PVZ-UI

课程设计分析

课程设计目标

在第一、第二次课程设计的基础上，实现图形化界面，保证一定的可玩性

课程设计基础内容

使用Qt改写现有的程序。增加背景音乐。增加暂停功能，增加铲子铲除植物的功能。

设计分析

开发工具

根据qt官方文档，由于PVZ主要使用二维的图像（无论是动态还是静态），所以采用GraphView 框架更加合适

GraphView中的Scene提供了图形对象Item展示的场景，View负责可视化Item。多个View可以观察单个Scene

View提供了对Item管理和操作的大量接口，这里主要使用了以下功能：

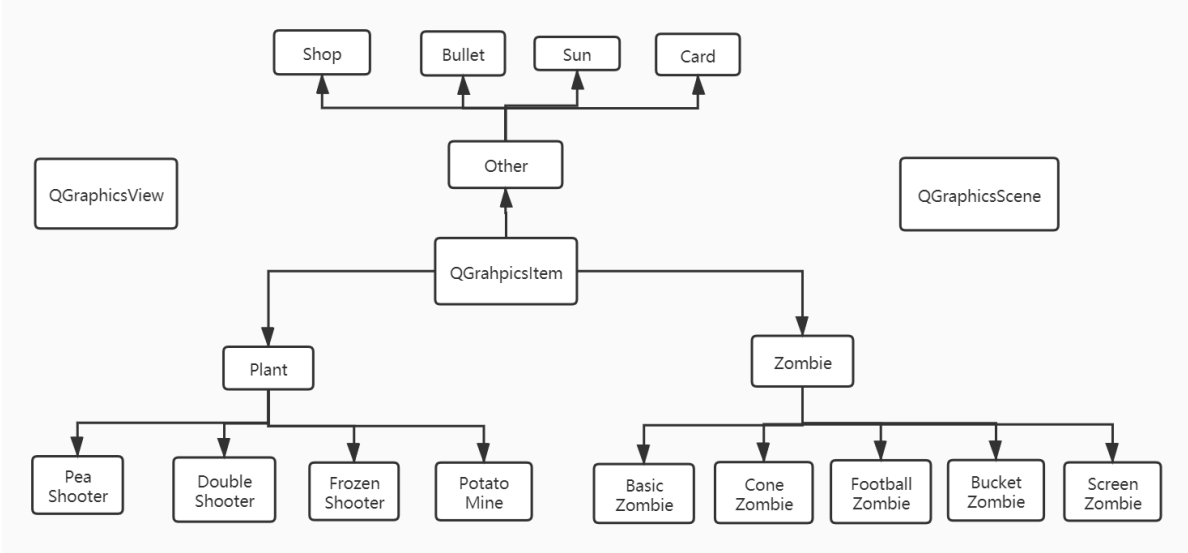
- 通过接口获取Item
- 将事件告知每个Item
- 管理Item状态，比如选中
- 打印

重构代码

与基于Qt的实现相比，之前的代码使用面向对象的思路还不够彻底。比如主流程仍然是过程式的，这也导致当时的刷新需要依赖花园来完成，植物和僵尸之间的信息传递也要依赖花园。而在Qt这一面向对象的开发平台中，由于各个对象显示和刷新是异步的，所以各个对象自然而然以相对独立的方式交换信息。因此，除了植物和僵尸，商店、铲子、商店内展示的植物卡牌，显示暂停的控件，都需要构建成类与对象。

生命值、攻击值等设置保持一致，检测碰撞、攻击等函数进行了继承、改写。考虑到实现难度和游戏可玩性，改写了植物种类

核心类的关系展示



代码实现

植物

植物整体的父类plant：

```
#include<QGraphicsItem>
#include<QPainter>
#include<QMovie>

class Plant:public QGraphicsItem
{
public:
    int life;//生命值
    int state;//状态
    enum{Type = UserType+1};//定义类型 便于其他类之间互相判断
    Plant();
    ~Plant() override;
    QRectF boundingRect() const override;//绘制矩形边界
    void paint(QPainter *painter, const QStyleOptionGraphicsItem *option,
QWidget *widget = nullptr) override;//打印出图案
    bool collidesWithItem(const QGraphicsItem *other, Qt::ItemSelectionMode
mode) const override;//检测碰撞
    int type() const override;
    void setMovie(QString path);//设置动画
```

```
protected:
    QMovie *movie;//动画效果
    int attack;//如果是攻击性植物 攻击值
    int counter;//计时器
    int time;//规定时间间隔
};

#endif // PLANT_H
```

PeaShooter:

```
PeaShooter::PeaShooter()
{
    attack=25;
    life=300;
    time = int(1.4*1000/33);//时间间隔 判断是否有行动 如发射子弹
    setMovie(":/image/PeaShooter.gif");//播放动画
}

void PeaShooter::advance(int phase)
{
    if(!phase)
        return;
    update();
    if(life<=0)
        delete this;
    else if(++counter>=time)
    {
        counter=0;
        if(!collidingItems().empty())
        {
            Bullet* bullet = new Bullet(attack);//子弹 思路与前两次一致 本行检测到僵尸
就生成子弹
            bullet->setX(x()+32);
            bullet->setY(y());
            scene()->addItem(bullet);
            return;
        }
    }
}

bool PeaShooter::collidesWithItem(const QGraphicsItem *other,
Qt::ItemSelectionMode mode) const
{
    Q_UNUSED(mode)
    return other->type()==Zombie::Type && qFuzzyCompare(other->y(),y());//检查本行
有无僵尸 不需要相邻
}
```

PotatoMine:

```
//主要展示state的用法，在此处，PotatoMine状态 0 可被攻击的准备态 1 准备爆炸 2 爆炸之后清除
PotatoMine::PotatoMine()
{
    attack=1800;
    life = 300;
```

```
    time = int(15.0*1000/33);
    setMovie("://image/PotatoMine1.gif");
}

QRectF PotatoMine::boundingRect() const
{
    return state==2 ? QRectF(-75,-75,150,150):Plant::boundingRect();
}

void PotatoMine::advance(int phase)
{
    if(!phase)
        return;
    update();
    if(life<=0)
        delete this;
    else if(state==0 && ++counter>=time)//超过一定时间才进入可以爆炸的阶段
    {
        state=1;
        counter=0;
        time=int(1.0*1000/33);
        setMovie("://image/PotatoMine.gif");
    }
    else if(state==1 && ++counter>=time)
    {
        counter=0;
        QList<QGraphicsItem*>items = collidingItems();
        if(!items.empty())
        {
            state=2;
            setMovie("://image/PotatoMineBomb.gif");
            foreach(QGraphicsItem *item,items)
            {
                Zombie* zom=qgraphicsitem_cast<Zombie*>(item);
                zom->life-=attack;
                if(zom->life<=0)
                    delete zom;
            }
        }
    }
    else if(state==2 && movie->currentFrameNumber()==movie->frameCount()-1)//帧数
播放到最后一帧作为停止判断条件
        delete this;//
}

bool PotatoMine::collidesWithItem(const QGraphicsItem *other,
Qt::ItemSelectionMode mode) const
{
    Q_UNUSED(mode)
    return other->type() == Zombie::Type && qFuzzyCompare(other->y(),y())
            && qAbs(other->x()-x())<50;
}
```

僵尸

Zombie：

```
class Zombie:public QGraphicsItem
{
public:
    int life;//生命值
    int attack;//攻击值
    int state;//状态 state 0 行走 1 攻击 2 死亡
    qreal speed;//前进速度
    enum {Type = UserType+2};
    Zombie();
    ~Zombie() override;
    QRectF boundingRect() const override;
    void paint(QPainter *painter, const QStyleOptionGraphicsItem *option,
QWidget *widget = nullptr) override;
    bool collidesWithItem(const QGraphicsItem *other, Qt::ItemSelectionMode
mode) const override;
    int type() const override;
    void setMovie(QString path);//播放动画
    void setHead(QString path);//掉头的动画
protected:
    QMovie *movie;//动画
    QMovie *head;//掉头动画
};
```

BasicZombie：

```
BasicZombie::BasicZombie()
{
    life = 270;
    attack = 100 *33/1000;
    speed = 80.0*33/1000/4.7;
    if(qrand()%2)
        setMovie(":/image/ZombieWalk1.gif");
    else
        setMovie(":/image/ZombieWalk2.gif");
}

void BasicZombie::advance(int phase)
{
    if(!phase)
        return;
    update();
    if(life<=0)
    {
        if(state<2)//state 0 行走 1 攻击 2 死亡
        {
            state=2;
            setMovie(":/image/ZombieDie.gif");
            setHead(":/image/ZombieHead.gif");
        }
        else if(movie->currentFrameNumber()==movie->frameCount()-1)
            delete this;
        return;
```

```
    }
    QList<QGraphicsItem*> items = collidingItems();
    if(!items.isEmpty())
    {
        Plant *plant = qgraphicsitem_cast<Plant*>(items[0]);
        plant->life-=attack;//僵尸发动攻击
        if(state!=1)
        {
            state=1;
            setMovie(":/image/ZombieAttack.gif");
        }
        return ;
    }
    if(state)
    {
        state=0;
        if(qrand()%2)
            setMovie(":/image/ZombieWalk1.gif");
        else
            setMovie(":/image/ZombieWalk2.gif");
    }
    setX(x()-speed);
}
```

其他页面控件

Shop:

```
Shop::Shop()
{
    sun_deposit=200;
    counter=0;
    time = int(7.0*1000/33);
    Card* card=nullptr;
    for(int i=0;i<Card::name.size();++i)
    {
        card = new Card(Card::name[i]);
        card->setParentItem(this);//绑定父类
        card->setPos(-157+65*i,-2);
    }
    //show the plant choice
}

void Shop::paint(QPainter *painter, const QStyleOptionGraphicsItem *option,
QWidget *widget)
{
    Q_UNUSED(option)
    Q_UNUSED(widget)
    painter->drawPixmap(QRect(-270, -45, 540, 90),
QPixmap(":/image/Shop.png"));
    QFont font;
    font.setPointSizeF(15);
    painter->setFont(font);
    painter->drawText(QRectF(-255, 18, 65, 22), Qt::AlignCenter,
QString::number(sun_deposit));
    painter->drawPoint(-220, 0);
}
```

```
QRectF Shop::boundingRect() const
{
    return QRectF(-270, -45, 540, 90);
}
void Shop::advance(int phase)
{
    if(!phase)
        return;
    update();
    if(++counter>=time)
    {
        counter=0;
        scene()->addItem(new Sun);//按照一定时间周期生成太阳
    }
}

void Shop::addPlant(QString s, QPointF pos)
{
    QList<QGraphicsItem *>items = scene()->items(pos);//获取该位置上对象
    foreach (QGraphicsItem *item,items)
    {
        if(item->type()==Plant::Type) //如果已经有植物 不能再种
            return;
    }

    Plant *plant = nullptr;
    int index = Card::map[s];
    switch(index)
    {
        case 0:
            plant = new Sunflower;break;
        case 1:
            plant =new PeaShooter;break;
        case 2:
            plant = new DoubleShooter;break;
        case 3:
            plant = new FrozenShooter;break;
        case 4:
            plant = new Nut;break;
        case 5:
            plant = new PotatoMine;break;
        case 6:
            plant = new CherryBomb;break;

    }
    plant->setPos(pos);
    int price = Card::price[index];
    int sun = sun_deposit;
    sun -= price;
    sun_deposit = sun_deposit-price;//更新太阳

    scene()->addItem(plant);
    QList<QGraphicsItem*> child = this->childItems();//获取卡牌
    foreach(QGraphicsItem *item,child)
    {
        Card *card = qgraphicsitem_cast<Card*>(item);
        if(card->text==s)
            card->counter=0;//卡牌进入冷却期
```

```
    }
    counter=0;
}
```

Card:

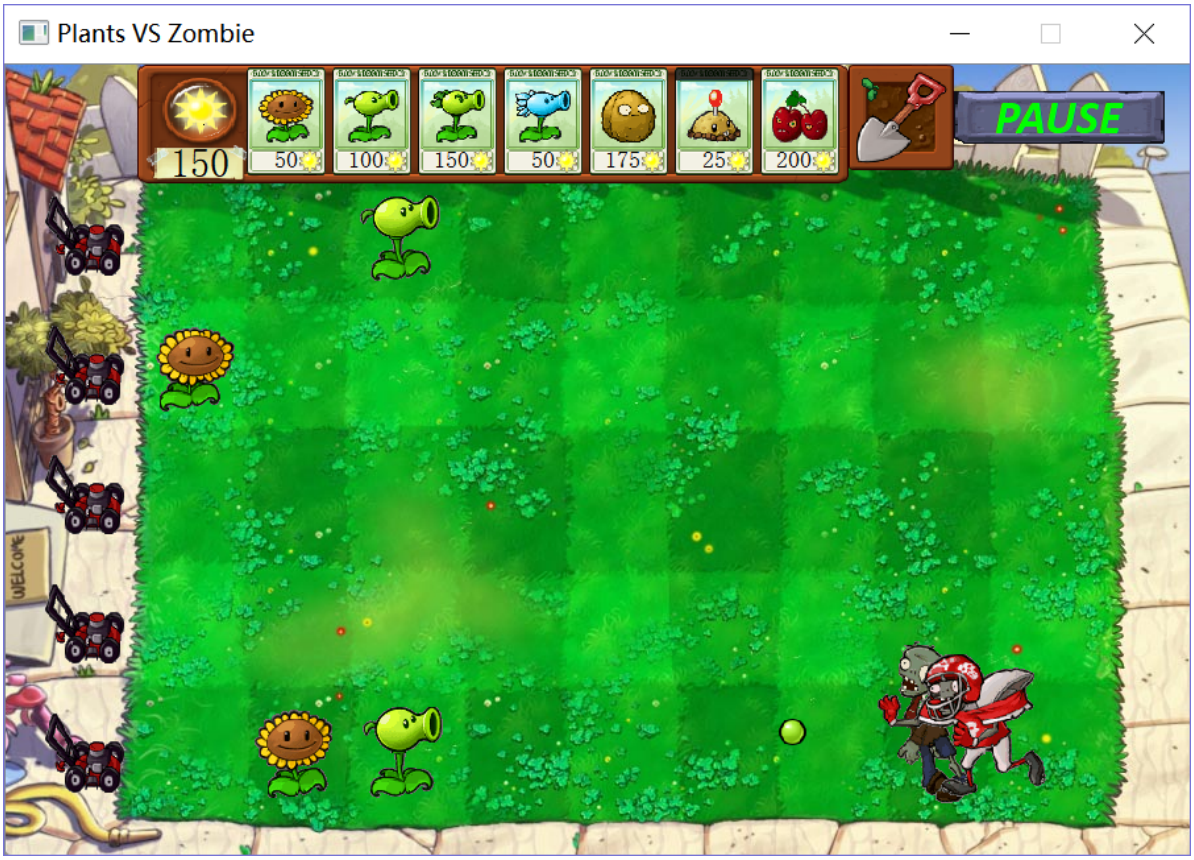
```
//初始化类通用的表格和哈希表
const QMap<QString,int>Card::map = {{"Sunflower",0},{"PeaShooter",1},
{"DoubleShooter",2}
                                    ,{"FrozenShooter",3},{"Nut",4},
{"PotatoMine",5}
                                    ,{"CherryBomb",6}};
const QVector<QString>Card::name = {"Sunflower","PeaShooter","DoubleShooter",
                                    "FrozenShooter","Nut","PotatoMine",
                                    "CherryBomb"};
const QVector<int>Card::price={50,100,150,50,175,25,200};
const QVector<int>Card::cool = {227,227,606,606,227,606,227};
Card::Card(QString s)
{
    text=s;
    counter=0;
}
QRectF Card::boundingRect() const
{
    return QRectF(-50,-30,100,60);
}
void Card::paint(QPainter *painter, const QStyleOptionGraphicsItem *option,
QWidget *widget)
{
   Q_UNUSED(option)
   Q_UNUSED(widget) //不使用此参数
   painter->scale(0.6,0.58);
   painter->drawPixmap(QRect(-50,-70,100,140),QPixmap("://image/Card.png"));
   painter->drawPixmap(QRect(-35,-42,70,70),QPixmap("://image/"+text+".png"));
   QFont font;
   font.setPointSizeF(15);//设置字号
   painter->setFont(font);
   painter->drawText(-30,60,QString().sprintf("%3d",price[map[text]]));
   if(counter<cool[map[text]])
   {
       QBrush brush(QColor(0,0,0,200));
       painter->setBrush(brush);//涂黑卡片表示在冷却
       painter->drawRect(QRectF(-48,-68,98,132*(1-
qreal(counter)/cool[map[text]])));//除法转成double 根据时间长短来决定涂黑部分在卡牌上的
占比
   }
}
void Card::advance(int phase)
{
    if(!phase)
        return ;
    update();//更新
    if(counter<cool[map[text]])
        ++counter;
}
void Card::mousePressEvent(QGraphicsSceneMouseEvent *event)
```

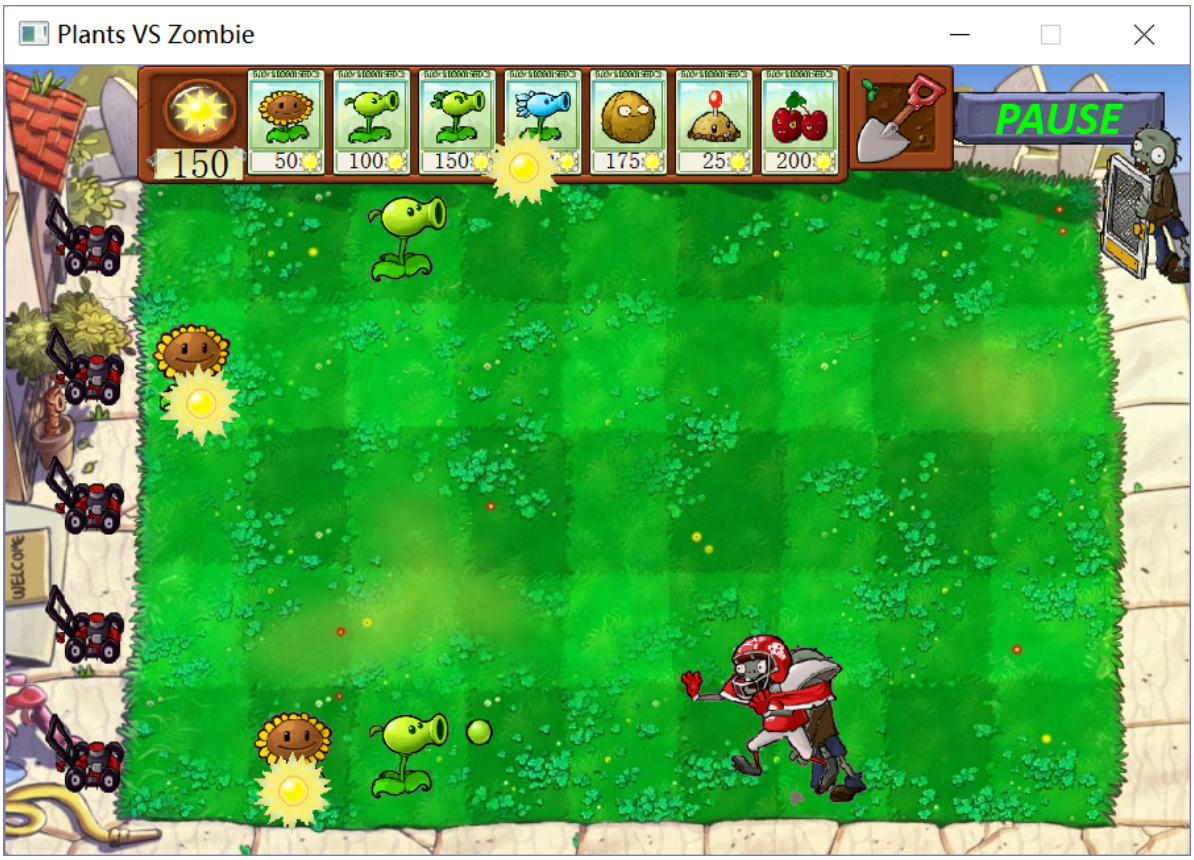
```
{
    Q_UNUSED(event)
    if(counter<cool[map[text]])
        event->setAccepted(false);
    Shop* shop=qgraphicsitem_cast<Shop*>(parentItem());
            //场景视图Item类转换 实例化转换的函数
    if(price[map[text]]>shop->sun_deposit)//无法购买 忽略点击事件
        event->setAccepted(false);
    setCursor(Qt::ArrowCursor);//保持对象的形状
}

void Card::mouseMoveEvent(QGraphicsSceneMouseEvent *event)
{
    if(QLineF(event->screenPos(),event-
>buttonDownScreenPos(Qt::LeftButton)).length()<
            QApplication::startDragDistance())
        return ;//拖拽的距离要明显一些
    QDrag* drag=new QDrag(event->widget());
    QMimeData *mime = new QMimeData;//多媒体数据类
    QImage image(";//image/"+text+".png");
    mime->setText(text);
    mime->setImageData(image);
    drag->setMimeData(mime);
    drag->setPixmap(QPixmap::fromImage(image));
    drag->setHotSpot(QPoint(35,35));//定位
    drag->exec();
    setCursor(Qt::ArrowCursor);
}
void Card::mouseReleaseEvent(QGraphicsSceneMouseEvent *event)
{
    Q_UNUSED(event)
    setCursor(Qt::ArrowCursor);
}
```

程序的运行方法

解压PVZ-Demo并在解压出的文件夹中直接运行PVZ-UI.exe


Plants VS Zombie
150
50
100
150
50
175
25
200
PAUSE
WELCOME


Plants VS Zombie
150
50
100
150
175
25
200
PAUSE
WELCOME

其他玩法与原版游戏一致，如拖拽植物卡牌种植植物，拖拽铲子铲除植物，点击右上角图标可以暂停/继续游戏

遇到的问题和解决方案

对新开发工具无从下手

一开始阅读了官方文档，发现还是一头雾水，于是尝试搜索类似用qt实现的游戏，借鉴他人用了怎样的框架，慢慢找到方向。因为Qt库支持非常多样的功能，不可能在通读手册之后再进行选择，以应用为导向搜索，学习他人优秀的代码架构，能够更快上手。

大量代码面临重构

第一、第二次课程设计仍然有许多面向过程设计的思想，比如植物和僵尸的信息交流，比如页面的刷新，商店功能。然而Qt是面向对象特征非常明显的开发平台，于是在第三次课程设计基本要推翻重写。但是有了第一、第二次课设中，植物类之间的继承关系，以及对于花园位置的规划的思考，重构并不需要花太多的时间。需要添加的主要是商店、植物卡牌、太阳等之前在课设中没有明显体现的对象。

我对Qt的理解是，通过面向对象的方式，Qt支持了不同对象在屏幕上各自按照提供的动画显示，如僵尸的前进与太阳的掉落并不互相干扰。相比于顺序的程序调用，Qt设计下的游戏主要是事件驱动。最明显的区别体现在商店的调用上，前两次课设设计下需要在循环之间等待，而本次只需要重定义鼠标点击和拖拽事件即可。在学习本课程之前，有同学建议我一开始就用Qt构建，但是我认为从控制台转移到GUI的过程能帮助我更好的理解面向过程和面向对象这两种程序设计范式的区别以及各自适用的应用场景。

程序打包比基于控制台的应用困难

之前提交的时候只需要直接提交exe文件，这次由于开发中调用了许多库，所以在打包的过程中需要通过下载、复制粘贴等方式补全。