高级语言程序设计 2-2 大作业实验报告

2112614　计算机卓越班 刘心源

南开大学 计算机学院

目录

1.项目介绍

使用QT开发的图形化小游戏，玩法类似于植物大战僵尸，但是与其他的游戏做了一些结合

2.开发软件

Qt 5.9.0 ， Qt Creator 4.3.0，使用Qt安装的MinGW 5.3.0 编译器

3.设计思路

3.1 软件架构

采用Qt的Graphics View 框架，Graphics View框架提供了一个用于管理和交互大量定制的图形项Item的场景Scene以及一个用于可视化这些图形项Item的视图View。

采用了面向对象的编程方法，使用了封装，继承，多态等等，自定义的三个类都继承于QGraphicsItem：

- 人物基类 yuanshen ；
- 史莱姆基类 slime ；
- 其他基类 other，其派生类包括Shop，Map，Card 。

3.1.1 人物

人物的属性包括：

- 生命值 hp
- 行动状态state（如炸弹有引爆状态）
- 攻击力atk
- 计数器counter于时间间隔time
- 动画movie，用于加载GIF图片
- 语音sound，用于人物上场的时候播放

人物的函数包括：

- BoundingRect(),返回人物的边界矩形
- paint(),绘制人物
- collidesWithItem(),判定是否碰撞
- advance(),根据计数器和状态，进行碰撞的检测，完成行动与状态的转移
- setMovie()，设置动画效果
- setSound(),设置声音效果
-

3.1.2 史莱姆

史莱姆的属性包括：

- 生命值 hp
- 行动状态 state （行走，被烧焦）

- 攻击力 atk
- 速度 speed ， 用于移动
- 动画 movie ，用于加载GIF动图
- 元素状态 now 用来记录当前的元素状态

史莱姆的函数包括：
- boundingRect(),返回史莱姆的边界矩形
- paint(),绘制史莱姆
- collodesWithItem()，判定是否碰撞
- advance()，根据状态，进行碰撞检测以及完成行动和状态的转移
- setMovie()，设置动画效果

3.1.3 其他

其他基类的派生类较多，包括卡片槽，卡片，铲子，按钮，地图，割草机，攻击的子弹，阳光等等。

这些部件和人物的类，史莱姆类的函数互相配合调用。

同时我们也可以看到，人物类与史莱姆类的属性是十分的相近，函数也大多是基类QGraphicsItem 的函数，支持重载。

3.2 界面设计

界面仿照的原版的植物大战僵尸，并对其进行了一些自己的改动，如卡牌，攻击系统等等的改变。

界面中的静态物体通过绘制PNG格式的图片进行实现，而动态物体通过绘制GIF格式的图片来实现。

3.3 Graphics View的框架的介绍

3.3.1 Graphics View

Graphics View提供了一种基于Item的Model-View编程，多个View可以观察单个Scene，Scene 中包含不同几何形状的Item。QGraphics View 提供了 View 组件，用于可视化场景中的内容。视图可以从键盘和鼠标接收事件，并将这些事件转换成场景事件（同时将坐标转换为场景坐标），然后将其发送给场景。

3.3.2 QGraphicsScene

QGraphicsScene提供了Graphics View的场景，可以
- 提供一个高性能的接口来管理大量的Items
- 将事件传播到每个Item
- 管理Item状态，如选择和焦点处理
- 提供未被变换的渲染能力，主要用于打印

3.3.3 QGraphicsItem

QGraphicsItem 是场景中图形项Item的基类，支持以下的功能：

- 鼠标按下，移动，释放和双击事件
- 键盘输入焦点以及按键事件
- 拖放事件
- 碰撞检测

Item有许多非常重要的函数，如：

- ✓ boundingRect()返回图形项的边界矩形坐标范围
- ✓ paint()用于绘制图形：绘制背景图，各种部件，人物，史莱姆等等
- ✓ 为了实现碰撞检测，可以重写collidesWith()实现自定义碰撞检测。

4.代码实现

4.1 MainWindow

MainWindow 控制整个游戏的运行，其具体流程为：

- 播放背景音乐
- 创建场景并且设置边界
- 创建部分控件并且加入场景
- 创建视图，设置背景和大小
- 创建计时器，并将计时器事件timeout()绑定到场景advance()
- 将计时器事件绑定到史莱姆生成的函数addSlime()和胜负的判断函数check()

下面是MainWindow()构造函数的主要内容:

```
setWindowIcon(QIcon("./images/window.ico"));

//随机数种子
qsrand(uint(QTime(0,0,0).secsTo(QTime::currentTime())));

sound = new QSound(":/images/Music2.wav");
sound->setLoops(QSound::Infinite);

timer = new QTimer;

//创建场景
scene = new QGraphicsScene(this);//建立坐标系
scene->setSceneRect(150, 0, 900, 600);
scene->setItemIndexMethod(QGraphicsScene::NoIndex);


//加入卡片槽等部件
Shop *shop = new Shop;
shop->setPos(520, 45);
scene->addItem(shop);

Shovel *shovel = new Shovel;
shovel->setPos(830, 40);
scene->addItem(shovel);

Button *button = new Button(sound, timer);
button->setPos(970, 20);
scene->addItem(button);

Map *map = new Map;
map->setPos(619, 325);
scene->addItem(map);
```

```
for (int i = 0; i < 5; ++i)
{
    Mower *mower = new Mower;
    mower->setPos(210, 130 + 98 * i);
    scene->addItem(mower);
}


//背景图场景
view = new QGraphicsView(scene, this);
view->resize(902, 602);

view->setRenderHint(QPainter::Antialiasing);
view->setBackgroundBrush(QPixmap(":/images/Background.jpg"));

view->setCacheMode(QGraphicsView::CacheBackground);
view->setViewportUpdateMode(QGraphicsView::BoundingRectViewportUpdate);

connect(timer, &QTimer::timeout, scene, &QGraphicsScene::advance);
connect(timer, &QTimer::timeout, this, &MainWindow::addSlime);
connect(timer, &QTimer::timeout, this, &MainWindow::check);

//播放音乐，启动定时器等等
sound->play();

timer->start(40);//表示每次timeout的时间间隔是40ms
view->show();
```

4.2 Yuanshen

yuanshen是植物基类

```
yuanshen::yuanshen()
{
    movie = nullptr;
    atk = counter = state = time = 0;
}

yuanshen::~yuanshen()
{
    delete movie;
    delete sound;
}

QRectF yuanshen::boundingRect() const
{
    return QRectF(-35, -35, 70, 70);
}

void yuanshen::paint(QPainter *painter, const QStyleOptionGraphicsItem *option, QWidget *widget)
{
    Q_UNUSED(option)
    Q_UNUSED(widget)
    painter->drawImage(boundingRect(), movie->currentImage());
}

bool yuanshen::collidesWithItem(const QGraphicsItem *other, Qt::ItemSelectionMode mode) const
{
    Q_UNUSED(mode)
    // 左右30像素内是否存在僵尸
    return other->type() == Slime::Type && qFuzzyCompare(other->y(), y()) && qAbs(other->x() - x()) < 30;
}

int yuanshen::type() const


int yuanshen::type() const
{
    return Type;
}

void yuanshen::setMovie(QString path)
{
    if (movie) delete movie;
    movie = new QMovie(path);
    movie->start();
}
void yuanshen::setSound(QString path)
{
    sound = new QSound(path);
    sound->play();
}
```

4.3 Slime

```
Slime::Slime()
{
    movie = nullptr;
    hp = atk = 0;
    speed = 0.0;
    state = SlimeType::WALK;
}

Slime::~Slime()
{
    delete movie;
}

QRectF Slime::boundingRect() const
{
    return QRectF(0, -30, 90, 80);
}

void Slime::paint(QPainter *painter, const QStyleOptionGraphicsItem *option, QWidget *widget)
{
    Q_UNUSED(option)
    Q_UNUSED(widget)
    QImage image = movie->currentImage();
    if (speed < 0.55 && state != SlimeType::BURN)
    {
        if (state != SlimeType::DIE) movie->setSpeed(40);
        int w = image.width();
        int h = image.height();
        for (int i = 0; i < h; ++i)
        {
            uchar *line = image.scanLine(i);

            for (int j = 5; j < w - 5; ++j)
                line[j << 2] = 150;
        }
    }
    painter->drawImage(QRectF(0, -30, 90, 80), image);
}

bool Slime::collidesWithItem(const QGraphicsItem *other, Qt::ItemSelectionMode mode) const
{
    Q_UNUSED(mode)
    return other->type() == yuanshen::Type && qFuzzyCompare(other->y(), y()) && qAbs(other->x() - x()) < 30;
}

int Slime::type() const
{
    return Type;
}

void Slime::setMovie(QString path)
{
    if (movie) delete movie;
    movie = new QMovie(path);
    movie->start();
}
```

4.4 Card

Card 支持鼠标事件，需要重写鼠标按下，移动和松开函数，实现拖曳

```
void Card::paint(QPainter *painter, const QStyleOptionGraphicsItem *option, QWidget *widget)
{
    //忽略“未引用的形参”警告
    Q_UNUSED(option)
    Q_UNUSED(widget)

    painter->scale(0.6, 0.58);

    //放上空卡片 和 植物图片
    painter->drawPixmap(QRect(-50, -70, 100, 140), QPixmap(":/images/Card.png"));
    painter->drawPixmap(QRect(-49, -69, 99, 105), QPixmap(":/images/" + text + ".jpg"));

    //写上价格
    QFont font;
    font.setPointSizeF(15);
    painter->setFont(font);
    painter->drawText(-30, 60, QString().sprintf("%3d", cost[map[text]]));


    if (counter < cool[map[text]])
    {
        QBrush brush(QColor(0, 0, 0, 200));
        painter->setBrush(brush);
        painter->drawRect(QRectF(-48, -68, 98, 132 * (1 - qreal(counter) / cool[map[text]])));
    }
}
```

```
//冷却时间未到达，继续冷却，counter++
void Card::advance(int phase)
{
    if (!phase)
        return;
    update();
    if (counter < cool[map[text]])
        counter++;
}


//鼠标事件

//鼠标按下
void Card::mousePressEvent(QGraphicsSceneMouseEvent *event)
{
    Q_UNUSED(event)

    //冷却时间未到，或者买不起，显示箭头
    if (counter < cool[map[text]]) event->setAccepted(false);
    Shop *shop = qgraphicsitem_cast<Shop *>(parentItem());
    if (cost[map[text]] > shop->sun) event->setAccepted(false);

    setCursor(Qt::ArrowCursor);
}
```

```
void Card::mouseMoveEvent(QGraphicsSceneMouseEvent *event)
{
    // 小于最小移动距离,不处理
    if (QLineF(event->screenPos(), event->buttonDownScreenPos(Qt::LeftButton)).length()< QApplication::startDragDistance())
        return;

    QDrag *drag = new QDrag(event->widget());//建立拖放对象
    QMimeData *mime = new QMimeData;
    QImage image(":/images/" + text + ".png");
    mime->setText(text);
    mime->setImageData(image);
    drag->setMimeData(mime);
    drag->setPixmap(QPixmap::fromImage(image));
    drag->setHotSpot(QPoint(40, 40));
    drag->exec();
    setCursor(Qt::ArrowCursor);
}

//鼠标释放，恢复箭头
void Card::mouseReleaseEvent(QGraphicsSceneMouseEvent *event)
{
    Q_UNUSED(event)
    setCursor(Qt::ArrowCursor);
}
```

4.5 Map

Map需要处理拖放事情，实现种植或者铲除植物

```
//开始拖拽
void Map::dragEnterEvent(QGraphicsSceneDragDropEvent *event)
{
    if (event->mimeData()->hasText())
    {
        event->setAccepted(true);
        dragOver = true;
        update();
    }
    else
        event->setAccepted(false);
}

//拖拽离开
void Map::dragLeaveEvent(QGraphicsSceneDragDropEvent *event)
{
    Q_UNUSED(event);
    dragOver = false;
    update();
}


//拖拽后的结果
void Map::dropEvent(QGraphicsSceneDragDropEvent *event)
{
    dragOver = false;
    if (event->mimeData()->hasText())
    {
        QString s = event->mimeData()->text();
        QPointF pos = mapToScene(event->pos());
        pos.setX((int(pos.x()) - 249) / 82 * 82 + 290);
        pos.setY((int(pos.y()) - 81) / 98 * 98 + 130);
        if (s == "Shovel")
        {
            Shovel *shovel = qgraphicsitem_cast<Shovel *>(scene()->items(QPointF(830, 15))[0]);
            shovel->removeYuanshen(pos);
        }
        else
        {
            Shop *shop = qgraphicsitem_cast<Shop *>(scene()->items(QPointF(300, 15))[0]);
            shop->addPlant(s, pos);
        }
    }
    update();
}
```

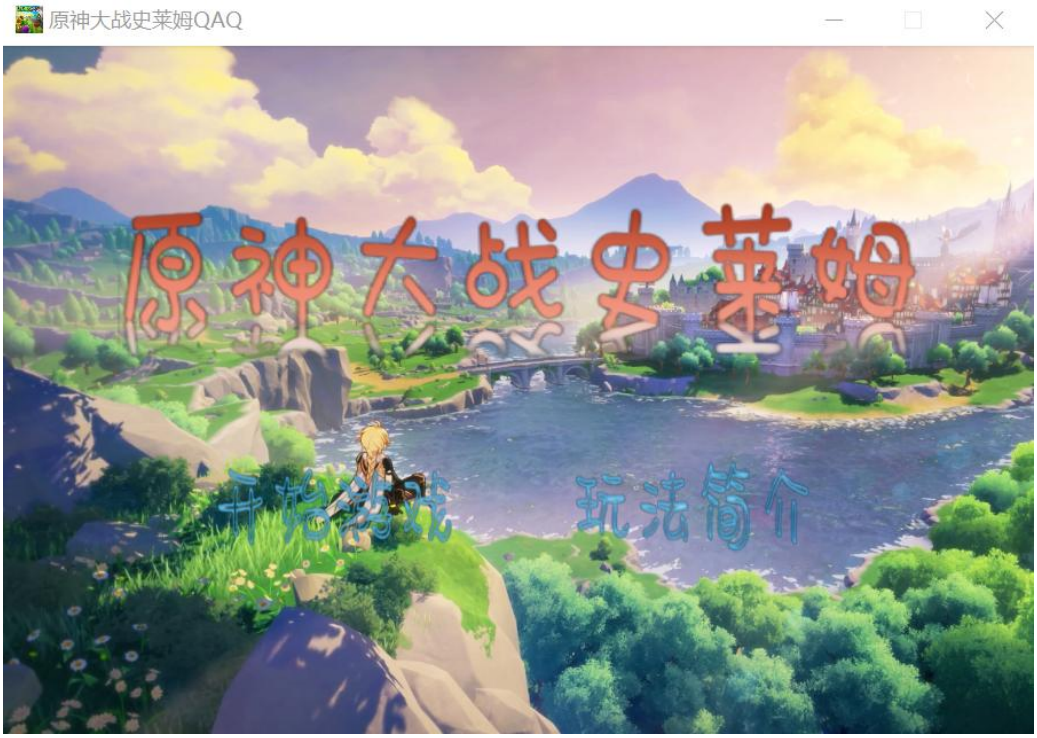
4.6 Button

Button是控制游戏暂停和继续的按钮，需要重写鼠标的按下事件

```
//绘制按钮
void Button::paint(QPainter *painter, const QStyleOptionGraphicsItem *option, QWidget *widget)
{
    Q_UNUSED(option)
    Q_UNUSED(widget)

    //设置按钮图片
    painter->drawPixmap(QRect(-80, -20, 160, 40), QPixmap(":/images/Button.png"));
    painter->setPen(Qt::green);

    //设置文本格式
    QFont font("Calibri", 18, QFont::Bold, true);
    painter->setFont(font);

    if (timer->isActive())
        painter->drawText(boundingRect(), Qt::AlignCenter, "PAUSE!");
    else
        painter->drawText(boundingRect(), Qt::AlignCenter, "CONTINUE");
}

//鼠标左键点击事件
void Button::mousePressEvent(QGraphicsSceneMouseEvent *event)
{
    if (event->button() == Qt::LeftButton)
    {
        if (timer->isActive())
        {
            sound->stop();
            timer->stop();
        }
        else
        {
            sound->play();
            timer->start();
        }
    }
    update();
}
```

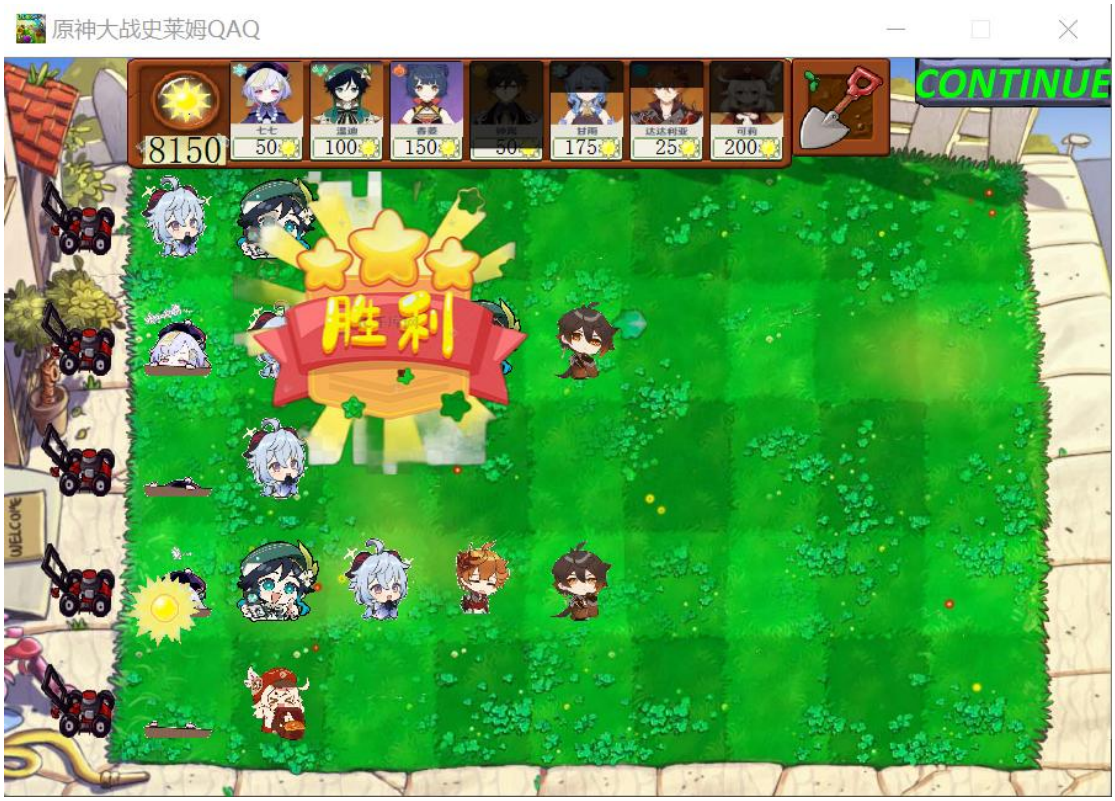
5. 运行与测试

游戏开始界面：



玩法简介窗口：

游戏胜利：


原神大战史莱姆QAQ
CONTINUE
8150
50
100
150
50
175
25
200
胜利

游戏失败：


CONTINUE
7750
50
100
150
50
175
25
200
GAME OVER

6. 问题与解决方案

6.1 未使用ui设计

未使用ui设计，则需要了解每个物体的坐标，反复调试；同时图片资源的插入也变得较为复杂。

6.2 GIF动图不够精美

由于自己不会绘画，GIF动图都是从米游社等找到的资源，所以在部分地方没有很好地展现动作，如人物的攻击动作，史莱姆的攻击动作，史莱姆的行走动作以及被炸弹炸死的状态

6.3 游戏运行

游戏有时候会有部分的卡顿，没有真正的植物大战僵尸那么的流畅

6.4 一些已经解决的问题以及收获

- ✓ 音频的输出，在工程文件中添加QT+=multimedia,使用Qsound，并且此时的音频文件必须是.wav文件
- ✓ override 可以重载虚函数
- ✓ 对于本学期所学的类的继承与派生，重载函数有了更深刻的理解

7.总结与反思

通过完成的这次的图形化小程序的大作业，我在短短的一个月内对于QT编程以及对于本学期所学的面向对象的思想有了更为深刻的了解，同时也对于游戏开发的基本思路有了一个较为基本的认识。

由于完成的时间有限，游戏还有很多的地方需要改进，比如不同关卡的设计，场景的转换等等。

然后想要做这个游戏的初衷是因为我在确定选题的时候在b站上看到了一个up主用unity制作的一个小游戏《原神大战丘丘人》。因为这个把我很喜欢的两个游戏结合起来，我感觉会比较好玩，所以就确定了这个游戏的大致方向。不过还有很多地方需要继续改进完善。

链接在这：https://www.ameloyi.top/GAME/Genshin/