# 测试分析

## 2.1 测试策略与测试方法

界面以及调色控件会使用人工进行测试，属性表达式会通过编码来进行测试

## 2.2自动化测试技术与测试工具。

该软件属于UMP应用，且使用C++的Qt框架进行全部开发，缺乏自动化测试工具，因此测试一般以人工和编码为主。

**3、分模块测试**

**3.1微调控件测试：**

**控件说明：**由于大多数元素的属性都是数值类型，比如长宽高，因此自定义了一个由文本输入框+滑动条+单位增减按钮来进行统一微调的控件，该控件支持整数、浮点数的微调（浮点数需设置精度）。

**控件样式：**



**类：**



**测试说明：**测试用例会从以下的属性来进行人工及编码测试。

该控件的创建需要提供四个值，分别为属性值的缺省值，最小值，最大值以及精度，修改属性值有4条路径：

* 通过文本框输入，按Enter后触发文本框的editFinshed信号进行修改
* 通过拖动滑动条的滑块，触发滑动条的sliderMoved事件进行修改
* 通过使用单位增减按钮调用add，sub函数进行微调
* 使用类通过的公有函数setValue修改

**测试用例：**

测试用例1：当构建控件时，输入的最小值大于最大值

测试用例2：当构建控件时，输入的缺省值不在最小最大值之间

测试用例3：测试通过文本框输入来改变属性值

测试用例4：测试在文本框中输入非法字符来改变属性值

测试用例5：测试在文本框中输入超出范围的数值来修改属性值

测试用例6：测试通过滑动条来修改控件值

测试用例7：测试通过按钮来修改控件值

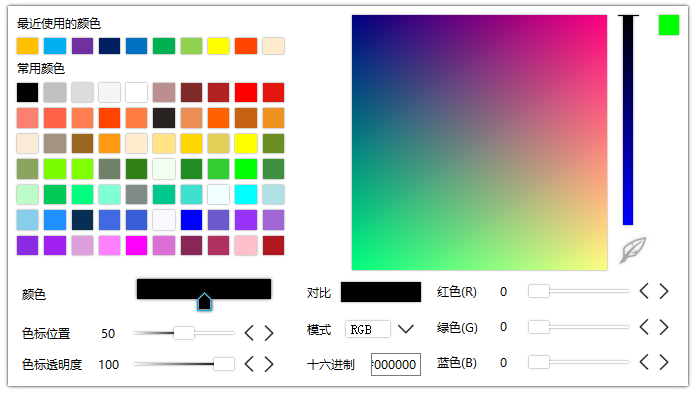
测试用例8：通过调用API来修改控件值

测试用例9：调用API时输入超出范围的数据进行修改。

**3.2 调色控件测试：**

**控件说明：**在做音乐可视化时大多素材需要调色，因此自定义了一个统一的调色控件，该控件支持RGB调色板，HSL调色板，常用颜色及最近使用颜色选择面板，渐变色调整，屏幕取色器，十六进制输入等

**控件样式：**



## **测试说明：**该控件是一个比较庞大的控件集合，测试的时候会划分成更小的模块进行单独测试

**测试用例：**

颜色选择面板测试：

测试1：通过鼠标点选常用颜色块，查看颜色是否变更

测试2：通过鼠标点选常用颜色块，查看最近使用颜色是否变更

测试3：通过鼠标点选最近使用颜色块，查看最近使用颜色列表的处理情况

调色板（RGB，HSL）测试：

测试1：通过点击调色板进行调色

测试2：通过微调控件进行测试

测试3：通过十六进制输入文本测试

测试4：在文本输入框输入非法字符

测试5：将HSL与RGB进行互转，查看调色板的变动情况

渐变条设置：

测试1：通过向下（或上）拖拽色标，进行删除

测试2：当只有一个色标时，对它进行拖拽删除

测试3：通过点击色条上的空白区域添加色标

测试4：重复点击同一位置来添加色标

测试5：通过微调控件调整色标的位置以及透明度

屏幕取色器：

测试1：点击叶子图形，进行屏幕取色

测试2：在屏幕取色过程中，切出窗口，查看处理结果。

**3.3属性表达式功能测试**

功能说明：由于一个图形具有很多属性，而这些属性大多是静态的，动画处理往往是通过处理这些静态属性来完成一个动画的效果。比方说，如果你想让一个图形旋转，按常规我们是通过静态的去设置【旋转速度】这一属性来让图形“动起来”，但本质上，能够旋转的原因是软件内部在动态修改静态属性【旋转角度】，也就是说，我们旋转图形只不过是调整【旋转角度】罢了。而属性表达式的目的就是为了素材能够去掉这些看起来是静态的“动态属性”，而提供一种对那些真正静态的属性进行动态修改的“表达式工厂”。目前这个表达式工厂支持如下操作：

* + 常规表达式+-/\*以及括号优先级的运算，例如”(1+2)\*3”
  + 支持幂运算，运算符号为\*\*
  + 外部变量导入，例如表达式“k\*x+b”，支持导入变量k，x，b来进行求值
  + 支持常用数学三角函数，可以解决如下表达式sin(exp)、cos(exp)、tan(exp)
  + 支持函数扩展，内部可以提供函数处理器，来处理特定函数，比如表达式”sum(1,5,7,4)”，可以提供一个sum()处理器来处理该表达式。

测试说明：测试会对大多数功能以及一些非法的情况进行的测试

测试用例：

测试表达式1：加法测试 1+2

测试表达式2：减法测试 2-1

测试表达式3：乘法测试 3\*4

测试表达式4：除法测试 8/3

测试表达式5：除法测试 1/0

测试表达式6：幂函数测试 2\*\*3

测试表达式7：优先级测试 (2\*(3+5)+6)/3

测试表达式8：外部变量测试 a+b\*cost [a=1,b=2,cost=5]

测试表达式9：三角函数测试 sin(1.57)

测试表达式10：三角函数组合测试 sin(1+0.57)+cos(2\*3.14)

测试表达式11：函数扩展测试 sum(1,5,7)

测试表达式12：括号匹配测试 (1+(2))\*(3)+4)

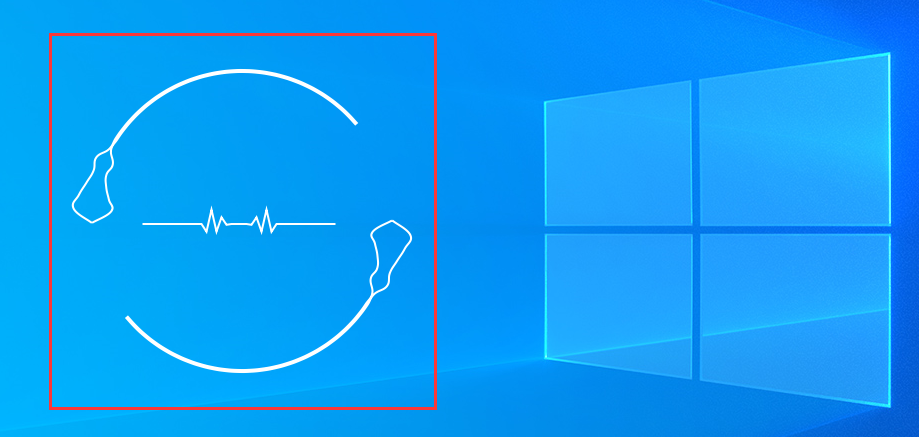
测试表达式13：非法符号测试 asf@afag\*4sdfsd&

测试表达式14：带符号数测试 1\*-1++2\*3

测试表达式15: 时间关联测试 sin(time)

## **3.4中心投影变换功能测试**

**功能说明：**一般在绘制二维图形的时候，都会给以该图形一个矩形的绘制区域，比如下方图形的绘制区域是这样的（红线部分）



假如我们想让这个图形契合到右边的田字方块中，直接移动过去会有很强的违和感（能明显感觉出两个图形并不在一个平面上），而它们明明都是在屏幕上，怎么会这样？这就是投影变换要做的事情。上述的图形，经过中心投影变换后，会得到这样的效果：



通过中心投影变换，我们很轻松的就能把一个图形调整到某一个平面上，让图形看上去就跟立体的一样，这就是中心投影变换的作用。

代码实现（Qt环境下）：实现功能：通过提供两个四边形的顶点集，提供数学运算得到一个转换矩阵，通过这个转换矩阵，我们可以将一个四边形内的点映射到另一个四边形中

测试说明：软件将通过一张矩形图片来调整它的四个顶点位置来进行测试，这里我通过单独实现一个窗口，提供四个可供鼠标拖动的角点来调整顶点。由于需要进行像素级的处理，因此采用OpenGL来进行图片显示。

**被动测试**：









