第二次课程设计

张涵之-191220154-计科

课程设计主要内容：坦克大战；

目标：仿制坦克防守游戏90坦克大战；

设计思路：以90坦克大战为基础参考；

模仿设计:

游戏地形(砖墙、铁墙、基地)

游戏道具(坦克、子弹、星星)

各种类型的坦克(轻型坦克, 装甲车, 反坦克炮, 重型坦克)

基本实现:

实现4种类型的坦克: 轻型坦克, 装甲车, 反坦克炮和重型坦克

实现4种坦克属性: 生命值, 攻击力, 移动速度和子弹射速

实现2种障碍物：砖墙、铁墙

实现当前局的状态栏: 显示分数, 玩家生命值, 敌军剩余坦克数量

游戏结束时显示胜负, 得分等统计信息

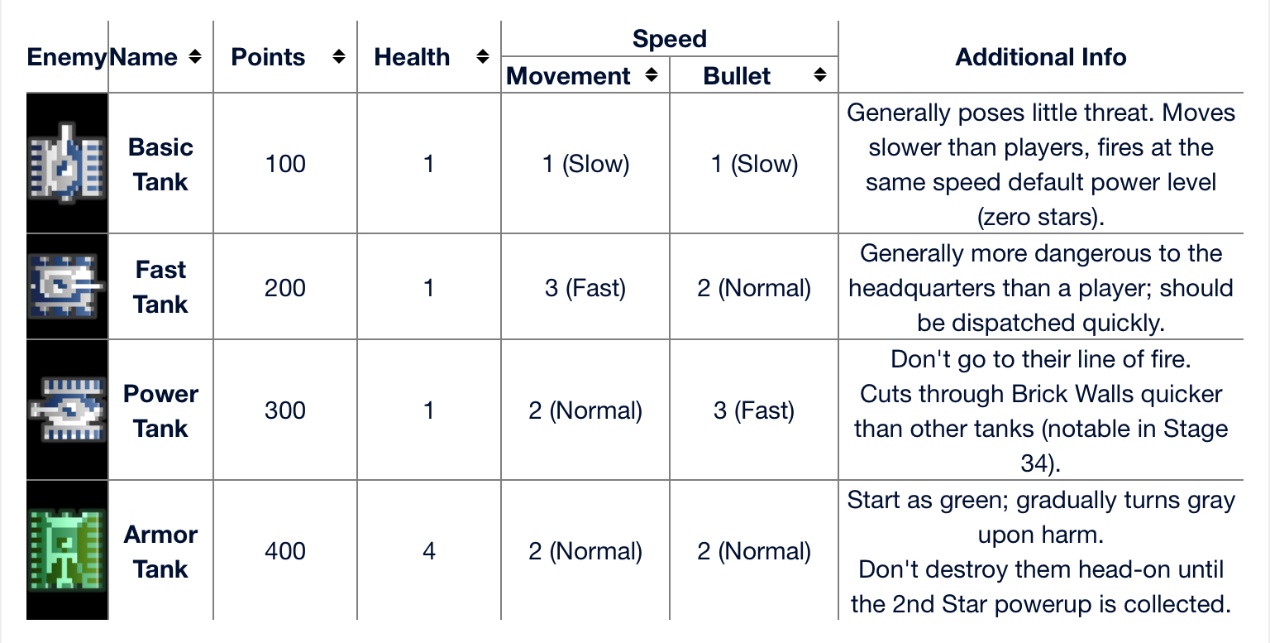
坦克设计:

轻型坦克: 没吃星星的普通坦克, 也即玩家控制的坦克

装甲车: 移动速度最快, 生命值较高

反坦克炮: 子弹射速跟吃了一个星星的坦克一样

重型坦克: 别名闪光坦克, 装甲最厚, 生命值最高



来源: <https://strategywiki.org/wiki/Battle_City/How_to_play>

子弹: 有射速, 伤害等属性, 由发射它的坦克决定, 对撞的时候会相互抵消

子弹可用于攻击敌方坦克, 毁坏墙, “吃”道具等

本游戏设定同一方坦克发射的子弹不会伤害己方坦克

道具设计:

坦克: 玩家可以增加一条命; 反之, 敌方吃了会增加防御力

星星：升级玩家坦克, 吃一次提升射速, 吃2次连续发射两个子弹, 吃3次打掉铁墙

被敌方射中一律重来. 反之, 敌方吃一次即可打掉铁墙

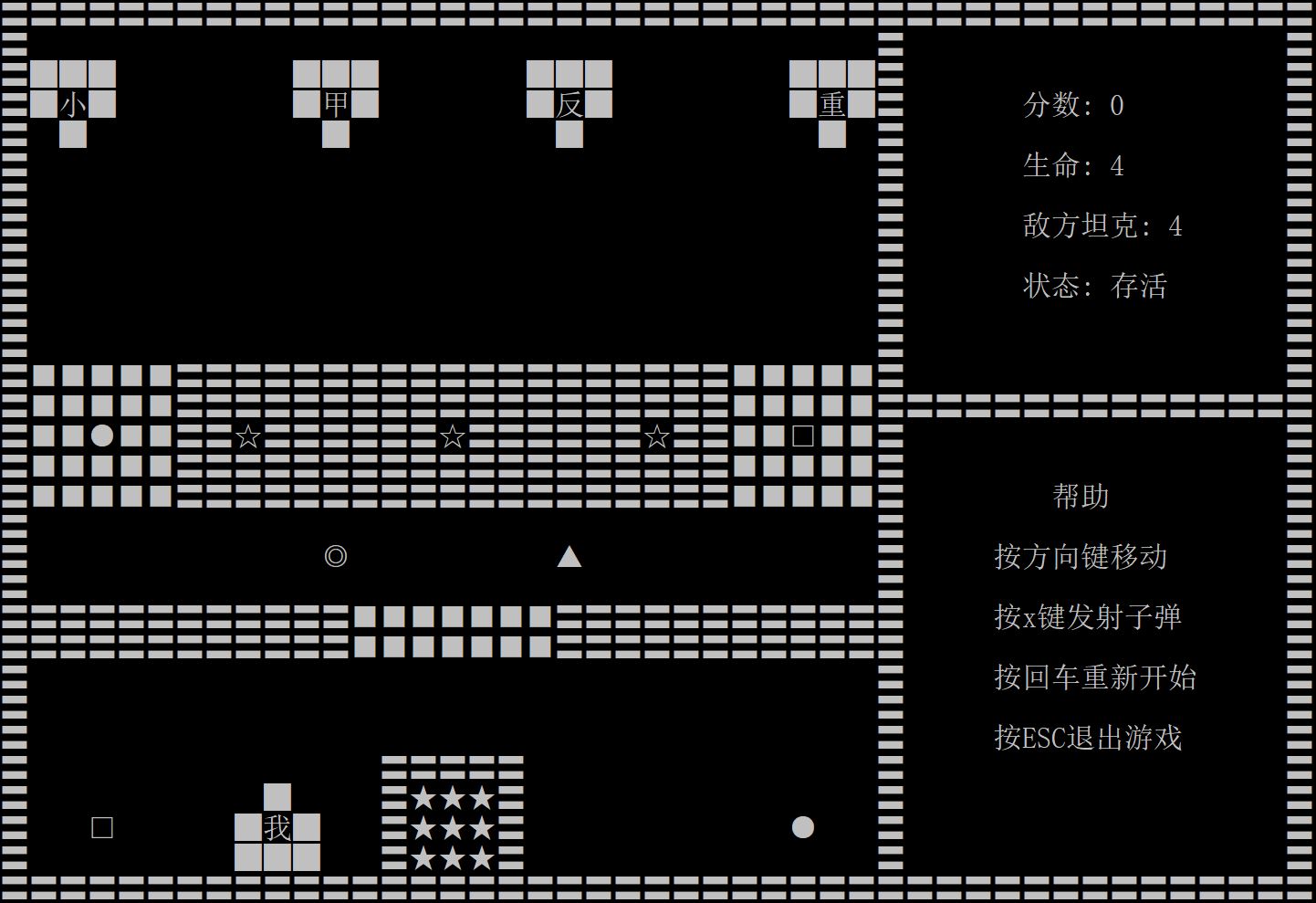
时钟: 可使敌方坦克全部停止数秒. 反之, 敌方吃了会把玩家停止

铁锹: 玩家基地的砖墙变成铁墙, 反之敌方吃了会把基地变成没有保护

炸弹: 全屏敌人遭到爆炸攻击, 全数毁灭. 反之, 敌方吃了玩家毁灭

本游戏设定“吃”的方式为用子弹打中该道具

界面设计:



界面元素说明:

坦克由 █ 拼成, 类型由汉字说明

砖墙 〓 可以被任何坦克摧毁

铁墙 ■ 可以被升级后的坦克摧毁

子弹 ⊙ 由双方坦克发射, 击中不同物体的效果不同

基地由 ★ 拼成, 被砖墙围起

道具有坦克 □ , 星星 ☆ , 时钟 ◎ , 铁锹 ▲ , 炸弹 ●

右侧状态栏显示分数, 玩家剩余命数, 敌军剩余坦克数量

游戏逻辑：

游戏开始时刷出四种各一辆敌军坦克, 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, 保卫基地不被摧毁并消灭敌军坦克

玩家按 ↑ ↓ ← → 键移动坦克, 按x键发射子弹

游戏开始时地图上刷出道具, 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: 消灭全部四辆敌军坦克时获得胜利

按回车重新开始, 按ESC键退出

扩展设计: 无

主要类的设计：

class Map {

friend class Bullet;

friend class MyTank;

friend class EnemyTank;

friend class Game;

private:

int WIDTH; //屏幕宽度

int HEIGHT; //屏幕高度

int LIFE; //玩家剩余生命值

int SCORE; //玩家得分

int ENEMY; //敌军剩余坦克

int STATUS; //游戏状态

int ME\_STOP; //玩家被停止计时器

int ENEMY\_STOP; //敌军被停止计时器

int\*\* BOARD; //表示坐标属性的二维数组

public:

Map(int w, int h); //初始化二维数组

~Map(); //归还二维数组申请的内存

void set\_home(); //每轮游戏开始前的初始化

void print\_map(); //打印地图

void set\_steel(); //把基地周围变成铁墙

void set\_null(); //把基地周围设为无防护

void update\_info(); //更新状态栏信息

bool check\_tank\_route(Position p, int dir); //检查坦克面前路径是否畅通

void clear\_tank(Position p); //从地图上抹去坦克

void print\_tank(int REPR, Position p, int dir) //在地图上打印坦克

void clear\_bullet(Position p) //从地图上抹去子弹(或其他小元素)

void print\_bullet(int NUMBER, Position p) //在地图上打印子弹

void win(); //胜利

void lose(); //失败

};

class Bullet {

int NUMBER; //子弹编号

int SPEED; //射速

int DAMAGE; //伤害

int DIRECTION; //方向

int OWNER; //射出子弹的坦克

bool MY; //是否为玩家

bool ABLE; //能否穿铁墙

bool EXIST; //是否存在(即已经打中而消失)

Position POS; //位置

friend class Game;

public:

Bullet(); //伪初始化

Bullet(int s, int d, int dir, int o, bool m, bool a, Position p);

//根据坦克发射子弹时传入的信息进行初始化

void fly(); //子弹在地图上的飞行路径

void hit(); //子弹打中不同元素时的行为

};

class Tank { //坦克的基类

friend class Game;

protected:

int ARMOR; //生命值

int LIFE; //生命数量

int DAMAGE; //攻击力

int MOVEMENT\_SPEED; //运动速度

int BULLET\_SPEED; //子弹射速

int REPR; //代表元素

int DIRECTION; //运动方向

bool STEEL\_ABLE; //是否能击穿铁墙

bool MY; //是否为玩家所有

bool ALIVE; //存活状态

Position POS; //在地图上的位置

public:

Tank(); //虚假的初始化

virtual void move(); //运动的虚函数

virtual void shoot(); //发射子弹的虚函数

void reduce\_armor(int amount); //被子弹击中掉血

virtual void eat\_tank(); //吃坦克的虚函数

virtual void eat\_star(); //吃星星的虚函数

virtual void eat\_clock(); //吃时钟的虚函数

virtual void eat\_shovel(); //吃铁锹的虚函数

virtual void eat\_bomb(); //吃炸弹的虚函数

virtual void death\_callback() //死亡的虚函数

virtual void restart(); //重启时初始化的虚函数

};

class MyTank :public Tank { //玩家坦克

int ATE; //吃星星的个数统计

friend class Map;

friend class EnemyTank;

friend class Game;

public:

MyTank(); //初始化

void move(); //运动

void shoot(); //射击

void eat\_tank(); //吃道具

void eat\_star();

void eat\_clock();

void eat\_shovel();

void eat\_bomb(); //吃道具

void death\_callback(); //死亡

void restart();

};

class EnemyTank : public Tank{ //敌军坦克的基类

protected:

int COOL\_DOWN; //随机发射子弹的冷却时间

public:

EnemyTank(); //初始化

void move(); //随机移动(有参考)

void shoot(); //射击

void eat\_tank(); //吃道具

void eat\_star();

void eat\_clock();

void eat\_shovel();

void eat\_bomb(); //吃道具

void death\_callback(); //死亡

void restart(); //重启时初始化的虚函数

};

class Tankette : public EnemyTank { //轻型坦克

friend class Game;

public:

Tankette(); //初始化

void restart(); //重启时初始化

};

class AmoredCar : public EnemyTank { //装甲车

friend class Game;

public:

AmoredCar(); //初始化

void restart(); //重启时初始化

};

class Piat : public EnemyTank { //反坦克炮

friend class Game;

public:

Piat(); //初始化

void restart(); //重启时初始化

};

class HeavyTank : public EnemyTank { //重型坦克

friend class Game;

public:

HeavyTank(); //初始化

void restart(); //重启时初始化

};

class Game { //游戏

public:

Game(); //虚假的初始化

void start\_up(); //游戏开始

void raise\_game\_over(); //游戏结束

void raise\_you\_win(); //玩家获胜

};

Map类是游戏的主体, 其他类通过Map中的地图二维数组实现互动

Tank的派生类为MyTank和EnemyTank, 其中EnemyTank又有四个派生类

Bullet类对象由Tank类的shoot()函数创建, 添加到全局数组BULLETS中

Game类负责实现与玩家的交互, 输出游戏胜负提示信息并提供重玩/退出的选择

全局变量:

以下分别将地图元素(空白, 坦克, 墙, 道具等), 状态信息, 速度, 方向

定义为int型常量, 便于地图数组标记及后续用switch case进行操作

#define \_NULL\_ 0 //地图元素(空白)

#define \_MY\_TANK\_ 1 //玩家坦克

#define \_TANKETTE\_ 2 //轻型坦克

#define \_AMORED\_ 3 //装甲车

#define \_PIAT\_ 4 //反坦克炮

#define \_HEAVY\_ 5 //重型坦克

#define \_BRICK\_ 6 //砖墙

#define \_WALL\_ 7 //地图边界栏

#define \_STEEL\_ 8 //铁墙

#define \_HOME\_ 9 //老家

#define \_BULLET\_ 10 //子弹

#define \_TANK\_ -1 //坦克(道具)

#define \_STAR\_ -2 //星星

#define \_CLOCK\_ -3 //时钟

#define \_SHOVEL\_ -4 //铁锹

#define \_BOMB\_ -5 //炸弹

#define \_ALIVE\_ 1 //存活

#define \_KILLED\_ 2 //死亡·

#define \_VICTORY\_ 3 //胜利

#define \_SLOW\_ 1 //慢速

#define \_NORMAL\_ 2 //常速

#define \_FAST\_ 3 //快速

#define \_UP\_ 1 //上

#define \_DOWN\_ 2 //下

#define \_LEFT\_ 3 //左

#define \_RIGHT\_ 4 //右

int COUNT = \_BULLET\_; //子弹计数器

Map MAP(45, 30); //地图

Bullet BULLETS[1000]; //子弹数组

MyTank MINE; //我的坦克

Tankette TANKETTE; //轻型坦克

AmoredCar AMORED; //装甲车

Piat PIAT; //反坦克炮

HeavyTank HEAVY; //重型坦克

全局函数:

struct Position {

int X;

int Y;

}; //控制台横纵长度相同的伪坐标

void MoveCursor(Position pos); //获得控制台句柄并移动到伪坐标

void HideCursor(); //控制台隐藏光标

Main函数：

int main()

{

Game GAME;

GAME.start\_up();

system("cls");

return 0;

}

遇到的问题和解决方案：

没玩过游戏, 不会玩游戏, 玩了两个小时游戏还没有打通第一关

放弃玩游戏, 在StrategyWiki上直接搜索游戏规则

敌军坦克的随机运动产生匪夷所思的愚蠢行为(撞墙, 对撞, 开太快被自己子弹打死)

部分参考网上代码中AI坦克运动实现方式, 虽然, 仍然很蠢

设置了子弹射速必须大于坦克运动速度, 且坦克不能被己方发射的子弹打死

玩家坦克运动龟速, 按键控制运动和射击极其不灵敏

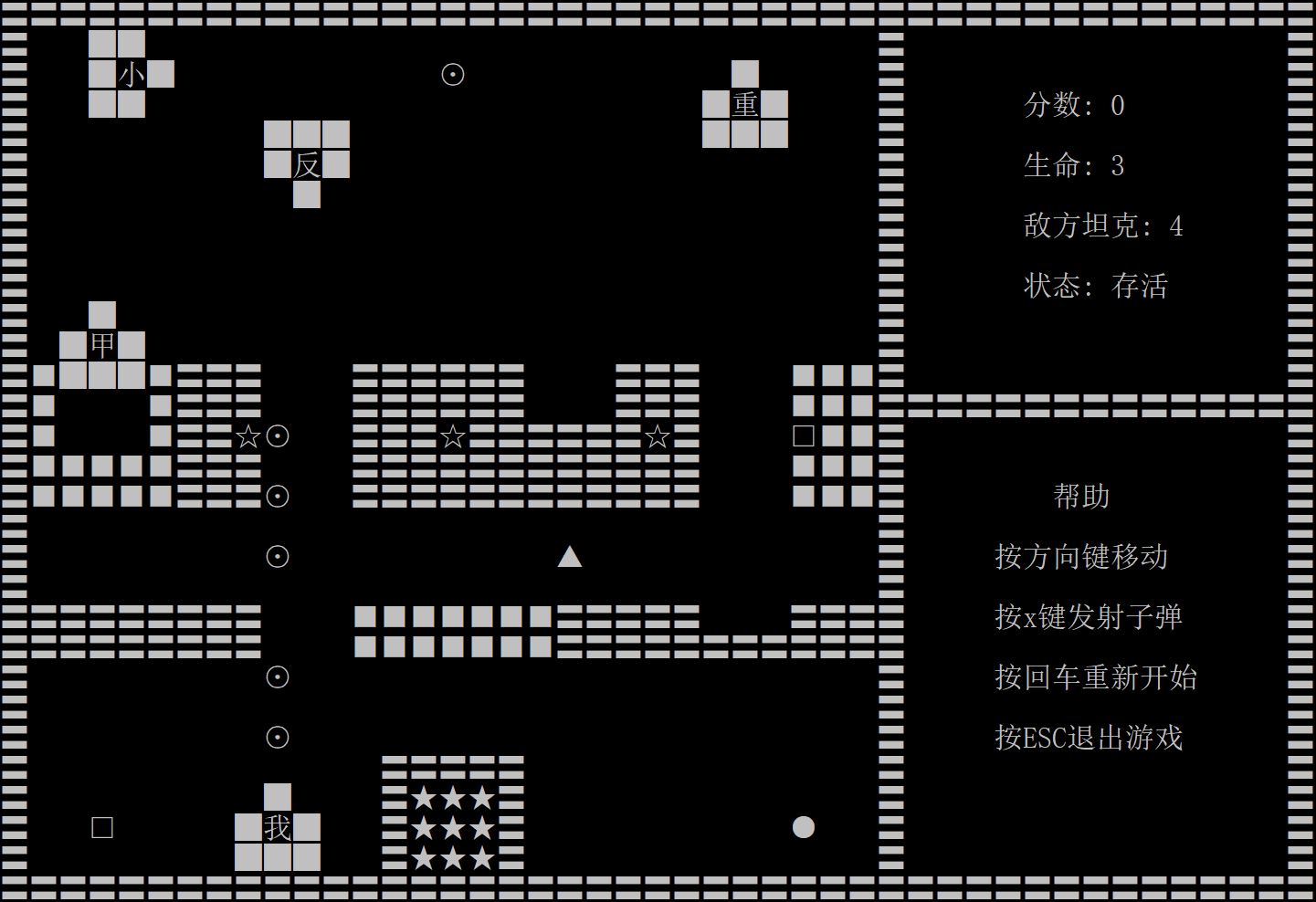
同比减少循环中Sleep()的时间间隔, 增大敌军坦克的冷却时间

游戏不好玩(不是太难了根本赢不了, 就是太简单了随便躺着赢)

没有解决(自己都不会玩游戏还想写好游戏呢)

感觉不是代码编写问题, 是设计思路问题, 来不及了, 暑假有时间再考虑

最终实现的效果：



代码完成日期：2020/4/26 设计报告完成日期：2020/4/27