高级程序设计

植物大战僵尸(控制台版)

版　　本	Plant Vs Zombies(v1.0)
院　　系	计算机科学技术系
学　　号	191220027
学生姓名	樊润璞
学　　号	191220027
时　　间	2020.9.21 - 2020.10.18

目录

1 概述部分

1.1 主要内容

以老少皆宜的植物大战僵尸为基础参考，实现了运行在 Windows 控制台下的一个键盘操控版植物大战僵尸游戏。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（相当于原版游戏中的无尽模式），击杀僵尸，不断获得更高的分数。在游戏中，只要任何一个僵尸到达地图的左边界，则判为游戏结束（参考原植物大战僵尸的设计）。玩家可使用数字键（目前只有 1,2）选择要种植植物；方向键控制格子选择框的移动；回车键确定格子的选择、Esc 键返回

1.2 已实现目标

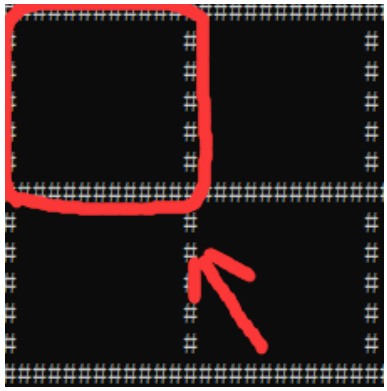
1. 2 种植物：向日葵、豌豆射手（可以发射豌豆）
2. 1 种僵尸：普通僵尸
3. UI 界面：当前局计分碑，当前局阳光数，当前局子弹显示（豌豆射手的豌豆），当前局商店界面打印，当前局地图刷新，植物颜色...
4. 计分规则：击杀一个僵尸获得 20 分（不设上限）
5. 僵尸刷新：僵尸随机刷新（暂时不支持一个格子出现多个僵尸）
6. 僵尸与植物的对战：僵尸可以不断前进，豌豆歙州同各国子弹对僵尸造成伤害

2 类的划分以及模块的设计

2.1 类的划分

2.1.1 Cell 类

此处的 Cell 类代表的是游戏地图中的一个格子。



其成员包括：

a) 植物（至多种植一株植物）

b) 僵尸 (至多有一个僵尸在格子里面)

可进行的操作为：是否被选中，是否种植植物，是否添加僵尸，是否删除植物...

以下是 Cell.h 的展示

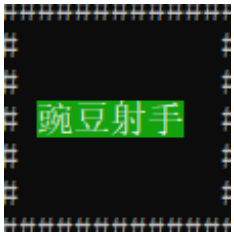
```
class Cell{
  //格子左上角的坐标位置
  int up_left_x, up_left_y;
  //是否被选中
  bool isSelected ;
  //是否已经种植植物
  bool isPlanted ;
  //种植的植物类型
  int plantType;
  //是否有僵尸
  bool haveZombie;
  //用于判断僵尸是否停止（true为停止)
  bool zombie_status;

public :
```

```
16          Plant* hereplant;
17          Zombie* herezombie;
18          Cell();
19          //设置坐标
20          void set_position(int x, int y);
21          //添加一个僵尸(添加成功返回true)
22          bool addZombie(int tmp_health);
23          //删除一个僵尸(删除成功返回true)
24          bool delZombie();
25          //种植一个植物 (种植成功返回true)
26          void plant_a_plant(PLANT_NAME plant,Game& nowgame);
27          //铲除一个植物 (铲除成功返回true)
28          bool del_a_plant();
29          //绘制被选中后界面图案
30          void draw_focus();
31          void select() { isSelected = true;}
32          void unselect() { isSelected = false;}
33          bool getSelect() {return isSelected;}
34          void draw_zombie();
35          void clear_zombie();
36          bool Check_zombie() {return haveZombie;}
37          void checkeating();
38          void zombie_init();
39          //获取僵尸状态 (停止为true)
40          bool get_zombie_status() {return zombie_status;}
41          void init_zombie_status() { zombie_status = false; }
42          void eating();
43
44          //植物打僵尸时候调用
45          int Getzombie_health() {return herezombie->gethealth();}
46          ~Cell();
```

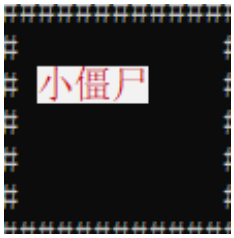
```
47 };
```

根据格子内的内容，有多种显示：

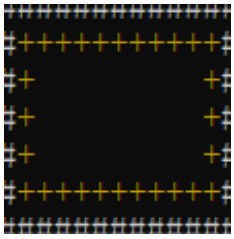
1、只包含植物：



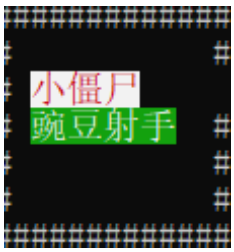
2、只包含僵尸：



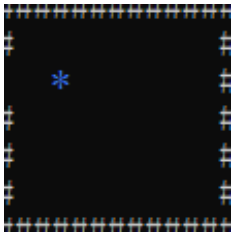
3、被选中状态：



4、同时包含植物和僵尸：

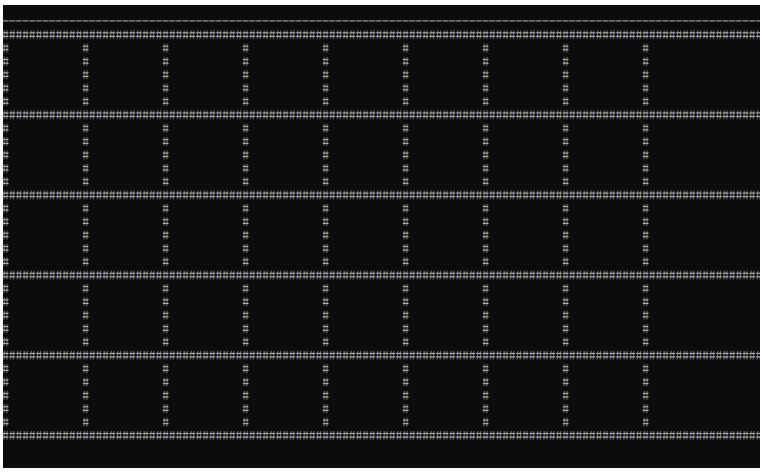


同时在格子内也可以显示豌豆射出的子弹



2.1.2 Map 类

Map 类用于模拟植物大战僵尸中的草坪 (边界线由 # 来划分)



其主要的操作有界面刷新，地图初始化，绘制地图。

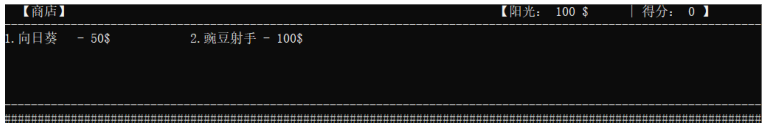
以下是 Map.h 的展示

```
1   class Map{
2  public :
3           Cell grass_block[MAP_NUM_Y][MAP_NUM_X + 1];
4           Map();
5           void init ();
6           //地图刷新
7           void refresh ();
8           ~Map();
9           //friend class Cell;
```

```
10          friend  class  Game;
11  };
```

2.1.3 Store 类

Store 类用于模拟植物大战僵尸中的植物商店，用于选择和购买植物，植物



注：此处的 Store 类中的植物卡片单独抽象出一个类 PlantSet 用于单独控制每个卡片的输出显示（为了后面单独实现商店更加丰富的显示效果而单独抽象出来）

Store 类的具体实现如下：

```
1   class  Store{
2       //当前阳光数
3       int  sun_num;
4       //当前得分
5       int  score;
6
7   public:
8       PlantSet  plant[PLANT_TYPE];
9
10      Store();
11      //商店界面初始化
12      void  init();
13
14      int  getSun() {
15              return  sun_num;
16      }
```

```
17          void  refresh () {
18                  paint_store_ui ();
19          }
20          int  getCost(PLANT_NAME plant_name);
21          void  run() {
22                  for (int  i = 0;i < PLANT_TYPE;++i) {
23                          plant[i].Cooling();
24                  }
25          }
26
27          //商店ui界面打印
28          void  paint_store_ui ();
29          //增加阳光数
30          void  add_sun(int isun_num);
31          ~Store();
32          int  getscore() {
33                  return  score;
34          }
35          void  change_score(int  num) {
36                  score = num;
37          }
38          friend  class  Game;
39  };
```

PlantSet 类的具体实现如下：

```
1       //植物集合
2   class  PlantSet {
3           //冷却时间
4           int  cooling_time;
5           //花费阳光数
6           int  cost;
7           //植物编号
```

```
8           int num;
9           //植物名称
10          string name;
11          //是否被选中
12          bool isSelected ;
13
14          int counter;//时间计数
15
16          bool status ;//(是否处于冷却中)
17          // friend class Game;
18  public :
19          PlantSet() {
20                  isSelected = false ;
21                  status = false ;
22                  counter = 0;
23          }
24          //设置植物卡片的初始参数
25          void set_plantcard ( int inum, const string & iname, int icooling_time , int icost );
26          //打印植物卡片
27          void print_card ();
28          //选中植物卡片
29          void selected () {
30                  isSelected = true;
31          }
32          //未选中植物卡片
33          void unselected () {
34                  isSelected = false ;
35          }
36          //判断是否选中植物卡片
37          bool JudgeSelected() {
38                  return isSelected ;
```

```
39          }
40          //获取植物的花费
41          int getCost() {
42                  return cost;
43          }
44          //判断植物是否处于冷却状态
45          bool Cooling() {
46                  if (status == true) {
47                          counter += 10;
48                          if (counter >= cooling_time) {
49                                  status = false;
50                                  counter = 0;
51                          }
52                          return false;
53                  }
54                  else
55                          return false;
56          }
57          //判断是否可以种植植物
58          bool canPlant() {
59                  if (status == false) {
60                          status = true;
61                          return true;
62                  }
63                  else
64                          return false;
65          }
66          ~PlantSet(){}
67  };
```

2.1.4 Bullet 类

该类用于设定子弹的一些具体参数，如位置，所在行号等以及子弹的具体绘制等。

但是子弹的产生于移动并未放在该类中，而是放在了游戏的运行类 Game 中实现，这样更便于管理屏幕中子弹的显示以及移动

子弹类的具体实现如下：

```
1   class Bullet {
2       int speed;
3       int counter;//时间计数
4       int x, y;//当前所在位置
5       int map_y;//当前所在地图的行号
6   public :
7       Bullet() {
8           speed = 5;
9           counter = 0;
10          x = y = 0;
11      }
12      //绘制子弹
13      void draw_bullet(int dx, int dy) {
14          setXY(dx, dy);
15          setColor(BULLET_COLOR);
16          cout << "*";
17          setColor(NORMAL_COLOR);
18      }
19      //清除子弹
20      void clear_bullet(int dx, int dy) {
21          setXY(dx, dy);
22          cout << " ";
23      }
24      //改变子弹的参数
25      void changeXY(int dx, int dy) {
```

```
26                  x = dx;
27                  y = dy;
28          }
29          //获取子弹的x位置
30          int getX() {
31                  return x;
32          }
33          //获取子弹的y位置
34          int getY() {
35                  return y;
36          }
37          //设置子弹所在的行
38          void set_row(int irow) {
39                  map_y = irow;
40          }
41          //获取子弹所在的行
42          int get_row() {
43                  return map_y;
44          }
45  };
```

2.1.5 Game 类

Game 类主要用来控制游戏的运行状态，状态的切换，特定函数的调用等等，其用到的类主要为 Stroe 类和 Map 类，负责控制各个部分的运行

Game 类的成员函数以及 Game 类的定义如下：

```
1   class Game {
2       //当前游戏状态
3       int status;
4       //种植植物的标号
5       int plant_type;
```

```
6           //阳光自动生产时间计数
7           int count;
8           //是否购买结束
9           bool shopping_end;
10          //地图选择开关位置
11          int select_x, select_y;
12          //游戏结束
13          bool end;
14
15          int counter_zombie;//僵尸产生计数
16          int move_counter;//僵尸移动计数
17
18          int count_kill_zombie;//杀死僵尸数
19
20          vector<zombies> all_zombie;
21          //friend class Cell;
22  public:
23          Map map;
24          Store store;
25          vector<BULLET> bullet;
26
27          Game();
28          //游戏初始化
29          void init();
30          //游戏主循环
31          void loop();
32          //购买并种植植物
33          void shopping();
34          //铲除植物
35          void shoveling();
36          //暂停
```

```
37          void pausing(int tmp_status);
38          //界面刷新
39          void refresh();
40          //游戏正常运行
41          void playing();
42          //开启地图焦点
43          void openfocus(int& x, int& y);
44          //关闭地图焦点
45          void closefocus(int& x, int& y);
46          ~Game();
47          //阳光自增
48          void sun_increase();
49          //游戏结束
50          void gameover();
51          //产生僵尸
52          void makezombie();
53          //僵尸移动
54          void movezombie();
55          //清除僵尸
56  void delzombie();
57  //得分函数运行
58          void score_run() {
59                  int new_score = count_kill_zombie * 20;
60                  this->store.change_score(new_score);
61          }
62          //添加子弹显示
63          void addbullet(int x, int y);
64          //移动子弹
65          void movebullet();
66          //清除子弹
67          void clearbullet();
```

```
68          friend  class  Plant;
69  };
```

2.1.6 僵尸以及植物类

最后，将是以及植物类主要为实现各个僵尸或者植物的主属性，

主要为：1、僵尸：生命值，攻击力，移动速度, 所在格子参数，自身状态

2、植物：生命值，攻击力，射击速度（产生阳光的速度），所在格子参数，自身状态

值得一提的是，由于实现了两个植物，当运行的各个植物的功能，如果根据植物的类型而调用不同的函数，会略显繁杂，debug 时也有很大的困难，因此这里采用了动态绑定，及设计一个基类 Plant, 而其派生出不同植物对应的子类类型（目前只有 Sunflower 和 Peashooter），在调用植物功能时，直接调用基类的 go 函数，从而就可以使得地图上对应植物的功能触发变得更加简洁，也为 debug 减轻了不少负担，如图2.1.6

```
virtual void go(Game& nowgame) {}
```

以下分别是向日葵和豌豆射手具体 go 函数的实现：

```
void Sunflower::go(Game& nowgame) {
    counter += 10;
    if (counter >= produce_time) {
        nowgame.store.add_sun(25);
        //setXY(10, 5);
        //cout << "produce";
        counter = 0;
    }
}
```

图 1: 向日葵

```
void Peashooter::go(Game& nowgame) {
    counter += 10;
    if (counter >= attack_time) {
        nowgame.addbullet(up_left_x, up_left_y);
        counter = 0;
    }
}
```

图 2: 豌豆射手

具体调用场景：

```
//植物功能触发
//cout << "going" << endl;
for (int i = 0;i < MAP_NUM_X;++i) {
    for (int j = 0;j < MAP_NUM_Y;++j) {
        if(map.grass_block[j][i].hereplant != nullptr)
            map.grass_block[j][i].hereplant->go(*this);
    }
}
```

图 3: 具体调用场景

3 代码中的创新点

3.1 界面刷新机制

为了不断刷新界面显示，此处采用 Sleep 函数，每隔 100ms 对界面进行一次刷新（可以视为游戏运行了一个很小的回合）。而游戏的主题放在主循环 loop 中，当游戏判断标志 end 变为 true 时，跳出循环，游戏结束

3.2 界面 UI 控制

此处的界面控制专门抽象出了一个头文件 windows_tools.h 用于进行各种颜色的设定，定位，窗口的绘制，一些位置宏的预定义，方便在绘制 UI 的时候进行参考，也方便整体 UI 界面的修改与再设计

3.3 僵尸与植物对战环节

此环节由于采用子弹作为攻击方式，因此存在子弹发射、消亡的问题，此处在 game 类中单独对这个部分进行多层控制实现了未出现僵尸时，豌豆射手不发射子弹，直到出现僵尸才发射子弹的效果。（与原版更为接近）

4 一份简单亟待扩充的用户手册

进入游戏后，每隔 2s 产生一次僵尸 1）植物的购买与种植：按下 b 键进入购买状态，再按下 1 或 2 进行具体植物的选择，在选择植物之后通过上下左右方向键控制具体种植地块的选择（一定要在选择植物后进行!!!），之后按下回车键确认选择，种植植物，种植植物后，再按 esc 退出植物购买，返回正常的游戏状态 2）停止：待完善 3）其他：... 待完善（目前只实现了购买与种植，有待扩充）具体运行截图：

图 4: 截图一

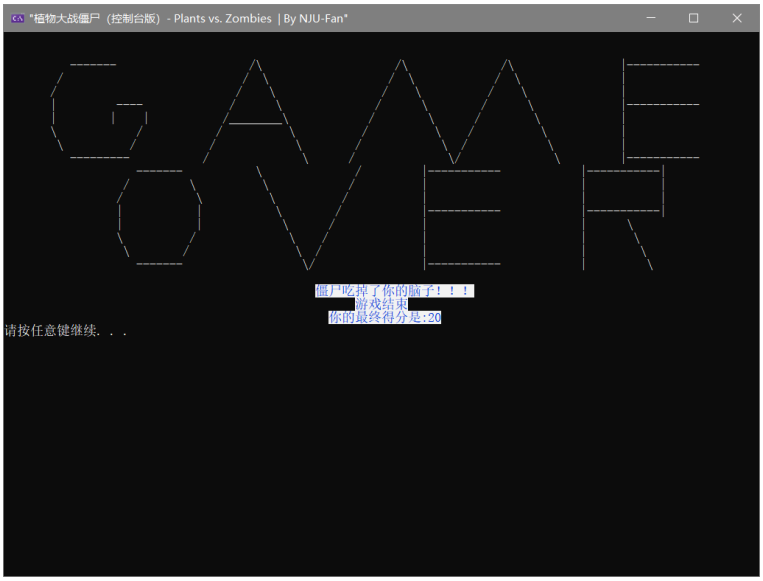


图 5: 截图二

5 结束

全文结束，thanks!!!