高级语言程序设计

实验报告

南开大学 计算机大类
姓名：黄俊雄
学号：2313896
班级：计算机科学与技术卓越班
2024 年 5 月 13 日

目录

高级语言程序设计大作业实验报告

一. 作业内容

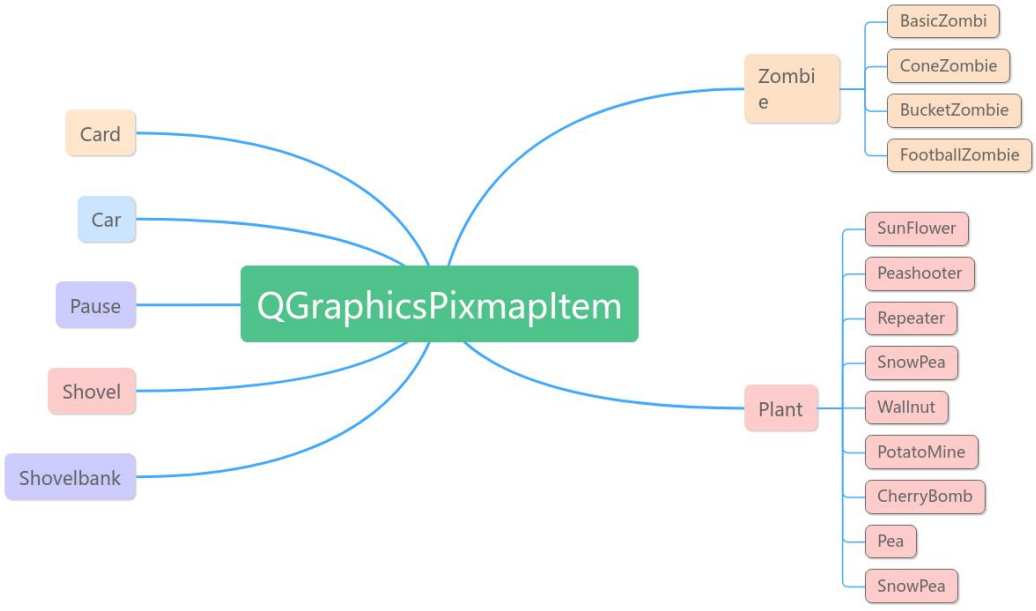
使用 Qt 实现植物大战僵尸

二. 开发软件

Qt6.7.0

三. 主要流程

1．继承关系



2．主要类

（1）Plant 类

头文件：

```
class Plant : public QObject,public QGraphicsPixmapItem
{
public:
    int line,row=0;
    Plant(const QString &PlantImagePath,QGraphicsPixmapItem *parent=nullptr);
    ~Plant();
    void update();
    void setGifPath(const QString &path);
    int HP=300;
    int timerId;
    int iskilltimerId=1;
    QMovie *movie;
};
```

line 和 row 用来表示植物的位置（在某行某列），HP 为植物生命值

PlantImagePath 为一个路径，为构造子类时子类动图路径

cpp 文件：

```
Plant::Plant(const QString &PlantImagePath,QGraphicsPixmapItem *parent)
: QGraphicsPixmapItem(parent), movie(new QMovie(this))
{
    setGifPath(PlantImagePath);
    connect(movie, &QMovie::frameChanged, this, &Plant::update);
}
Plant::~Plant()
{
    delete movie;
}
void Plant::update()
{
    setPixmap(movie->currentPixmap());
    if(HP<=0)
    {
        movie->stop();
        isOccupied[row][line]=1;
        if(iskilltimerId)
            killTimer(timerId);
        scene()->removeItem(this);
    }
}
void Plant::setGifPath(const QString &path)
{
    // 设置 GIF 文件的路径并启动动画
    movie->setFileName(path);
    movie->setSpeed(100); // 设置播放速度为正常速度的1.5倍
    movie->start();
}
```

Plant 基类实现植物动图播放已经判断植物是否死亡，植物各自的特性由具

体的子类实现。

（2）Zombie 类

头文件：

```
class Zombie: public QObject,public QGraphicsPixmapItem
{
    Q_OBJECT
public:
    Zombie(const QString path,QGraphicsPixmapItem *parent=nullptr);
    ~Zombie();
    void death();
    void Burn();
    void GameLose();
    void GameWin();
    int ItemKind(QGraphicsItem *item);
    int row;
    QMovie *WalkMovie;
    QMovie *AttackMovie;
    QMovie *DieMovie;
    QMovie *BurnMovie;
    double walkspeed=1.5;
    int attack=2;
    int HP=270;
    int iswear=0;
    int ZombieKind;
    int isWalk=1;
    int isPlant=0;
    int isGamelose=0;
protected:
    virtual void timerEvent(QTimerEvent *event) override;
private slots:
    void updateWalk();
    void updateAttack();
    void updateDie();
    void updateBurn();
    void Walk();
    void Attack();
    void stopQMovie();
    void killSelf();
private:
    int timerId;
    int isdead=1;
};
```

cpp 文件：

```
Zombie::Zombie(const QString path,QGraphicsPixmapItem *parent)
    : QGraphicsPixmapItem(parent), WalkMovie(new QMovie(this)),AttackMovie(new QMovie(this)),DieMovie(new QMovie(this)),BurnMovie(new QMovie(this))
{
    connect(WalkMovie, &QMovie::frameChanged, this, &Zombie::updateWalk);
    connect(AttackMovie, &QMovie::frameChanged, this, &Zombie::updateAttack);
    connect(DieMovie, &QMovie::frameChanged, this, &Zombie::updateDie);
    connect(BurnMovie, &QMovie::frameChanged, this, &Zombie::updateBurn);
    BurnMovie->setFileName(":/zombie/images/Burn.gif");
    WalkMovie->setFileName(path);
    WalkMovie->setSpeed(150); // 设置播放速度为正常速度的1.5倍
    WalkMovie->start();
    timerId = startTimer(20);
}
```

连接四个 QMovie 更新的信号和更新 Pixmap 的槽函数，设置初始状态为行走模式

```
void Zombie::updateWalk()
{
    setPos(pos().x()-walkspeed,pos().y());
    setPixmap(WalkMovie->currentPixmap());
}
```

槽函数中更新 pixmap 为 QMovie 的当前 pixmap，行走状态中更新僵尸位置，使其向左移动 walkpeed 的距离，

重点：timerEvent 函数

```
void Zombie::timerEvent(QTimerEvent *)
{
    if(!iscontinue)
    {
        WalkMovie->stop();
        AttackMovie->stop();
        return;
    }
    if(isWalk)
        Walk();
    if(isGamelose==0&&pos().x()<-100)
    {
        GameLose();isGamelose=1;}
```

iscontinue 为一个全局变量，初始值为 1，当按下暂停键时变为 0，此时所有 timerEvent 都会直接返回，这里暂停 QMovie 并返回。isWalk 变量初始时为 1，在僵尸遇到植物后会变为 0 进入攻击模式，植物死亡后又变为 1，就会回到行走模式。第三个是判断僵尸的位置，当僵尸到达指定位置是游戏结束。

碰撞检测:

```
QList<QGraphicsItem *> items=collidingItems();
for(auto item:items)
{
    int kind=ItemKind(item);
    switch (kind) {
    case 0:
        break;
    case 1:
    {
        scene()->removeItem(item);
        delete item;
        HP-=20;
        if(isdead==1&&HP<90)
        {
            isdead=0;
            isZombie[row]--;dienumber++;
            if(dienumber==14)
                GameWin();
            death();
            killTimer(timerId);
            return;
        }
        break;
    }
    case 2: {...}
    case 3: {...}
    case 4: {...}
    case 5: {...}
    default:
        break;
    }
}
```

```
int Zombie::ItemKind(QGraphicsItem *item)
{
    if(typeid(*item)==typeid(Pea)||typeid(*item)==typeid(PeaSnow))
        return 1;
    if(typeid(*item)==typeid(SunFlower)||typeid(*item)==typeid(SnowPea)||typeid(*item)==typeid(PeaShooter))
        return 2;
    if(typeid(*item)==typeid(WallNut)||typeid(*item)==typeid(Repeater))
        return 2;
    if(typeid(*item)==typeid(PotatoMine))
        return 3;
    if(typeid(*item)==typeid(CherryBomb))
        return 4;
    if(typeid(*item)==typeid(car))
        return 5;
    return 0;
}
```

ItemKind 为判断与僵尸碰撞的组件种类的函数，Switch 语句根据 ItemKind 返回的值执行对应操作，如 ItemKind 返回 1 表示与僵尸碰撞的是豌豆，则僵尸生命值减少，同时判断僵尸是否死亡等。

Zombie 基类实现了僵尸的移动攻击死亡等大部分功能，也是代码较长的一个类

（3）Card 类

头文件：

```
#include<QGraphicsItem>
#include<QPainter>
#include<QTimer>
class card:public QObject,public QGraphicsItem
{
    Q_OBJECT
public:
    card(QGraphicsItem *parent = nullptr);
    int kind;//kind 表示植物种类，0-向日葵，1-豌豆射手，2-寒冰射手，3-双发射手，4-坚果，5-土豆地雷，6-樱桃炸弹
    int cardCD=7500;
    int x,y;
    double scale;
    int Price;
    QString PlantImagePath;
    int isblack=0;
    QTimer *timer;
    int n=150;
    int Originaln=150;
protected:
    int isPlant=1;
    void paint(QPainter *painter, const QStyleOptionGraphicsItem *option, QWidget *widget) override;
    void mousePressEvent(QGraphicsSceneMouseEvent *event) override;
    void mouseMoveEvent(QGraphicsSceneMouseEvent *event) override;
    void mouseReleaseEvent(QGraphicsSceneMouseEvent *event) override;
    void setPlant(QPointF position);
    void CardCD();
    void updatemap();
    QRectF boundingRect() const override {
        return QRectF(0, 0, 50,70);
    }
private:
    QPointF OriginalPosition;
    card *tempCard= nullptr;
};
```

cpp：

```
card::card(QGraphicsItem *)
    :timer(new QTimer(this))
{
    setFlag(QGraphicsItem::ItemIsMovable, true);
    connect(timer,&QTimer::timeout,this,&card::updatemap);
}
void card::paint(QPainter *painter, const QStyleOptionGraphicsItem *option, QWidget *widget)
{
    Q_UNUSED(option)
    Q_UNUSED(widget)
    painter->drawPixmap(QRect(0,0, 50,70), QPixmap(":/other/images/Card.png"));
    painter->drawPixmap(QRect(10, 15, 30, 35), QPixmap(PlantImagePath));
    painter->drawText(5,65, QString::number(Price));
    if (isblack)
    {
        QBrush brush(QColor(0, 0, 0, 200));
        painter->setBrush(brush);
        painter->drawRect(QRectF(0,0,50,70*qreal(n)/Originaln));
        isblack=0;
    }
}
```

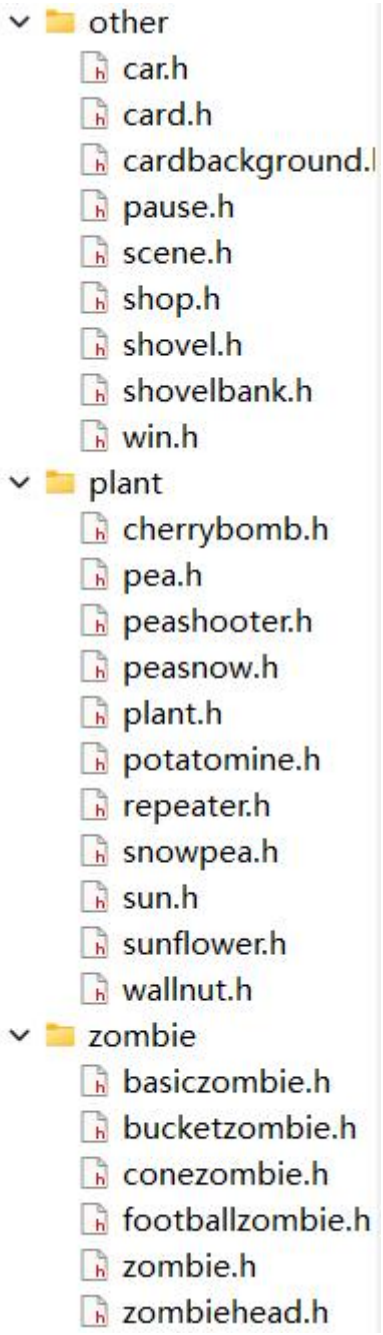
这部分内容主要实现植物种植后植物卡片进入冷却，无法再次种植，植物卡片变暗并从底部慢慢变亮，完全变量后即可再次种植。

```
void card::CardCD()
{
    setFlag(QGraphicsItem::ItemIsMovable, false);
    timer->start(50);
    isblack=1;
}
void card::updatemap()
{
    if(!iscontinue)
        return;
    isblack=1;
    update();
    n--;
    if(n==0)
    {
        isPlant=1;
        isblack=0;
        timer->stop();
        setFlag(QGraphicsItem::ItemIsMovable, true);
        n=Originaln;
    }
}
```

这部分内容主要实现植物种植后植物卡片进入冷却，无法再次种植，植物卡片变暗并从底部慢慢变亮，完全变量后即可再次种植。

```
void card::setPlant(QPointF position)
{
    if(isPlant&&position.x()-75>=0&&position.y()-94>=0)
    {
        int line=(position.x()-94)/82,row=(position.y()-75)/100;
        if(0<=row&&row<5&&0<=line&&line<9&&isOccupied[row][line]==1&&sunshine>=Price)
        {
            switch (this->kind)
            {
            case 0:
            {
                SunFlower *plant=new SunFlower;
                plant->setPos(94+line*82,90+row*98);
                plant->line=line;plant->row=row;
                scene()->addItem(plant);
                break;}
            case 1: {...}
            case 2: {...}
            case 3: {...}
            case 4: {...}
            case 5: {...}
            case 6: {...}
            default:
                break;
            }
            isOccupied[row][line]=0;
            sunshine-=Price;
            showsunshine->setPlainText(QString::number(sunshine));
            isPlant--;
            CardCD();
        }
    }
}
```

setPlant 函数实现植物种植，检测种植位置是否为可种植区域（超出格子范围或格子上已有植物则无法种植）以及阳光数量是否足够，根据 kind 的不同值种植不同的植物并减少相应的阳光数量。



除此之外一共有这些类，这里就不一一展示，想了解详细代码的可以通过以下网站

https://github.com/NKU-waiting/pvz

四. 个人收获

这一两个月来我还是收获蛮多的吧（修 bug 的能力明显提升），

林林总总加起来一共有上千行代码，写这些代码时各种大大小小的 bug 层出不穷，这一千多行代码也是各种修修改改，或许还有许多可以优化的地方，但这已经是我在 deadline 之前所能做到的最好的结果了。（原先我一直以为 deadline 是 26 号，结果发现 14 号就是 deadline 了）

五. 后续

后续如果有时间的话可能会继续完善这个项目吧，现在目前还是有很多可以优化可以改善的地方，也有一些东西还没完成（比如寒冰射手的减速效果，以及原本打算设计的路障僵尸）