### Image Beautify 模块报告之 model 层

报告人：叶昕洋

报告日期：2018年7月27号

## 一、MVVM模式中Model层简介

MVVM架构使得View层与Model层完全实现解耦合，Model层专门负责业务逻辑、操作实现，是程序的数据访问层。Model层与ViewModel层的关系最为密切，ViewModel可以通过聚合Model对象直接调用Model的方法对Model中的数据进行更新。当Model触发事件，更新数据或操作完成时，ViewModel层接受事件并传递给View层面。

由此可以发现，Model层的主要工作有一下三个方面：

1. 与数据进行联系并实现对数据的更新，根据需求调用模块对数据进行操作、接收ViewModel层的命令。

2. 实现属性通知的发出器，当操作实现或者数据进行更新时，发出通知给ViewModel。

3. 实现数据的存储，将Model层处理的数据保存，更新现有的数据。

## 二、本项目中的Model层的实现

### 2.1 Model层概览

|  |  |
| --- | --- |
| Model层文件结构 | 描述 |
|  | 在Model层中，我们需要对图片进行操作，我们将对图片的操作分成三类，为滤镜操作、图片代数操作以及日志三个操作。其中由于滤镜操作比较多也比较繁琐，所以我们在filters中存储器具体的操作算法，在filteroperation中调用filters中的函数以实现功能。在imageoperation中存储的图片的代数操作算法，在aux\_image\_alg.cpp中存储具体算法，在imageoperation.cpp中存储返回bool值的函数，若失败则返回false，成功则传递图片。Logoperation同理，aux\_log\_alg.cpp中存储具体算法，在Logoperation.cpp中存储返回bool值的函数，若失败则返回false，成功则传递图片。 |

### 2.2 Model层的数据操作

|  |  |
| --- | --- |
| 基础操作 | |
| 函数名 | 实现功能 |
| bool **open\_file**(const QString &path);  bool **open\_sub\_file**(const QString &path); | 图片的打开功能 |
| bool **save\_file**(const QString &path); | 图片的存储功能 |
| bool **sub2main**();  bool **origin2main**();  bool **main2sub**();  bool **sub2tmp**(); | 监视图片的数据变化 |
| bool **mix\_tmp\_main**(int alpha); | 将图片进行叠加处理 |
| 图象操作 | |
| bool **imageAdd**(double param1, double param2);  bool **imageSubtract**(double param1, double param2);  bool **imageMultiply**(); | 图象的加减乘除操作 |
| bool **getSingleChannel**(ChannelType channel);  bool **grayScale**(); | 图象获得单颜色通道  图象灰度化 |
| bool **adjustHue**(QVector<int> hueValues);  bool **adjustSaturation**(int saturation);  bool **adjustLightness**(int lightness); | 调整图象的色相、饱和度与亮度 |
| bool **otsu**();  bool **dualThreshold**(int thresh1, int thresh2); | 图象二值化 |
| bool **nearnestInterpolation**(int scale, int rotation);  bool **BilinearInterpolation**(int scale, int rotation); | 插值 |
| bool **meanFilter**(int col, int row, int x, int y);  bool **medianFilter**(int col, int row, int x, int y);  bool **gaussianFilter**(int col, int row, int x, int y, double sigma); | 滤波器 |
| bool **sobelEdgeDetection**(int threshold);  bool **laplacianEdgeDetection**(int threshold);  bool **cannyEdgeDetection**(int lo, int hi); | 边缘检测 |
| bool **houghLineDetect**();  bool **houghCircleDetect**(int lo, int hi); | 线路检测  圆检测 |
| bool **dilation**(int size, int x, int y, int \*array);  bool **erosion**(int size, int x, int y, int \*array);  bool **opening**(int size, int x, int y, int \*array);  bool **closing**(int size, int x, int y, int \*array); | 图象的腐蚀和膨胀 |
| bool **obr**(int size, int x, int y, int \*array);  bool **cbr**(int size, int x, int y, int \*array); | 形态学操作 |
| bool **linearContrastAdjust**(int x1, int y1, int x2, int y2);  bool **pieceLinContrastAdjust**(int x1, int y1, int x2, int y2);  bool **logContrastAdjust**(double a, double b);  bool **expContrastAdjust**(double a, double b); | 对比度调整 |
| bool **histogramEqualization**(int \*histo);  bool **colorLevel**(PColorLevelData clData); | 直方图均衡化 |
| 日志操作 | |
| bool **redo**();  bool **undo**(); | 撤回操作 |
| bool **clear**(); | 清空操作 |
| 滤镜操作 | |
| void **\_emboss**();  void **\_sculpture**();  void **\_dilate**();  void **\_erode**();  void **\_frostGlass**();  void **\_sketch**();  void **\_oilPaint**();  void **\_woodCut**();  void **\_inverted**();  void **\_memory**();  void **\_freezing**();  void **\_casting**();  void **\_comicStrip**();  void **\_sharpen**();  void **\_colortoblack**();  void **\_defog**();  void **\_soft**();  void **\_balance**();  void **\_nostalgia**();  void **\_BlackComic**();  void **\_timetuunel**();  void **\_classiclomo**();  void **\_whiteFace**();  void **\_beautifyFace**();  void **\_pinkLady**(); | // 浮雕  // 雕刻  // 虚幻  // 惊悚  // 磨砂玻璃  // 手稿  // 油画  // 木刻  // 反色  // 回忆  // 冰冻  // 熔铸  // 黑白漫画  //锐化  //黑白  //去雾气  //柔和  //均衡图  //怀旧  //连环画  //时光隧道  //经典lomo  // 美白  // 美颜  // 粉红佳人 |

### 2.3 Model层与ViewModel层的信息传递

|  |  |
| --- | --- |
| 属性变化 | |
| ViewModel中的函数聚合部分 | ViewModel中的数据指针 |
|  |  |
| 命令执行 | |
| ViewModel中的函数聚合部分 | ViewModel中的数据指针 |
|  |  |

### 2.4 Model层属性通知信号发送器

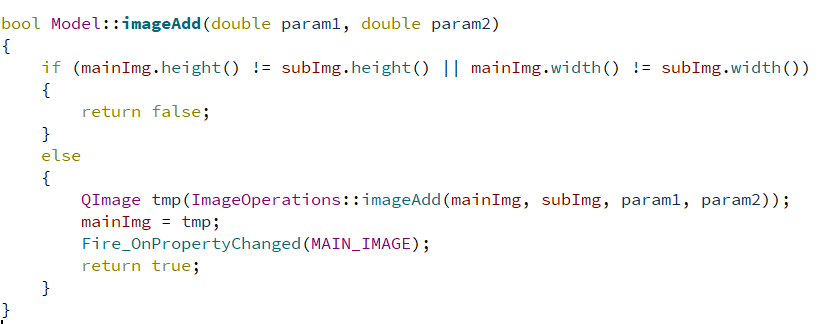
Model层的属性通知信号发送器负责当图片的属性发生改变时发出相应的通知给ViewModel层，通过ViewModel层传递给View层。每一个操作都在判断其是否成功，然后发送器发出指令。



三个函数监视着数据中图片的变化，若属性发生变化，则立刻返回正确的布尔值给ViewModel层。

### 2.5Model层命令通知接收器

接下来以图象的相加为例来解释当图片的操作成功时是如何返回布尔值给Modelview。在Model层中有一个实体操作，和一个返回布尔值的操作。后者调用前者实现相应的操作，同时当操作成功时，后者会返回一个正确的布尔值给ViewModel层。



在这个函数中，我们先调用了imageAdd这个实体操作函数，得到我们想要的结果后赋值给mainImg这个存储结构，监视其数据是否发生变化，返回布尔值。

其他操作结构与图象的加法结构大致相同，引用一个实体的操作函数，同时监视属性值的变化，若变化，则返回一个正确的布尔值。