
# 植 物 大 战 僵 尸

报告名称 PVZ设计

课程名称 高级程序设计

学生姓名 时欣

学　　号 191220097

实验日期 2021年4月3日

# 目录

# 1 实验内容

实现一个基本版的植物大战僵尸，主要内容包括：

## 1.1 界面显示

1. 3行7列的地块；
2. 商店内容的显示；
3. 计分表分值；
4. 阳光拥有值的显示
5. 光标显示————用于进入商店时，移动光标来选择购买的植物；确定购买后，移动光标来选择植物存放的地块，并在相应的地块显示植物；（除了种植的确认）如果确定后又取消（”x”），则再次显示光标，重复以上步骤。
6. 僵尸显示。

## 1.2 游戏主体

1. 僵尸：普通僵尸；
2. 植物：豌豆射手，向日葵；
3. 攻击工具：子弹；

## 1.3 游戏逻辑

1. 塔防逻辑：植物有其对应的生命值和攻击力（每秒对僵尸造成多少伤害），僵尸有其对应的生命值，攻击力（每秒对植物造成多少伤害）和速度（走到下一个地块需要多少秒）。
   当僵尸走到植物所在地块时可以对植物进行攻击；当植物的生命值减到0则植物死亡，从植物列表中移除，而僵尸可以走到下一个地块对

其中的植物攻击。如果僵尸将最左侧地块的植物攻击致死，则游戏失败。

豌豆射手的作用是每秒发射子弹，真正起作用的是子弹。子弹有其伤害值（一个子弹对僵尸造成多少伤害）和速度，当子弹与僵尸在同一个地块时，子弹对僵尸造成伤害，随后子弹消亡。当僵尸的生命值减到0则僵尸死亡。

2. 购买逻辑：植物有其对应的阳光消耗值，选定购买以后，阳光拥有值扣除相应的数量。如果阳光拥有值不足，则无法购买，对用户给出提示。

# 2 实验目标

实现较好的交互界面，完成游戏正确的逻辑，尽可能丰富植物和僵尸的种类。

# 3 实验环境

1. 设计语言：c++
2. 运行环境：Windows
3. 设计环境：Visual Studio 2019

# 4 实验设计思路

运用c++语言，运用面向对象的思路设计。

1. 需要实现的各类间关系如下图

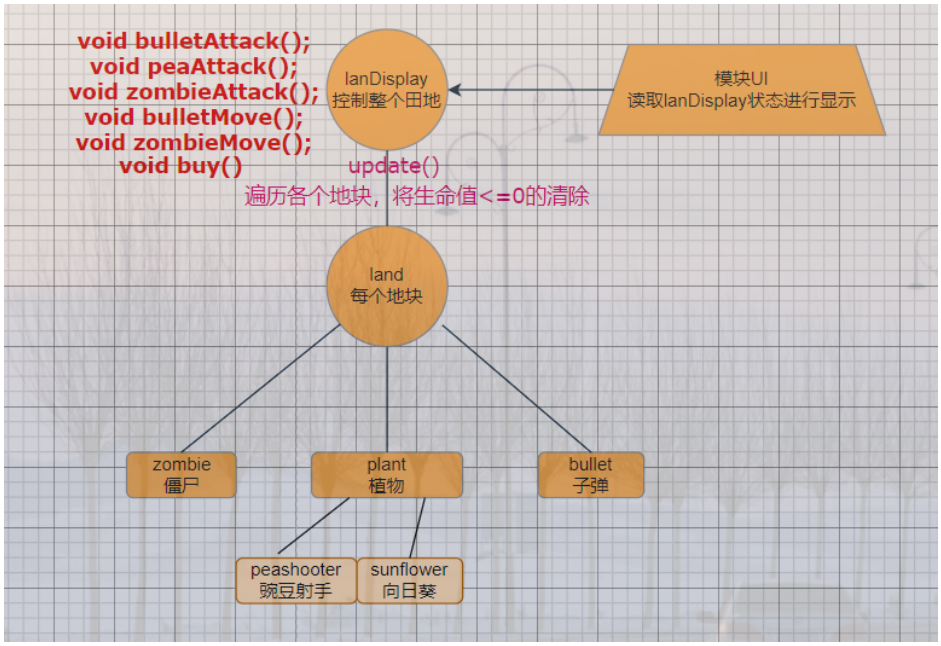


图 1: 各类间关系

图中的各个图形标出的都是一个类（除了UI模块）。zombie，bullet，peashooter由land控制，而land由lanDisplay控制。（由于还没学到继承，所以plant下的植物以及zombie下的各种僵尸只是初步想法，实际实现的时候因为目前游戏主体还比较少，暂时还是用单独类实现的）。

2. 设计思路

如上图所示，lanDisplay类下有一个land类成员，而land类下有zombie，bullet，peashooter成员。采用类对象直接通信的机制。即lanDisplay类一旦发现zombie，bullet，peashooter的 time==speed（在下面会有详细说明），则相当于这三个类对lanDisplay发送消息————我要移动了，或者我要攻击了。然后lanDisplay来控制，经过判断以后看能不能允许它们移动或者攻击。也就是能对地块状态进行改动的只有lanDisplay类，如果其它的类想要对地块状态进行改动，只能给lanDisplay发消息，通过lanDisplay来仲裁能否改动。

而对于程序显示和定时刷新，我选择的是，每一轮攻击之后使用Sleep函数让程序休眠500ms，再继续，这样每一轮的输出都会停顿一下，以

实现视觉上的动态变化效果。每一轮刷新，UI模块会读取lanDisplay的状态来输出。

最后在main函数里面调用函数实行整个游戏的逻辑————用一个while循环，在游戏未失败的情况下，反复执行一轮一轮的攻击与移动。并且每当一段时间之后利用random随机选择一行产生僵尸，以及每当一段时间之后对系统阳光增加50.

# 5 各个类的设计

## 5.1 zombie类

```
class zombie {
private:
    int life;       //生命值100
    int speed;      //多少次动一下，对应于time（6）
    int time;       //在循环中被遍历了几次,规定次数以后移动
    int power;      //伤害值（10）
    int attackTime;     //遍历多少次攻击一下（5）
    int attackCnt;      //遍历多少次了

public:
    zombie();
    zombie(const zombie& zb);       //拷贝构造函数

friend class land;
friend class lanDisplay;
};
```

图 2: zombie类数据与操作

各个数据成员与成员函数的含义如注释所标。之所以要实现拷贝构造函数是因为，僵尸移动时需要带着它的状态（生命值等）走，也就是需要先在原先的地块删掉它，然后在下一个地块新建一个和它一样的对象。如果被寒冰射手打中，那么减速时间持续五次遍历的时间，所以需要在每次遍历的时候对遍历次数的计数做额外处理。我没有另外写一个函数，而是直接放在僵尸的move里面了，也就是在判断完是否还活着以后，直接根据是否在减速的状态，就对其的计数做操作。

僵尸的攻击比较复杂，采用虚函数和动态绑定，针对不同的情况给

总控制类发出不同的返回值信号来显示自己需要干什么注意有道具的僵尸————撑杆僵尸和读报僵尸，在道具消失之后应该让其成为普通僵尸（而且生命值等状态需要延续，为此需要写非默认构造函数）而其中对于move来说

1. 撑杆僵尸: 它的move应该首先判断左边一格是否有植物并且此植物并非高坚果，如果true那么就向控制类发出信号1————表示请求跳过左侧植物；如果false，那么对遍历次数计数，和普通僵尸一样，到次数移动

2. 投石僵尸：它的move需要判断石头有没有扔完石头，如果扔完了又到了移动的时候，那么就往左走。如果没有扔完但是左边没有任何植物，那么它也会往左走。

3. 读报僵尸：它的move和普通僵尸一样，但是需要注意，给它变成普通僵尸的时候要增加移动速度，此处也需要通过非默认构造函数进行设置

4. 路障僵尸和小丑僵尸：move和普通僵尸一样

其中对于attack来说

1. 撑杆僵尸：它的attack是和普通僵尸一样（其实只有在面对高坚果的时候有效）

2. 投石僵尸：它的attack是首先需要判断还有没有石头。如果有，那么到扔石头的时候，就告诉控制类要攻击植物了（从它的位置开始往左遍历，攻击最近的僵尸）。如果没有，就和普通僵尸一样，当前地块上有植物的话就让植物死亡

3. 读报僵尸和路障僵尸：和普通僵尸一样

4. 小丑僵尸：首先会有一个概率得出它要不要爆炸，如果不爆炸就和普通僵尸一样。如果爆炸就将周围3*3的植物都生命值置零。

## 5.2 peashooter类

```
class peashooter {
    private:
        int life;       //生命力
        int power;      //攻击力
        int cost;       //花费阳光数
        int time;       //在循环中被遍历了几次
        int speed;      //被遍历多少次产生一个子弹
    public:
        peashooter();
    friend class land;
    friend class lanDisplay;
};
```

图 3: peashooter类数据与操作

各个数据成员与成员函数的含义如注释所标。这个类比较简单，只需要实现默认构造函数即可。

## 5.3 bullet类

```
class bullet {
    private:
        int speed;      //多少次动一下
        int power;      //攻击力
        bool life;      //是否活着（在对僵尸进行攻击后，生命即消失）
        int time;       //在循环中被遍历了几次
    public:
        bullet();
        bullet(const bullet& bt);
    friend class land;
    friend class lanDisplay;
};
```

图 4: bullet类数据与操作

各个数据成员与成员函数的含义如注释所标。比较简单，不再赘述。

## 5.4 land类

```
class land {
    private:
        vector<zombie> Zombie;      //存储植物
        peashooter Pea;             //豌豆
        vector<bullet> Bullet;      //子弹
        bool ifPlant;       //是否有植物
    public:
        land();
        void addBullet();       //当前地块增加一个子弹
        bool addPeashooter();   //当前地块增加一个豌豆射手
        void addZombie();       //当前地块增加一个僵尸
        int getZbSize();        //当前地块僵尸的数量
        int getBtSize();        //当前地块子弹的数量
        bool ifplant();         //当前地块是否种有植物
    friend class lanDisplay;
};
```

图 5: land类数据与操作

各个数据成员与成员函数的含义如注释所标。每一个land类的对象都是一块地块，各自管理各自的有关植物、僵尸和子弹的状态。

## 5.5 lanDisplay类

```
class lanDisplay {
private:
    land Land[3][7];    //田地所有的3行7列的地块
    int score;      //玩家得分
    int suNum;      //阳光总数
    bool lose;      //状态是否为失败
public:
    lanDisplay();           //默认构造函数
    void update();          //更新状态，将生命值为0的从地块中删除
    void bulletAttack();    //让所有地块上的子弹发动攻击
    void peaAttack();       //让所有地块上的豌豆射手发动攻击
    void zombieAttack();    //让所有地块上的僵尸发动攻击
    void allAttack();       //让所有地块上的植物、僵尸、子弹该移动移动，该攻击攻击
    void bulletMove();      //让所有地块上的子弹移动
    void zombieMove();      //让所有地块上的僵尸移动
    void genZombie(int x);  //在第x+1行的第7个地块产生一只僵尸
    void addSun();          //增加阳光总数
    void buy();             //购买植物
    void plant();           //种下植物
    bool ifLose();          //获取lose状态，看是否已经失败
    friend void Show(lanDisplay ld);        //友元，读取lanDisplay状态进行输出
};
```

1. 各个数据成员与成员函数的含义如注释所标。lanDisplay类作为整个设计的核心控制类在本程序中占有极其重要的地位。

2. 可以看到lanDisplay类中有数据成员land Land[3][7]，即将所有地块以二维数组的形式进行组织，各个地块中又存储着属于自己的有关植物、僵尸和子弹的状态。

3. 而lanDisplay类是land、植物、僵尸和子弹类的友元，因此通过lanDisplay类可以对地块的各种状态进行改动，从而实现对于地块中各个游戏主题的移动和攻击逻辑进行控制。

4. lanDisplay类中的成员函数bulletAttack();是通过对于Land[3][7]中的每一个地块中的bullet成员进行遍历来实现的。回顾bullet类，它的数据成员有

```
1
2 int speed;
3 int power;
4 bool life;
5 bool ifAttack;
6 int time;
```

bullet.h

每一次遍历，如果发现子弹所在地块有僵尸，则僵尸被扣除子弹攻击一次会产生的伤害值，然后将子弹的生命值置零。

5. peaAttack();也是通过对于Land[3][7]中的每一个地块中的bullet成员进行遍历来实现的。其实豌豆射手本身是不能对僵尸发动攻击的，因此它的攻击其实就是定时产生子弹。每一次遍历，如果发现time的值已经和speed的值相等则令豌豆射手产生子弹，否则time的值加一————以此来模拟豌豆射手的定时发射子弹。

6. zombieAttack();是通过对于Land[3][7]中的每一个地块中的zombie成员进行遍历来实现的。攻击逻辑为：在当前地块有僵尸且其生命值大于0，并且当前地块有植物且其生命值大于0的情况下，如果僵尸的attackCnt的值已经和attackTime的值相等，则僵尸可以对植物发动攻击，对应的植物扣除相应的生命值。否则attackCnt的值加1————以此来实现僵尸对植物的定时攻击。

7. bulletMove()是通过对于Land[3][7]中的每一个地块中的bullet成员进行遍历来实现的。如果当前地块没有僵尸，且子弹已经到了其该移动的时候————time==speed，则在下一个地块新建一个bullet，把当前地块的bullet删除，然后把time置零。如果还没到移动的时候，time的值加一。
   当然，如果已经在最后一个地块了，那么直接将其删除即可。当然，如果当前地块有僵尸，那么它暂时不能移动，因为接下来它需要对僵尸进行攻击。

8. zombieMove()是通过对于Land[3][7]中的每一个地块中的zombie成员进行遍历来实现的。如果当前地块没有植物（ifPlant == 0），并且僵尸已经到了其移动的时候（time == speed），那么将当前地块的僵尸拷贝到下一个地块，将当前地块的僵尸删除。如果还没到移动的时候，那么将time的值增加一。
   需要注意，如果僵尸已经在最左边的地块而且也没有植物了，那么僵尸越过这个地块以后lanDisplay的lose状态将会被置为1，也就是说玩家失败。

总的来说，lanDisplay类是通过与zombie，bullet，peashooter的通信来控制地块状态的，一旦发现它们的time==speed则相当于本类对lanDisplay发送消息————我要移动了，或者我要攻击了。而lanDisplay来判断，能不能允许它移动或者攻击。

1. 而update函数则是对地块状态进行更新，主要是全部遍历，将生命值为0的游戏主体从地块中删除，发现有多少死亡的僵尸，则将总分加上相应的加分，并且根据lose状态判断玩家是否失败来决定是否游戏结束。

2. buy函数和plant函数本来是想放在lanDisplay类外面，单独作为一个类并和lanDisplay类交互的。可是因为是直接对lanDisplay中的地块状态进行操作，所以干脆放在了一个类里面。通过查阅资料，我知道了如何获取键码。
   我是通过光标的停留位置来判断用户的选择的。用户可以通过按下wasd键

等来移动光标位置，y和enter来确认选择，x来取消。对地块的状态改动，是发生在选择好植物并确认种植以后，一旦获取到相应的键码，则在对应的地块上增加用户选择的植物。（当然如果判断到键码但发现阳光总数不够，则不会对地块进行改动，而是在界面下方对用户进行提示。

相关代码为

```
int ch;
while (1)
{
    if (_kbhit())
    {

        ch = _getch();
        if(ch == x) do y;
    }
}
```

mouse.cpp

先判断是否有按键按下，如果有，获取键码，根据键码做相应事件。从买到种植，如果中间没有取消工作，是一个顺序循环读取键码的过程，所以采用while循环。

# 6 程序亮点

1. 选择用一个类实现总控制的方法，模仿各个进程给操作系统发信息请求资源调度等让操作系统仲裁决定的方式，让其它类都给lanDisplay类发送自己的状态，而lanDisplay类接收到信息以后做出判断和相应操作。

2. UI和游戏逻辑分离，只读取lanDisplay状态进行输出，不做别的多余的动作

3. 游戏有主界面，当前有两个选项，如下图

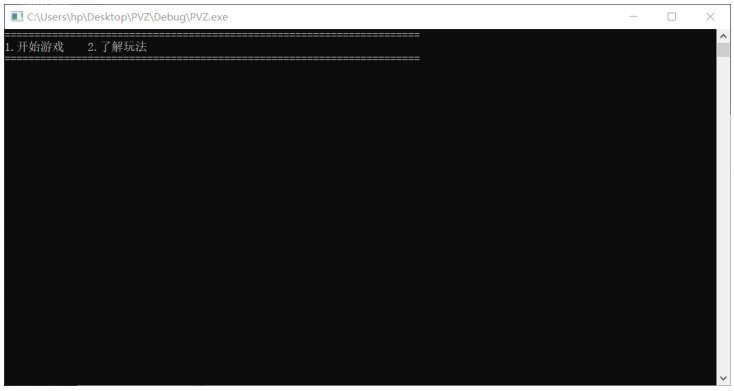


图 6: 主界面

选择1则进入游戏界面，如下图

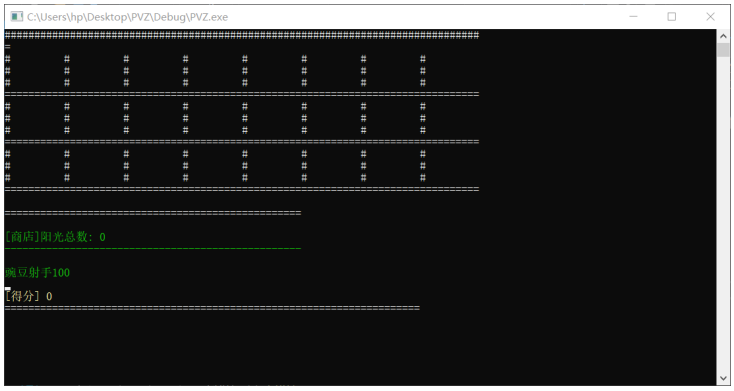


图 7: 游戏界面

选择2则进入游戏玩法说明，如下图

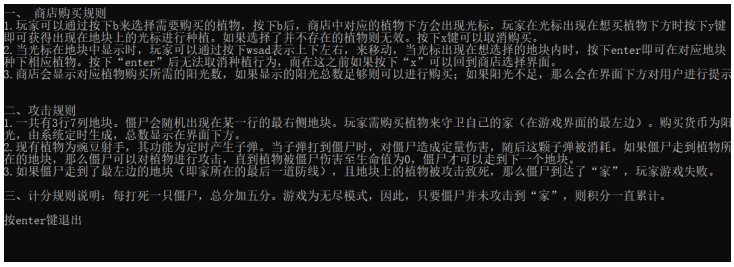


图 8: 游戏规则

4. 对用户友好，如果阳光购买植物时不足会给出提示

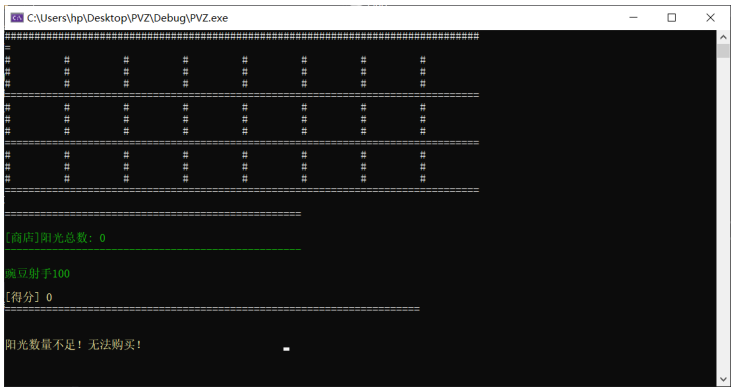


图 9: 提示如最后一行

不同的颜色更加醒目

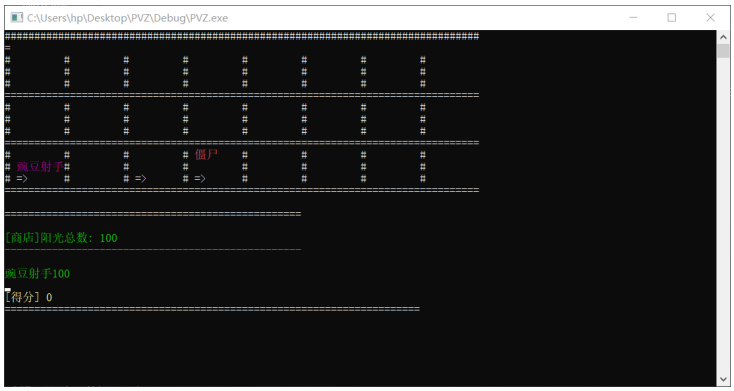


图 10: 颜色展示

如果中途购买或种植时选择错误，可以按下“x”取消。如果用户游戏失败，会有一个持续1s的“游戏失败”提示，然后返回主界面，进行下一轮新的游戏。如果用户在已种植植物的地块上选择种植新的植物，会在游戏界面下方提示，并且可以重新选择。

# 7 玩法

## 7.1 游戏开始界面

1. 选择开始游戏则进入游戏界面

2. 选择了解玩法，则显示游戏规则；按下enter则返回主界面。

## 7.2 商店购买规则

1. 玩家可以通过按下”b”来选择需要购买的植物，按下”b”后，商店中对应的植物下方会出现光标，玩家在光标出现在想买植物下方时按下”y”键即可获得出现在地块上的光标进行种植。如果选择了并不存在的植物则无效。按下“x”键可以取消购买。

2. 当光标在地块中显示时，玩家可以通过按下”w””s””a””d”表示上下左右，来移动，当光标出现在想选择的地块内时，按下“enter”即可在对应地块种下相应植物。按下“enter”后无法取消种植行为，而在这之前如果按下”x”可以回到商店选择界面。

3. 商店会显示对应植物购买所需的阳光数，如果显示的阳光总数足够则可以进行购买；如果阳光不足，那么会在界面下方对用户进行提示

## 7.3 攻击规则

1. 一共有3行7列地块。僵尸会随机出现在某一行的最右侧地块。玩家需购买植物来守卫自己的家（在游戏界面的最左边）。购买货币为阳光，由系统定时生成，总数显示在界面下方。

2. 现有植物为豌豆射手，其功能为定时产生子弹。当子弹打到僵尸时，对僵尸造成定量伤害，随后这颗子弹被消耗。如果僵尸走到植物所在的地块，那么僵尸可以对植物进行攻击，直到植物被僵尸伤害至生命值为0，僵尸才可以走到下一个地块。

3. 如果僵尸走到了最左边的地块（即家所在的最后一道防线），且地块上的植物被攻击致死，那么僵尸到达了“家”，玩家游戏失败。

## 7.4 计分规则说明

每打死一只僵尸，总分加五分。游戏为无尽模式，因此，只要僵尸并未攻击到“家”，则积分一直累计。

## 7.5 总结与感想

实验一开始的时候根本无从下手，感觉脑袋一团乱麻。后来自己一遍一遍的捋，总算大概有了个想法出来。只是当时是各个类自己管自己的攻击，于是写到这的时候发现不行，当每个类自己移动攻击的时候是无法判断操作是否合理的。于是又重构，设计了一个总的控制类，让它来做所有会改变状态的操作。这样一来，思路清楚了很多。

主要遇到的bug其实是在输出对齐上。由于没有用宏，一些输出的偏移量感到不合适以后进行了改动，却在将输出位置转化为对应的地块坐标时出了错误。debug才发现没有全部改动好。于是改成了宏，不得不说，在必要的地方用宏还是一个比较好的习惯。

而写完之后再回顾，发现还是并没有想象中那么难，要写的东西也并不多。主要是把逻辑理顺了，该注意的细节注意了就行。

后面学了继承应该是要对植物和僵尸的类进行重构的，否则一个一个类写过于繁杂重复。

希望如果有设计不合理的地方能够得到学长/学姐的建议（让我免于到最后时刻发现整个架构不行得要重构的绝望— ω ‘)），目前感觉好像还可以。