程序设计实验报告

施尧恩2018200937

**负责工作（具体）：**

1. 塔防游戏的基础功能（有塔、有怪、能攻击）
2. 塔的分类、升级与动画效果
3. 玩家大招、动画效果与冷却时间
4. BGM与音效的播放
5. 一部分的组员代码整合

**心得体会：**

我性格急切，易于焦躁，总是希望事情很快就得出结果，于是面对经济学和数学这种需要好事多磨，不能随时检验成效的学科，总是有种人生长途漫漫，一眼望不着边界的感受。程序设计这门课带给了我不一样的感受，只要写出代码，经过编译便出现结果。如结果正确，那就是对的，那这一部分就相当确信无疑地掌握了；如果有问题，那立马再改，一定是知识存在漏洞。总之，会还是不会，理解还是没有理解，判断立决。

我回想起接到大作业任务的第一周，每天留出时间学习CSDN上的教程，在第一个星期里做好了基础框架。看到自己花费的努力与时间很快就有了结果，确实是自开学以来一成不变单调反复的日子里值得欣喜欢舞的时刻，也是满足了所学见用的实用主义情结吧。

又记得一次星期六的晚上，一宿未眠到了天明，只是为了找出程序里的bug。当时效率当然是极低的，但那已经不是我所关心的事了。这个过程让人沉浸在其中，看到自己的作品在一点一滴臻于完善，确是自从大学以来长久未曾遇到的，值得去投入热情的事情啊。其中耗费了大把时间终于找到存在于一行中的bug，或者疏忽只是一个小小的疏忽导致了全盘崩溃（比如未初始化指针），说不出是欢喜雀跃，还是五味杂陈。

在这个学期结束之后，我们班就没有有关编程的专门课程了。我也没有什么动情的话可说，这里不论未来，只谈过去。但程序设计的确是这个学期投入时间最多的一门课，是2019上半年留待将来值得回顾的一段记忆。把游戏导出成了应用程序，尽管我时常笑称我们做的是90年代的街机游戏，但也是值得珍贵的事物了。

**核心代码：**

1. **塔的建造与升级：**

void MainWindow::keyPressEvent(QKeyEvent \*e)

{

if(e->key() == Qt::Key\_J ||e->key() == Qt::Key\_K ||e->key() == Qt::Key\_L )

{

//操控塔防放置

int xpos = this->playerme->getx();

int ypos = this->playerme->gety();//得到玩家所在位置的坐标

auto it = m\_towerPositionsList.begin();//准备循历一遍可放置塔的区域坐标

QPoint player\_position(xpos,ypos);//将玩家坐标整合为QPoint

while (it != m\_towerPositionsList.end())//开始循历

{

//如果①金额足够②点击位置在范围中③未有塔

if (canBuyTower()

&& it->containPoint(player\_position)

&& ! it->hasTower())

{

m\_gold -= TowerCost;

if(e->key() == Qt::Key\_J)//按下J键，建造炮塔

{

Tower \*tower = new Tower(it->centerPos(), this); //生成新塔对象

m\_towersList.push\_back(tower); //将新塔添入塔列表中

it->setHasTower(tower); //将塔位状态修改为已有塔

}

else if(e->key() == Qt::Key\_K)//按下K键，建造魔法塔

{

TowerMagic \*tower = new TowerMagic(it->centerPos(), this);

m\_towersList.push\_back(tower); //将新塔添入塔列表中

it->setHasTower(tower); //将塔位状态修改为已有塔

}

else if(e->key() == Qt::Key\_L)//按下L键，建造减速塔

{

TowerFreeze \*tower = new TowerFreeze(it->centerPos(), this);

m\_towersList.push\_back(tower); //将新塔添入塔列表中

it->setHasTower(tower); //将塔位状态修改为已有塔

}

update(); //更新mainwindow

break;//跳出循环

}

//如果：①金钱足够②已有塔③点击位置在范围中④塔可以升级

else if(canBuyTower()

&& it->hasTower()

&& it->containPoint(player\_position)

&& it->getTower()->canLevelUp())

{

m\_gold -= TowerCost;

it->getTower()->levelUp();

update(); //更新mainwindow

break; //跳出循环

}

it++;

}

}

1. **敌人的出动：**

bool MainWindow::loadWave()

{

//一共6波，每波波数\*2+6个敌人

if (m\_waves >= 6){return false;}

WayPoint \*startWayPoint = m\_wayPointsList.back(); // 航点QList是逆序的，末项是起点

int enemyStartInterval = 1000; //设置敌人出现时间间隔

for (int i = 0; i < m\_waves \* 2 + 6; i++)

{

QTimer::singleShot( i \* enemyStartInterval, this, SLOT(enemyAppear()));//每隔一段时间触发敌人出现的音效

if( i % 2 ==0 )//第奇数个是弱敌人

{

Enemy \*enemy = new Enemy(startWayPoint, this, QPixmap(":/pic/picture/enemy1.png"),40,1,100);//生成指向新敌人的指针

m\_enemyList.push\_back(enemy);//将指针加入敌人列表中

QTimer::singleShot( i \* enemyStartInterval, enemy, SLOT(doActivate()));

//singleShot相当于定时器，好处在于不需使用timer，在设置时间到后自动触发SLOT中的函数//每隔一段时间放出一个敌人

}

Else//第偶数个是强敌人

{

Enemy \*enemy = new Enemy(startWayPoint, this, QPixmap(":/pic/picture/enemy2.png"),80,2,200);

m\_enemyList.push\_back(enemy);

QTimer::singleShot( i \* enemyStartInterval, enemy, SLOT(doActivate()));

}

}

return true;

}

1. **敌人的移动：**

void Enemy::move(){

//①未激活，不移动

if (!m\_active) {return;}

//②如果抵达一个航点

if (collisionWithCircle(m\_pos, 1, m\_destinationWayPoint->pos(), 1)) //判断距离半径用1不用0，因为时差因素，不可能恰好坐标重合

{

// 如果还有下一个航点，继续走

if (m\_destinationWayPoint->nextWayPoint())

{

m\_pos = m\_destinationWayPoint->pos();//把当前位置改为碰到的航点位置

m\_destinationWayPoint = m\_destinationWayPoint->nextWayPoint();//把目标位置改为下一个航点位置

}

//③如果进入基地，城堡受到伤害，塔停止攻击，并且移除敌人

else

{

foreach (Tower \*tower, m\_attackedTowerList)//遍历攻击该敌人的所有塔的列表

{

tower->clearTargetEnemy();

tower->stopFire();

}

if(m\_game != NULL)//如果是第一关

{

m\_game->getHpDamage(m\_damage);

m\_game->removedEnemy(this); //移除该敌人

}

Else//如果是第二关

{

m\_game2->getHpDamage(m\_damage);

m\_game2->removedEnemy(this); //移除该敌人

}

return;

}

}

//④如果还在前往航点

QPoint targetPoint = m\_destinationWayPoint->pos(); // 目标航点的坐标

QVector2D normalized(targetPoint - m\_pos); // 这是由当前坐标指向目标航点的2D向量

normalized.normalize(); //进行向量标准化：normalized被修改为该2D向量方向上的**单位向量**

m\_pos = m\_pos + normalized.toPoint() \* m\_currentSpeed; //加上单位向量乘以移动速度，修改坐标，实现移动

}

1. **塔的攻击：**

void MainWindow::updateMap()

{

//一座塔一时只能攻击一个目标，一个敌人一时可被多个敌人攻击

//判断tower与enemy的距离

foreach (Tower \*tower, m\_towersList)//遍历游戏中所有已存在塔的列表

{

//如果tower没有攻击

if(tower->getTargetEnemy() == NULL)

{

foreach (Enemy \*enemy, m\_enemyList)//遍历游戏中所有已存在敌人的列表

{

if(ifInRange(tower->getPos(), tower->getAttackRange(), enemy->getPos(), 1))//如果有一个敌人在该塔攻击范围内

{

tower->beginAttackEnemy(enemy);//塔攻击该敌人

break;

}

}

}

//如果tower正在攻击

else

{

if( !ifInRange(tower->getPos(), tower->getAttackRange(), tower->getTargetEnemy()->getPos(), 1))//如果被该塔攻击的敌人超出了攻击范围

{

tower->loseEnemy();

}

}

}

1. **炮塔射出的子弹击中敌人造成伤害：**

void Bullet::hitTarget() //击中敌人时，是与animation->finished()关联的

{

// 可能多个炮弹击中敌人,而其中一个已将其消灭,导致敌人已被delete

// 因此先判要断敌人是否还在

if( m\_game != NULL )//如果是第一关

{

if ( m\_game->findEnemy(m\_targetEnemy) )//如果能找到该敌人

//Returns -1 if no item matched.

{ m\_targetEnemy->getDamaged(m\_damage); } //目标敌人受到伤害

m\_game->removedBullet(this); //令mainwindow删除子弹

}

else if( m\_game2 != NULL )//如果是第二关

{

if ( m\_game2->findEnemy(m\_targetEnemy) ) //如果能找到该敌人

{ m\_targetEnemy->getDamaged(m\_damage); } //目标敌人受到伤害

m\_game2->removedBullet(this); //令mainwindow删除子弹

}

}

1. **魔法塔的波纹的动画效果：**

void Ray::move()

{

static const int duration = 500; // 存放动画持续时间

QPropertyAnimation \*animation = new QPropertyAnimation(this, "m\_currentRadius");//创建一个新动画，选择的变量是当前波纹半径m\_currentRadius

animation->setDuration(duration);//设置持续时间

animation->setStartValue(0);//设置初值

animation->setEndValue(m\_targetRadius);//设置终值

connect(animation, SIGNAL(finished()),this,SLOT(hitTarget()));//关联动画结束animation->finished()与波纹造成攻击hitTarget()；当动画结束时，对敌人造成伤害

animation->start();//动画开始

}

1. **减速塔的波纹的减速效果：**

void MainWindow::iceEffect(Ice \*ice)

{

foreach(Enemy \*enemy, m\_enemyList)//遍历每一个敌人

{

if(ifInRange(enemy->getPos(),1,ice->getPos(),ice->getTargetRadius())

&& ! enemy->ifDecelerated())//在波纹范围之内

{

enemy->getDecelerated(ice->getDeceleration());//敌人受到减速

}

}

removeIce(ice);

}

void Enemy::getDecelerated(double deceleration)

{

m\_currentSpeed -= deceleration; //降低当前速度数值

m\_ifDecelerated = true;// m\_ifDecelerated的值联系draw()，会画出蓝色透明方块表示被减速

QTimer::singleShot( 3000, this, SLOT(recoverSpeed()));//3秒后恢复速度

}

1. **大招的冷却动画：**

void MainWindow::paintEvent(QPaintEvent \*)

{

//画施毒图标

if(m\_canPoison)//如果施毒可用，只画出一个图标

{pt->drawPixmap(930, 10, QPixmap(":/pic/picture/poisonbottle.png"));}

Else//如果施毒还在冷却中，改换图标，并且画出随时间变化的冷却条

{

pt->drawPixmap(930, 10, QPixmap(":/pic/picture/poisonbottle2.png"));

pt->save();

pt->setBrush(QColor(128,0,128));

pt->setOpacity(0.5);

pt->drawRect(930, 10, m\_iconWidth,100);// , m\_iconWid用动画QPropertyAnimation实现随时间变化

pt->restore();

}

1. **音乐的播放：**

//音效，触发才播放

m\_player.setMedia(QUrl("qrc:/pic/music/goldsound.mp3"));

m\_player.setVolume(30);

EA.setMedia(QUrl("qrc:/pic/music/enemyappear.mp3"));

EA.setVolume(30);

EK.setMedia(QUrl("qrc:/pic/music/enemykilled.mp3"));

EK.setVolume(45);

CD.setMedia(QUrl("qrc:/pic/music/CD.mp3"));

CD.setVolume(90);

HP.setMedia(QUrl("qrc:/pic/music/HP.mp3"));

HP.setVolume(45);

//背景音乐，初始化时就开始播放

bgmplayer.setMedia(QUrl("qrc:/pic/music/mw1bgm.mp3"));

bgmplayer.setVolume(15);

bgmplayer.play();

**遇到的问题：**

1. 遇到问题1：最大的困难——不会写游戏，毫无头绪，无从下手

如何解决1：找CSDN上的教程学习，每天攻克一点点

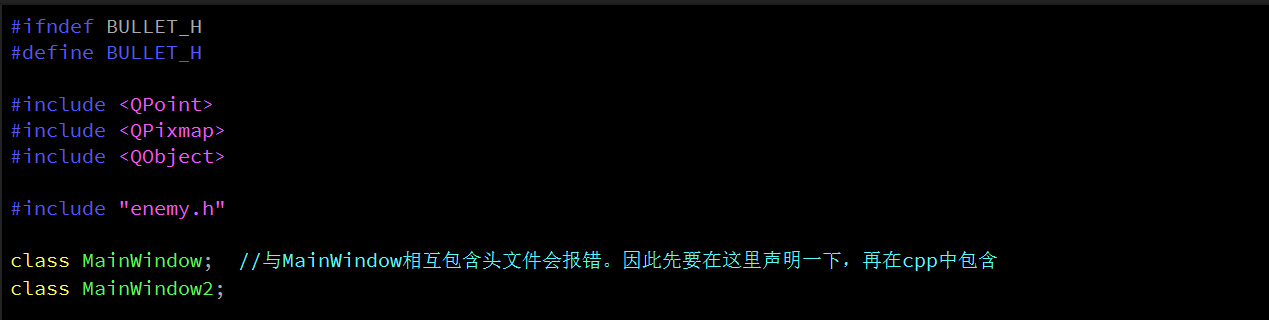
1. 遇到问题2：不能理解注册属性（QProperty），无法使用动画

如何解决2：翻阅Qt帮助，理解意义

1. 遇到问题3：两个头文件相互包含，导致类未定义

如何解决3：先在甲类的头文件中用class声明乙类，再在甲类的cpp文件里包含乙类的头文件即可，如：

Bullet.h



Bullet.cpp



1. 遇到问题4：防御塔无法攻击敌人（原因有很多）

如何解决4：用注释法，沿循逻辑发生的过程，一步步地找原因