

2018级软件学院小学期课程报告

|  |  |
| --- | --- |
| 二O 一九 年 | 七月18日 |

**2D游戏设计报告**

|  |  |
| --- | --- |
| 姓名： | 谢沛辰 |
| 学号： | 24320182203301 |
| 学院： | 软件学院 |
| 专业： | 软件工程系 |
| 指导老师： | 苏淑文 |

**摘要**

关键词：电子游戏；2D游戏设计；C++程序设计；EasyX库

电子游戏，是指所有依托于电子设备平台而运行的交互游戏。根据媒介的不同多分为五种：主机游戏（或称家用机游戏、电视游戏）、掌机游戏、电脑游戏、街机游戏和移动游戏（主要是手机游戏）。由于电子游戏具有互动性、模拟性，所以深受大量用户群体的欢迎，各大电子信息公司（如腾讯）也将游戏作为流量的主要变现通道。故在现代的商业竞争中，游戏设计也就成为了一片红海，在程序设计的市场中占有一席之地。

本文将以EasyX库为第三方平台，以笔者所设计的Defending Earth为例，深入探究2D游戏设计的原理、流程及其注意事项。

目录

[第三方类库：EasyX 4](#_Toc14090990)

[1.1 EasyX简介 4](#_Toc14090991)

[1.2选择EasyX的原因 5](#_Toc14090992)

[1.3 EasyX学习路径图 5](#_Toc14090993)

[游戏策划 6](#_Toc14090994)

[游戏功能清单 7](#_Toc14090995)

[游戏构架设计 8](#_Toc14090996)

[3.1如何处理用户输入 9](#_Toc14090997)

[3.2如何保存游戏的各种状态 11](#_Toc14090998)

[3.3如何刷屏显示 11](#_Toc14090999)

[3.4如何实现游戏循环 11](#_Toc14091000)

[核心类的设计与分析 12](#_Toc14091001)

# 第三方类库：EasyX

## EasyX简介

EasyX 是针对 C++ 的图形库，可以帮助 C 语言初学者快速上手图形和游戏编程。

比如，可以用 VC + EasyX 很快的用几何图形画一个房子，或者一辆移动的小车，可以编写俄罗斯方块、贪吃蛇、黑白棋等小游戏，可以练习图形学的各种算法，等等等等。

EasyX只是一个图形库，但这个图形库功能却十分的广泛。有绘图环境设置函数、颜色模型、图形颜色与样式设置、图形绘制函数、文字输出函数、图像处理相关函数、鼠标相关函数等等数个大类。可以说就凭EasyX的这几大类函数，就足以去编写一个小游戏程序。

EasyX另一大特点就是绝大部分操作都依赖于颜色、坐标与指针。在图形页面初始化之后，无论是读取图片还是文字输出亦或是绘制图形（如按钮）都需要这三者中的两者，这样使得以上介绍的各种函数的具体实现成为了可能。

EasyX在最近又增添了不少的新函数，譬如绘制任意多边形、设置二元光栅等，使得EasyX的功能更为强大。

## 1.2选择EasyX的原因

相比于其他第三方类库，EasyX算不上很两眼，但笔者依然选择了使用EasyX，主要出于以下三个原因：

第一，相比于其他的第三方类库如unity，EasyX虽然没有那么复杂的功能如渲染，physics 2D，但是正是由于EasyX相对简洁，导致其更容易上手，原理清晰明了，其他的第三方类库由于笔者不能熟悉并掌握这些功能，导致很多笔者想要的功能无法较为理想地呈现出来。

第二，由于笔者设计的游戏并不需要极其复杂的功能，仅仅EasyX的功能就足以满足游戏的需要。

第三，由于EasyX的功能远不如其他第三方类库强大，所以要想在EasyX上做出像unity上做出来的2D游戏，原本可以由平台带未完成的很多底层工作都需要自己完成，比如物体的移动等等，而在这个过程中是可以锻炼构想程序设计思路的能力。而且使用EasyX来编写小游戏对于设计者的算法要求较高，也正是笔者期望在小学期学习中得到锻炼提升的能力。

## 1.3 EasyX学习路径图

笔者先是在官网下载EasyX时看到了附带的EasyX help（相当于是Easy X的说明书），通过这个help大致了解了EasyX的函数用法后，前往EasyX官网观摩了一些案例的设计思路，并在哔哩哔哩网站上看了相关的教程。

https://easyx.cn/readme

https://www.bilibili.com/video/av27519937?from=search&seid=1204661585416624325

# 游戏策划

游戏概况：

Defending Earth是一款第三人称STG游戏，是以电影《复仇者联盟》2中绯红女巫让钢铁侠看到的一段幻觉为原型进行改编再创作。

背景故事简介：

《复仇者联盟》中入侵地球的齐塔瑞大军再次挥师卷土重来，而复仇者们（以下简称为复联）决定要将战线从大气层内拉到大气层外，于是昆式战斗机从基地起飞，飞出大气层，正面阻击齐塔瑞人为后续大部队争取时间。

角色设计：

阵营分为复联与齐塔瑞人两个阵营，复联方为一架昆式战斗机，可发射能量炮；齐塔瑞人方有两种角色，一个是齐塔瑞鲸鱼飞船，只能近距对敌进行撕咬与接触破坏，另一个是圣殿号巨型飞船，可以发射炮火进行打击。玩家将操纵复联方的昆式战斗机，尽可能多的击落齐塔瑞人的飞船。

游戏规则与操作方式：

键盘上的w、a、s、d键分别代表前进，向左移动、后退、向右移动，k为发射能量炮，对方圣殿号飞船的炮火击中昆式战斗机或者鲸鱼飞船接触到昆式时己方战机会受损，当损伤程度严重到己方血条为0时宣告失败，支援抵达时即为成功。

# 游戏功能清单

表1-1 游戏功能清单

|  |  |
| --- | --- |
|  | 炮弹移动/擦除功能 |
|  | 判断玩家输赢功能 |
|  | 双方战机移动/擦除功能 |
|  | 中弹判断功能 |
|  | 血量扣除功能 |
|  | 页面切换功能 |
|  | 键盘输入处理功能 |
|  | 飞船/飞机中弹后血量扣除与爆炸功能 |
|  | 双方飞船/飞机暨炮弹坐标存储功能 |

# 游戏构架设计

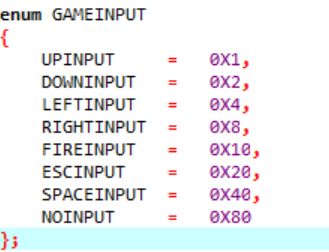
图1-1 游戏内层构架

图1-2 游戏外层架构

## 3.1如何处理用户输入

本游戏使用int枚举类型来定义以下几种操作如图1-3所示：

图1-3 被承认的用户操作标记



本游戏共有两个函数一起处理用户输入，先是使用GetAsyncKeyState函数获取并判断按键是否要进行距离然后再分门别类进行处理，代码如下。

void GetOrder()

{ ClkMakeBulletFinish = clock(); // 己方战机子弹产生时间差

if (GetAsyncKeyState('W') & 0x8000)

{

myinput |= UPINPUT;//标记为向前

}

if (GetAsyncKeyState('S') & 0x8000)

{

myinput |= DOWNINPUT;//标记为向后

}

if (GetAsyncKeyState('A') & 0x8000)

{

myinput |= LEFTINPUT;//标记为向左

}

if (GetAsyncKeyState('D') & 0x8000)

{

myinput |= RIGHTINPUT;//标记为向右

}

if (GetAsyncKeyState('K') & 0x8000 && (ClkMakeBulletFinish - ClkMakeBulletStart > 200))

{

// 己方战机炮弹

myinput |= FIREINPUT;

ClkMakeBulletStart = ClkMakeBulletFinish;//标记为开火

}

if (GetAsyncKeyState(' ') & 0x8000)

{

myinput |= SPACEINPUT;//标记为空格

}

if (GetAsyncKeyState(VK\_ESCAPE) & 0x8000)

{

myinput |= ESCINPUT;//标记为退出

}

}

void DealInPut()

{

if ((myinput & UPINPUT) && (myship.m\_nCol >= 10))

{

myship.m\_nCol -= 2;//坐标纵坐标减小

}

if ((myinput & DOWNINPUT) && (myship.m\_nCol <= GAMEHIGHT - ship.getheight() - 10))

{

myship.m\_nCol += 2;//坐标纵坐标增加

}

if ((myinput & LEFTINPUT) && (myship.m\_nRow >= 10))

{

myship.m\_nRow -= 2;//坐标横坐标减小

}

if ((myinput & RIGHTINPUT) && (myship.m\_nRow <= GAMEWIDTH - ship.getwidth() - 10))

{

myship.m\_nRow += 2;//坐标横坐标增加

}

if (myinput & FIREINPUT)

{

CPPBullet\* pNewBullet = new CPPBullet(myship.m\_nRow + ((ship.getwidth() / 2)), myship.m\_nCol - 5);

VectorBullets.push\_back(pNewBullet);//创建炮弹

}

if (myinput & SPACEINPUT)

{

\_getch();//暂停

}

if (myinput & ESCINPUT)

{

exit(0);//退出

}

myinput = NOINPUT;//复原

}

## 3.2如何保存游戏的各种状态

本游戏中需保存的游戏各种状态主要是各种坐标数据（int整数类型）以及枚举类型所标识的状态，譬如图1-3所展示的用户输入操作标识以及各种角色的坐标。复联方昆式战机的坐标数据除了初始化外都是由玩家的输入进行修改的。而齐塔瑞方的状态数据则是在随机初始化后按程序自动进行修改保存的（直至被消灭）。而双方发射的炮弹统一都由vector进行保存并对由用户输入的操作和程序起义对其中的坐标进行修改和存储。

## 3.3如何刷屏显示

本游戏中的刷屏主要是用EasyX自带的cleardevice函数和putimage函数实现的，在图1-1显示的内层架构中，每次循环到绘制双方阵营的飞船前，都会将双方之前的绘制图像进行擦除。笔者在这里只展示复联方昆式战机的刷屏。

void CPPQuinJet::ClearUp() // 掩盖己方战机

{

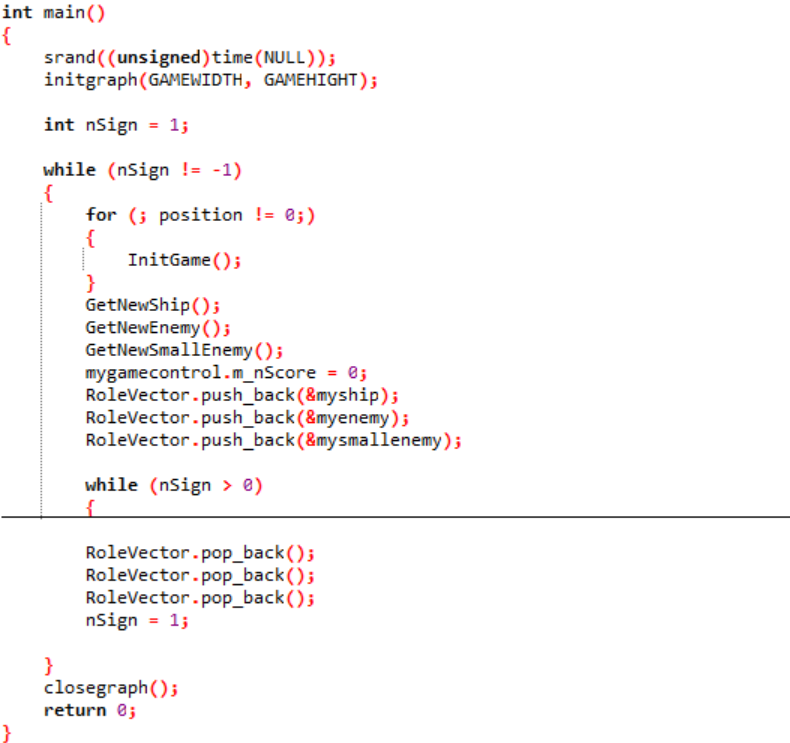
putimage(m\_nRow, m\_nCol, &QuinJet1, SRCAND);

}

本游戏由于是使用EasyX这个图形库而不是专门的游戏平台，所以像物体或角色移动这样可以在其他平台上较为简易的操作的底层功能在本游戏也同样要加以编辑。所以在本游戏中笔者设计了动画功能，以实现物体角色的运动。这个动画功能先是用方形黑色矩阵掩盖了原来的角色图像，然后根据变化过的坐标在新坐标绘制图像。在角色牺牲前，这个循环是不会停的。

## 3.4如何实现游戏循环

图1-4 main.cpp



在图1-1和图1-2中本文已经分别展示了内层架构与外层架构，这里借由图1-4来做进一步的说明。

我们用一个int变量nSign来进行标记。nSign设定的初始值为1，打开程序就可以直接进入初始页面。内层架构的循环中会有处理结尾的函数（这个函数的触发条件就是玩家胜利或失败）修改nSign的值，当玩家想再来一局时，nSign就会被修改为-2并使程序回到初始界面，如果玩家想要退出，nSign就会被修改为-1，游戏直接结束。

# 核心类的设计与分析

本游戏的核心类是三个角色类CPPShip、CPPEnemy、CPPSmallEnemy以及他们的共同基类：CGameControl。所有的继承方式和成员访问限制均为public。

图1-5 CGameRole类

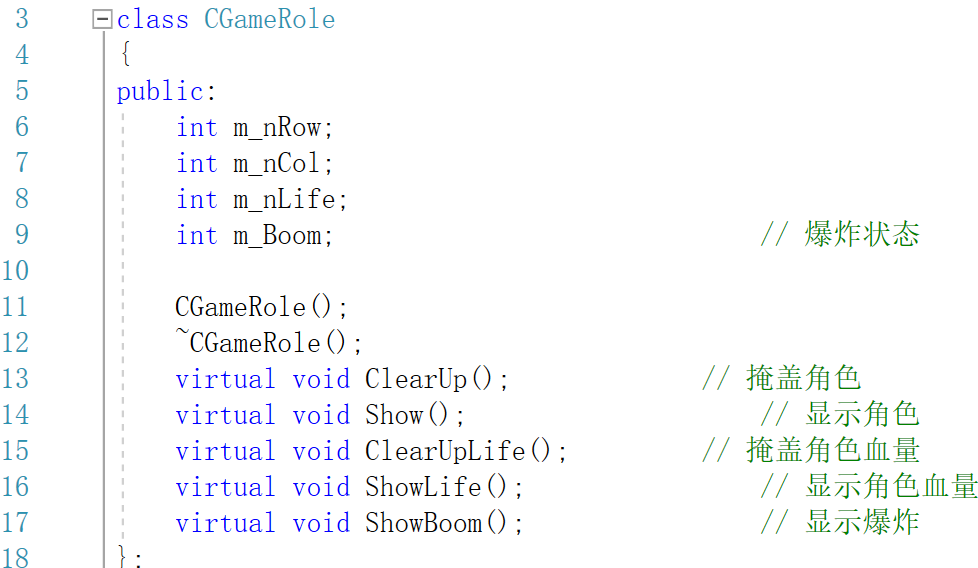


图1-6 CPPEnemy类

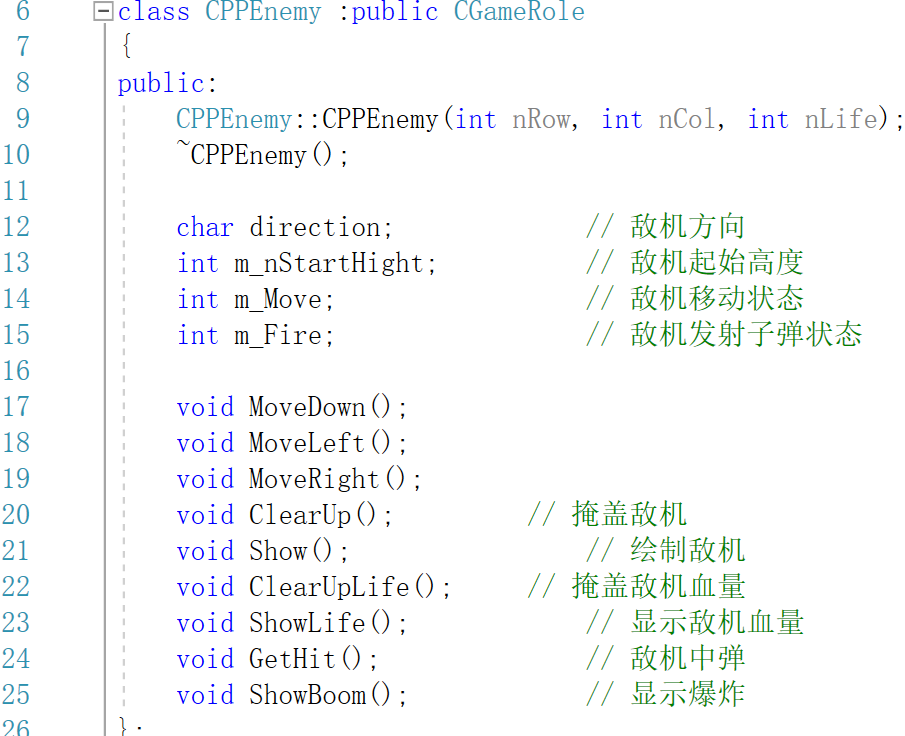


图1-7 CPPShip类

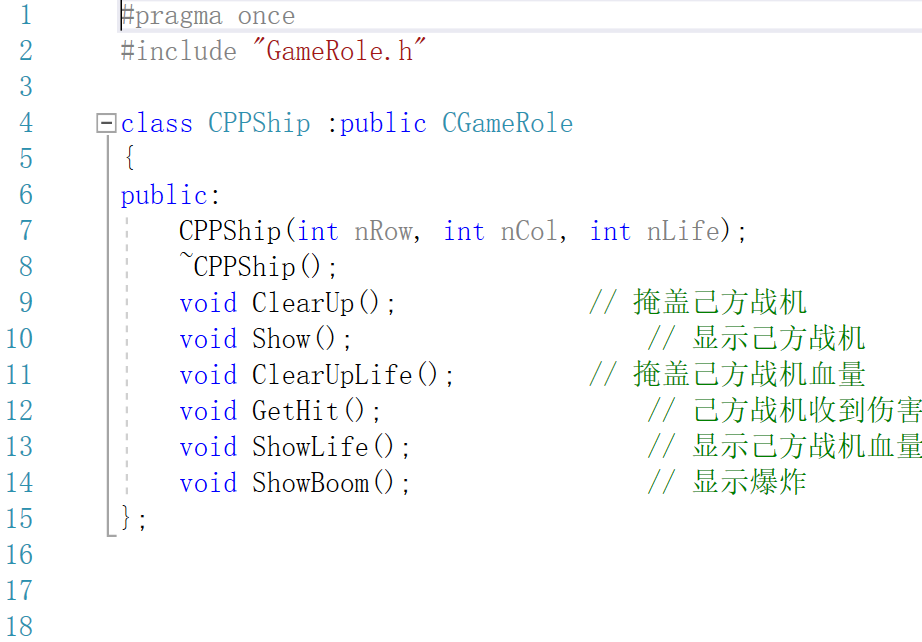
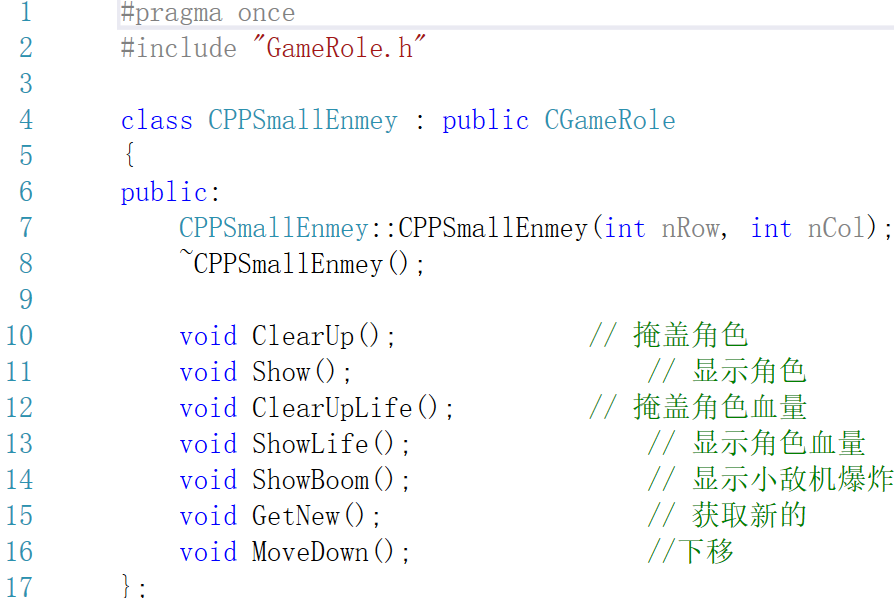


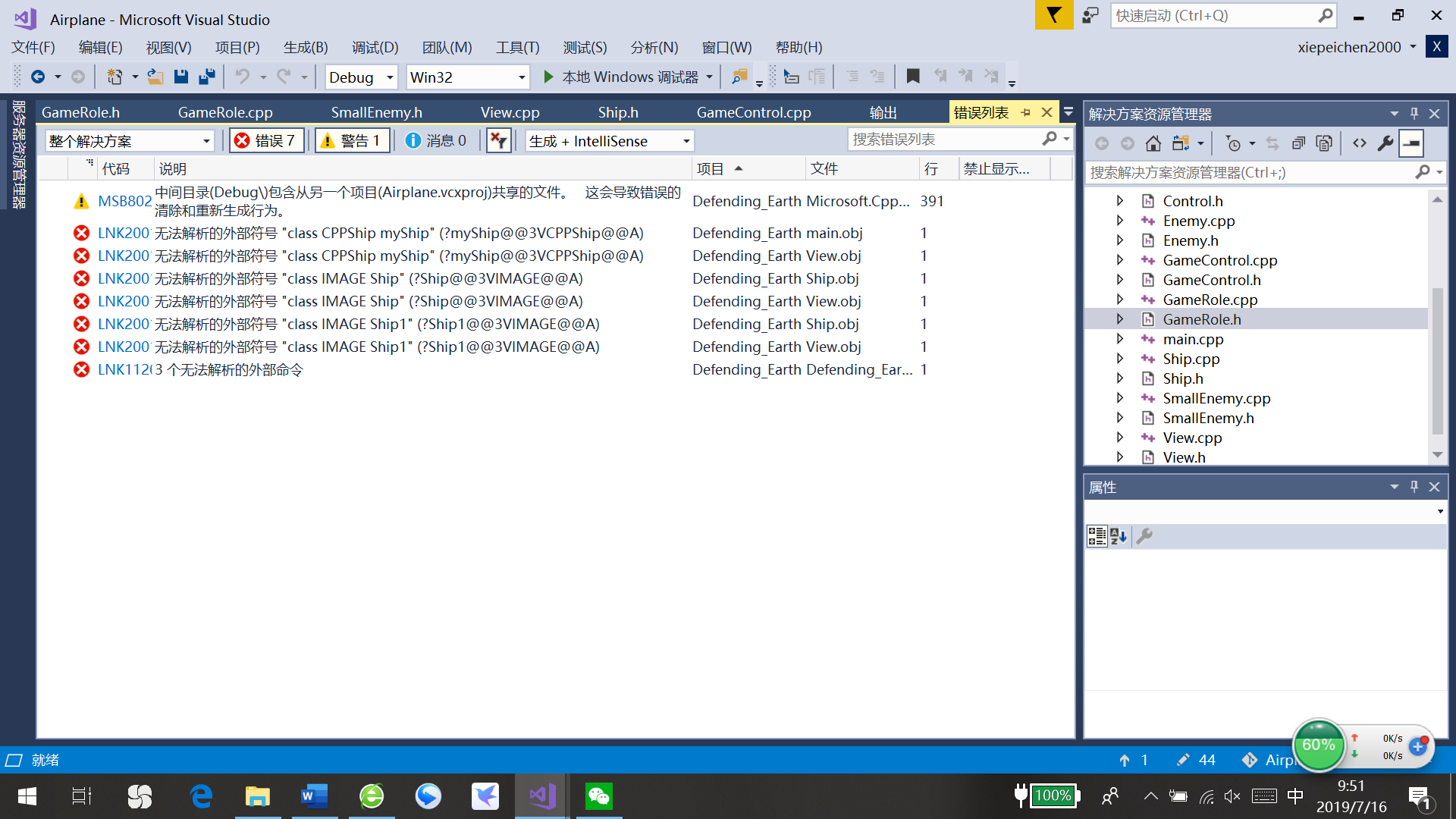
图1-8 CPPSmallEnemy类



可以看出，在继承的过程中，子类（图1-6、图1-7、图1-8）对于父类（图1-5）的虚函数进行了一系列的重载，这样使得程序的可扩展性大大增强。

# 调试过程

图1-9 调试过程截图



# 结果展示



