第二次课程设计报告

植物大战僵尸②

时间： 4月13日~5 月 11 日

姓名：张玄逸　　学号：201220194

目录

注：没有标注（更新）的部分可能有少许改动，但不是太多

一、概述

A.主要内容

仿照植物大战僵尸，制作一个基于控制台的简易版游戏。玩家需要收集并花费阳光，种植植物以击退试图入侵房屋的僵尸，当有僵尸越过草坪左侧防线时游戏结束。

B.目标（更新）

场景：白天（纯草坪无水池迷雾，有阳光掉落）

模式：无尽模式（除非僵尸越过左侧防线，否则游戏不会停止）

规则：每击杀一个僵尸都会获得一定分数，分数累积（不超过 100000）

植物 16 种：豌豆射手，向日葵，樱桃炸弹，坚果墙，土豆雷，寒冰射手，食人花，双发射手，窝瓜，三线射手，火爆辣椒，地刺，火炬树桩，高坚果，南瓜头，大蒜

僵尸 10 种：普通僵尸，摇旗僵尸，路障僵尸，撑杆僵尸，铁桶僵尸，报纸僵尸，舞王僵尸，伴舞僵尸，小丑僵尸，投石车僵尸

C.设计思路

①复习（学习）前置知识：面向对象编程，控制台界面编程

②划分模块：将整个程序划分为 game,store,lawn,plant,zombie,overall 六个版块

其中前五个模块包含对应的类，最后一个 overall 不含类，只含常量和全局函数，常量如窗口大小，颜色，草坪大小，帮助栏宽度，全局函数如隐藏光标，移动光标，按某种颜色输出等。

③确定从属关系：根据空间和逻辑关系来区分主次，确定包含与被包含的关系

④明确功能：五个大类分别执行各自功能（后文会详细阐述）

二、主要类的设计

A.类的数据与操作

①Game 类

Game 类是游戏的总体框架，负责进行整个游戏界面的初始化与更新，僵尸与植物的行动，并且接受玩家的按键并给出相应的反馈。

Game 共有四个状态：商店，种植，铲除，暂停。具体操作细节在三、A.运行操作方法中会阐述。

```
//开始循环
void Game::start() {
    while ( true ) {
        Sleep( single_time );
        if ( state == Pause ) { ... }
        if ( state == Shopping ) { ... }
        if ( state == Planting ) { ... }
        if ( state == Wiping ) { ... }
        if ( state == Pause ) continue;
        store_action( *this );
        lawn_action();
        bullet_action();
        lawn.refresh( *this );
        print_bullet();
    }
}
```

```
class Game {
    friend class Store;
    friend class Plant;
    friend class Lawn;
    friend class Bullet;
    friend class Grid;
private:
    Lawn lawn;
    Store store;
    list<Bullet*>bullet_list;//子弹列表
    enum State { Shopping, Planting, Wiping, Pause }state;//状态
    int store_plant_x;
    int store_plant_y;//当前要种植的植物相对坐标
    int x;//当前光标相对位置，与状态有关
    int y;
```

Game 中包含草坪，商店，子弹，状态，植物坐标和光标相对位置。子弹作为一个单独的 class，在 Plant 中只保存了一个表明类型的 int，而当前子弹的完整数据保存在 Game 中而不是 Plant 中的原因是，子弹和植物其实是相对独立的两个部分，植物只需要产生子弹，之后不会对子弹造成任何其它影响。植物消亡后不会再产生新的子弹，但未击中僵尸或碰到边界的子弹仍然处于运行中。

全部成员函数

```
//初始化
Game();
//开始循环
void start();

//商店模式
void shop();
//是否选中商店植物
void plant_selected(int x, int y,int z);
//是否取消种植
void unplant(int z);
//购买植物
bool buy(int z,int w);

//种植模式
bool plant( int z , int w );
//选中草块
void grid_selected(int x, int y,int z);
//取消选中草块
void grid_unselected(int x, int y);

//铲除模式
void wipe();
//是否选中铲除
void shovel_selected( int z );
//是否取消铲除
```

```
//将植物铲除
bool cancel_plant(int x, int y);

//暂停模式
void pause();

//子弹行动
void bullet_action();
//打印子弹
void print_bullet();

//草坪行动
void lawn_action();
//商店行动
void store_action(Game& game);

//失败
void game_over();
```

②Store 类（更新）

Store 类主要负责界面下方的版块，从左至右分别是提示栏，植物栏，阳光、分数、铲子栏，帮助栏。

其中，提示栏主要根据玩家某些不合理的操作作出相应的提示。

```
chat [ 0 ] = (char*) "";
chat [ 1 ] = (char*) "这块地已经种植了植物";
chat [ 2 ] = (char*) "这块地没有种植植物";
chat [ 3 ] = (char*) "阳光不足";
chat [ 4 ] = (char*) "植物冷却中";
chat [ 5 ] = (char*) "正在添加中……5月11日开放（如果我还能活到第二次课设的话）";
chat [ 6 ] = (char*) "这块地已经种植了南瓜头";
chat [ 7 ] = (char*) "一大波僵尸正在靠近";
```

（第五条已经没用了）

```
class Store {
    friend class Game;
    friend class Plant;
    friend class Grid;
private:
    int sun_cooling;//目前生产单个阳光进度
    int sun_circle;//生产阳光的间隔
    int sun;//阳光总数
    int sun_flash;//阳光闪烁阶段
    int chat_circle;//提示信息显示时长
    int chat_cooling;//提示信息已显示时长
    int now_chat;//当前提示编号
    Plant plant_list [ 3 ] [ 4 ];//可购买的植物
    int score;//分数
    bool score_refresh;//分数是否更新
    char* chat [ 101 ];
```

Store 需要记录提示信息，阳光，植物的冷却情况，在需要更新时重新输出。还需要存储每个格子对应的植物，以便在购买时判定是否冷却完毕以及阳光是否足够。

自然阳光生产时间在 5.5-9.5s 之间波动。

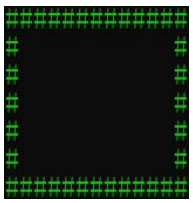
③Grid 类（更新）

```
int x;
int y;//绝对位置
int sta_x;
int sta_y;//相对位置
Plant* plant;//该格子上的植物
Pumpkin* pumpkin;//该格子上的南瓜头
list<Zombie*>zombie_list;//该格子上的僵尸
bool in_refresh;//内部是否需要重新绘制
bool out_refresh;//外部是否需要重新绘制
int color;//当前颜色（受游戏状态及光标位置影响）

int bombing;//爆炸时间段
int squashing;//窝瓜时间段
int flashing;//舞王灯光时间段
```

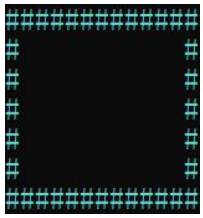
Lawn.h 里面有 Grid 和 Lawn 两个类，其中 Grid 是单个草块，被包含在草坪 Lawn 中。

单个草块的大小为 5*11（不包含边界），颜色为深绿色，用‘#’绘制。



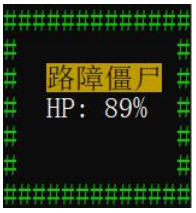
有植物时

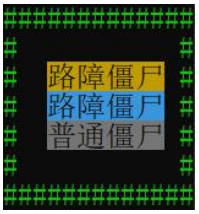
种植模式被选中时



铲除模式被选中时

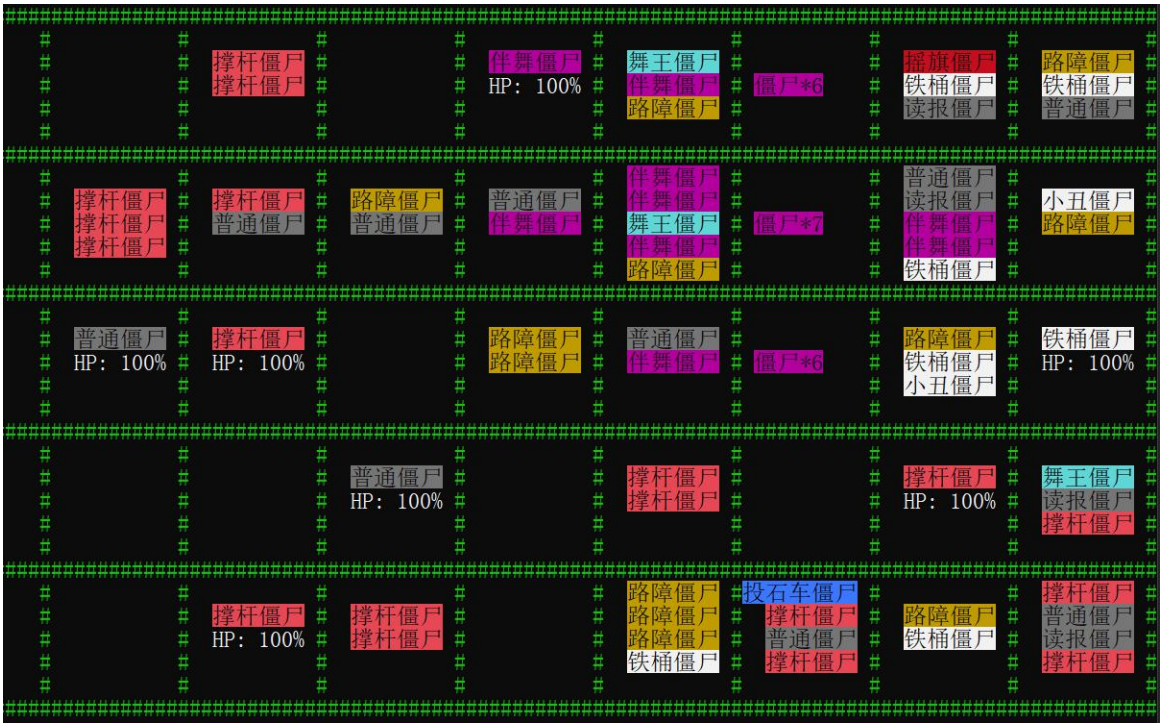
有僵尸时





僵尸与植物同存时

僵尸及植物共存时展示方式如上，单独有僵尸时具体展示方式如下。



（时间加快了 10 倍）

全部成员函数

```
//初始化
void init( int x , int y );
//添加植物
void add_plant( Plant* x);
//清空内部
void clear();
//打印植物，僵尸
void print( Game& game );
//打印生命值
void print_defence( int x , int y , int z,int w);
//打印名称
void print_name( string x , int y,int z , int w );
//植物行动
void plant_action( Game& game );
```

```
//僵尸减血
void zombie_minus( Zombie* zombie, int x , Game& game );
//植物减血
void plant_minus( Plant*plant,int x );
//打印边框
void print_circle(Game&game);
//更新
void refresh(Game&game);
//清除死亡的僵尸
void clear_zombie();
```

④Lawn 类

```
class Lawn {
    friend class Game;
    friend class Plant;
    friend class Grid;
private:
    int zombie_cooling;//生产单个僵尸的进度
    int zombie_circle [ 21 ] [ 2 ];//僵尸产生的时间间隔与数目
    int whole_wave;//完整的一轮包含的时间间隔数目
    Grid grid_list [ 9 ] [ 5 ];
```

Lawn 只包含 5*9 个草块和一些关于生产僵尸的常量，草坪的作用主要就是遍历实施单个 Grid 的操作和生产僵尸。在 Zombie 类会详细解释 zombie_circle 与 whole_wave 的含义。

草块的存储是先列后行，因为<windows.h>中设置光标的函数参数为先列后行，为避免混乱就统一采用此种顺序。

⑤Plant 类（更新）

Plant 是一个基类，12 种植物是其子类，共用一个虚函数 action。

```
protected:
    string name;//名称
    Plant_type type;//种类
    int color;//当前颜色
    int origin_color;//原始颜色
    int high_light;//被攻击时的颜色
    int blood;//防御力
    int defence;//现有生命值
    int cost_sun;//价格
    int cooling;//当前已冷却时间
    int cool_time;//冷却时间
    int x;
    int y;//相对位置

    bool can_jump;//能否被跳过
    int eating;//被吃阶段
```

植物	总血量	攻击	攻击间隔	攻击范围	无尽评星
豌豆射手	300	豌豆 20	1.4s	正前一排	☆
向日葵	300	-	-	-	必备
樱桃炸弹	300	爆炸1800(灰烬)	1s爆	3*3格内圆	★★★★
坚果	4000	-	-	-	★★
土豆雷	300	爆炸 1800	15s出土	自身格	★★★☆
寒冰射手	300	冰豌豆 20	1.4s	正前一行	★☆
大嘴花	300	吞噬(秒杀)/咬40	咬1s消化42s	正前1.5格	★★
双发射手	300	豌豆 20*2	1.4s	正前一行	★☆
小喷菇	300	孢子 20	1.4s	正前3格	★★★☆
阳光菇	300	-	-	-	-
大喷菇	300	孢子 20(群)	1.4s	正前4格	★★★☆
墓碑吞噬者	300	-	4.5s吞碑	-	-
魅惑菇	300	魅惑	-	-	★
胆小菇	300	孢子 20	1.4s	正前一行	★★☆
寒冰菇	300	冻结伤害 20	1s爆	全屏	★★★★☆
毁灭菇	300	爆炸1800(灰烬)	1s爆	7*7格内圆	★★★☆
睡莲	300	-	-	-	必备
窝瓜	300	跳跳 1800	1.8s锁中	身后正前1格	★★★★
三线射手	300	豌豆 20*3	1.4s	上中下正前	★★☆
缠绕海草	300	缠绕(秒杀水僵)	-	前后小范围	★☆
火爆辣椒	300	火烧1800(灰烬)	1s爆	自身所在行	★★★
地刺	300	戳刺20(群)	1s	自身格	★★
火炬树桩	300	燃射 13	-	命中小范围	★★☆
高坚果	8000	-	-	-	★★★
海蘑菇	300	孢子 20	1.4s	正前3格	☆
路灯花	300	-	-	-	-
仙人掌	300	尖刺 20	1.4s	正前一排	★
三叶草	300	吹走气球	0.5s吹	全屏	★★★★
裂荚射手	300	豌豆 20	1.4s	自身所在行	★★
杨桃	300	星星 20*5	1.4s	五个方向	★★☆
南瓜头	4000	-	-	-	★★★★★
磁力菇	300	吸铁	15s再吸	5*5格内圆	★☆
爱心菜投手	300	爱心菜 40	2.9s	正前一排	☆
花盆	300	-	-	-	☆
玉米投手	300	玉米20/黄油40	2.8s	正前一排	★★
咖啡豆	300	-	1s吞1s唤醒	-	★★☆
大蒜	400	-	-	-	★★★
叶子保护伞	300	-	-	3*3格	★★★★
金盏花	300	-	-	-	-
西瓜投手	300	西瓜80/溅射26	2.9s	1.5*3格区域	★★★
机枪射手	300	豌豆 20*4	1.4s	正前一行	★★★
双子向日葵	300	-	-	-	★★★★★
忧郁菇	300	孢子20*4(群)	1.9s	3 *3格区域	★★★★★
香蒲	300	尖刺 20*2	1.4s	全屏	★★★
冰瓜	300	冰瓜80/溅射26	2.9s	1.5*3格区域	★★★★☆
吸金磁	300	-	-	-	-
地刺王	450	戳刺 20*2(群)	1s	自身格	★★★
玉米加农炮	300	爆炸1800(灰烬)	装35命中3.7	3*3格内圆	★★★★★
模仿者	300	-	变身3.2s	-	★★★★★

1.豌豆射手(peashooter)

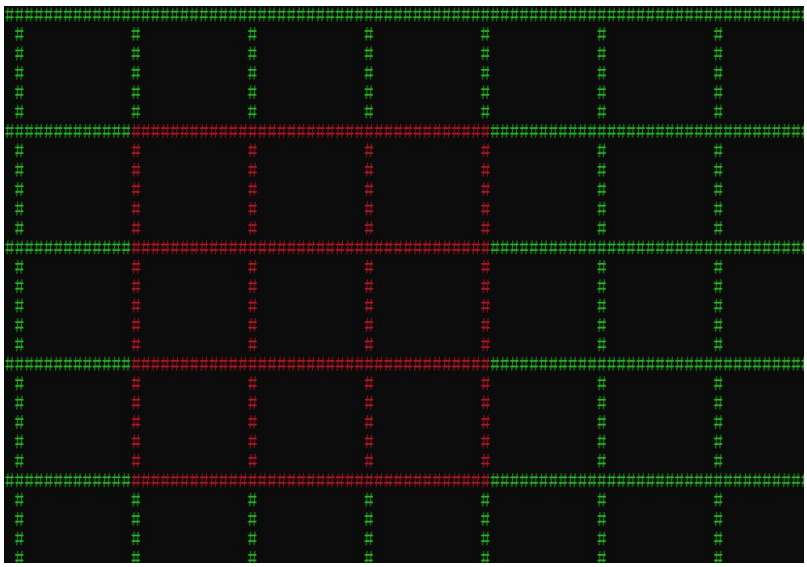
刚种植时，即使前方有僵尸也不会立即发射豌豆，而是停滞大约 1 秒后发射。

2.向日葵(sunflower)

第一次隔 7.5 秒生产阳光，之后每隔 24 秒生产一次。

左边是平常状态，右图是产生阳光时的高亮阶段，持续 1 秒。

3.樱桃炸弹(cherry_bomb)

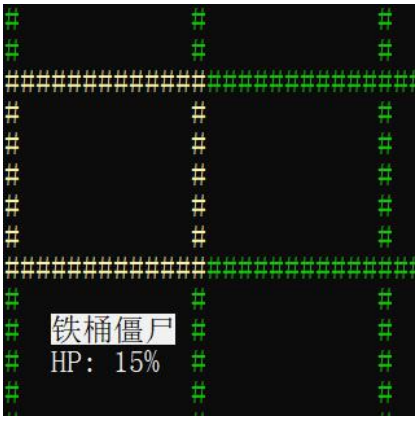


种下 1 秒后爆炸，爆炸效果持续 1 秒。

4.坚果墙(wall_nut)

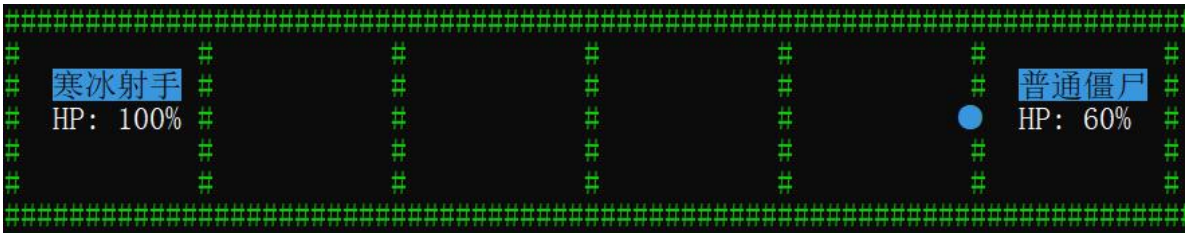
没啥好说的，一个肉盾。

5.土豆雷(potato_mine)



前者为准备时间，后者为可以随时爆炸的状态，僵尸触碰后立即爆炸，爆炸效果持续 1 秒。（杀伤力只在爆炸的一瞬间，效果只是给人看的）

6.寒冰射手(snow_pea)

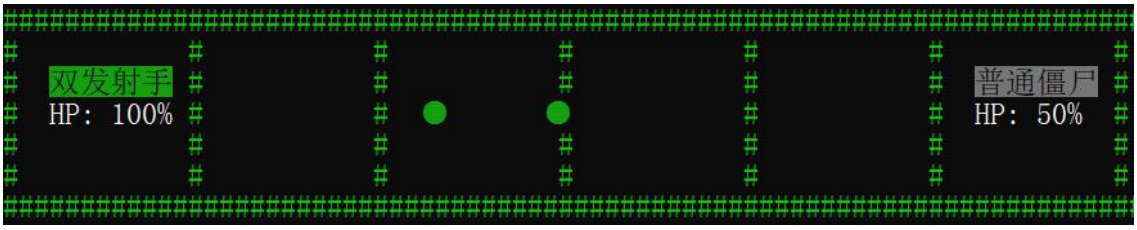


被冰冻后，僵尸速度和攻击力减半，颜色变为深青色，若没有受到新的冰豌豆攻击，则 10 秒后恢复正常。

7.食人花(chomper)

前者为正常状态，后者为消化状态，能吃掉此格和前方一格范围内第一个僵尸。

8.双发射手(repeater)

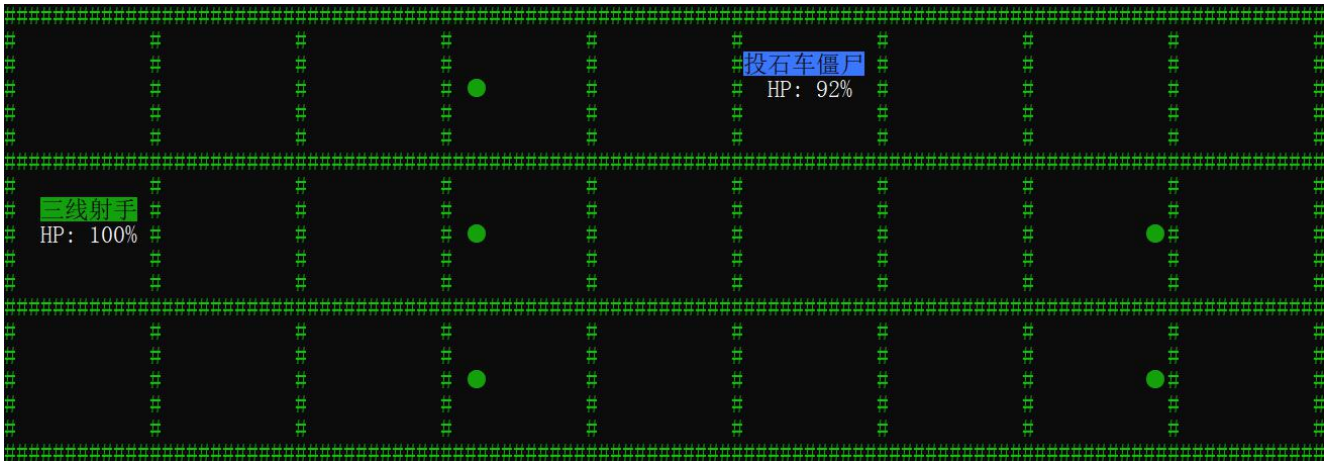


颜色比豌豆射手略深。

9.窝瓜(squash)

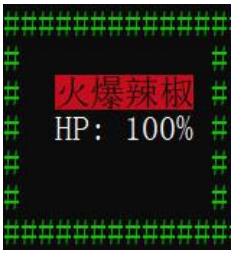
能消灭本格或前一格内的全部僵尸（除撑杆僵尸），有 1 秒的反应时间，草块效果与土豆雷爆炸一样。

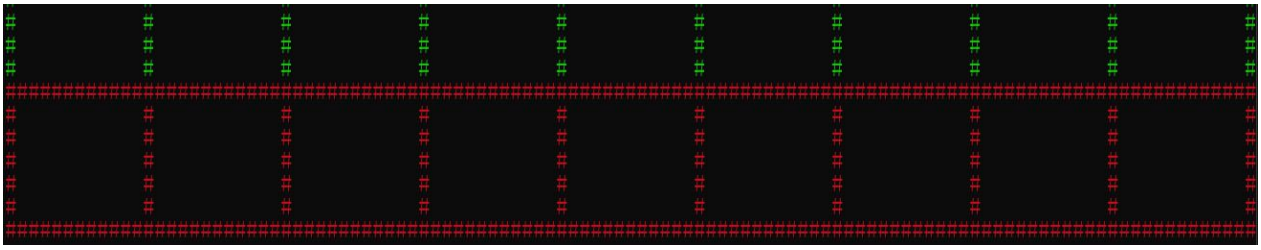
10.三线射手(threepeater)



同时在上中下三条线上发射豌豆。

11.火爆辣椒(jalapeno)





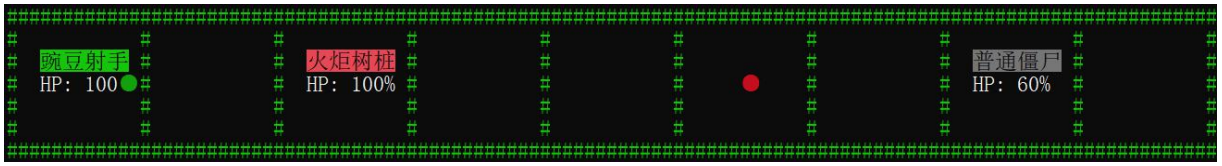
与樱桃炸弹爆炸过程类似，只不过范围为一整行。

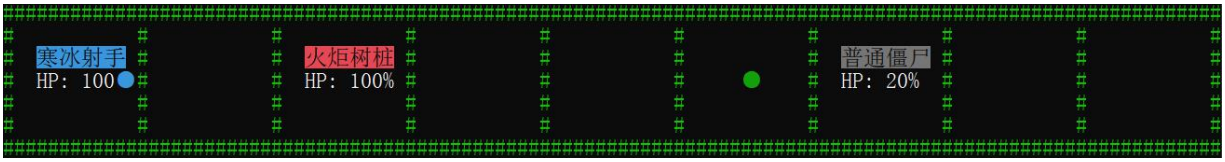
12.地刺(spikeweed)

投石车僵尸经过时会使之爆炸，自身也消失（因为投石车会停下来攻击前方植物，所以能让地刺攻击到的方法只有种在投石车当前所在格或者最右一格）。
其它僵尸不会攻击到地刺，会正常行进。

13.火炬树桩

普通豌豆经过会变成火豌豆，冰豌豆经过会变成普通豌豆，火豌豆不变。
火豌豆碰到被冰冻的僵尸会使之恢复正常。





14.高坚果(tall_nut)

也没啥好说的，一个更高强度的肉盾。

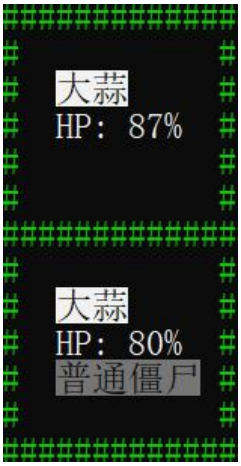
15.南瓜头(pumpkin)

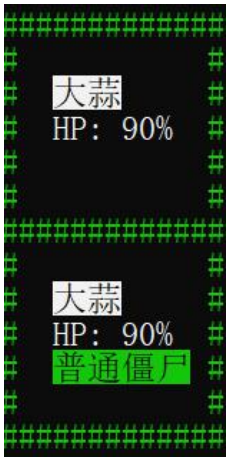
如果南瓜头里面种有植物，则显示南瓜头的血量。铲子先铲除里面的植物，然后才是南瓜头。

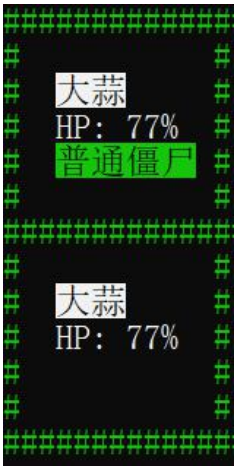
16.大蒜(garlic)

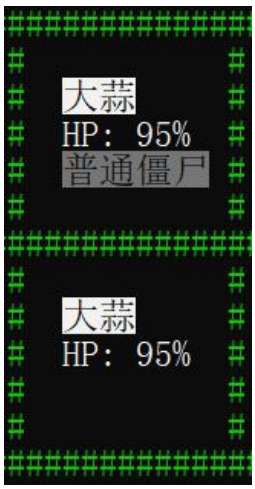
僵尸遇到大蒜的流程

咬一口->颜色变绿，停滞 0.7 秒->随机移到上面或下面一格->停滞 0.7 秒->颜色恢复正常，开始行动









（关于为什么大蒜血量不对，因为我截了 n 次才截到这几张）

另外，所有植物（除了南瓜头，它只是一对[]）在被吃的时候都会有闪烁。

闪烁的颜色与平常颜色是前景色与背景色调换的关系。

⑥Zombie 类（更新）

```
string name;//名称
Zombie_type type;
int attack;//攻击力
int defence;//当前生命值
int blood;//总生命值
int speed;//速度（实际上是走完一格需要的时间）
int speed_cooling;//当前行进完一格的进度
int score;//击败获得的得分
int x;
int y;//相对位置
int origin_color;//原本的颜色
int color;//当前的颜色

bool if_slowed;//是否正在减速
int slowing;//当前余下的减速时间
int garlicing;//当前余下的被毒时间
bool can_jump;//能否跳过植物
```

僵尸	总血量	死亡点	攻击	移动速度/格	无尽评星
普通僵尸	270	89	啃食100/s	5.6s较大波动	☆
旗帜僵尸	270	89	啃食100/s	3.7s	☆
路障僵尸	头370 +270	89	啃食100/s	同普僵	★
撑杆僵尸	500	165	啃食100/s	跑2.5s走5s	★★☆
铁桶僵尸	头1100+270	89	啃食100/s	同普僵	★☆
读报僵尸	报150 +270	89	啃食100/s	同普僵怒1.8s	★
铁栅门僵尸	门1100+270	89	啃食100/s	同普僵	★★
橄榄球僵尸	头1400+270	89	啃食100/s	2.5s	★★★
舞王僵尸	500	165	啃食100/s	滑步1.1s	★★
伴舞僵尸	270	89	啃食100/s	–	★★★★
鸭子救生圈僵尸	同对应僵尸	89	啃食100/s	同普僵	–
潜水僵尸	270	89	啃食100/s	4s	★★☆
冰车僵尸	1350	199	碾压（秒杀，对地刺王50）	逐渐减速3-8s	★★★★
雪橇小队僵尸	雪橇 300	队员同普僵	每个啃食100/s	雪橇1.3s	★☆
海豚骑士僵尸	500	165	啃食100/s	骑行0.9s	★★★☆
小丑僵尸	500	165	啃食100/s/自爆（3*3秒杀）	2.4s	★★★★☆
气球僵尸	气球20+270	89	啃食100/s	飞行2.7s	★★☆
矿工僵尸	头100 +270	89	啃食100/s	钻地1.2s	★★★
跳跳僵尸	500	165	啃食100/s	跳1.8走3.7	★★
雪人僵尸	1350	449	啃食100/s	4.1s逃2.1s	★
蹦极僵尸	450	0	抱走（除玉米炮都可）	停留3.6s	★★★
梯子僵尸	梯500+500	165	啃食100/s	有梯2.1s	★☆
投石车僵尸	850	199	篮球75 /碾压（同冰车）	2.7s	★★☆
巨人僵尸	3000	0	砸击（秒杀，对地刺王50）	同普僵	★★★☆
小鬼僵尸	270	89	啃食100/s	同普僵	★★★
IZ小鬼僵尸	70	22	啃食100/s	1.9s	–
红眼巨人僵尸	6000	0	砸击（秒杀，对地刺王50）	同普僵	★★★★★
豌豆射手僵尸	270	89	豌豆20 /啃食100/s	同普僵	–
坚果僵尸	头1100+270	89	啃食100/s	同普僵	–
高坚果僵尸	头2200+270	89	啃食100/s	同普僵	–
机枪射手僵尸	270	89	豌豆20*4 /啃食100/s	同普僵	–
窝瓜僵尸	270	89	跳跃（秒杀，对地刺王0）	2.1s	–
火爆辣椒僵尸	500	165	啃食100/s/自爆（一排秒杀）	同普僵	–
僵王博士	40000冒险/60000迷你	1	①随机放僵尸进攻②吐冰球或火球压植物及推车③悬挂蹦极僵尸偷植物④用小车碾植物⑤用脚踩植物②④⑤血量低于50%使用		–

1. 植物被豌豆篮球攻击血量<0才会消失，僵尸达到死亡点一般表现为断头。
2. 部分消耗类植物爆炸或攻击前进入硬直状态，免疫啃食砸击碾压，但可被僵尸自爆消灭，也可被偷走，被砸击立即爆炸，三叶草免疫啃食但可被砸击秒杀，小丑开盒后也会硬直，免疫一般攻击可被灰烬消灭或被大嘴吞噬，菇类和模仿植物需要完全唤醒和变身后才能进入硬直状态，否则会被啃食或被秒杀。
3. 灰烬攻击：攻击范围内全方位打击所有僵尸附带清除扶梯，灰烬攻击无视防具直接攻击僵尸本体。
4. 冰豌豆和冰瓜减速10s，冰菇冻结僵尸4s以上（已减速僵尸和水僵尸只能冻结3s）然后减速（总20s），对地下矿工及冰车无效，无法冻结只能减速空中僵尸（气球跳跃中的撑杆海豚跳跳被扔的小鬼）。
5. 僵尸防具有头盔型（路障等）盾牌型（报纸铁门梯子）气球三种，一般攻击须先打掉头盔才能伤害本体；盾牌能格挡部分平射类植物，能防冰和溅射对大喷菇地刺投掷类无效，气球只能被尖刺和仙人掌打到。
6. 南瓜扶梯：梯子下若增加植物扶梯不会消失，若减少则会消失，铲植物，种植紫卡及消耗类植物（咖啡豆除外）、补南瓜、植物被打死或被偷均会导致梯子消失。
7. 魅惑菇效果为强制向右前手变换方向，被矿工吃了不会掉头进屋，舞王被魅惑召唤的伴舞也会是魅惑状态，小丑被魅惑后爆炸效果类似[illegible]炸弹，豌豆僵尸和辣椒僵尸被魅惑也会击打或火烧己方植物。
8. 舞王和矿工不会出现在屋顶关卡，冰车不会出现在黑夜关卡。
9. 黄油概率25%，击中后定身僵尸所有动作，持续4秒，僵王和免疫被冰菇冻结的僵尸免疫定身

在游戏的最开始，僵尸的出场有固定的时间间隔。因为刚开始收集阳光，植物未形成有效的防御线，加上键盘操作增加了困难度，且植物栏未完善，所以不可能按照无尽模式 6 秒一波的速度来产生僵尸。于是我又双叒叕掐了表。

```
zombie_circle [ 0 ] [ 0 ] = 0;
zombie_circle [ 1 ] [ 0 ] = 200;
zombie_circle [ 2 ] [ 0 ] = 500;
zombie_circle [ 3 ] [ 0 ] = 800;
zombie_circle [ 4 ] [ 0 ] = 1100;
zombie_circle [ 5 ] [ 0 ] = 1400;
zombie_circle [ 6 ] [ 0 ] = 1700;
zombie_circle [ 7 ] [ 0 ] = 2000;
zombie_circle [ 8 ] [ 0 ] = 2300;
zombie_circle [ 9 ] [ 0 ] = 2600;
zombie_circle [ 10 ] [ 0 ] = 2900;
zombie_circle [ 11 ] [ 0 ] = 3200;
zombie_circle [ 12 ] [ 0 ] = 3500;
zombie_circle [ 13 ] [ 0 ] = 3650;
zombie_circle [ 14 ] [ 0 ] = 3800;
```

```
zombie_circle [ 0 ] [ 1 ] = 1;
zombie_circle [ 1 ] [ 1 ] = 1;
zombie_circle [ 2 ] [ 1 ] = 1;
zombie_circle [ 3 ] [ 1 ] = 1;
zombie_circle [ 4 ] [ 1 ] = 2;
zombie_circle [ 5 ] [ 1 ] = 2;
zombie_circle [ 6 ] [ 1 ] = 3;
zombie_circle [ 7 ] [ 1 ] = 3;
zombie_circle [ 8 ] [ 1 ] = 3;
zombie_circle [ 9 ] [ 1 ] = 4;
zombie_circle [ 10 ] [ 1 ] = 4;
zombie_circle [ 11 ] [ 1 ] = 4;
zombie_circle [ 12 ] [ 1 ] = 6;
zombie_circle [ 13 ] [ 1 ] = 6;
zombie_circle [ 14 ] [ 1 ] = 9;
```

上图是按照挑战模式 1-1 关卡的时间间隔，加上部分调整后得出的，在第一个摇旗僵尸出现前后的僵尸时间数量分布。其中 1-14 行代表第 1-14 波僵尸的数据，第 0 列为从游戏开始到僵尸出现的总时间（以 0.1 秒计），第 1 列为该波僵尸数目。比如，zombie_circle[7][0]=2000,zombie_circle[7][1]=3，代表在第 7 波在第 200 秒，有 3 个僵尸出现。而 zombie_circle[0][0]无用，zombie_circle[0][1]用来记录当前正在靠近的时间线，比如现在过了 530 个单位时间，正在准备产生第 3 波僵尸，那么 zombie_circle[0][1]=3。

whole_wave 用于记录总的时间间隔数目，即 14，用于判断是否已完成最初的循环。如果是，则当前时间跳回第 12 波，相当于接下来永远循环第 12-14 波的产生。

1.普通僵尸(ordinary，这个名字乱起的，因为它本来就叫 zombie，和基类名撞了)

2.摇旗僵尸(flag)

只有最后一波的时候才会出现，又因为无限循环僵尸潮故标记了是否出现过，所以理论上只出现一次。速度比普通僵尸大一点。

3.路障僵尸(conehead)

4.撑杆僵尸(pole_vaulting)

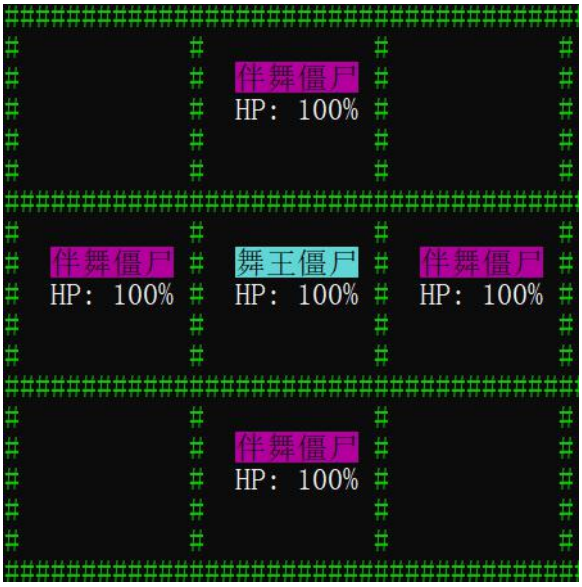
撑杆僵尸可以跳过土豆雷，食人花，窝瓜，并且窝瓜会消耗掉但是砸不中（已经跳过去了）。

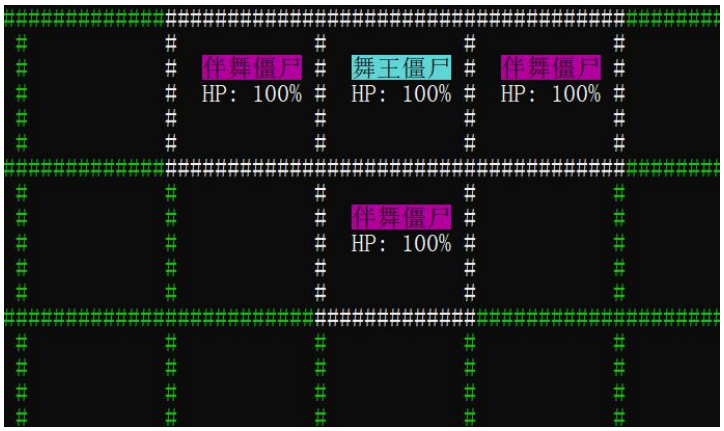
5.铁桶僵尸(bucket)

6.读报僵尸(newspaper)

失去报纸后速度由 4.7s/格变为 1.8s/格，颜色变红。

7.舞王僵尸(dancing)与伴舞僵尸(back_up)





如果某一个方位的伴舞僵尸缺失，舞王会在 2.5 秒后补上，召唤时会有闪光效果。
五个僵尸一体，其中任何一个被植物阻挡，其余的都会停止行动，若有一个被冰豌豆减速也会导致全体减速。总之，舞王与伴舞始终保持一致的行进速度。

8.小丑僵尸

5%几率在 4.5-7.8 秒自爆，95%几率在 13-23 秒自爆，吃植物的时间也算在内，所以理论上不可能走到最左边。

9.投石车僵尸

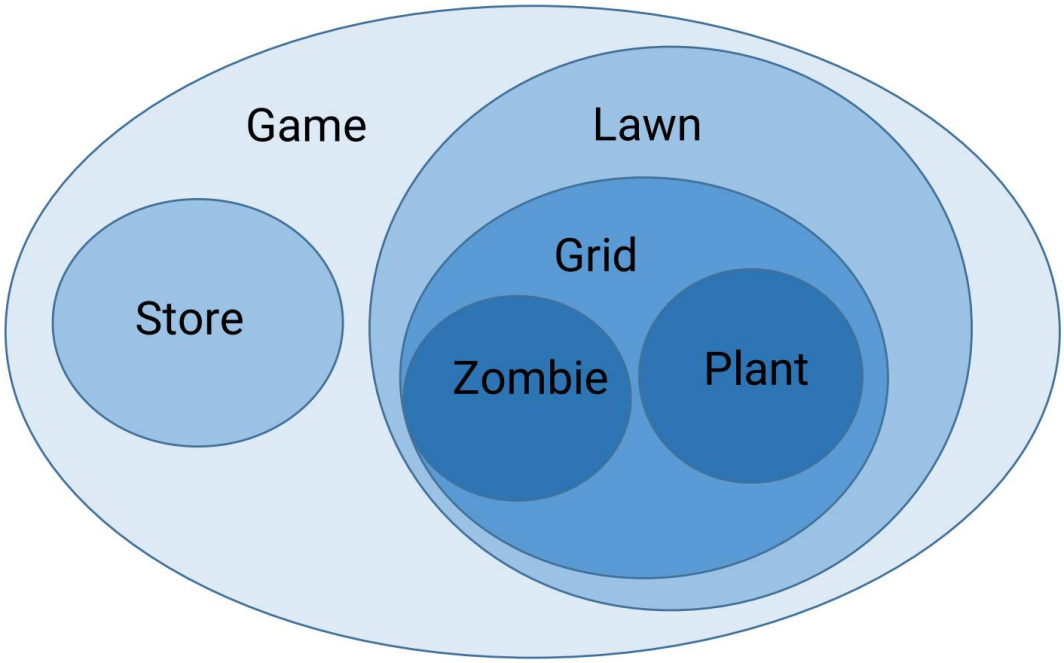
有 20 颗篮球，每 4 秒投掷一颗到最左边的植物上。如果前方没有植物或篮球投完，会以 2.5 秒/格的速度碾压行进。

```
Zombie* now = NULL;
while ( now == NULL ){
    int type = rand() % 100;
    if ( type < 13 ) now = new Ordinary();
    else if ( type < 43 && num >= 4 ) now = new Conehead();
    else if ( type < 53 && num >= 6 ) now = new Pole_vaulting();
    else if ( type < 73 && num >= 7 )now = new Bucket();
    else if ( type < 83 && num >= 7 )now = new Newspaper();
    else if ( type < 90 && num >= 9 ) now = new Dancing();
    else if ( type < 95 && num >= 11 ) now = new Jack_in_the_box();
    else if ( type < 100 && num >= 12 ) now = new Catapult();
}
```

僵尸出现几率如图所示，普通僵尸 13%，路障僵尸 30%，撑杆僵尸 10%，铁桶僵尸 20%，读报僵尸 10%，舞王僵尸 7%，小丑僵尸 5%，投石车僵尸 5%。num 的限制代表该僵尸只能在第几波之后出现，如第三行(type<53&&num>=6)表示撑杆僵尸只能在第 6 波之后出现，所以在第五波的时候只可能有普通僵尸和路障僵尸，两者按比例分配出现几率。

B.类之间的关系

6 个类的关系大致如下。



其中 Zombie 和 Plant 是基类，分别有 16 个和 10 个子类。

三、程序运行

A.运行操作方法（更新）

游戏主界面如图，主要部分是 5*9 的草坪，左侧为房屋的边界线。

商店：此为游戏的默认状态，光标（此处光标不是控制台光标，而是指当前选择的草块，植物，铲子等相对位置）可以在植物栏中移动，可以翻页，空格键确认后若满足购买条件则进入种植模式。

种植：光标可以在草坪间移动，种植成功则返回商店模式。购买状态下，选择的植物栏位置自动变为“取消种植”按钮，若想取消，则只需按↓键将光标移动到此按钮上，确认即可。

铲除：与种植类似。特别地，在植物栏最右一列按→键可选中铲子，确认即可进入铲除状态。

暂停：按 esc 键使游戏暂停，直到按下空格键才开始正常循环。注：只有在商店状态下才能暂停，也就是必须归还铲子或者种植完毕后按 esc 键才会起效。

此为游戏结束界面。（下次直接弄个墓碑图，不用自己画了）

注：因为操作比原游戏要麻烦一点，所以植物的选择与种植可能会花费一点时间。如果打不过就在 shopping 模式下按回车键，按一次阳光+1000，不超过 100000。

B.功能亮点

①循环方式

Overall.h 中定义了 single_time=100，代表单位时间。Game 运行的方式是无限循环，而 single_time 被用于 Sleep(single_time)，使得每次循环都会暂停 0.1 秒。将连续的时间分割有利

于对植物，僵尸，阳光的冷却计时，使草坪商店不因频繁刷新而产生闪烁，也保留了足够小的间隔使得肉眼能将画面衔接起来。

②视觉效果

为了让玩家能清楚地知道当前光标的位置，我设置了多个 selected 和 unselected 函数，用于更新草块，植物栏，铲子状态。

A 光标停留的植物栏（未按空格键确认）

B 未停留的植物栏

C 确认购买后（未种植）

D 光标停留时（未按空格键确认）

E 光标停留的铲子

F 未停留的铲子

G 选中铲子后（未铲除）

H 光标停留时

I 冷却时

J 光标停留时

AB,CD,EF,GH,IJ 是光标是否停留的不同表现；在 A 处和 E 处按空格键后跳转到 C 和 G，光标移动到草坪左下角；在 D 和 H 处按空格键后跳转到 A。

此为商店植物栏第一，二页，底部的小三角提示玩家可以往上或往下翻页。

③刷新机制

```
clear_zombie();
if ( in_refresh ) print( game );
if ( out_refresh ) print_circle( game );
if ( squashing ){
    squashing--;
    if ( squashing == 0 ) color = grass_color , print_circle( game );
    else if ( squashing == 10 ) color = light_yellow , print_circle( game );
}
if ( bombing ){
    bombing--;
    if ( bombing == 0 ) color = grass_color , print_circle( game );
    else if ( bombing == 10 )color = dark_red , print_circle( game );
}
if ( flashing ){
    flashing--;
    if ( flashing % 3 == 1 )color = default_color , print_circle( game );
    else  color = grass_color , print_circle( game );
}
in_refresh = 0;
out_refresh = 0;
```

在每次循环中，程序都会遍历每个草块执行更新函数。我用了 in_refresh 和 out_refresh 表示草块内部和边框是否需要重新绘制，squashing,bombing,flashing 分别表示浅黄，深红，白色高亮效果，用于模拟土豆雷，樱桃炸弹，窝瓜，小丑僵尸，舞王召唤等产生的效果。这样避免了在草块未更改时作出不必要的频繁刷新，保证了游戏的流畅度。

④各类数据的设置

草坪：采用标准的 5*9 格，缺点在于还没有实现像真正的植物大战僵尸那样，在最右侧多出可以看到僵尸出现但植物还攻击不到的时间段。

自然阳光间隔，向日葵阳光间隔，初始僵尸循环的时间数目：均为掐秒表得出，不一定十分准确，但尽可能保持接近。

植物，僵尸的攻击力，防御力，冷却时间等数据：查询搜狗百科而得。

子弹速度：未查询到，主要依靠暂停时观测两颗子弹的距离得出。

总之，为尽可能接近游戏的原本体验，我尽可能地获取了标准的数据。

⑤效果模拟

游戏中有许多特殊效果。

子弹：遇火炬树桩变种类
僵尸：撑杆的跳跃及前后状态变化，读报的前后状态变化，舞王灯光效果，吃大蒜后的颜色变化及停滞，冰冻与恢复的状态变化
植物：向日葵生产阳光时的高亮，植物被吃时的闪烁，窝瓜，土豆雷，樱桃炸弹，火爆辣椒的爆炸效果
这些都尽量实现了一定程度上的模拟，以便于玩家更清晰地感受到游戏当前状态。

四、遇到的问题与解决方案（更新）

char 类型的实参与 LPCWSTR 类型的形参类型不兼容
解决方案：
①项目->XXX 属性->配置属性->高级->字符集，由使用 Unicode 字符集改为使用多字节字符集
②#define char TCHAR

Visual Studio 2017 中“const char *”类型的值不能用于初始化 “char *” 类型的实体
解决方案：项目属性->C/C++->语言>符合模式项>选择否

list 的遍历删除
解决方案：

```
for(list<int>::iterator i=team.begin();i!=team.end();){
    if(f(*i)) team.erase(i++);
    else i++;
}
```

int 转化为 string
解决方案：#include<string> to_string(x);

等待玩家输入特定的按键并不显示在屏幕上
解决方案：

```
#include<conio.h>
while(true){
    if(_kbhit()){
        char c=_getch();
    }
}
```

头文件包含问题
解决方案：
①只声明 class A
②用指针
③尽量在.cpp 文件里包含头文件

子弹有时会径直穿过僵尸而不是消失
解决方案：我发现是因为每个循环都有一次僵尸移动，子弹移动，而子弹移动完毕后才判断该格是否有僵尸。如果在第 x 秒，子弹在僵尸左边一格，第 x+1 秒子弹右移，僵尸又刚好左移的话就会错开。之后我添加了子弹在移动前和移动后的两次判定。

撑杆僵尸与土豆雷，食人花，窝瓜的行动关系
解决方案：

首先有一点规则：我设置的撑杆僵尸跳跃效果是向前移动一格，虽然实际上似乎没有跳到一格这么远，但是为了显示明显的视觉效果以提醒玩家它确实跳了，所以就这样定了。

1.土豆雷是最简单的，因为它的范围只有自己这一格。因为我设计的顺序是全部植物行动完毕之后再是全部僵尸行动，然后继续循环，所以在撑杆僵尸这一步行进到土豆雷格子时，向前跳跃一格即可。

2.食人花的攻击范围包括了前一格，而撑杆僵尸的设定是当前格有植物才能跳（要是前一格就能跳的话刚好跳在食人花这一格），所以我直接设定了食人花无法攻击到撑杆僵尸。原始游戏的动画大致是，撑杆僵尸跑到食人花前一格->食人花张嘴往前咬->这一瞬间撑杆僵尸成功越过->食人花咬到一嘴空气。等下次用图形界面再试试这个效果。

3.窝瓜与食人花攻击范围相同，但不同的一点是食人花不是一次性植物，而窝瓜会被撑杆僵尸消耗掉。但撑杆僵尸又不能像遇到食人花那样跳到下一格，按照习惯来说应该是跳到窝瓜所在的一格，同时窝瓜砸向前面一格，相当于两者互换位置，窝瓜消耗，撑杆僵尸变无杆僵尸。若撑杆僵尸在窝瓜前一格跳跃则正好跳到窝瓜所在格，若窝瓜忽视撑杆僵尸在前一格的行动则失去了互换位置的效果，不合常理。所以撑杆僵尸的“移动一格”功能不是由撑杆僵尸的 action 函数实现的，而是由窝瓜的 action 函数实现的。当攻击目标是撑杆僵尸时，窝瓜将该僵尸前移一格，自身消失。

舞王与伴舞僵尸统一行动问题
解决方案：

这个问题真的要把我搞死了（我为什么要添加舞王）。首先，舞王以 1.8 秒/格的速度前进两格，然后闪光灯亮召唤伴舞僵尸。最重要的一点是，舞王与伴舞同步。影响僵尸行进速度的有四个因素：被减速，恢复速度，遇到植物，吃了大蒜。这些都必须同步，而且舞王中途召唤的伴舞僵尸也必须与大部队同步……

1.冰冻与速度恢复：控制同步的函数由 Bullet 实现，当冰豌豆或火豌豆攻击僵尸时，添加对僵尸类型的判定，五个僵尸一同改变。

2.植物阻挡：
suo 舞王行动标准：伴舞僵尸及自己所在格子上都没有植物或自己格子上有植物
伴舞僵尸行动标准：舞王能行动或自己格子上有植物
（行动是指移动或者攻击植物，显然只能择其一）

3.中途召唤伴舞僵尸：速度，速度冷却，减速时间都要与舞王统一。舞王位于中央，召唤伴舞僵尸后，程序按照遍历草坪的顺序，还应当经过舞王下方和后方的僵尸所在的草块，才算是完成一个遍历。所以下方与后方的伴舞僵尸在复制舞王速度冷却与减速时间时，速度冷却应当-1，减速时间应当+1，这样执行过 action 函数后才能保证五者速度统一。另外，当舞王未被冰冻（即冰冻时间==0）时不应该+1，否则伴舞僵尸下一循环中会被认为速度恢复，从而导致速度和速度冷却异常。

这里有一个问题，最开始一次召唤时，若将召唤时间选定在舞王恰好前移一格的时间点，则会造成下方伴舞僵尸的速度比其它四个少 1（比如舞王 1.2 秒走一格，则 2.4 秒召唤则会出

现这种错误，在 2.5 秒，2.9 秒等时间点则一切正常）。虽然只慢了 0.1 秒，不过不知怎地我就看出来了。BEDUG 一天之后终于发现，正是因为舞王向前移动了一格，新增的下方僵尸在此次不会被遍历到，因为程序会接着舞王原来的那格往下遍历，即会经过新增僵尸的后方一格。于是舞王后方的伴舞僵尸不会受影响，只有下方的会产生错误。

更新：12 不变，但 3 考虑不完整。3 的写法事实上还是将 5 个僵尸看作独立的个体，只是赋值部分相同所以看上去是一体。虽然解决了新僵尸的速度冰冻问题，但对于冰豌豆的效果，统一赋值又会产生类似的问题，即不确定之后是不是恰好前移了一格。大蒜效果类似，因为行动有先后关系，因此可能现有部分僵尸行动，然后执行到吃大蒜的僵尸，再之后的僵尸就会停止行动，又将导致不同步。

总之，一切问题都是因为遍历处理导致的不同步问题。

所以我直接在每次遍历时跳过伴舞僵尸,以舞王为中心执行 4 个伴舞僵尸及自身的 action 函数，每次统一速度和冰冻效果，全部草块遍历完毕后再遍历一遍，将跨越格子的僵尸移动。

诡异的疯狂僵尸问题

不知道出了啥差错，在路障僵尸出场后，有时其中的一个路障僵尸会突然呈现疯狂的暴走状态，以摧枯拉朽之势啃食高坚果与南瓜头，速度快如风，在我愣神之际已然冲到草坪最左边并送给我一个代表挂掉的墓碑。

经过两个小时的 DEBUG，我终于发现问题出在火豌豆上面。只是因为路障僵尸出场后我习惯加火炬树桩，所以才误以为是路障僵尸的问题。

僵尸的参数中有一个 slowing，用于记录冰冻状态。每次循环先-1 再判断，100 时表示被冰冻，速度攻击力减半，0 时表示冰冻结束，速度攻击力加倍。所以冰豌豆攻击时将 slowing 改为 101，而火豌豆能解除冰冻效果，所以火豌豆攻击时将 slowing 改为 1，这样-1 后为 0，僵尸恢复正常。

但我忘了被火豌豆攻击的僵尸可能没有处于冰冻状态。它的速度攻击力是正常的，但火豌豆将它的 slowing 改为 1 之后，执行它的 action 函数时代码会以为它刚刚从冰冻状态恢复，于是将其速度攻击力加倍。普通僵尸很容易就被消灭了，但路障僵尸防御力比较强，最前面的那个挨了几下火豌豆之后还没死，而速度和攻击力已经变为了 2 的 n 次方倍，于是被强化的路障僵尸所向披靡，当时那种诡异的状况着实吓人。

给僵尸加 buff 的火豌豆，这个极具戏剧性和视觉冲击力的现象很好地暴露了我在编写代码时考虑不全面和思维不严谨的问题，所以很有必要作为压轴问题记录下来警醒自己。

五、总结与反思

程序的不足之处在于，各个模块的功能其实并不是很明确地被区分开来。因为有包含与被包含的关系，所以下一层的类的操作也能由上一层直接操作来实现，比如 game 包含 lawn,lawn 包含 grid，则 game 可以直接处理 grid，这样就导致我有时并不是很明确应该将具体的操作交给哪一层，模块功能比较混乱，在我的程序中大概见不到 Demeter 法则的影子。

另外，因为我太弱，在课设开始的两个周之前连.h 和.cpp 都不会用，更不知控制台为何物，一上来就写简易版植物大战僵尸，未免有些手忙脚乱，一边学控制台界面编程一边努力理清.h 和.cpp 的关系。在写的过程中，由于严重缺乏面向对象编程的经验，我的程序经常需要修补。所以整个程序代码并没有呈现出一个很清晰的框架，基本上是走一步算一步。

回过头发现，其实控制台这部分确实不难，只不过一开始由于没有了解过，加上众多复杂的函数才让我心生恐惧。事实上写一个简易版的植物大战僵尸用不到什么特别高深的代码知识，上图众多版块的大量函数中其实只用到了设置窗口，光标，文字颜色三类操作，多余的根本派不上用场。

关于控制台的操作函数都放在了 overall.h 里面，下图即为 overall.h 包含的全部函数。

```
//设置窗口标题
void set_screen_title(TCHAR* x);
//设置窗口大小
void set_screen_size();
//设置颜色
void set_color(int color);
//带颜色输出
void print_in_color(const string& x, int color = default_color);
//设置光标位置
void set_cursor(int x, int y);
//隐藏光标
void hide_cursor();
```

一旦完成了最基础的全局函数，剩下的工作与 windows.h 就没什么关系了。整个 project 考察的主要就是面向对象的思维方式，控制台的确只是一个实现的工具。代码的实现需要的更多是细心与耐心，以及统筹规划的能力，而不是高级的算法和数据结构。

这次课设让我想起了上学期的 SICP，两门选修课的共同点就是可以把人逼疯的 project。当然，这也的确让我受益匪浅。正是高难度的课程逼迫我发奋学习相关知识点，并且通过写大量的实践练习发掘容易忽视的细节，掌握不熟悉的方法，改正错误的习惯，从而提升自己的编程水平。说实话，如果植物大战僵尸这个项目没有 ddl，那写起来真是一种休闲娱乐。

谢谢！