大作业的设计和实现

曹竞轩　2018202075

一、 个人分工

在大作业中，我作为组长，主要负责**组织工作**、**编写代码**两方面工作。

组织工作方面：

- 工作之初，我负责组织小组成员出来讨论，并确定了我们组大作业总体采用 QGraphics 架构来编写这个游戏。
- 工作中，给每个组员分配任务、验收成果、整合各组员代码。
- 工作最后，负责上台进行小组展示。

编写代码方面：

- 负责工程的最初搭建。新建了某几个基本类，写了它们所继承的 QT 内置类并引用了相应头文件，并在每个类的.h 文件中写了几个基本的成员变量和成员函数。
- 负责 Tower 类及其所有函数的设计和实现。
- 负责 Enemy 类的 void ***march***()函数的实现。
- 负责 Game 类的以下函数的设计和实现：

  Game()

  void **lostHp**(int damage)

  void **gameOver**()

  void **loadText**()

  void **setCursor**(QString filename)

  void ***mouseMoveEvent***(QMouseEvent *event)

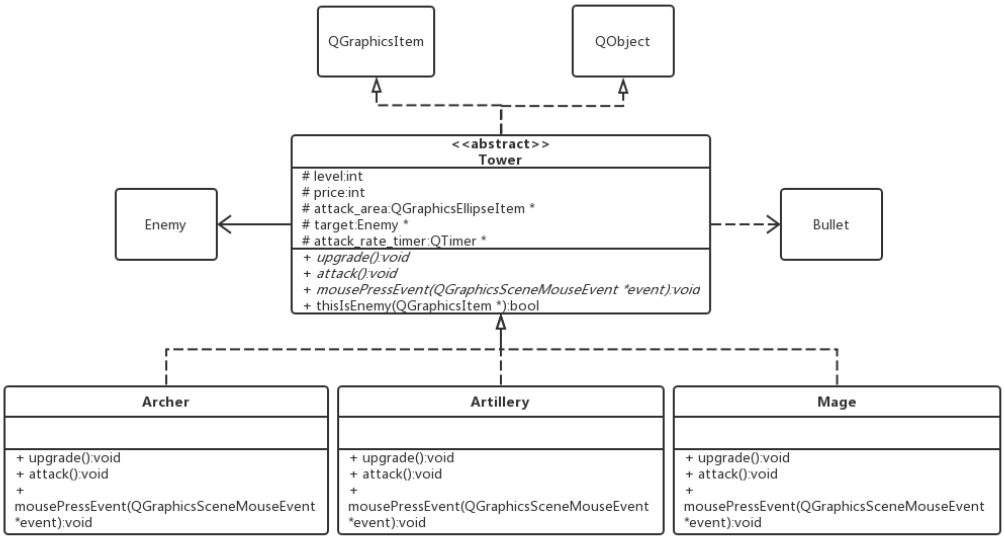
  void **removeItem**(QGraphicsItem *item)

  void **clearScene**(QGraphicsItem *pic)

  void **gameWin**()

  void **enterNextStage**()

二、 类图



图：Tower 类的 UML 类图

三、核心代码

1. void Tower::***attack***()（由于这是 Tower 类的一个纯虚函数，故以 Tower 的一个子类 Archer 中的此函数为例）

```
void Archer::attack() {
    Arrow *arrow = new Arrow(level);

    arrow->setPos(x() + 44, y() + 44);
    arrow->target = target;

    QLineF path(arrow->pos(), arrow->target->pos());
    arrow->setRotation(path.angle());

    game->scene->addItem(arrow);
}
```

塔攻击时，创建新 Bullet，并加入 game->scene 中。

2. void Tower::***upgrade***()（由于这是 Tower 类的一个纯虚函数，故以 Tower 的一个子类 Archer 中的此函数为例）

```
void Archer::upgrade() {
    level++;
    game->gold -= price;
    setPixmap(QPixmap(":/Resource/Tower/Archer/Archer_2.png"));
    QSound::play(":/Resource/Tower/Archer/Upgrade.wav");

    delete attack_area;
    QPointF adjust(190, 220);
    attack_area = new QGraphicsEllipseItem(QRectF(this->pos() - adjust, QSizeF(500, 500)));
    game->scene->addItem(attack_area);
    attack_rate_timer->start(480);
}
```

升级塔时，更新属性、切换图片并改变攻击范围。

3. void Tower::***mousePressEvent***(QGraphicsSceneMouseEvent *event) （由于这是 Tower 类的一个纯虚函数，故以 Tower 的一个子类 Archer 中的此函数为例）

```
void Archer::mousePressEvent(QGraphicsSceneMouseEvent *event) {
    if (this->contains(event->pos())) {
        if (event->button() == Qt::LeftButton) {
            if (level == 1 && game->gold >= 70) {
                upgrade();
            };
        } else if (event->button() == Qt::RightButton) {
            game->gold += price * 0.7;
            QSound::play(":/Resource/Tower/Tower_Sell.wav");
            delete attack_area;
            delete this;
        }
    }
}
```

左键点塔升级，右键点塔出售。

4. void Tower::**chooseEnemyToAttack**()

```
void Tower::chooseEnemyToAttack() {
    QList<QGraphicsItem *> items_found = attack_area->collidingItems();
    QList<Enemy *> enemies_found;

    foreach (QGraphicsItem *item, items_found) {
        if (thisIsEnemy(item) && dynamic_cast<Enemy *>(item)->died == false) {
            enemies_found.append(dynamic_cast<Enemy *>(item));
        }
    }

    if (enemies_found.isEmpty()) {
        target = nullptr;
        return;
    }

    if (!enemies_found.contains(target)) {
        target = enemies_found[0];
        for (int i = 1; i < enemies_found.size(); i++) {
            if (enemies_found[i]->getSpawnedOrder() < enemies_found[i - 1]->getSpawnedOrder()) {
                target = enemies_found[i];
            }
        }
    }

    attack();
}
```

用 QList<QGraphicsItem *> QGraphicsItem::collidingItems()函数来获取塔攻击范围内的所有 Item，并判断它们是否是 Enemy，攻击搜索到的 Enemy 中最早生成的那一个。

5. void Enemy::***march***()（由于这是 Enemy 类的一个纯虚函数，故以 Enemy 的一个子类 Goblin 中的此函数为例）

```
void Goblin::march() {
    QLineF path(pos(), next_point);

    if (path.length() < 5) {
        point_index++;
        if (next_point == way_points.back()) {
            march_timer->stop();
            game->lostHp(damage);
            QSound::play(":/Resource/Background/Lose_Life.wav");
            game->removeItem(this);
            return;
        }
        if (point_index < way_points.size() - 1) {
            next_point = way_points[point_index + 1];
        }
    }

    if (path.angle() >= 135 && path.angle() < 225) {
        setPixmap(QPixmap(":/Resource/Enemy/Goblin/Left.png"));
    } else if (path.angle() < 45 || path.angle() >= 315) {
        setPixmap(QPixmap(":/Resource/Enemy/Goblin/Right.png"));
    } else if (path.angle() >= 45 && path.angle() < 135) {
        setPixmap(QPixmap(":/Resource/Enemy/Goblin/Back.png"));
    } else {
        setPixmap(QPixmap(":/Resource/Enemy/Goblin/Front.png"));
    }

    double dy = (-1) * speed * qSin(qDegreesToRadians(path.angle()));
    double dx = speed * qCos(qDegreesToRadians(path.angle()));

    setPos(x() + dx, y() + dy);
}
```

敌人随移动方向改变显示的图片；到达一个目标点时切换目标点，到达最后一个目标点时移除自己并使玩家失去一定生命值。

6. void Game::**setCursor**(QString filename)

```
void Game::setCursor(QString filename) {
    if (cursor) {
        scene->removeItem(cursor);
        delete cursor;
    }

    cursor = new QGraphicsPixmapItem;
    cursor->setPixmap(QPixmap(filename));
    scene->addItem(cursor);
}
```

cursor 即玩家点击 TowerIcon 后随着鼠标移动而移动的那个待建造的塔。

7. void Game::***mouseMoveEvent***(QMouseEvent *event)

```
void Game::mouseMoveEvent(QMouseEvent *event) {
    if (cursor){
        cursor->setPos(event->pos());
    }
}
```

8. void Game::**loadText**()

```
void Game::loadText() {
    hp_ = new QGraphicsSimpleTextItem;
    gold_ = new QGraphicsSimpleTextItem;
    wave_ = new QGraphicsSimpleTextItem;

    hp_->setBrush(Qt::white);
    hp_->setFont(QFont("宋体",12));
    hp_->setPos(103, 46);
    gold_->setBrush(Qt::white);
    gold_->setFont(QFont("宋体",12));
    gold_->setPos(170, 46);
    wave_->setBrush(Qt::white);
    wave_->setFont(QFont("宋体",12));
    wave_->setPos(310, 46);

    QTimer *timer = new QTimer(this);
    connect(timer, SIGNAL(timeout()), this, SLOT(setText()));
    timer->start(10);

    scene->addItem(hp_);
    scene->addItem(gold_);
    scene->addItem(wave_);
}
```

加载游戏左上角显示的玩家生命值、金钱和当前波数。

9. void Game::**clearScene**(QGraphicsItem *pic)

```
void Game::clearScene(QGraphicsItem *pic) {
    delete inwave_timer;
    inwave_timer = nullptr;

    delete waves_timer;
    waves_timer = nullptr;

    delete game_win_timer;
    game_win_timer = nullptr;

    delete archericon;
    archericon = nullptr;

    delete artilleryicon;
    artilleryicon = nullptr;

    delete mageicon;
    mageicon = nullptr;

    delete build;
    build = nullptr;

    delete cursor;
    cursor = nullptr;

    delete BGM;
    BGM = nullptr;

    wave.clear();

    tower_positions.clear();

    QList<QGraphicsItem *> all_items = pic->collidingItems();

    foreach (QGraphicsItem *item, all_items) {
        scene->removeItem(item);
        item = nullptr;
    }
}
```

大作业展示中提到的“屠城函数”。

10. Game::**Game**()

```
Game::Game() {
    stage = 1;
    gold = 200;
    hp = 10;

    enemy_left = 18;
    num_of_waves = 4;

    Enemy_order=0;
    wave_order = 0;

    wave_enemy_number=0;

    scene = new QGraphicsScene(this);
    scene->setSceneRect(0, 0, 1280, 720);

    setScene(scene);

    cursor = nullptr;
    build = nullptr;
    setMouseTracking(true);

    QGraphicsPixmapItem *backgroundpic = new QGraphicsPixmapItem;
    backgroundpic->setPixmap(QPixmap(":/Resource/Background/Stage_1.png").
                             scaled(1280, 720, Qt::KeepAspectRatio, Qt::SmoothTransformation));
    scene->addItem(backgroundpic);

    BGM = new QSound(":/Resource/Background/BGM_1.wav");
    BGM->setLoops(-1);
    BGM->play();

    setFixedSize(1280, 720);
    setHorizontalScrollBarPolicy(Qt::ScrollBarAlwaysOff);
    setVerticalScrollBarPolicy(Qt::ScrollBarAlwaysOff);

    loadTowerPositions();
    loadWayPoints();
    loadIcon();
    loadText();

    Add_Waves();

    game_win_timer  = new QTimer;
    connect(game_win_timer, SIGNAL(timeout()), this, SLOT(gameWin()));
    game_win_timer->start(10);
}
```

Game 类的构造函数，加载游戏所需的各种信息并开始游戏。

11. void Game::**enterNextStage**()

```
void Game::enterNextStage() {
    stage = 2;
    gold = 200;
    hp = 10;

    enemy_left = 25;
    num_of_waves = 5;

    Enemy_order = 0;
    wave_order = 0;

    wave_enemy_number = 0;

    delete game_win_pic;
    game_win_pic = nullptr;

    cursor = nullptr;
    build = nullptr;
    setMouseTracking(true);

    QGraphicsPixmapItem *backgroundpic = new QGraphicsPixmapItem;
    backgroundpic->setPixmap(QPixmap(":/Resource/Background/Stage_2.png").
                             scaled(1280, 720, Qt::KeepAspectRatio, Qt::SmoothTransformation));
    scene->addItem(backgroundpic);

    BGM = new QSound(":/Resource/Background/BGM_2.wav");
    BGM->setLoops(-1);
    BGM->play();

    setFixedSize(1280, 720);
    setHorizontalScrollBarPolicy(Qt::ScrollBarAlwaysOff);
    setVerticalScrollBarPolicy(Qt::ScrollBarAlwaysOff);

    loadTowerPositions();
    loadWayPoints();
    loadIcon();
    loadText();

    Add_Waves();

    game_win_timer  = new QTimer;
    connect(game_win_timer, SIGNAL(timeout()), this, SLOT(gameWin()));
    game_win_timer->start(10);
}
```

第一关胜利后，加载第二关所需的各种信息并开始第二关。