高级语言程序设计

实验报告

南开大学 计算机大类

王旭

2312166

网安模拟2-2班

2024年5月14日

目录

[高级语言程序设计大作业实验报告 1](#_Toc26979)

一、作业题目 1

二、开发软件 1

三、主要流程 1

1.整体设计思路 1

2.具体各类实现方法 2

3.设计开始界面和结束界面

四、收获

高级语言程序设计大作业实验报告

1. **作业题目**

飞机大战（萌新版）

1. **开发软件**

QT.5.12.8

1. **主要流程**
2. 整体设计思路：

我们要做一个飞机大战类型的小游戏，那么就要写好几个自定义的类对象，我们首先思考，游戏中的元素应该有哪些，首先就是主场景界面、背景场景的布置，其次就是向背景中添加其他对象，自然是有玩家操控的主角（hero飞机）、飞机发射的子弹、敌方飞机，那么其实基本的主要元素就是这些，为了美观起见，我又添加了两个类，用来写开始界面和结束界面，使这个游戏更加完整、美观。

那么总结一下就是如下的类Class：

1. 主界面MainScene
2. 地图背景Map
3. 我方飞机HeroPlane
4. 子弹Bullet
5. 敌方飞机EnemyPlane
6. 开始界面StartWidget
7. 结束界面EndWidget
8. 具体各类实现方法：

有了整体要做的目标各类，我们的目标更加清晰，接下来要做的就是把各类实现一下，那么我们接下来就分别来分析：

1. Map类：

map.h：

#ifndef MAP\_H

#define MAP\_H

#include<QPixmap>

class **Map**

{

public:

//构造函数

**Map**();

// 更新地图滚动后位置的函数

void **mapPosition**();

public:

//地图图片对象

QPixmap map1;

QPixmap map2;

//地图y轴坐标

int map1\_position;

int map2\_position;

//地图滚动速度

int map\_scroll\_speed;

};

#endif // MAP\_H

具体成员函数就是如上.h文件中所示，对于地图而言要有最基本的构造函数、地图的坐标位置、地图位置的更新改变、地图图像Icon、地图的滚动速度，那么对应地，我们在.cpp文件中加以具体实现即可

#include "map.h"

#include "config.h"

Map::**Map**()//在.cpp中实现构造函数

{

//先初始化加载地图图片

map1.load(MAP\_PATH);

map2.load(MAP\_PATH);

//再确定初始地图位置

map1\_position = -Game\_height;

map2\_position = 0;

//再确定地图滚动速度

map\_scroll\_speed = MAP\_SCROLL\_SPEED;

}

void Map::**mapPosition**(){

//更新图片1滚动后的位置

map1\_position += MAP\_SCROLL\_SPEED;

if(map1\_position>=0){

map1\_position = -Game\_height;

}

//更新图片2滚动后的位置

map2\_position +=MAP\_SCROLL\_SPEED;

if(map2\_position>=Game\_height){

map2\_position=0;

} //这里我们的“图片坐标位置”指的是图片上边沿的Y坐标位置

}

实现方法如上图所示，要写构造函数，我们要对于图标首先进行load初始化加载，其次初始化地图的初始位置坐标、初始化地图滚动速度的值。对于成员函数来说，我们要在构造函数之外进行成员函数的编写，注意加上Map：：的域限制！

那么这个Map类我们就自定义封装好了！

接下来的其他类思路都是一致的，我们要首先在类的.h文件中进行类的成员声明以及成员函数的声明，用以确定该类的所有属性和函数作用，其次在确定好类的成员后，我们就要在类的.cpp文件中进行类的所有成员函数的实现，最主要的就是构造函数以及成员函数的实现。在构造函数中我们要写好各个成员的初始化值，而在成员函数中，我们就要写好这个函数的实现方法以及底层逻辑，这样我们的类的封装就完成啦~~~

那么我们再举一个例子吧：比如我们heroplane类的封装：

#ifndef HEROPLANE\_H

#define HEROPLANE\_H

#include <QPixmap>

#include "bullet.h"

#include "config.h"

class **HeroPlane**

{

public:

**HeroPlane**();

//添加发射子弹的函数

void **shoot**();

//添加设置飞机位置的函数

void **setPosition**(int x,int y);

public:

//飞机图标

QPixmap plane\_icon;

//飞机坐标

int plane\_X;

int plane\_Y;

//飞机矩形边框

QRect plane\_rect;

//飞机状态

bool plane\_free;

//弹匣

Bullet bullets[BULLET\_NUM];

//发射间隔记录

int fire\_recorder;

};

#endif // HEROPLANE\_H

首先写好我们heroplane.h的成员确定：

对于heroplane，我们希望他的属性有：

(1)图标(2)飞机的坐标位置(3)飞机的矩形边框（用于判断碰撞检测）(4)飞机的状态(true代表空闲，即坠毁；false代表不空闲，即正常运行) (5)子弹弹匣以及发射间隔

成员函数包括：(1)构造函数（毋庸置疑）(2)更新坐标位置的函数(3)发射子弹的函数

那么我们接下来在.cpp文件中实现：

#include "heroplane.h"

#include "config.h"

#include<QIcon>

HeroPlane::**HeroPlane**()

{

//初始化设置飞机的图标

plane\_icon.load(PLANE\_ICON);

//初始化飞机的坐标位置(这里指的是图标左上角的坐标位置)

plane\_X = Game\_width\*0.5 - plane\_icon.width()\*0.5;

plane\_Y = Game\_height - plane\_icon.height();

//初始化状态

plane\_free = false;

//初始化飞机的矩形边框

plane\_rect.setWidth(plane\_icon.width());

plane\_rect.setHeight(plane\_icon.height());

plane\_rect.moveTo(plane\_X,plane\_Y);

//初始化子弹发射间隔

fire\_recorder = 0;

}

void HeroPlane::**shoot**(){

//累加时间间隔记录变量

fire\_recorder++;

//判断是否达到发射时间

if(fire\_recorder<=BULLET\_INTERVAL){

return;

}

//若到达发射时间，则重置发射时间fire\_recorder

fire\_recorder = 0;

//发射子弹

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

//如果是空闲的子弹进行发射

if(bullets[i].bullet\_free==true){

//发射后将状态改为false

bullets[i].bullet\_free = false;

//设置发射时的子弹坐标位置

bullets[i].bullet\_x = plane\_X + plane\_rect.width()\*0.5 - 5;

bullets[i].bullet\_y = plane\_Y - 10;//注意这里之所以减去10和5，是因为原来的位置坐标处在飞机hero.icon的内部，并不位于飞机的头部发射位置，所以要调整

break;

}

}

}

void HeroPlane::**setPosition**(int x,int y){

plane\_X = x;

plane\_Y = y;

plane\_rect.moveTo(plane\_X,plane\_Y);

}

相对应的进行每一个成员函数的实现即可：

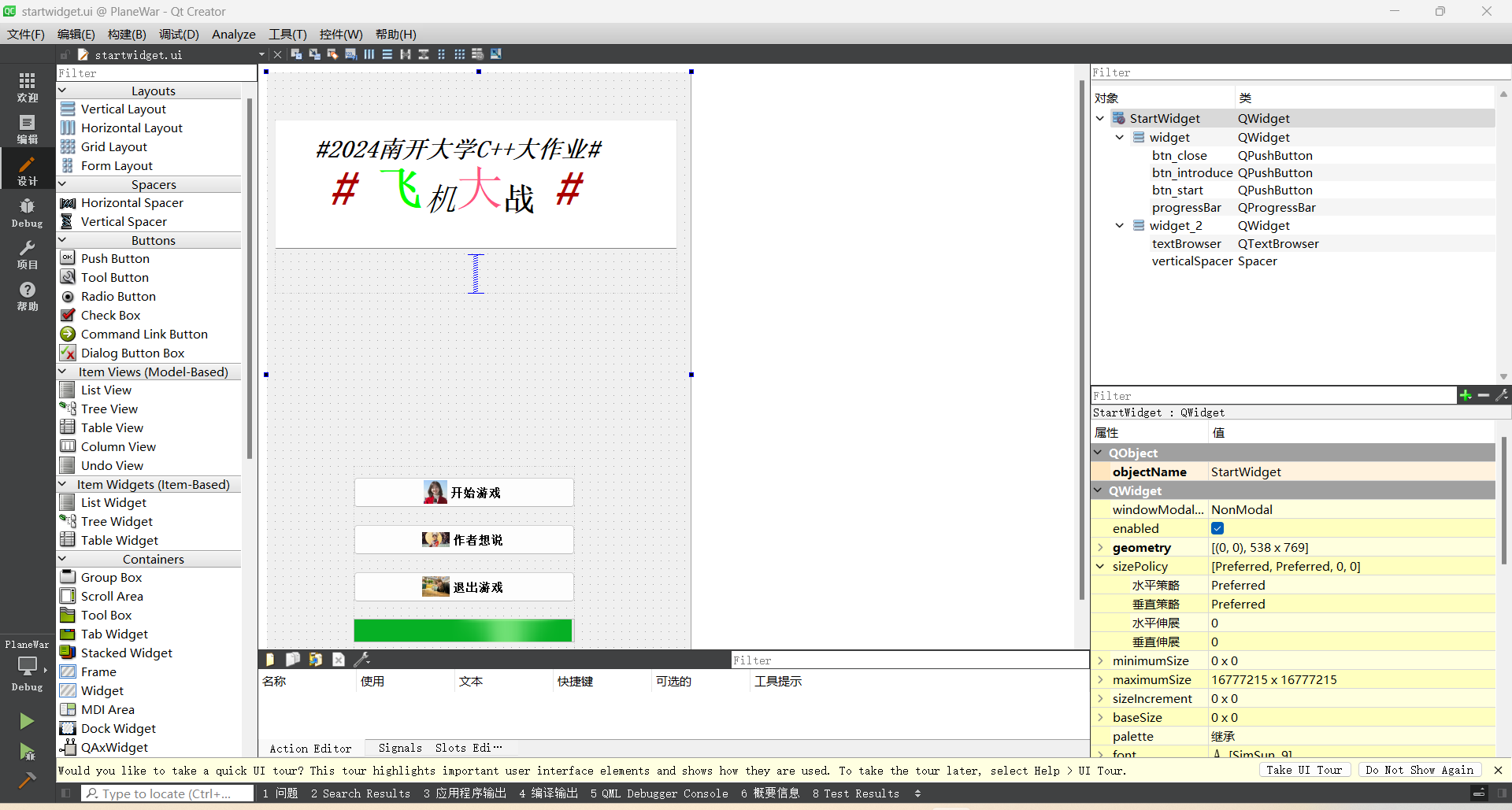
比如构造函数中初始化和各个成员变量

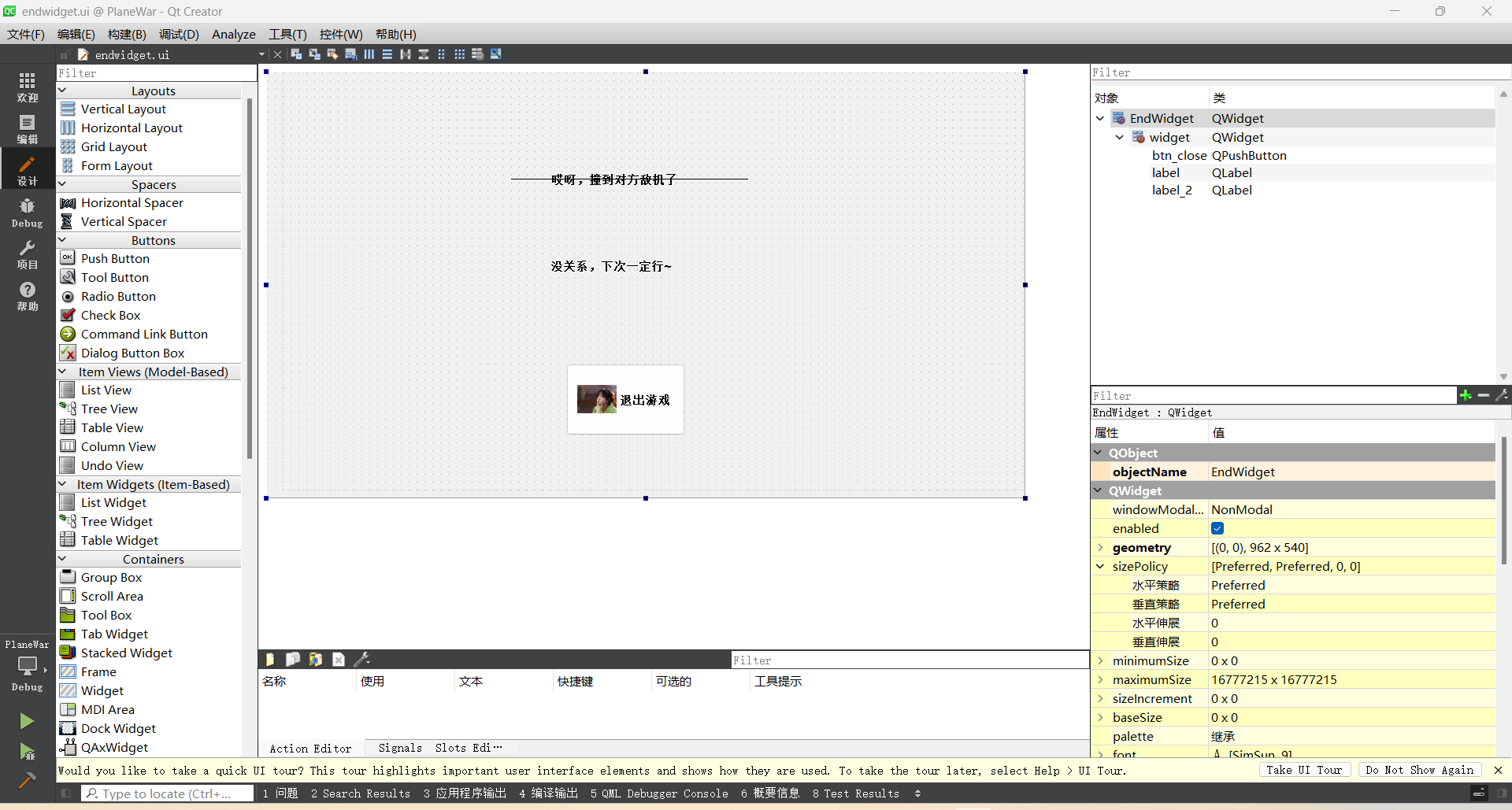
在shoot()函数中写好bullet的发射位置，以及后续的坐标位置变化，同时确定好bullet的状态等等（发射时为false即子弹不空闲，是正常可用状态）

这样一来，我们HeroPlane类的封装就完成啦！

3.设计开始界面和结束界面

特别地，对于开始界面和结束界面来说，由于是直接显示出窗口Widget，所以我们就可以用更简便的方法，就是利用UI界面进行对两个界面的布局，从而避免了书写代码不断调试的过程，更加方便直观~





如上图为UI设计界面

分别为StartWidget开始界面和EndWidget结束界面的UI设计界面，更加直观简洁。

1. 收获：

通过QT的学习以及运用过程，我中途遇到了许多大大小小的障碍和困难，尤其是bullet类中Qrect的位置改变函数这一部分，刚开始忘记了对子弹矩形的坐标位置进行同步改变而只改变了子弹的绘图位置，从而掩盖了问题，使子弹与敌机的碰撞检测始终不能成功，后来经过通过使用qDebug来进行一步步调试，最终发现了问题所在，成就感满满。

同时我也感觉到，学无止境，以及学以致用的重要性，重要的不是“听懂”，而是“会用”。同时也要不断突破自己，勇于接受新事物新挑战，遇到困难和障碍不要气馁，要通过使用巧妙的方法不断地进行调试和尝试，最终一定会“柳暗花明又一村”！