
Plants vs Zombie 软件 用户手册

2023 年 4 月

目录

一、 概述	2
二、 设计主要的类	2
map 类	4
PlantCard 类	5
Store 类	6
Bullet 类	8
类之间的包含关系	9
植物属性总结如下:	9
僵尸的属性总结如下:	11
三、 游戏运行及操作方法	13
四、 总结	14

一、概述

以塔防游戏植物大战僵尸为基础参考，实现了运行在 Windows 控制台下的一个游戏。玩家目标是合理利用阳光、植物布局，守住一波一波的僵尸的攻击，击杀僵尸，获得分数。游戏中，任何一只僵尸到达地图的左边界，则游戏结束。玩家可使用”1-9”数字键以及”a-c”键选择要种植植物、方向键控制格子选择框移动、”x”键铲除植物、回车键确定、Esc 键返回，空格键暂停（游戏按键均不区分大小写，且在运行界面中均有说明）

二、设计主要的类

1. grid 类

这里的一个 Grid 类对象，是游戏草地中的一个“格子”，游戏地图中的一个小单元。一个格子中，可有的元素为：1 或者 0 个植物、大于等于 0 个僵尸。这里 Grid 类的最主要的作用就是如何展示本格子内的元素，以及对外提供对本格元素的一些操作功能，比如设置是否选中、增加/删除植物、增加/移除一个僵尸等

效果图如右图所示，部分代码如下方所示



```
class Grid {
    int dx, dy;
    Plant* plant;

    vector<Zombie*> zombies;
    bool selected;
    bool flag_refresh;
    void setRefresh() { flag_refresh = true; }
    bool bomb_flag;

public:
    Grid() { plant = nullptr; selected = false; flag_refresh = false; bomb_flag = false; }
    void setXY(int x, int y);
    void paint();
    bool setPlant(Plant* iPlant);
    void delPlant() { delete plant; plant = nullptr; flag_refresh = true; }
    clearBombFlag(); }
    void addZombie(Zombie* iZombie);
    void delZombie(Zombie* iZombie);
    void hitZombies(int attack);
```

Plants vs Zombies 软件

```
void eatPlant(int attack);
void judgeEating();

void setSelected() { selected = true; flag_refresh = true; }
void setUnSelected() { selected = false; flag_refresh = true;}

void setBombFlag() { bomb_flag = true; setRefresh(); }
void clearBombFlag() { bomb_flag = false; setRefresh(); }
void flipBombFlag() { bomb_flag = !bomb_flag; setRefresh(); }

~Grid() { delete plant; }

friend class Map;
friend class Bullet;
friend class Store;
friend class PeaShooter;
friend class SnowPea;
friend class Repeater;
friend class Squash;
friend class PotatoMine;
friend class Garlic;
friend class Pole_Zombie;
};
```

这里根据格子内的内容，有多种显示模式：

(1) 只包含植物（植物的血量和相关信息等），如图为土豆地雷，显示其血量以及蓄力阶段



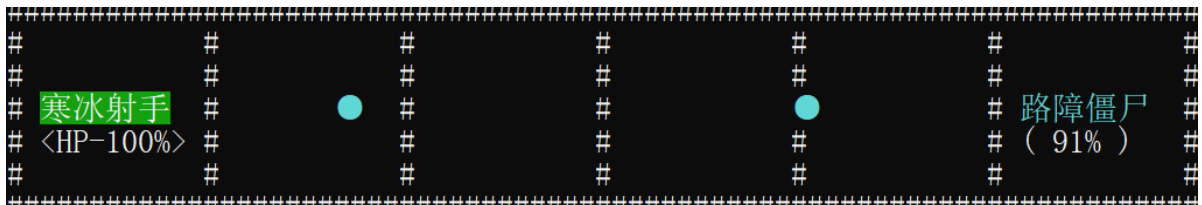
(2) 包含僵尸（僵尸的血量和相关信息等），如图为撑杆跳僵尸，显示器血量和其有无撑杆



(3) 包含植物和僵尸（植物的血量和僵尸的数量），如图为高坚果墙抵挡了一个僵尸，僵尸在吃高坚果。

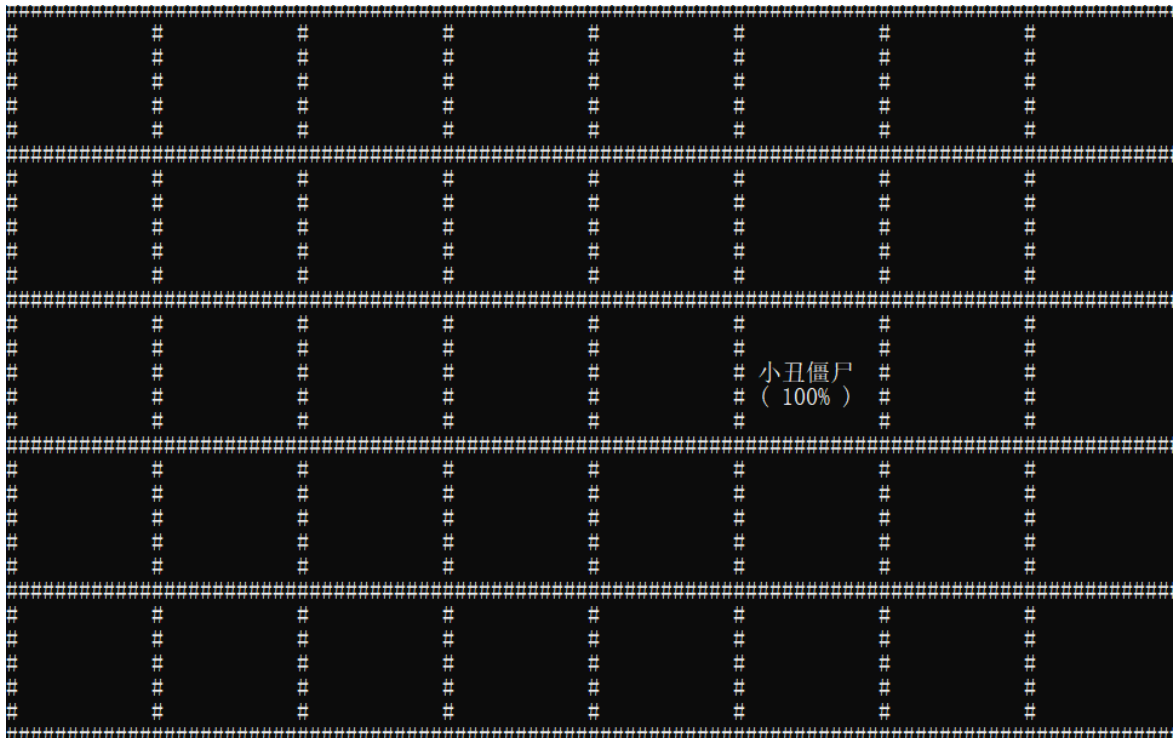


(4) 此外，还有一些特殊效果的显示，比如：寒冰射手击中后的僵尸会以蓝色字体显示（表明其移动/攻击速度减慢状态）：



2. map 类

这个类主要是用于处理地图信息，从整个地图的布局来看，地图就是由二维的多个“草地格子”以及格子间的边界线组成。如下图



部分代码如下所示

```
class Map {
    Grid grid[GRID_NUM_X+1][GRID_NUM_Y];
public:
    //地图的初始化
    void init();
    //网格线（草地块的边界）的绘制
    void paintGridLine();
    //遍历网格：处理植物、僵尸
    bool travGrid(Game &game);
    //种植某类植物
    bool setPlant(int ix, int iy, int type);
    //遍历所有格子，检查是否需要刷新显示
    void refresh();
    //(x,y)格子开启/关闭爆炸特效
    void setBombFlag(int x, int y);
    void clearBombFlag(int x, int y);
    friend class Game;
    friend class Bullet;
    friend class Zombie;
    friend class Pole_Zombie;
    friend class PeaShooter;
    friend class SnowPea;
    friend class Repeater;
    friend class Squash;
    friend class PotatoMine;
    friend class CherryBomb;
    friend class Jalapeno;
    friend class Garlic;
    friend class Spikeweed;
};
```

用于表示整个的游戏地图，包含一个 Grid 类对象的二维数组。主要是提供一些地图层面上的功能，比如地图初始化，刷新显示，负责 绘制地图（包括格子和边界线）等等

3. PlantCard 类

游戏涉及到植物的购买，而植物的购买还涉及到价格、冷却时间等信息，以及这些信息在对应位置的展示，所以这里商店中展示出的可购买植物的信息封装成一个类：PlantCard 类（意为“植物卡片”）。负责某个植物卡片的信息的设定、展示输出。

1. 向日葵	- 50\$	2. 豌豆射手	- 100\$ (18%)
5. 坚果墙	- 50\$	6. 寒冰射手	- 175\$
9. 火爆辣椒	- 125\$	a. 土豆地雷	- 25\$

部分代码如下所示

```
class PlantCard{
    int index;
    string name;
    int price;
    int CD;
    bool flag;
    int counter;

    friend class Store;
public:
    void set(int i, const string& iname, int iprice, int iCD) {
        index = i;
        name = iname;
        price = iprice;
        CD = iCD * 1000 / 10 / SLEEP_TIME;
        counter = CD;
        flag = false;
    }
    void print();
    void setSelect() { flag = true; print(); }
    void setUnSelect() { flag = false; print(); }
    void cooling();
    bool coolingDone();
};
```

4. Store 类

Store 类表示“商店”，处理游戏中和植物购买的相关的操作。比如展示植物卡片、记录已有阳光、自然掉落阳光等等。这里的核心是所有植物的卡片构成的一个一维数组。

【商店】		阳光: 200 \$		[分数: 2]
1. 向日葵 - 50\$	2. 豌豆射手 - 100\$	3. 窝瓜 - 50\$	4. 樱桃炸弹 - 150\$	
5. 坚果墙 - 50\$	6. 寒冰射手 - 175\$	7. 双发射手 - 200\$	8. 大蒜 - 50\$	
9. 火爆辣椒 - 125\$	a. 土豆地雷 - 25\$	b. 地刺 - 100\$	c. 高坚果墙 - 125\$	

部分代码如下所示

```
class Store
{
    int sun;
    int speed;
    PlantCard plants[PLANT_TYPE_MAX];
    int counter;
public:
```

```
Store() {
    plants[0].set(0, "向日葵", 50, 75);
    plants[1].set(1, "豌豆射手", 100, 75);
    plants[2].set(2, "窝瓜", 50, 200);
    plants[3].set(3, "樱桃炸弹", 150, 200);
    plants[4].set(4, "坚果墙", 50, 200);
    plants[5].set(5, "寒冰射手", 175, 75);
    plants[6].set(6, "双发射手", 200, 75);
    plants[7].set(7, "大蒜", 50, 75);
    plants[8].set(8, "火爆辣椒", 125, 300);
    plants[9].set(9, "土豆地雷", 25, 200);
    plants[10].set(10, "地刺", 100, 75);
    plants[11].set(11, "高坚果墙", 125, 200);
}

void refreshSun();
void addSun(int isun=50) { sun += isun; refreshSun(); }
bool pay(int choice, int x, int y, Map &map);
void run();
//完成所有冷却
void renew();

friend class Game;
};
```

5. game 类

这个类算是最重要的类，在这里重新定义了一个类，包含了 map 类和 store 对象。这里子弹和僵尸都是需要执行移动操作的，且会消亡，需要删除，所以在 Game 类中使用 list 容器来组织。（对于植物来说，植物种植在“格子”中，且每个格子只有一个植物，所以植物这一结构存放在的是 Grid 类中。（当然，“格子”也需要访问到本格子内的僵尸，所以其实“格子”保留了僵尸类对象的指针的副本，但对于僵尸而言，真正的组织结构其实还是在 Game 类的 list 中）。Game 类中，记录了当前游戏的不同状态，以及有不同状态下使用的按键识别函数）

并且也在其中声明了实现其他功能的逻辑函数原型。

```
void shopping();
int x, y;
int planting_type;
void openFocus();
void closeFocus();
void delPlant(int ix, int iy);
void moveBullet();
```



```
void printBullet();
void addZombie(int x, int y, int type);
int make_speed;
int make_counter;
void makeZombies();

bool moveZombie();

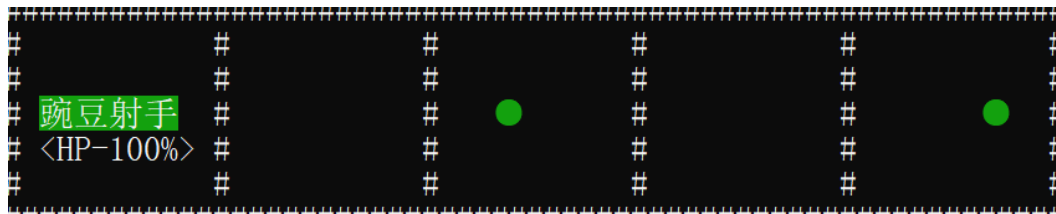
void clearZombie();

int score;
void refreshScore();
int score_speed;
int score_counter;
void calScore();
void helpInfo();
bool refreshHelpInfo;

void pause();
void gameover();
```

6. Bullet 类

Bullet 类表示游戏中射手型植物发射的子弹。子弹逻辑也稍复杂，所以单独拿出来写了一个类，而且也便于实现两种子弹：普通豌豆、冰冻豌豆。下图是最基本的子弹，普通豌豆。（成员变量为 protected 类型，便于派生类使用）



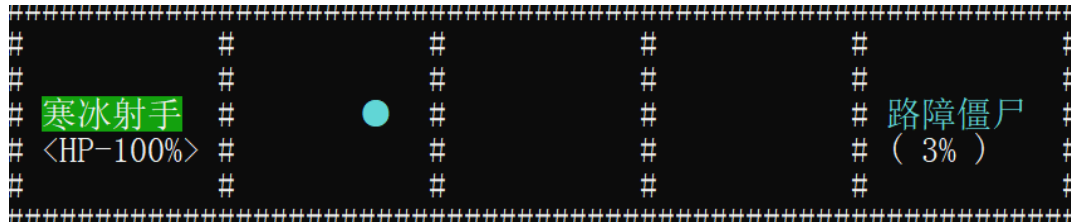
部分代码如下

```
class Bullet
{
protected:
    int x, y;
    int speed;
    int attack;
    int counter;

public:
    Bullet();
    void setXY(int dx, int dy);
    void move(Map &map);
```

```
virtual void paint();  
virtual void hitZombie(vector<Zombie*> &zombie);  
bool hit;  
};
```

而对于冰冻豌豆(SnowBullet 类)，这里继承了 Bullet 类，并重写了两个方法（基类中是虚函数），因为冰冻豌豆本身的绘制、攻击效果与普通豌豆不同（击中僵尸后，会使得僵尸 开启冷冻状态，攻击/移动速度减半，有报纸/铁门的僵尸除外），而其他则一模一样。



重写：

```
class SnowBullet :public Bullet {  
    void hitZombie(vector<Zombie*> &zombie);  
    void paint();  
}
```

7. 类之间的包含关系

还有几个类没有介绍，这里不再赘述，有兴趣可以看源码。这几个类的包含关系大概是：game 类中包含了所有的类，在其中 buller 类和 zombie 类又是独立的；planet 类不仅被 grid 类包含也被 map 类包含；planetcard 类被 store 类包含；grid 类包含 planet 类同时又被 map 包含。最后这些都被 game 类包含。

（这里说的包含是继承的关系，被包含即是类的成员类）

8. 植物的设计细节

植物属性总结如下：

向日葵：购买花费 50 金币，冷却时间 8 秒，攻击力为 0，生产时间 300 秒（即每次种植后 300 秒后才能再次种植）。

豌豆射手：购买花费 100 金币，冷却时间 7.5 秒，攻击力为 20，攻击/生产间隔为 1.4 秒，生产时间也是 300 秒。

窝瓜：购买花费 50 金币，冷却时间 20 秒，攻击力为 0，生产时间 1800 秒。

樱桃炸弹：购买花费 150 金币，冷却时间 20 秒，攻击力为 400，攻击/生产间隔为 0.5 秒，生产时间为 1800 秒。

坚果墙：购买花费 50 金币，冷却时间 0 秒，生命值为 4000。

寒冰射手：购买花费 175 金币，冷却时间 7.5 秒，攻击力为 20，攻击/生产间隔为 1.4 秒，生产时间为 300 秒。

双发射手：购买花费 200 金币，冷却时间 7.5 秒，攻击力为 20（单发），攻击/生产间隔为 1.4 秒，生产时间为 300 秒。

大蒜：购买花费 50 金币，冷却时间 7.5 秒，攻击力为 0，生产时间为 800 秒。

Plants vs Zombies 软件

火爆辣椒：购买花费 125 金币，冷却时间 30 秒，攻击力为 400，攻击/生产间隔为 0.6 秒，生产时间为 1800 秒。

土豆地雷：购买花费 25 金币，冷却时间 20 秒，攻击力为 1800，攻击/生产间隔为 7.5 秒，生产时间为 1800 秒。

地刺：购买花费 100 金币，冷却时间 7.5 秒，攻击力为 20，攻击/生产间隔为 1.0 秒，生产时间为 300 秒。

高坚果墙：购买花费 125 金币，冷却时间 0 秒，生命值为 8000。

总体来说，各种具体植物的实现有一定的相通之处，比如都是需要在派生类的构造函数种调用基类的 `set` 函数，设定植物属性。然后，根据具体的特性去重写对应方法。

下面分别介绍各类植物的设计细节：

①向日葵

额外的属性如下图（8 秒生成一次阳光）。且重写了 `go` 函数，在其中实现了周期性的增加阳光的功能。

②豌豆射手、寒冰射手、双发射手

首先，是最典型的豌豆射手。这里重写了 `go` 函数，实现了周期性发射豌豆的功能。

同理，寒冰射手也是基本类似，只是周期性生产的“子弹”不是普通的 `Bullet` 类对象，而是 `SnowBullet` 这个派生类对象。

而双发射手几乎与豌豆射手一样，只是在重写的 `go` 函数中，每个周期要产生两个子弹，同时要注意两个子弹的间距要恰当。

③坚果墙实现较为简单，因为没有特殊效果，只需要设置 `HP` 为 4000，不需要重写 `go` 函数。

高坚果墙类似，`HP` 设置为 8000，但同时要将 `skipable` 成员变量置为 `false`，这样就使得 撑杆跳僵尸无法越过。

④地刺

地刺与其他植物有较大的不同。

所有的僵尸都不会停下来吃它；其次，它有攻击性，重写 `go` 函数，在其中实现周期性的对本格子内的所有僵尸产生伤害，并且产生伤害的时候本格子红色闪烁一下。

窝瓜、土豆地雷、樱桃炸弹、火爆辣椒

首先，窝瓜的设计是对附近一个格子内的所有僵尸造成 1800 点伤害，且自身消亡。这里为了使得效果接近原版，这里窝瓜要 0.8s 后才产生效果（这段时间中，窝瓜所在格子会有红色闪烁）。0.8s 后，如果附近不存在僵尸，等待直到僵尸靠近就产生效果。

同理，土豆地雷则设定一个较长的装填时间（还重写了 `printExtra()` 方法，所以会额外显示装填进度），大约 7.5s，装填完毕后（所在格子会开启红色闪烁）与窝瓜效果一致。

而对樱桃炸弹来说，这里设计是种植一段时间后（0.5s，这段时间中爆炸范围内的格子红色闪烁），不管有无僵尸，立即对周围九个格子内造成 1800 点伤害，且自身消亡。火爆辣椒同理，只是爆炸范围变成所在的一行。

⑤大蒜

大蒜也是重写了 `go` 函数，在其中，会调用同一个格子内的僵尸对象的 `setScape` 方法，使其开启“大蒜效果”。而“大蒜效果”的具体实现由 `Zombie` 类完成，其行为是一段一段时间后，随机切换到另一行。

9. 僵尸的设计细节

僵尸的属性总结如下：

普通僵尸：击杀得分为 50，血量为 200，速度为 4.5 秒/格，攻击力为 100 点伤害/秒。

摇旗僵尸：击杀得分为 50，血量为 200，速度为 3 秒/格，攻击力为 110 点伤害/秒，会为其它僵尸提供加速效果。

路障僵尸：击杀得分为 75，血量为 570，速度为 4.5 秒/格，攻击力为 120 点伤害/秒，有较强的防御能力。

铁桶僵尸：击杀得分为 125，血量为 1300，速度为 4.5 秒/格，攻击力为 120 点伤害/秒，有更强的防御能力。

橄榄球僵尸：击杀得分为 175，血量为 1600，速度为 2.5 秒/格，攻击力为 120 点伤害/秒，很难被撞飞。

铁门僵尸：击杀得分为 125，血量为 1300，速度为 4.5 秒/格，攻击力为 120 点伤害/秒，拥有很强的防御能力。

读报僵尸：击杀得分为 125，血量为 300，速度为 4.5 秒/格，攻击力为 100 点伤害/秒（普通攻击）或者 200 点伤害/秒（读报攻击），具有两种攻击方式。

撑杆僵尸：击杀得分为 100，血量为 430，速度为 2 秒/格（携带撑杆），4.5 秒/格（抛掷撑杆），攻击力为 100 点伤害/秒，可以远程攻击。

小丑僵尸：击杀得分为 250，血量为 800，速度为 1.2 秒/格，攻击力为 100 点伤害/秒，行动时会在自己周围放置玩偶，阻碍植物攻击。

舞王僵尸：击杀得分为 350，血量为 500，速度为 1.2 秒/格（前进），5.5 秒/格（后退），攻击力为 150 点伤害/秒，可以反弹被攻击的植物。

伴舞僵尸：击杀得分为 50，血量为 200，速度为 5.5 秒/格，攻击力为 100 点伤害/秒，会为其它僵尸提供加速效果。

下面分别介绍各类僵尸的实现细节：

①普通僵尸

与 `Plant` 类不同，僵尸的基类 `Zombie` 类是有实际含义的，它表示普通僵尸，属性都是默认参数。它提供了 `set` 函数用于设定属性参数，如下：

```
void set(const string& str, int iscore=5e, int ilife = 200, int ispeed = 45, int iattack = 100)
```

②摇旗僵尸、路障僵尸、铁桶僵尸、橄榄球僵尸

这四类僵尸，在行为上与普通僵尸一致，只是属性参数设置得不同。所以只需要在派生类的构造函数中调用下 `set` 函数设置相应的参数即可。

摇旗僵尸仅是普通僵尸稍快，且总是每波僵尸第一个刷出（在 `Game` 类中的 `makeZombies` 函数中实现）；路障僵尸、铁桶僵尸都是 HP 高，速度与普通僵尸一样；橄榄球僵尸是速度快且 HP 高。以上三种僵尸均比普通僵尸伤害略高。

③铁门僵尸

属性与铁桶僵尸一致，但铁门僵尸有在有铁门状态下，被寒冰射手击中不会进入冰冻效果（攻击/移动速度减半）。这里按 HP 值划分，铁门僵尸 HP 低于 500 时，表示铁门被打破。

为了实现这个特性，这里重写了 `setFreezing` 函数（被寒冰豌豆击中后被调用），在新的实现中会先判断有无铁门，再是否进入冰冻状态。

同时，这里也重写了 `printExtra` 函数（需要输出额外信息时被调用，基类中函数体为空），因为铁门僵尸有“有铁门”和“无铁门”两种状态，需要作为额外的信息输出。

④读报僵尸

读报僵尸 HP 小于 200 时，表示报纸被打烂，进入红眼状态，此时速度变得很快，攻击翻倍。同时，他像铁门僵尸一样，有报纸的状态下，免疫寒冰。

所以，就像铁门僵尸，这里也是重写了 `setFreezing` 函数以及 `printExtra` 函数。

但额外地，还重写了 `hit` 函数（僵尸受伤扣血时被调用），因为这里要根据 HP 的变化，改变属性参数，所以要监测 HP 的变化；也重写了 `printName` 函数，因为红眼状态下，名字变红。

⑤撑杆僵尸

撑杆僵尸有两种状态：有/无撑杆。

而撑杆僵尸的行进逻辑与普通僵尸不同，所以这里重写了 `move` 函数（僵尸的行动逻辑），调整其逻辑为越过第一个可以吃的植物（所以地刺不阻挡它），但不能越过 `skipable` 属性为 `false` 的植物（如高坚果墙）。且越过后，速度减慢。

⑥小丑僵尸

小丑僵尸与其他僵尸均不同，他有自己的特殊效果：爆炸（主要是因为这个效果会影响到其他对象，其他“格子”，而不是仅仅是影响自己，所以特殊）。

Plants vs Zombies 软件

所以这个时候，就需要重写 go 函数（每个时钟周期都会被调用，且传入的参数为当前 Game 类对象的引用）。

所以，小丑僵尸的 go 函数逻辑为：有个计时器，每 2 秒种决策一下，以 2/3 的概率决定要不要自爆。爆炸波及周围 9 格子的植物，格子红色闪烁一下，自己也会消亡。



⑦舞王僵尸、伴舞僵尸

如同小丑僵尸，舞王僵尸有自己的特殊效果，所以也是重写了 go 函数，其中的逻辑为：出现 4 秒后，在其上下左右四个格子各召唤出一个伴舞僵尸，召唤完成后，速度变慢。

且这里为了平衡游戏难度，只给了舞王僵尸一次的召唤机会，即用了个变量来记录是否已召唤过。而伴舞僵尸，属性与普通僵尸基本一致，只是速度略慢（但和召唤后的舞王僵尸速度是一致的），且不被随机刷出，只可能被舞王僵尸召唤出。

三、游戏运行及操作方法

直接双击运行目录下的 PVZ.exe 文件，或者打开项目源文件编译生成，使用 visual studio2017 及以上版本均可。

名称	修改日期	类型	大小
 PVZ.exe	2023/5/7 23:04	应用程序	1,699 KB
 PVZ.pdb	2023/5/7 23:04	Program Debug ...	11,532 KB

开始游戏后，每个周期会产生一波僵尸，每波僵尸必有一个摇旗僵尸。一开始是每波僵尸有一只随机种类的僵尸，后续，随着玩家分数的增加，数量逐渐增加，难度递进。

游戏界面：



在每个格子的地方会显示植物名称和僵尸名称及属性，在下方商店一栏中会显示剩余多少阳光和当前的分数以及不同的植物及它们种植所需要的阳光

Plants vs Zombies 软件

每个植物前面都会有数字和字母，操作的时候种植的位置按下对应的数字后再按下相应的上下左右箭头键就可以操作。下面一栏是种植植物后的冷却时间，当达到百分之百就可以再种植一次。

左下方显示着当前的游戏状态，且不同的状态，后面的帮助信息不同：

其中按下 `shift` 键+“+”键会增加一千阳光和刷新植物的冷却时间，类似开挂操作。按 `x` 键可以铲除植物，空格键暂停游戏。前面说过有些特殊的植物攻击的时候会有不同的特效，这里的寒冰射手就属于一种特殊的植物，被它攻击过会变成蓝色的。

当僵尸太强我们开挂也没办法的时候，那就只能挂掉。结束界面如下图所示



四、总结的创新点

1. 针对画面刷新问题

使用了一个 `bool` 变量来记录本次时钟周期内是否需要刷新显示，这样在时钟周期的尾部，每个周期内最多只刷新一次，避免了画面过于频繁地闪烁问题，同时也有利于不同层次元素的输出。

2. 对于子弹飞行逻辑方面

由于子弹●占两个字符的宽度，会出现与中文字符重叠错位的问题，所以开发者通过将子弹宽度设定为两个字符格子，并通过设置格子边界的宽度为偶数的方式，使子弹每次要么完整地遮住一个中文字符，要么不遮住，避免了子弹与中文字符错位的问题。

3. 使用继承的思想

复用基类的代码，减少编码量和编译时间，提高了代码的可读性和可维护性。例如在撑杆僵尸的移动逻辑中，重写了 `move` 函数，并在跳完分支情况下直接调用基类的 `move` 函数，从而避免了重复编写代码。

```
else
{
    //否则调用普通僵尸(父类)的行进逻辑
    return zombie::move(map);
}
```