Changelog0625

周昊男 2019202458

（注：这个 changelog 按照每个.h 头文件和.cpp 资源文件的顺序来记录这个版本相对上个版本的改动）

一、新增功能和优化

- 为攻击塔添加了升级和拆除按钮，这两个功能成功实现
- 改造了 Mainwindow 类，实现了两个关卡的选择和切换，可在游戏开始界面选择关卡的难易并进入相应的关卡
- 引入了另外四种属性不同的敌人，增加了游戏体验，且游戏难度随着游戏进行逐渐增加
- 为每种敌人添加了被冰冻速度减慢和中毒造成持续伤害的特殊效果
- 优化了玩家经济和怪物信息系统，使得玩家可以更好地在相关位置看到相关信息

二、修复的 bug

- 解决了上一个版本中毒敌人无法及时移除、血量会变成负的问题，修改了移除判定规则
- 解决了两个子弹同时击中敌人可能导致闪退的问题
- 解决了敌人可能卡在某个航点不动的问题

三、具体改动内容

（注：增加用+表示，删除用-表示，修改用#表示）

（注：由于我在写程序的过程中相较于第二版程序有变动的地方都作了//++（//## //--）标识，而函数具体的改动实在是太繁琐了，故在这里只体现了做了改动或添加的函数的名称，具体实现的修改可通过程序中//++ //## //--标识具体看到）

（注：在介绍完每个类的函数变动之后在最后说明了这次修改/增加做改动的关键函数及其具体实现）

Bullet.h 无变动

+ void PoisonBullet::hitTarget()//扣血的同时造成中毒伤害

Button.h/Button.cpp 无变动

Enemy.h

新增了 Enemy2, Enemy3, Enemy4, Enemy5，均公有继承于 Enemy 类，在构造函数中实现对怪物属性的不同赋值。

```
class Enemy2:public Enemy
{
    Q_OBJECT//所有应用 QT 槽的类都需要声明
public:
    Enemy2(Waypoint *startWayPoint, MainWindow *game,int level = 1,
           const QPixmap &sprite = QPixmap(":/image/enemy3.png"),
           int maxHp = 40,qreal walkingSpeed = 2.0);
    void draw(QPainter *painter) const;        //画怪兽  //++
    ~Enemy2();
};

class Enemy3:public Enemy
```

```
{
    Q_OBJECT//所有应用 QT 槽的类都需要声明
public:
    Enemy3(Waypoint *startWayPoint, MainWindow *game,int level = 1,
           const QPixmap &sprite = QPixmap(":/image/enemy4.png"),
           int maxHp = 40,qreal walkingSpeed = 2.0);
    void draw(QPainter *painter) const;        //画怪兽  //++
    ~Enemy3();
};

class Enemy4:public Enemy
{
    Q_OBJECT//所有应用 QT 槽的类都需要声明
public:
    Enemy4(Waypoint *startWayPoint, MainWindow *game,int level = 1,
           const QPixmap &sprite = QPixmap(":/image/enemy5.png"),
           int maxHp = 40,qreal walkingSpeed = 2.0);
    void draw(QPainter *painter) const;        //画怪兽  //++
    ~Enemy4();
};
```

Mainwindow.h

因为新建了一个关卡，所以这个类有较大的的改动。基类 Mainwindow 现在只具有每个关卡共有的成员函数（需要重载的以虚函数实现，并且每个关卡不同的信息：如塔的放置位置、怪物信息、航点信息等函数以给空实现，两个关卡分别重载这些函数给出不同的关卡信息；共有的判定逻辑和功能则不是虚函数：如鼠标点击事件、怪物到达基地扣血、一处敌人、移除子弹、奖励金币、可买塔/升级等函数）和数据成员（如波数、金币、血量信息；攻击塔位、攻击塔、航点、敌人、子弹列表等）。

两个派生类 Mainwindow1 和 Mainwindow2 均公有继承于 Mainwindow 类，关卡信息函数重载，放在关卡构造函数中调用。第二个关卡因为有两条怪物前进路线所以增加了新的方法 addWayPoints1()和数据成员 m_waypointsList1 分别管理两套航点。

```
class MainWindow : public QMainWindow
{
    Q_OBJECT//所有应用 QT 槽的类都需要声明

public:
    virtual void paintEvent(QPaintEvent *);///
    explicit MainWindow(QWidget *parent = 0);

    void getHpDamage(int damage = 1);                    //敌方进入基地，扣血
    void removedEnemy(Enemy *enemy);                     //敌方被打死，消失
    void removedBullet(Bullet *bullet);                  //子弹击中敌人后，消失(移除)
    void addBullet(Bullet *bullet);                      //装填新的子弹
    void awardGold(int gold);                            //打死一个敌人之后奖励金币
```

```
    void mousePressEvent(QMouseEvent *);                    //鼠标点击触发事件
    bool canBuyTower() const;                               //判断是否能够买新的塔
    bool canUpgradeTower() const;                           //++ //判断是否能升级
    virtual void drawWave(QPainter *painter);///
    virtual void drawHP(QPainter *painter);///
    virtual void drawPlayerGold(QPainter *painter);///          //这三个函数在顶顶上显
示当前信息，随时更新，暂时先以丑陋的文字实现
    virtual void drawOtherTexts(QPainter *painter);///          //+ //其他文字信息

    SoundControl * soundcontrol() const;                    //+
    QList<Enemy *> enemyList() const;                       //游戏中涉及到的怪物以
QList 形式存储

    ~MainWindow();

    //##  全部修改成了 public  用于继承
    virtual void loadTowerPositions();///                       //初始化塔的摆放位置
    virtual void addWayPoints();///                             //初始化怪物的行走
路线（航点）
    virtual void loadTowerSelectButtons();///                   //+ //初始化选塔按钮
    virtual void loadEndButtons();                              //++ //结束按钮
    virtual bool loadWave();///                                 //初始化波数信息

    void doGameOver();                                      //游戏结束，播放胜利或者
失败动画，未实现

public slots://QT 的 private 槽当前类及其子类可以将信号与之相连接
    void updateMap();                                       //30ms 刷新一次，模拟动画
帧数

protected:
    Ui::MainWindow *ui;

    int                 m_currentTowerMode=1;           //+ //通过这个变量来确定安
放的不同塔

    int                                                 m_waves;
//波数
    int                                                 m_playerHp;
//基地血量
    int                                                 m_playerGold;
//玩家金币
```

```
    bool                    m_gameStart;            //++游戏是否开始或者暂停
    bool                    m_gameEnded;
    bool                    m_gameWin;

    SoundControl *          m_soundcontrol;         //+

    QList<TowerPosition>    m_towerPositionsList;
    QList<Tower *>          m_towersList;
    QList<Waypoint *>       m_waypointsList;
    QList<Enemy *>                      m_enemyList;
    QList<Bullet *>             m_bulletList;

};

class MainWindow1:public MainWindow//第一关
{
    Q_OBJECT//所有应用 QT 槽的类都需要声明

public:
    virtual void paintEvent(QPaintEvent *);
    explicit MainWindow1(QWidget *parent = 0);

    void drawWave(QPainter *painter);
    void drawHP(QPainter *painter);
    void drawPlayerGold(QPainter *painter);         //这三个函数在顶顶上显示当前
信息，随时更新，暂时先以丑陋的文字实现
    void drawOtherTexts(QPainter *painter);         //+ //其他文字信息
    //位置改变需要重载

    SoundControl * soundcontrol() const;            //+
    QList<Enemy *> enemyList() const;               //游戏中涉及到的怪物以
QList 形式存储

    ~MainWindow1();

    //#  全部修改成了 public  用于继承
    void loadTowerPositions();                      //初始化塔的摆放位置
    void addWayPoints();                            //初始化怪物的行走路线（航
点）
    void loadTowerSelectButtons();                  //+ //初始化选塔按钮
    void loadEndButtons();                          //++ //结束按钮
    bool loadWave();                                //初始化波数信息
};
```

```
class MainWindow2:public MainWindow//第二关
{
    Q_OBJECT//所有应用 QT 槽的类都需要声明

public:
    virtual void paintEvent(QPaintEvent *);
    explicit MainWindow2(QWidget *parent = 0);

    void drawWave(QPainter *painter);
    void drawHP(QPainter *painter);
    void drawPlayerGold(QPainter *painter);                    //这三个函数在顶顶上显示当前
信息，随时更新，暂时先以丑陋的文字实现
    void drawOtherTexts(QPainter *painter);                    //+ //其他文字信息
    //位置改变需要重载

    SoundControl * soundcontrol() const;                        //+
    QList<Enemy *> enemyList() const;                           //游戏中涉及到的怪物以
QList 形式存储

    ~MainWindow2();

    //#  全部修改成了 public  用于继承
    void loadTowerPositions();                                  //初始化塔的摆放位置
    void addWayPoints();                                        //初始化怪物的行走路线（航
点）
    void addWayPoints1();                                       //初始化第二条路线  //++
    void loadTowerSelectButtons();                              //+ //初始化选塔按钮
    void loadEndButtons();                                      //++ //结束按钮
    bool loadWave();                                            //初始化波数信息

private:
    QList<Waypoint *>           m_waypointsList1;               //第二条路线
};
```

Scene.h 欢迎界面

构造函数和画图函数均做了修改，新添了选关的 button 和关卡缩略图、以及游戏名字

```
## explicit Scene(QWidget *parent = nullptr);
## void paintEvent(QPaintEvent *);
```

SoundControl.h

新增了两种音效

Tower.h

为实现卖塔和升级新增了三个函数

```
class Tower : public QObject
{
++ const QPoint gettowerpos() const;                    //++ //拆塔的时候需要判断中心点是否重合
++ virtual void setTowerLevel(int level);                //++ 升级并且更新能力参数
++ virtual void disconnectEnemyForAttack(Tower *attacker);    //++ //拆塔后取消与敌人的关联
}

class IceTower:public Tower{
++ void setTowerLevel(int level);//++ //塔的属性不同，所以需要重载
}

class PoisonTower:public Tower{
++ void setTowerLevel(int level);//++ //塔的属性不同，所以需要重载
}
```

Towerposition.h

第二关和第一关坑的样子不一样，故新增了一个 draw1()函数

```
++ void draw1(QPainter *painter) const;                //++ //第二关画坑坑（因为坑坑的样子不一样）
```

Utility.h 未修改

Waypoint.h 未修改

这次修改/增加做改动的关键函数及其具体实现

//这三个函数修改了敌人移除的逻辑，修复了敌人中毒血量扣为负值但却未被移除的 **bug**

```
void Enemy::getDamage(int damage)
{
    m_currentHp -= damage;//扣减敌人收到的伤害

    if (m_isPoisoned)//+
    {
        QTimer *stoptimer = new QTimer();//这个用来控制结束时间
        QTimer *poisontimer = new QTimer();
        poisontimer->setInterval(1000);//每次中毒伤害时间
        stoptimer->setInterval(5000);//持续中毒时间
        stoptimer->setSingleShot(true);//只触发一次
        connect(poisontimer,SIGNAL(timeout()),this,SLOT(getPoisonDamage()));
        connect(stoptimer,SIGNAL(timeout()),this,SLOT(CurePoison()));
        poisontimer->start();
        stoptimer->start();
    }
```

```
    // 阵亡,需要移除
    if (m_currentHp <= 0)
    {
        m_game->awardGold(200);//打死一个敌人奖励金币
        getRemoved();//敌人消失
    }
}

void Enemy::getRemoved()//++ //加上了中毒效果，此处需要修改逻辑 //好像并没有找到正确的逻辑解决中毒 bug... //++解决了!
{
//    if (m_attackedTowersList.empty() && m_currentHp>0 )
//        return;

//    if (m_currentHp<=0)
//    {
        foreach (Tower *attacker, m_attackedTowersList)
            attacker->targetKilled();
        // 通知 game,此敌人已经阵亡
        m_game->removedEnemy(this);
//    }
}

void Enemy::getPoisonDamage()//+ //中毒伤害 //槽函数
{
    m_currentHp -= 10;
    if (m_currentHp<=0)
    {
        foreach (Tower *attacker, m_attackedTowersList)
            attacker->targetKilled();
        // 通知 game,此敌人已经阵亡
        m_game->awardGold(200);
        m_game->removedEnemy(this);
    }//++ //解决了中毒效果会导致负血出现的问题
}

//玩家信息（血量、金币）、怪物波数显示系统均做了优化处理，调整了字体，新添了背景（由于逻辑和实现是一样的，故下面只给出了一个示例）
void MainWindow1::drawPlayerGold(QPainter *painter)
{
    painter->drawPixmap(215,5,QPixmap(":/image/goldframe.png"));
    QFont font;
    font.setPointSize(15);
```

```
        font.setFamily("Segoe Script");
        font.setLetterSpacing(QFont::AbsoluteSpacing,0);
        font.setBold(true);
        painter->setFont(font);
        painter->setPen(Qt::black);
        painter->drawText(275,50, QString("%1").arg(m_playerGold));
}
```

//这个函数新添了一个结束按钮，**connect** 后可在游戏进行过程中或者一个关卡结束后返回欢迎页面选择新的关卡

```
void MainWindow1::loadEndButtons()
{
        Button * endgame = new Button(":/image/endgame.png");
        endgame->setParent(this);
        endgame->setFixedSize(68,68);
        endgame->move(870,20);

        connect(endgame,&QPushButton::clicked,this,[=](){
                Scene *menu = new Scene;
                menu->show();
                this->hide();
                delete this;
        });
}
```

//鼠标点击事件中实现了拆塔和升级，因为试了另一种方法并未成功，也在此处以//注释的形式给出

```
void MainWindow::mousePressEvent(QMouseEvent * event)
{
        QPoint pressPos = event->pos();
        auto it = m_towerPositionsList.begin();
        while (it != m_towerPositionsList.end())
        {
                if (canBuyTower() && it->containPoint(pressPos) && !it->hasTower() &&
m_currentTowerMode==1)//+ //普通塔
                        //有钱 && 遍历所有安放位置，点在安放位置内 && 安放位置是空的
                {
                        m_playerGold -= TowerCost;//扣钱
                        it->setHasTower();//占坑
                        Tower *tower = new Tower (it->centerPos(),this);
                        m_soundcontrol->playSound(PlantTower);//+ //声音
                        m_towersList.push_back(tower);
                        update();
                        break;
```

```
        }

        if (canBuyTower() && it->containPoint(pressPos) && !it->hasTower() &&
m_currentTowerMode==2)//+ //冰冻塔
        {
            m_playerGold -= IceTowerCost;//扣钱
            it->setHasTower();//占坑
            IceTower *tower = new IceTower (it->centerPos(),this);
            m_soundcontrol->playSound(PlantTower);//+ //声音
            m_towersList.push_back(tower);
            update();
            break;
        }

        if (canBuyTower() && it->containPoint(pressPos) && !it->hasTower() &&
m_currentTowerMode==3)//+ //毒塔
        {
            m_playerGold -= PoisonTowerCost;//扣钱
            it->setHasTower();//占坑
            PoisonTower *tower = new PoisonTower (it->centerPos(),this);
            m_soundcontrol->playSound(PlantTower);//+ //声音
            m_towersList.push_back(tower);
            update();
            break;
        }

        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
            }

            ++it;//注意这句话的位置
        }

        //++ 升级功能
//      auto upgradetect = m_towersList.begin();
//      while (upgradetect != m_towersList.end())
//      {
//          if (canUpgradeTower() && (*upgradetect)->containupgradebutton(pressPos)
&& !(*upgradetect)->isMaxlevel())//足够金币 点击到了按钮上 未满级
//          {
//              m_playerGold -= TowerUpgradeCost;//扣钱
//              (*upgradetect)->setUpgradeTower();//升级
//              (*upgradetect)->setTowerLevel();//升级效果
//              m_soundcontrol->playSound(UpgradeTower);//声音
//              update();
//              break;
//          }
//          ++upgradetect;//这一步可能不需要 //去掉就闪退
//      }

        //++ 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;
            }

//          //拆塔
//          if (attacker->containsellbutton(pressPos)&&it->hasTower())
//          {
//              m_soundcontrol->playSound(SellTower);//声音
//              m_playerGold += TowerSell;//加 500 金币
//              m_towersList.removeOne(attacker);
```

```
//                  delete attacker;
//                  it->setHasTower(false);//这里没有塔了
//                  update();
//                  break;
//              }
        }
}
```