Welcome to our Tankwar!!

Here, we want to first show how to compile and run this game,and then introduce deeper the pattern and paradigm we use, after that, we'd like to share some isuues and bugs' solving strategy we use during coding.

How to compile and run this tankwar game?

- compile the game:

  The whole game consists of three files named layer1.h, layer2.h, and layer3.cpp. So you need to first get all three files in the same project in your compiler and add this following sentence in CMakeLists:

```
add_executable(layer3 layer3.cpp layer1.h layer2.h)
```

Then you are totally ok to run the layer3.cpp to enjoy the game.

- run the game:

  The whole game uses commandline argument so that you need to set some information before running it.

  ```
  ./layer3 --mode=PVP --initial-life=5
  ```

  Here is a sample set before running the file, the mode setting and initial life settig are compulsory, or the game can't run successfully. You can print:

  ```
  ./layer3 -h
  ./layer3 --help
  ```

  in the program to see some more detailed settings.

  ## Layered Architecture Pattern we use

Just as the name of each three files, we decompose the whole program into three layers, and each serves different functions and usages.

- layer1: 1.basic definitions of the `Class Tank`, `Class Bullet` 2.declarations of their functions.
- layer2: main functions setting of the program:
  - **for tank**: 1.function to initialize 2.function to turn and move tanks 3.function to judge collision 4.function to shoot bullet
  - **for bullet**: 1.function to move 2.function to judge collision
  - **for the whole game**: 1.function to judge winning 2.function to shrink the border 3.function to generate AI behaviour 4.function to plot the map
- layer3:

  the main program to run the game:

  1. the procedure to read in command-line argument and adjust them to the initial situation
  2. the different setting of different mode(PVP,PVE,DEMO)

Object-Oriented Paradigm we use

In this game, two classes are enough to realiza the OOP, but in order to ease the setting and change of position and direction, we difine three classes each named `Vec`, `Tank` and `Bullet`.

- **`Vec`**

```
class Vec:
int x;int y;      //two int numbers representing x and y axis
```

- **`Tank`**

```
class Tank:
Vec Pos;            //instant location
Vec direction;      //instant direction
int life_point;     //instant lifepoint
void initialize();  //function to initialize tank
void move();       //function to move
void turn();       //function to turn
```

- `Bullet`

```
class Bullet:
Vec Pos;           //instant location
Vec direction;     //instant direction
```

Issue and Bug we met during coding

1.