****

2017-2018（2）

《C++程序开发实训》

课程报告

**题目：基于Qt视图框架的C++游戏开发**

**学 院 软件学院**

**专 业 软件工程**

**学生姓名 郭晓凡**

**学生学号 xxxxxxxxxxxx**

**指导教师 xxxxx**

**起始日期 2018年7月11日**

# 简介

1. 概述：

Qt是一个1991年由Qt Company开发的跨平台C++图形用户界面应用程序开发框架。它既可以开发GUI程序，也可用于开发非GUI程序，比如控制台工具和服务器。但Qt不是游戏引擎，它只是提供了一套比较完善的，通用型的轮子，跟boost，folly等一样，是游戏开发的技术选型中占比不到5%的那一小部分。所以此次实训选择的这个题目，对于一个只有两学期C++基础且Qt零基础的我来说无疑是个艰巨的任务。

此次实训我选择做一款依赖于Qt图形视图框架的二合一桌面游戏，游戏一的基本玩法是控制玩家在地图上移动，并通过吃孢子，星星等等使自己的体重增加，还要避免被体重大于自身的敌人吃掉；游戏二是数字华容道，有3x3,4x4,5x5和6x6四种玩法。为了实现这一款游戏，我从零开始学Qt，一步一步琢磨，解决问题，体会到了编程的乐趣，不断应用和巩固所学到的C++知识，提高了自己的编程能力。

1. 主要的任务与时间节点安排：
2. Qt的学习：7月12日~7月22日
3. 需求分析：7月18日~7月22日
4. 基本操作的实现：7月23日~7月31日
5. 玩家皮肤，游戏背景的设计与实现：7月28日~7月31日
6. 登录界面的设计与实现：7月29日~7月31日
7. 菜单界面的设计与实现：8月1日~8月4日
8. 程序的测试：7月23日~8月23日

# 软件需求分析

1. 功能需求
2. 菜单功能：

1）初始界面：在进入游戏后，提供初始界面，可以选择游戏一或游戏二；

2）游戏一的菜单界面：在菜单界面，玩家可设置敌人数量，游戏时间，可以选择游戏模式，也可以返回到初始界面，在游戏中，可按下后退键返回到菜单；

3）游戏二的菜单界面：可以选择游戏模式，可以返回到初始界面，可在游戏中由鼠标点击唤醒；

4）游戏二的重置功能：可在游戏中或游戏结束后重置游戏。

1. 游戏一的逻辑功能：
2. 玩家对角色的控制：玩家通过鼠标点击来控制角色360。任意方向的移动，通过键盘控制玩家吐出气泡或进行攻击；
3. 显示功能：在游戏界面中显示角色的名字、体重，敌人的名字、体重，玩家设定的时间，并实时更新排行榜；
4. 背景的视差滚动：游戏提供9000px\*5000px的大地图，通过视差滚动技术，形成角色在地图中移动的视觉效果；
5. 其他角色的生成：电脑自动生成敌人并控制敌人移动，敌人自动收集球团，追击玩家角色；以及生成孢子，星星，星云等；
6. 碰撞检测：玩家、敌人与气泡、孢子、星星、星云之间均有碰撞检测，碰撞后，两者产生反馈；
7. 排名功能：游戏结束后，显示最终结果。
8. 游戏二的逻辑功能：
9. 移动数字方块：玩家可以使用方向键控制数字方块的移动；
10. 游戏胜利的判定：每一步移动，游戏将对当前方块位置进行判定，若全部归位，则显示游戏完成面板，包括历史记录和当前所用时间；
11. 计时功能：游戏开始，界面左边计时器就开始计时，精度为秒，范围是0~999秒。
12. 性能需求：
13. 界面：初始界面，菜单界面，游戏界面，要求界面精美，运行流畅，反应时间短；
14. 可玩性：提供多种游戏模式。

# 软件设计与实现

1. 基本框架

（1）流程图



（2）UML类图



1. 初始界面：
2. 槽函数void Assistance::enter()产生过渡动画：

void Assistance::enter(){

if(isok==false){

if(pix0->opacity()<=1&&pix0->opacity()>0)

pix0->setOpacity(pix0->opacity()-0.01);

else{

if(pix0->scene()!=NULL){

game->scene->removeItem(pix0);

delete pix0;

}

pix1->setVisible(true);

if(pix1->opacity()<=1&&pix1->opacity()>0){

pix1->setOpacity(pix1->opacity()-0.01);

}

else{

if(pix1->scene()!=NULL){

game->scene->removeItem(pix1);

delete pix1;

}

isok=true;

}

}

}

else{

timer0->stop();

delete timer0;

game->control();

}

}

1. 定时器控制动画的时间：

QObject::connect(assistance->timer0,SIGNAL(timeout()),assistance,SLOT(enter()));

assistance->timer0->start(15);

1. control()函数生成主菜单：

void Game::control(){

scene->clear();

scene->setSceneRect(-900,-500,1800,1000);

scene->setBackgroundBrush(QBrush(Qt::black));

interface=CONTROL;

title=new QGraphicsPixmapItem;

battleofbubbles=new QGraphicsPixmapItem;

slidingblocks=new QGraphicsPixmapItem;

title->setPixmap(QPixmap(":/images/Game2in1.png"));

battleofbubbles->setPixmap(QPixmap(":/images/BattleOfBubbles.png"));

slidingblocks->setPixmap(QPixmap(":/images/SlidingBlocks.png"));

scene->addItem(title);

scene->addItem(battleofbubbles);

scene->addItem(slidingblocks);

title->setPos(-310.5,-300);

battleofbubbles->setPos(-200,-100);

slidingblocks->setPos(-200,100);

battleofbubbles->setTransformOriginPoint(200,45.5);

slidingblocks->setTransformOriginPoint(200,45.5);

setMouseTracking(true);

show();

}

1. 游戏一菜单：
2. Game类成员函数void Game::menu()生成游戏一菜单界面：

void Game::menu(){

menu\_switch=new QMediaPlayer;

menu\_click=new QMediaPlayer;

menu\_switch->setMedia(QUrl("qrc:/music/switch.mp3"));

menu\_click->setMedia(QUrl("qrc:/music/click.mp3"));

//create menu items

menu\_planet=new QGraphicsPixmapItem();

menu\_planet->setPixmap(QPixmap(":/images/menu\_planet.png"));

scene->addItem(menu\_planet);

menu\_planet->setPos(-250,-250);

menu\_planet->setZValue(2);

menu\_planet->setOpacity(0.5);

//1p button

p\_1=new QGraphicsPixmapItem;

p\_1->setPixmap(QPixmap(":/images/1p.png"));

scene->addItem(p\_1);

p\_1->setPos(-250,-250);

p\_1->setZValue(1000);

p\_1->setVisible(false);

//np button

p\_n=new QGraphicsPixmapItem;

p\_n->setPixmap(QPixmap(":/images/np.png"));

scene->addItem(p\_n);

p\_n->setPos(-250,-250);

p\_n->setZValue(1000);

p\_n->setVisible(false);

//more button

p\_more=new QGraphicsPixmapItem;

p\_more->setPixmap(QPixmap(":/images/more.png"));

scene->addItem(p\_more);

p\_more->setPos(-250,-250);

p\_more->setZValue(1000);

p\_more->setVisible(false);

item=new QGraphicsRectItem;

scene->addItem(item);

item->setRect(-22,-54,205,34);

item->setBrush(QBrush(Qt::white));

item->setOpacity(0.4);

item->setZValue(1000);

item->setVisible(false);

yes=new QGraphicsPixmapItem;

yes->setPixmap(QPixmap(":/images/yes.png"));

scene->addItem(yes);

yes->setZValue(1002);

yes->setPos(-1000,-1000);

//modetext

modetext=new QGraphicsTextItem;

modetext->setPlainText(QString("Classic"));

modetext->setDefaultTextColor(Qt::white);

modetext->setFont(QFont("times",12));

scene->addItem(modetext);

modetext->setPos(28,-54);

modetext->setOpacity(0.7);

modetext->setZValue(1000.5);

//playeramount and timelimt

playeramount=new QGraphicsTextItem;

timelimt=new QGraphicsTextItem;

//create the animation as background

menu\_item1=new Myitem();

menu\_item2=new Myitem();

menu\_item1->setPixmap(QPixmap(":/images/enemy0.png").scaled(QSize(200,200)));

menu\_item2->setPixmap(QPixmap(":/images/enemy3.png").scaled(QSize(150,150)));

scene->addItem(menu\_item1);

scene->addItem(menu\_item2);

menu\_item1->setPos(qrand()%length-length/2,qrand()%width-width/2);

menu\_item2->setPos(qrand()%length-length/2,qrand()%width-width/2);

menu\_item1->setZValue(0);

menu\_item2->setZValue(0);

menu\_item1->setOpacity(0.4);

menu\_item2->setOpacity(0.4);

menu\_item1->start();

menu\_item2->start();

interface=MAINMENU;

setMouseTracking(true);

}

1. 游戏一鼠标点击事件，在不同界面中提供不同的响应：

void Game::*mousePressEvent*(QMouseEvent \*event){

... ...

if(interface==MAINMENU){

if(((event->pos().x()-763)\*(event->pos().x()-763)+(event->pos().y()-638)\*(event->pos().y()-638)<6400)

&&((event->pos().x()-900)\*(event->pos().x()-900)+(event->pos().y()-645)\*(event->pos().y()-645)>8100)){

menu\_click->setPosition(0.0);

menu\_click->play();

//1p was pressed

p1\_setting=new QGraphicsPixmapItem;

p1\_setting->setPixmap(QPixmap(":/images/1psetting.png"));

scene->addItem(p1\_setting);

p1\_setting->setPos(-150,-50);

p1\_setting->setZValue(1001);

//playeramount and timelimt

playeramount=new QGraphicsTextItem;

timelimt=new QGraphicsTextItem;

playeramount->setPlainText(QString::number(pl\_a));

playeramount->setDefaultTextColor(Qt::white);

playeramount->setFont(QFont("times",12));

playeramount->setParentItem(p1\_setting);

playeramount->setPos(175,10);

timelimt->setPlainText(QString::number(ti\_l));

timelimt->setDefaultTextColor(Qt::white);

timelimt->setFont(QFont("times",12));

timelimt->setParentItem(p1\_setting);

timelimt->setPos(175,45);

interface=SETTING;

return;

}

else if(((event->pos().x()-900)\*(event->pos().x()-900)+(event->pos().y()-645)\*(event->pos().y()-645)<8100)){

menu\_click->setPosition(0.0);

menu\_click->play();

//np was pressed

scene->clear();

if(mode==CLASSIC)

rungame();

else if(mode==MELEE)

rungame2();

}

else if(((event->pos().x()-1039)\*(event->pos().x()-1039)+(event->pos().y()-638)\*(event->pos().y()-638)<6400)

&&((event->pos().x()-900)\*(event->pos().x()-900)+(event->pos().y()-645)\*(event->pos().y()-645)>8100)){

//more was pressed

menu\_click->setPosition(0.0);

menu\_click->play();

}

else if(event->pos().x()>878&&event->pos().x()<1083&&event->pos().y()>446&&event->pos().y()<480){

menu\_click->setPosition(0.0);

menu\_click->play();

//choose mode

interface=CHOOSINGMODE;

gamemode=new QGraphicsPixmapItem;

gamemode->setPixmap(QPixmap(":/images/gamemode.png"));

scene->addItem(gamemode);

gamemode->setPos(-150,-100);

gamemode->setZValue(1001);

return;

}

}

else if(interface==CHOOSINGMODE){

if(event->pos().x()>1010&&event->pos().x()<1025&&event->pos().y()>410&&event->pos().y()<425){

menu\_click->setPosition(0.0);

menu\_click->play();

//exit gamemodechoosing

delete gamemode;

yes->setPos(-1000,-1000);

interface=MAINMENU;

}

//if choose the clssic mode

if(((event->pos().x()-790)\*(event->pos().x()-790)+(event->pos().y()-472)\*(event->pos().y()-472)<=625)){

menu\_click->setPosition(0.0);

menu\_click->play();

mode=CLASSIC;

modetext->setPlainText(QString("Classic"));

yes->setPos(-135,-54);

}

//if choose the paint mode

if(((event->pos().x()-790)\*(event->pos().x()-790)+(event->pos().y()-550)\*(event->pos().y()-550)<=625)){

menu\_click->setPosition(0.0);

menu\_click->play();

mode=MELEE;

modetext->setPlainText(QString("Melee"));

yes->setPos(-135,26);

}

}

else if(interface==SETTING){

if(event->pos().x()>1021&&event->pos().x()<1036&&event->pos().y()>463&&event->pos().y()<478){

menu\_click->setPosition(0.0);

menu\_click->play();

//exit setting

delete p1\_setting;

interface=MAINMENU;

}

//if add the playeramount

if(event->pos().x()>964&&event->pos().x()<978&&event->pos().y()>474&&event->pos().y()<487){

menu\_click->setPosition(0.0);

menu\_click->play();

if(pl\_a<20)

pl\_a++;

playeramount->setPlainText(QString::number(pl\_a));

}

//if subtract

if(event->pos().x()>981&&event->pos().x()<996&&event->pos().y()>474&&event->pos().y()<487){

menu\_click->setPosition(0.0);

menu\_click->play();

if(pl\_a>5)

pl\_a--;

playeramount->setPlainText(QString::number(pl\_a));

}

//if add the time

if(event->pos().x()>964&&event->pos().x()<978&&event->pos().y()>509&&event->pos().y()<523){

menu\_click->setPosition(0.0);

menu\_click->play();

if(ti\_l<30)

ti\_l++;

timelimt->setPlainText(QString::number(ti\_l));

}

//if subtract

if(event->pos().x()>981&&event->pos().x()<996&&event->pos().y()>509&&event->pos().y()<523){

menu\_click->setPosition(0.0);

menu\_click->play();

if(ti\_l>1)

ti\_l--;

timelimt->setPlainText(QString::number(ti\_l));

}

}

... ...

}

1. 游戏一大循环：
2. 成员函数void Game::rungame()实现游戏一的classic模式：

void Game::rungame(){

//scale(0.2,0.2);

interface=GAMING;

enemy\_amount=pl\_a;

timing=new QTimer();

connect(timing,SIGNAL(timeout()),assistance,SLOT(gameover()));

timing->start(60000\*ti\_l);

timecount = new TimeCounter;

scene->addItem(timecount);

timecount->setZValue(20);

timecount->setPos(mapToScene(890,10));

timingcount=new QTimer;

QObject::connect(timingcount,SIGNAL(timeout()),timecount,SLOT(showtime()));

timingcount->start(1000);

//scale(0.2,0.2);

qsrand(QTime(0,0,0).secsTo(QTime::currentTime()));

// create the player

player = new Player();

scene->addItem(player);

player->setPos(0,0);

player->setZValue(2);

player->setFocus();

player->setCacheMode(QGraphicsItem::ItemCoordinateCache);

enemy.resize(enemy\_amount);

//create the enemy

for(int i=0;i<enemy.size();i++){

enemy[i]=new Enemy();

scene->addItem(enemy[i]);

enemy[i]->setPos(qrand()%9000-4500,qrand()%5000-2500);

enemy[i]->createRect();

enemy[i]->setCacheMode(QGraphicsItem::ItemCoordinateCache);

enemy[i]->id=i;

enemy[i]->col\_id=-1;

enemy[i]->isCollided=false;

}

// create the score

weight = new Weight();

scene->addItem(weight);

weight->setZValue(2);

weight->setPos(mapToScene(10,10));

//set the billboard

billboard = new Billboard();

//initialize the timers

timer1 = new QTimer();

timer2 = new QTimer();

timer3 = new QTimer();//opl2

timer4.resize(enemy\_amount);

timer5.resize(cloud\_amount);

for(int i=0;i<enemy.size();i++)

timer4[i]=new QTimer();

for(int i=0;i<clouds.size();i++)

timer5[i]=new QTimer();

//move the player

QObject::connect(timer1,SIGNAL(timeout()),player,SLOT(updateView()));//ok

QObject::connect(timer1,SIGNAL(timeout()),player,SLOT(move()));

QObject::connect(timer1,SIGNAL(timeout()),billboard,SLOT(updateBillboard()));//ok

QObject::connect(timer2,SIGNAL(timeout()),player,SLOT(createfood()));//ok

QObject::connect(timer3,SIGNAL(timeout()),player,SLOT(playmusic()));

//move the enemies

for(int i=0;i<enemy.size();i++){

QObject::connect(timer4[i],SIGNAL(timeout()),enemy[i],SLOT(updateRect()));//ok

QObject::connect(timer1,SIGNAL(timeout()),enemy[i],SLOT(move()));

timer4[i]->start(1000+50\*(i-1));

}

//create the clouds

for(int i=0;i<clouds.size();i++){

clouds[i]=new Cloud();

clouds[i]->setCacheMode(QGraphicsItem::ItemCoordinateCache);

QObject::connect(timer5[i],SIGNAL(timeout()),clouds[i],SLOT(rotate()));//ok

timer5[i]->start(50);

}

timer1->start(50);

timer2->start(100);

timer3->start(40000);

// set music

music = new QMediaPlayer();

music->setMedia(QUrl("qrc:/music/bg.mp3"));

music->play();

}

1. 成员函数void Game::rungame2()实现游戏一的melee模式：

void Game::rungame2(){

scene->setBackgroundBrush(QBrush(QColor(40,40,40)));

interface=GAMING2;

qsrand(QTime(0,0,0).secsTo(QTime::currentTime()));

shieldamount=5;

liquidamount=5;

iceamount=5;

speedupamount=5;

blackholeamount=6;

hole.resize(0);

for(int i=0;i<shieldamount;i++){

Shield \*sld=new Shield;

scene->addItem(sld);

sld->setPos(qrand()%9000-4500,qrand()%5000-2500);

}

for(int i=0;i<liquidamount;i++){

Liquid \*lqd=new Liquid;

scene->addItem(lqd);

lqd->setPos(qrand()%9000-4500,qrand()%5000-2500);

}

for(int i=0;i<iceamount;i++){

Ice \*ice=new Ice;

scene->addItem(ice);

ice->setPos(qrand()%9000-4500,qrand()%5000-2500);

}

for(int i=0;i<speedupamount;i++){

Speedup \*spd=new Speedup;

scene->addItem(spd);

spd->setPos(qrand()%9000-4500,qrand()%5000-2500);

}

for(int i=0;i<blackholeamount;i++){

Blackhole \*blh=new Blackhole;

scene->addItem(blh);

blh->setPos(qrand()%9000-4500,qrand()%5000-2500);

blh->id=i;

hole.append(blh);

}

QTimer \*supplement=new QTimer;

QObject::connect(supplement,SIGNAL(timeout()),assistance,SLOT(createItems()));

supplement->start(1000);

//add others

enemy\_amount=pl\_a;

timing=new QTimer();

connect(timing,SIGNAL(timeout()),assistance,SLOT(gameover()));

timing->start(60000\*ti\_l);

timecount = new TimeCounter;

scene->addItem(timecount);

timecount->setZValue(20);

timecount->setPos(mapToScene(890,10));

timingcount=new QTimer;

QObject::connect(timingcount,SIGNAL(timeout()),timecount,SLOT(showtime()));

timingcount->start(1000);

// create the player

player = new Player();

scene->addItem(player);

player->setPos(0,0);

player->setZValue(2);

player->setFocus();

player->setCacheMode(QGraphicsItem::ItemCoordinateCache);

enemy.resize(enemy\_amount);

//create the enemy

for(int i=0;i<enemy.size();i++){

enemy[i]=new Enemy();

scene->addItem(enemy[i]);

enemy[i]->setPos(qrand()%9000-4500,qrand()%5000-2500);

enemy[i]->createRect();

enemy[i]->setCacheMode(QGraphicsItem::ItemCoordinateCache);

enemy[i]->id=i;

enemy[i]->col\_id=-1;

enemy[i]->isCollided=false;

}

// create the score

weight = new Weight();

scene->addItem(weight);

weight->setZValue(100);

weight->setPos(mapToScene(10,10));

//set the billboard

billboard = new Billboard();

//initialize the timers

timer1 = new QTimer();

timer2 = new QTimer();

timer3 = new QTimer();//opl2

timer4.resize(enemy\_amount);

timer5.resize(cloud\_amount);

for(int i=0;i<enemy.size();i++)

timer4[i]=new QTimer();

for(int i=0;i<clouds.size();i++)

timer5[i]=new QTimer();

//move the player

QObject::connect(timer1,SIGNAL(timeout()),player,SLOT(updateView()));//ok

QObject::connect(timer1,SIGNAL(timeout()),player,SLOT(move()));

QObject::connect(timer1,SIGNAL(timeout()),billboard,SLOT(updateBillboard()));//ok

QObject::connect(timer2,SIGNAL(timeout()),player,SLOT(createfood()));//ok

QObject::connect(timer3,SIGNAL(timeout()),player,SLOT(playmusic()));

//create the clouds

for(int i=0;i<clouds.size();i++){

clouds[i]=new Cloud();

scene->addItem(clouds[i]);

clouds[i]->setPos(qrand()%9000-4500,qrand()%5000-2500);

clouds[i]->setCacheMode(QGraphicsItem::ItemCoordinateCache);

QObject::connect(timer5[i],SIGNAL(timeout()),clouds[i],SLOT(rotate()));//ok

timer5[i]->start(80);

}

//move the enemies

for(int i=0;i<enemy.size();i++){

QObject::connect(timer4[i],SIGNAL(timeout()),enemy[i],SLOT(updateRect()));//ok

QObject::connect(timer1,SIGNAL(timeout()),enemy[i],SLOT(move()));

timer4[i]->start(300+50\*(i-1));

}

timer1->start(80);

timer2->start(100);

timer3->start(40000);

// set music

music = new QMediaPlayer();

music->setMedia(QUrl("qrc:/music/bg.mp3"));

music->play();

}

1. 鼠标事件以及键盘事件处理游戏中的操作：

void Game::*mousePressEvent*(QMouseEvent \*event){

... ...

else if(interface==GAMING||interface==GAMING2){

//qDebug()<<player->mapFromScene(mapToScene(event->pos()));

//the point of mouse on the scene

QPointF point = mapToScene(event->pos());

//draw the cursor

circle->setRect(QRectF(point.x()-30, point.y()-30, 60, 60));

//this will change the scenepos of the mouse after

scene->addItem(circle);

//get the rotation

if(player->scenePos().x()+player->size\_x/2==point.x()&&

player->scenePos().y()+player->size\_y/2==point.y())

;

else{

qreal rotate\_angle=angle(player->scenePos().x(),

player->scenePos().y(),point.x(),point.y());

player->setRotate(rotate\_angle);

}

}

else if(interface==GAMEOVER){

if(event->pos().x()>717&&event->pos().x()<805&&event->pos().y()>710&&event->pos().y()<740){

menu\_click->setPosition(0.0);

menu\_click->play();

//home

scene->clear();

interface=MAINMENU;

menu();

}

if(event->pos().x()>856&&event->pos().x()<944&&event->pos().y()>710&&event->pos().y()<740){

menu\_click->setPosition(0.0);

menu\_click->play();

//restart

delete weight;

scene->clear();

if(mode==CLASSIC)

rungame();

else if(mode==MELEE)

rungame2();

}

if(event->pos().x()>994&&event->pos().x()<1083&&event->pos().y()>710&&event->pos().y()<740){

menu\_click->setPosition(0.0);

menu\_click->play();

//exit

scene->clear();

close();

}

}

}

void Game::*mouseReleaseEvent*(QMouseEvent \*event){

if(interface!=GAMING&&interface!=GAMING2)

return;

if(circle->scene()!=0)

scene->removeItem(circle);

}

void Game::*mouseMoveEvent*(QMouseEvent \*event){

if(interface==MAINMENU){

//change color when the mouse is on the menu buttons

if(((event->pos().x()-763)\*(event->pos().x()-763)+(event->pos().y()-638)\*(event->pos().y()-638)<6400)

&&((event->pos().x()-900)\*(event->pos().x()-900)+(event->pos().y()-645)\*(event->pos().y()-645)>8100)){

if(lastmouse!=p1){

menu\_switch->setPosition(0.0);

menu\_switch->play();

}

//1p

p\_1->setVisible(true);

lastmouse=p1;

}

else

p\_1->setVisible(false);

if(((event->pos().x()-900)\*(event->pos().x()-900)+(event->pos().y()-645)\*(event->pos().y()-645)<8100)){

if(lastmouse!=pn){

menu\_switch->setPosition(0.0);

menu\_switch->play();

}

//np

p\_n->setVisible(true);

lastmouse=pn;

}

else

p\_n->setVisible(false);

if(((event->pos().x()-1039)\*(event->pos().x()-1039)+(event->pos().y()-638)\*(event->pos().y()-638)<6400)

&&((event->pos().x()-900)\*(event->pos().x()-900)+(event->pos().y()-645)\*(event->pos().y()-645)>8100)){

if(lastmouse!=pmore){

menu\_switch->setPosition(0.0);

menu\_switch->play();

}

//more

p\_more->setVisible(true);

lastmouse=pmore;

}

else

p\_more->setVisible(false);

//rectitem on the menu

if(event->pos().x()>878&&event->pos().x()<1083&&event->pos().y()>446&&event->pos().y()<480){

if(lastmouse!=pitem){

menu\_switch->setPosition(0.0);

menu\_switch->play();

}

item->setVisible(true);

lastmouse=pitem;

}

else

item->setVisible(false);

}

}

void Game::*keyPressEvent*(QKeyEvent \*event){

if(interface==GAMING||interface==GAMING2){

if(player->shoot){

if(event->key()==Qt::Key\_Space){

if(interface==GAMING){

player->createBubble(player->rotate\_angle);

}

else if(interface==GAMING2){

if(player->hasshield==false&&player->hasice==false&&player->hasliquid==false&&player->hasspeedup==false)

player->createBubble(player->rotate\_angle);

else if(player->hasliquid==true){

Liquid \*temps=dynamic\_cast<Liquid \*>((player->childItems())[1]);

temps->setParentItem(NULL);

temps->setPixmap(QPixmap(":/images/energy.png"));

ani=new QPropertyAnimation(temps,"pos");

ani->setDuration(400);

ani->setKeyValueAt(0,QPointF(player->pos().x()+(player->size\_x)\*qCos(player->rotate\_angle)-15,player->pos().y()-(player->size\_y)\*qSin(player->rotate\_angle)-18.5));

ani->setKeyValueAt(1,QPointF(player->pos().x()+5\*player->size\_x\*qCos(player->rotate\_angle)-15,player->pos().y()-5\*player->size\_y\*qSin(player->rotate\_angle)-18.5));

ani->start();

QTimer \*temp=new QTimer;

QTimer \*temp\_det=new QTimer;

QObject::connect(temp,SIGNAL(timeout()),temps,SLOT(deleteLater()));

QObject::connect(temp,SIGNAL(timeout()),temp\_det,SLOT(deleteLater()));

QObject::connect(temp,SIGNAL(timeout()),temp,SLOT(deleteLater()));

QObject::connect(temp\_det,SIGNAL(timeout()),temps,SLOT(detect()));

temp->start(400);

temp\_det->start(80);

player->hasliquid=false;

liquidamount-=1;

}

else if(player->hasice==true){

Ice \*temps=dynamic\_cast<Ice \*>((player->childItems())[1]);

temps->setParentItem(NULL);

temps->setPixmap(QPixmap(":/images/energy.png"));

ani=new QPropertyAnimation(temps,"pos");

ani->setDuration(400);

ani->setKeyValueAt(0,QPointF(player->pos().x()+(player->size\_x)\*qCos(player->rotate\_angle)-15,player->pos().y()-(player->size\_y)\*qSin(player->rotate\_angle)-18.5));

ani->setKeyValueAt(1,QPointF(player->pos().x()+5\*player->size\_x\*qCos(player->rotate\_angle)-15,player->pos().y()-5\*player->size\_y\*qSin(player->rotate\_angle)-18.5));

ani->start();

QTimer \*temp=new QTimer;

QTimer \*temp\_det=new QTimer;

QObject::connect(temp,SIGNAL(timeout()),temps,SLOT(deleteLater()));

QObject::connect(temp,SIGNAL(timeout()),temp\_det,SLOT(deleteLater()));

QObject::connect(temp,SIGNAL(timeout()),temp,SLOT(deleteLater()));

QObject::connect(temp\_det,SIGNAL(timeout()),temps,SLOT(detect()));

temp->start(400);

temp\_det->start(80);

player->hasice=false;

iceamount-=1;

}

else if(player->hasshield==true){

Shield \*temps=dynamic\_cast<Shield \*>((player->childItems())[1]);

if(temps->state==0){

player->hasshield=false;

temps->state=1;

temps->setOffset(15-player->size\_x/2,18.5-player->size\_y/2);

temps->setPixmap(QPixmap(":/images/gradient.png").scaled(QSize(player->size\_x,player->size\_y)));

ani=new QPropertyAnimation(temps,"opacity");

ani->setDuration(10000);

ani->setKeyValueAt(0,0);

ani->setKeyValueAt(0.125,1);

ani->setKeyValueAt(0.25,0);

ani->setKeyValueAt(0.375,1);

ani->setKeyValueAt(0.5,0);

ani->setKeyValueAt(0.625,1);

ani->setKeyValueAt(0.75,0);

ani->setKeyValueAt(0.875,1);

ani->setKeyValueAt(1,0);

ani->start();

player->isinvisible=true;

QTimer \*temp=new QTimer;

QObject::connect(temp,SIGNAL(timeout()),temps,SLOT(endlife()));

QObject::connect(temp,SIGNAL(timeout()),player,SLOT(becomeInvisible()));

QObject::connect(temp,SIGNAL(timeout()),temp,SLOT(deleteLater()));

temp->start(10000);

shieldamount-=1;

}

}

else if(player->hasspeedup==true){

Speedup \*temps=dynamic\_cast<Speedup \*>((player->childItems())[1]);

delete temps;

player->isfast=true;

player->hasspeedup=false;

QTimer \*temp=new QTimer;

QObject::connect(temp,SIGNAL(timeout()),player,SLOT(becomeFast()));

QObject::connect(temp,SIGNAL(timeout()),temp,SLOT(deleteLater()));

temp->start(5000);

speedupamount-=1;

}

}

}

}

}

}

1. 游戏一玩家角色控制：
2. 修改鼠标点击光标为半透明黄色填充圆：

//the point of mouse on the scene

QPointF point = mapToScene(event->pos());

//draw the cursor

circle->setRect(QRectF(point.x()-30, point.y()-30, 60, 60));

//this will change the scenepos of the mouse after

scene->addItem(circle);

1. 计算光标位置与玩家位置的偏转角：

//get the rotation

if(player->scenePos().x()+player->size\_x/2==point.x()&&

player->scenePos().y()+player->size\_y/2==point.y())

;

else{

qreal rotate\_angle=angle(player->scenePos().x(),

player->scenePos().y(),point.x(),point.y());

player->setRotate(rotate\_angle);

}

1. 检测碰撞并移动玩家：

void Player::move(){

if(isactive){

setZValue(20);

//set the offset of the bounding

offset\_x=0;

offset\_y=0;

//judge the collisions

QList<QGraphicsItem \*> colliding\_items = collidingItems();

for (int i = 0, n = colliding\_items.size(); i < n; ++i){

if (typeid(\*(colliding\_items[i])) == typeid(QGraphicsEllipseItem)&&

colliding\_items[i]->opacity()==1&&typeid(\*(colliding\_items[i])) != typeid(Bubble)){

eatfood->setPosition(0.0);

eatfood->play();

// increase the weight

increase(5);

game->weight->showWeight();

if(colliding\_items[i]->scene()!=NULL&&scene()!=NULL){

foodamount-=1;

delete colliding\_items[i];

}

}

else if (typeid(\*(colliding\_items[i])) == typeid(QGraphicsPixmapItem)

&&typeid(\*(colliding\_items[i])) != typeid(Enemy)&&colliding\_items[i]->opacity()==1){

eatfoodstar->setPosition(0.0);

eatfoodstar->play();

// increase the weight

increase(8);

game->weight->showWeight();

if(colliding\_items[i]->scene()!=NULL&&scene()!=NULL){

foodstaramount-=1;

delete colliding\_items[i];

}

}

else if(typeid(\*(colliding\_items[i])) == typeid(Enemy)&&

colliding\_items[i]->scene()!=NULL&&scene()!=NULL){

Enemy \*temp=dynamic\_cast<Enemy \*>(colliding\_items[i]);

qreal d=distance(temp->scenePos().x(),temp->scenePos().y(),pos().x(),pos().y());

if(size\_x<=temp->sx&&isinvisible==false){

if(2\*d<=temp->sx&&game->ratio\*size\_x<=temp->sx){

if((childItems()).size()>1){

if(typeid(\*((childItems())[1]))==typeid(Shield))

game->shieldamount-=1;

else if(typeid(\*((childItems())[1]))==typeid(Ice))

game->iceamount-=1;

else if(typeid(\*((childItems())[1]))==typeid(Speedup))

game->speedupamount-=1;

else if(typeid(\*((childItems())[1]))==typeid(Liquid))

game->liquidamount-=1;

delete ((childItems())[1]);

}

game->scene->removeItem(this);

reset();

temp->increase(size\_x);

return;

}

}

else if(size\_x>temp->sx){

if(2\*d<=size\_x&&game->ratio\*temp->sx<=size\_x){

increase(temp->sx);

temp->reset();

game->enemy\_amount-=1;

game->weight->showWeight();

}

}

}

else if(typeid(\*(colliding\_items[i])) == typeid(Cloud)&&

colliding\_items[i]->scene()!=NULL&&scene()!=NULL){

if(size\_x<75){

eatcloud->setPosition(0.0);

eatcloud->play();

// increase the weight

increase(30);

game->weight->showWeight();

if(colliding\_items[i]->scene()!=NULL&&scene()!=NULL){

game->cloud\_amount-=1;

colliding\_items[i]->setPos(qrand()%9000-4500,qrand()%5000-2500);

}

}

else{

createBubble((qrand()%629)/100);

createBubble((qrand()%629)/100);

createBubble((qrand()%629)/100);

createBubble((qrand()%629)/100);

game->weight->showWeight();

}

}

else if(typeid(\*(colliding\_items[i])) == typeid(Bubble)&&

colliding\_items[i]->scene()!=NULL&&scene()!=NULL){

// increase the weight

increase(10);

game->weight->showWeight();

if(colliding\_items[i]->scene()!=NULL&&scene()!=NULL){

delete colliding\_items[i];

}

}

else if(collideWithShield==false&&typeid(\*(colliding\_items[i])) == typeid(Liquid)&&

colliding\_items[i]->scene()!=NULL&&scene()!=NULL){

if(shoot==true&&hasshield==false&&hasice==false&&hasliquid==false&&hasspeedup==false){

//collect the liquid

hasliquid=true;

colliding\_items[i]->setParentItem(this);

colliding\_items[i]->setPos(-15,-18.5);

}

}

else if(collideWithShield==false&&typeid(\*(colliding\_items[i])) == typeid(Ice)&&

colliding\_items[i]->scene()!=NULL&&scene()!=NULL){

if(shoot==true&&hasshield==false&&hasice==false&&hasliquid==false&&hasspeedup==false){

//collect the ice

hasice=true;

colliding\_items[i]->setParentItem(this);

colliding\_items[i]->setPos(-15,-18.5);

}

}

else if(collideWithShield==false&&typeid(\*(colliding\_items[i])) == typeid(Shield)&&

colliding\_items[i]->scene()!=NULL&&scene()!=NULL){

collideWithShield=true;

if(shoot==true&&hasshield==false&&hasice==false&&hasliquid==false&&hasspeedup==false){

//collect the shield

hasshield=true;

colliding\_items[i]->setParentItem(this);

colliding\_items[i]->setPos(-15,-18.5);

}

}

else if(collideWithShield==false&&typeid(\*(colliding\_items[i])) == typeid(Speedup)&&

colliding\_items[i]->scene()!=NULL&&scene()!=NULL){

if(shoot==true&&hasshield==false&&hasice==false&&hasliquid==false&&hasspeedup==false){

//collect the speedup

hasspeedup=true;

colliding\_items[i]->setParentItem(this);

colliding\_items[i]->setPos(-15,-18.5);

}

}

else if(collideWithShield==false&&typeid(\*(colliding\_items[i])) == typeid(Blackhole)&&

colliding\_items[i]->scene()!=NULL&&scene()!=NULL){

if(shoot==true&&hasshield==false&&hasice==false&&hasliquid==false&&hasspeedup==false){

Blackhole \*temp=dynamic\_cast<Blackhole \*>(colliding\_items[i]);

int id\_bind;

do{

id\_bind=qrand()%6;

}while(id\_bind==temp->id);

QPointF point=game->hole[id\_bind]->pos();

setPos(point);

temp->setPos(qrand()%9000-4500,qrand()%5000-2500);

game->hole[id\_bind]->setPos(qrand()%9000-4500,qrand()%5000-2500);

}

}

}

//move the player

if(rotate\_angle==-1)

return;

//whether change direction

if(rotate\_angle!=last){

if(size\_x<=300)

v=-0.025\*size\_x+20;

else

v=10;

last=rotate\_angle;

setRotate(rotate\_angle);

}

else if(v>0&&rotate\_angle==last){

if(size\_x<=400)

v=-0.025\*size\_x+20;

else

v=10;

}

if(isfast==true){

v\*=2;

}

//move the player

if((1.57<=rotate\_angle&&rotate\_angle<=3.14&&pos().x()>-4500&&pos().y()>-2500)||

(4.71<=rotate\_angle&&rotate\_angle<=6.28&&pos().x()<4500&&pos().y()<2500)||

(0<=rotate\_angle&&rotate\_angle<1.57&&pos().x()<4500&&pos().y()>-2500)||

(3.14<rotate\_angle&&rotate\_angle<4.71&&pos().x()>-4500&&pos().y()<2500)){

//setPos(pos().x()+int(v\*qCos(rotate\_angle)),pos().y()-int(v\*qSin(rotate\_angle)));

moveBy((v\*qCos(rotate\_angle)),-(v\*qSin(rotate\_angle)));

}

else if(pos().x()<=-4500&&-2500<pos().y()&&pos().y()<2500){

if(1.57<=rotate\_angle&&rotate\_angle<=4.71){

//setPos(pos().x(),pos().y()-int(v\*qSin(rotate\_angle)));

moveBy(0,-(v\*qSin(rotate\_angle)));

}

}

else if(-4500<pos().x()&&pos().x()<4500&&pos().y()<=-2500){

if(0<=rotate\_angle&&rotate\_angle<=3.14){

//setPos(pos().x()+int(v\*qCos(rotate\_angle)),pos().y());

moveBy((v\*qCos(rotate\_angle)),0);

}

}

else if(pos().x()>=4500&&-2500<pos().y()&&pos().y()<2500){

if((0<=rotate\_angle&&rotate\_angle<=1.57)||

(4.71<=rotate\_angle&&rotate\_angle<=6.28)){

//setPos(pos().x(),pos().y()-int(v\*qSin(rotate\_angle)));

moveBy(0,-(v\*qSin(rotate\_angle)));

}

}

else if(-4500<pos().x()&&pos().x()<4500&&pos().y()>=2500){

if((3.14<=rotate\_angle&&rotate\_angle<=6.28)){

//setPos(pos().x()+int(v\*qCos(rotate\_angle)),pos().y());

moveBy((v\*qCos(rotate\_angle)),0);

}

}

}

}

1. 体重增加、减少函数：

void Player::increase(qreal d){

qreal r=d/2;

qreal R=size\_x/2;

qreal delta\_R=2\*pow(r\*r+R\*R,0.5);

size\_x=delta\_R;

size\_y=delta\_R;

playerWeight=int(size\_x\*200-4000);

if(size\_x>40){

shoot=true;

shooting->setVisible(true);

}

}

void Player::decrease(qreal d){

qreal r=d/2;

qreal R=size\_x/2;

qreal delta\_R=2\*pow(R\*R-r\*r,0.5);

size\_x=delta\_R;

size\_y=delta\_R;

playerWeight=int(size\_x\*200-4000);

if(size\_x<=40){

shoot=false;

shooting->setVisible(false);

}

}

1. 发射孢子：

void Player::createBubble(){

bubble=new Bubble();

game->scene->addItem(bubble);

bubble->setRect(0,0,20,20);

bubble->setBrush(QBrush(QColor(qrand()%256,qrand()%256,qrand()%256)));

a=new QPropertyAnimation(bubble,"pos");

a->setDuration(400);

a->setKeyValueAt(0,QPointF(pos().x()+(size\_x)\*qCos(rotate\_angle)-10,pos().y()-(size\_y)\*qSin(rotate\_angle)-10));

a->setKeyValueAt(1,QPointF(pos().x()+5\*size\_x\*qCos(rotate\_angle)-10,pos().y()-5\*size\_y\*qSin(rotate\_angle)-10));

a->start();

decrease(10);

}

1. 玩家被敌人吃掉后重置：

void Player::reset(){

x0=0;

y0=0;

size\_x=20.0;

size\_y=20.0;

v=-0.025\*size\_x+20;

playerWeight=0;

game->scene->addItem(this);

setPos(0,0);

shoot=false;

shooting->setVisible(false);

isinvisible=true;

QTimer \*temp=new QTimer;

QObject::connect(temp,SIGNAL(timeout()),this,SLOT(becomeInvisible()));

temp->start(5000);

}

1. 游戏一敌人生成和移动：
2. 创造敌人：

enemy.resize(enemy\_amount);

//create the enemy

for(int i=0;i<enemy.size();i++){

enemy[i]=new Enemy();

scene->addItem(enemy[i]);

enemy[i]->setPos(qrand()%9000-4500,qrand()%5000-2500);

enemy[i]->createRect();

enemy[i]->setCacheMode(QGraphicsItem::ItemCoordinateCache);

enemy[i]->id=i;

enemy[i]->col\_id=-1;

enemy[i]->isCollided=false;

}

1. 游戏一检测碰撞，移动敌人，发动攻击：

void Enemy::move(){

if(isactive){

setZValue(sx/100);

if(sx<=400)

v=20-0.025\*sx;

else

v=10;

if(isfast==true){

v\*=2;

}

//judge the collisions

QList<QGraphicsItem \*> colliding\_items = collidingItems();

for (int i = 0, n = colliding\_items.size(); i < n; ++i){

if (typeid(\*(colliding\_items[i])) == typeid(QGraphicsEllipseItem)&&

colliding\_items[i]->opacity()==1){

// collided with food

increase(5);

if(colliding\_items[i]->scene()!=NULL&&scene()!=NULL){

game->player->foodamount-=1;

delete colliding\_items[i];

}

}

else if (typeid(\*(colliding\_items[i])) == typeid(QGraphicsPixmapItem)&&colliding\_items[i]->opacity()==1

&&typeid(\*(colliding\_items[i])) != typeid(Player)&&typeid(\*(colliding\_items[i])) != typeid(Enemy)){

// collided with foodstar

increase(8);

if(colliding\_items[i]->scene()!=NULL&&scene()!=NULL){

game->player->foodstaramount-=1;

delete colliding\_items[i];

}

}

else if(typeid(\*(colliding\_items[i])) == typeid(Cloud)&&

colliding\_items[i]->scene()!=NULL&&scene()!=NULL){

if(sx<75){

// collided with cloud

increase(30);

if(colliding\_items[i]->scene()!=NULL&&scene()!=NULL){

game->cloud\_amount-=1;

colliding\_items[i]->setPos(qrand()%9000-4500,qrand()%5000-2500);

}

}

else{

createbubble((qrand()%629)/100);

createbubble((qrand()%629)/100);

createbubble((qrand()%629)/100);

createbubble((qrand()%629)/100);

direction=Angle(pos().x(),pos().y(),colliding\_items[i]->pos().x(),colliding\_items[i]->pos().y())+3.14;

}

}

else if(typeid(\*(colliding\_items[i])) == typeid(Enemy)&&

colliding\_items[i]->scene()!=NULL&&scene()!=NULL){

// collided with another enemy

Enemy \*temp=dynamic\_cast<Enemy \*>(colliding\_items[i]);

col\_id=temp->id;

isCollided=true;

game->assistance->collisions.append(this);

game->assistance->enemyCollision();

}

else if(typeid(\*(colliding\_items[i])) == typeid(Bubble)&&

colliding\_items[i]->scene()!=NULL&&scene()!=NULL){

// increase the weight

increase(10);

if(colliding\_items[i]->scene()!=NULL&&scene()!=NULL){

delete colliding\_items[i];

}

}

else if(collideWithShield==false&&typeid(\*(colliding\_items[i])) == typeid(Liquid)&&

colliding\_items[i]->scene()!=NULL&&scene()!=NULL&&colliding\_items[i]->parentItem()==NULL){

if(shoot==true&&hasshield==false&&hasice==false&&hasliquid==false&&hasspeedup==false){

//collect the liquid

hasliquid=true;

colliding\_items[i]->setParentItem(this);

colliding\_items[i]->setPos(-15,-18.5);

}

}

else if(collideWithShield==false&&typeid(\*(colliding\_items[i])) == typeid(Ice)&&

colliding\_items[i]->scene()!=NULL&&scene()!=NULL&&colliding\_items[i]->parentItem()==NULL){

if(shoot==true&&hasshield==false&&hasice==false&&hasliquid==false&&hasspeedup==false){

//collect the ice

hasice=true;

colliding\_items[i]->setParentItem(this);

colliding\_items[i]->setPos(-15,-18.5);

}

}

else if(collideWithShield==false&&typeid(\*(colliding\_items[i])) == typeid(Shield)&&

colliding\_items[i]->scene()!=NULL&&scene()!=NULL&&colliding\_items[i]->parentItem()==NULL){

collideWithShield=true;

if(shoot==true&&hasshield==false&&hasice==false&&hasliquid==false&&hasspeedup==false){

//collect the shield

hasshield=true;

colliding\_items[i]->setParentItem(this);

colliding\_items[i]->setPos(-15,-18.5);

}

}

else if(collideWithShield==false&&typeid(\*(colliding\_items[i])) == typeid(Speedup)&&

colliding\_items[i]->scene()!=NULL&&scene()!=NULL&&colliding\_items[i]->parentItem()==NULL){

if(shoot==true&&hasshield==false&&hasice==false&&hasliquid==false&&hasspeedup==false){

//collect the speedup

hasspeedup=true;

colliding\_items[i]->setParentItem(this);

colliding\_items[i]->setPos(-15,-18.5);

}

}

else if(collideWithShield==false&&typeid(\*(colliding\_items[i])) == typeid(Blackhole)&&

colliding\_items[i]->scene()!=NULL&&scene()!=NULL){

if(shoot==true&&hasshield==false&&hasice==false&&hasliquid==false&&hasspeedup==false){

Blackhole \*temp=dynamic\_cast<Blackhole \*>(colliding\_items[i]);

int id\_bind;

do{

id\_bind=qrand()%6;

}while(id\_bind==temp->id);

QPointF point=game->hole[id\_bind]->pos();

setPos(point);

temp->setPos(qrand()%9000-4500,qrand()%5000-2500);

game->hole[id\_bind]->setPos(qrand()%9000-4500,qrand()%5000-2500);

}

}

}

qreal dd=Distance(pos().x(),pos().y(),game->player->pos().x(),game->player->pos().y());

qreal aa=Angle(pos().x(),pos().y(),game->player->pos().x(),game->player->pos().y());

if(dd<=game->player->size\_x/2+sx/2+sx&&sx>game->ratio\*game->player->size\_x){

direction=aa;

if(hasliquid){

Liquid \*temps=dynamic\_cast<Liquid \*>((childItems())[1]);

temps->setParentItem(NULL);

temps->setPixmap(QPixmap(":/images/energy.png"));

ani=new QPropertyAnimation(temps,"pos");

ani->setDuration(400);

ani->setKeyValueAt(0,QPointF(pos().x()+sx\*qCos(direction)-15,pos().y()-sy\*qSin(direction)-18.5));

ani->setKeyValueAt(1,QPointF(pos().x()+5\*sx\*qCos(direction)-15,pos().y()-5\*sy\*qSin(direction)-18.5));

ani->start();

QTimer \*temp=new QTimer;

QTimer \*temp\_det=new QTimer;

QObject::connect(temp,SIGNAL(timeout()),temps,SLOT(deleteLater()));

QObject::connect(temp,SIGNAL(timeout()),temp\_det,SLOT(deleteLater()));

QObject::connect(temp,SIGNAL(timeout()),temp,SLOT(deleteLater()));

QObject::connect(temp\_det,SIGNAL(timeout()),temps,SLOT(detect()));

temp->start(400);

temp\_det->start(80);

hasliquid=false;

game->liquidamount-=1;

}

else if(hasice){

Ice \*temps=dynamic\_cast<Ice \*>((childItems())[1]);

temps->setParentItem(NULL);

temps->setPixmap(QPixmap(":/images/energy.png"));

ani=new QPropertyAnimation(temps,"pos");

ani->setDuration(400);

ani->setKeyValueAt(0,QPointF(pos().x()+sx\*qCos(direction)-15,pos().y()-sy\*qSin(direction)-18.5));

ani->setKeyValueAt(1,QPointF(pos().x()+5\*sx\*qCos(direction)-15,pos().y()-5\*sy\*qSin(direction)-18.5));

ani->start();

QTimer \*temp=new QTimer;

QTimer \*temp\_det=new QTimer;

QObject::connect(temp,SIGNAL(timeout()),temps,SLOT(deleteLater()));

QObject::connect(temp,SIGNAL(timeout()),temp\_det,SLOT(deleteLater()));

QObject::connect(temp,SIGNAL(timeout()),temp,SLOT(deleteLater()));

QObject::connect(temp\_det,SIGNAL(timeout()),temps,SLOT(detect()));

temp->start(400);

temp\_det->start(80);

hasice=false;

game->iceamount-=1;

}

else if(hasspeedup){

Speedup \*temps=dynamic\_cast<Speedup \*>((childItems())[1]);

delete temps;

isfast=true;

hasspeedup=false;

QTimer \*temp=new QTimer(this);

QObject::connect(temp,SIGNAL(timeout()),this,SLOT(becomefast()));

QObject::connect(temp,SIGNAL(timeout()),temp,SLOT(deleteLater()));

temp->start(5000);

game->speedupamount-=1;

}

}

else if(dd<=game->player->size\_x/2+sx/2+sx&&sx<game->ratio\*game->player->size\_x){

direction=aa+3.14;

if(hasshield){

Shield \*temps=dynamic\_cast<Shield \*>((childItems())[1]);

if(temps->state==0){

hasshield=false;

temps->state=1;

temps->setOffset(15-sx/2,18.5-sy/2);

temps->setPixmap(QPixmap(":/images/gradient.png").scaled(QSize(sx,sy)));

ani=new QPropertyAnimation(temps,"opacity");

ani->setDuration(10000);

ani->setKeyValueAt(0,0);

ani->setKeyValueAt(0.125,1);

ani->setKeyValueAt(0.25,0);

ani->setKeyValueAt(0.375,1);

ani->setKeyValueAt(0.5,0);

ani->setKeyValueAt(0.625,1);

ani->setKeyValueAt(0.75,0);

ani->setKeyValueAt(0.875,1);

ani->setKeyValueAt(1,0);

ani->start();

isinvisible=true;

QTimer \*temp=new QTimer(this);

QObject::connect(temp,SIGNAL(timeout()),temps,SLOT(endlife()));

QObject::connect(temp,SIGNAL(timeout()),this,SLOT(becomeInvisible()));

QObject::connect(temp,SIGNAL(timeout()),temp,SLOT(deleteLater()));

temp->start(10000);

game->shieldamount-=1;

}

else if(hasspeedup){

Speedup \*temps=dynamic\_cast<Speedup \*>((childItems())[1]);

delete temps;

isfast=true;

hasspeedup=false;

QTimer \*temp=new QTimer;

QObject::connect(temp,SIGNAL(timeout()),this,SLOT(becomefast()));

QObject::connect(temp,SIGNAL(timeout()),temp,SLOT(deleteLater()));

temp->start(5000);

game->speedupamount-=1;

}

}

}

if(pos().x()>=-4500&&pos().x()<=4500&&pos().y()>=-2500&&pos().y()<=2500)

moveBy((v\*qCos(direction)),-(v\*qSin(direction)));

else{

qreal k=(-pos().y())/(-pos().x());

if(pos().x()<=0&&pos().y()<=0)

direction=6.28-qAtan(k);

else if(pos().x()<=0&&pos().y()>=0)

direction=-qAtan(k);

else

direction=3.14-qAtan(k);

moveBy((v\*qCos(direction)),-(v\*qSin(direction)));

}

}

}

1. 游戏一敌人的移动方向：
2. 由于地图的开阔性，角色移动的方向是任意的。定时器设置敌人的移动频率为20次每秒,转向频率为1~2.2秒一次。
3. 图解：



1. 每一个敌人都保留有一个矩形框，矩形框的大小和位置随着敌人的大小和位置的改变而改变。矩形框每经过1~2.2秒更新一次：

void Enemy::createRect(){

//create the rects

QPointF point0(scenePos().x()-rect\_x,scenePos().y()-rect\_y);

rect\_lu = new QRectF(point0,QSizeF(2\*rect\_x,2\*rect\_y));

}

void Enemy::updateRect(){

judge();

//update the rects

rect\_x=10\*sx;

rect\_y=rect\_x;

QPointF point0(scenePos().x()-rect\_x,scenePos().y()-rect\_y);

rect\_lu->setRect(point0.x(),point0.y(),2\*rect\_x,2\*rect\_y);

}

1. 其中，judge()函数完成下次方向的判断：

void Enemy::judge(){

lu=0;

QList<QGraphicsItem \*> lu\_list=game->items(game->mapFromScene(\*rect\_lu));

calculate(lu\_list,lu);

}

在这个函数中，利用QGraphicsView的items()函数获取矩形框的图形项列表。根据列表选择下一次移动的方向。Calculate函数：

void Enemy::calculate(QList<QGraphicsItem \*> &list){

for(int i=0,n=list.size();i<n;i++){

if(typeid(\*(list[i]))==typeid(QGraphicsTextItem)||typeid(\*(list[i]))==typeid(TimeCounter)

||typeid(list[i])==typeid(Weight)||list[i]->opacity()<1)

continue;

else if(typeid(\*(list[i]))==typeid(Player)){

temp=dynamic\_cast<Player \*>(list[i]);

if(sx<temp->size\_x){

direction=Angle(pos().x(),pos().y(),temp->pos().x(),temp->pos().y())+3.14;

}

else if(sx>temp->size\_x){

direction=Angle(pos().x(),pos().y(),temp->pos().x(),temp->pos().y());

}

}

else if(typeid(\*(list[i]))==typeid(Enemy)){

tempp=dynamic\_cast<Enemy \*>(list[i]);

if(tempp->id==id)

continue;

if(sx<tempp->sx){

score-=int(100\*sx);

direction=Angle(pos().x(),pos().y(),tempp->pos().x(),tempp->pos().y())+3.14;

return;

}

else if(sx>tempp->sx){

score+=int(1000\*tempp->sx);

direction=Angle(pos().x(),pos().y(),tempp->pos().x(),tempp->pos().y());

return;

}

}

else if(typeid(\*(list[i]))==typeid(Cloud)){

tempc=dynamic\_cast<Cloud \*>(list[i]);

if(sx<75){

direction=Angle(pos().x(),pos().y(),tempc->pos().x(),tempc->pos().y())+3.14;

return;

}

else{

direction=Angle(pos().x(),pos().y(),tempc->pos().x(),tempc->pos().y());

return;

}

}

else if(typeid(\*(list[i])) == typeid(QGraphicsPixmapItem)&&list[i]->opacity()==1

&&typeid(\*(list[i])) != typeid(Player)&&typeid(\*(list[i])) != typeid(Enemy)){

direction=Angle(pos().x(),pos().y(),list[i]->pos().x(),list[i]->pos().y());

return;

}

else{

direction=Angle(pos().x(),pos().y(),list[i]->pos().x(),list[i]->pos().y());

return;

}

}

direction=(qrand()%629)/100;

return;

}

1. 游戏一球团生成：

（1）球团，包括孢子，星星和星云，都是由定时器控制生成，数目始终维持在一定范围内，保证游戏的可玩性。

（2）生成函数：

void Player::createfood(){

if(foodamount>=1000)

;

else{

for(int i=0;i<100;i++){

qreal radias=4.0;

qreal x=qrand()%9000-4500;

qreal y=qrand()%5000-2500;

QRectF rect(x, y, 2\*radias, 2\*radias);

food=new QGraphicsEllipseItem();

game->scene->addItem(food);

QRadialGradient r(QPointF(x,y),radias,QPointF(x-0.5,y+0.5));

r.setColorAt(0,QColor(255,255,255,255));

r.setColorAt(1,QColor(qrand()%256,qrand()%256,qrand()%256,255));

food->setBrush(r);

food->setRect(rect);

food->setCacheMode(QGraphicsItem::ItemCoordinateCache);

}

foodamount+=100;

}

if(foodstaramount>=200)

;

else{

for(int i=0;i<20;i++){

qreal radias=30;

foodstar=new QGraphicsPixmapItem();

game->scene->addItem(foodstar);

foodstar->setPixmap(QPixmap(":/images/food.png").scaled(radias,radias));

foodstar->setPos(qrand()%9000-4500, qrand()%5000-2500);

foodstar->setCacheMode(QGraphicsItem::ItemCoordinateCache);

}

foodstaramount+=20;

}

}

1. 游戏一碰撞检测：
2. 重定义Qt视图框架QGraphicsItem类的边界矩形函数[virtual] QRectF QGraphicsPixmapItem::boundingRect() const和形状函数[virtual] QPainterPath QGraphicsPixmapItem::shape() const；
3. 利用collidingItems()函数获取某一时刻发生碰撞的图形项列表，并借助C++typeid函数判断碰撞物体的数据类型，从而实现不同类型对象发生碰撞后作出不同的反馈。部分代码：

for (int i = 0, n = colliding\_items.size(); i < n; ++i){

if (typeid(\*(colliding\_items[i])) == typeid(QGraphicsEllipseItem)&&

colliding\_items[i]->opacity()==1&&typeid(\*(colliding\_items[i])) != typeid(Bubble)){

eatfood->setPosition(0.0);

eatfood->play();

// increase the weight

increase(5);

game->weight->showWeight();

if(colliding\_items[i]->scene()!=NULL&&scene()!=NULL){

foodamount-=1;

delete colliding\_items[i];

}

}

... ...

}

1. 游戏一视差滚动：

利用定时器和QGraphicsView的centerOn()函数修改每一帧视口的位置。由于centerOn()不是粘性的，需要一直调用来使玩家在一定范围内保持在视角中心，所以在低帧率的情况下，画面会出现明显的不连贯。

1. 游戏一画面绘制：
2. 基于视图框架的2D绘图：视图框架包括一个事件传播架构，支持场景中的图元进行精确的双精度交互功能。图元可以处理键盘事件、鼠标按下、移动、释放和双击事件，同时也能跟踪鼠标移动。图形视图使用一个BSP树，以提供对图形元素的快速查找，正因为如此，它可以使超大的场景实时地可视化，即使包含数百万的图元。
3. Game类继承自QGraphicsView，用于大量自定义2D图元的管理与交互，并拥有一个QGraphicsScene对象，作为容纳大量图形项的容器。
4. 玩家，敌人，星云等均继承自QObject和QGraphicsPixmapItem，通过接口函数添加进Game类的场景（scene）中，实现交互。
5. 游戏一数据显示功能：
6. 体重的显示：

void Weight::showWeight(){

setPlainText(QString("Weight: ") + QString::number(game->player->playerWeight)+QString(" g"));

}

1. 时间的显示：

TimeCounter::TimeCounter(QGraphicsItem \* parent):QGraphicsTextItem(parent){

setPlainText(QString("Time: ") + QString::number(game->ti\_l\*60)+QString(" s"));

setDefaultTextColor(Qt::green);

setFont(QFont("times",16));

count=0;

}

void TimeCounter::showtime(){

count++;

setPlainText(QString("Time: ") + QString::number(game->ti\_l\*60-count)+QString(" s"));

}

1. 排行榜显示：

void Billboard::placeBillboard(){

int insertion=0;

for(int i=0;i<billboard.size();i++){

if(game->player->playerWeight>=billboard[i]->weight){

insertion=i;

break;

}

insertion=billboard.size();

}

for(int i=0;i<insertion;i++){

text[i]->setPlainText(QString::number(i+1)+QString(": ")+

billboard[i]->name+QString(" ")+QString::number(billboard[i]->weight)+QString(" g"));

game->scene->addItem(text[i]);

text[i]->setZValue(200);

text[i]->setPos(game->mapToScene(1550,10+i\*25));

}

text[insertion]->setPlainText(QString::number(insertion+1)+QString(": ")+

game->player->name+QString(" ")+QString::number(game->player->playerWeight)+QString(" g"));

game->scene->addItem(text[insertion]);

text[insertion]->setZValue(200);

text[insertion]->setPos(game->mapToScene(1550,10+insertion\*25));

if(insertion!=billboard.size()){

for(int i=insertion+1;i<billboard.size();i++){

text[i]->setPlainText(QString::number(i+1)+QString(": ")+

billboard[i-1]->name+QString(" ")+QString::number(billboard[i-1]->weight)+QString(" g"));

game->scene->addItem(text[i]);

text[i]->setZValue(200);

text[i]->setPos(game->mapToScene(1550,10+i\*25));

}

}

}

void Billboard::showresult(){

game->interface=game->GAMEOVER;

result=new QGraphicsPixmapItem;

result->setPixmap(QPixmap(":/images/billboard.png"));

game->scene->addItem(result);

result->setPos(-200,-250);

result->setZValue(19);

int insertion=0;

for(int i=0;i<billboard.size();i++){

if(game->player->playerWeight>=billboard[i]->weight){

insertion=i;

break;

}

insertion=billboard.size();

}

if(insertion<8){

for(int i=0;i<insertion;i++){

text[i]->setPlainText(billboard[i]->name+QString(" ")+QString::number(billboard[i]->weight)+QString(" g"));

text[i]->setPos(-140,-190+i\*50);

}

text[insertion]->setPlainText(game->player->name+QString(" ")+QString::number(game->player->playerWeight)+QString(" g"));

text[insertion]->setPos(-140,-190+insertion\*50);

if(insertion!=7){

for(int i=insertion+1;i<billboard.size();i++){

if(i<8){

text[i]->setPlainText(billboard[i-1]->name+QString(" ")+QString::number(billboard[i-1]->weight)+QString(" g"));

text[i]->setPos(-140,-190+i\*50);

}

}

}

}

else{

for(int i=0;i<billboard.size()&&i<8;i++){

text[i]->setPlainText(billboard[i]->name+QString(" ")+QString::number(billboard[i]->weight)+QString(" g"));

text[i]->setPos(-140,-190+i\*50);

}

}

}

1. 游戏一排名功能：快速排序与Qt容器QVector的结合：

void sort(QVector<Enemy\*> &vec, int low, int high){

if (low < high){

int l = low;

int r = high;

int key = vec[l]->weight;

Enemy \*temp=vec[l];

while (l < r){

while (l < r&&key >= vec[r]->weight)

--r;

vec[l] = vec[r];

while (l < r&&key <= vec[l]->weight)

++l;

vec[r] = vec[l];

}

vec[l] = temp;

sort(vec, low, l-1);

sort(vec, r + 1, high);

}

}

1. 游戏二菜单：由Game类的void Game::menu2()函数生成：

void Game::menu2(){

menuboard=new QGraphicsPixmapItem;

three=new QGraphicsPixmapItem;

four=new QGraphicsPixmapItem;

five=new QGraphicsPixmapItem;

six=new QGraphicsPixmapItem;

home2=new QGraphicsPixmapItem;

menuboard->setPixmap(QPixmap(":/images/menu.png"));

three->setPixmap(QPixmap(":/images/three.png"));

four->setPixmap(QPixmap(":/images/four.png"));

five->setPixmap(QPixmap(":/images/five.png"));

six->setPixmap(QPixmap(":/images/six.png"));

home2->setPixmap(QPixmap(":/images/home.png"));

scene->addItem(menuboard);

scene->addItem(three);

scene->addItem(four);

scene->addItem(five);

scene->addItem(six);

scene->addItem(home2);

menuboard->setPos(590,92);

three->setPos(690,242);

four->setPos(690,392);

five->setPos(690,542);

six->setPos(690,692);

home2->setPos(875,812);

three->setTransformOriginPoint(210,55);

four->setTransformOriginPoint(210,55);

five->setTransformOriginPoint(210,55);

six->setTransformOriginPoint(210,55);

home2->setOpacity(0.01);

interface2=MENU;

}

1. 游戏二操作的实现：由键盘事件void Game::keyPressEvent(QKeyEvent \*event)进行判断和响应，以下是3x3模式操作的实现：

if(interface2==GAME3){

sliding->setPosition(0.0);

sliding->play();

if(lastkey!=1&&event->key()==Qt::Key\_Up){

buttons[0]->setOpacity(1.0);

lastkey=0;

if(empty>=7&&empty<=9)

return;

else{

if(empty%3!=0&&itemAt(700+200\*(empty%3-1),300+200\*(empty/3)+200)!=NULL){

animation=new QPropertyAnimation(dynamic\_cast<GraphicsItem \*>(itemAt(700+200\*(empty%3-1),300+200\*(empty/3)+200)),"pos");

animation->setDuration(timedelay);

animation->setKeyValueAt(0,itemAt(700+200\*(empty%3-1),300+200\*(empty/3)+200)->pos());

animation->setKeyValueAt(1,QPointF(615+200\*(empty%3-1),220+200\*(empty/3)));

animation->start();

empty+=3;

}

else if(itemAt(700+2\*200,300+200\*(empty/3-1)+200)!=NULL){

animation=new QPropertyAnimation(dynamic\_cast<GraphicsItem \*>(itemAt(700+2\*200,300+200\*(empty/3-1)+200)),"pos");

animation->setDuration(timedelay);

animation->setKeyValueAt(0,itemAt(700+2\*200,300+200\*(empty/3-1)+200)->pos());

animation->setKeyValueAt(1,QPointF(615+2\*200,220+200\*(empty/3-1)));

animation->start();

empty+=3;

}

}

}

else if(lastkey!=0&&event->key()==Qt::Key\_Down){

buttons[1]->setOpacity(1.0);

lastkey=1;

if(empty>=1&&empty<=3)

return;

else{

if(empty%3!=0&&itemAt(700+200\*(empty%3-1),300+200\*(empty/3)-200)!=NULL){

animation=new QPropertyAnimation(dynamic\_cast<GraphicsItem \*>(itemAt(700+200\*(empty%3-1),300+200\*(empty/3)-200)),"pos");

animation->setDuration(timedelay);

animation->setKeyValueAt(0,itemAt(700+200\*(empty%3-1),300+200\*(empty/3)-200)->pos());

animation->setKeyValueAt(1,QPointF(615+200\*(empty%3-1),220+200\*(empty/3)));

animation->start();

empty-=3;

}

else if(itemAt(700+2\*200,300+200\*(empty/3-1)-200)!=NULL){

animation=new QPropertyAnimation(dynamic\_cast<GraphicsItem \*>(itemAt(700+2\*200,300+200\*(empty/3-1)-200)),"pos");

animation->setDuration(timedelay);

animation->setKeyValueAt(0,itemAt(700+2\*220,300+200\*(empty/3-1)-200)->pos());

animation->setKeyValueAt(1,QPointF(615+2\*200,220+200\*(empty/3-1)));

animation->start();

empty-=3;

}

}

}

else if(lastkey!=3&&event->key()==Qt::Key\_Left){

buttons[2]->setOpacity(1.0);

lastkey=2;

if(empty%3==0)

return;

else if(itemAt(700+200\*(empty%3-1)+200,300+200\*(empty/3))!=NULL){

animation=new QPropertyAnimation(dynamic\_cast<GraphicsItem \*>(itemAt(700+200\*(empty%3-1)+200,300+200\*(empty/3))),"pos");

animation->setDuration(timedelay);

animation->setKeyValueAt(0,itemAt(700+200\*(empty%3-1)+200,300+200\*(empty/3))->pos());

animation->setKeyValueAt(1,QPointF(615+200\*(empty%3-1),220+200\*(empty/3)));

animation->start();

empty+=1;

}

}

else if(lastkey!=2&&event->key()==Qt::Key\_Right){

buttons[3]->setOpacity(1.0);

lastkey=3;

if(empty%3==1)

return;

else{

if(empty%3!=0&&itemAt(700+200\*(empty%3-1)-200,300+200\*(empty/3))!=NULL){

animation=new QPropertyAnimation(dynamic\_cast<GraphicsItem \*>(itemAt(700+200\*(empty%3-1)-200,300+200\*(empty/3))),"pos");

animation->setDuration(timedelay);

animation->setKeyValueAt(0,itemAt(700+200\*(empty%3-1)-200,300+200\*(empty/3))->pos());

animation->setKeyValueAt(1,QPointF(615+200\*(empty%3-1),220+200\*(empty/3)));

animation->start();

empty-=1;

}

else if(itemAt(700+2\*200-200,300+200\*(empty/3-1))!=NULL){

animation=new QPropertyAnimation(dynamic\_cast<GraphicsItem \*>(itemAt(700+2\*200-200,300+200\*(empty/3-1))),"pos");

animation->setDuration(timedelay);

animation->setKeyValueAt(0,itemAt(700+2\*200-200,300+200\*(empty/3-1))->pos());

animation->setKeyValueAt(1,QPointF(615+2\*200,220+200\*(empty/3-1)));

animation->start();

empty-=1;

}

}

}

timer=new QTimer;

QObject::connect(timer,SIGNAL(timeout()),helper,SLOT(resetKey()));

timer->start(timedelay);

}

其中，animation实现过渡动画，empty记录空格位置。

1. 游戏二结果判定：通过bool函数isfinished()判定结果并返回bool值

bool Game::isfinished(){

for(int i=1;i<blocks.size();i++){

if(gamenum==GAME3){

if(i%3!=0){

if(dynamic\_cast<GraphicsItem\*>(itemAt(700+200\*(i%3-1),300+200\*(i/3)))!=blocks[i])

return false;

}

else{

if(dynamic\_cast<GraphicsItem\*>(itemAt(700+2\*200,300+200\*(i/3-1)))!=blocks[i])

return false;

}

}

else if(gamenum==GAME4){

if(i%4!=0){

if(dynamic\_cast<GraphicsItem\*>(itemAt(675+150\*(i%4-1),275+150\*(i/4)))!=blocks[i])

return false;

}

else{

if(dynamic\_cast<GraphicsItem\*>(itemAt(675+3\*150,275+150\*(i/4-1)))!=blocks[i])

return false;

}

}

else if(gamenum==GAME5){

if(i%5!=0){

if(dynamic\_cast<GraphicsItem\*>(itemAt(660+120\*(i%5-1),260+120\*(i/5)))!=blocks[i])

return false;

}

else{

if(dynamic\_cast<GraphicsItem\*>(itemAt(660+4\*120,260+120\*(i/5-1)))!=blocks[i])

return false;

}

}

else if(gamenum==GAME6){

if(i%6!=0){

if(dynamic\_cast<GraphicsItem\*>(itemAt(650+100\*(i%6-1),250+100\*(i/6)))!=blocks[i])

return false;

}

else{

if(dynamic\_cast<GraphicsItem\*>(itemAt(650+5\*100,250+100\*(i/6-1)))!=blocks[i])

return false;

}

}

}

return true;

}

1. 游戏二数字方块的打乱：

首先，数字华容道的数字不是随机打乱就可以的，因为偶数阶情况下，空格的初始位置和数字的逆序数决定了此排列是否有解，最终结论是：

1. 奇数阶，移动方块不改变奇偶性，所以空格的位置任意；
2. 偶数阶，移动方块会改变奇偶性，当初始序列为偶排列时，空格所在行与游戏成功时最终空格所在行之差必须为偶数，否则为奇数。
3. 以下两个函数分别用于打乱数组和计算逆序数：

void Game::mixarray(QVector<int> &a){

int first, second, temp;

for(int i=1;i<a.size();i++){

first=qrand()%(a.size()-1)+1;

do{second=qrand()%(a.size()-1)+1;}while(second==first);

temp=a[first];

a[first]=a[second];

a[second]=temp;

}

}

void Game::countinverse(QVector<int> &a){

for(int i=1;i<a.size()-1;i++){

for(int j=i+1;j<a.size();j++){

if(a[i]>a[j])

inverse++;

}

}

}

# 总结

本次实训，总的来说基本完成了任务要求，取得了不错的成果。

1. 为了做出好看的界面，花了将近半个月的时间学习Qt，初步了解了Qt窗口部件，布局管理，事件系统，界面外观，2D绘图，图形视图框架，动画框架，状态机框架，3D绘图，多媒体播放，多线程等核心知识，并基本上掌握了运用图形视图框架开发游戏的能力。对比于之前学习的简单的图形库easyX，Qt提供了更加强大的框架，用于C++程序的开发。让我印象深刻的，是Qt的信号与槽机制，因为它使得各个类的交互更加便捷和安全。以及Qt的QTimer类，因为它可以实例化为任意间隔的定时器，与Qt事件循环系统相辅相成，从而开发出功能更为复杂多样的程序。但是，由于我是现学现用，很多知识点都是浅尝辄止，并没有完全掌握就急于应用，所以这也导致了开发过程中遇到了很多小问题。比如：
2. 视口的位置移动问题。刚开始，我没有阅读Qt Creator自带的帮助文档，局限于书上介绍的内容，因此很多QGraphicsView的接口函数都不了解。由于我想实现玩家在一个大地图中移动的时候形成视差滚动的效果，于是我一开始就想，视图框架这么强大，应该会有相关的函数用于移动视口在场景中的坐标，结果找了好几天都没有头绪，网上也查了很多，都没有相关的解决方案。当场景远大于视口的时候，视口默认显示场景矩形的中心，并出现滚动条，滚动条用于手动调整视口的位置，但我想实现视口自动滚动。我想出的第一个解决方案是动态地改变场景矩形的原点，这样，视口坐标和场景坐标的映射发生了改变，但实际上，视口依然默认显示场景矩形的中心，也就是相当于没有发生移动。第二次，我利用一个图形项作为背景，当角色接收到移动的命令时，使背景向相反方向移动，这样确实实现了视差滚动，但由于背景时刻在移动，这种情况下，当除玩家外的其他角色移动时，由于运动的相对性会出现明显的抖动，界面看着很不舒服。第三次，我在帮助文档中找到了centerOn()函数，可以改变视口的显示中心，但是这个函数不是粘性的，因此我又设置了一个定时器，以一定的频率调整视口位置。这样，这个问题基本解决。
3. 角色的抖动问题。前面提到，我用定时器定时改变视口的显示中心，从而实现了视差滚动。然而这个方案又有另一个弊端，就是当刷新频率过低时，如果玩家角色的偏转角（下一次转向的偏转角度，以x轴正方向为零度）过大时，在转向的瞬间，角色会出现明显的抖动。我认为这个问题就是由centerOn()函数引起的，即centerOn()引起了转向前后两个状态中角色的位置偏差。为了减少这种抖动，我尽量调高了定时器的频率。这样做，很明显动画效果更加连贯了。可是，同时它也带来了另一个问题。
4. 定时器问题。接上一个问题，为了画面的连贯，我强行把定时器的刷新频率提高到每秒100次（很明显这是个错误的方案），当时我并不知道这样做会带来什么严重的后果。在那之后，我发现游戏画面突然卡到不行，而且让我觉得匪夷所思是，就这么一个小程序，在任务管理器中看到CPU占用率居然高达10%，怎么调试，修改都不行，好几次都想放弃了。大约折腾了好几天，我突然想起来把定时器时间加长，降低刷新频率，发现果然不卡顿了。然而，这样游戏画面又出现了明显的抖动。
5. 还有一个比较棘手的问题，视图框架和窗口部件不能混合在一起使用。一开始我在帮助文档中找到了一个代理类QGraphicsProxyWidget，可以把窗口部件（比如QPushButton，QComboBox）“转化”成图形项，从而在场景中使用。本来这个确实很有用，但是，由于我在Game类中重定义了鼠标事件和键盘事件，使得QComboBox的输入功能失效了，这样我不得不弃用代理，自己实现所需的按钮部件，输入框，和选择对话框。
6. 还有一些动画框架的问题，内存管理的问题，在此就不多赘述了。
7. 心得体会：
8. C++结合Qt可以开发出很多优秀的窗口程序，当然也能开发出一些游戏。但众所周知，Qt只是一个图形库，不是游戏引擎，相比于Unity3D，cocos2dx还是有很多不足，对于像我这样的新手来说，用Qt来开发游戏难度确实有点大。但是我并不后悔选择Qt，难度越大越是具有挑战性，越是能学到更多底层的东西。
9. 游戏开发尤其需要更好的框架和架构，而不能像这次实训那样从各个具体模块入手，那样的后果就是见效快但是问题也多，增大了后期的编码难度（如果架构得不好）。
10. 学习相关的开发平台，一定要认真阅读帮助文档，不能马虎。很多时候，许多小问题通过帮助文档就能轻松解决，却花了大量时间在网上找答案，得不偿失。
11. 对于C++指针的应用，内存管理，类的继承，Qt的事件系统，信号和槽系统有了更加深刻的认识，为以后的程序开发积累了大量的经验。
12. 独立开发很能锻炼一个人的实战能力和解决问题的能力，但时间成本也更大。这次实训很有意义，希望以后还有更多这样的实训。
13. **参考文献**

霍亚飞. Qt Creator快速入门[M]. 北京:北京航天航空大学出版社, 2017. 1-515