函数画板(FunctionSketch)设计文档

一、程序简述

本软件是一个轻量级的图像绘制软件。在实现较为丰富的函数绘制功能的同时，避免了庞大的体量。与Matlab、Mathematica等专业软件的二维图像绘制功能相比，更方便上手，便于使用。可用于满足学习和简单研究的二维绘图需要。

二、实现思路

1、表达式字符串解析

（1）解析目标

本程序的主要输入内容为表达式字符串，用以表示单变量函数、参数方程或是隐函数。为了进行计算，我们需要将其转化为可使用计算机计算的函数。我们可以将三种方程分别看成单参数、单返回值的函数；单参数、双返回值的函数；双参数，单返回值的函数（即`f(x,y)=0`）。这可以利用`C#`的`delegate`实现。`C#`中预先定义了`Func<>`委托。利用泛型可分别存储三种类型的方法。

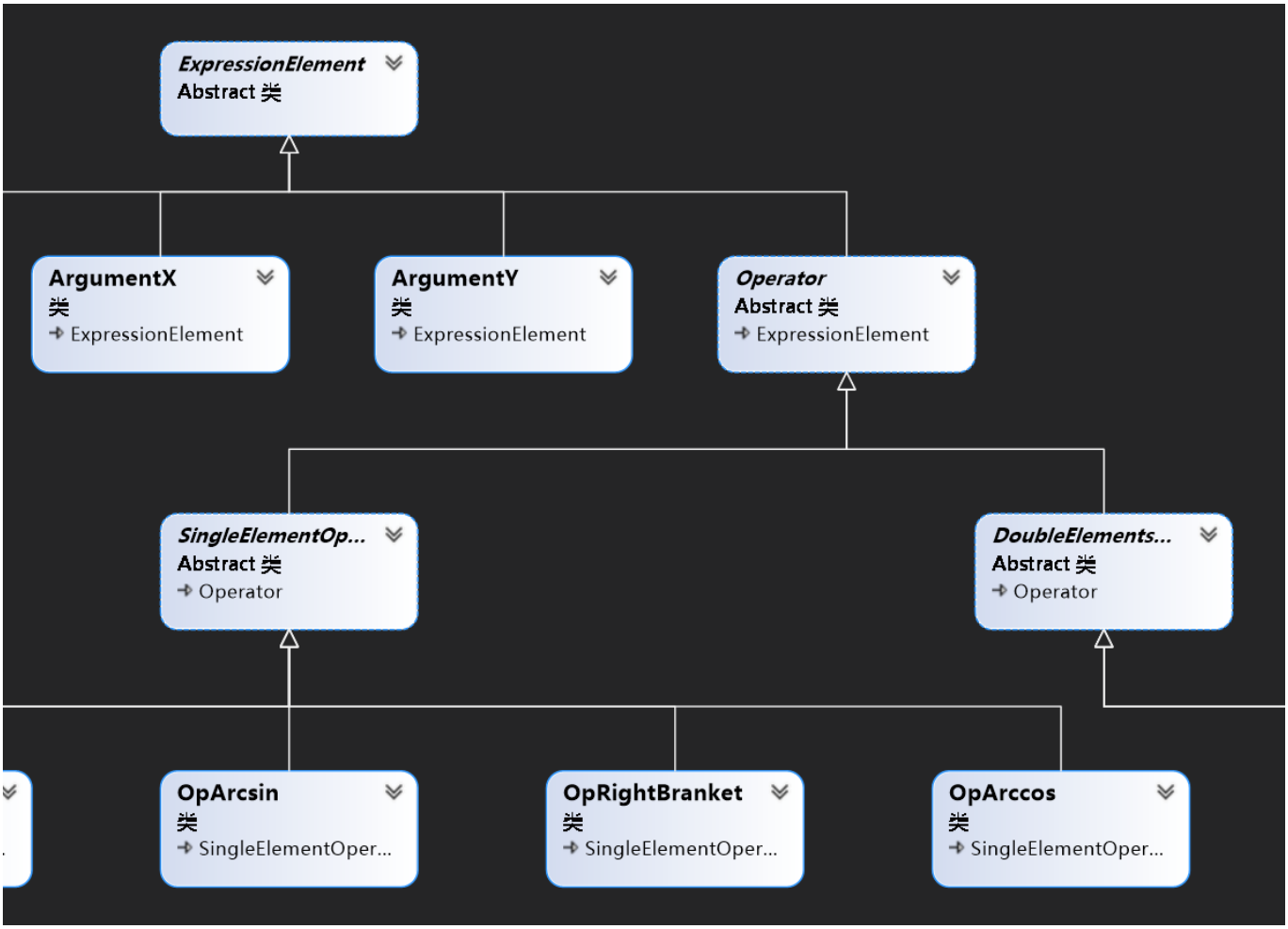
因此现在的问题便是，如何依据不同的字符串，构造出对应的可反映该表达式计算过程的方法。这里采取表达式树的结构。

（2）FunctionParser类——字符串转表达式树

字符串转表达式树的基本算法已经在大一的数据结构课上讲过了。这里不赘述。

不同的地方在于面向对象可以方便的利用多态代替判断。因此对于表达式树，不再需要节点内储存数据表示不同的节点类型。只需要所有的节点继承共同的抽象父类`ExpressionElement`，同时`ExpressionElement`具有函数`Calculate`,子类，包括`Value`、`ArgumentX`、`ArgumentY`、`Operator`等等，各自实现不同的`Calculate`操

作。



最后，将`Calculate`方法赋给`Func<>`类型的变量，就可以实现表达式的存储了。

（3）FunctionFactory类——字符串转函数封装信息

但是，不同类型的函数有着不同的信息，比如单变量函数可以方便地求导、积分；参数方程需要指定参数的范围。因此在绘制图像之前，还需要对函数的信息做一次封装。

`FunctionFactory`读入字符串，判断其类型，将其封装为不同的类（`SingleVarFuncStorage`、`DoubleVarFuncStorage`和`ParamVarFuncStorage`。其中各自具有对应函数的属性和方法），并以`FunctionStorage[]`父类数组的形式返回。

2、FunctionDrawing类——图像绘制

（1）基础算法

`FunctionStorage`类的成员将传入`FunctionDrawing`类中，成员被存储在一个`List<FunctionStorage>`内。每次调用`Refresh`函数重绘，都将对列表内的每一个元素调用绘制。

不同的函数类型将采取不同的绘制策略。绘制时先用里氏转换将函数转换会真正的子类。随后调用多态方法`DrawFunction`。

虽然绘制算法并不相同，但本质上都采取了取点描线的方式。绘制利用了`WPF`的绘图API，主要是运用了`DrawingGroup`、`DrawingContext`类。并将绘制结果以`DrawingImage`的形式作为位于主界面左侧窗口的`Image`的`Source`。

（2）平滑算法

单变量函数和参数方程的基本绘制，就是简单的以某一小间隔`dx`为单位，从自变量初值递增到终值，对每一个自变量，计算对应的坐标。最后将所有在绘制范围内的坐标点连接起来。

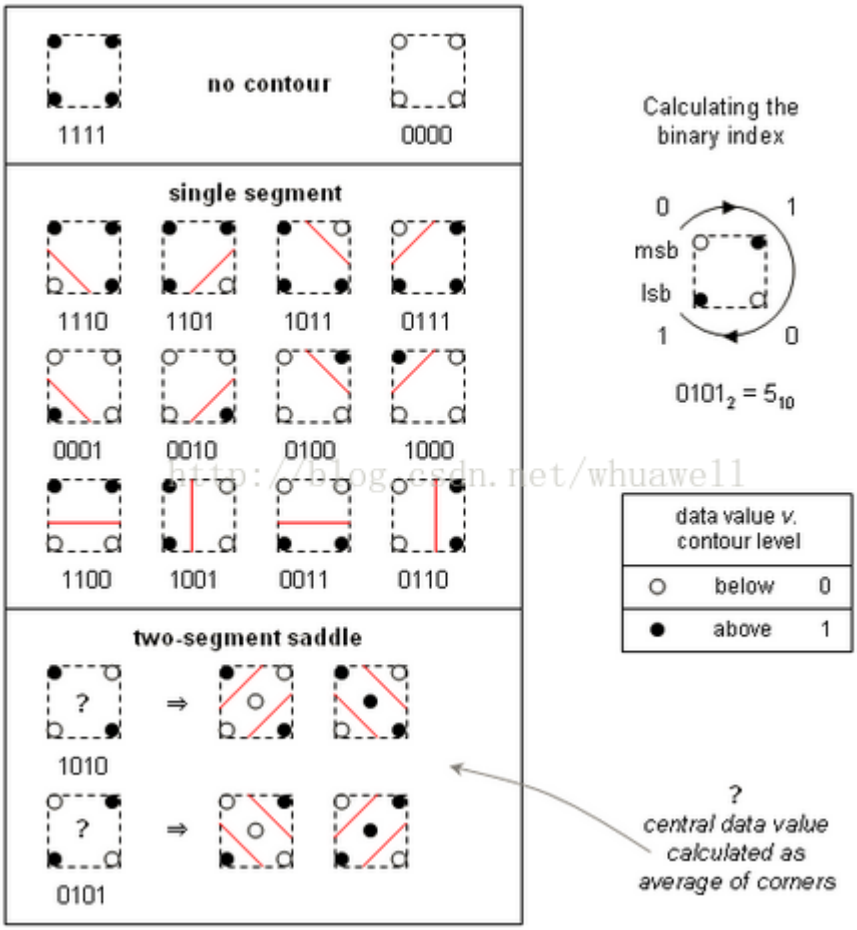
而平滑算法是用于改善一些极端情况，如`y=sin(1/x)`，的绘制效果的算法。用于平滑过于“尖锐”的图像。这是一个递归算法，具体操作如下

1. 输入参数为三个点，构成两条线段
2. 如果这两条线段较为“平滑”（本程序采用斜率之差的绝对值小于某一值），则退出
3. 否则，分别对于两条线段，各自取端点参数的中值，算出坐标，将原线段的端点和中值的坐标作为参数，重复1。

（3）Marching Square算法

对于图像一般形式`f(x,y)=0`，就算遍历全部绘制范围也较难恰好取到等于0的情况。另外如果以某一确定的`eps`作为误差范围，也会因为不同的梯度导致绘制的曲线误差较大。因此需要采用新的算法。

Marching Square算法一定程度上解决了这一问题。该算法通过将绘制区域划分成等大的方块，并取样方块顶点上的函数值，根据函数值的正负估计曲线的走向。最后利用线性插值估计`f(x,y)=0`在方块边上的位置，连线得出图像。



3、UI设计

（1）图像尺寸自适应

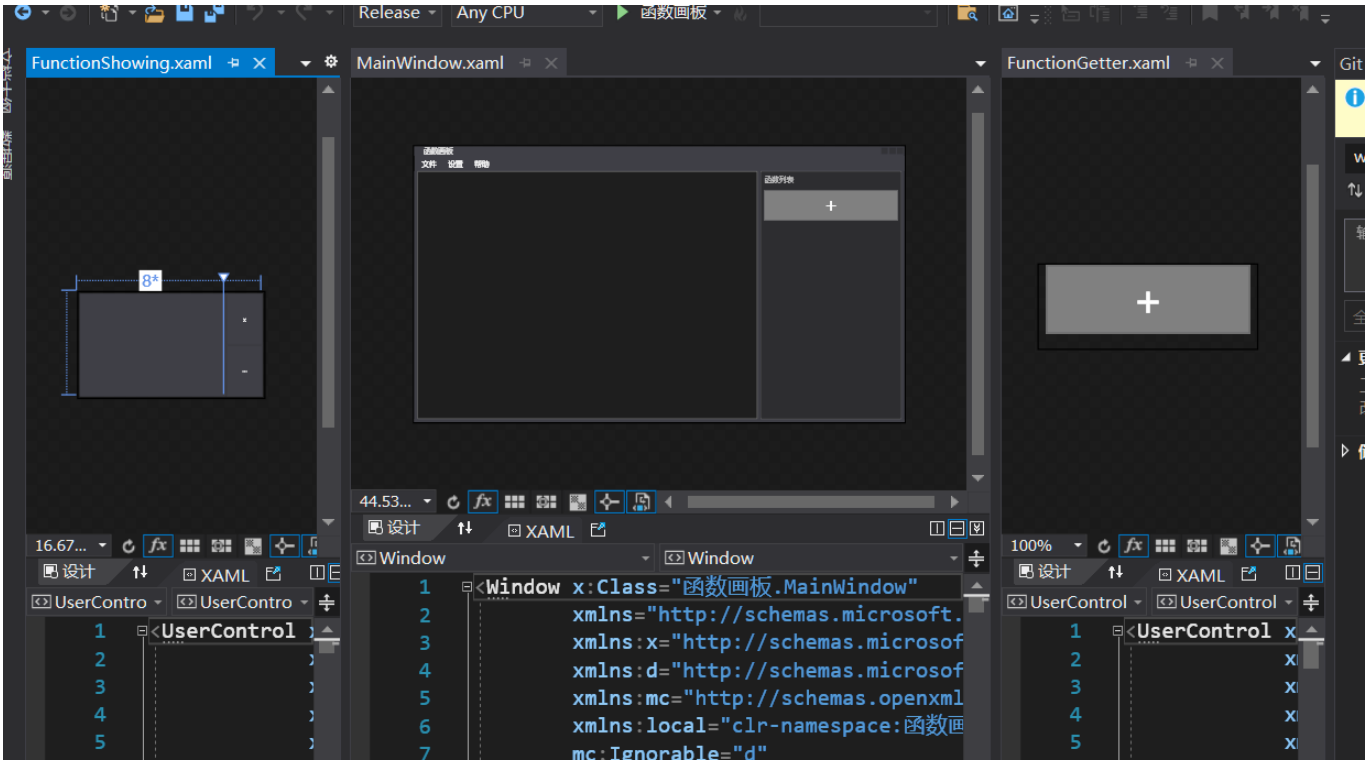
软件需要能实现尺寸的自由变化，而如果只绘制某一特定尺寸的图像，那么就无法实现较好的变化效果。

采取的解决方式是在FunctionDrawing类内添加属性AspectRatio用于表示长宽比。同时监听窗口的尺寸变化事件。在尺寸变化后，获取新的长宽比，更新AspectRatio并调用Refresh方法重绘界面。

（2）UserControl自定义控件

在添加函数后，主界面右侧函数列表会出现记录该函数的信息卡片。并具有删除和设置函数的按钮。该卡片集成了该函数所能做的所有操作，是FunctionStorage在UI的反映。

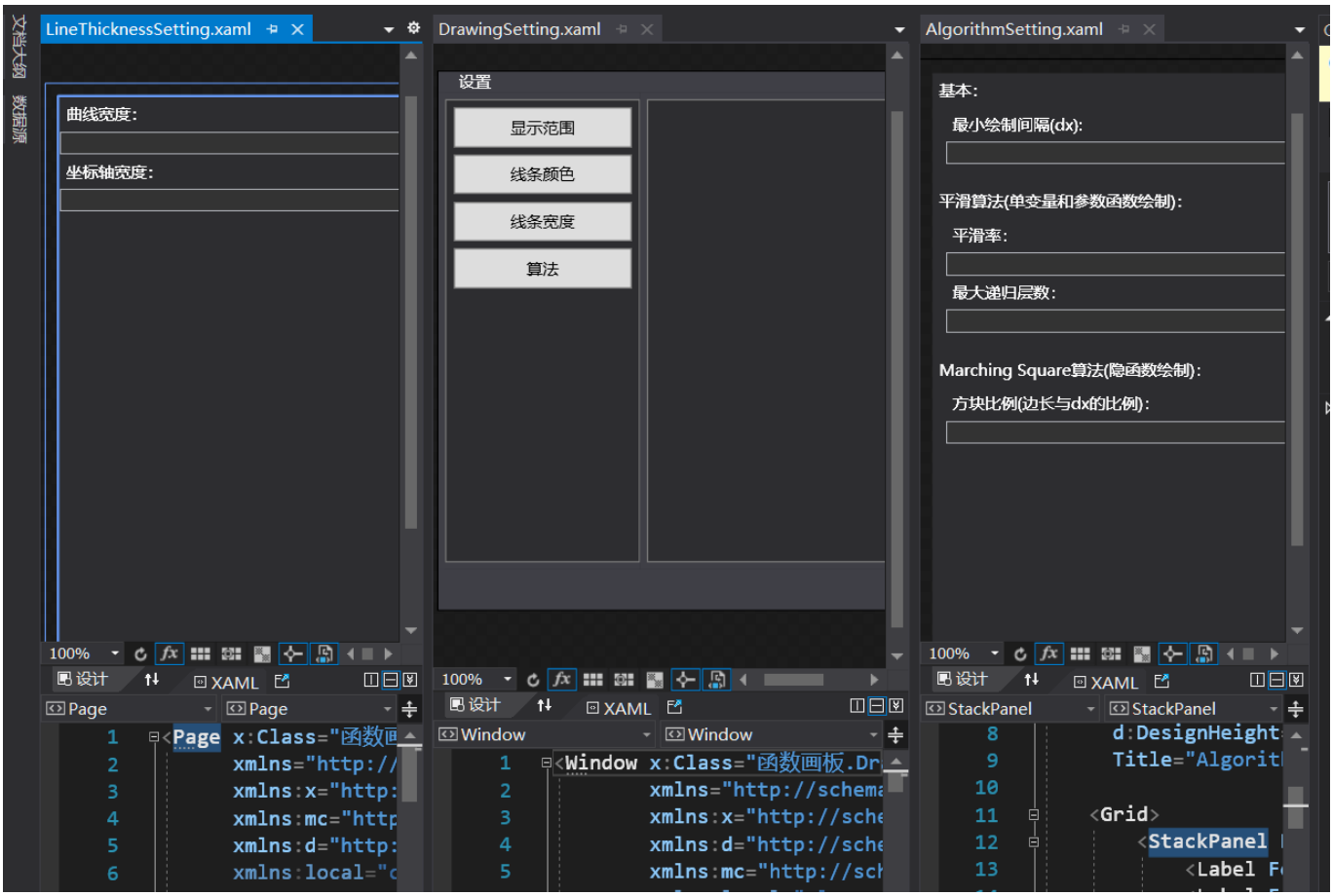
如果每创建一个新的函数，都要执行一大段生成包括Button、Label、TextBlock PopUp等控件的代码，很显然是不明智的。因此采用用户控件UserControl在新的窗口设计信息卡片需要的功能。主页面则只需要在添加函数时创建关于该函数的卡片实例即可。类似的，表达式输入框等内容也是如此。



（3）Frame——Page实现单窗口多页面

点击菜单栏的设置选项，会出现一个用于设置的新窗口。选择左侧边栏的不同设置选项，右侧会显示不同的页面。这个单窗口多页面的功能，由Frame加Page控件实现。

主设置页添加一个Frame控件，在打开该窗口时，Frame加载显示不同设置的Page。在点击左侧边栏按钮的时候，调用Frame的Navigate方法导航到对应的设置页。



三、运行环境

编写时使用Visual Studio版本为VS2019 16.11.17 .NET版本为netcoreapp3.1 c#版本为C#8.0

四、收获

1. 学习了面向对象编程的思想，实践了简单的代码重构。
2. 获得了开发近4000行代码的项目的经验。
3. 学习了一些图形学算法。
4. 对C#的理解更加深入，并获得了简单的WPF经验。