南开大学 计算机大类

姓名：申宗尚

学号：2213924

班级：计算机科学卓越班

2023年5月11日

高级语言程序设计

实验报告

目录

[高级语言程序设计大作业实验报告 1](#_Toc21286)

[一. 作业题目 1](#_Toc29969)

[二. 开发软件 1](#_Toc18364)

[三. 课题要求 1](#_Toc742)

[四. 主要流程 1](#_Toc25240)

[1． 整体流程 2](#_Toc20452)

[2． 实现步骤 2](#_Toc25354)

[（1） 图片类 2](#_Toc22508)

[（2） 水管、鸟、地板图元 2](#_Toc28665)

[（3） 跳跃动作 5](#_Toc32428)

[（4） 碰撞判定 6](#_Toc23860)

[（5） 计分、结束系统 9](#_Toc13349)

[3． 单元测试 9](#_Toc24674)

[五. 单元测试 10](#_Toc28265)

[六. 收获 10](#_Toc23116)

[1． Qt的图形视图框架 10](#_Toc10799)

[2． QPropertyAnimation 11](#_Toc16275)

高级语言程序设计大作业实验报告

1. **作业题目**

经典游戏《Flappy Bird》Qt实现。

1. **开发软件**

Qt Creator 4.11.1(community)

Qt 5.14.2 MinGW 32-bit

1. **课题要求**
2. 面向对象。
3. 单元测试。
4. 模型部分
5. 验证
6. **主要流程**
   1. **整体流程**

实现方式：运用Qt图形视图框架中的场景类QGraphicsScene和图元类QGraphicsItem组合实现游戏。将静止的图片设置为QGraphicsPixmapItem置于游戏中，而对于有特殊动作、判定、函数的（如小鸟，管道，地板等）设置为图元类，再通过对具体类制作具体函数来实现整体游戏的运行。

* 1. **实现步骤**
     1. **图片类**

对于静止不动的图片类，通过制造QGraphicsPixmapItem将图片放置在主scene中，最后通过在ui中对视图框使用setScene()函数使其统一显示，具体代码示例如下：

对于静止不动的startpage，gameoverpage，background，NKU sign，均使用该代码（以startpage为例）

startImage = new QGraphicsPixmapItem(QPixmap(":/new/prefix1/start.png"));

addItem(startImage);

startImage->setPos(0,0);

* + 1. **水管、鸟、地板图元**

对于有特殊函数、判定的水管、鸟、地板，通过设置单独的继承于QObject和QGraphicsPixmapItem的类pipeitem，birditem，grounditem来进行对应的实现，以下分别说明。

对于水管，要实现其等时间生成，随机高度，并向左移动，这里使用QTimer实现等时间生成

pipetimer = new QTimer(this);

connect(pipetimer,&QTimer::timeout,[=](){

PipeItem\* pipe = new PipeItem;

使用Srand随机种子实现高度的随机

srand((unsigned)time(NULL)；

ramy=-(200+rand()%(480-200+1));

apipe->setPos(500,ramy);

使用QPropertyAnimation和Linear的QEasingCurve实现匀速向左运动

xani = new QPropertyAnimation(this,"x",this);

xani->setStartValue(480);

xani->setEndValue(-200);

xani->setEasingCurve(QEasingCurve::Linear);

xani->setDuration(3000);

connect(xani, &QPropertyAnimation::finished, [=](){

scene()->removeItem(this);

delete this;

});

xani->start();

addToGroup(apipe);

至此，完成水管图元的设置。

对于地板图元，对于地板图元，只需要有一个不断向左运动的动画，所以还是使用QPropertyAnimation，但加了一个setLoopCount

groundani = new QPropertyAnimation(this, "groundx", this);

groundani->setStartValue(0);

groundani->setEndValue(-100);

groundani->setDuration(1000);

groundani->setLoopCount(-1);

groundani->start();

至此，完成地板图元的设置

对于小鸟图元，主要完成两个功能，一是小鸟扇动翅膀，二是小鸟的自然坠落。对于扇动翅膀，通过设置QTimer，对小鸟翅膀三种情况进行循环替换图元图片的方式实现。

QTimer\* wingtimer = new QTimer(this);

connect(wingtimer,&QTimer::timeout,[=](){wings();});

wingtimer->start(100);

void birditem::**wings**()

{

if(wingpos==0)

{

setPixmap(QPixmap(":/new/prefix1/1.png"));

wingpos=1;

wingdirect=0;

}

if(wingpos==1)

{

if(wingdirect==0)

{

setPixmap(QPixmap(":/new/prefix1/2.png"));

wingpos=2;

}

else

{

setPixmap(QPixmap(":/new/prefix1/0.png"));

wingpos=0;

}

}

if(wingpos==2)

{

setPixmap(QPixmap(":/new/prefix1/1.png"));

wingpos=1;

wingdirect=1;

}

}

对于自然坠落，通过QPropertyAnimation和InQuad的QEasingCurce实现

//设置鸟的坠落

yani = new QPropertyAnimation(this,"y",this);

yani->setStartValue(scenePos().y());

yani->setEndValue(groundline);

yani->setEasingCurve(QEasingCurve::InQuad);

yani->setDuration(1000);

//设置坠落时的体态——嘴朝地

rotationani = new QPropertyAnimation(this,"rotation",this);

rotationani->setStartValue(rotation());

rotationani->setEndValue(90);

rotationani->setEasingCurve(QEasingCurve::InQuad);

rotationani->setDuration(1200);

至此，完成对三个图元的基础设置。

* + 1. **跳跃动作**

在游戏内需要通过按空格键使小鸟向上跳跃，所以通过写一个跳跃函数并connect到电脑输入空格实现，同时，又因为跳跃后应接着坠落，所以还应再写一个坠落函数，在跳跃函数执行完后后执行，而跳跃函数和坠落函数内容都可以用QPropertyAnimation实现。

跳跃函数：void birditem::**jump**()

{

yani->stop();

rotationani->stop();

//让鸟跳跃完成后自然下坠

connect(yani,&QPropertyAnimation::finished,[=]{falling();});

//设置鸟的跳跃动画

yani->setStartValue(y());

yani->setEndValue(y() - 75);

yani->setEasingCurve(QEasingCurve::OutQuad);

yani->setDuration(255);

yani->start();

//设置鸟的跳跃旋转动画

rotationani->setStartValue(rotation());

rotationani->setEndValue(-20);

rotationani->setEasingCurve(QEasingCurve::InOutQuad);

rotationani->setDuration(200);

rotationani->start();

}

坠落函数：void birditem::**falling**()

{

if(y()<(this->scenePos().y()+450))

{

rotationani->stop();

yani->setStartValue(y());

yani->setEndValue(groundline);

yani->setEasingCurve(QEasingCurve::InQuad);

yani->setDuration(1100);

yani->start();

rotationani->setStartValue(rotation());

rotationani->setEndValue(90);

rotationani->setEasingCurve(QEasingCurve::InQuad);

rotationani->setDuration(1000);

rotationani->start();

}

}

连接空格与跳跃函数：void Scene::***keyPressEvent***(QKeyEvent \*event)

{

if(startsign==0)

mainstart();

//第一次空格将移除开始界面图片，并让小鸟，水管开始运动

if(!gameoverbool)

{

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

bird->jump();

}

//若游戏已结束 则空格不再有任何作用

}

QGraphicsScene::keyPressEvent(event);

}

* + 1. **碰撞检测**

对于碰撞检测，采用Qt自带的collidingItem函数，需要对水管和地面都进行碰撞检测，将其写在水管的setx（）Property函数中，而写在小鸟的sety（）Property函数中，从而水管的x，小鸟的y每更新一次，都会检测一次collidingItem函数，判断是否与小鸟/地板相撞，若相撞，就发出碰撞信号，通过connect碰撞信号与gameover函数，来实现游戏的碰撞检测。

void birditem::**sety**(qreal y)

{

setPos(this->pos().x(),y);

//若碰撞地板 则发出碰撞信号

if(collision()){

emit collidesignal2();

}

m\_y = y;

}

//碰撞检测函数

bool birditem::**collision**()

{

QList<QGraphicsItem\*> collidingItems = this->collidingItems();

foreach (QGraphicsItem \* item , collidingItems){

groundItem \* groundtemp = dynamic\_cast<groundItem\*>(item);

if(groundtemp){

return true;

}

}

return false;

}

//如果小鸟撞上了管道，发出碰撞信号

if(collision()){

emit collidesignal();

}

bool PipeItem::**collision**()

{

QList<QGraphicsItem\*> collidingItems = apipe->collidingItems();

foreach (QGraphicsItem \* item , collidingItems){

birditem \* birdie = dynamic\_cast<birditem\*>(item);

if(birdie){

return true;

}

}

return false;

}

而对于信号检测，//如果小鸟与水管碰撞，游戏结束

connect(pipe,&PipeItem::collidesignal,[=](){

pipetimer->stop();

gameover();

});

//如果小鸟与地板碰撞，游戏结束

connect(bird,&birditem::collidesignal2,[=](){

gameover();

});

最后，写gameover函数：

void Scene::**gameover**()

{

gameoverbool=1; //已结束游戏

bird->birdstop(); //鸟停止运动

ground->groundstop(); //地板停止运动

showscore();

addItem(gameoverImage);

gameoverImage->setPos(0,0);

gameoverImage->setZValue(100);

addItem(nankaiImage);

nankaiImage->setPos(138,400);

nankaiImage->setZValue(100);

//显示游戏结束和校徽画面

//将画面内所有水管都停止运动

QList<QGraphicsItem\*> sceneItems = items();

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

PipeItem \* pipe = qgraphicsitem\_cast<PipeItem\*>(sceneItems[i]);

if(pipe){

pipe->pipestop();

}

}

pipetimer->stop();

//停止水管计时器 不再生成新水管

}

* + 1. **计分、结束系统**

对于计分，写在水管的setx（）Property函数中，如上，可以达到每更新一次x判断是否通过鸟，从而积分的目的。

//如果小鸟通过了管道，增加1分

if(x<0&&!pass){

pass=1;

QGraphicsScene\* nowscene=scene();

Scene\* myscene = dynamic\_cast<Scene \*>(nowscene);

if(nowscene){

myscene->Scoreadd();

}

}

而对于结束系统，只需要在上面gameover函数中再写一个showscore函数，把计算的总积分显示出来即可：

void Scene::**showscore**()

{

scoretext = new QGraphicsTextItem();

QString lastscore="Your Score:" + QString::number(score);

scoretext->setHtml(lastscore);

//设置分数显示界面字体和颜色

QFont font("Consolas",20,QFont::Bold);

scoretext->setFont(font);

QColor color(126,12,110);

scoretext->setDefaultTextColor(color);

//设置位置

addItem(scoretext);

scoretext->setPos(35 ,280);

}

至此，整个游戏基本就完成了实现。

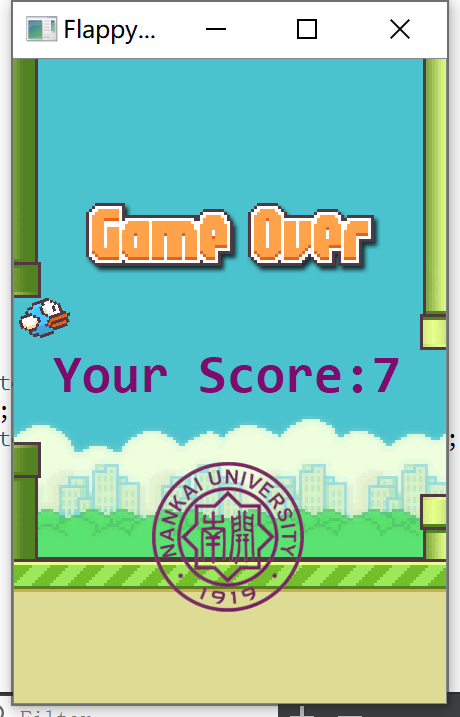
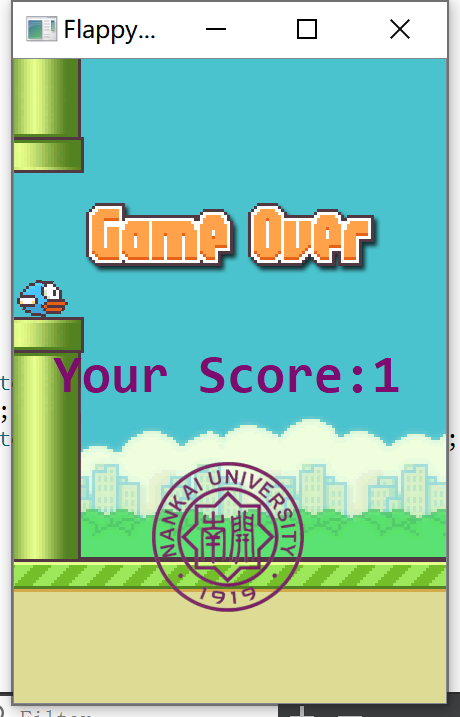
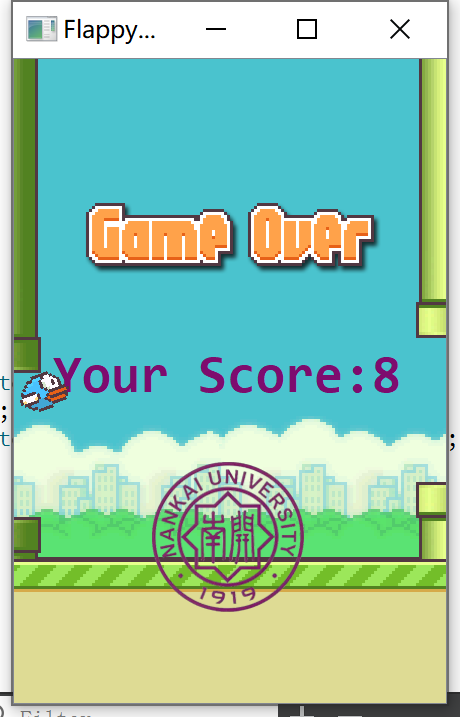
* 1. **单元测试**

玩家进行数次游戏的游玩，判断是否与经典游戏《Flappy Bird》一致，是否出现明显的bug或错误。

1. **单元测试**

测试结果

经过数名玩家的数次游玩，游戏都可以正常运行，完成基本操作，并未出现bug或错误，可以得出结论，该程序可以较好地模拟实现经典游戏《Flappy Bird》



1. **收获**
   1. **Qt的图形视图框架**

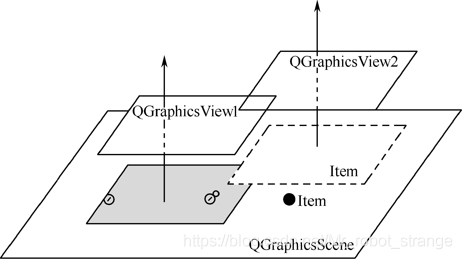
Graphics View框架结构的主要特点如下：

（1）、在Graphics View框架结构中，系统可以利用Qt绘图系统的反锯齿、OPenGL工具来改善绘图的性能。

（2）、Graphics View支持时间传播体系结构，可以使图元在场景（scene）中的交互能力提高1倍，图元能够处理键盘事件和鼠标事件。其中，鼠标事件包括鼠标被按下、移动、释放和双击，还可以跟踪鼠标的移动。

（3）、在Graphics View框架中，通过二元控件划分树（Binary Space Partitioning，BSP）提供快速的图元查找，这样就能够实时地显示包含上百万个图元的大场景。

Graphics View框架结构主要包含三个类，即场景类（QGraphicsScene）、视图类（QGraphicsView）和图元类（QGraphicsItem），统称为“三元素”。其中，场景类提供了一个用于管理位于其中的众多图元容器，视图类用于显示场景中的图元，一个场景可以通过多个视图表现，一个场景包括多个几何图形。Graphics View三元素之间的关系如下图所示：



在本例中，Qt的Graphics View框架结构在封装、分别实现功能等多方面表现优异，是完成该作品的重要支柱。

* 1. **QPropertyAnimation**

QPropertyAnimation 对 Qt 属性进行插值。由于属性值存储在 **QVariant** 中，该[类继承](https://so.csdn.net/so/search?q=%E7%B1%BB%E7%BB%A7%E6%89%BF&spm=1001.2101.3001.7020)了 **QVariantAnimation**，并支持与其超类相同元类型的动画。

在本例子中，QPropertyAnimation完成了大部分游戏中图元的运动，值得注意的是，在运用过程中，有以下两点需要注意：

1. 需要用QVariantAnimation检测你自定义的QVariant类型是否支持。
2. 声明属性的类必须是一个QObject，必须为属性提供一个setter（这样，QPropertyAnimation才可以设置属性的值）。