需要手动添加/修改的代码

ImageProcess.cpp:

6-11,41-78,447-635

ImageProcess.h

7-105,110-115 的注释,121-133 的申明及属性改动，141-145 的构造函数，162-167 光栅化，268-282 的函数申明

图元对接说明：

功能

图元生成：需提供图元类型，左上角右下角位置（用一个矩形把图元刚好框起来）

```
auto image2 = new Layer(TRIANGLE, 0, 0, 200, 100);
auto image3 = new Layer(LINE1, 100, 100, 500, 500);
auto image13 = new Layer(LINE2, 100, 100, 500, 500);
```

参数依次：图元类型，左上 X，左上 Y，右下 X，右下 Y

需要注意的是，直线区分两种类型，斜率不同。

采用如下函数加入显示队列。（一定要用*）

```
newProcessor.Layers.addLayerAsTop(*image2);
newProcessor.Layers.addLayerAsTop(*image3);
newProcessor.Layers.addLayerAsTop(*image13);
```

图元移动与缩放，颜色，尺寸修改：

通过静态类调用，移动提供整体的 dx，dy，缩放提供两个点的 dx，dy

改变颜色传入 BGR 值（顺序别错了），改变绘制线条的粗细传入 size（-1 为内部填充）

```
ImageProcess::movePrimitive(newProcessor, newProcessor.Layers.front() , 100, 100);
newProcessor.Layers.addLayerAsTop(*image2);
ImageProcess::scalePrimitive(newProcessor, newProcessor.Layers.front(), 0, 0, 100, 100);
newProcessor.Layers.addLayerAsTop(*image2);
ImageProcess::changePrimitiveColor(newProcessor, newProcessor.Layers.front(), 150, 100, 100);
ImageProcess::movePrimitive(newProcessor, newProcessor.Layers.front(), 200, 200);
ImageProcess::changePrimitivePenSize(newProcessor, newProcessor.Layers.front(), -1);
//ImageProcess::scalePrimitive(newProcessor, *image4, 100, 100, BOTTOM);
```

文字类似。

```
auto image4 = new Layer("Hello!", 200, 100);
```

采用如下函数加入显示队列。（一定要用*）

```
newProcessor.Layers.addLayerAsTop(*image2);
newProcessor.Layers.addLayerAsTop(*image3);
newProcessor.Layers.addLayerAsTop(*image13);
```

```
//图元图层操作
static void drawPrimitive(MAT& src, ElementType e, int leftUpX, int leftUpY, int rightDownX,
    int rightDownY, int size, double B, double G, double R);
static void movePrimitive(ImageProcess &process, Layer &layer, int dx, int dy);     //移动图元图层
static void scalePrimitive(ImageProcess &process, Layer &layer, int leftUpdx, int leftUpdy,
    int RightDowndx, int RightDowndy);
static void changePrimitiveColor(ImageProcess &process, Layer &layer, double B, double G, double R);
static void changePrimitivePenSize(ImageProcess &process, Layer &layer, int size);

//文字图层操作
static void drawText(MAT& src, std::string s, int leftUpX, int leftUpY, int rightDownX, int rightDownY,
    int size, int scale, FondFace face, double B, double G, double R);
static void changeTextColor(ImageProcess &process, Layer &layer, double B, double G, double R);
static void ImageProcess::changeTextThickness(ImageProcess &process, Layer &layer, int thickness);
static void ImageProcess::changeTextScale(ImageProcess &process, Layer &layer, int scale);
static void ImageProcess::changeTextFace(ImageProcess &process, Layer &layer, FondFace face);
static void ImageProcess::moveText(ImageProcess &process, Layer &layer, int dx, int dy);
static void ImageProcess::rewriteText(ImageProcess &process, Layer &layer, std::string t);
```

画笔参数修改，提供修改对应的参数，调用静态方法即可：

```cpp
//使用一个静态类储存当前画笔的颜色，线宽，在创建对应图元时将信息存入对应图元图层,cv暂不提供笔触，硬度等参数
static class penParameter {
private:
    static double B;
    static double G;
    static double R;
    static int size;    //线宽，-1为实心
    static double scale;
    static FondFace face;

public:
    static void setPenColor(double b, double g, double r) { B = b; G = g; R = r; }
    static double getPenColorB() { return B; }
    static double getPenColorG() { return G; }
    static double getPenColorR() { return R; }
    static void setPenSize(int s) { size = s; }
    static int getPenSize() { return size; }
    static void setPenScale(double s) { scale = s; }
    static double getPenScale() { return scale; }
    static void setPenFace(FondFace f) { face = f; }
    static FondFace getPenFace() { return face; }
};
```