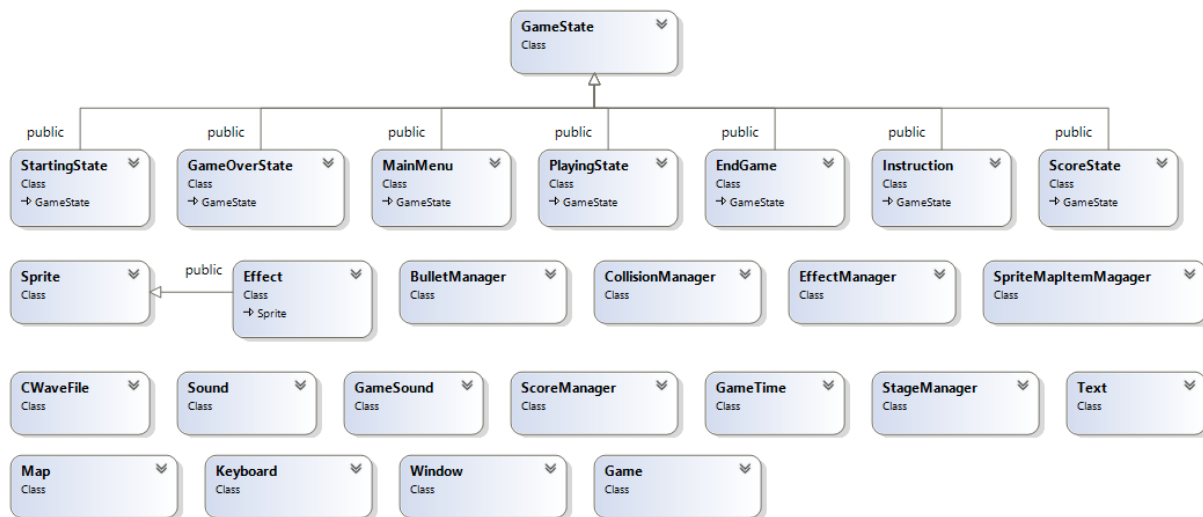
#### 1. Các lớp đối tượng tương tác chính



- Nguyên tắc thiết kế các lớp đối tượng chính mà người chơi sẽ tương tác trong game:
  - Tất cả các đối tượng mà người chơi tương tác trực tiếp trong game đều có một hình chữ nhật bao lấy phần diện tích thể hiện trên màn hình và dùng để kiểm tra xử lý va chạm do đó ta xây dựng lên lớp cơ sở là lớp MyRectangle dùng để chứa thông tin về hình chữ nhật bao lấy đối tượng và các phương thức lấy kích thước hay tọa độ.
  - Tất cả các đối tượng ta cần quản lý để biết đối tượng thuộc nhóm đối tượng nào (động hay tĩnh) và loại đối tượng là gì (xe tank, gạch, nước...) thì ta cần định nghĩa ra một lớp trừu tượng Object để quản lý các đối tượng này và có các phương thức trừu tượng để thực hiện thao tác vẽ và cập nhật các hành vi, trạng thái cho từng loại đối tượng.

- Trong game sẽ có 2 nhóm đối tượng chính: 1 nhóm đối tượng tĩnh không có khả năng di chuyển (ví dụ: gạch, tường thép, nước,...) và 1 nhóm đối tượng động nghĩa là có khả năng di chuyển từ 2 nhóm đối tượng trên ta xây dựng 2 lớp DynamicObject (quản lý các đối tượng động) và StaticObject (quản lý các đối tượng tĩnh). Hai lớp này kế thừa lại lớp Object đã định nghĩa ở trên.
- Từ lớp DynamicObject ta định nghĩa ra các lớp đối tượng động kế thừa lại từ lớp này (PlayerTank, Enemy, LightTank,...)
- Trong gameplay sẽ có nhóm các đối tượng động là xe tank địch mà người chơi có nhiệm vụ phải tiêu diệt hết, do đó để thuận tiện cho việc quản lý và tạo ra các phương thức xử lý AI, gắn item cho tank và va chạm trong game ta tạo ra lớp Enemy để các loại xe tank địch cụ thể sẽ kế thừa lại lớp này.
- Từ lớp StaticObject ta xây dựng các lớp là các đối tượng tĩnh có mặt trong game mà người chơi trực tiếp tương tác như gạch(BrickWall), tường thép(SteelWall), nước(Water)...

## 2. Các lớp quản lý các thành phần khác trong game

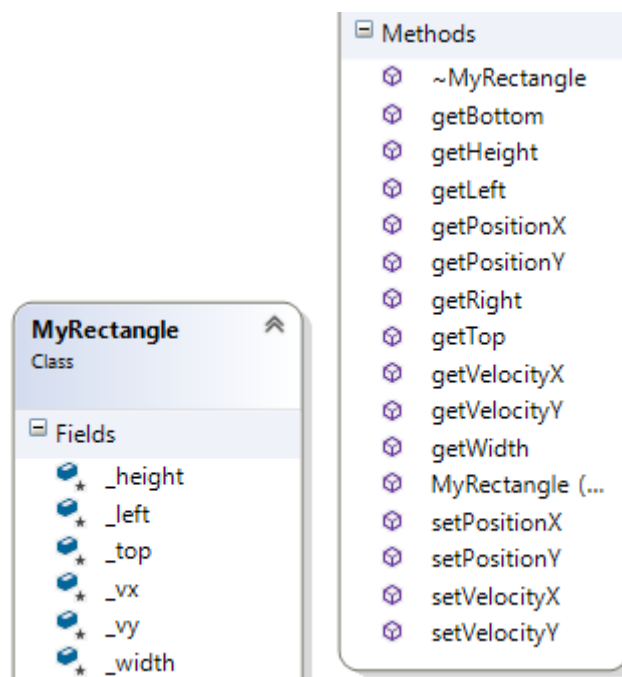


- Lớp GameState và các lớp kế thừa: Quản lý các trạng thái của game
- Lớp Sprite: Tạo sprite ảnh cho đối tượng
- Lớp Effect: Tạo hiệu ứng

- Lớp EffectManager: Quản lí hiệu ứng
- Lớp BulletManager: Quản lí đạn
- Lớp CollisionManager: Quản lí va chạm
- Lớp SpriteMapItemManager: Quản lí sprite của các đối tượng tĩnh trên map
- Lớp CwaveFile: Hỗ trợ đọc file wav
- Lớp Sound: Tạo âm thanh cho game
- Lớp GameSound: Quản lí âm thanh
- Lớp ScoreManager: Quản lí điểm người chơi
- Lớp GameTime: Quản lí thời gian lặp game
- Lớp StageManager: Quản lí màn chơi (stage)
- Lớp Text: Hỗ trợ vẽ chữ lên màn hình
- Lớp Map: Quản lí tất cả các đối tượng trên bản đồ game
- Lớp KeyBoard: Quản lí dữ liệu người chơi nhập từ bàn phím
- Lớp Window: Hỗ trợ tạo cửa sổ game
- Lớp Game: Quản lí game.

### 3. Chi tiết về các lớp trong Game:

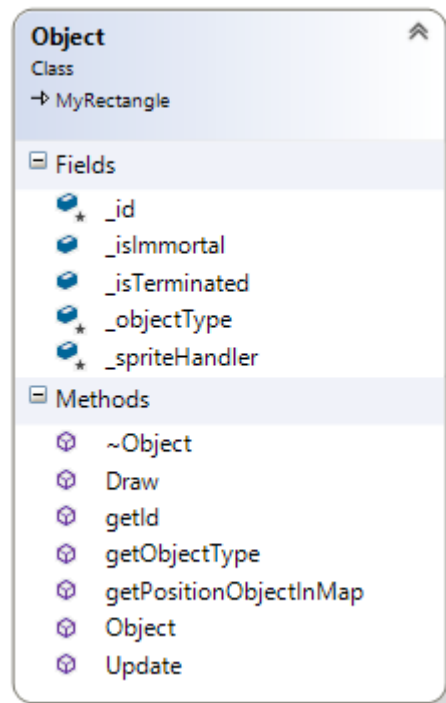
#### 3.1 Lớp MyRectangle:





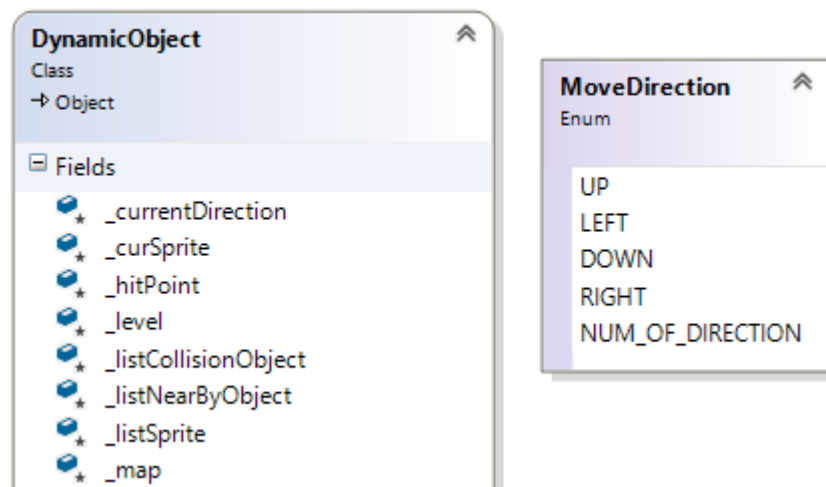
- Đây là lớp cơ sở cho mọi đối tượng mà người chơi tương tác kế thừa lại, lớp này có các thuộc tính quản lý kích thước của hình chữ nhật bao quanh đối tượng trong game và vận tốc (mặc định những đối tượng tĩnh sẽ có vận tốc là 0), đi kèm là các phương thức để lấy các giá trị từ thuộc tính.
- Các thuộc tính chính:
  - \_width, height: độ dài chiều ngang và dọc của đối tượng
  - \_left, \_top: tọa độ x và y của đối tượng
  - \_vx, \_vy: vận tốc theo trục x và y
- Các phương thức chính:
  - getTop(), getBottom(), getLeft(), getRight(): Lấy tọa độ các cạnh trong không gian
  - getWidth(), getHeight(): lấy kích thước đối tượng
  - getVelocityX(), getVelocityY(): lấy vận tốc của vật theo trục x và y
  - setPositionX(int), setPositionY(int): set tọa độ \_left, \_top cho đối tượng
  - setVelocityX(int), setVelocityY(int): set vận tốc theo trục x và y cho đối tượng

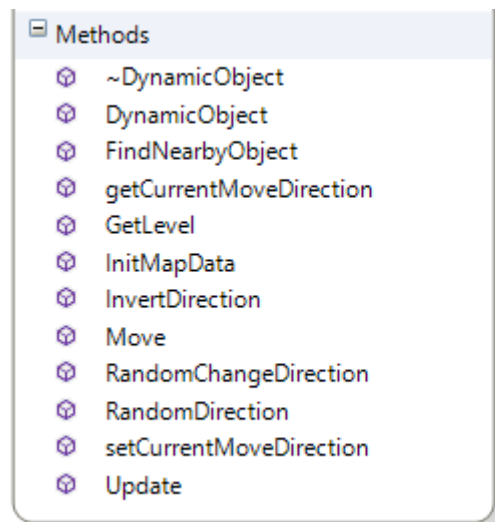
### 3.2 Lớp Object:



- Chứa các thuộc tính định danh cho đối tượng (loại đối tượng và kiểu đối tượng là động hay tĩnh) và các thuộc tính chỉ trạng thái của đối tượng như đối tượng đã bị tiêu diệt(`_isTerminated`), đối tượng ở trạng thái bất tử (`_isImmortal`)...
- Các thuộc tính chính:
  - `_id`: mã định danh của loại đối tượng
  - `_isImmortal`: trạng thái bất tử của object
  - `_isTerminated`: trạng thái đã bị tiêu diệt
  - `_objectType`: loại đối tượng động hay tĩnh
  - `_spriteHandler`: đối tượng vẽ object
- Các phương thức chính:
  - `Draw()`: phương thức trờu tượng để vẽ một đối tượng
  - `getId()`: lấy mã định danh của đối tượng
  - `getObjectType()`: lấy loại đối tượng
  - `getPositionObjectInMap()`: lấy ra tọa độ của đối tượng theo (row, column) trong ma trận map
  - `Update()`: cập nhật các hành vi, trạng thái của đối tượng

### 3.3 Lớp **DynamicObject**:





- Lớp quản lý các đối tượng động với các thuộc tính chỉ hướng di chuyển, máu, cấp của đối tượng, các thuộc tính tham chiếu đến ma trận bản đồ, list các đối tượng xung quanh object và list toàn bộ các đối tượng tĩnh trên màn chơi để tiến hành xét va chạm. Các phương thức để vẽ đối tượng, di chuyển đối tượng, cập nhật các trạng thái của đối tượng khi di chuyển và khi xảy ra va chạm, các phương thức xử lý AI cho đối tượng, tìm các đối tượng xung quanh để xét va chạm và khởi gán thông tin của màn chơi cho đối tượng.
- Các thuộc tính chính:
  - `_currentDirection`: hướng di chuyển hiện tại của đối tượng
  - `_curSprite`: sprite cần vẽ hiện tại trong list sprite các hướng
  - `_hitPoint`: máu của đối tượng
  - `_level`: cấp độ của đối tượng
  - `_listCollisionObject`: list tham chiếu toàn bộ các đối tượng tĩnh có thể xét va chạm trong bản đồ
  - `_listNearbyObject`: list các đối tượng tĩnh xung quanh đối tượng động để xét va chạm
  - `_listSprite`: list các sprite ứng với các hướng di chuyển
  - `_map`: tham chiếu đến ma trận Map
- Các phương thức chính:

FindNearByObject(): tìm các đối tượng tĩnh xung quanh đối tượng động để xét va chạm dựa trên vị trí (x,y) của đối tượng động quy ra tọa độ theo (dòng,cột) trong ma trận.

getCurrentMoveDirection(): lấy hướng di chuyển hiện tại

GetLevel: lấy level của đối tượng

InitMapData(): Gán tham chiếu list tất cả đối tượng tĩnh có thể xét va chạm trong bản đồ và ma trận Map cho đối tượng động để lấy đối tượng xét va chạm.

InvertDirection(): Đảo hướng di chuyển

Move(): Phương thức trừu tượng để di chuyển đối tượng

RandomChangeDirection(): Sinh ngẫu nhiên hướng di chuyển không trùng hướng hiện tại

RandomDirection(): Sinh ngẫu nhiên hướng di chuyển

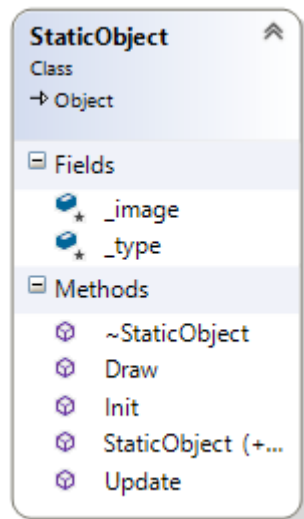
setCurrentMoveDirection(): gán hướng di chuyển cho đối tượng

Update(): Cập nhật hành vi và trạng thái đối tượng

- Các bước cập nhật trạng thái của đối tượng động

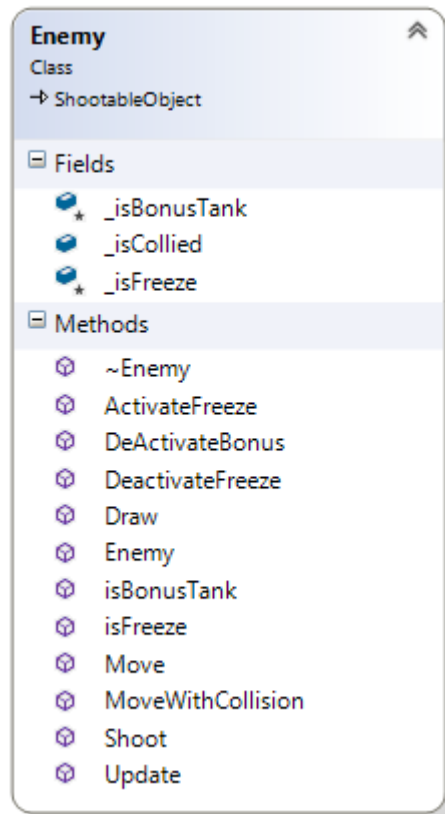


### 3.4 Lớp StaticObject:



- Dùng để quản lý các đối tượng tĩnh bao gồm thuộc tính định danh loại hình vẽ cho loại đối tượng( ví dụ: đối tượng gạch sẽ có 4 hình vẽ khác nhau nên ta cần có 1 biến chỉ rõ 4 loại này) và 1 biến sprite để chứa hình ảnh cần vẽ. Các phương thức khởi tạo hình ảnh lấy từ SpriteManager, các phương thức vẽ đối tượng, cập nhật đối tượng.
- Các thuộc tính chính:
  - \_image: sprite của đối tượng
  - \_type: index của đối tượng trên sprite ảnh
- Các phương thức chính:
  - Draw(): vẽ đối tượng lên màn hình
  - Update(): cập nhật trạng thái của đối tượng
  - Init(): khởi tạo giá trị ban đầu cho đối tượng (sử dụng khi khởi tạo đối tượng bằng constructor mặc định)

### 3.5 Lớp Enemy:



- Lớp quản lý các xe tank địch trong game, lớp gồm có các thuộc tính chỉ ra xe tank địch có mang Item không, xe tank địch có xảy ra va chạm không hay có đang ở trạng thái đóng băng không. Các phương thức vẽ đối tượng, cập nhật, bắn, di chuyển, và kích hoạt hoặc hủy kích hoạt các trạng thái đặc biệt như trạng thái đóng băng, phương thức phát sinh ngẫu nhiên hướng di chuyển để tạo AI
- Các thuộc tính chính:
  - \_isBonusTank: xác định tank này có mang item không
  - \_isCollided: có xảy ra va chạm không
  - \_isFreeze: có bị đóng băng không
- Các phương thức chính:
  - ActivateFreeze(), DeactivateFreeze(): kích hoạt và hủy kích hoạt trạng thái bị đóng băng cho tank địch
  - isFreeze(): kiểm tra xem có đang ở trạng thái đóng băng không

Draw(): vẽ đối tượng

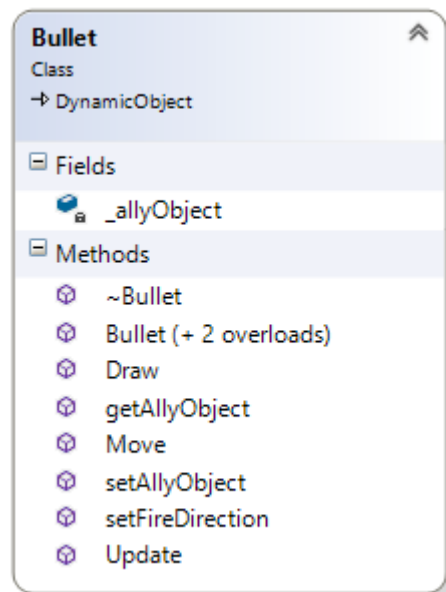
Move(): xử lý di chuyển

MoveWithCollision(): Xử lý di chuyển khi có va chạm

Shoot(): Xử lý bắn

Update(): Cập nhật trạng thái, hành vi

### 3.6 Lớp Bullet:



- Trong game các xe tank do người chơi và địch đều phải có đạn để có thể chiến đấu với nhau nên ta cần xây dựng một lớp đạn kế thừa lại lớp DynamicObject (vì đạn cũng là một đối tượng động) lớp này ngoài có các thuộc tính được kế thừa từ lớp cơ sở ta cần bổ sung một thuộc tính để phân biệt được đạn này do tank của người chơi hay của địch bắn ra dựa trên thuộc tính này mà ta có cách xét va chạm phù hợp. Về các phương thức cũng giống như các đối tượng động khác cũng có các phương thức vẽ, di chuyển, cập nhật trạng thái.
- Thuộc tính chính:  
\_allyObject: xác định đạn này do player hay enemy bắn
- Các phương thức chính:  
Draw(): Vẽ viên đạn  
getAllyObject: lấy ra giá trị đạn do đối tượng nào bắn ra

setAllyObject: gán đối tượng bắn ra đạn

setFireDirection: gán hướng bắn

Update(): cập nhật trạng thái viên đạn

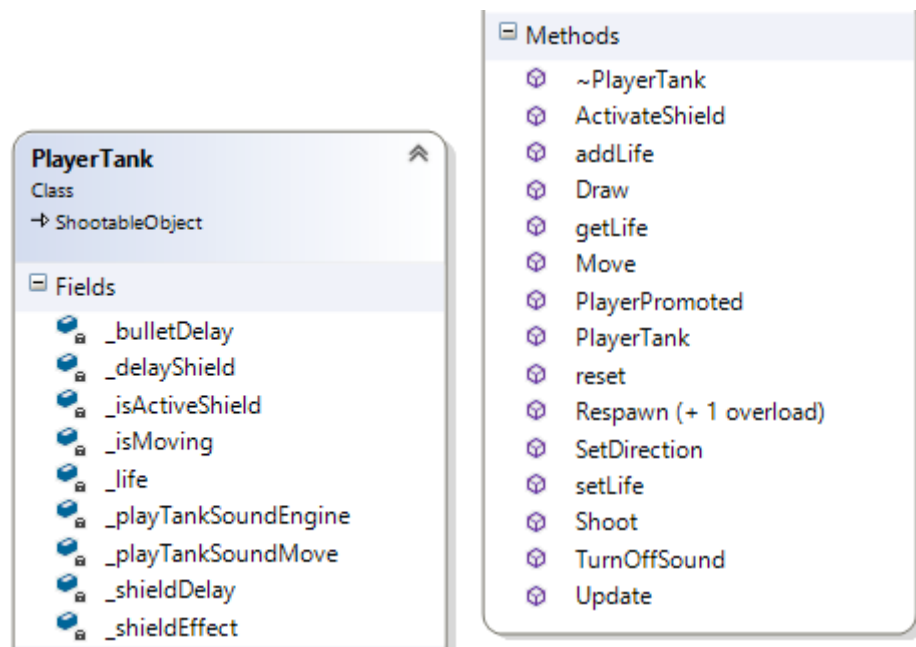
Move(): xử lý đạn di chuyển

DeactivateFreeze(): hủy trạng thái đứng yên

ActivateFreeze(): kích hoạt trạng thái đóng băng

DeActivateBonus(): hủy trạng thái mang item cho xe tank địch

### 3.7 Lớp PlayerTank:



- Đây là lớp tạo ra đối tượng xe tank mà người chơi trực tiếp điều khiển trong game. Ngoài các thuộc tính được kế thừa từ các lớp cơ sở thì lớp còn có thêm các thuộc tính để tính toán thời gian bắn, thời gian tồn tại của lá chắn, mạng và âm thanh. Các phương thức vẽ, di chuyển, bắn, cập nhật trạng thái và các phương thức xử lý khi xe tank của người chơi thu thập được item. Ngoài ra, để di chuyển được thì xe tank cho người chơi điều khiển còn kết hợp với nhận sự kiện bàn phím, nên trong phương thức di chuyển ta phải xử lý các phím do người chơi nhấn
- Các thuộc tính chính:



\_bulletDelay: khoảng thời gian giữa 2 lần bắn

\_delayShield: thời gian kích hoạt lá chắn

\_isActiveShield: có đang kích hoạt lá chắn không

\_isMoving: có đang di chuyển không

\_life: mạng của player

\_playerTankSoundMove, \_playerTankSoundEngine: âm thanh của tank lúc đứng yên hay đang di chuyển

\_shieldEffect: đối tượng để vẽ lá chắn

- Các phương thức chính:

ActivateShield(): Kích hoạt lá chắn

addLife(): Tăng thêm mạng

Draw(): Vẽ player tank

getLife(): lấy số mạng hiện tại

Move(): xử lý di chuyển

PlayerPromoted(): Tăng level và sức mạnh khi thu thập item star

reset(): trả trạng thái tank về ban đầu.

Respawn(): hồi sinh lại tank

SetDirection(): gán hướng di chuyển

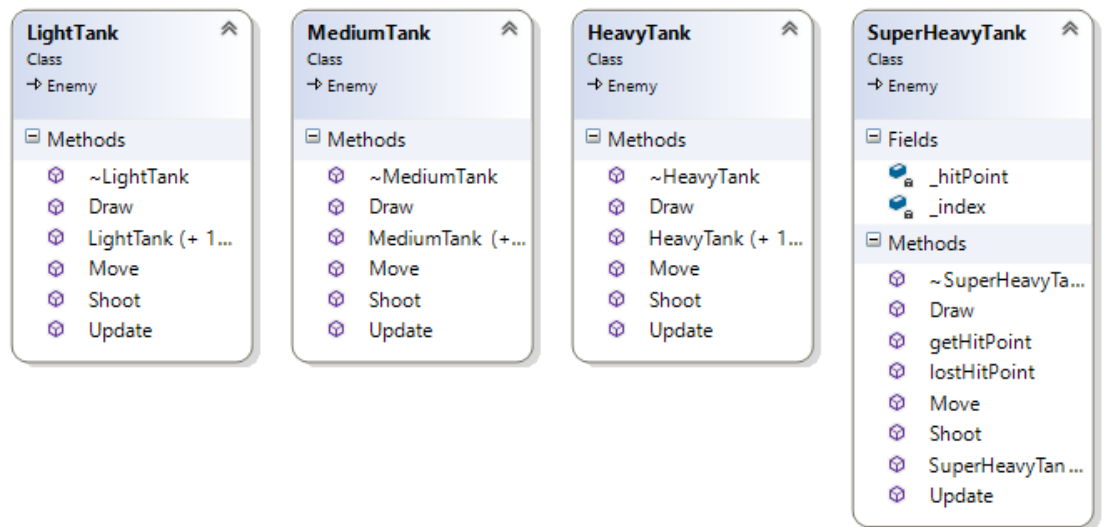
Shoot(): Xử lý bắn

TurnOffSound(): tắt âm thanh của tank

Update(): cập nhật xử lý hành vi, trạng thái

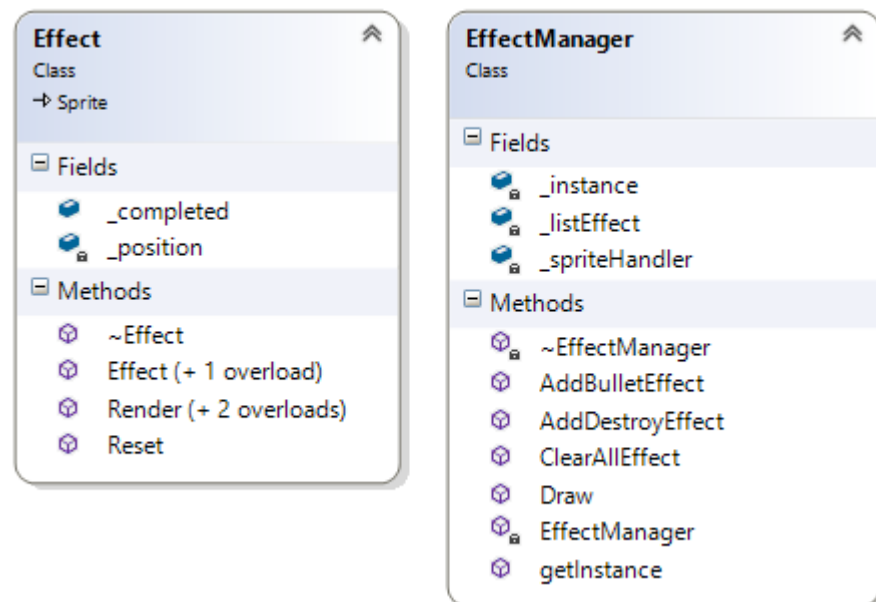
### 3.8 Nhóm các lớp phân loại xe tank địch

- Tank địch gồm có: LightTank, MediumTank, HeavyTank, SuperHeavyTank:



- Các lớp này có chung các thao tác vẽ, di chuyển, bắn và cập nhật trạng thái.  
Ngoài ra, lớp SuperHeavyTank là loại tank đặc thù người chơi phải dùng nhiều phát bắn trúng mới tiêu diệt nên sẽ có thêm thuộc tính để chỉ máu.

### 3.9 Các lớp quản lý hiệu ứng:



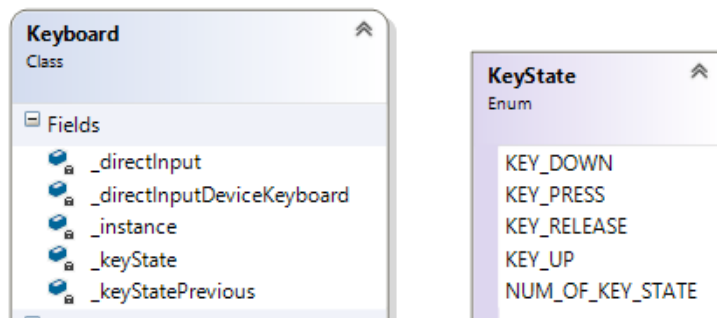
- Trong game khi có đối tượng bị tiêu diệt do bị trúng đạn, để làm cho game hấp dẫn hơn ta cần tạo các hiệu ứng cháy nổ. Để giải quyết vấn đề này, trong game ta cần xây dựng 2 lớp để tạo hiệu ứng là Effect và EffectManager:

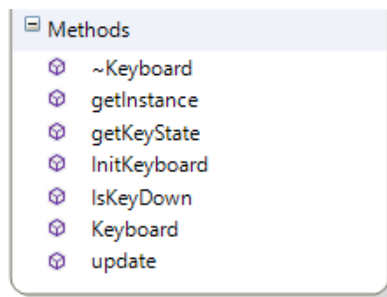
- Lớp Effect: Lớp này được kế thừa lại lớp Sprite để vẽ hình ảnh hiệu ứng lên màn hình, và ta bổ sung thêm hai thuộc tính để biết vị trí cần vẽ hiệu ứng và một biến để biết hiệu ứng đã vẽ xong tất cả các Frame chưa. Lớp có phương thức khởi tạo hiệu ứng, phương thức vẽ hiệu ứng và phương thức set index của sprite hiệu ứng về vị trí ban đầu.
  - Các thuộc tính chính:
    - `_completed`: biến bool để chỉ ra hiệu ứng này đã vẽ hết tất cả các frame chưa.
    - `_position`: tọa độ vẽ hiệu ứng trên màn hình
  - Các phương thức chính:
    - `Render()`: Dùng để vẽ hiệu ứng theo vị trí `_position` đã khởi tạo.
- Lớp EffectManager: Lớp được thiết kế theo singleton pattern mục đích để khởi tạo một thể hiện duy nhất tạo hiệu ứng cho toàn game và cũng như để tách thành phần hiệu ứng độc lập ra khỏi các đối tượng trong game để đơn giản hóa và tiết kiệm chi phí. Mỗi khi có hiệu ứng xảy ra ta chỉ cần thêm hiệu ứng vào 1 list các hiệu ứng cần vẽ ra bằng cách gọi hàm thêm hiệu ứng đạn nổ (`AddBulletExPlode`) hoặc hiệu ứng xe tank hay căn cứ bị nổ (`AddDestroyEffect`) ta chỉ việc gọi các hàm này tại bất kỳ vị trí nào trong chương trình để tạo hiệu ứng sau đó dùng thể hiện duy nhất của lớp này vẽ các hiệu ứng đó tại hàm `Draw()` trong vòng lặp game.
  - Các thuộc tính chính:
    - `_instance`: Thể hiện duy nhất của lớp khi cần tạo hiệu ứng thì sử dụng thể hiện này để tạo thêm hiệu ứng và lưu vào thuộc tính này.
    - `_listEffect`: Thuộc tính này có chức năng lưu trữ toàn bộ các hiệu ứng cần vẽ ra trên màn hình.
    - `_spriteHandler`: Đối tượng handle sprite dùng cho phương thức vẽ.

▪ Các phương thức chính:

- AddBulletEffect(): Phương thức dùng để thêm một hiệu ứng đạn nổ trong màn chơi
- AddDestroyEffect(): Phương thức dùng để thêm hiệu ứng khi có một xe tank hoặc căn cứ của người chơi bị phá hủy.
- Draw(): Phương thức vẽ tất cả các hiệu ứng được lưu trong thuộc tính \_listEffect trong hàm Draw() của vòng lặp game. Tại đây sẽ thực hiện vẽ hết tất cả các frame của từng hiệu ứng, sau khi vẽ hết (\_completed = true) thì hiệu ứng đó sẽ được xóa ra khỏi \_listEffect.
- ClearAllEffect(): Sau khi kết thúc màn chơi có thể có các hiệu ứng chưa thực hiện vẽ xong và GameState đã chuyển sang một trạng thái (state) mới do đó ta cần xóa hết các hiệu ứng này để tránh tình trạng khi sang màn chơi mới thì các hiệu ứng này được vẽ ra mà không có bất kỳ va chạm cần hiệu ứng nào xảy ra.
- getInstance(): Phương thức lấy thể hiện của lớp để tiến hành thêm hoặc vẽ hiệu ứng.

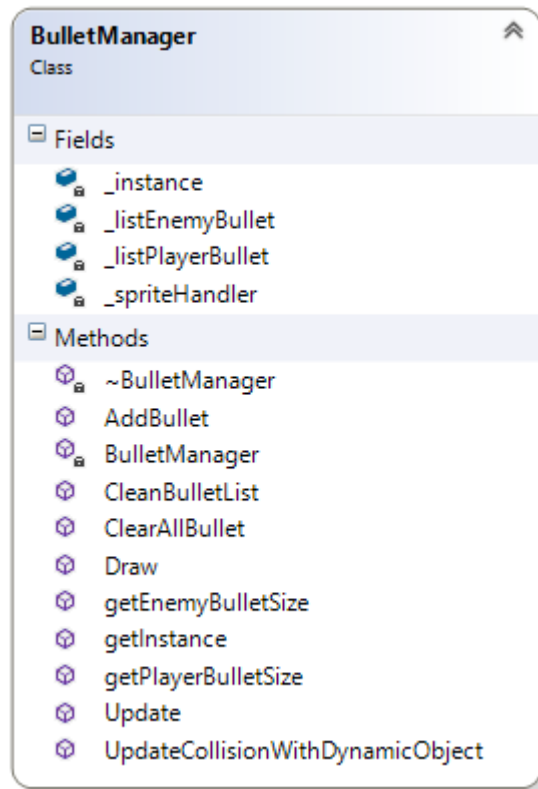
**3.10 Lớp quản lý bất sự kiện bàn phím từ người chơi (Keyboard):**





- Lớp này được xây dựng từ thư viện DirectInput (dinput.h) của DirectX hỗ trợ cho ta bắt sự kiện các phím và lớp này cũng thiết kế theo singleton pattern để tạo một thể hiện duy nhất sử dụng cho toàn game. Để lấy được sự kiện của phím ta sử dụng hàm update trong vòng lặp game để thu thập dữ liệu từ bàn phím sau đó ta sử dụng hàm IsKeyDown để kiểm tra xem phím nào người dùng đã nhấn và xử lý thích hợp khi người dùng nhấn đúng phím do game ta quy định.
- Các thuộc tính chính:
  - \_directInput: đối tượng đại diện cho thiết bị nhập của máy.
  - \_directInputDeviceKeyboard: Đối tượng đại diện cho thiết bị bàn phím.
  - \_instance: thể hiện duy nhất của lớp singleton pattern
  - \_keyState[256]: lưu giữ trạng thái của bàn phím
  - \_keyStatePrevious[256]: lưu giữ trạng thái của bàn phím trước thời điểm đang xét hiện tại
- Các phương thức chính:
  - getInstance(): lấy thể hiện của lớp để thao tác
  - getKeyState(): lấy trạng thái của bàn phím hiện tại
  - InitKeyboard(HINSTANCE hInstance, HWND hWnd): Khởi tạo thiết bị bàn phím cho game của chúng ta.
  - update(): Hàm cập nhật lại trạng thái của bàn phím

### 3.11 Lớp quản lý đạn trong game (BulletManager):



- Nếu như theo thiết kế thông thường ta sẽ thiết kế đạn là một thành phần (thuộc tính) của các đối tượng xe tank, tuy nhiên khi làm như vậy ta sẽ gặp khó khăn khi cập nhật các đối tượng vì theo nguyên tắc khi một xe tank địch bị tiêu diệt ta cần hủy vùng nhớ của đối tượng này để giúp tiết kiệm chi phí và thao tác này vô tình cũng hủy vùng nhớ viên đạn mà xe tank địch này bắn ra, trong khi theo logic thì dù xe tank có còn tồn tại hay không thì viên đạn đã bắn ra phải được cập nhật đến khi có va chạm thì mới hủy. Để giải quyết vấn đề này ta sẽ xây dựng ra lớp BulletManager để quản lý tất cả đạn do người chơi hay tank địch bắn ra và tiến hành cập nhật, vẽ, xét va chạm trong lớp này thể hiện của lớp này cũng được tạo duy nhất trong toàn chương trình và độc lập với các đối tượng xe tank. Trong lớp này sẽ có các thuộc tính là list đạn do player bắn ra, list đạn của xe tank địch bắn ra và các phương thức vẽ, cập nhật xét va chạm giữa đạn với các đối tượng, hủy các viên đạn đã va chạm.
- Các thuộc tính chính:

\_instance: thể hiện duy nhất của lớp

\_listEnemyBullet, \_listPlayerBullet: list đạn của enemy và player

\_spriteHandler: tương tự các lớp khác

- Các phương thức chính:

AddBullet(): thêm 1 viên đạn vào list tương ứng dựa trên AllyObject

CleanBulletList(): Xóa các viên đạn đã bị nổ trong list

ClearAllBullet(): Xóa toàn bộ đạn (dùng khi reset sang màn chơi mới)

Draw(): vẽ các list đạn

getEnemyBulletList(): lấy ra list đạn của enemy

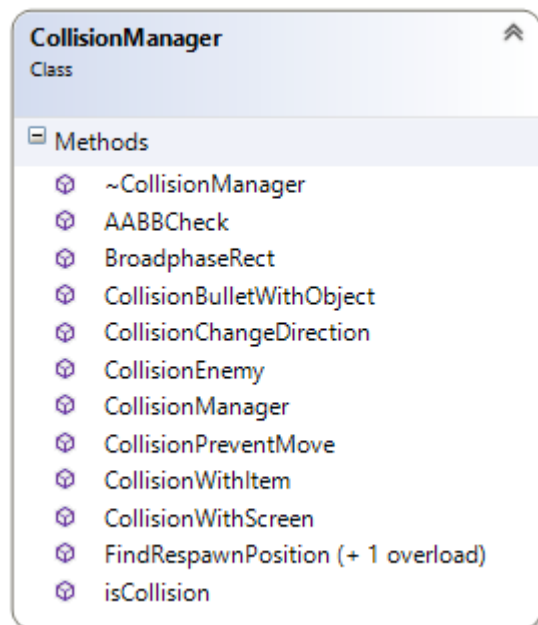
getInstance(): lấy thể hiện của lớp

getPlayerBulletList(): lấy ra list đạn của player

Update(): Cập nhật trạng thái của các list đạn

UpdateCollisionWithDynamicObject(): Xét va chạm đạn-đạn, player-đạn, đạn-enemy, đạn-căn cứ

### 3.12 Lớp quản lý các loại va chạm trong game (CollisionManager):



- Lớp này chứa các phương thức xét va chạm dạng static để dễ dàng sử dụng bất cứ vị trí nào trong chương trình.

- Các phương thức chính:

AABBCheck(MyRectangle\* A, MyRectangle\* B): Hàm kiểm tra va chạm bằng thuật toán AABB trả về true nếu có xảy ra va chạm.

BroadphaseRect(Object\* A) : hàm tạo hình chữ nhật broad-phase.

CollisionBulletWithObject(Bullet\* A, Object\* B): hàm xét va chạm giữa đạn và các đối tượng khác trong game khi xảy ra va chạm ứng với từng loại đối tượng sẽ xử lý riêng từng trường hợp

CollisionChangeDirection(DynamicObject\* A, DynamicObject\* B): hàm xét va chạm giữa xe tank player và xe tank địch khi xảy ra va chạm sẽ kết hợp chuyển hướng xe tank địch nhằm tạo AI cho xe tank địch.

CollisionEnemy(Enemy\* A, Enemy\* B): hàm xét va chạm giữa 2 xe tank địch nếu có xảy ra va chạm sẽ thực hiện chuyển hướng để tạo AI cho xe tank

CollisionPreventMove(Object\* A, Object\* B): Hàm xét va chạm giữa đối tượng động và đối tượng tĩnh nếu có va chạm thì thực hiện chặn đối tượng và xử lý vị trí đối tượng động để ngăn không cho đối tượng động này đi lún sâu vào đối tượng tĩnh

CollisionWithItem(PlayerTank\* A, PowerUp\* B): hàm xét va chạm giữa xe tank người chơi và item đang xuất hiện trên màn chơi, nếu có xảy ra va chạm thì ta xử lý tương ứng tùy theo loại item mà người chơi thu thập được

CollisionWithScreen(Object\* A): Hàm xét va chạm giữa các đối tượng di chuyển trong game với các cạnh của màn hình

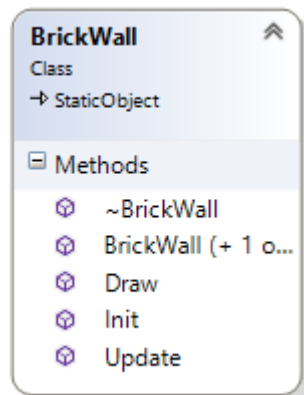
FindRespawnPosition(vector<MyRectangle\*>\* listposition, int currentposition, PlayerTank\* A, vector<Enemy\*>\* enemyOnMap): Hàm này có tác dụng tìm vị trí xuất hiện một xe tank địch mới trong 3 vị trí mặc định đã có, tuy nhiên nếu như tại vị trí sắp xuất hiện lại có một xe tank đang tồn tại thì khi ta cho xe tank này xuất hiện sẽ gây ra lỗi va chạm do đó hàm này sẽ tìm vị trí trống trong 3 vị trí để khắc phục lỗi trên.

FindRespawnPosition(vector<MyRectangle\*>\* listposition, int currentposition, vector<Enemy\*>\* enemyOnMap): Tương tự như xe tank địch thì xe tank của



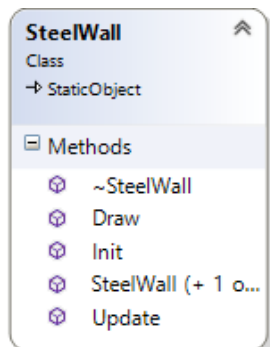
player khi bị tiêu diệt nếu còn mạng thì ta phải hồi sinh lại xe tank của người chơi, trong game sẽ có 2 vị trí xuất hiện player do đó ta sẽ tìm vị trí nào không có va chạm với các đối tượng khác trong game thì ta sẽ đặt xe tank player tại đó.

### 3.13 Lớp BrickWall



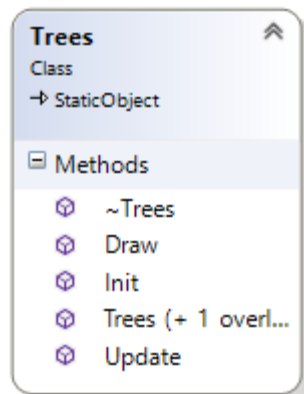
- Lớp này được kế thừa từ lớp StaticObject.
- Là lớp đặc trưng cho đối tượng tường gạch trong map game.
- Người chơi và xe tank địch không đi qua được. Bị phá hủy bởi đạn.
- Có 4 loại tường gạch ( ID\_BRICKWALL\_0, ID\_BRICKWALL\_1, ID\_BRICKWALL\_10, ID\_BRICKWALL\_11) , loại tường gạch được lưu trữ trong biến \_type nhưng thuộc tính \_id có chung giá trị là ID\_BRICKWALL.

### 3.14 Lớp SteelWall



- Lớp này được kế thừa từ lớp StaticObject.
- Là lớp đặc trưng cho đối tượng tường thép trong map game.
- Người chơi, xe tank địch và đạn không đi qua được. Bị phá huỷ bởi đạn của người chơi khi người chơi đạt cấp độ level 4
- Có 4 loại tường thép (ID\_STEELWALL\_2, ID\_STEELWALL\_3, ID\_STEELWALL\_12, ID\_STEELWALL\_13), loại tường thép được lưu trữ trong biến \_type nhưng thuộc tính \_id có chung giá trị là ID\_STEELWALL.

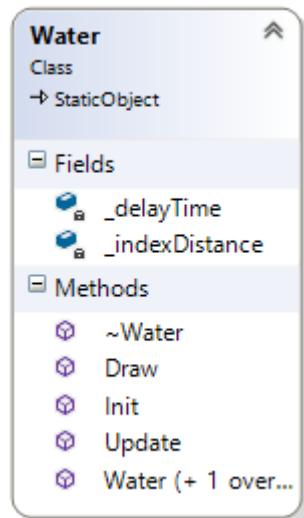
### 3.15 Lớp Trees



- Lớp này được kế thừa từ lớp StaticObject.
- Là lớp đặc trưng cho đối tượng rừng cây trong map game.
- Người chơi và xe tank địch đi qua được. Không bị phá huỷ bởi đạn. Người chơi, xe tank địch, đạn sẽ bị ẩn dưới rừng cây.
- Có 4 loại rừng cây (ID\_TREES\_6, ID\_TREES\_7, ID\_TREES\_16, ID\_TREES\_17), loại rừng cây được lưu trữ trong biến \_type nhưng thuộc tính \_id có chung giá trị là ID\_TREES

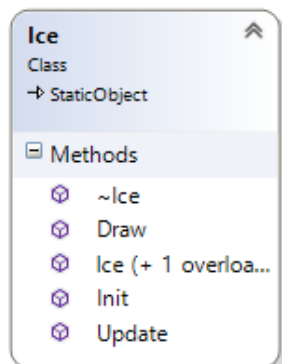
### 3.16 Lớp Water





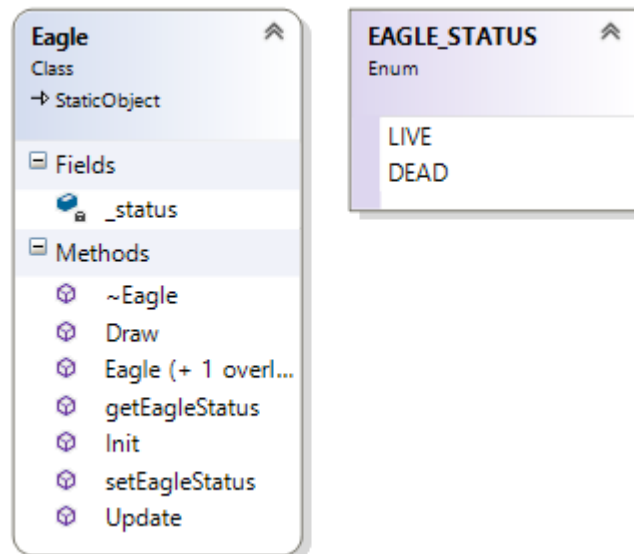
- Lớp này được kế thừa từ lớp StaticObject.
- Là lớp đặc trưng cho đối tượng sông trong map game.
- Người chơi và xe tank địch không đi qua được. Không bị phá huỷ bởi đạn. Đạn sẽ bay qua và nằm trên sông.
- Có 4 loại sông (ID\_WATER\_4, ID\_WATER\_5, ID\_WATER\_14, ID\_WATER\_15), loại sông được lưu trữ trong biến \_type nhưng thuộc tính \_id có chung giá trị là ID\_WATER
- Lớp này có thêm các thuộc tính:
  - \_delayTime: Thời gian chuyển index sprite, giúp tạo hiệu ứng động
  - \_indexDistance: Khoảng cách các index

### 3.17 Lớp Ice



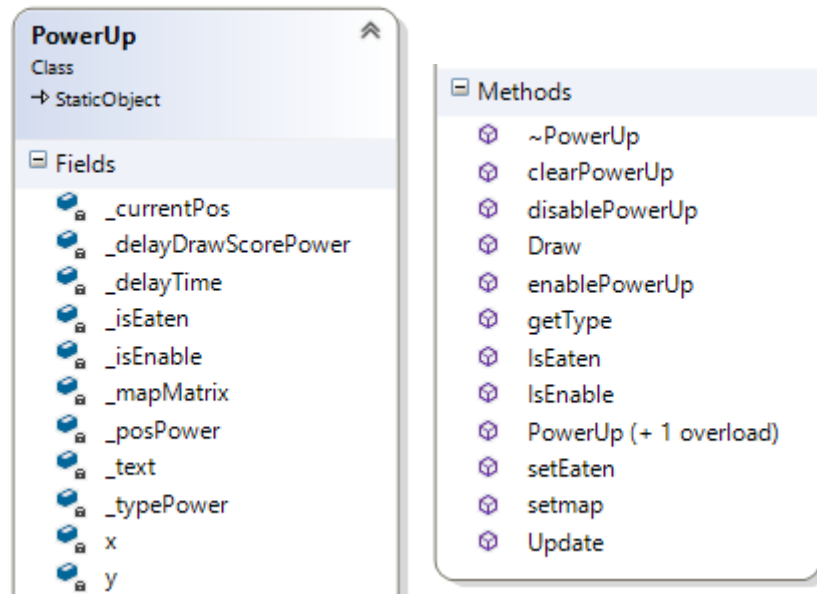
- Lớp này được kế thừa từ lớp StaticObject.
- Là lớp đặc trưng cho đối tượng băng trong map game.
- Người chơi và xe tank địch đi qua được. Không bị phá huỷ bởi đạn. Người chơi và xe tank địch sẽ nằm trên băng, đạn sẽ nằm dưới băng
- Có 4 loại băng (ID\_ICE\_8, ID\_ICE\_9, ID\_ICE\_18, ID\_ICE\_19), loại băng được lưu trữ trong biến \_type nhưng thuộc tính \_id có chung giá trị là ID\_ICE

### 3.18 Lớp Eagle



- Lớp này được kế thừa từ lớp StaticObject.
- Là lớp đặc trưng cho đối tượng hình con đại bàng – biểu tượng của căn cứ trong game.
- Người chơi và xe tank địch không đi qua được. Bị phá huỷ bởi đạn. Đối tượng này bị phá huỷ thì người chơi sẽ thua cuộc.
- Thuộc tính \_status chỉ trạng thái bị phá huỷ(DEAD) hay còn nguyên vẹn (LIVE)
- Phương thức setEagleStatus và getEagleStatus dùng để đặt và lấy trạng thái của đối tượng.
- Thuộc tính \_id có giá trị là ID\_EAGLE

### 3.19 PowerUp



- Lớp này được kế thừa từ lớp StaticObject.
- Là lớp đặc trưng cho đối tượng vật phẩm bị rơi ra khi bắn xe tank địch có nhấp nháy màu đỏ. Vật phẩm sẽ nhấp nháy liên tục trên màn hình chơi.
- Có 5 loại vật phẩm
  - Boom: Các xe tank địch trên màn hình sẽ bị phá hủy
  - Shield: người chơi sẽ được bảo vệ khỏi đạn của xe tank địch trong 1 khoảng thời gian nhất định
  - FreezeTime: xe tank địch sẽ bị đông cứng không di chuyển được trong 1 khoảng thời gian nhất định
  - Life: Thêm 1 mạng cho người chơi.
  - Star: Nâng cấp độ cho tank của người chơi.
- Chỉ có người chơi mới ăn được. Không bị phá hủy bởi đạn. Sẽ biến mất sau khi được ăn hoặc khi xe tank địch có nhấp nháy màu đỏ tiếp theo xuất hiện.
- Giải thích các thuộc tính:

`_isEnabled` chỉ trạng thái vật phẩm được kích hoạt hay chưa (kích hoạt = bị văng ra khi bắn xe tank địch nhấp nháy đỏ)

`_isEaten` chỉ trạng thái vật phẩm đã được ăn hay chưa.

`_typePower` chỉ loại vật phẩm, thuộc tính này sẽ được random.

`_text`: vẽ điểm của powerup sau khi được ăn tại vị trí của nó.

`_posPower`: vị trí của powerup, hỗ trợ vẽ điểm

`_delayTime`: Khoảng thời gian giãn cách nhấp nháy

`_delayDrawScorePower`: khoảng thời gian điểm đạt được khi ăn powerup được vẽ ra tại vị trí powerup

Các thuộc tính `x[NUM_POWER_PER_MAP], y[NUM_POWER_PER_MAP], _currentPos` phục vụ cho việc random vị trí vật phẩm. Vị trí xuất hiện vật phẩm không được nằm ở vị trí con đại bàng, trong tường thép, trong dòng sông, không ra ngoài map. Nếu để lúc bắn xe tank địch đỏ mới random ra vị trí thì game sẽ có hiện tượng đứng hình do random ra kết quả không thoả mãn điều kiện, Vậy nên ta sẽ xử lí bằng cách random ra 3 vị trí xuất hiện vật phẩm trước khi load map lên màn hình.

`_mapMatrix` hỗ trợ việc random ra vị trí xuất hiện vật phẩm.

- Giải thích các phương thức:

`ClearPowerUp()`: Xóa bỏ item nếu không được ăn

`DisablePowerUp()`: Huỷ bỏ powerup nếu nó đã được ăn

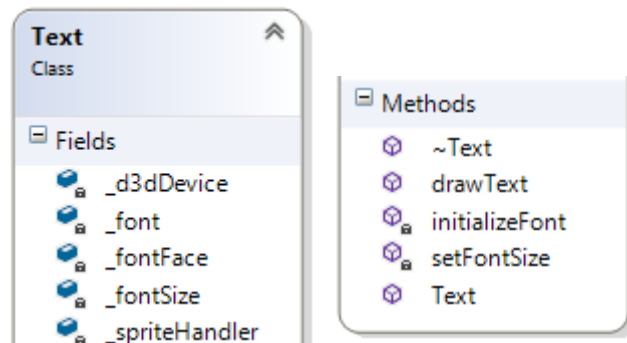
`EnablePowerUp()`: Kích hoạt powerup nếu tank địch màu đỏ bị hạ

`getType()`: lấy ra loại powerup

`setEaten()`: đặt thuộc tính `_isEaten = true` sau khi đã ăn powerup

`setMap()`: hỗ trợ việc random vị trí powerup

### 3.20 Text



- Lớp này dùng để quản lí việc vẽ các text ra màn hình bằng cách sử dụng D3DXFont
- Cần đưa vào một đối tượng D3DXSPRITE để tạo đối tượng D3DXFont. Game sử dụng font “Press Start”.
- Hàm chính của lớp này là hàm drawText: (vẽ 1 hình chữ nhật rồi điền text vào bên trong)

```
void drawText(std::string Content, D3DXVECTOR3 Position, D3DCOLOR Color =  
D3DCOLOR_XRGB(0, 0, 0), int Size = DEFAULT_FONTSIZE, UINT DT_Type =  
DT_LEFT, int Max_Lengh = 0, int Max_Line = 0);
```

- Mô tả các đối số:

Content: nội dung cần vẽ lên màn hình.

Position: vị trí cần vẽ (vẽ top left)

Color: màu chữ

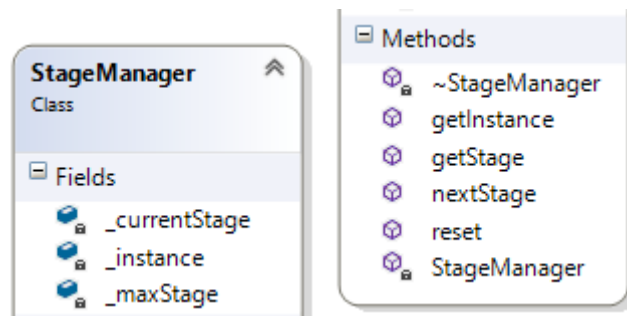
Size: kích thước chữ

DT\_Type: kiểu canh lề theo các cạnh hình chữ nhật bao quanh

Max\_Lengh: Độ dài tối đa của chuỗi, dùng để ước lượng khoảng cần thiết để vẽ, tránh trường hợp bị đè lên nhau khi chuỗi quá dài.

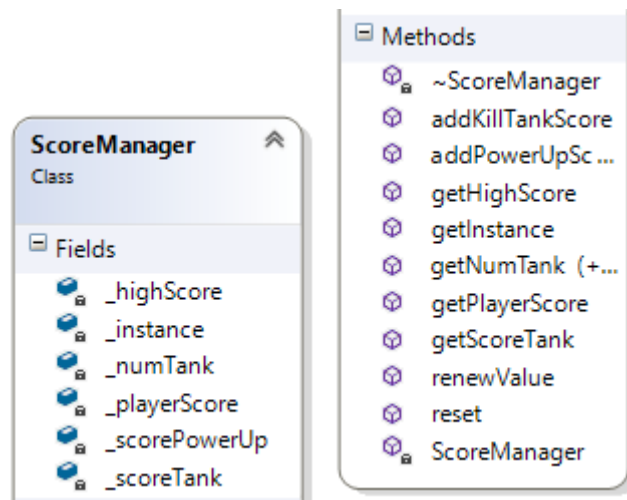
Max\_Line: Điều chỉnh chiều cao hình chữ nhật sao cho vừa đủ với số dòng cần vẽ.

### 3.21 StageManager



- Lớp này dùng để quản lí các màn chơi.
- Game có 3 màn chơi, sau mỗi màn chơi sẽ gọi hàm `nextStage`. Hàm `getStage` trả về màn chơi hiện tại, giá trị này được lưu trong biến `_currentStage`. Sau khi vượt qua 3 màn chơi, người chơi sẽ trở về Main Menu nên ta sẽ gọi hàm `reset`.
- Lớp này có áp dụng kĩ thuật Singleton.

### 3.22 ScoreManager



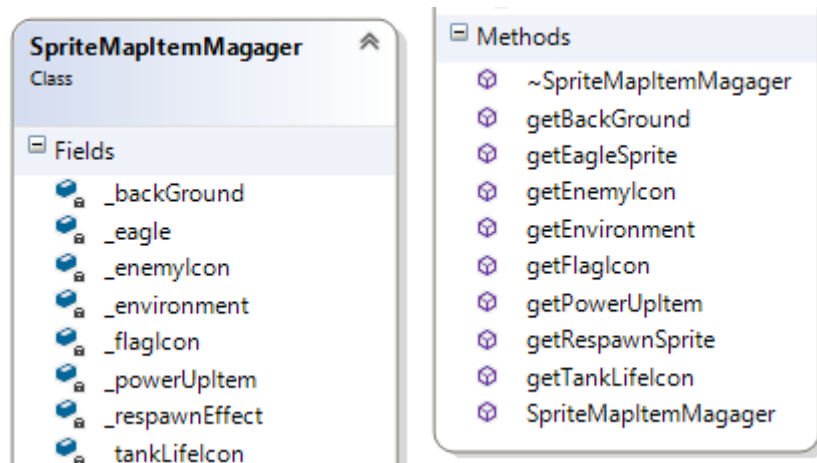
- Lớp này dùng để quản lí điểm của người chơi và điểm cao nhất.
- Các thuộc tính:
  - `_playerScore`: điểm hiện tại của người chơi
  - `_highScore`: điểm cao nhất
  - `_scoreTank[4]`: điểm của từng loại tank 1->4 tương ứng với Medium -> Light-> Heavy -> SuperHeavyTank
  - `_scorePowerUp`: điểm đạt được khi ăn vật phẩm



\_numTank[4]: số lượng tank bị người chơi bắn chết, 1->4 giống như \_scoreTank. Đầu mỗi màn chơi, thuộc tính này sẽ được reset về 0, khi bắn chết được xe tank địch nào thì số lượng xe tank địch ấy sẽ tăng tương ứng.

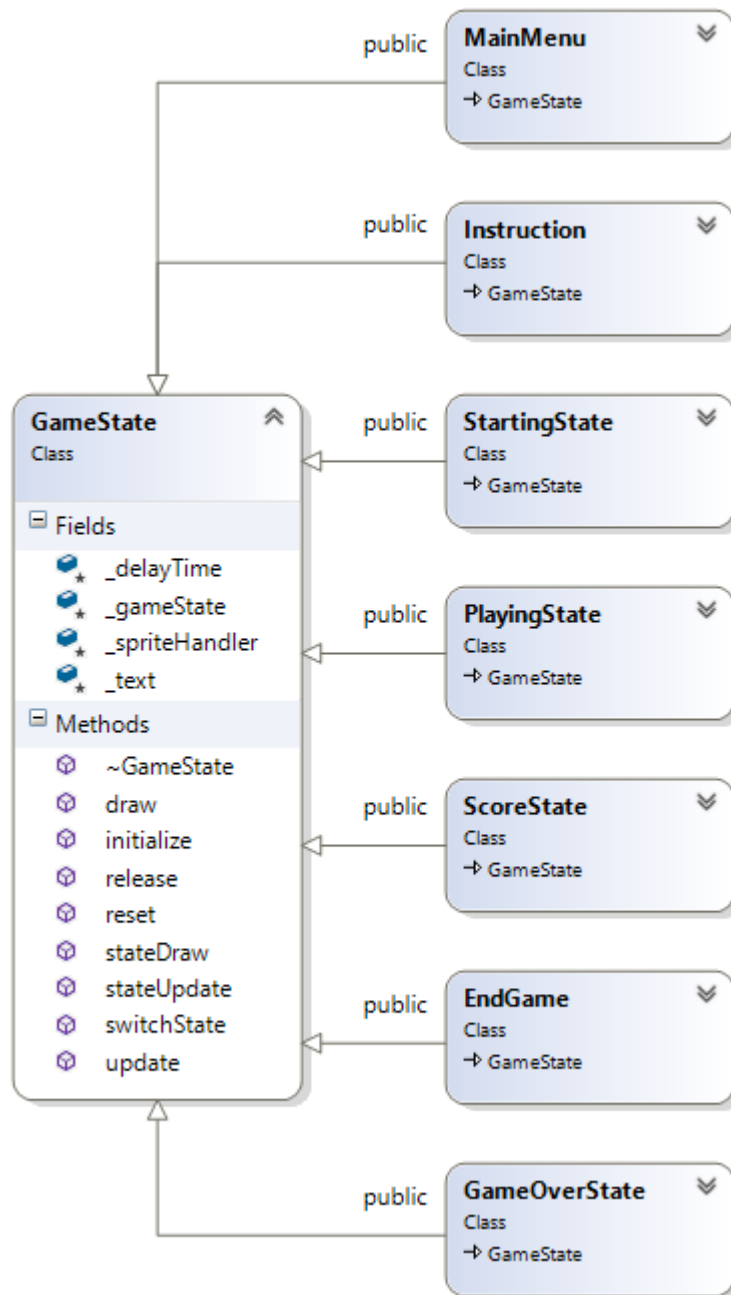
- Hàm reset dùng để gán điểm của người chơi về lại bằng 0 ghi game đã kết thúc và trở về MainMenu
- Khi người chơi đạt điểm cao hơn điểm cao nhất hiện tại, điểm của người chơi sẽ được chọn là điểm cao nhất và được lưu xuống file txt
- Lớp này có áp dụng kỹ thuật Singleton.

### 3.23 SpriteMapItemMagager



- Lớp này dùng để quản lý các sprite liên quan đến các thành phần của map game. Ví dụ: nền đen, tường gạch, đại bàng, powerup, biểu tượng enemy nhỏ góc trên phải hiển thị số xe tank địch còn lại, .....
- Vì game có rất nhiều đối tượng dạng gạch, thép, rừng,.. mà mỗi loại lại mang trong mình một sprite ảnh riêng để vẽ(các sprite ảnh thì gần giống nhau) thì tốn rất nhiều chi phí. Vậy nên lớp này được tạo ra nhằm mục đích sẽ tạo duy nhất 1 lần và các đối tượng kia tham chiếu đến, từ đó có thể tiết kiệm được chi phí vùng nhớ.
- Các hàm get hỗ trợ việc trả về địa chỉ vùng nhớ chứa sprite ảnh cần thiết cho đối tượng.

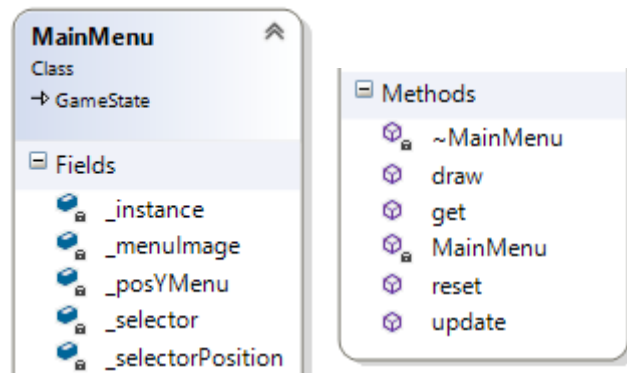
### 3.24 GameState



- Lớp này dùng để quản lí các trạng thái của game. Các lớp MainMenu, StartingState, Instruction, PlaytingState, ScoreState, EndGame, GameOver sẽ được kế thừa từ lớp này và áp dụng kĩ thuật Singleton.
- Tại một thời điểm, game chỉ có một trạng thái duy nhất, trạng thái sẽ được lưu trữ trong biến `_gameState`.

- Biến `_spriteHandler` giúp cho các lớp kế thừa từ `GameState` có thể vẽ các đối tượng lên màn hình.
- Biến `_text` hỗ trợ việc vẽ chữ lên màn hình chơi.
- Biến `_delayTime` dùng để tạo thời gian delay khi chuyển giữa các state với nhau
- Hàm `update()` và `draw()` dùng để cập nhật và vẽ các đối tượng tương ứng với từng state. Vì có từ khoá `virtual` nên mỗi state sẽ có cách định nghĩa khác nhau. Hàm `initialize` dùng để khởi tạo giá trị cho biến `_spriteHandler`, `_text`, và `_gameState`. Và quan trọng nhất là hàm `switchState` dùng để di chuyển qua lại giữa các trạng thái của game.
- Khi người chơi vượt qua hết các màn chơi hoặc thua cuộc. Game sẽ trở về `MainMenu`, khi đó hàm `reset` sẽ được gọi, người chơi lúc này sẽ chơi mới lại hoàn toàn.
- Hàm `release` dùng để giải phóng vùng nhớ đã cấp phát. Được gọi khi thoát game.

### 3.25 MainMenu



- Lớp này thể hiện cho trạng thái game ở menu chính như hình trên. Trạng thái này giới thiệu tên game, thông tin nhóm phát triển, điểm cao, cho phép người chơi tiến vào màn chơi, xem hướng dẫn chơi.
- Giải thích các thuộc tính:
  - `_instance`: thể hiện của lớp `MainMenu`. Khi game được chạy, chỉ có duy nhất biến này chứa thể hiện của lớp này.

\_menuImage: sprite của logo game “Battle City”.

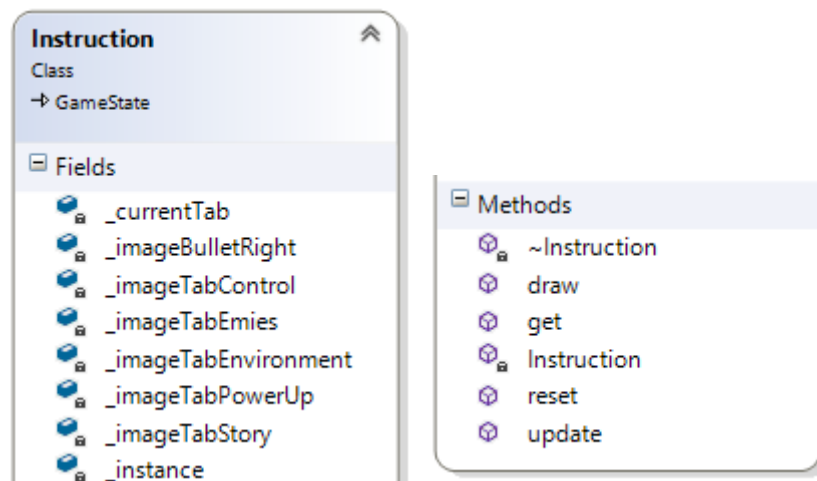
\_selector: sprite của con trỏ hình xe tank chạy chạy.

\_selectorPosition: vị trí của con trỏ select (2 vị trí tương ứng vs Play và Instruction)

\_posYMenu: hỗ trợ việc dịch chuyển menu chạy từ dưới lên.

- Trình tự chạy: Đầu tiên kiểm tra menu đã chạy lên chính giữa màn hình hay chưa bằng cách kiểm tra biến \_posYMenu. Nếu chưa thì tiếp tục cho menu chạy, nếu chạy xong rồi thì vẽ con trỏ hình xe tank để hỗ trợ việc chọn chức năng Play hoặc Instruction. Có thể nhấn nút Enter để menu lên chính giữa màn hình mà không cần đợi. Kiểm tra 2 nút Up và Down để chuyển con trỏ selector lên xuống 2 chức năng. Khi người chơi nhấn Enter, chuyển đến state tiếp theo tương ứng với sự lựa chọn của người chơi. Ngoài ra, cần vẽ lên menu điểm của người chơi trước và điểm cao nhất trong số những người đã từng chơi. Thao tác vẽ lên màn hình sẽ nằm trong hàm draw, thao tác cập nhật các đối tượng nằm trong hàm update

### 3.26 Instruction



- Lớp này thể hiện cho trạng thái game ở màn hướng dẫn chơi như hình trên. Trạng thái này cung cấp cho người chơi thông tin về game, cốt truyện, các phím điều khiển, các đối tượng trên màn chơi, các loại xe tank địch, vật phẩm..
- Giải thích các thuộc tính:

`_currentTab`: Chỉ tab hiện tại mà người dùng đang xem. Có các tab: Story, Control, PowerUp,....

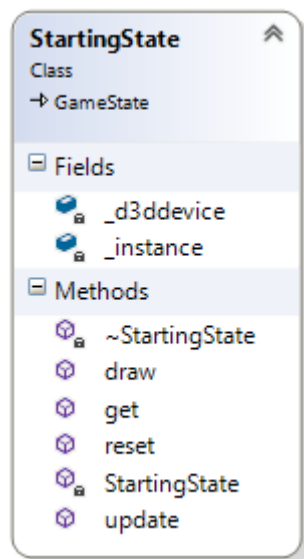
Các biến có phần đầu `_image` dùng để hỗ trợ việc hiển thị hình ảnh nhằm chú thích cho nội dung trong các tab.

`_instance`: thể hiện duy nhất của lớp này.

Người dùng có thể nhấn hai phím LEFT và RIGHT để di chuyển qua lại giữa các tab, nhấn ESC để trở main menu chính. Xử lý phím người dùng nhập nằm trong hàm `update`.

Hàm `draw` giúp vẽ ra nội dung chi tiết của từng tab tương ứng với sự lựa chọn của người dùng.

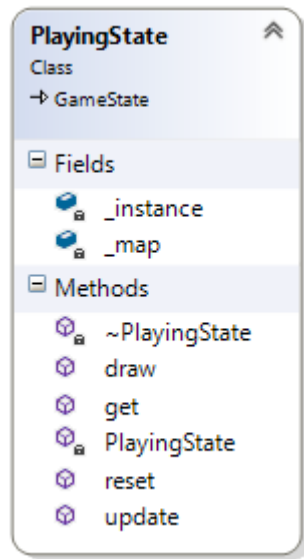
### 3.27 StartingState



- Lớp này thể hiện cho trạng thái game ở màn báo số Stage chuẩn bị chơi như hình trên. Trạng thái này cung cấp thông tin cho người chơi về số thứ tự của stage chuẩn bị chơi và tạo khoảng thời gian để người chơi chuẩn bị cho state tiếp.
- Giải thích các thuộc tính:
  - `_d3ddevice`: hỗ trợ việc xóa xám màn hình, giúp tạo hình nền.
  - `_instance`: thể hiện duy nhất của lớp này.

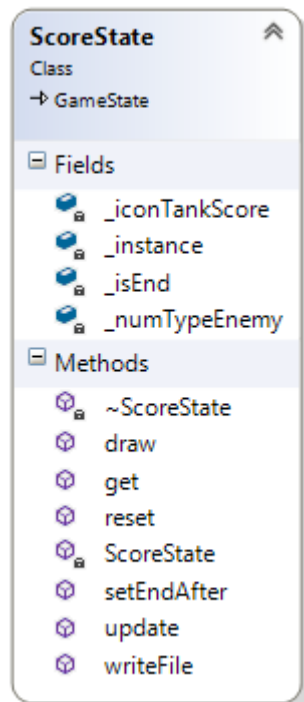
- Màn hình sẽ được xoá xám trắng, Vẽ lên nền trắng số thứ tự stage chuẩn bị chơi kèm theo nhạc hiệu chuẩn bị chơi. Trạng thái này sẽ được chạy trong vòng 3s, sau đó sẽ tự chuyển qua trạng thái PlayingState.

### 3.28 PlayingState



- Lớp này thể hiện cho trạng thái game ở màn chơi chính như hình trên. Đây là trạng thái chính của game.
- Giải thích các thuộc tính:  
`_map`: đối tượng quan trọng nhất trong game. Tất cả các đối tượng trên màn hình chơi đều là thuộc tính của đối tượng này.
- Ở trạng thái này, người chơi sẽ điều khiển tank của mình đi bắn các xe tank địch, đồng thời không quên nhiệm vụ bảo vệ căn cứ. Khi tiêu diệt hết tank địch, game sẽ chuyển qua trạng thái ScoreState , sau đó tiếp tục qua StartingState. Ngược lại, nếu căn cứ bị phá huỷ hoặc người chơi hết mạng thì cũng chuyển qua trạng thái ScoreState nhưng sau đó qua GameOverState. Nếu vượt qua hết 3 màn chơi, sau khi qua màn ScoreState, game sẽ đến trạng thái EndGame. Sau trạng thái GameOverState và EndGame, game sẽ trở lại trạng thái MainMenu. Tất cả các thao tác cập nhật và vẽ sẽ được thực hiện trong lớp Map, sẽ trình bày rõ hơn trong lớp map.

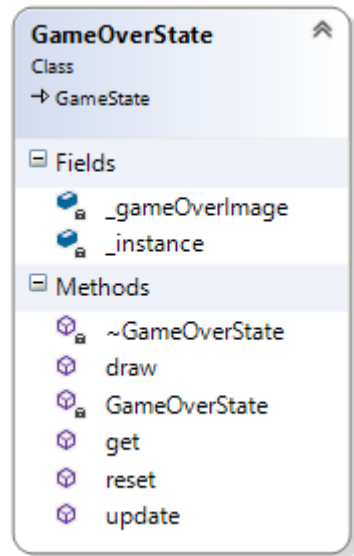
### 3.29 ScoreState



- Lớp này thể hiện cho trạng thái game ở màn tính điểm.
- Giải thích các thuộc tính:
  - `_instance`: thể hiện duy nhất của lớp này.
  - `_isEnd`: thuộc tính nhận biết sau khi kết thúc trạng thái này sẽ chuyển tới trạng thái `StartingState` (false) hay `GameOverState` (true).
  - `_numTypeEnemy`: Hỗ trợ việc hiển thị điểm đạt được với từng loại tank ra lần lượt.
- Hàm `setEndAfter`: nếu người chơi thua cuộc thì tham số đưa vào là true, ngược lại là false.
- Hàm `writeFile`: nếu người chơi đạt điểm cao hơn điểm cao nhất thì cập nhật điểm cao nhất xuống file txt.
- Ở màn này sẽ đưa thống kê về số lượng tank bị người chơi bắn hạ ở stage vừa chơi bao gồm: loại tank + số lượng + điểm. Ngoài ra còn hiển thị cho người chơi biết điểm hiện tại của mình và điểm cao nhất là bao nhiêu. Tiếp theo sẽ chuyển đến 1 trong 3 trạng thái:
  - o `GameOverState`: người chơi thua cuộc.

- StartingState: nếu chưa vượt qua hết 3 stage.
- EndGame: nếu đã vượt qua state 3.

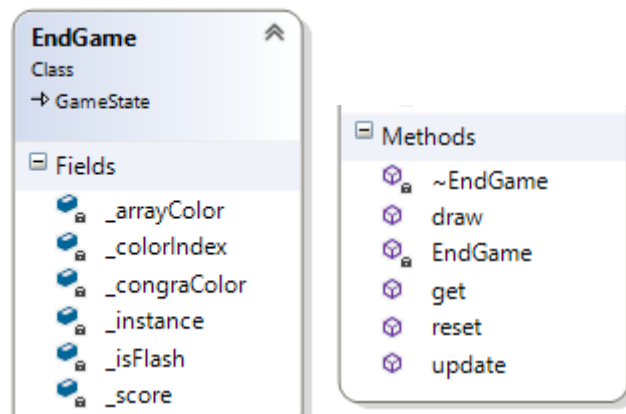
### 3.30      **GameOverState**



- Lớp này thể hiện cho trạng thái game ở màn báo thua cuộc.
- Giải thích các thuộc tính:  
   \_gameOverImage: sprite ảnh “GameOver”  
   \_instance: thể hiện duy nhất của lớp này.
- Trạng thái này xuất hiện khi người chơi thua cuộc (hết mạng hoặc căn cứ bị phá hủy). Trạng thái này đơn giản chỉ đưa ra 1 hình “GameOver” rồi chuyển về menu chính (MainMenu).

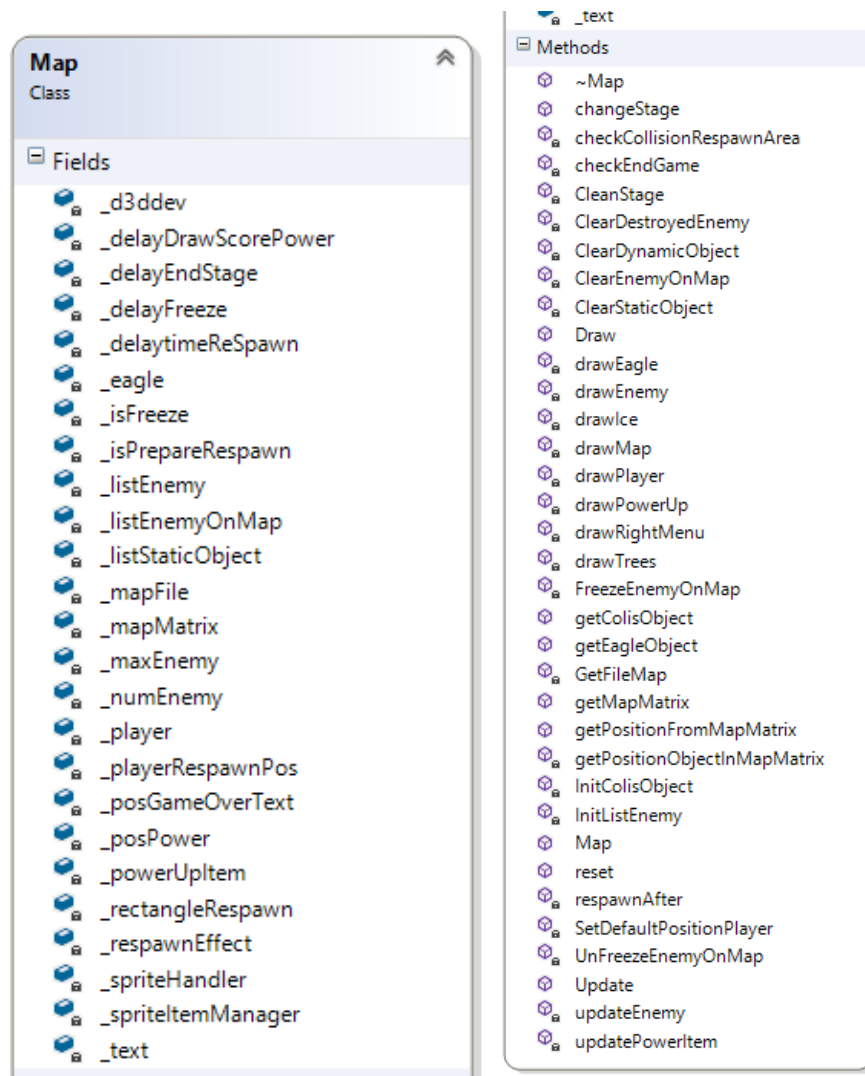
### 3.31      **EndGame**





- Lớp này thể hiện cho trạng thái game ở màn báo người chơi đã hoàn thành nhiệm vụ bảo vệ căn cứ (vượt qua 3 stage).
- Giải thích các thuộc tính:
  - \_arrayColor: các màu mà chữ “Congratulation” có thể nhận
  - \_colorIndex: vị trí màu hiện tại trong bảng màu
  - \_congraColor: màu chữ “Congratulation”
  - \_isFlash: báo khi nào TEXT\_ESC được vẽ ra
  - \_score: điểm của người chơi khi kết thúc game.

### 3.32 Map



- Đây là lớp quản lý tất cả các cập nhật và vẽ của tất cả các đối tượng trên màn hình ở màn chơi chính.
- Giải thích các thuộc tính:
  - \_player: Đối tượng tank được người chơi điều khiển (tank màu vàng).
  - \_mapMatrix: Ma trận hai chiều kiểu int lưu giữ vị trí các đối tượng tĩnh (gạch, thép, sông, băng, rừng)
  - \_numEnemy: số lượng tank địch còn lại chưa xuất hiện
  - \_maxEnemy: số lượng tank địch tối đa trong một stage.

- `_isPrepareRespawn`: biến cờ hiệu dùng để báo có một chiếc tank địch chuẩn bị xuất hiện
- `_delaytimeReSpawn`: thời gian giãn cách xuất hiện tank địch
- `_delayEndStage`: thời gian chờ chuẩn bị chuyển qua ScoreState
- `_delayFreeze`: thời gian powerup Shield hoạt động
- `_isFreeze`: biến cờ hiệu báo rằng vật phẩm Shield đang được phát huy tác dụng
- `_text`: hỗ trợ vẽ điểm powerup, dòng chữ “GAMEOVER” khi người chơi thua cuộc, số stage hiện tại, số mạng còn lại của người chơi.
- `_posGameOverText`: vị trí vẽ lên dòng chữ “GAMEOVER”
- `_listStaticObject`: danh sách các đối tượng tĩnh trên màn hình bao gồm: tường gạch, tường thép, rừng, sông, băng.
- `_listEnemy`: danh sách tank địch sẽ xuất hiện trong 1 stage.
- `_listEnemyOnMap`: danh sách tank địch đang xuất hiện trên màn hình.
- `_rectangleEnemyRespawn`: danh sách các vị trí mà tank địch sẽ xuất hiện
- `_playerRespawnPos`: danh sách vị trí mà người chơi sẽ xuất hiện.
- `_mapFile`: hỗ trợ việc đọc file map có định dạng txt
- `_spriteItemManager`: quản lí các sprite của StaticObject trên màn chơi.
- `_eagle`: đối tượng căn cứ (hình con đại bàng) mà người chơi phải bảo vệ
- `_powerUpItem`: Quản lí powerup
- `_respawnEffect`: hỗ trợ việc tạo hiệu ứng tại vị trí tank chuẩn bị xuất hiện
- `_spriteHandler`, `_d3ddev`: hỗ trợ việc vẽ các đối tượng lên màn hình
- Giải thích các phương thức:
  - `Map(LPD3DXSPRITE spriteHandler)`: Khởi tạo các đối tượng trên map như: ma trận map, `_text`, powerup, căn cứ, tank người chơi..
  - `changeStage()`: dùng để chuyển từ stage này sang stage khác; reset một số thuộc tính về giá trị mặc định; đọc file map và lấy các thông tin cần thiết như ma trận map, số lượng tank địch,...; tạo danh sách tank địch sẽ xuất hiện trên map, tạo danh sách các đối tượng tĩnh như gạch, thép, rừng,...; đưa ma trận map vào cho `_player` để hỗ trợ xét va chạm

`getStaticObject()`: trả về `_listStaticObject`

`getPositionFromMapMatrix(int row, int column)`: đưa vào vị trí trong ma trận map để lấy ra tọa độ thực trên màn hình chơi.

`getPositionObjectInMapMatrix(int x, int y)`: đưa vào tọa độ trên màn hình chơi để lấy ra vị trí trong ma trận map

`GetFileMap()`: Mở file map định dạng txt

`InitStaticObject()`: dựa vào ma trận map để tạo ra các đối tượng tĩnh như gạch, thép, rừng,...;

`InitListEnemy(int numOfEnemy[], string order)`: tạo ra danh sách các tank địch sẽ xuất hiện trong stage tiếp

`checkEndGame()`: kiểm tra kết thúc game. Nếu người chơi hết mạng hoặc căn cứ bị phá hủy thì vẽ dòng chữ “GAMEOVER”, game sẽ kết thúc sau màn `ScoreState` (`ScoreState::get()->setEndAfter(true)`). Nếu người chơi đã tiêu diệt hết tank địch thì game sẽ chuyển tới stage `StartingState` sau màn `ScoreState`. Trước khi chuyển state, phải gọi hàm `CleanStage()` để xóa các đối tượng của stage vừa mới chơi xong.

`updateEnemy()`: cập nhật các đối tượng tank địch trên map. Nếu số lượng tank địch trên map  $< 4$  thì cho thêm tank địch xuất hiện và đảm bảo số lượng tank địch luôn  $\leq 4$ .

`checkCollisionRespawnArea()`: kiểm tra vị trí sắp xuất hiện của tank địch có đối tượng nào đang đứng ở đó không, nếu có thì chuyển hướng xuất hiện qua 1 trong 2 vị trí còn lại, nếu không thì xuất hiện bình thường

`respawnAfter(int delaytime)`: phân phối thời gian xuất hiện tank địch một cách hợp lí. Nếu powerup Freeze đang được kích hoạt thì cho tank địch chuẩn bị xuất hiện cũng bị ảnh hưởng bởi powerup này. Nếu tank địch thứ 11 hoặc 18 xuất hiện mà powerup vẫn chưa ăn thì powerup sẽ biến mất.

`updatePowerItem()`: Nếu powerup bị người chơi ăn thì kích hoạt chức năng của powerup. Tùy theo từng loại powerup mà có các chức năng khác nhau. Nếu chức

năng đóng băng di chuyển tank địch đã được kích hoạt thì kiểm tra thời gian đóng băng đã hết chưa để đưa tank địch về lại trạng thái bình thường.

drawMap(): vẽ các đối tượng tĩnh trong \_listStaticObject lên map game, nơi nào không có đối tượng thì vẽ nền màu đen. Nhưng có điều đặc biệt là trong hàm này ta chỉ vẽ gạch, thép, sông. Rừng và băng sẽ được vẽ sau vì độ sâu của nền của các đối tượng này khác nhau.

drawEagle(): vẽ căn cứ lên map

drawPlayer(): vẽ tank của người chơi lên map

drawTrees(): duyệt \_listStaticObject và vẽ các đối tượng thuộc loại rừng lên map

drawIce(): duyệt \_listStaticObject và vẽ các đối tượng thuộc loại băng lên map

drawPowerUp(): vẽ power lên màn hình nếu nó được kích hoạt bởi việc bắn chết tank địch có nhảy đỏ

drawEnemy(): Vẽ các xe tank địch lên map. Ở những xe tank địch chuẩn bị xuất hiện sẽ vẽ thêm hiệu ứng nhấp nháy

drawRightMenu(): vẽ số lượng tank địch còn lại chưa xuất hiện trên map. Vẽ số thứ tự của stage hiện tại, số mạng còn lại của người chơi.

ClearDestroyedEnemy(): Vì khi xét va chạm không huỷ liền đối tượng mà chỉ xét thuộc tính \_isTerminated = true vậy nên cần có hàm này để xoá các đối tượng bị terminated.

ClearStaticObject(): Sau khi hoàn thành 1 stage, trước khi chuyển qua ScoreState thì cần phải dọn dẹp các đối tượng tĩnh chưa bị phá huỷ trên map để nạp vào các đối tượng tĩnh của stage sau, đó chính là mục đích của hàm này.

ClearDynamicObject(): Tương tự hàm trên, nếu người chơi bị thua cuộc, danh sách tank địch cần phải được xoá để nạp lại danh sách mới chuẩn bị cho người chơi sau.

SetDefaultValuePlayer(): Đưa tank của người chơi về vị trí mặc định. Là một trong những thao tác cần làm trước khi chuyển qua ScoreState

ClearEnemyOnMap(): Huỷ tất cả các đối tượng tank địch hiện có trên map. Hàm này được gọi khi tank của người chơi ăn powerup Bomb

FreezeEnemyOnMap(): “Đóng băng”(ngưng di chuyển) tất cả các đối tượng tank địch hiện có trên map.

UnFreezeEnemyOnMap(): Ngược lại với hàm trên, đưa tank địch về trạng thái di chuyển bình thường.

CleanStage(): Hàm này được gọi trước khi kết thúc PlayingState. Dọn dẹp map game để chuẩn bị load map sau.

#### IV. CÁC KỸ THUẬT ỨNG DỤNG TRONG GAME

##### 1. Thiết kế các lớp đối tượng theo singleton pattern:

- Trong game ta sẽ có các thành phần như âm thanh, hiệu ứng, đạn là các loại đối tượng đặc biệt ta chỉ cần khởi tạo một thể hiện duy nhất của lớp rồi sử dụng cho toàn bộ chương trình, do đó ta cần áp dụng kỹ thuật singleton pattern để thiết kế các lớp này để tạo ra một thể hiện duy nhất và có thể dùng như một biến toàn cục nhằm giảm các thao tác khởi tạo và xóa gây tốn chi phí.
- Cách thiết kế này áp dụng vào các lớp: EffectManager, BulletManager, GameSound,.....

##### 2. Vẽ các đối tượng động có animation:

- Trong game các đối tượng như player, enemy khi vẽ trên màn hình để tạo cho người chơi cảm giác như tank đang chuyển động thực (các bánh xích của xe như đang lăn) thì ta áp dụng kỹ thuật vẽ lặp lại các frame của 1 sprite. Trong game, để làm được ta tổ chức sprite sheet theo cách như sau: Mỗi tượng động có 4 hướng di chuyển là lên, xuống, trái ,phải nên ta sẽ tạo ra 4 sprite sheet tương ứng với 4 hướng và mỗi sprite cho một hướng di chuyển gồm có 2 dòng, và số cột tùy thuộc vào loại đối tượng trong game (ví dụ: player có 4 level nên sẽ có 4 cột, các loại enemy thường sẽ có 2 cột để vẽ 2 trạng thái là tank thường hay có bonus item)
- Ví dụ về cách tạo sprite trong game:
  - Player với hướng di chuyển lên



- LightTank



- Khi muốn vẽ sprite chuyển động ta chỉ việc gọi hàm chuyển sang frame ở dòng tiếp theo ứng với cột theo từng loại đối tượng là đã có thể vẽ ra animation cho đối tượng

### 3. Xét va chạm trong game

#### 3.1 Xét va chạm giữa đối tượng động và đối tượng tĩnh

- Để xét va chạm giữa đối tượng động (cụ thể là xe tank) và đối tượng tĩnh (gạch, tường thép, nước,...) ta áp dụng thuật toán broad-phasing để tạo một hình chữ nhật bao lấy đối tượng động bao gồm cả vận tốc hiện tại của đối tượng, sau đó dùng thuật toán AABB xét va chạm đối tượng động này với đối tượng tĩnh, nếu phát hiện có va chạm ta sẽ xử lý ngăn không cho đối tượng di chuyển lún sâu vào đối tượng tĩnh bằng cách gán vận tốc trở về 0 và tính toán lại vị trí của đối tượng động sao cho khi người chơi hay enemy cố tính di chuyển vào đối tượng tĩnh thì sẽ không thể đi sâu vào được.
- Một vấn đề tiếp theo ta cần giải quyết đó là do cách phân hoạch không gian để vẽ các đối tượng tĩnh lên màn hình thì thực hiện đọc từ file map sau đó khởi tạo các đối tượng này vẽ ra màn hình, tuy nhiên đặc thù game phải chia nhỏ ra rất nhiều đối tượng hiển thị tất cả lên màn chơi do đó số lượng đối tượng tĩnh sẽ rất lớn, để xét va chạm được, nếu tiến hành kiểm tra tuần tự các đối tượng động (player, enemy) với các đối tượng tĩnh này thì sẽ tiêu tốn rất nhiều chi phí, ta sẽ phải dùng vòng lặp, lặp đi đến hàng ngàn lần sẽ làm game bị chậm. Do đó, vấn đề đặt ra là làm sao xác định chính xác cần xét va chạm với một nhóm các đối tượng động nào ngay tại vị trí hiện tại để tiết kiệm chi phí nhất. Để giải quyết vấn đề này, trước hết ta phải tiền xử lý file map trước khi đọc vào game, sau đó dựa trên ma trận Map ta tiến hành xây dựng thuật toán tìm các đối tượng xung

quanh đối tượng động tại vị trí (x,y) mà đối tượng này đang đứng so với tọa độ (dòng,cột) trong ma trận Map để tìm.

- Thuật toán tìm các đối tượng tĩnh xét va chạm:

- Ý tưởng: Map của game được xây dựng theo dạng tile với mỗi tile có kích thước là 8x8 và toàn bộ ma trận Map của game gồm có (dòng, cột) = (52,52) tức là trong 1 map có tối đa 52x52 đối tượng tĩnh. Dựa vào tọa độ (x,y) của đối tượng động ta có thể quy đổi ra đối tượng động này đang thuộc tọa độ (dòng, cột) nào trong ma trận Map sau đó ta sẽ duyệt tìm các đối tượng tĩnh xung quanh để xét va chạm

- Cách thực hiện:

Bước 1: Tính tọa độ (dòng, cột) của object động dựa trên tọa độ (x,y)

Bước 2: Tính toán cần lấy bao nhiêu đối tượng xung quanh để xét va chạm bằng cách:

- Nếu đối tượng động là tank (kích thước 32x32 (pixel) tương đương 4x4 (tile)) trường hợp tọa độ (x,y) hiện tại của tank chia hết cho kích thước 1 tile thì ta sẽ lấy mỗi cạnh xung quanh tank 4 đối tượng tĩnh, ngược lại không chia hết nghĩa là tank không nằm trọn trong 4x4 tile trên map mà vị trí này có giao với 1 phần của tile kế tiếp thì ta phải lấy mỗi cạnh của tank là 5 đối tượng để không bị thiếu
- Nếu đối tượng động là đạn (kích thước 8x8 (pixel) tương đương 1x1(tile)) thì cả khi chia hết hay không ta chỉ lấy mỗi cạnh của đạn 1 đối tượng để xét va chạm.

Bước 3: Sau khi tính được số lượng đối tượng cần lấy để xét va chạm ta tiến hành lấy giá trị từ ma trận map dựa vào dòng và cột của đối tượng động. Khi đọc giá trị từ ma trận map ta kiểm tra nếu giá trị này khác -1 (trong ma trận map -1 nghĩa là không có đối tượng nào tại vị trí đó) sau đó ta tiến hành xét lấy các đối tượng có thể va chạm được bằng cách như sau:

- Chia lấy phần dư giá trị này ta sẽ biết được đối tượng ta cần lấy thuộc loại nào trong list vector 2 chiều chứa tất cả đối tượng tĩnh



- Chia lấy phần nguyên giá trị này ta sẽ biết được index của đối tượng này trong list ta đã tìm được từ phép tính như trên.

Bước 4: Add tham chiếu con trỏ đối tượng tìm được vào list những phần tử xung quanh để tiến hành xét va chạm

Bước 5: Sau khi đã có được list các đối tượng tĩnh xung quanh đối tượng động đang xét va chạm tiến hành kiểm tra và xử lý va chạm

Bước 6: Xóa hết các tham chiếu đến các đối tượng xung quanh để vòng lặp tiếp theo chỉ thêm vào tham chiếu các đối tượng mới nhằm tiết kiệm chi phí.

- Như vậy khi áp dụng thuật toán này, trường hợp phức tạp nhất là khi đối tượng nằm giữa 2 tile thì tổng đối tượng cần xét cho 1 đối tượng động với các đối tượng tĩnh tại 1 vòng lặp game cũng chỉ là 20 đối tượng so với phải lặp hàng trăm lần!

### 3.2 Xét va chạm giữa đối tượng động và đối tượng động

- Trong game khi xảy ra va chạm giữa 2 đối tượng động với nhau (player-enemy hay enemy-enemy) theo logic thì 2 đối tượng không thể va chạm rồi lún vào nhau. Dựa trên thuật toán broad-phasing để tiến hành xét va chạm giữa 2 đối tượng di chuyển, ta sẽ xét va chạm giữa 2 hình chữ nhật broad-phasing của 2 đối tượng. Làm theo cách trên, ta không áp dụng vận tốc tương đối nhưng tại thời điểm xét có kèm vận tốc ta sẽ biết được trong lần update kế tiếp có xảy ra va chạm không, nếu có xảy ra va chạm thì ta tiến hành set vận tốc của 2 đối tượng về 0 sau đó chọn 1 đối tượng làm chuẩn rồi set lại vị trí sao cho khi đối tượng này có cố duy chuyển tiến sát vào đối tượng kia cũng không xảy ra lún. Đến đây ta phải xử lý tiếp áp dụng AI vào để cho đối tượng xe tank địch chuyển hướng tìm hướng di chuyển khác hoặc tấn công để tiêu diệt người chơi.
- Do tại một thời điểm trong màn chơi có tối đa 5 đối tượng động ( 1player và 4 enemy) nên ta có thể đơn giản xét va chạm bằng vòng lặp giữa các đối tượng đó với nhau mà không phải chú ý nhiều đến vấn đề hiệu suất.

### 3.3 Xét va chạm giữa đạn và các đối tượng khác trong game

- Để xét va chạm giữa đạn và đối tượng tĩnh ta chỉ việc áp dụng lại thuật toán tìm các đối tượng tĩnh xung quanh đạn và xét va chạm nếu xảy ra va chạm ta sẽ xét tiếp đạn va chạm với loại đối tượng tĩnh nào vì trong gameplay sẽ quy định đạn có thể phá hủy đối tượng tĩnh như thế nào, cụ thể:
  - Khi đạn va chạm với gạch thông thường thì cả đạn và gạch đều bị hủy
  - Khi đạn va chạm với tường thép nếu level của đạn (level của đạn tương ứng với level của tank bắn ra) nhỏ hơn level 4 thì đạn sẽ không thể phá hủy được tường thép, trường hợp xe tank của player đã được nâng cấp đến cấp cao nhất sẽ có thể bắn ra đạn phá hủy tường thép.
  - Các đối tượng như cây (trees), băng (ice) hay nước (water) ta không xét va chạm với đạn
- Xét va chạm giữa đạn với player và enemy: Để xét được loại va chạm này, khi một xe tank bắn ra đạn ra phải biết được đạn này do enemy hay player bắn ra để xét va chạm chính xác. Theo gameplay, đạn của player và đạn của enemy khi va chạm với nhau cả 2 sẽ bị hủy, đạn giữa các enemy bắn ra sẽ không bị hủy khi va chạm với nhau.

#### **4. Thiết kế AI cho các xe tank địch:**

- Trong game Battle City thành phần không thể thiếu tạo độ khó và hấp dẫn cho người chơi đó là AI, các đối tượng enemy phải có cách hành vi cụ thể để xử lý trong game. AI trong game này không đòi hỏi phải áp dụng các thuật toán tìm đường phức tạp, vì làm như vậy sẽ không tạo cảm giác hứng thú cho người chơi do với mọi màn chơi tất cả các enemy sẽ luôn có xu hướng di chuyển trên hướng đi đã tìm ra từ các thuật toán tìm đường đến vị trí căn cứ của player và tiêu diệt căn cứ, do đó người chơi sẽ có cảm giác nhàm chán khi mà các enemy cứ chỉ dựa trên một vài hướng di chuyển đã tính toán sẵn. Để thiết kế AI một cách đơn giản nhưng hiệu quả, vừa tạo yếu tố bất ngờ, tạo ra các tình huống khó khăn bất ngờ cho người chơi thì ta chỉ áp dụng các kỹ thuật phát sinh ngẫu nhiên giá trị để từ đó enemy đưa ra hành vi tương ứng.

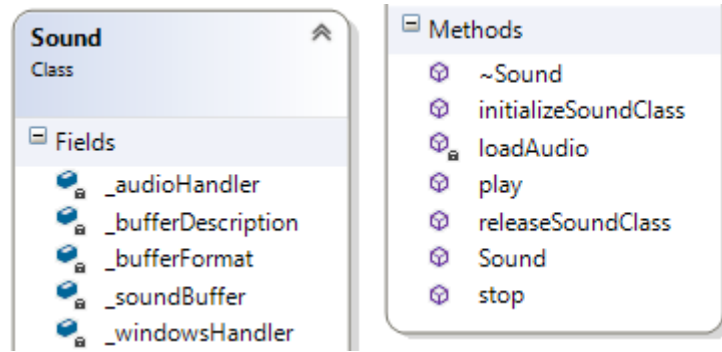
- Ý tưởng: Dựa trên khảo sát gameplay gốc của trò chơi rút ra được là các xe tank dịch di chuyển 1 cách rất ngẫu nhiên và xe tank địch đến được căn cứ của player cũng dựa trên sự ngẫu nhiên đó chứ không dựa trên một thuật toán tìm đường cố định, do đó để tạo ra AI cho enemy ta dùng hàm random ra giá trị theo 4 hướng di chuyển của enemy, mỗi khi có va chạm xảy ra giữa enemy với đối tượng hay với màn hình thì sẽ tiến hành chuyển hướng. Xe tank địch luôn có xu hướng tấn công liên tục để làm cho xe tank địch tự bắn thì ta dùng hàm tính thời gian và kiểm tra tại thời điểm lần bắn trước với lần bắn tiếp theo đã thỏa khoảng thời gian giữa 2 loạt đạn chưa, nếu đã thỏa thì mới được phép bắn.

## 5. Xử lí âm thanh trong game

- Trong game để tạo sự hứng thú cho người chơi thì không thể thiếu âm thanh, để tạo được âm thanh trong game ta cần thiết kế ra ba class để quản lý:
  - o Lớp để xử lý file âm thanh (CWaveFile): Lớp này dựa trên framework xây dựng sẵn do Microsoft cung cấp để phân tích, xử lý âm thanh dựa trên multimedia I/O stream.
  - o Lớp đọc, và lưu lại âm thanh (Sound): Khi cần sử dụng loại âm thanh nào thì tạo thể hiện thuộc lớp này để handle âm thanh. Lớp này cho phép lưu âm thanh xuống bộ đệm và khi cần ta có thể play âm thanh trong này.
  - o Lớp trực tiếp điều khiển play, stop âm thanh theo mong muốn.(GameSound) : Lớp được thiết kế theo singleton pattern có thành phần là tất cả các loại âm thanh sử dụng trong toàn game và có thể sử dụng âm thanh bất cứ khi nào cần trong tất cả các lớp khác mà chỉ cần khởi tạo 1 thể hiện duy nhất.
- Lớp CWaveFile: Lớp này là một trong các lớp để xử lý âm thanh (âm có phần mở rộng là .wav) do Microsoft cung cấp đã được xây dựng sẵn các thuộc tính để kiểm soát một file âm thanh kèm theo các phương thức xử lý dùng để mở âm thanh, đọc âm thanh, reset trạng thái của file âm thanh, đóng một file âm thanh và ghi âm thanh. Tuy nhiên nhu cầu của chúng ta chỉ cần phục vụ cho việc load

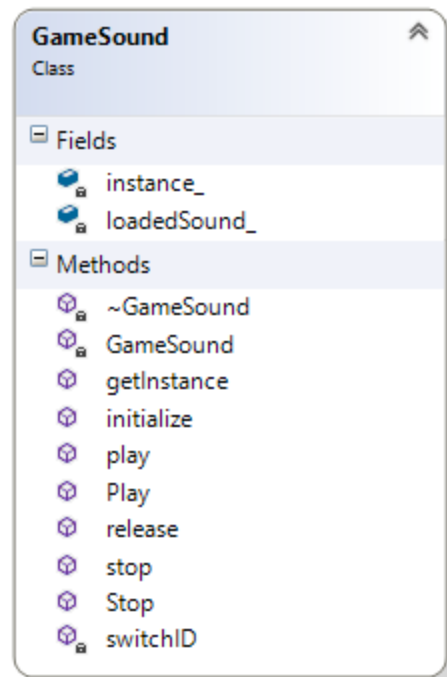
một file âm thanh và đọc file âm thanh nên ta chỉ sử dụng một số phương thức nhất định

- Lớp Sound:



Lớp này dùng để handle âm thanh ta cần dùng trong game, bao gồm các thành phần đối tượng handle âm thanh, thuộc tính quản lý buffer, buffer âm thanh và các phương thức khởi tạo âm thanh, load âm thanh, play một file âm thanh, hay stop âm thanh

- Lớp GameSound:



Lớp GameSound được thiết kế theo singleton pattern, từ lớp này ta sẽ chỉ tạo ra một thể hiện duy nhất (\_instance) ta sẽ dùng trực tiếp instance này tại bất cứ vị trí nào trong game cần play âm thanh, vì đây là biến static nên ta sẽ chỉ định lấy thể hiện của lớp sau đó gọi hàm Play với các đối số truyền vào đường dẫn đến file âm thanh, và chọn chế độ chơi lặp lại hay chỉ chơi qua 1 lần.

Trong lớp GameSound có một thuộc tính SoundMap dùng để ánh xạ một ID âm thanh (trong game có nhiều âm thanh mỗi âm thanh sẽ có một ID) là đối số string trong cấu trúc trên, nó sẽ ánh xạ đến một âm thanh đã được khởi tạo sẵn.

```
SoundMap : map<string,Sound*>  
Typedef
```

Cấu trúc để khởi tạo một ánh xạ âm thanh SoundMap.

```
SoundPair : pair<string,Sound*>  
Typedef
```

Cơ chế hoạt động của âm thanh trong game: Khi muốn phát một âm thanh ta sẽ thực hiện gọi hàm từ lớp GameSound : GameSound::getInstance()->Play(int id, bool repeat): với id là mã âm thanh ứng với từng loại âm thanh, repeat là chế độ có muốn phát âm thanh đó lặp đi lặp lại liên tục không.

Khi ta gọi hàm trên lớp GameSound thì sẽ tiến hành tìm kiếm trong list các âm thanh đã được load từ trước đã tồn tại âm thanh cần phát chưa, nếu đã có thì tiến hành phát âm thanh đó theo chế độ có repeat hay không, ngược lại ta sẽ tiến hành load âm thanh này lên bộ đệm.

Khi tiến hành load một âm thanh mới vào thì một thể hiện của lớp CWaveFile sẽ được tạo trong lớp Sound để lưu trữ các thông số của âm thanh vừa đọc sau đó tiến hành định dạng và khởi tạo âm thanh này lưu

xuống bộ đệm của card âm thanh hay bộ nhớ (tùy lúc khởi tạo thiết bị âm thanh) và sau đó sẽ tiến hành phát âm thanh này.

## 6. Chuyển đổi file map tạo được từ phần mềm Tiled

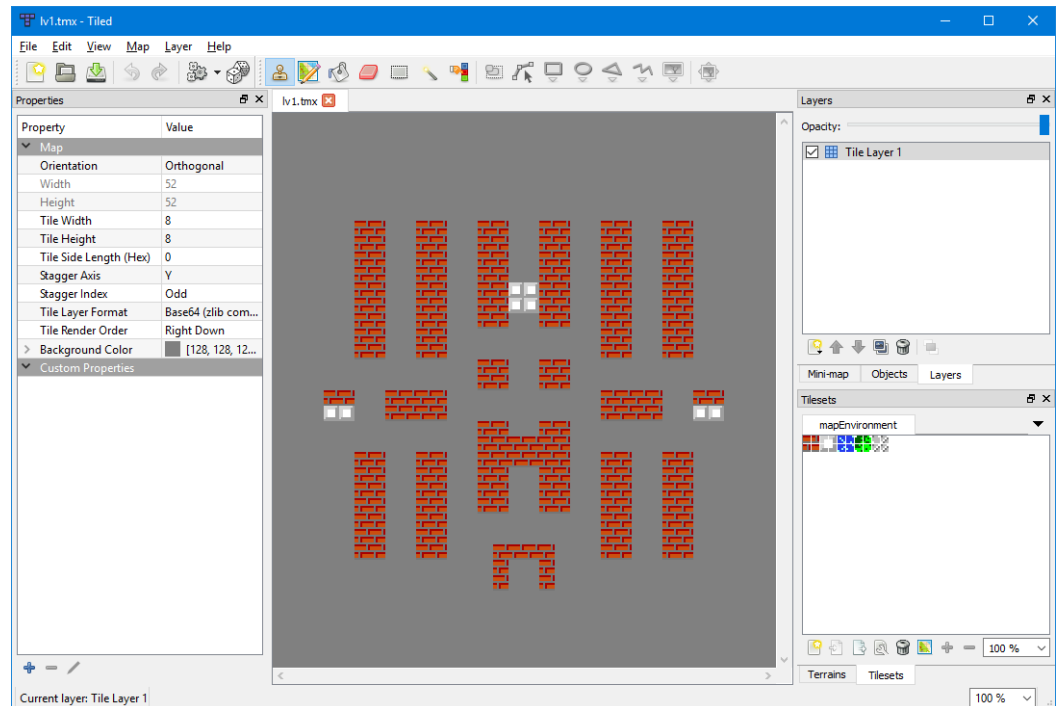
- Tiled là phần mềm mã nguồn mở hỗ trợ việc xây dựng map cho game.

Link download: <http://www.mapeditor.org/download.html>

Hướng dẫn sử dụng:

<http://gamedevelopment.tutsplus.com/tutorials/introduction-to-tiled-map-editor-a-great-platform-agnostic-tool-for-making-level-maps--gamedev-2838>

- Ta sẽ sử dụng phần mềm Tiled để vẽ map cho game. Sử dụng file Environment.png để tạo tileset. Màn hình giao diện của Tiled như sau



- Map game sẽ có chiều rộng 52 ô tile và chiều dài 52 ô.
- Sau khi vẽ xong, phần mềm sẽ xuất ra cho chúng ta 1 file txt chứa ma trận map và các thông tin liên quan bên trong đó. Tuy nhiên file này chưa thể sử dụng vào game được. Chúng ta cần xử lý ma trận map và thêm các thông tin khác vào để game có thể đọc được. Nhóm đã làm 1 công cụ trực quan giúp làm được điều đó. Tên công cụ là ToolParseMap.

The screenshot shows a software window titled "ToolParseMap" with three main sections:

- Step 1: Choose map file**
  - Choose file: [Browse...]
  - File name: lv1.txt
- Step 2: Generate Enemy Tank**
  - Radio buttons: ☐ Random, ☒ Custom, ☐ Default
  - [Random] button (disabled)
  - Choose level: [1] (dropdown)
  - Total: [20] (input)
  - Medium Tank: [11] (input)
  - Light Tank: [1] (input)
  - Heavy Tank: [5] (input)
  - SuperHeavy Tank: [3] (input)
  - Order: [0123] (input)
- Step 3: Parse map**
  - Parse map: [Parse] button

- Mục tiêu của Tool này:
  - o Đưa vào tổng số lượng tank định sẽ xuất hiện trên map
  - o Số lượng tank định theo từng loại
  - o Thứ tự xuất hiện từng nhóm tank.
  - o Ma trận đã được xử lí
- Chức năng quan trọng nhất của tool này là xử lí ma trận map. Vì ID của ô đầu tiên trong tilesets bằng 1 mà trong khi vẽ map thì index bắt đầu từ 0 nên ta cần trừ giá trị của mỗi phần tử trong ma trận đi 1. Các phần tử ma trận có giá trị bằng 0 (không có đối tượng nào tại vị trí đó) thì ta gán lại giá trị -1 cho chúng. Để hỗ trợ cho việc xét va chạm trong game được dễ dàng, chính xác và nhanh chóng hơn, mỗi phần tử khác 0 trong ma trận đều phải thay đổi giá trị theo mẫu:  

$$\text{Giá trị mới} = \text{Số thứ tự xuất hiện của giá trị} * 100 + (\text{Giá trị cũ} - 1)$$

Ví dụ: Số 4 thứ 144 trên ma trận: 4 -> 14403 (144: số thứ tự, 3:Giá trị cũ - 1)

Trong chương trình, hàm ParseMap() sẽ thực hiện chức năng này.

- Ngoài ra tool còn cho phép người dùng tạo số lượng tank địch theo 3 cách: random, người dùng tự nhập, theo mặc định của game. Ngoài ra thứ tự xuất hiện tank địch sẽ hoàn toàn phát sinh ngẫu nhiên ngẫu nhiên
- File output của tool này cũng là file txt. Ta sẽ dùng file này để làm map cho game.

## **V. ĐÁNH GIÁ MỨC ĐỘ HOÀN THIỆN CỦA GAME:**

- Game được xây dựng hoàn thiện những thành phần cơ bản mà một game cần có
- Các thành phần đã làm được:
  - Có màn hình chính, có menu, có hướng dẫn chơi, có trạng thái kết thúc game cụ thể (khi người chơi hết số mạng hoặc căn cứ bị tiêu diệt thì sẽ thua và bắt đầu lại)
  - Game có 3 màn chơi
  - Các màn chơi được lưu rồi và có khả năng thay đổi nội dung game
  - Game có bonus cho người chơi khi tiêu diệt xe tank địch, có bảng hiển thị thông tin người chơi và màn chơi
  - Game có sử dụng âm thanh
  - Game có cốt truyện rõ ràng giúp người chơi dễ nắm bắt. Trong game có các item để hỗ trợ cho người chơi.
  - Game có AI cho xe tank địch, tạo độ khó cho người chơi.
  - Trong quá trình xây dựng game có xây dựng một tool mapeditor để tiện xử lý các thông tin của map.
  - Game có hiệu ứng tạo cảm giác hứng thú cho người chơi
  - Game có thiết kế cốt truyện thay đổi so với game gốc nhằm tăng thêm độ khó cho người chơi.
- Hạn chế:
  - Chưa áp dụng được kỹ thuật quadtree hay binary tree vào phân hoạch không gian



- Chưa phải là một game large world
- Thuật toán trong game nhìn chung còn tương đối dài dòng.
- Các kiến thức đã học được:
  - Cách sử dụng các thành phần đồ họa trong DirectX như vẽ sprite, cách thiết kế sprite để tạo chuyển động.
  - Cách vẽ text để thể hiện các thông tin
  - Cách sử dụng DirectInput để điều khiển nhân vật
  - Cách sử dụng DirectSound để tạo âm thanh cho Game
  - Cách xét va chạm trong game: thuật toán AABB, thuật toán broad-phasing
  - Sử dụng design pattern để xây dựng một số lớp trong game (singleton pattern)
  - Cách thiết kế xây dựng các đối tượng áp dụng đa hình, kế thừa
- Hướng phát triển:
  - Thêm các item mới cho người chơi
  - Thêm các loại xe tank địch mới tạo độ khó cho người chơi
  - Tạo các màn chơi game large world để tăng hứng thú cho người chơi
  - Thêm các hiệu ứng hình ảnh đẹp hơn
  - Nâng cấp AI để xe tank địch di chuyển thông minh hơn
  - Tối ưu lại cách phân hoạch không gian
  - Tối ưu lại mã nguồn
  - Thêm chế độ hai người cùng chơi

## **VI. KẾT LUẬN CHUNG**

Là một game đơn giản, dễ chơi với cốt truyện quen thuộc dễ dàng cho người chơi nắm bắt có nhiều thành phần tạo sự tò mò cho người chơi, game hướng đến những người có nhu cầu giải trí đơn giản trong khoảng thời gian ngắn sau giờ làm việc, học tập căng thẳng.