大作业报告——HUNGRY HERO

学号：2019202458
姓名：周昊男

一．游戏概述

- 名称：HUNGRY HERO
- 开发工具：Qt
- 是否基于他人程序开发（若是，请详细说明你所做的改动）
  是，该塔防游戏是根据 csdn 上三篇连载教程写的，
  原教程地址：https://blog.csdn.net/satanzw/article/details/10418063
  改动：
  由于教程并不完整，所以基本功能的实现（加载地图、建塔、装填子弹攻击敌人）均是在阅读了教程之后自己补充相关逻辑并完成的。
  除此之外，原作只实现了一种塔、一种敌人、同一难度、不能选关、无拆除和升级功能、没有声音，我的程序在原作的基础上实现了前述所有功能，并对原项目的部分逻辑做了优化（如原作在关联子弹与敌人是有多余的函数，这在我的两次 changelog 中都有所体现；再比如原作移除敌人的逻辑无法实现中毒后死亡的效果，将会出现负血值但是敌人并不消失的 bug，我修改了相关逻辑来修复了这个 bug）。
  此外，原作使用了 plistreader 来实现对关卡信息、塔坑位置的加载，因为时间限制的原因这一部分我并没有实现，而是以简单的 Loadwave 函数和 QList 容器的 push_back 方法来简单实现的。
  最后，在我的代码中对修改的部分和大量原创的部分都做了详细的注释。
- 项目 Github 地址
  https://github.com/Harryhut0709/Towerdefense2020.1
- 基本功能介绍 （一百字以内）

1. 通过欢迎界面选择简单或困难的关卡，游戏胜利或结束后退回主界面重新选关。
2. 通过游戏下方的选塔区域选择普通/冰冻/中毒塔，实现对敌人不同类型的攻击。
3. 通过塔右边的按钮实现拆除/升级功能。
4. 多种属性能力不同的敌人，游戏难度随着游戏的进行不断加大。
5. 通过页面上方的玩家信息提醒金币/剩余血量/怪物波数的信息。

二．游戏视频链接

https://www.bilibili.com/video/BV1AZ4y1p7X5（视频平台：bilibili）

三. 类设计与 uml 图

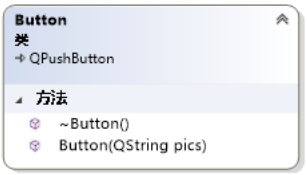
3.1 类 Mainwindow

关卡类，基类 Mainwindow 实现基本功能，简单关卡（Mainwindow1）和困难关卡（Mainwindow2）均继承于 Mainwindow，通过构造函数填充不同的关卡信息，从而实现特定的功能。

3.2 类 Scene

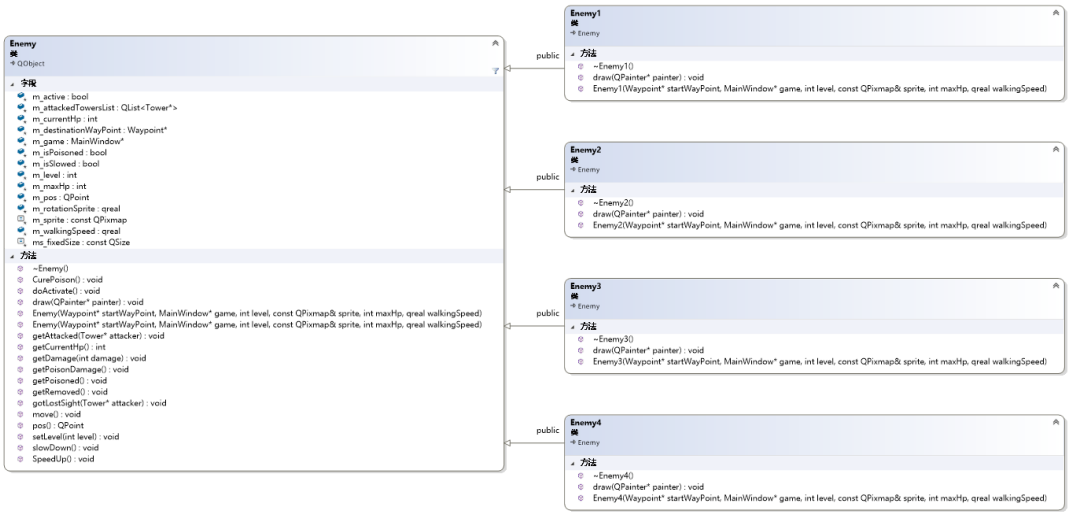
欢迎界面，通过 QPushButton 与 connect 函数实现选关和回到主界面的功能。

3.3 类 Button



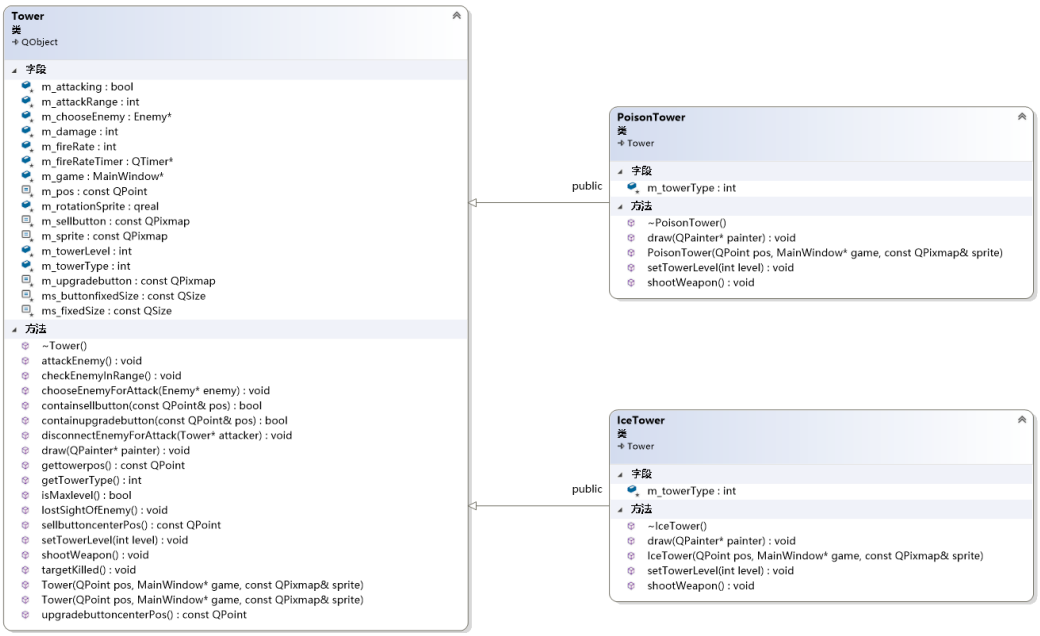
按钮类，初始化按钮的大小和形状，在 Scene 和 Mainwindow 类中均有用到，分别实现关卡选择、攻击塔选择、返回主界面的功能。

3.4 类 Enemy



不同类型敌人 Enemy1，Enemy2（行进速度快且对寒冰减速效应有较强抗性），Enemy3（对中毒效果有较强抗性），Enemy4（行进速度慢，血量高）均继承自 Enemy，随着游戏进行将会出现越来越强的敌人，增加游戏难度。

3.5 类 Tower

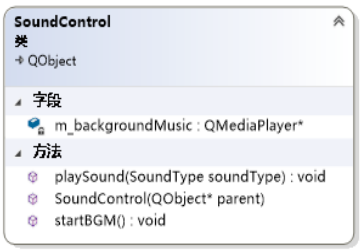


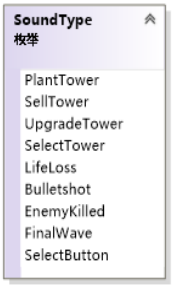
特殊炮塔 PoisonTower（中毒伤害持续减血），IceTower（冰冻减速）均继承于 Tower（普通塔）。且可对炮塔进行升级，以增强炮塔的攻击能力。同时也可以对炮塔进行拆除得到一定的玩家经济补偿。不同炮塔发射子弹击中敌人时具有不同特效。

3.6 类 Bullet

IceBullet，PoisonBullet 均继承于 Bullet，配合相应炮塔实现对敌人的攻击；且击中敌人时各自带有不同特效、对敌人造成不同类型的伤害。

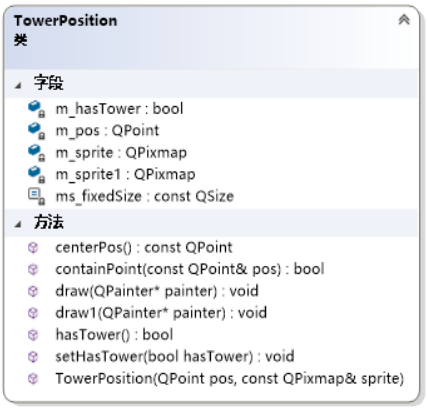
3.7 类 SoundControl





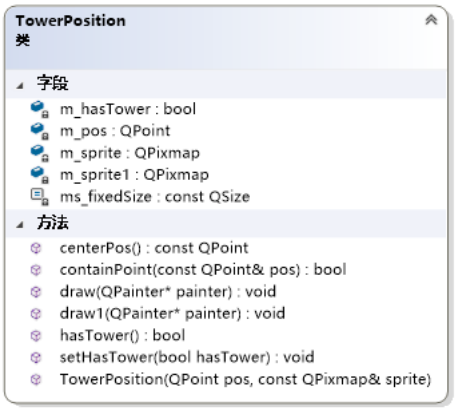
实现播放背景音乐；实现放置炮塔、敌人受到子弹攻击、敌人下线、选择不同类型的攻击塔、建塔、升级/拆除塔、最后一波怪物进攻时播放不同音效。

3.8 类 WayPoint



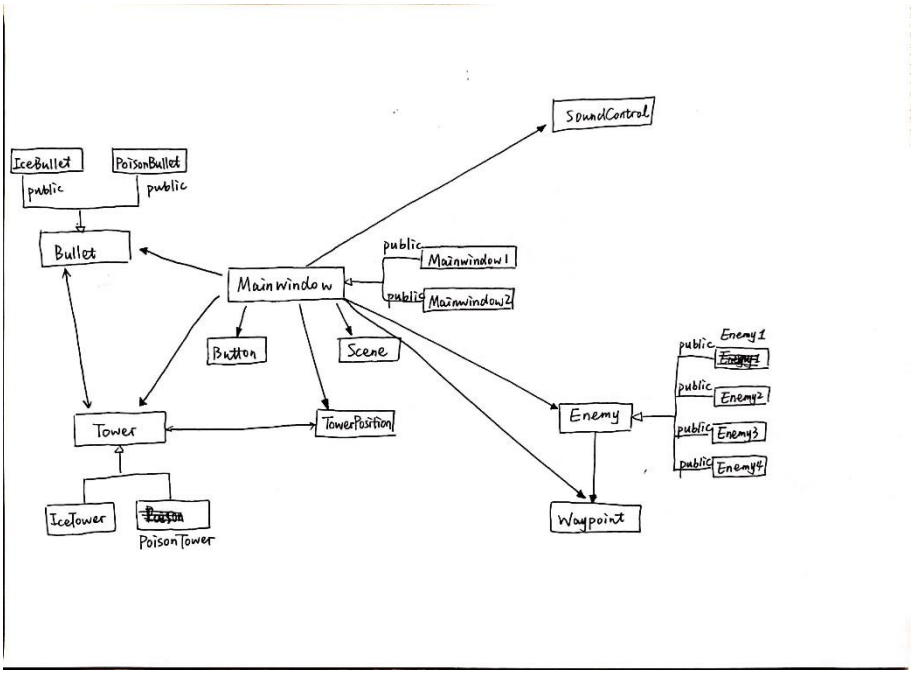
实现在不同场景下对不同敌人航点的设置。

3.9 类 TowerPosition



加载塔的初始位置。

Uml 图:



四. 核心代码说明

本部分贴出拆塔和升级功能实现的代码。

选择拆塔和升级功能作为核心代码是因为这部分我想了比较长的时间，一直被只能拆掉/升级最后一座塔，否则程序异常结束这一个 bug 所困扰。后来经过仔细思考发现了问题所在。

难点：m_towerPositionsList 和 m_towersList 是两个互相独立的 QList，如果使用双重循环实现拆塔功能时没有判断 towerPositions 和 towers 的中心是否重叠，则会出现指针指飞的情况，从而导致程序崩溃。

- 拆塔和升级的代码实现

```
void MainWindow::mousePressEvent(QMouseEvent * event)
{
    QPoint pressPos = event->pos();
    auto it = m_towerPositionsList.begin();
    while (it != m_towerPositionsList.end())
    {
        foreach(Tower *attacker,m_towersList)
            //相当于是双重循环,注意towersList和towerpositionsList是两个独立的容器,所以在拆塔的时候需要判断位置是否重合
        {
            //拆塔
            if (attacker->containsellbutton(pressPos) && it->hasTower() && it->centerPos()==attacker->gettowerpos() )
                //点到按钮上 有塔 拆的塔和坑是对应的
            {
                m_soundcontrol->playSound(SellTower);//声音
                m_playerGold += TowerSell;//加500金币
                //取消敌人与攻击塔的关联
                attacker->disconnectEnemyForAttack(attacker);
                m_towersList.removeOne(attacker);
                delete attacker;
                it->setHasTower(false);//这里没有塔了
                update();
                break;
            }//拆塔的逻辑有问题 //会导致闪退 //已解决
        }
    }
    //++ 升级功能
    foreach(Tower *attacker,m_towersList)
    {
        //升级
        if (m_playerGold >= TowerUpgradeCost && !attacker->isMaxlevel()
                && attacker->containupgradebutton(pressPos))//钱够&&未满级&&点到按钮上
        {
            m_soundcontrol->playSound(UpgradeTower);//声音
            m_playerGold -= TowerUpgradeCost;
            attacker->setTowerLevel(2);
            update();
            break;
        }
    }
}
```

这一部分的实现是受建塔功能的启发完成的，最开始本来想用的是封装好的 Button 类与每个 Tower 相关联，通过 connect 与 lambda 函数的配合使用完成对攻击塔的拆除和升级，但在实际操作时发现 Button 与 Tower 相关联后无法在屏幕上显示出来；于是我换了一个解决问题的思路，即把拆塔和升级的两个按钮作为图片现在 Tower 的 draw 函数中实现，再通过与 mousePressEvent 的配合完成对这两个功能的实现。

```
void Tower::draw(QPainter *painter) const
{
    //由于要改变painter的一些设置，所以加上save/restore来配合使用
    painter->save();
    painter->setPen(Qt::white);
    //绘制攻击范围
    painter->drawEllipse(m_pos,m_attackRange,m_attackRange);

    //绘制偏转坐标，由中心+偏移=左上
    static const QPoint offsetPoint(-ms_fixedSize.width()/2, -ms_fixedSize.height()/2-15);
    static const QPoint upgradeoffsetpoint(10,-25);//升级按钮偏移量 //+
    static const QPoint selloffsetpoint(10,5);//拆除按钮偏移量 //+

    //绘制并选择炮塔
    //这里将坐标原点移动到m_pos
    painter->translate(m_pos);
    painter->drawPixmap(upgradeoffsetpoint,m_upgradebutton);//画升级按钮 //+
    painter->drawPixmap(selloffsetpoint,m_sellbutton);//画拆塔按钮 //+
    //这两步需要在rotate之前，否则按钮会跟着塔一起旋转
    painter->rotate(m_rotationSprite+180);//
    painter->drawPixmap(offsetPoint,m_sprite);
    painter->restore();
}
```

（这一部分是对 Tower 的绘画的处理，在倒数第四、五行先实现了将两个按钮与攻击塔绑定，并保证按钮不会和攻击塔一起旋转）

然后，在 mousePressEvent 函数中的 auto 变量和 foreach 函数实质上是两个循环的嵌套，前者遍历 m_towerPositionsList 中所有塔坑的位置，后者遍历 m_towersList 中所有建好

的攻击塔，需要注意的是 m_towerPositionsList 中的成员（QPoint 类型）在关卡执行构造函数的时候就已经通过 loadTowerPositions 这个函数所固定了，并且它们的顺序也是固定不变的；然而 m_towersList 则是一个动态的容器（建塔的顺序与塔坑的顺序没有关联，拆塔和建塔实际上是在 QList 这个容器中删除和添加数据成员）。所以，在拆塔的时候如果没有判断 attacker 和 it 所分别对应的塔和塔坑中心是否重合，则默认执行的是拆掉之前所建的最后一座塔，所以无法完成按照任意顺序拆塔的操作，会因为指针问题导致程序异常结束。

还需要注意的一点是在拆塔是需要解除塔与正在攻击的敌人（如果有）的关联，这是通过 disconnectEnemyForAttack 这个函数实现的，如果没有这个函数，同样会因为指针问题导致程序异常结束。值得一提的是，这个函数和 lostSightOfEnemy 的逻辑很像，但是有细微的差别，如果没有注意到这个差别，把两者混为一谈，同样会因为指针问题导致程序异常结束。在解决这个 bug 的同时还发现了 lostSightOfEnemy 这个函数中的逻辑漏洞，上述在贴出的代码注释中都有体现。

```
void Tower::lostSightOfEnemy()//#
{
    //敌人跑出攻击塔的攻击范围，需要取消塔与敌人的关联，同时停止攻击
    m_chooseEnemy->gotLostSight(this);
//    if (m_chooseEnemy)//这句话是多余的，后面的disconnectEnemyForAttack才需要这个函数
    m_chooseEnemy = NULL;
    m_fireRateTimer->stop();
    m_rotationSprite = 0.0;//炮塔归位
}

void Tower::disconnectEnemyForAttack(Tower *attacker)
{
    if (m_chooseEnemy != NULL)//此时与敌人相连
    {
        m_chooseEnemy->gotLostSight(attacker);//这个函数其实是取消攻击塔和敌人的关联
        m_chooseEnemy=NULL;//空指针
        m_fireRateTimer->stop();
        m_rotationSprite = 0.0;
    }
}//这个函数和getlostsight的逻辑很像，但有一点不同，就是这是主动把敌人移出攻击范围的
```

五. 感（tu）想（cao）

最开始拿到这个大作业的时候我的内心是崩溃的：和上课的内容差得不知道有多远，老师给的 QTMap 示例代码实现的是 RPG 游戏，跟塔防游戏也差距很大，完全无从下手。后来我找到了统院大二我的一位直系师姐，她告诉我 csdn 上有一个比较详细的教程，先阅读、模仿，按着教程把建塔、攻击、敌人移动、敌人受伤等关键逻辑搞清楚再尝试着写出更多的功能————于是我的大作业艰难的开始了，但最开始也是不求甚解、囫囵吞枣，教程的逻辑不是完整的，而是有跳步，所以常常会因为教程里缺一个函数但我并没有发现而导致各种各样“出不来”的问题——塔坑出不来、敌人出不来、子弹出不来…但逐渐到编程后期，当我在实现原作中没有的功能（拆塔升级、中毒冰冻、多关卡选择时）我才真正开始慢慢理解原作中代码的逻辑和关联，不仅发现了原作中冗余的逻辑和函数，还根据自己的需要修改了相关的关键函数，并且在整个过程中我也对虚函数有了更深刻的理解、对指针指空有了更加深刻的认识（笑）。总而言之，这个大作业花费我大量的时间或许并不与课程的 2 学分相匹配，但每次 debug 成功和最后呈现出这样一个相对完整的作品给我带来的成就感也的确是无与伦比的。