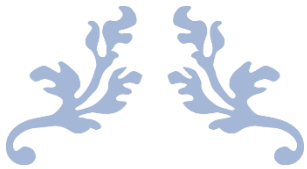


说明文档

程设大作业 TankBattle

2021-6-5 整合

YBB 出品

1 选题与需求分析报告

1 选题背景意义：8%（具体说明为什么选这个题，有什么理论和实践意义，至少简单说明有什么实用性）

近些年来，随着电子计算机的发展，电子游戏，也称电玩游戏得到了极大的发展。完善的电子游戏出现在二十世纪末，它改变了人类进行游戏的行为方式和对游戏一词的定义，逐渐变为属于一种随科技发展而诞生的文化活动。在它诞生的短短几十年间，游戏的内容，操作，页面以及终端设备都产生了翻天覆地的变化。从对传统游戏的继承，比如足球、棒球、国际象棋的电子游戏，到对虚幻世界的探索，比如幻想世界的冒险、战争、格斗等；从简单的按键操作，到摇杆，VR 等多种方式连接；从小小的插卡游戏机，到现在的智能手机，家用电脑等多种设备。电子游戏以其独有的魅力吸引着各种人群。而**坦克大战**就在这电子游戏不长的历史中堪称经典。

《坦克大战》是由日本南梦宫 Namco 游戏公司开发的一款平面射击游戏，于 1985 年发售。游戏以坦克战斗及保卫基地为主题，属于策略型联机类。它最早出现在 FC 机上，但为了进一步扩大受众范围，它随后也安装在掌上俄罗斯方块机上，虽然功能有所简化，但还是在包括笔者在内的广大小学生中掀起了一股坦克大战的热潮。它规则简单，容易上手，且游戏过程变化无穷，使用户既能感受到游戏中的乐趣，也给用户提供了一个展现自己高超技艺的场所。同时低廉的价格也足以让小学生们鼓起勇气向父母开口寻求帮助。可以说，在掌上俄罗斯方块机上的坦克大战，已经成为了专属于那个时代小学生们共同的童年记忆，是弥足珍贵的文化遗产。

但遗憾的是，无论是在杂志还是网络上的相关资料，关于游戏机的历史却对它只字未提。同时，当前网络上以“坦克大战”为名的游戏多数已经无法找到童年时期游戏的影子；对于小部分游戏，即使含有相关元素，也不再会是黑白方块模式，这给用户找回童年的回忆造成了小小的缺憾。

2 同类软件调研分析：4%（如果查到同类软件，要求分析其特点、不足之处，以及自己要在哪些方面做出新意。但是不要求达到和超过同类软件的优点，只要自己有新意就行；如果找不到同类软件，要说明用了什么检索手段、检索关键词）

在网络上现有的“坦克大战”软件中，主要分为广义坦克大战游戏和经典 90 坦克大战。

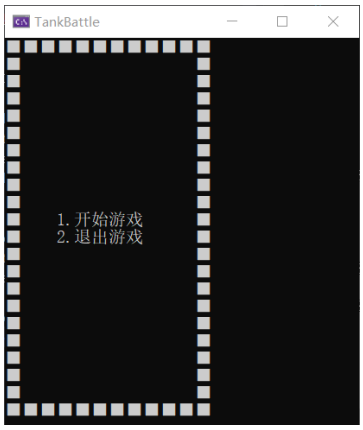
在广义坦克大战游戏中包含多款游戏，但都以坦克为题材，在保留了射击类游戏操作的同时，也改进了射击类游戏设置，加入了多种车型的选择和相关历史背景，强调游戏的真实性和多样性。游戏中，既可以组队挑战各种丰富的剧情战役，也可以与其他玩家进行多人对抗，更可以参加数千人规模的大型比赛。培养人物、打造坦克，RPG 的成长和养成带给休闲游戏更多乐趣。这对于原来的“坦克大战”来说无疑是巨大的飞跃。

而经典 90 坦克大战则试图回归传统，简化了游戏设置，只保留基本的游戏界面的游戏规则，在很大程度上与经典版本相一致。但是，目前网络上坦克大战的复原多数是以 1990 年前后 FC 版为范本的，这与很多用户所习惯的黑白方块坦克大战有一定的区别。所以现在要开发的程序是对网络上现有的经典 90 坦克大战的进一步抽象，在简化界面和规则的同时，也尽可能贴近俄罗斯方块机的效果，最终实现对游戏的完全复原。

3 详细的功能设计：8%（达到使用说明书的详细程度）

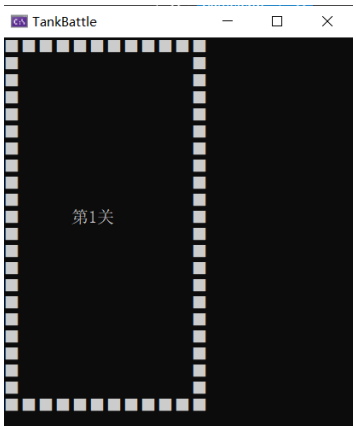
大致篇幅建议：A4 纸 5 号字单倍行距，3 页以上，图表代码都算（篇幅不够，但是写的很清楚的，也可以得好成绩）。

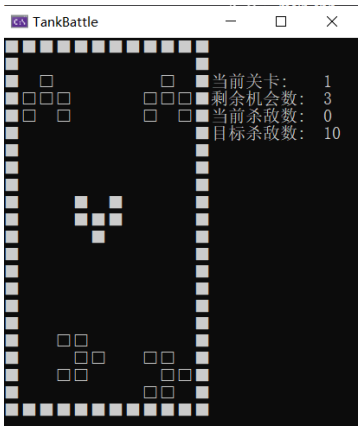
———— 开始页面 ————



出现方式：控制台标题显示“Tank Battle”，出现边框，
暂停两秒，显示选择页面；
游游戏选择：有两个选项（1.开始游戏；2.退出游戏；）
选择方式：分别按“W”，“S”键上下调整选项
（当前选项显示为高亮）
按“J”键确认当前选项；
开始游戏：选中“开始游戏”选项，
按“J”键确认，进入游戏界面；
退出游戏：选中“退出游戏”选项
按“J”键确认，程序结束。

———— 游戏页面 ————





页面布局：左侧为游戏区，右侧为记录区；
游戏元素：
己方坦克：初始位置位于中间
玩家可移动及发射炮弹；
敌方坦克：在画面四角随机出现
游戏时随机移动及发射炮弹；
炮弹：白色圆球
玩家坦克发射炮弹击中敌方坦克，坦克消失；
敌方坦克发射的炮弹击中己方坦克，玩家淘汰；
记录元素：
当前关卡：记录当前所处的关卡；
剩余机会数：记录玩家挑战游戏通关剩余的机会数；
当前杀敌数：记录当前关卡内玩家的杀敌数；
目标杀敌数：显示玩家通过此关卡所需要的杀敌数；
关卡转换：每通过一个关卡，清除坦克和子弹
游戏界面中心位置显示下一关卡，即“第 n 关“

———— 游戏规则 ————

坦克操控：“W”“S”“A”“D”键分别操控坦克上下左右移动，“J”键发射炮弹；

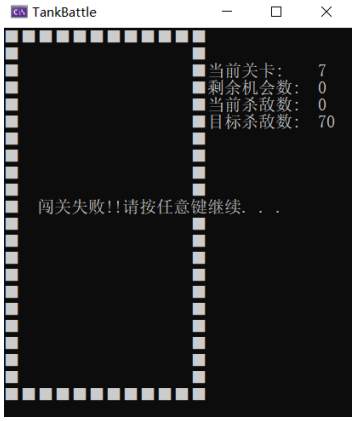
杀敌数增加：发射炮弹，击中敌方坦克则杀敌数 +1；

过关条件：杀敌数达到当前关卡的目标杀敌数，则成功过关；

通关条件：成功闯过五个关卡，即为通关；

机会数减少：被敌方炮弹击中己方坦克则玩家机会数 -1；

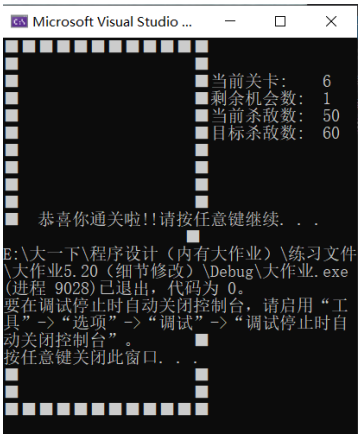
游戏结束：玩家机会数减至零（即被敌方坦克击中三次）则游戏结束。



———— 结结束画面 ————

闯关失败：若玩家机会数减至零，则显示“闯关失败!! ”

游戏通关：若玩家顺利通关，则显示"恭喜你通关啦!! "



2 设计说明书

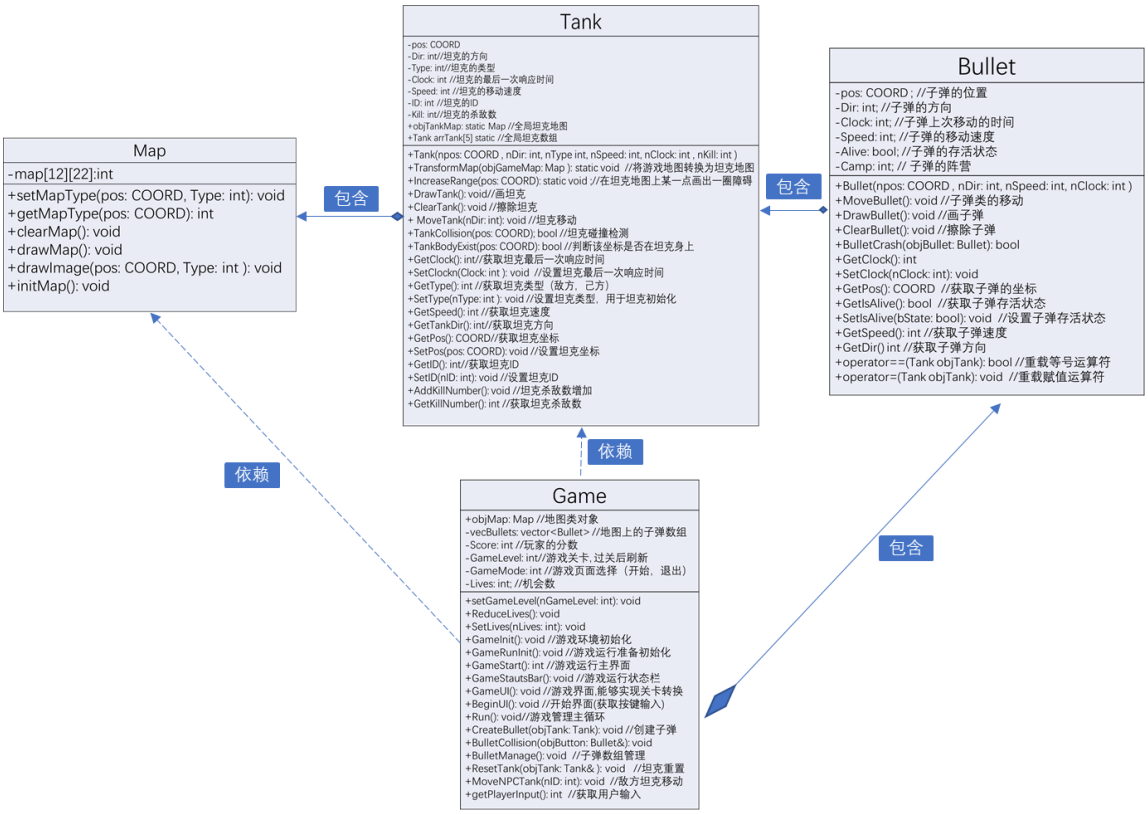
1 类设计

1.1 类图：

（类的成员、类与类之间的关系要画清楚）

（1）类的功能：Map 类实现地图的相关操作；Tank 类记录坦克的信息并实现坦克的相关操作；Bullet 类记录子弹信息并实现子弹的相关操作；Game 类实现游戏进程。

（2）类的关系：Tank 类包含 Map 类，Bullet 类包含 Tank 类，Game 类包含 Bullet 类；



1.2 类的成员说明

每个类的数据成员说明：要说明存储什么数据，数据的含义、用途

每个类的函数成员说明：说明函数原型、主要功能

1.2.1 Map 类

（1）数据成员：

-map[12][22]: int //储存地图信息，值的不同代表地图上对应点类型的不同

（2）函数成员：

+setMapType(pos: COORD, Type: int): void //载入地图类型，修改地图数组

+getMapType(pos: COORD): int //读取地图数组在某点的值

+clearMap(): void //清除地图，重画边界（对数组没有改变）

+drawMap(): void //根据地图数组 画出整个地图

+drawImage(pos: COORD, Type: int): void //在指定位置根据类型输出相关元素
+initMap(): void //初始化地图（给数组赋值，加上边框）

1.2.2 Tank 类

（1）数据成员：

-pos: COORD //记录坦克的坐标
-Dir: int //坦克的方向
-Type: int //坦克的类型(玩家坦克 PLAYER_TANK，敌人坦克 ENEMY_TANK)
-Clock: int //坦克的最后一次响应时间，用于控制坦克的速度
-Speed: int //坦克的移动速度
-ID: int //坦克的 ID
-Kill: int//坦克的杀敌数
+objTankMap: static Map //全局坦克地图
+arrTank[5]: static Tank //全局坦克数组，包含所有坦克，第一个下标是玩家

（2）函数成员：

+Tank(npos: COORD , nDir: int, nType int, nSpeed: int, nClock: int , nKill: int)//构造
+TransformMap(objGameMap: Map): static void　//将游戏地图转换为坦克地图
+IncreaseRange(pos: COORD): static void ;//在坦克地图上某一点画出一圈障碍
+DrawTank(): void//画坦克
+ClearTank(): void //擦除坦克
+ MoveTank(nDir: int): void //坦克移动
+TankCollision(pos: COORD); bool //坦克碰撞检测
+TankBodyExist(pos: COORD): bool //判断该坐标是否在坦克身上
+GetClock(): int//获取坦克最后一次响应时间
+SetClockn(Clock: int): void　//设置坦克最后一次响应时间
+GetType(): int //获取坦克类型（敌方，己方）
+SetType(nType: int): void //设置坦克类型，用于坦克初始化
+GetSpeed(): int //获取坦克速度
+GetTankDir(): int//获取坦克方向
+GetPos(): COORD//获取坦克坐标
+SetPos(pos: COORD): void //设置坦克坐标
+GetID(): int//获取坦克 ID
+SetID(nID: int): void //设置坦克 ID
+AddKillNumber(): void //坦克杀敌数增加
+GetKillNumber(): int //获取坦克杀敌数

1.2.3 Bullet 类

（1）数据成员：

-pos: COORD ; //子弹的位置
-Dir: int; //子弹的方向
-Clock: int; //子弹上次移动的时间
-Speed: int; //子弹的移动速度
-Alive: bool; //子弹的存活状态
-Camp: int; // 子弹的阵营，用于避免敌方坦克自相残杀

（2）函数成员：

+Bullet(npos: COORD , nDir: int, nSpeed: int, nClock: int) //构造一个子弹

+MoveBullet(): void //移动子弹
+DrawBullet(): void //画出子弹
+ClearBullet(): void //擦除子弹
+BulletCrash(objBullet: Bullet): bool //子弹碰撞检测
+GetClock(): int //获取子弹上次移动的时间（用于控制子弹的速度）
+SetClock(nClock: int): void //设置子弹时间，每次移动后则重新设置
+GetPos(): COORD　//获取子弹的坐标
+GetIsAlive(): bool　//获取子弹存活状态
+SetIsAlive(bState: bool): void　//设置子弹存活状态
+GetSpeed(): int //获取子弹速度
+GetDir() int //获取子弹方向
+operator==(Tank objTank): bool //用为判断子弹与坦克是否是同一阵营
+operator=(Tank objTank): void　//将坦克的属性传递给发射的炮弹

1.2.4 Game 类

（1）数据成员：

+objMap: Map //地图类对象，里面只保存障碍物信息
-vecBullets: vector<Bullet> //地图上的子弹动态数组
-Score: int //记录玩家的分数
-GameLevel: int//游戏关卡，过关后刷新
-GameMode: int //游戏页面选择类型（开始，退出）
-Lives: int; //机会数，用于控制玩家失败的条件

（2）函数成员：

+setGameLevel(nGameLevel: int): void //设置当前关卡数，用于初始化
+ReduceLives(): void //减少玩家的机会数
+SetLives(nLives: int): void //设置玩家机会数
+GameInit(): void //游戏环境初始化，即设置控制台，改变输入法，隐藏光标
+GameRunInit(): void //游戏运行准备初始化，即初始化坦克
+GameStart(): int //开始每一关的主循环
+GameStatusBar(): void //游戏运行状态栏刷新
+GameUI(): void //游戏界面,能够实现关卡转换
+BeginUI(): void //开始界面(获取按键输入)
+Run(): void//游戏管理主循环
+CreateBullet(objTank: Tank): void //创建子弹，(参数为开炮的坦克)
+BulletCollision(objButton: Bullet&): void //子弹碰撞处理
+BulletManage(): void　//子弹数组管理（清除无效子弹)
+ResetTank(objTank: Tank&): void //根据传入坦克的参数决定重置或初始化)
+MoveNPCTank(nID: int): void　//敌方坦克随机移动
+getPlayerInput(): int　//获取用户输入，返回值为上下左右开火

2 主要技术难点及实现方案/算法设计

这一项要详细写，要体现你作业的亮点，不仅占文档分，而且对答辩得分有利。

2.1 游戏逻辑结构：

这个小游戏流程的**逻辑结构比较复杂**，既要考虑整体的关卡循环，也要考虑关卡内部的循环，这点给我造成了很大的困扰。最后，我将整体游戏的逻辑结构设计为以下几个层次：

（1）程序主流程：主要利用 Run()函数实现：首先初始化机会数，使之在后续的循环中单向改变；然后进入选择界面，即 BeginUI()函数实现的部分，若返回值为 1，则进入正式游戏界面，由 Game()函数实现；

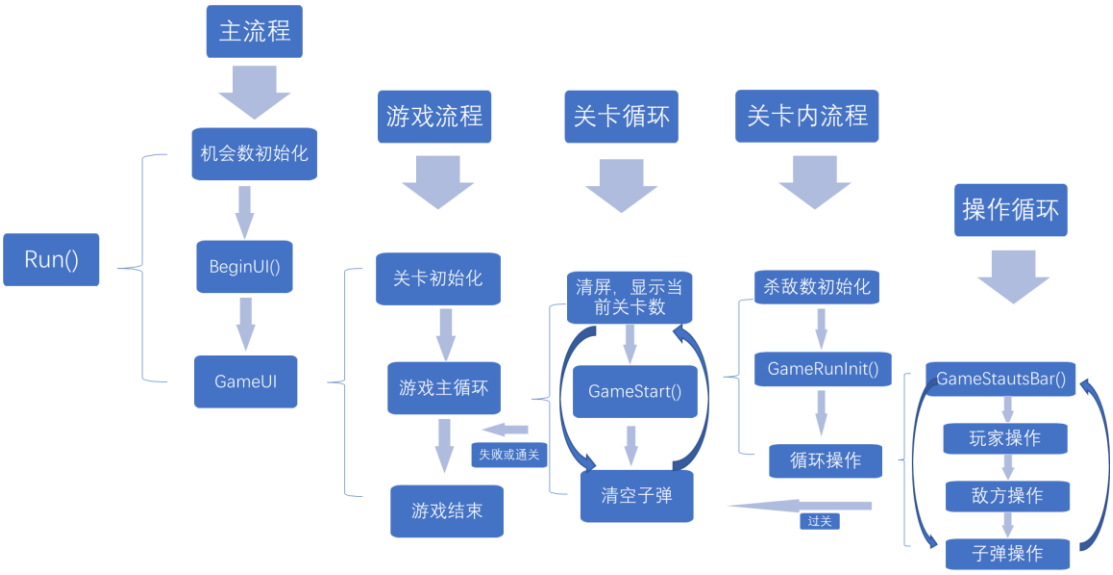
（2）游戏流程：利用 GameUI()函数实现：首先初始化关卡；然后进入游戏主循环，实现关卡的转化，即从第一关到第五关；最后根据主循环返回值的不同，输出不同的结束界面，结束程序；

（3）关卡循环：仍然在 GameUI()函数内实现：首先清屏，在屏幕中心显示当前关卡；然后进入 GameStart()函数，开始当前关卡的游戏，如果返回值为 1~5，则清空子弹数组，继续循环；如果返回值为 6（表示通关）或 7（表示失败），则退出循环，游戏结束；

（4）关卡内流程：利用 GameStart()函数实现：首先初始化杀敌数，然后用 GameRunInit()函数初始化坦克，再进入操作的循环；

（5）操作循环：仍在 GameStart()函数内实现：首先调用 GameStatusBar()函数更新状态栏的数据；然后依次进行玩家操作，敌方操作和坦克操作（若玩家无操作，则直接跳到敌方操作；每次操作都有响应时间控制，若未达到时间也跳过），期间会进行相关检测，若过关或被击杀，则跳出循环。

下图为设计流程图：



2.2 可视化界面：

如何**构建可视化界面**，实现用户和系统的交互也是一个技术难点。我查阅了很多资料，参考了网上几款程序的设计，最终决定使用以下方案：

（1）基础函数：实现可视化界面的最重要的函数是 SetEle()函数，这个函数来源于网络，我对其运作细节不甚了解。但这个函数可以很好地实现在指定位置输出指定字符，这是之后所有算法设计的基础。

```
//在指定坐标输出元素符号(来源于网络)
void SetEle(short x, short y, const char * ele) {
    //获取句柄
    HANDLE hOut = GetStdHandle(STD_OUTPUT_HANDLE);
    //设置坐标位置
    COORD loc;
    loc.X = x * 2;
    loc.Y = y;
    SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE), loc);
    //打印字符串
    printf(ele);
    //画图形
}
```

（2）地图可视化：构建一个二维数组，储存地图信息。1 表示墙壁，输出特殊字符“■”；0 表示空地，输出空格。通过 setMapType()函

数和 getMapType()函数实现对地图数组数据的更改；通过 clearMap()函数和 drawMap()函数实现对地图的可视化。

（3）坦克的可视化：

a. 单个坦克：构建一个四维数组，第一分量表示敌我双方；第二分量表示坦克的朝向；第三分量是行数，第四分量储存每一行的字符串。通过 DrawTank()函数实现对坦克的显示，以坦克中心为基准，在九宫格内输出四维数组内容；通过 ClearTank()函数实现对坦克的清除，以中心基准，在九宫格内输出空格；

b. 全部坦克：构建一个坦克类的对象数组，储存全部五个坦克，使用时利用下标区分不同的坦克；

c. 坦克的移动：首先擦除原有的坦克，然后根据传入方向计算目标位置坐标，然后利用 TankCollision()函数判断坦克能否移动到该坐标，若可以移动则以该坐标为中心画出坦克；若不能移动，则只改变方向，然后再以原坐标为中心画出坦克。

（4）子弹的可视化：

a. 子弹的创建：重载赋值运算符，将坦克对象的特性传递给相应的子弹（包括当前时刻，方向，速度，坐标，阵营），然后用 DrawBullet()画出子弹；

b. 全部子弹：构建子弹类的动态数组，利用 vector 的相关函数实现子弹的管理（如遍历数组，添加数据，清除数据等）；

c. 子弹的移动：与坦克的移动相似。

2.3 子弹阵营判断：

前期设计游戏未考虑子弹的阵营问题，导致敌方坦克自相残杀严重，几乎可以躺赢，所以为了提升难度，我又加入子弹的阵营属性。方案如下：

（1）阵营继承：重载赋值运算符，将坦克对象的阵营传递给创建的子弹；

（2）阵营判断：重载等号运算符，判断传入子弹与坦克是否是同一阵营。

3 用户手册

用户手册

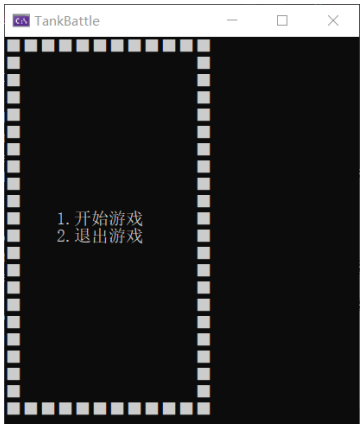
就是系统使用说明书，要使用户仅仅看说明书就能够正确使用你的程序，特别是如果对用户的操作、输入的数据有限制的话，一定要在说明书中写清楚。

1 游戏简介：

本程序为坦克大战模拟游戏，玩家可通过按键控制屏幕中的坦克移动和开火，挑战不同的关卡。在黑白方块模式下，玩家可以找回童年时期在俄罗斯方块机上玩坦克大战的回忆并获得不一样的乐趣。（游戏系仿照俄罗斯方块机，功能有限）

2 操作说明：

2.1 开始界面



1 运行程序，稍等片刻，游戏出现如右主界面：

2 在此界面中可以进行选项的选择：

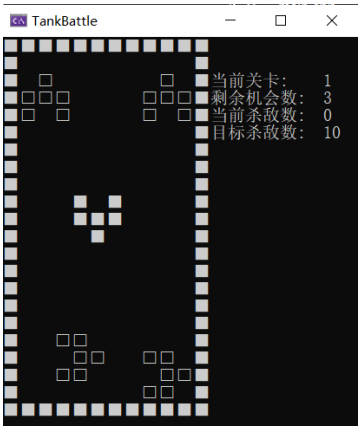
（1）分别按“W”，“S”键上下调整选项；

（2）按“J”键确认当前选项；

3 如需退出游戏，选中“退出游戏”并确认，程序结束；

4 如需开始游戏，选中“开始游戏”并确认，游戏开始；

2.2 元素说明



1 开始游戏后出现游戏界面，左侧为游戏区，右侧为记录区；

2 实心方块组成的坦克为**玩家坦克**，初始位置位于中间
玩家可操控其上下左右移动或开炮；

3 空心方块组成的坦克为**敌方坦克**，在画面四角随机出现
在游戏开始后随机移动并随机发射炮弹；

4 白色圆球为**炮弹**，击中敌方坦克，杀敌数 +1 ，敌方坦克重置；击中己方坦克，玩家机会数 -1 ；

5 右侧为记录元素，会根据玩家当前关卡和状态实时更新；

（1）**当前关卡**：记录当前所处的关卡；

（2）**剩余机会数**：记录玩家挑战游戏通关剩余的机会数；

（3）**当前杀敌数**：记录当前关卡内玩家的杀敌数；

（4）**目标杀敌数**：显示玩家通过此关卡所需要的杀敌数；

2.3 按键操作

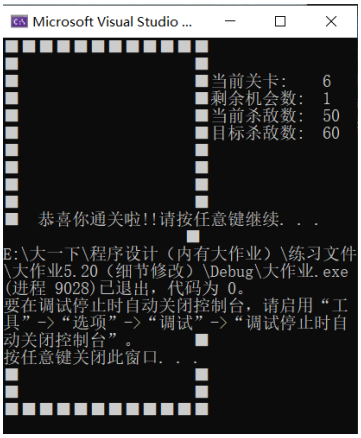
1 按“W”，“S”，“A”，“D”键分别操控坦克上下左右移动；

2 按“J”键发射炮弹；

2.4 游戏规则

1 开始游戏，移动坦克躲避敌方坦克发射的炮弹，
发射炮弹努力击中敌方坦克；
2 玩家发射的炮弹每击中一辆敌方坦克，杀敌数 +1 ；
3 杀敌数达到目标杀敌数，玩家成功过关，继续下一关游戏；
4 每关目标杀敌数有所不同，将会在右侧记录区更新；
5 玩家坦克被敌方炮弹击中，玩家机会数 -1（共三次机会）

注：炮弹迎头碰撞后将双双失去威力，遗留炮弹壳在碰撞位置。

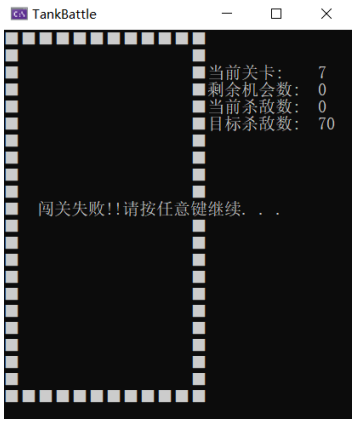


2.5 玩家通关

1 当玩家成功闯过五个关卡，即为通关；
2 若玩家顺利通关，则显示"恭喜你通关啦!！"，如右图：

2.6 闯关失败

1 玩家机会数减至零（即被敌方炮弹击中三次），游戏结束；
2 界面显示“闯关失败!！”，如右图：



2.7 结束游戏

玩家通关或闯关失败后，按任意键即可结束程序。

3 注意事项

1 打开可执行文件后，请稍等片刻，不要急于点击鼠标或按键，以免出现未知错误；
2 尽量**避免让坦克到主界面四角**，否则有可能会让敌方坦克无法刷新而造成游戏停止运行；
3 坦克操作有响应时间，尽量不要按住操作键不放，可能会造成程序异常；
4 炮弹迎头碰撞后圆球可能会残留，不必担心，后续的操作会将其清除；
5 若玩家操作速度过快，则游戏结束时将来不及显示结束页面而直接退出程序，请玩家注意。

5 测试用例

5. 测试用例（必须有，不计分）

在程序完成以后，一定要进行测试，要用不同的数据（或不同的操作流程）对程序进行反复测试，每一个函数都要被执行过，函数中的每一个分支都尽量执行到。

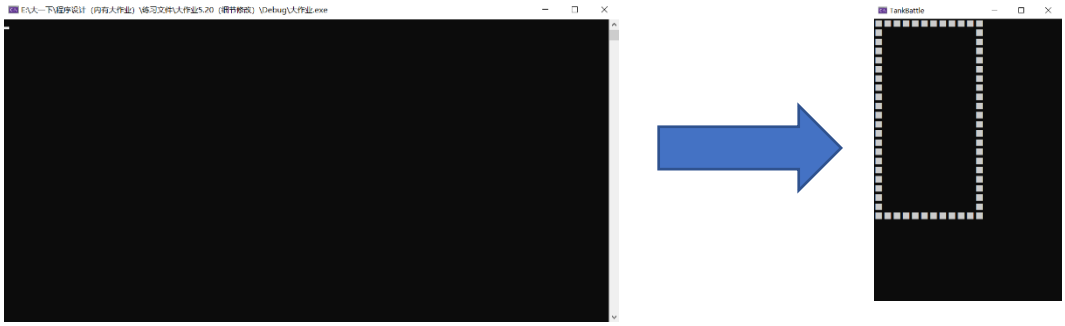
每次测试时使用的数据叫做测试用例，在测试时请记录用例，以及每次测试结果。这样便于查错，也自然记录了需要提交的测试用例。

最后要为答辩演示准备好演示用的数据。

1 初始化测试：

测试操作：运行程序，等待片刻

测试效果：隐藏光标，切换输入法，控制台大小改变，标题显示为“Tank Battle”；



2 开始界面测试：

（1）主页面出现：

测试操作：运行程序，等待片刻

测试效果：界面出现“1.开始游戏”，“2.退出游戏”两个选项

（2）开始游戏

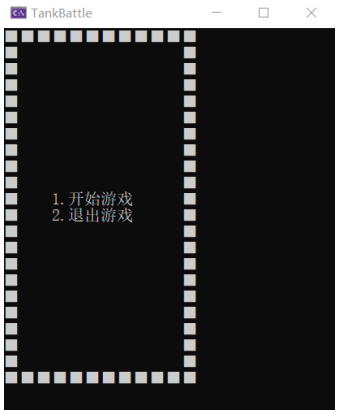
测试操作：选中“1.开始游戏”，按 J 键确认；

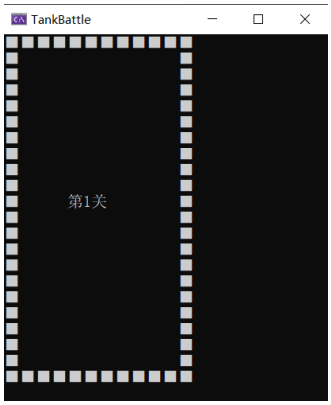
测试效果：游戏开始，出现“第 1 关”的字样；

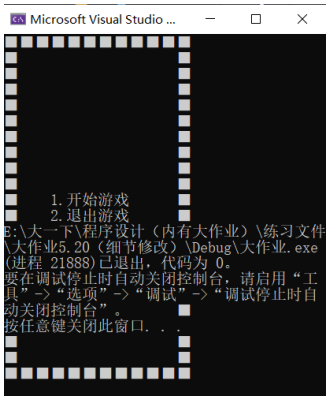
（3）结束游戏：

测试操作：按 S 键下调选项，按 J 键确认；

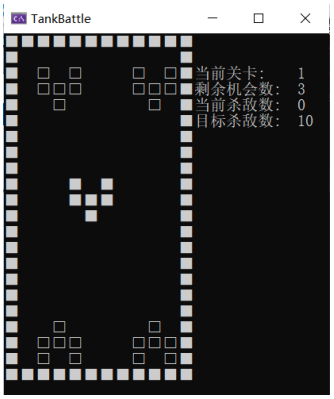
测试效果：游戏结束，出现如图界面，再按任意键即退出；







3 游戏初始状态测试：



测试操作：游戏开始，观察坦克初始位置与初始数据

测试效果：敌方四辆坦克出现在游戏主界面四角；
玩家坦克出现在主界面中心；
初始关卡为 1；
初始机会数为 3；
初始杀敌数为 0；
第一关目标杀敌数为 10；

4 坦克行动测试：

（1）敌方坦克随机性测试：

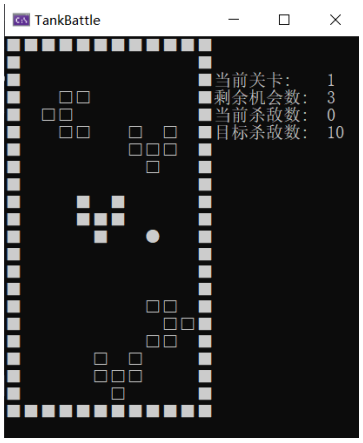
测试操作：不按键，观察敌方坦克的运动；

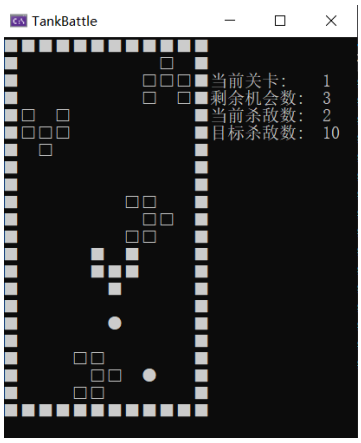
测试效果：敌方坦克随机移动，并随机发射炮弹；

（2）玩家操作测试：

测试操作：分别按“W”“S”“A”“D”键，再按“J”键；

测试效果：坦克分别上，下，左，右移动，然后发射炮弹；
若坦克方向与按键代表方向相反，则只改变方向而不改变位置；

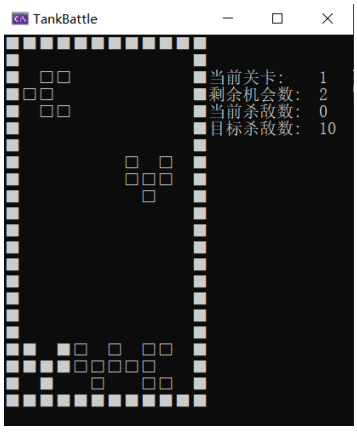


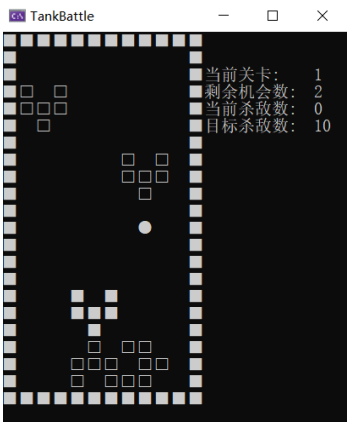


5 坦克碰撞测试：

测试操作：控制坦克接触到边界和敌方坦克，继续按键前进；

测试效果：坦克无法继续前进；





6 炮弹碰撞测试：

（1）炮弹打中边界：

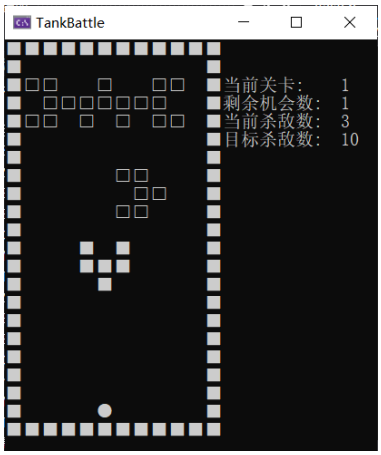
测试操作：发射炮弹，击中边界

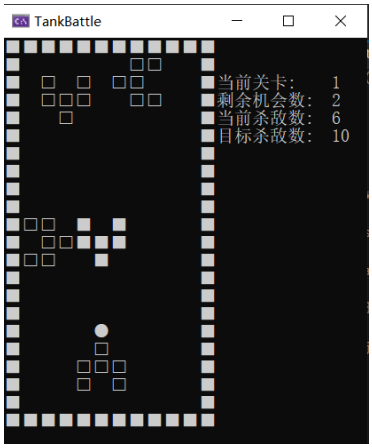
测试效果：炮弹消失；

（2）炮弹正面碰撞：

测试操作：发射炮弹，使之与敌方炮弹正面碰撞

测试效果：炮弹丧失杀伤力，相碰处留下球形弹壳，在后续刷新中消失；





7 击杀测试：

（1）炮弹击中敌方坦克：

测试操作：控制坦克发炮，击中敌方坦克；

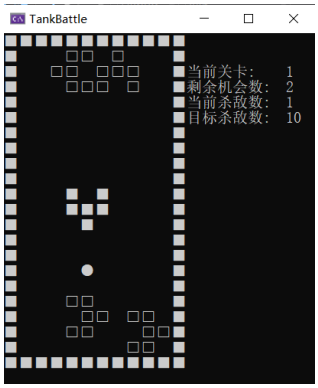
测试效果：敌方坦克消失，杀敌数+1，四角处随机出现一个坦克；

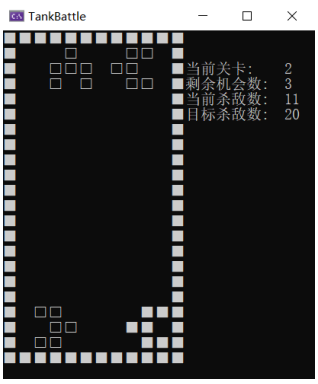
注：若玩家坦克位于四角时击中敌方坦克，即四个角被已经占满，则游戏将卡住无法继续运行。

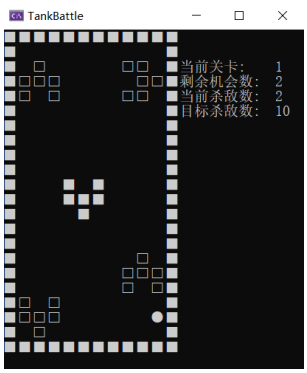
（2）炮弹击中玩家坦克：

测试操作：让敌方炮弹击中玩家坦克；

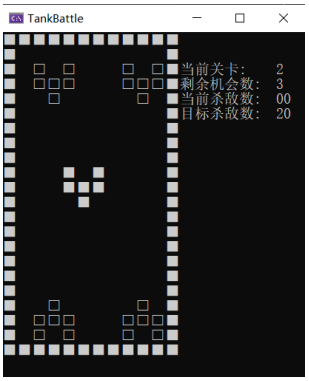
测试效果：玩家坦克消失，，机会数-1，玩家坦克位置重置在屏幕中心；







8 关卡转换测试：



测试操作：击杀敌方坦克，达到目标杀敌数；

测试效果：清屏，显示“第 n 关”（n 为下一关的编号），
重置所有坦克，关卡数+1，
当前杀敌数清零，目标杀敌数更新；

9 游戏结束测试：

（1）通关测试：

测试操作：闯过五关；

测试效果：屏幕中出现“恭喜你通关啦!！”；

（2）闯关失败测试：

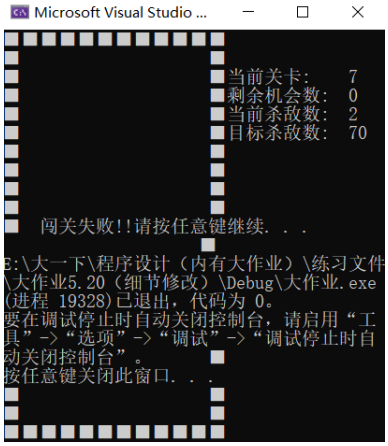
测试操作：机会数减至零（被敌方炮弹击中三次）

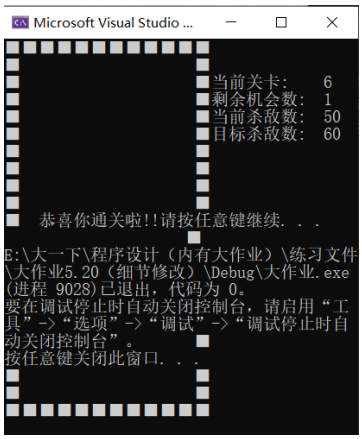
测试效果：屏幕中显示“闯关失败!！”；

（3）结束程序测试：

测试操作：在上述两种情况下，按任意键；

测试效果：程序结束；





6 总结报告

1 选题：

起初打算做一个矩阵计算器，但由于功能单一，算法复杂就放弃了。后来在网上搜索时发现有人做了贪吃蛇小游戏，我忽然有了做坦克大战的想法。既可以实现可视化界面，有一定的趣味性，又在网络上有可以借鉴的程序，可操作性强，就最终选了这个题目。在规则和界面设计上，我主要参考了小时候玩过的俄罗斯方块机上的坦克大战，界面简单，却也包含了一款小游戏具备的所有必要功能，也算是对童年游戏的一种回忆，我甚至在淘宝上买了一个类似的游戏机做参考。这个有意思的选题也让我在之后的程序设计中始终怀有一种热情，不会由于太枯燥而中途放弃。

2 设计:

首先是类的设计，这一点几乎没有什么困难。我用 Map 类实现地图的相关操作；Tank 类记录坦克的信息并实现坦克的相关操作；Bullet 类记录子弹信息并实现子弹的相关操作；Game 类实现游戏进程。

而这个题目的主要难题在于如何在 Game 类里理清游戏的逻辑结构，分出不同的函数来层层推进游戏，这个问题思考了好长一段时间。最初的想法很理想化，试图一下子实现所有功能，但在各种考虑的纠缠下，我找不出来明确的思路。后来看到群里老师提醒，可以先实现一些简单的功能，然后我就从主干逻辑入手，将游戏分为五个层次，用不同的函数来控制，最后在加上关卡数，玩家机会数，阵营判断等各种功能。

当然在其他类的实现中也有各种问题，比如如何控制时间，如何实现在指定位置输出元素符号等，这些问题如果我无法解决，就会查询网络，在网络上也借鉴了很多方案和相关函数，比如控制台大小的改变，光标的隐藏等等。虽然某些函数的具体原理我不太明白，但只要它们能很好的实现功能，我也就心安理得地使用了。

3 调试：

由于在程序整体完成之前，我没有进行大的调试，只是对于某一些单元简单地测试了一下。这导致最后调试的时候工作量巨大。比如刚打开程序时，坦克根本无法正确显示，只是一堆方块在不停的跳动。但是当我单独测试坦克数组的时候又正确显示了，这让我百思不得其解。我前后将代码检查了很多遍，最后才发现，是某一个函数忘了没添加，加上这一行代码就可以了。

后续也出现了很多 Bug，我也进行了很多调试，但由于时间原因，最后仍然留下了一些漏洞还没有解决，比如在某种情况下，当敌方坦克无法找到位置刷新时游戏就会卡住，这时候就不得不重新开始，也给我造成了很大的困扰。希望以后还会有机会继续改进，让这个小游戏更加完善。

4 总结：

这次大作业是一次很好的实践机会，在这过程中，我分析问题解决问题的能力都有所提升。同时，对小游戏的设计过程也让我对程序设计的一般步骤有了直观的认识，让我在实际操作中理解程序设计的思维，理解如何用代码完成某一项功能。遗憾的是，在程序中仍然有一些不足之处，希望在以后的学习中可以再次改进。