

# 程序设计综合实践报告

|  |  |  |  |
| --- | --- | --- | --- |
| 姓名 | 尹浩男 | 学号 | 20180105 |
| 班级 | 计算机科学与技术2018-5 | 项目名称 | qt飞机大战游戏设计与实现 |
| 实习地点 |  | 实习时间 | 2020-6-22至2020-7-3 |
| 实习成绩 |  | 指导教师签字 |  |

**计算机科学与工程学院**

2020年 7月3日

**目 录**

# 1. 概述

这次所做的项目是飞机大战游戏，主要实现了战机的移动和导弹自动发射、敌机的自动移动和子弹的随机发射、背景的滚动和自动切换。

# 2. 相关技术

## 2.1 战机的移动

通过键盘的WSAD方向键来控制战机实现上下左右移动，用到的技术是监听键盘按键，整个过程分为 ***keyPressEvent 和keyReleaseEvent，***表示键盘按下和键盘释放信号，通过qt的connect和disconnect函数来将主时钟和战机的上下左右移动函数相绑定，通过检测主时钟的脉冲频率和战机移动函数来控制战机的移动速度，config.h中的 PLANE\_SPEED宏 是用来规定战机的移动速度的。

## 2.2 随机产生敌机

因为要随机产生敌机，所以要用到随机数，在项目中我使用的是rand()函数，敌机有三个随机的地方，

1. enemyPlane类中随机决定该敌机是否是高速敌机

2. enemyPlane类中随机决定该敌机从哪个方向进入选择有两个（上方和下方）

3. 主窗口中**enemyToMainWindow()** 函数随机决定敌机出现的x坐标。

## 2.3 游戏背景

游戏的背景是滚动的，我用两张相同的图片来实现背景滚动，第一张图片的的初始位置在游戏界面里，第二张在其底下，不能被看见，图片开始滚动之后，如果第二张图片超出屏幕那么恢复初始状态。

下边是用图片解释这一过程。

黑框代表程序显示的界面，能够被用户看见

蓝色图片是第一张图片 初始是能被用户看见的 然后他开始向上移动

橘色色图片是第二张图片 初始是不能被用户看见的 然后他开始向上移动

最终被用户看见

第一张图片

第一张图片

第二张图片

第二张图片

恢复初始状态的瞬间

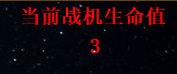
初始状态

## 2.4 击杀敌机数量板（得分板）



绘制得分板使用的是QLabel类，并且使用了该类的setStyleSheet()函数，可以用css来渲染样式，非常方便快捷 ，通过碰撞检测函数中更新killEnemyPlaneNum全局变量的值，并在主时钟每次脉冲时间进行刷新label中的数字，这样就做到了实时更新击杀敌机数量。

## 2.5 生命值显示



绘制生命值和得分板类似，战机的生命值是存储在Plane类中的 ，lifeNum 在战机被创建时通过config.h 的宏定义PLANE\_INITLIVES 来规定，我设置的是5条命。

## 2.6 碰撞检测

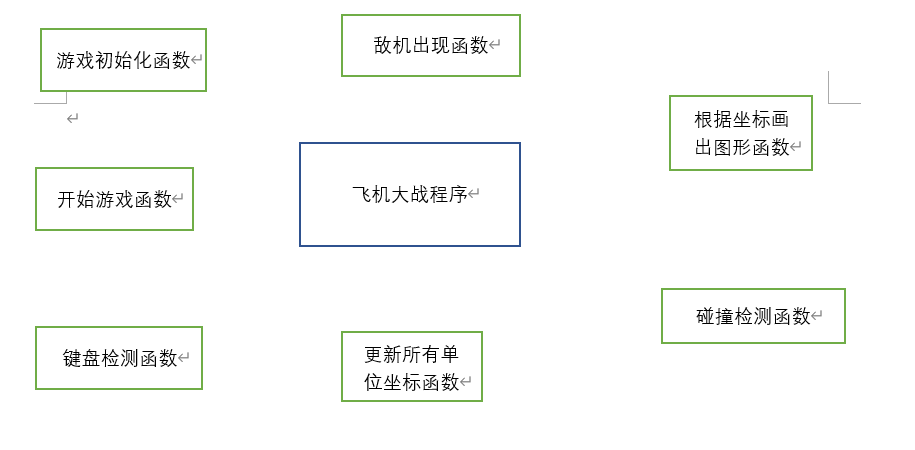
碰撞检测主要实现的原理是，通过主时钟脉冲来执行Widget类的**collisionDetection()，**通过for循环来进行三个碰撞的检测，每个有贴图的类都有一个成员是用QRect类来定义的

它的位置信息随贴图的位置更新，如果检测到一下三种情况的矩形框相交，那么就判定碰撞发生。

1. 检测战机子弹和敌机的碰撞
2. 检测战机和敌机的碰撞
3. 检测战机和敌机子弹的碰撞

# 3. 总体设计与详细设计

## 3.1 系统模块划分



# 编码实现

1. 开始游戏的函数

void Widget::playGame()

{

    timer.start(); //启动定时器

    planeInTimer.start();  // 启动进场定时器

    keyBoradTimer.start();

    enemyPlaneBulletTimer.start(); //启动敌机发射定时器

    //设置飞机的一个进场效果

    connect(&planeInTimer,&QTimer::timeout,[&](){

        if(WINDOW\_HEIGHT -  4 \* PLANE\_HEIGHT >= myPlane.getY() ){  //飞机距离底部一段距离的时候停下

            planeInTimer.stop();

        }

        myPlane.setXY(myPlane.getX(),myPlane.getY() - 5);

        myPlane.planeRect.moveTo(myPlane.getX() ,myPlane.getY() -5);

    }); //监听定时器的发送信号  使用信号槽来进行连接

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

        enemyToMainWindow(); //敌机出场

        updatePosition(); //更新游戏中所有元素的坐标

        update();   //绘制到屏幕中

        collisionDetection();//检测碰撞

        labOfEnemyPlaneKillNum->setText(QString::number(killEnemyPlaneNum));  //实时更新击杀敌机数目

        labOfPlaneLivesNum->setText(QString::number(myPlane.getLifeNum()));

        //判断敌机生命是否超时 如果超时 就暂停重新开始游戏

        if(myPlane.getLifeNum()<=0){

            timer.stop();

            keyBoradTimer.stop();

            enemyPlaneBulletTimer.stop();

            resurrectionBtn->show();

            exitButton->show();

        }

    }); //监听定时器的发送信号  使用信号槽来进行连接

    connect(&enemyPlaneBulletTimer,&QTimer::timeout,[=](){ //设置敌机子弹发射定时器

        for(int i=0;i<ENEMYBULLET\_AMOUNT ;i++){

            if(enemyBulletArray[i].isFreeOrNot() ){//说明敌机子弹有空闲

                enemyBulletArray[i].shoot();

                break;

            }

        }

    });

}

2. 碰撞检测功能的实现

void Widget::collisionDetection()  //碰撞检测

{

    for(int i=0;i<BULLET\_CAPACITY;i++){  //遍历所有子弹

        if(myPlane.clips[i].isFreeOrNot())  //子弹当前空闲

            continue;

        for(int j =0;j<ENEMYPLANE\_NUM;j++){

            if(enmeyPlaneArray[j].isFreeOrNot() )  //当前的敌机空闲

                continue;

            if(myPlane.clips[i].bulletRect.intersects(enmeyPlaneArray[j].enemyPlaneRect)){

                myPlane.clips[i].setFree(true);

                enmeyPlaneArray[j].setFree(true);

                //播放爆炸的效果

                for(int k =0;k<BOMB\_NUM;k++){

                    if(bombs[k].isFreeOrnot() ) //空闲状态

                    {

                        //空闲的爆炸

                       bombs[k].setFree(false);

                       //更新爆炸坐标

                       bombs[k].setXY(enmeyPlaneArray[j].getX(),enmeyPlaneArray[j].getY());

                       killEnemyPlaneNum++; //击杀敌机数目增加

                       break;

                    }

                }

            }

        }

    }

    for(int i=0;i<ENEMYBULLET\_AMOUNT;i++){  // 敌机子弹和战机碰撞

        if(enemyBulletArray[i].enemyBulletRect.intersects(myPlane.planeRect)

                && enemyBulletArray[i].isFreeOrNot() ==false ){  //这里两个条件都要满足 否则会产生多次爆炸

            //播放爆炸的效果

            enemyBulletArray[i].setFree(true);  //

            for(int m =0;m<BOMB\_NUM;m++){

                if(bombs[m].isFreeOrnot() ) //空闲状态

                {

                    //空闲的爆炸

                   bombs[m].setFree(false);

                   //更新爆炸坐标

                   bombs[m].setXY(myPlane.getX(),myPlane.getY());

                   myPlane.setLifeNum(myPlane.getLifeNum() - 1);  //战机生命值-1

                   break;

                }

            }

        }

    }

    for(int i=0;i<ENEMYPLANE\_NUM;i++){  // 战机和敌机碰撞

        if(enmeyPlaneArray[i].enemyPlaneRect.intersects(myPlane.planeRect)  //如果敌机和战机有相交的地方

                && enmeyPlaneArray[i].isFreeOrNot() ==false ){

            //播放爆炸的效果

            enmeyPlaneArray[i].setFree(true);  //  敌机直接消失

            for(int m =0;m<BOMB\_NUM;m++){

                if(bombs[m].isFreeOrnot() ) //空闲状态

                {

                    //空闲的爆炸

                   bombs[m].setFree(false);

                   //更新爆炸坐标

                   bombs[m].setXY(myPlane.getX(),myPlane.getY());

                   myPlane.setLifeNum(myPlane.getLifeNum() - 1);  //战机生命值-1

                   break;

                }

            }

        }

    }

}

3. 将贴图绘制到程序界面中去

void Widget::paintEvent(QPaintEvent \*event)

{

    QPainter painter(this);

    //两张背景图片 绘制

    painter.drawPixmap(0,map.getMap1YPosition(1),map.getMapById(1));  //绘制第一张图

    painter.drawPixmap(0,map.getMap1YPosition(2),map.getMapById(2));  //绘制第二张图

    painter.drawPixmap(myPlane.getX(),myPlane.getY(), myPlane.getPlanePic());  //绘制自己的飞机图

    //绘制所有非空闲的子弹

    for(int i=0;i<BULLET\_CAPACITY;i++){

        //如果非空闲 , 画出来

        if(myPlane.clips[i] .isFreeOrNot() == false){

           painter.drawPixmap(myPlane.clips[i].getX()

                              ,myPlane.clips[i].getY()

                              ,myPlane.clips[i].getBulletPic()) ;

//           painter.setPen(QColor(255,255,255));

//           painter.drawRect(myPlane.clips[i].bulletRect);

        }

    }

    //绘制每个非空闲的敌机r

    for(int i=0;i<ENEMYPLANE\_NUM;i++){

        //如果非空闲 , 画出来

        if(enmeyPlaneArray[i].isFreeOrNot() == false){

           painter.drawPixmap(enmeyPlaneArray[i].getX()

                              ,enmeyPlaneArray[i].getY()

                              ,enmeyPlaneArray[i].getEnemyPlanePic()) ;

//           painter.setPen(QColor(255,255,255));

//           painter.drawRect(enmeyPlaneArray[i].enemyPlaneRect);

        }

        //绘制爆炸的图

        for(int i=0;i<BOMB\_NUM;i++) {

            if(bombs[i].isFreeOrnot() == false)

                painter.drawPixmap(bombs[i].getX(),bombs[i].getY(),bombs[i].pix\_arr[bombs[i].getIndex()]);

        }

    }

    //绘制敌机子弹

    for(int i=0;i<ENEMYBULLET\_AMOUNT ;i++){

        if(enemyBulletArray[i].isFreeOrNot() ==false ){//说明敌机子弹应该被更新

            painter.drawPixmap(enemyBulletArray[i].getX()

                               ,enemyBulletArray[i].getY()

                               ,enemyBulletArray[i].getBulletPic()) ;

//            painter.setPen(QColor(255,255,255));

//             painter.drawRect(enemyBulletArray[i].enemyBulletRect);  //绘制子弹

        }

    }

}

4. 更新所有贴图位置的函数

void Widget::updatePosition()

{

    // 检测是否出现越界的情况 如果出现 立即矫正

    if(myPlane.getX() <= 0 )

        myPlane.setXY(0,myPlane.getY());

    if(myPlane.getY() <= 0 )

        myPlane.setXY(myPlane.getX(),0);

    if(myPlane.getX() >= WINDOW\_WIDTH -PLANE\_WIDTH )

        myPlane.setXY(WINDOW\_WIDTH -PLANE\_WIDTH ,myPlane.getY());

    if(myPlane.getY() >= WINDOW\_HEIGHT -PLANE\_HEIGHT )

        myPlane.setXY(myPlane.getX(), WINDOW\_HEIGHT -PLANE\_HEIGHT);

    map.mapPosition(); //更新地图的坐标

    //敌机出场

    for(int i=0;i<ENEMYPLANE\_NUM;i++){

        if(enmeyPlaneArray[i].isFreeOrNot() == false) {// 说明非空闲

           enmeyPlaneArray[i].updatePosition();

        }

    }

    //敌机子弹出场

    for(int i=0;i<ENEMYBULLET\_AMOUNT;i++){

        if(enemyBulletArray[i].isFreeOrNot() == false) {// 说明非空闲

           enemyBulletArray[i].updatePosition();

        }

    }

    //发射子弹

    myPlane.shoot();

    //计算所有非空闲子弹的当前坐标

    for(int i=0;i<BULLET\_CAPACITY;i++){

        //如果非空闲 , 计算发射位置

        if(myPlane.clips[i] .isFreeOrNot() == false){

            myPlane.clips[i].updatePosition();

        }

    }

    //爆炸效果

    for(int i=0;i<BOMB\_NUM ;i++){

        if(bombs[i].isFreeOrnot() == false){

            bombs[i].updateInfo();

        }

    }

//    planeBullet.setFree( false);

//    planeBullet.updatePosition();

}

1. 贴图类的实现 (以Plane类为例子)

plane.h

class Plane  : public QWidget   //自己的飞机

{

    Q\_OBJECT

private:

    int x,y;  //飞机的位置

    QPixmap planePic;  //飞机的图片

    int record;

    int lifeNum;

public:

    Bullet clips[BULLET\_CAPACITY];

    Plane();

   QRect planeRect;   //飞机看不见的框 用于碰撞检测 和 子弹射中检测

    //调整飞机位置

    void setXY(int xx,int yy);

    //获取飞机当前位置

    int getX();

    int getY();

    int getLifeNum();

    void setLifeNum(int x);

    QPixmap & getPlanePic();

    void up();

    void down();

    void left();

    void right();

    //发射子弹调用函数

    void shoot();

};

plane.cpp

#include "plane.h"

Plane::Plane()

{

    //初始化飞机生命为PLANE\_INITLIVES

    lifeNum = PLANE\_INITLIVES;

    //加载飞机的图片

    planePic.load(PLANEPIC\_PATH);

    QBitmap mask =  planePic.createMaskFromColor(QColor(0,0,0),Qt::MaskInColor);  //设计一个遮罩 把黑色部分弄没

    planePic.setMask(mask);   //图片添加遮罩

    //初始化飞机的位置

    setXY((WINDOW\_WIDTH - PLANE\_WIDTH) \*0.5 , WINDOW\_HEIGHT + PLANE\_HEIGHT );

    //初始化矩形边框

    planeRect.setWidth(PLANE\_WIDTH);

    planeRect.setHeight(PLANE\_HEIGHT);

    planeRect.moveTo(getX(),getY());

    //默认先发射一枚子弹

    this->record = BULLET\_FREQUENCY;

}

void Plane::setXY(int xx, int yy)

{

    x = xx;

    y = yy;

}

int Plane::getX()

{

    return x;

}

int Plane::getY()

{

    return y;

}

int Plane::getLifeNum()

{

    return  lifeNum;

}

void Plane::setLifeNum(int x)

{

    lifeNum = x;

}

QPixmap &Plane::getPlanePic()

{

    return planePic;

}

void Plane::up()

{

    setXY(getX(),getY() - PLANE\_SPEED);

    planeRect.moveTo(getX() ,getY() -PLANE\_SPEED);

}

void Plane::down()

{

    setXY(getX() ,getY() +2 );

    planeRect.moveTo(getX() ,getY() +PLANE\_SPEED );

}

void Plane::left()

{

    setXY(getX() -PLANE\_SPEED,getY() );

    planeRect.moveTo(getX() -PLANE\_SPEED ,getY() );

}

void Plane::right()

{

    setXY(getX() + PLANE\_SPEED,getY() );

    planeRect.moveTo(getX() +PLANE\_SPEED ,getY());

}

void Plane::shoot()

{

    record ++;

    if(record < BULLET\_FREQUENCY)

        return;

    {

        record = 0;

        for(int i=0;i<BULLET\_CAPACITY;i++){

            //如果是空闲子弹 就发射出去

            if(clips[i].isFreeOrNot()){

                //将空闲状态改为假

                clips[i].setFree(false);

                //设置子弹坐标

                clips[i].setXY(getX()  + 0.5\*PLANE\_WIDTH - 0.5\* BULLET\_WIDTH,getY());  //子弹应该偏移一下 因为子弹是根据飞机的位置确定的

                break;  //找到子弹后停止寻找

            }

        }

    }

}

# 5. 实训中遇到的主要问题及解决方法

因为我采用了和老师不一样的qt框架，所以有些地方变简单了，有些地方的难度也随之上升了，qt的代码相较于MFC比较简洁，但是老师给的参考资料我是一点也用不上了，只能自己一点一点摸索着参考网上把程序代码编出来。

遇到的问题例子

1. 怎么使用文本框来显示内容，怎么实现其样式的改变
2. connect函数的使用
3. 按钮是怎么进行定位使用的
4. 如何利用键盘来控制战机的移动
5. 如何绘制图像

解决的方法

1. 利用QLabel类 可以调用 其setStyleSheet() 函数
2. connect是连接函数 可以实现信号的激发和函数调用连接的功能，通过主时钟的脉冲信号进行贴图位置的更新，碰撞的检测等等
3. 按钮直接调用QPushButton 类 通过move() resize() 等函数实现按钮的定位和调整大小
4. 通过重写 ***keyPressEvent和 keyReleaseEvent***函数来控制战机的移动.
5. 通过重写***paintEvent***函数来进行

# 6. 实训体会

这次实训让我学到了很多，通过自学qt来完成一个游戏，一开始的三天我都沉浸在qt的基础的学习中，掌握了qt的基本的使用，我才着手开始自己构建游戏， 因为没有经验，我查阅了很多资料，翻阅了qt的帮助手册，观看了很多qt的实践视频，之前神秘的游戏制作的面纱被我打开，其实理解了原理之后的操作并不是很困难，最重要的思想的学习才是最难的，实训的时间不算短 ，但是我的项目的完成度实际上还不是很高，只是完成了基本的一些功能，例如音效，boss 武器系统等都没有进行设计与思考，代码写的也比较混乱，整个项目的逻辑我没有梳理的很好，如果有时间我会继续完善。

通过此次的学习让我受益匪浅，最终的游戏做出来的时候，我的内心成就感十足。也谢谢老师给我们这次机会，虽然我用的qt但是很多思想也是老师传递给我的，