# MFC期末大作业：

## 一、类介绍

### 1.1海图相关类：

海图点类：CChartPoint

海图线类：CChartLine

海图面类：CChartFace

海图类：CChart: 嵌套CChartPoint，CChartLine，CChartFace，存储海图点线面及其他基本信息

增加电类：CCreatePoint，包含点的坐标，属性及画点操作

增加线类：CCreateLine，包含线的坐标，属性及画线操作

增加面类：CCreateFace，包含面的坐标，属性及画面操作

链表头类：Hed，方便链表的增删改查等操作

全局数据类：Global\_items，存储了线颜色，线型，线宽，面颜色，灯浮文字说明等信息，可以通过对话框改变全局变量来改变点线面属性

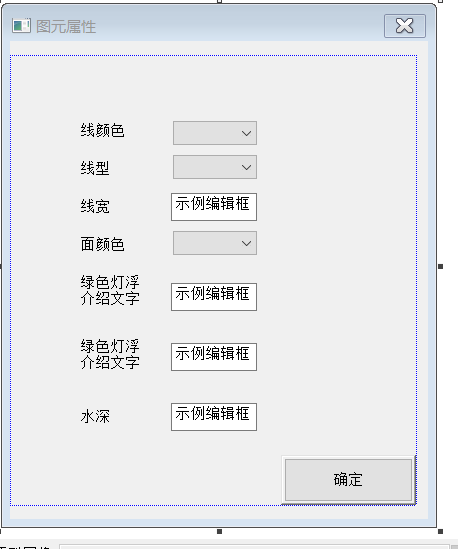


图1 全局数据对话框

函数：

Main()及操作函数，可以读取数据，操作数据等

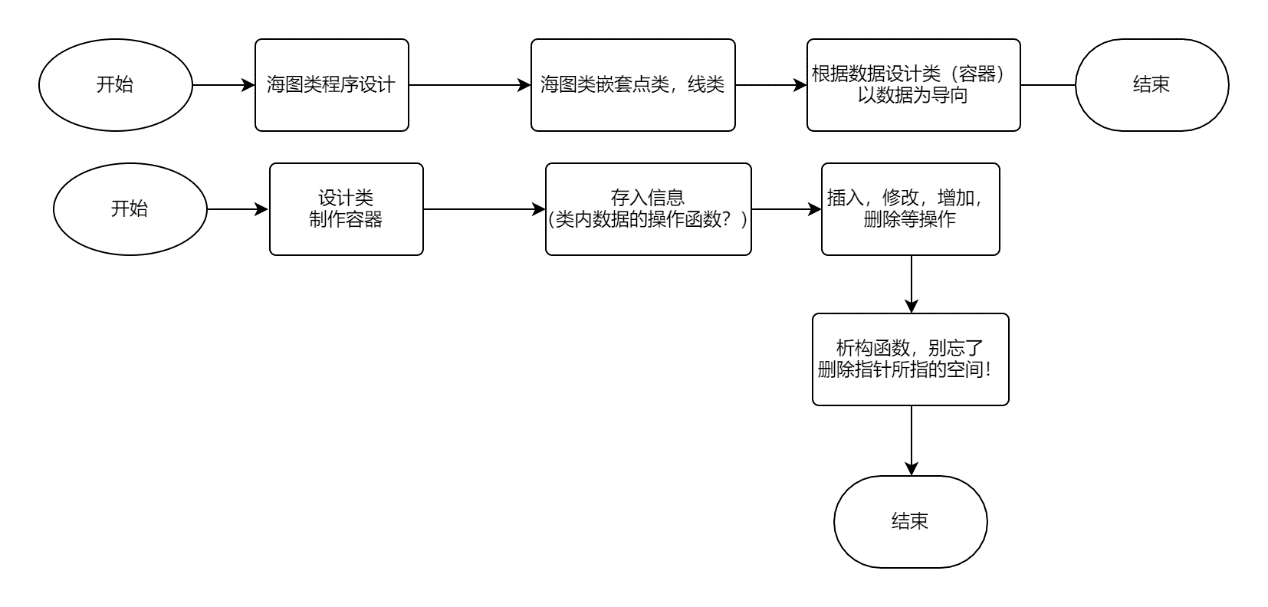
图

图1 海图类程序设计



图2 绘制的海图

### 1.2海战游戏相关类：

游戏基类：CGameObject，继承自CObject，包含了所有游戏类的基本属性，比如坐标；虽然每个类的Load图片操作相似，但由于路径不同，难以找到基本函数，即使写了到时候因为路径也得重载，所以并没有函数

本船类：MyShip，自己的战船

大敌船类：BigEnemy

普通敌船类：Enemy

本船子弹类：Bullet

View类:将各个类实例化，并通过OLButtonDown()等接口对各个类进行数据操作等

如：void CMy1120210020View::OnLButtonDown(UINT nFlags, CPoint point)

{

// TODO: 在此添加消息处理程序代码和/或调用默认值

CDC\* pDC0 = GetDC();

//如果画点类被创建了，那么调动他的OnLButtonDown(nFlags, point)函数画点，

//其他两个类似

if (CMy1120210020View::tempDrawPoint)

{

//需要PDC0画线

tempDrawPoint->pDC = pDC0;

tempDrawPoint->OnLButtonDown(nFlags, point);

}

if (CMy1120210020View::tempDrawLine)

{

tempDrawLine->Line->pDC = pDC0;

tempDrawLine->OnLButtonDown(nFlags, point);

}

if (CMy1120210020View::tempDrawFace)

{

tempDrawFace->pDC = pDC0;

tempDrawFace->OnLButtonDown(nFlags, point);

}

ReleaseDC(pDC0);

CView::OnLButtonDown(nFlags, point);

}

## 二、设计思路：

### 2.1海图画线思路

1. 添加画线类，类内设计线坐标及其属性，画线函数响应View类的OLButtonDown()接口以画线

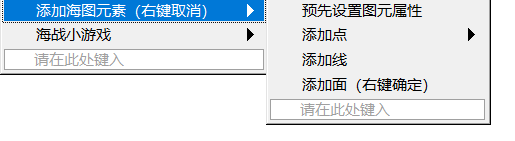


图3 添加元素菜单

1. View类内->声明类指针->构造内初始化为NULL;
2. 右键菜单栏添加线响应函数：申请画线类的空间，申请空间后，画线类指针非NULL
3. 左键画图

if (CMy1120210020View::tempDrawLine)

{

tempDrawLine->Line->pDC = pDC0;

tempDrawLine->OnLButtonDown(nFlags, point);

}

1. 右键销毁画线类空间

if (tempDrawLine)

{

delete tempDrawLine;

tempDrawLine = NULL;

}

其他的添加元素类思路类似

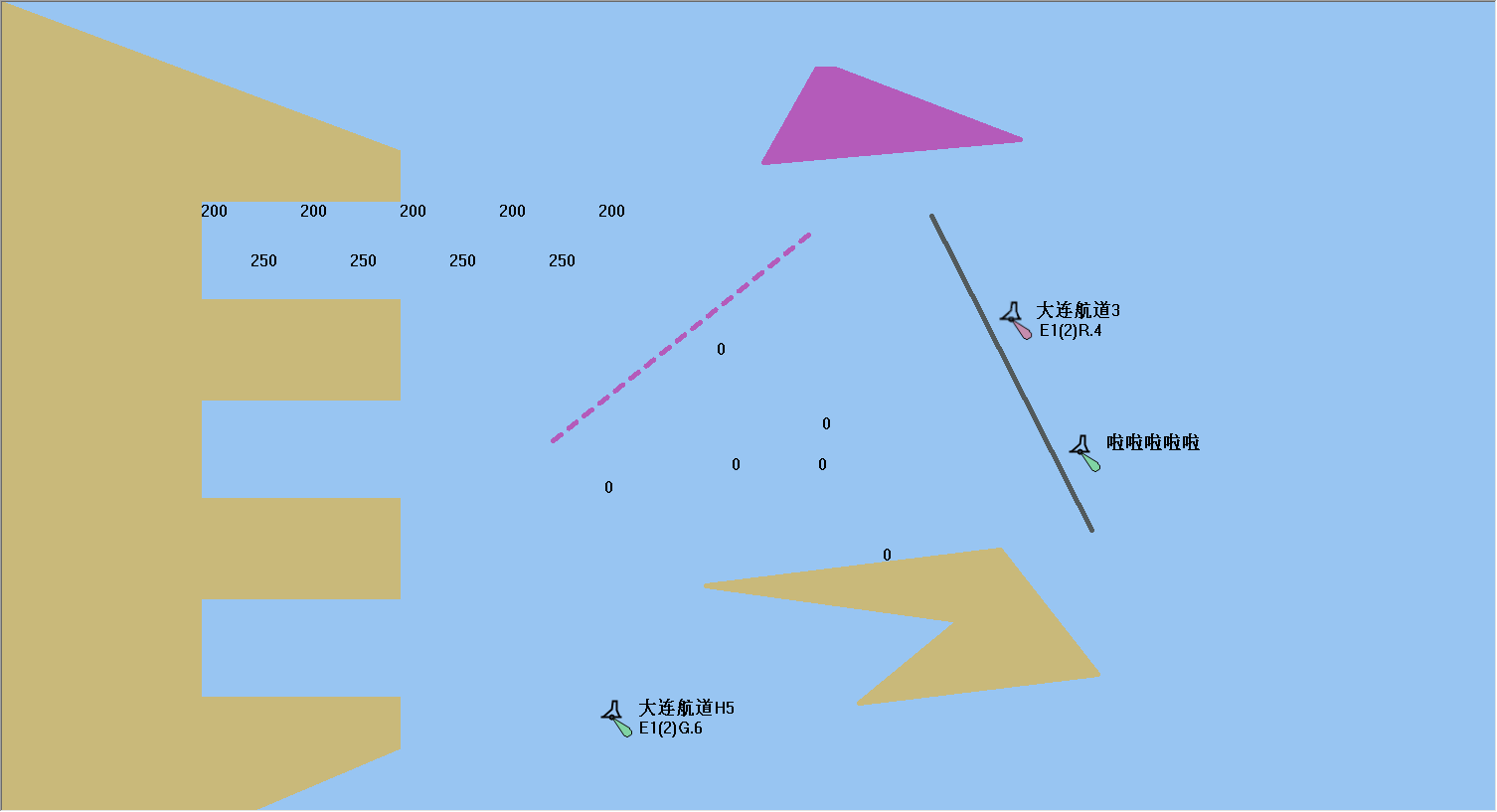


图3 添加点线面示意图

### 2.2海战游戏思路

游戏 == 绘图；

**2.2.1** 第一关敌军不能发射子弹，50分通关，小船1分，大船3分；

第二关敌军发射直子弹，50分通关，小船1分，大船3分；

第三关敌军发射追踪弹，击杀大船即可通关。

**2.2.2** 敌船和子弹在出了屏幕或爆炸后会重新生成，子弹回到敌船子弹口

**2.2.3**因为所有的游戏类都得包括坐标，为了减少重复，设计了CGameObject基类

游戏类的核心就是坐标，在各个游戏类的内部设立坐标并在相应的坐标上贴图，以完成游戏绘制

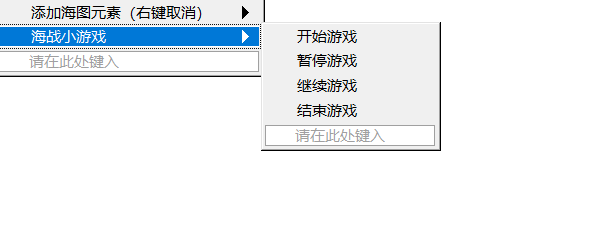


图4 海战小游戏菜单

**2.2.4** 游戏刷新利用的是SetTimer()定时器，OnTimer()函数内重绘屏幕以达到动画的效果，值得注意的是：游戏的绘制采用的是双缓冲机制，即必须一次性将所有画面绘制到缓存DC上，然后拷贝到屏幕DC上，以避免闪烁。但我设计的游戏背景不是白色，还是会出现闪烁，这是因为屏幕在刷新时会先将屏幕变成白板，这个时候需要重写View类的OnEraseBkgnd(CDC\* pDC)函数，不让屏幕在刷新时变成白板

BOOL CMy1120210020View::OnEraseBkgnd(CDC\* pDC)

{

// TODO: 在此添加消息处理程序代码和/或调用默认值CView::OnEraseBkgnd(pDC)

return TRUE;

}

**2.2.5**设计void CMy1120210020View::OnKeyDown()函数以改变本船坐标以达到改变本船位置的动画效果，本程序采用的坐标直接加减的方法

void CMy1120210020View::OnKeyDown(UINT nChar, UINT nRepCnt, UINT nFlags)

{

// TODO: 在此添加消息处理程序代码和/或调用默认值

if (nChar == 37)//向左移

{

if (Myship.x > 0)

Myship.x -= 10;

}

else

{

if (nChar == 39)//向右移

if (Myship.x < 450)

Myship.x += 10;

}

CView::OnKeyDown(nChar, nRepCnt, nFlags);

}

另一种改变坐标的方法是用Crect类，示意图如下：

CRect

CRect

**OffSet(offx,offy)**

图5 Crect类坐标转换示意图

**2.2.6**

在寻找海图素材的时候找到一个好网站，可以将图片转换成像素风格，并去除背景！

//像素素材网站

[PixelMe : Convert your photo into pixelart. (pixel-me.tokyo)](https://pixel-me.tokyo/en/)

**2.2.7** 实现了游戏背景音乐播放：

mciSendString((L"open res\\BackMusic.mp3 Alias BackMusic"), NULL, 0, NULL)

mciSendString(\_T("play BackMusic repeat"), NULL, 0, NULL);



图6 海战游戏示意图

**2.2.8** 更改了程序图标

## 三、心得

**3.1**类的继承，是MFC的特点，如果可以合理设计基类，可以减少代码量，实现代码复用。但是在实践过程中，我感觉基类是很难设计的；类本身就是一些事物的抽象集合体，在此基础上进一步抽象，找到其共同点，不容易。

我认为基类不用一开始设计好，可以随着功能的逐渐丰富，代码重复使用程度，合理简化功能，提取共性，建立基类。类的设计是一门哲学。

**3.2**不管脑子里有多少想法，千里之行，始于足下。万事开头难，我们需要做的就是先把最简单的功能实现了，验证想法。先去做，随着实践的需求增加，自然就丰富了代码，实现了丰富的功能。一个产品不是一开始就设计好的，然是随着用户反馈，时间积累，逐渐臻至完美的！

**3.3** Debug:链接器错误——》可能是未知的奇怪的命令：

支持串行化：DECLARE\_SERIAL(CGameObject)，

如果没有.cpp文件内接上IMPLEMENT\_SERIAL(CGameObject, CObject, 0)，会产生链接器错误

**3.4** 游戏设计最难的就是船及子弹的坐标设计及计算，需要数学功底，挨个计算，且耗费时间

**3.5**由于时间关系，文档写的匆忙，可能有些语句不通顺，错别字等。程序的一部分功能并没有实现，程序还有完善的空间，比如船在陆地上不能航行等功能亟待完善。