

课程设计二

控制台版植物大战僵尸

171860574 胡育玮 计算机科学与技术系

目录

- 一、项目概述
- 二、设计重点
- 三、具体设计
- 四、运行流程
- 五、遇到的问题 and 解决方案

一、项目概述

1.1 概述

根据题目要求，本项目要求实现简化的控制台版的植物大战僵尸游戏。具体来说，至少要实现如下的要素：

- 三行七列的战场
- 系统自然阳光的生成
- 2 种僵尸
- 3 种僵尸属性：生命值、攻击力、速度
- 2 种植物
- 3 种植物属性：购买所费阳光数、生命值、攻击力
- 植物的购买、种植地块的选择、植物的种植
- 分数统计

最终，在 Windows 10 平台上，我实现的要素有：

- 可扩展的 x 行 y 列的战场: x 和 y 可以是任意大于 1 的整数
- 系统自然阳光的生成
- 5 种僵尸：*普通僵尸、路障僵尸、铁桶僵尸、橄榄球僵尸、读报僵尸*
- 6 种僵尸属性：生命值、攻击力、速度、是否被冰冻、杀死后得到的分数、攻击间隔
- 7 种植物：*向日葵、豌豆射手、双枪射手、坚果墙、西瓜投手、*

火爆辣椒、寒冰菇

- 4种植物属性：购买所费阳光数、生命值、攻击力、攻击间隔
- 植物的购买（商店）、种植地块的选择、植物的种植
- 铲子功能
- 僵尸的生成：不同种类的僵尸按照不同的概率生成
- 分数和阳光的统计

最终的游戏效果如下：（以下展示 3 行 7 列的战场，下同。实

际上战场尺寸可调）

```
C:\WINDOWS\system32\cmd.exe

=====
# # # # # # # Papezom
# # # # # # # Buckzom
# # # # # # #
# SunFlower # SunFlower # SunFlower # ICESHROOM! # # # #
# # # # # # #
# # # # # # #
=====
# # # # # # # Con#Papezom
# # # # # # # Nzomb
# SunFlower # SunFlower # SunFlower # # # # Pap
# # # # # # #
# # # # # # #
=====
# # # # # # # Nzom# Nzombie
# # # # # # # Papezom
# SunFlower # SunFlower # SunFlower # # # # N
# # # # # # #
# # # # # # #
=====

[STORE] SUN:1200 [SCORE]:0

SUNFLOWER:50 PEASHOOTER:100 DOUBLESHOOTER:190
NUT:250 MELONPITCHER:400 PEPPER:600
ICESHROOM:550
=====
```

```
C:\WINDOWS\system32\cmd.exe

=====
# # # # # # # Papezom # Nzombie # Conezom
# # # # # # # Nzombi# Nzombie# Nzombi
# # # # # # # O#O % Nzom# #
# SunFlower # SunFlower # DoubleShooter # MelonPitcher # Nut # # #
# # # # # # # # # # #
# # # # # # #
=====
# # # # # # # Buc# Nzombie
# # # # # # # Nzom
# # # # # # # OO %# # % #
# SunFlower # SunFlower # DoubleShooter # MelonPitcher # # #
# # # # # # # # # # #
# # # # # # #
=====
# # # # # # # Papezom # Nzombie # Papezom
# # # # # # # OO # # Nzomb# Nzombie
# SunFlower # SunFlower # DoubleShooter # MelonPitcher # # # Con# Nz
# # # # # # # # # # #
# # # # # # #
=====

[STORE] SUN:1130 [SCORE]:90

SUNFLOWER:50 PEASHOOTER:100 DOUBLESHOOTER:190
NUT:250 MELONPITCHER:400 PEPPER:600
ICESHROOM:550
=====
```

1.2 游戏操作说明

1. 按 Enter 键开始游戏。*推荐在开始游戏后，右击控制台的标题栏，选择属性，调整字体为 Consolas，以获得更好的效果。*

2. 按空格键进入商店，进店后按 x 键退出商店，wasd 键控制光标的移动，按 Enter 键选中要购买的商品。若要购买的商品的价格大于当前阳光数，则按 enter 键后无法选中商品。

3. 选中商品后，光标自动转到战场上。按 wasd 来挑选安放位置（wasd 键的操作会自动跳过无法安放的单元格），按 Enter 键来安放植物并扣除阳光值。注意，本程序中植物和僵尸不能同处一格，因此植物只可能放置在没有僵尸的单元格中。

4. 在未进入商店或铲子模式的情况下，按 f 键进入铲子模式。进入后，光标自动移动到从上到下，从左到右第一个植物处。按 x 键退出铲子模式，按 f 键跳到下一个植物处。按 Enter 键铲除植物，同时退出铲子模式。

1.3 游戏元素介绍

1.3.1 植物

- 向日葵: 无攻击能力, 仅会在产生阳光的周期到来时带来更多阳光。

- 豌豆射手：周期性地发射豌豆炮弹。豌豆炮弹会对“碰见”的第一个僵尸，以及与该僵尸处于同一单元格且与它的格内进度相同的其它所有僵尸施加攻击。

- 双枪射手：周期性地发射两颗豌豆炮弹。西瓜炮弹会对“碰见”的第一个僵尸所在的单元格的全部僵尸施加攻击。

- 西瓜投手：周期性地发射西瓜炮弹。

- 坚果墙：无攻击能力，血厚

- 火爆辣椒：放置后，在第一个攻击时刻到来时，杀死当前行的所有僵尸，同时自己也随之消失。

- 寒冰菇：放置后，在第一个攻击时刻到来时，冰冻全场所有僵尸，使它们停止攻击植物或移动，同时自己也随之消失。一定时间后僵尸会自动解除冰冻状态，恢复移动或攻击。

1.3.2 僵尸

- 普通僵尸：无特色。

- 路障僵尸：相比普通僵尸，速度稍微快一点点，血量更多。
- 铁桶僵尸：与普通僵尸速度相同，相比路障僵尸血量更多。
- 橄榄球僵尸：相比普通僵尸速度快一些，相比铁桶僵尸血量更多，约为普通僵尸血量的两倍多。
- 读报僵尸：血量略低于铁桶僵尸，一开始速度与普通僵尸相同，血量低于一定值时，移动速度提升为原来的 5 倍，对植物的攻击力提升至原来的 1.6 倍。

二、设计重点

本游戏的设计重点如下：

- 让画面动起来
- 植物发射炮弹，炮弹打击僵尸
- 让僵尸攻击植物
- 炮弹的移动，僵尸的移动
- 僵尸、植物的死亡，炮弹的消失

三、具体设计

3.1 总体设计

由设计重点可知，在本游戏中僵尸、植物、炮弹三者战场上互相交互，它们都可以作为战场上的独立元素存在。因此，应为每一种植物、每一种僵尸、每一种炮弹设立单独的类，分别控制它们的行为。然而，所有的植物、所有的僵尸、所有的炮弹都有各自的共同属性，因此可以为它们各自设立一个总的基类，来把它们各自共同的特征抽象出来；然后让每种单独的植物、僵尸、炮弹继承它们的基类，再在基类的特征的基础上进行修改或添加新的属性。

为了将战场表示出来，可以设立一个二维数组，数组中每个元素都是一个 `vector` 类对象，其中保存了所有处在战场中的这个位置的元素（战场元素包括植物、僵尸、炮弹）。因此这个二维数组便存储了 x 行 y 列的战场中每个位置的所有元素。此时注意到，植物、僵尸、炮弹都是战场上的元素，它们都要被添加到上述 `vector` 对象内，而 `vector` 对象只支持添加一种类型的元素进去，这就引发了矛盾。解决的办法是：使用一个总的基类 `FieldObj` 类来代表所有战场元素，让前面提到的植物基类、僵尸基类、炮弹基类继承 `FieldObj` 基类，然后设 `vector` 对象保存的类型为 `FieldObj*`，也即 `FieldObj` 基类的指针，这样便可以利用 C++ 中指针的多态性来解决这一矛盾。往 `vector` 对象中添加元素时，只需创建相应类型元素的基类（如植物基类）的指针，为该指针分配动态空间，然后将该指针添加到 `vector` 中即可。

为了让画面动起来，可以使用“帧”的机制。首先确定每一秒内

含有多少帧，以此确定帧与帧之间的时间间隔，设为 t 。然后便可以在程序中编写如下的循环语句：首先 `Sleep(t)`，让程序“休眠”时间 t ；然后调用帧处理函数，控制战场上各个元素的交互；最后显示战场。于是每一帧中，战场上各个元素之间的交互便会得到处理，并显示，这样就能让画面动起来。

3.2 设计细节

下面介绍各个主要类的设计。

3.2.1 Field 类

`Field` 类即战场类，负责保存战场信息，控制战场元素的交互，以及显示战场到控制台上。注意，这里提到的战场包括商店。

`Field` 类提供的对外接口只有一个，`loop()`方法。该方法内部实现 3.1 中提到的帧循环。在每一帧中，通过调用私有方法 `frame()`来实现逐帧的控制。`frame()`中做如下几件事：

1. 处理键盘事件：实现商店交互和铲子。
2. 产生阳光：自动地收集阳光。
3. 产生新的僵尸
4. 调用战场上各个已有元素的帧处理函数，并处理不同元素之间的交互。

`frame()`函数处理完后，`loop()`内会调用私有的 `print_all()`方法来将战场（包括商店）打印到控制台上。

`frame()`有 `bool` 类型的返回值，若返回 `true` 则代表游戏结束。

3.2.2 FieldObj 类

该类是战场上所有元素的公共接口。植物基类、僵尸基类、炮弹基类都继承该类。该类中包含公有的枚举类型变量 `type`，表明该对象的类型：是植物 `PLANT`，僵尸 `ZOMBIE` 还是炮弹 `BULLET`。尽管该类不是抽象类（类中没有方法，且仅有一个成员，即上述的 `type`），但考虑到它在程序中的语义和作用，不宜用它直接创建对象。

3.2.3 Plant 类

`Plant` 类是植物基类，继承 `FieldObj` 类。所有植物对应的类都要继承 `Plant` 类。其公有成员包括：

- `planttype`: 枚举类型变量，表明当前植物的类型。该变量应由子类来初始化。

- `virtual bool frame()`方法：为虚函数。该方法为植物的帧处理函数，控制植物在每一帧的行为。由于不同植物可能有不同的行为，因此设为虚函数；由于大部分植物都有相同的行为，因此可以在植物基类中实现 `frame()`方法以涵盖大部分情况，因此不设为纯虚函数。返回值代表是否需要发射炮弹。

- `bool recv_damage()`方法：不是虚函数。该方法实现僵尸吃植物的功能，让植物扣血。返回值代表当前植物是否已被杀死。

保护型（protected）成员包括：

- **int health**: 植物的生命值。该属性应由所有植物共享，因此设为植物基类的保护型成员。该属性应由子类来初始化。

- **int tick**: 帧的响应间隔计数变量。其含义见下。

- **const int tick_cnt**: 帧的响应间隔。每次调用植物的 **frame** 函数，都会让 **tick** 变量自增 1，然后判断 **tick** 与 **tick_cnt** 的大小。若相等，则执行相应的帧处理，并让 **tick = 0**，开始新的循环；否则不做任何帧处理。这样的设计使得不同植物能有不同的帧响应间隔，例如有的植物可以每 5 帧响应一次，有的则可以每 3 帧响应一次。要实现不同的帧响应间隔，只需在子类初始化时给 **tick_cnt** 赋不同的值即可。

3.2.4 Zombie 类

Zombie 类是僵尸基类，继承 **FieldObj** 类。所有僵尸对应的类都要继承 **Zombie** 类。其公有成员包括：

- **zomtype**: 枚举类型变量，表明当前僵尸的类型。该变量应由子类来初始化。

- **int get_progress()**: 获取当前前进进度。前进进度（**progress**）指的是僵尸在当前单元格内的前进进度。若战场为 **x** 行 **y** 列的，则战场上共有 **x * y** 个单元格。而在本程序中，每个僵尸在单元格内可以有前进进度。当前进进度达到最大时，僵尸需要移动到下一格。

- **virtual bool recv_damage(int damage)**: 让僵尸扣血，实现炮弹攻击僵尸的功能。由于读报僵尸的扣血操作与其它大部分僵尸不同

(读报僵尸在血量低于一定值时，会改变攻击力和移动速度)，因此设为虚函数以方便覆盖。返回值代表当前僵尸是否已被杀死。

- `void freeze()`: 让僵尸进入冰冻状态。该函数用于实现寒冰菇这种植物的功能。

- `int get_score()`: 获取杀死该僵尸后得到的分数。每杀死一个僵尸，都要调用该函数，以获得其得分，然后累加到游戏得分中。

- `virtual bool frame()`: 僵尸的帧处理函数。为虚函数但非纯虚函数，原因同植物基类的 `frame()` 函数。返回值代表当前僵尸是否需要移动到下一个单元格。

`protected` 类型成员:

- `const int frozen_tick_cnt`: 被冰冻后的帧响应计数
- `const int max_prog = 14`: 僵尸在格内最多前进的距离，也即最大前进进度。

- `bool frozen`: 是否已被冰冻
- `int progress`: 在一个格内已经前进的距离
- `int tick`: 帧响应计数
- `bool damaging`: 是否正在攻击植物
- `int damage_tick`: 攻击植物的响应间隔
- `int health`: 生命值
- `int damage`: 攻击力
- `int score`: 杀死这个僵尸后，得到的分数

- `int tick_cnt`: 帧响应间隔
- `const int damage_tick_cnt`: 攻击植物时的响应间隔
- `void defreeze()`: 解除冰冻状态。

3.2.5 Bullet 类

Bullet 类是炮弹基类，继承 FieldObj 类。所有炮弹对应的类都要继承 Bullet 类。其公有成员包括：

- `bullettype`: 枚举类型成员，表示炮弹的类型。该变量应由子类来初始化。

- `int get_progress()`: 获取当前在格内的前进进度。同僵尸一样，炮弹在单元格内也有前进进度。

- `int get_damage()`: 获取炮弹的伤害值。

- `virtual bool frame()`: 帧处理函数。返回值代表当前炮弹是否需要移动到下一格。

protected 类型成员：

- `int progress`: 在单元格内前进的进度。
- `bool newly_set`: 炮弹是否刚刚移动到新的一格中
- `const int tick_cnt`: 帧响应间隔
- `int damage`: 炮弹的攻击力
- `int tick`: 帧响应间隔计数

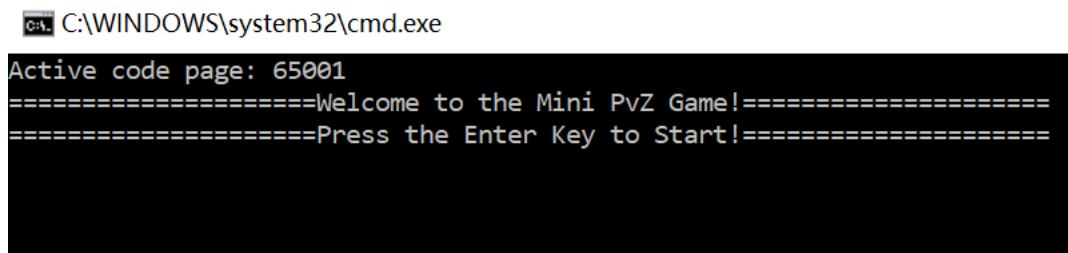
- `const int max_prog`: 炮弹在格内的最大前进进度

以上是各个主要类的介绍。程序中还为每种植物（共 7 种）、每种僵尸（共 5 种）、每种炮弹（共 2 种）设立单独的类，以控制相应的行为。

四、运行流程

4.1 开始游戏

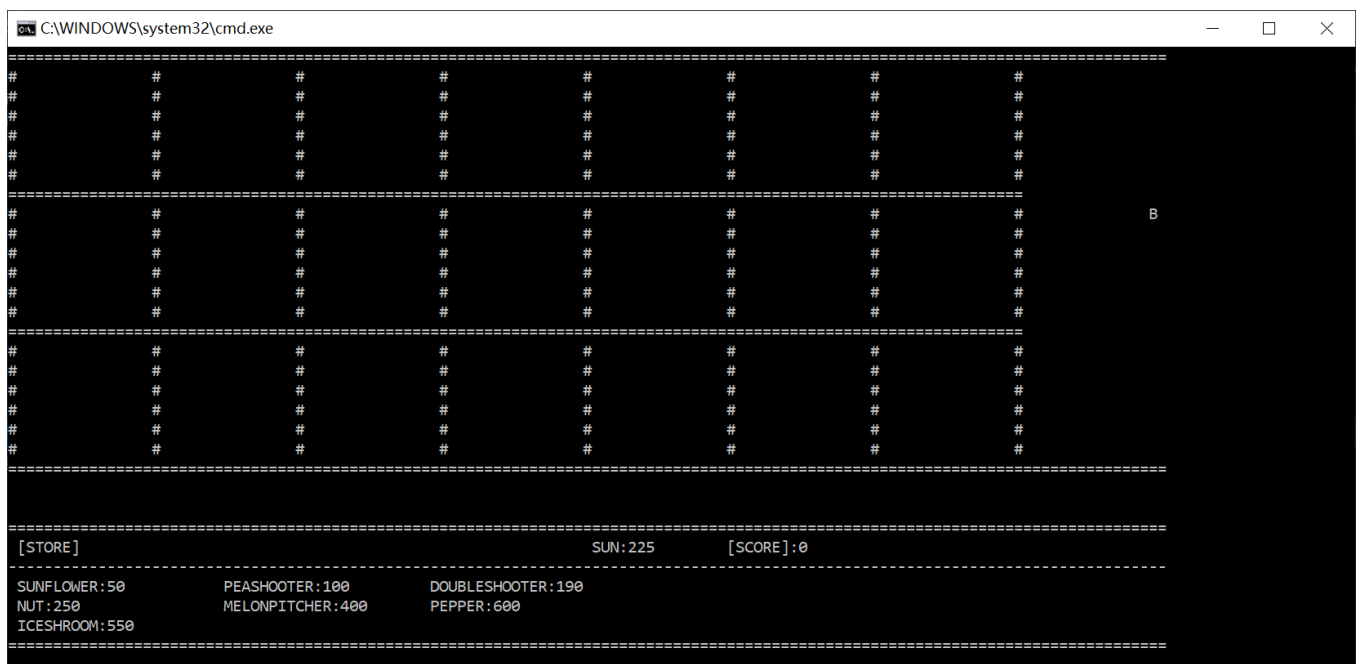
运行程序，首先进入这样的界面：



```
C:\WINDOWS\system32\cmd.exe
Active code page: 65001
====Welcome to the Mini PvZ Game!====
====Press the Enter Key to Start!====
```

按下 `enter` 键后，便可以开始游戏。

4.2 游戏初始界面



```
C:\WINDOWS\system32\cmd.exe
#####
#      #      #      #      #      #      #      #
#      #      #      #      #      #      #      #
#      #      #      #      #      #      #      #
#      #      #      #      #      #      #      #
#      #      #      #      #      #      #      #
#####
#      #      #      #      #      #      #      #
#      #      #      #      #      #      #      #
#      #      #      #      #      #      #      #
#      #      #      #      #      #      #      #
#      #      #      #      #      #      #      #
#####
#####
[STORE]                                SUN: 225    [SCORE]: 0
#####
SUNFLOWER: 50    PEASHOOTER: 100    DOUBLESHOOTER: 190
NUT: 250          MELONPITCHER: 400    PEPPER: 600
ICESHOOTER: 550
#####
```

可以看到，游戏界面中，上部是 $3 * 7$ （可调）的主战场，下部是商店。商店栏中同时也显示出了当前玩家拥有的阳光数和得分。

4.3 游戏流程

下面以处理一帧的过程，来全面阐述游戏的运行流程。

如前所述，在游戏主循环方法中，调用 Field 类中私有的游戏帧处理函数 `frame()` 来实现每一帧的处理。`frame()` 函数执行的具体指令如下：

1. 处理键盘事件

如项目概述中所述，本程序支持进入商店，在商店中选择植物，将植物放置在战场上，使用铲子等键盘操作。在 `frame()` 的一开始，应先处理键盘事件。

2. 自动产生阳光

根据题目要求，本游戏实现了自动收取阳光的功能。若玩家没有种植向日葵，则也会在每隔 `sun_tick_cnt` 个帧后产生阳光；玩家每多种一棵向日葵，每次产生阳光的时机到来时收获的阳光也越多。

3. 产生新的僵尸

每隔 `zom_tick_cnt` 个帧，战场上就会产生一个新的僵尸。由随机数来确定僵尸应出现在哪一行，同样由随机数来确定产生的是何种类

型的僵尸。不同类型的僵尸的产生概率不同。

4. 调用战场上各个已有元素的帧处理函数，并处理不同元素之间的交互

这部分的处理是 `frame()` 函数的内容重点，下面详述。

`frame()` 需要对战场上的每个元素进行处理，因此需要遍历每个单元格，同时遍历单元格内的每个元素，并对它们进行处理。

(1) 确定当前单元格的类型

将单元格分成三种类型：

- 无僵尸：仅有植物和炮弹
- 仅有僵尸
- 僵尸和炮弹共存

对单元格进行分类可以更方便地进行处理。

(2) 根据不同的单元格类型进行处理。遍历当前单元格的各个元素：

1. 无僵尸：仅有植物和炮弹

若当前元素是炮弹，则调用炮弹的帧处理函数 `frame()`，若返回 `true`，则需要将炮弹移动到下一格单元格中，具体的操作就是将当前炮弹指针从当前单元格的元素列表（`vector`）中删除，并添加到下一个单元格的元素列表中。添加之前还需要判断炮弹是否已移动到战场最右侧的单元格，若是则无需添加到新的单元格。

若当前元素是植物，则调用植物的帧处理函数 `frame()`，若返回

`true`，则代表需要发射炮弹。（值得一提的是，大部分植物都是攻击型植物，有发射炮弹的可能性，因此植物基类实现的 `frame()` 用来控制发炮。而坚果墙和向日葵不是攻击型植物，没有发炮的可能性，因此它们重写的 `frame()` 永远返回 `false`。）发炮时，首先判断当前植物是不是火爆辣椒或寒冰菇。若不是，则根据植物类型产生相应类型的炮弹，并将其添加到下个单元格的元素列表中。若是豌豆或双枪射手，则产生豌豆炮弹（`PeashooterBullet` 类）；若是西瓜投手，则产生西瓜炮弹（`MelonBullet` 类）。

若当前植物是火爆辣椒，则在当前单元格中删去该植物，并遍历当前行，删除当前行的各个单元格中的全部僵尸。

若当前植物是寒冰菇，则在当前单元格中删去该植物，并遍历整个战场，调用各个单元格中的每个僵尸元素（指针）的 `freeze()` 方法，设置所有僵尸为 `frozen` 状态。进入 `frozen` 状态的僵尸在执行自己的 `frame()` 帧处理时不做任何事，仅仅把自己的 `tick` 计数值+1，以此实现僵尸不移动、不吃植物的效果。当 `tick` 值达到常量 `frozen_tick_cnt` 时，僵尸的 `frame()` 调用僵尸类的 `defreeze()` 方法，为自己解冻，解冻之后便可以开始正常的帧处理，僵尸开始正常地移动或吃植物。

2. 仅有僵尸

先调用当前僵尸的帧处理函数。若返回 `true`，则代表这个僵尸应向前移动一格。此时先判断僵尸当前所在的格子是否是底线之前的格子，若是，则意味着僵尸将突破底线，于是帧处理函数（`Field` 类的）

直接返回 `true`，表明游戏结束。若不是，则删去当前格的僵尸，将这个僵尸添加到下一个格子中。

若返回 `false`，则代表僵尸正在格内移动，或者正在吃植物。吃植物和格内移动（让僵尸的 `progress` 值加 1）的操作由僵尸的帧处理函数完成，植物被吃掉后会被从相应单元格中删去。

3. 僵尸和炮弹共存

这是最复杂的情况。首先对当前单元格中全部元素（仅含有炮弹和僵尸）进行排序。按照如下规则进行排序：炮弹排在僵尸前面；僵尸当中，格内进度大者排在前面，小者排在后面；炮弹当中，格内进度大者排在前面，小者排在后面。

然后，依次遍历各个元素。若是炮弹，则调用其 `frame()` 帧处理函数，该函数让炮弹在格内向前移动。由于炮弹和僵尸同时存在于一格中，故可以肯定炮弹和僵尸都不会出现需要移动到下一格的情况。然后监测炮弹是否会与僵尸发生碰撞：若没发生则结束，发生了碰撞，则删去该炮弹。如果炮弹是豌豆炮弹，则将该炮弹的伤害值作用于该格中 **每一个**跟与炮弹发生碰撞的那个僵尸处于相同的格内进度的僵尸上；若是西瓜炮弹，则作用于该单元格内全部僵尸上（模拟溅射效果）。若僵尸被炮弹打死了，还需要删去该僵尸。

若是僵尸，则直接调用其 `frame()`，让其向前移动 1 个格内单位即可。由于排序后一定是先处理炮弹再处理僵尸，因此到了处理僵尸的时候，无需考虑僵尸与炮弹的互动，因为这已在处理炮弹时被考虑

了。

五、遇到的问题和解决方案

1. 控制台闪屏问题

Windows 的控制台中，若不断使用清屏函数，再打印新的内容，则会出现内容一闪一闪的情况。为了解决这个问题，根据网络上搜索到的方法，我使用了控制台缓存。在需要更新屏幕内容时，直接往控制台缓存里写入内容，然后使用相应的系统 API 函数来将缓存中的内容显示在控制台中，这样就避免了闪屏问题。

2. 如何将各种战场元素添加到同一个 `vector` 中：设立总的基类 `FieldObj` 类，并用其指针类型作为 `vector` 的类型，让植物、僵尸、炮弹基类都继承 `FieldObj` 类，这样就可以利用 C++ 的指针多态性来解决该问题。