第三次课程设计

181860154 朱倩

一、设计内容：坦克大战GUI版本

1. 设计目标

保留课程设计二完成的坦克大战控制台版本的所有功能，用Qt实现图形界面

以下为课程设计二中实现的功能（略长，可跳过）：

1）模仿设计四类游戏模型/属性：

- 游戏地形
    - 砖墙：除四周围墙外均可被一级子弹打穿
    - 海水：坦克和子弹均不可穿越
    - 铁墙：可被二级子弹打穿
    - 森林：坦克和子弹均可穿越，但显示上会被森林覆盖
    - 基地：“★”表示基地中心，被打中则游戏结束
- 游戏道具
    - 坦克：玩家可以增加一条命。
    - 炸弹：全屏敌人遭到爆炸攻击，全数毁灭。
- 三种坦克属性：
    - 生命值：即HP，表示坦克可以被子弹命中多少次
    - 攻击力：一方面表现为子弹种类，一级子弹只能打穿砖墙，二级子弹可打穿铁墙；另一方面表现为子弹射速，即子弹的移动速度
    - 移动速度：坦克的移动速度
- 五种类型的坦克：
    - 轻型坦克：移动速度普通，一级子弹，子弹射速普通，生命值为1
    - 装甲车：移动速度快，一级子弹，子弹射速普通，生命值为2
    - 反坦克炮：移动速度普通，二级子弹，子弹射速快，生命值为2
    - 重型坦克：移动速度普通，一级子弹，子弹射速普通，生命值为3
    - 玩家坦克：移动速度普通，一级子弹，子弹射速普通，生命值为10

2）实现基本游戏逻辑：

- 按照一定的策略刷出敌军坦克，敌军坦克随机移动并攻击。玩家控制的坦克在基地旁边刷出
- 玩家控制的坦克通过灵活移动保卫基地不被摧毁并消灭敌军坦克
- 游戏过程中在地图上会随机刷出道具，玩家可通过道具强化自己
- 失败条件：基地被摧毁或者玩家生命耗尽
- 胜利条件：消灭所有敌军坦克

3）实现当前局的状态栏

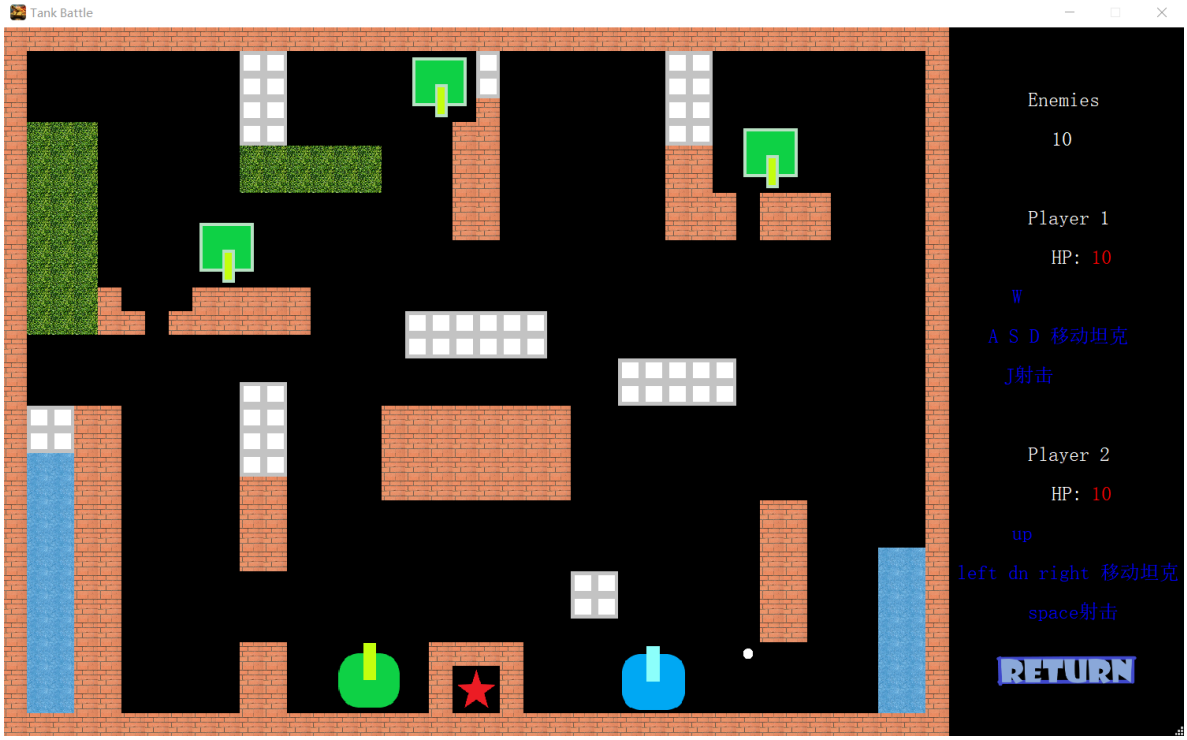
- 显示玩家生命值、敌军剩余坦克数量
- 游戏结束时显示：
    - 胜负：赢了显示“Win!"，输了显示”Game Over"；
    - 得分：根据玩家消灭的敌方坦克的种类和数量计算得分

4）支持双人模式

- 游戏开始时选择模式，游戏结束时显示计分板

2. 基本实现效果

初始界面：



二、设计思路

1. **窗口交互和图形界面的实现**

   1）**窗口的实现：**

   设计了两个窗口类：MainWindow和subWindow类

   MainWindow负责游戏初始界面，subWindow负责游戏开始后的界面

   窗口类可以在ui文件中可视化添加控件，并且可以接收控件点击消息、键盘消息等，可以通过为窗口类添加相应事件处理函数来实现交互。

   ```
   void keyPressEvent(QKeyEvent *ev);//键盘事件处理函数
   void on_pushButton_clicked();//pushbutton被按下处理函数
   ```

   2）**图片和文字的显示**：

   主要通过QPainter实现图片和文字的绘制，在以下函数中实现：

   ```
   void paintEvent(QPaintEvent *event);
   ```

   关于地图的绘制：这里将界面划分为一个个小格，通过map数组记录每个小格需要显示的图片，所以在函数paintEvent中只需读取map数组中的数据即可知道在相应位置应该绘制哪张图片。坦克，子弹或地形发生改变时，调用函数`update()`即可实现地图的更新。

   3）**界面的切换**：

   subWindow中会涉及多个界面，如模式选择界面、游戏进行时的界面、游戏结束后的计分板界面。这里通过设置flag标志位让paintEvent函数知道应该读取哪些数据来绘制界面。例如：游戏结束条件触发时设标志位endflag为1，表示游戏结束，此时paintEvent函数读取玩家击败敌方坦克的数量和分数，并在相应位置显示。

2. **游戏逻辑的控制**

   原来的控制台版本通过设计`Control`类来控制游戏的初始、进行和结束；但在Qt中因为键盘消息、时钟信号都由窗口类捕捉，所以把游戏逻辑控制部分移到了`subWindow`类中。

   1）**地图上的坦克、子弹变量：**

   在`subWindow`类的数据成员中存储着地图上现存的坦克指针（包括地方坦克和玩家坦克），而`Tank`类的数据成员中有其控制的子弹，通过遍历这些坦克及其控制的子弹并调用其move函数实现这些模型的运动。

   约定同一时刻存在的敌方坦克数量最多为3，当有敌方坦克被摧毁并且有备用坦克时，刷新新的敌方坦克。玩家被击中时退回其出生地。

   2）**坦克、子弹的移动：**

   通过设置几个时间间隔不同的时钟QTimer，并将时钟信号与槽函数..._move连接，每隔一段时间调用坦克或子弹的move函数，实现坦克和子弹移速的控制。

   3）**整体逻辑的控制：**

   最最主要的游戏逻辑控制在`subWindow`中的`bullet_move`函数中，因为坦克的消亡、玩家坦克血量的减少、基地被摧毁等与游戏结束触发条件相关的事件均由子弹引发。

   而当基地被摧毁、现存地方坦克数量为0、玩家生命值耗尽时游戏结束，设endflag为1，paintEvent函数会显示胜负情况，计算并显示玩家得分情况。

复用课程设计二的部分：

1. **地图的设计**

   通过int型的二维数组`map`存储地图上的模型信息，不同的数字表示占用当前位置的不同游戏模型，其中

   1~10表示地形：空地，砖墙，铁墙，基地，海洋，森林等

   10~20表示敌方坦克的id

   20~30表示玩家坦克的id

   30~40表示敌方坦克射出的子弹id，减去20即可得敌方坦克id

   40~50表示玩家坦克射出的子弹id，减去20即可得玩家坦克id

   （以上数字区间中不一定所有数都被用到）

   通过这样的区间约定可以有效判断子弹或坦克移动时遇到的是哪一类游戏模型并作出相应的反应

2. **坦克的设计**

   坦克可以分为两类：敌方坦克和玩家坦克，这两类坦克有很多共同属性和功能，但又有不同之处（如：敌方坦克通过一定策略自动移动射击，玩家坦克通过键盘输入移动射击）。所以设计了`Tank`基类来包含它们的共同属性和功能，在其派生类`Enemy`和`Player`中对不同之处做各自的调整。

   - **坦克的移动**

     通过`void Tank::basic_move();`函数实现。根据坦克目前的方向决定下一刻的坦克坐标，将之前坦克所在位置区域对应的map[i]置零，并打印空格；根据目前坦克位置打印坦克模型。

     ```
     void print_tank();//在地图上打印坦克
     void delete_tank();//使坦克在地图上消失
     ```

- **坦克的射击**

  通过`void Tank::shoot();`实现，约定一个坦克在同一时刻只能拥有一颗子弹，所以只有当坦克的数据成员`bullet`为NULL时，才可以调用shoot函数产生新子弹，并根据坦克id决定子弹id，达到可通过子弹id知道其坦克id的目的。

- **坦克的属性**

  坦克的移速，其射出的子弹的移速都通过调节其sleeptime来控制，sleeptime越大，调用其move函数的时间间隔越长，表现为其移动速度越慢。

3. **子弹的设计**

   - **子弹的移动**

     子弹一直沿坦克射出子弹时的方向移动，直至碰到障碍物时子弹消亡。

     如果是地形障碍，根据地形种类和子弹种类决定该地形障碍是否消失；如果是坦克，将其id记录在bullet类的数据成员`hurt_id`中，坦克生命值减一，生命值为0时该坦克死亡；如果是子弹，也将其id记录在`hurt_id`中，并令该子弹消亡。

三、实现细节

1. 类的设计

共设计了七个类：

`MainWindow`：主窗口

`subWindow`：子窗口

`Bullet`,：子弹

`Tank`：坦克基类

`Enemy`：地方坦克类（Tank派生类）

`Player`：玩家坦克类（Tank派生类）

`Map`：地图类

类的声明：

mainwindow.h:

```
class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    MainWindow(QWidget *parent = nullptr);
    ~MainWindow();

private slots:
    void on_pushButton_clicked();
    void on_pushButton_2_clicked();

private:
    Ui::MainWindow *ui;
```

```
};
```

subwindow.h:

```
class subWindow : public QMainWindow
{
    Q_OBJECT

public:
    explicit subWindow(QWidget *parent = nullptr);
    ~subWindow();

    void paintEvent(QPaintEvent *event);
    QPainter painter;
    void init();//初始化
    void gameover();

private slots:
    void on_returnclick_clicked();
    void tank_move_slow();
    void tank_move_fast();
    void bullet_move_slow();
    void bullet_move_fast();
    void bullet_move(int speed);
    void keyPressEvent(QKeyEvent *ev);
    void showgrades();
    void on_pushButton_clicked();
    void on_pushButton_2_clicked();

signals:
    void sendsignal();
private:
    Ui::subWindow *ui;

    Enemy* enemies[3];//地图上现有的敌方坦克
    Player* players[2];//玩家坦克
    int enemies_num;//剩余的敌方坦克数量
    int new_num;//新构造的敌方坦克数量
    int players_num;//目前存活的玩家数量
    int score;//玩家1得分
    int score_2;//玩家2得分
    int mode;//1单人模式,2双人模式
    int kill[4];//玩家1击杀敌方坦克情况
    int kill_2[4];//玩家2击杀敌方坦克情况
    int tools[5][2];//道具坐标
    int tools_exist[5];//道具存在情况
    bool startflag;
    bool endflag;
    bool gradesflag;
    bool winflag;
};
```

tankwar.h:

```
class Bullet
{
    int pos_x;//子弹当前横坐标
    int pos_y;//子弹当前纵坐标
    int pre_x;//子弹之前横坐标
    int pre_y;//子弹之前前纵坐标
    int sleep_time;//子弹移动的间隔时间
    int dir;//子弹移动方向
    int type;//子弹种类
    int master;//子弹所属的坦克
    int hurt_id;//子弹命中目标id
public:
    Bullet();//构造函数
    bool move();//控制子弹的移动
    void delete_bullet();//子弹在地图上消失
    friend class Tank;
    friend class subWindow;
};

class Tank
{
protected:
    int type;//种类,1轻型坦克,2装甲车,3反坦克炮，4重型坦克
    int dir;//移动方向
    int pre_dir;//之前的移动方向
    int life;//生命值
    int atk;//攻击力
    int move_sleep_time;//移动速度
    int bullet_sleep_time;//子弹速度
    int pos_x;//横坐标
    int pos_y;//纵坐标
    int id;//坦克id
    Bullet* bullets;//射出的子弹
public:
    Tank();
    Tank(int t, int x, int y, int i);//构造函数
    ~Tank()//析构函数
    {
        delete bullets;
    }
    void basic_move();//控制坦克的移动
    void shoot();//射出子弹
    void delete_tank();//使坦克在地图上消失

    friend class subWindow;
};

class Map
{
    static int *map;//存储地图数据
public:
    Map();//地图的初始化
    friend class subWindow;
    friend class Tank;
    friend class Player;
    friend class Enemy;
    friend class Bullet;
```

```
};


class Enemy :public Tank
{
public:
    Enemy(int t, int x, int y,int i) :Tank(t, x, y,i)//构造函数
    {
        dir = down;
        pre_dir = dir;
        switch (t)
        {
        case 1:
            life = 1;
            atk = 1;
            move_sleep_time = 300;
            bullet_sleep_time = 80;
            break;
        case 2:
            life = 2;
            atk = 1;
            move_sleep_time = 200;
            bullet_sleep_time = 50;
            break;
        case 3:
            life = 2;
            atk = 2;
            move_sleep_time = 300;
            bullet_sleep_time = 50;
            break;
        case 4:
            life = 3;
            atk = 2;
            move_sleep_time = 300;
            bullet_sleep_time = 80;
            break;
        }
    }
    void move();//根据一定策略控制敌方坦克的移动
    friend class Bullet;
    friend class Control;
};

class Player :public Tank
{
public:
    Player(int t,int x, int y,int i) :Tank(t, x, y,i)//构造函数
    {
        dir = up;
        pre_dir = dir;
        life = 10;
        atk = 1;
        move_sleep_time = 300;
        bullet_sleep_time = 80;
    }
    void move();//根据玩家的键盘输入控制玩家坦克的移动
    friend class Bullet;
```

```
};
```

2. OOP设计思想

> OOP=对象+类+继承+多态+消息，其中核心概念是类和对象。

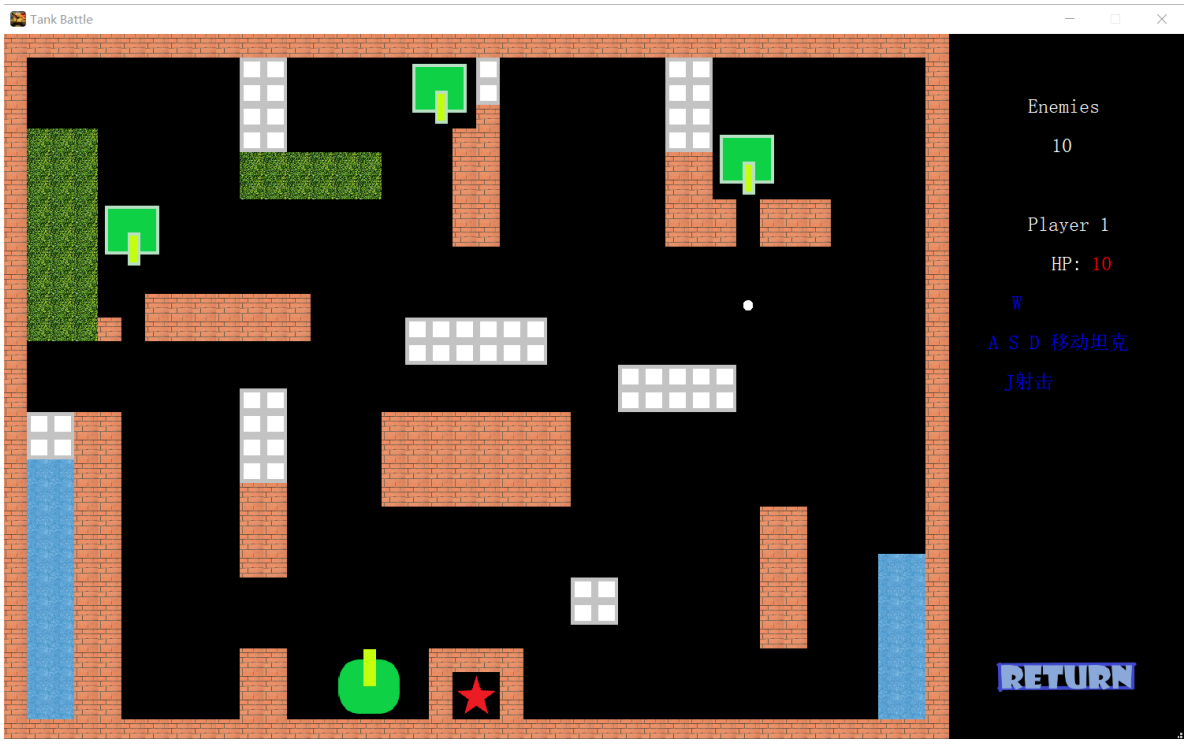
本次课程设计主要用到的是对象+类+继承+消息，所有功能都通过设计相应的类进行实现，没有全局变量或全局函数。

四、功能介绍

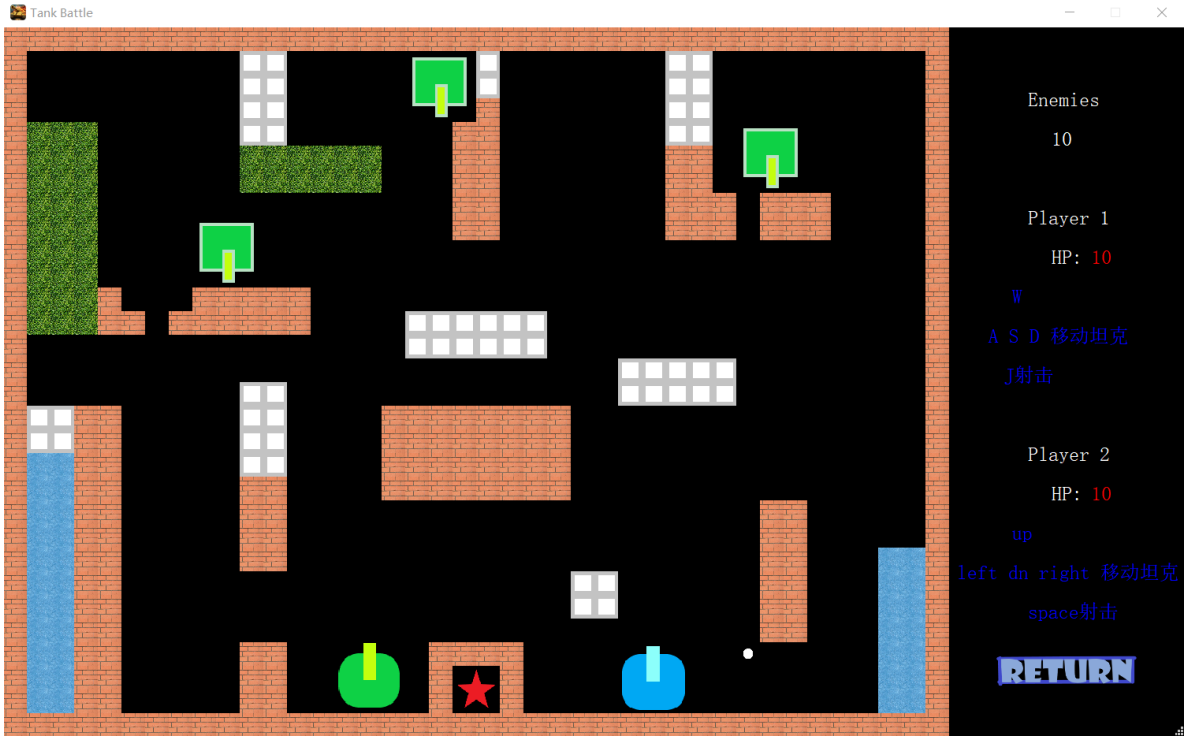
游戏初始界面：

鼠标左键点击"**START**"键进入模式选择：（或点击“**EXIT**”键退出程序）

鼠标左键点击"**1 PLAYER**"键进入单人模式：

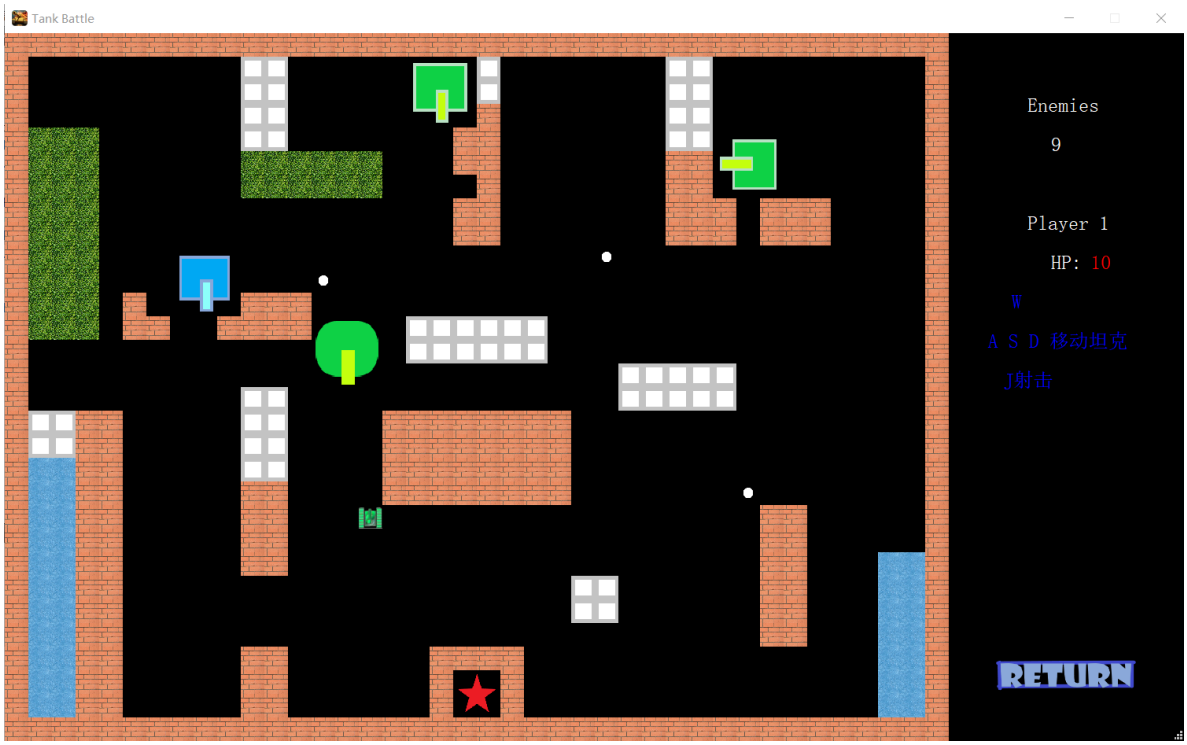


点击"**2 PLAYER**"键进入双人模式：



> 注：点击右下方的“**RETURN**"键可返回模式选择界面

游戏过程中可掉落游戏道具（以坦克道具为例）：



游戏结束后显示胜负和得分，（胜：WIN! 负：GAME OVER)，点击“RETURN"键重新开始。

五、遇到的问题和解决方案

1. **Qt中库函数的使用**

   大部分在网上都可以查到，这次课程设计用到了很多，经验值++

2. **代码复用问题**

   虽说只是增加了图形界面，原来的游戏逻辑没变，但这次修改还是花了很大力气。主要还是因为最复杂的游戏逻辑那块（原本通过control类实现），原有的实现机制在Qt中用不了，只能全部打散重写了。

   原来的实现方法：

   坦克和子弹都有sleeptime这一属性，表示隔多长时间调用一次其对应的move函数，从而实现其移速的控制。

   可以通过类似以下的方式实现不同移速：

```
int count=0;
Tank tank;
Bullet bullet;
while(1)
{
    if(count%tank.sleeptime)
        tank.move();
    if(count%bullet.sleeptime)
        bullet.move();
    Sleep(1);
    count++;
}
```

   这样sleeptime越长，其move函数被调用的频率越低。

但在Qt中这样的实现方式并不行，while循环不结束，地图就不会显示出来。（也不知道为什么）

但其实Qt有自身的信号槽机制可以实现上述程序所实现的功能。（不过也是发现上述问题才去了解了）

3. **图片素材**

   找不太到合适的坦克图片，所以通过系统自带的画图3D软件画几何图形实现，略丑。不过3D字体的效果还是很好的。

Thanks for reading!