第三次课程设计报告

植物大战僵尸③

时间： 5月11日~6 月 8 日

姓名：张玄逸　　学号：201220194

目录

一、概述

A.主要内容

仿照植物大战僵尸，制作一个基于图形界面的简易版游戏。玩家需要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

模式：无尽模式（除非僵尸越过左侧防线，否则游戏不会停止）

①白天

1.纯草坪无水池迷雾，有阳光掉落

2.植物选择通过有冷却时间的植物栏实现

3.僵尸数目有上限

4.低难度

②夜晚

1.无阳光掉落

2.植物选择通过随机的传送带实现

3.僵尸数目无上限

4.高难度

植物 14 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，火炬树桩，高坚果，南瓜头

僵尸 7 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，铁栅门僵尸

C.设计思路

1.植物卡用 Card 类表示，不再需要 store 类，并增加冷却效果。

2.因为图形界面不需要考虑键盘操作，所以去掉 game 类，增加 QMaindialog 类。

3.去除了文字的输出，僵尸运动由一格一格地跳跃变成连续移动，故 Grid 也失去了作用。

4.添加开始界面和选择界面。

5.为所有的可视类增加动图，并且添加背景音乐和特定音效。

6.增加 Sun 类，实现阳光的移动与点击。

7.增加小推车。

二、主要类的设计

A.类的结构与操作

①Object 类

```
class Object: public QLabel
{
    Q_OBJECT
    friend class Lawn;
    friend class Day_lawn;
    friend class Night_lawn;
protected:
    bool alive;
    Lawn* lawn;
public:
    Object(QWidget* parent = 0);
    virtual void action() {}
    int get_x();
    int get_y();//计算当前在哪一个草块
};
```

```
Object::Object(QWidget* parent): QLabel(parent)
{
    alive = 1;
    setMouseTracking(true);
    lawn = (Lawn*)parent;
}
```

为了显示图片和标记是否死亡，我创建了 class Object，继承自 QLabel。Object 是程序中所有可视对象的基类，初始化时 setmousetracking(true)，以追踪鼠标，并且设 alive 为 1，当后续运行时若为 0 则被清除，lawn 变量可以使子类操纵基类。

②Maindialog 类

```
class Maindialog: public QDialog
{
    Q_OBJECT
public:
    Maindialog(QWidget* parent = 0);
    ~Maindialog();
public slots:
    void turn_to_day();//跳转到普通模式
    void turn_to_night();//跳转到挑战模式
    void turn_back();//跳转到选择界面
private:
    Ui::Dialog* ui;
    QMovie* background = new QMovie(":/else/mainlogo.png");
    Lawn* lawn;
    Start_screen* start_screen;
    Choose_screen* choose_screen;
};
```

继承自 QDialog 类，用于显示主窗口。

成员变量：包含一个 choose_screen（选择界面），一个 start_screen（开始界面），一个 Lawn（草坪）。负责接收界面和草坪的信号，对应跳转到相应的界面或草坪。

③Lawn 类

day.jpg

night.jpg

Lawn 是一个基类，有 Day_lawn 和 Night_lawn 两个子类。Day_lawn 是普通模式，Night_lawn 是挑战模式。Lawn 类包含几乎所有的可视对象。

普通模式的场景是白天，玩家收集阳光并用于购买植物，植物有冷却时间。僵尸的出现有较大间隔，类似于普通的关卡。当僵尸进入第 15 波之后，僵尸出现机制与挑战模式类似，从 6 开始上涨，15 封顶。普通模式的植物栏因为只有 10 个空位，所以是随机刷出 10 种，但一定有向日葵。

挑战模式的场景是夜晚，没有阳光，植物在传送带上随机出现，没有冷却时间，不花费阳光，玩家可以直接种植。僵尸出场间隔较小且相对固定，每一波僵尸的数目随着时间的增长而增长，无上限（也就是说玩家和我的电脑总会挂一个）。

```
Plant* plant[whole_length][whole_width]; //植物列表
Pumpkin* pumpkin[whole_length][whole_width];//南瓜头
Object* weed[whole_width]; //小推车列表
QList<Zombie*>zombie_list[whole_width];  //僵尸列表
QList<Bullet*>bullet_list;  //子弹列表
QList<Sun*>sun_list;  //阳光列表
QList<Special*>special_list;//特殊效果列表
QList<Card*>card_list;  //植物卡列表
```

这是草坪中主要的成员变量。每隔 0.02 秒刷新一次，遍历所有列表，执行 action 函数，若该 Object 的 alive==0，表明已死亡或应当消失，故直接删除。行动后应当使用 raise()函数，使该对象提升到最表面。

```
for (int i = 0; i < bullet_list.size();) {
    if (bullet_list[i]->alive == 0) {
        delete bullet_list[i];
        bullet_list.removeAt(i);
    } else {
        bullet_list[i]->action();
        bullet_list[i]->raise();
        i++;
    }
}
```

另外，草坪对列表的遍历有一定的顺序。因为植物，僵尸，阳光等经常会有遮挡现象，需要使遮挡效果符合逻辑，比如不能让植物遮挡了阳光，移动的铲子一定在最上面。先是按行分别存储的元素，遍历顺序为从 0 行到 4 行，依次是植物，南瓜，僵尸，小推车，再是没有分行的元素，遍历顺序为子弹，特殊效果，铲子。

```
if (type == day) make_sun();
else make_card();
```

```
for (int i = 0; i < card_list.size();) {
    if (type == day) card_list[i]->action();//普通模式冷却卡片
    else {
        Card* now = card_list[i];
        if (now->alive == 0) {
            delete now;
            card_list.removeAt(i);
        } else {
            int x = now->pos().x();
            if (i == 0) {
                if (x >= 25 + grass_left) now->x -= 5;
            } else {
                if (x >= card_list[i - 1]->pos().x() +
                    card_length + card_gap) now->x -= 5;
            }
            now->move(now->x, now->y);
            i++;
        }
    }//挑战模式移动卡片
}
```

特别地，普通模式在循环中需要制造阳光，而挑战模式需要制造植物卡并移动现有的所有卡片。因为阳光在挑战模式中无用，故产生的卡片没有向日葵，所以一共可产生 13 种植物。

④Card 类


```
class Card: public Object
{
    friend class Lawn;
    friend class Day_lawn;
    friend class Night_lawn;
protected:
    int x;
    int y;//坐标
    Plant_type plant_type;//植物种类
    Object* background = new Object(this);//图片
    int cost_sun;//价格
    int cooling;//当前已冷却时间
    int cool_time;//冷却时间
    QPalette* palette = new QPalette;
    Object* front = new Object(this);//未冷却的覆盖部分
public:
    Card(QWidget* parent = 0);
    void set_pos(int x);//放置到植物栏的对应位置
    void mousePressEvent(QMouseEvent* event);
    //重写鼠标点击函数
    bool is_card();//判断是否是真正的植物卡
    void action();
    ~Card();
```

Card 类主要用于展示植物卡，进行冷却和购买操作。

```
Shovel* shovel = new Shovel(this);//铲子
Object* shovel_back = new Object(this); //铲子框
Card* leave = new Card(this); //菜单键
Object* leave_text = new Object(leave);
Card* menu = new Card(this); //菜单
Card* exit = new Card(this); //返回选择界面键
Object* exit_text = new Object(exit);
Card* back = new Card(this); //返回游戏键
Object* back_text = new Object(back);
```

这里是 Lawn 中几个比较特殊的 Card 类。Card 类本来是专门用于植物卡，但因为各种按键也要实现点击反应，单独抽出一个类再重写 mousePressEvent 函数不太现实，所以合并入 Card 类，plant_type 标记为 shovel,leave,back,menu,rubbish 等，以便判断到底执行的是选择植物还是按键操作。

```
void Card::mousePressEvent(QMouseEvent* event)
{
    if (lawn->state == Lawn_state::over) return;
    if (event->button() != Qt::LeftButton || lawn->start_time) return;

    if (plant_type == menu) { ...}

    if (plant_type == back) { ...}

    if (plant_type == leave) { ...}

    if (lawn->state == pause) return;

    if (plant_type == rubbish) return;

    if (lawn->now_plant == NULL) { ...}
    } else { ...}
}
```

⑤Plant 类

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

植物状态的改变其实就对应着大小的改变以及图片的替换。

关于每种植物的成员和具体功能在课设二中已经阐述过，这里只放图。

1.豌豆射手(peashooter)

2.向日葵(sunflower)

3.樱桃炸弹(cherry_bomb)

4.坚果墙(wall_nut)



5.土豆雷(potato_mine)



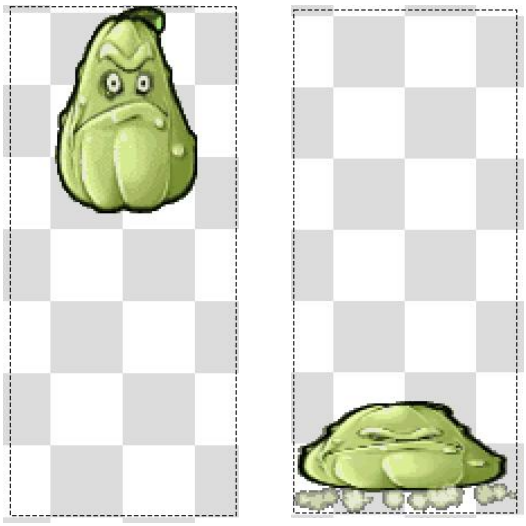
6.寒冰射手(snow_pea)

7.食人花(chomper)



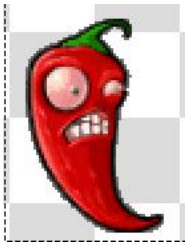
8.双发射手(repeater)

9.窝瓜(squash)

10.三线射手(threepeater)

11.火爆辣椒(jalapeno)



12.火炬树桩(torchwood)

13.高坚果(tall_nut)



14.南瓜头(pumpkin)

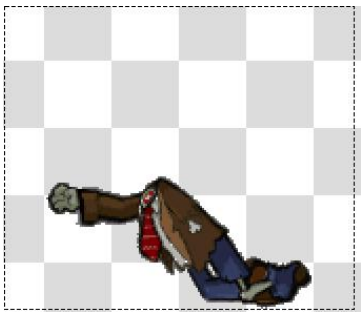



⑥Zombie 类

与 plant 类相同，只放图。

1.普通僵尸(ordinary)

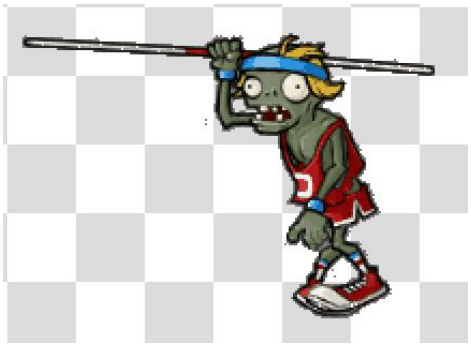


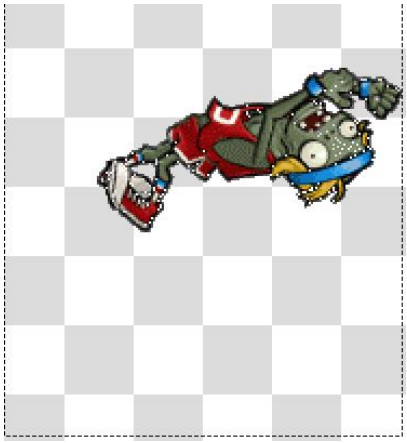



2.摇旗僵尸(flag)

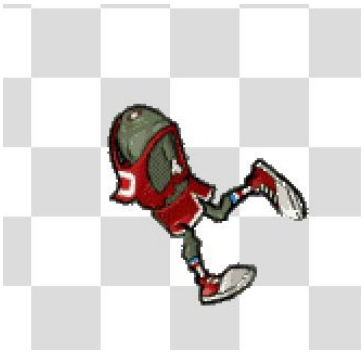

3.路障僵尸(conehead)


4.撑杆僵尸(pole_vaulting)










5.铁桶僵尸(bucket)


6.读报僵尸(newspaper)

7.铁栅门僵尸(screen_door)

⑦Special 类

```
class Special: public Object
{
    friend class Cherry_bomb;
    friend class Potato_mine;
private:
    int life_time;
public:
    Special(int x, QWidget* parent = 0);
    ~Special();
    void action();
};//用于播放动图
```

```
void Special::action()
{
    if (movie()->currentFrameNumber() < movie()->frameCount() - 1) return;
    movie()->stop();
    life_time--;
    //    if (movie() == NULL) return;
    if (life_time == 0) alive = 0;
}
```

Special 类只有一个功能，就是在特定位置播放动图，播放完成后被清除。原本樱桃炸弹爆炸后的烟雾动画我是打算直接用樱桃炸弹本身播放，也就是把原来的图片换一张。但很快发现这样不太合理，因为爆炸就是它生命的终结，理论上在爆炸后但烟雾还未散去的这段时间（虽然只有 1 秒），格子上可以种植其它植物。但如果单纯替换图片，草坪会认为这里仍然有植物，于是拒绝了种植的请求。这种情况下就需要创建一个 Special 来播放烟雾，避免草坪作出不合理的判定。土豆雷爆炸，僵尸灰烬同理。

action 函数的逻辑是，当目前动图帧数是最后一张，即播放完毕后，开始倒数 life_time，为 0 后清除该对象。life_time 以 0.02 秒（草坪循环间隔）为单位，表明图片播放到最后还需要在草坪上停留多久。

B.类之间的关系

以成员变量建立的包含关系已经在上面的截图中清晰地展示出，下面是继承关系。

使用 QLabel 作为基类，可以方便地实现对鼠标的追踪，重写鼠标点击函数，播放动图，移动和放缩，显示文字等功能。

- 红色为QT内置类
- QDialog
  - Maindialog
- QLabel
  - Object
    - Sun
    - Frame（植物栏）
    - Card（植物卡）
      - sunflower_card
      - peashooter_card
      - ...
      - shovel
      - leave
      - menu（菜单）
      - back（菜单中的返回键）
      - exit（菜单中的退出键）
    - Plant
      - sunflower
      - peashooter
      - ...
    - Zombie
      - ordinary
      - conehead
      - ...
    - Bullet
      - Pea_bullet
      - Snow_bullet
      - Fire_bullet
    - Special（特殊效果）

三、程序运行

A.运行操作方法

我觉得不用写了，就是正常的操作方法。

B.功能亮点

①特殊效果

樱桃炸弹的爆炸效果

土豆雷的爆炸效果

火炬树桩的作用

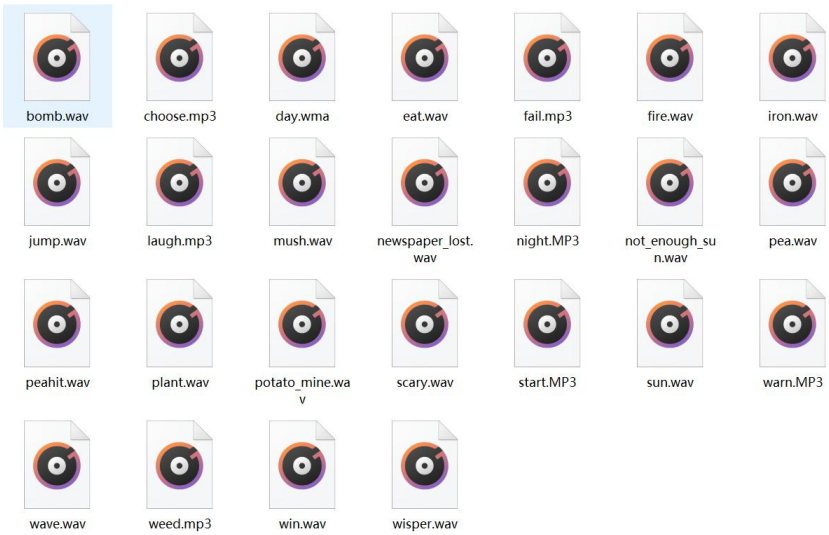
②僵尸死亡

使用 Zombie 类的虚函数 die 实现，当射手类植物造成僵尸减血时，判断僵尸血量是否减为 0，若是，则由该植物调用僵尸的 die 函数，创建一个 Special 对象，用于展示僵尸死亡状态。

僵尸有三种死亡类型，除读报僵尸和撑杆僵尸有自己专有的死亡图像之外，其它的都用的是普通僵尸死亡的图像。具体的图像在上面已经展示过，死的过程很快我实在截不了图了。

③背景音乐与音效

我使用了众多音乐与音效来模拟真实的游戏场景。



音乐：
开始加载界面
普通模式
挑战模式

音效：
樱桃炸弹与土豆雷爆炸
射手发射豌豆
普通豌豆和冰豌豆的打击
火豌豆的打击
打击到钢铁制品的声音

僵尸啃食植物
撑杆僵尸的弹跳
读报僵尸的愤怒

植物的种植和铲除
阳光不足或未冷却完成的提示音
阳光的收集
小推车开动

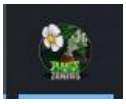
僵尸最开始出现时的警报声
失败的音乐

失败的喊叫

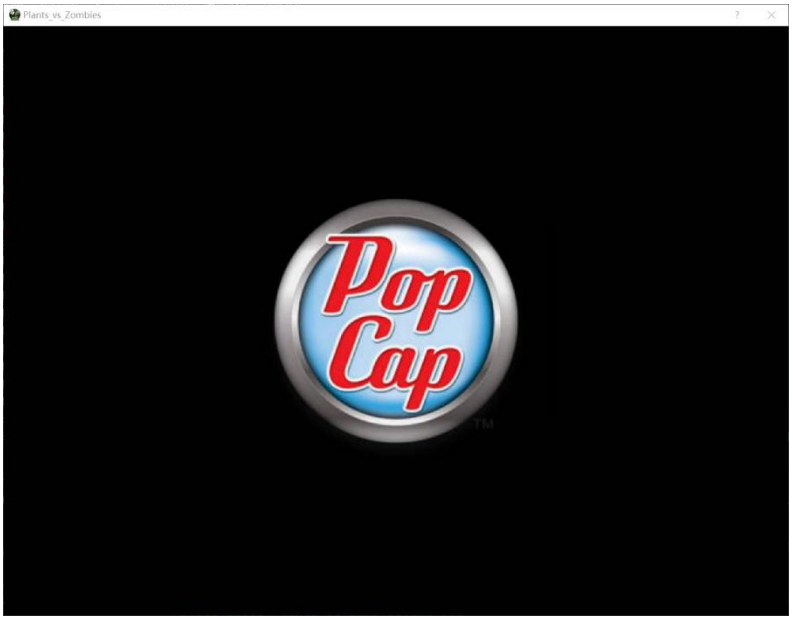
进入某一模式时的僵尸笑声

④场景模拟

图标



加载界面



开始界面

选择界面


Plants_vs_Zombies
普通模式
挑战模式

普通模式


Plants_vs_Zombies
5000
100
50
150
50
25
175
150
200
125
125
菜单
准备...

挑战模式

菜单

失败界面

⑤细节

1.南瓜头的不同表现方式

南瓜头在里面没有植物时，是整个的形状

有植物时换为另一个只有前面的图像

（没有后面那一块，不然对植物形成遮挡，不像是被包含在内部）

2.准备和失败阶段的静止

直到“准备-开始-种植植物”开始的动画播放完毕之前，植物卡不会冷却，鼠标点击不生效，也不能返回，失败画面同理，与原游戏一致。

3.阳光不足的提示

当普通模式中阳光不足以购买时，植物栏的阳光数字会在红黑两色之间交替闪烁。

4.阳光的不同状态

当下落到草坪上后，如果一段时间后玩家没有点击，则阳光会闪烁几次以示提醒，仍然未被点击则会消失。若被点击则不再闪烁，飞到左上角阳光数的位置，接近时透明度逐渐升高，最后消失，同时阳光数增加。

5.暂停前后

点击菜单键后会有提示音，并且背景音乐暂停播放，直到玩家选择返回游戏才会继续播放。

四、遇到的问题与解决方案

大都是血泪教训

1. 类创建时默认 MouseTracking=0，并且改变后 QMainWindow 类还需要加上 ui->centralWidget->setMouseTracking=true

2.QLabel 创建的子类不能移动到父类范围之外

3.如果重写了子类的 mousePressEvent 或 mouseMoveevent 函数，则该函数不会传给父类，除非在程序中显式调用父类该函数。

4.QLabel 必须使用 resize 和 move(setGeometry)以及 show 和 raise 才能正确显示

5.只有在 class 声明中加入了 Q_OBJECT 才能使用信号与槽函数。

6.继承自 QLabel 的 QLabel 的 mousePressEvent 会失效。（还不清楚为什么）

7.QMediaplayer 的 setmedia 只能设置绝对路径。（也不清楚相对路径怎么设，这一点会导致该项目在其它电脑上虽然能正常运行，但无法播放三首背景音乐和部分音效，但所有.wav 音乐不受影响，因为是用的 QSound 播放）

8.用 new 创建的继承自 QObject 的 QObject 不需要再自己析构，父类析构时会自动调用子类的

析构函数。

9.分清楚横纵坐标！！！

五、总结与反思

这一次我自认为比较好地应用了继承与多态等特性，比较遗憾的事情就是找不到更多图（加上我太菜），所以其它的植物，僵尸以及场景还没能实现。

如果说控制台只是开始吓了我一下，QT 就是从头折磨人到尾。我的大部分时间都耗费在找坐标，定大小，卡时间和调参数上面了，因为第二次课设已经把面向对象编程的框架搭好，第三次基本上就是倒腾图片以及对着一大堆内置类和花哨的函数干瞪眼。比较蠢一件事的就是发现自己调了参数也打不过挑战模式，于是加班加点增加了三线射手和火爆辣椒。

QT 让我更加体会到面向对象编程的特点，内置类使得几乎所有操作都是对对象的函数调用。在前两次的课设中，因为初次写一个面向对象的控制台程序，代码比较混乱，并且具有较多的面向过程痕迹残留。这次被迫使用 QT 的内置类，我稍微了解了一些类之间的继承关系以及使用方法，更加适应面向对象的编程思想，学会了 QT 中的很多基础方法，为之后写更复杂的图形界面程序作了铺垫。

之前没想过能活到现在，可能是到期中为止还可以的上机成绩支撑着我没有退课。这门课让我收获良多，从三月份第一次上机时连.h 和.cpp 都不会写，到三个月后写完了基于图形界面的植物大战僵尸。虽然只是略微接触到了图形界面编程的皮毛，但对于我个人而言确实是不小的进步。

谢谢！