Ex1 测试文档

测试环境：windows10，Visual Studio 2017

要使用 cimg，只需从官网下载 cimg.h 的源代码，然后添加到工程里即可。

每步操作都写成了类的形式（Ex1 类）。类提供了两种构造函数的形式：

```
Ex1::Ex1() {
    CImg<unsigned char> tempImg(300, 300, 1, 3);
    tempImg.fill(0);
    img = tempImg;
}

Ex1::Ex1(const char* str) {
    img.load_bmp(str);
}
```

每个操作函数的作用和声明如下：

```
class Ex1 {
public:
    Ex1();                          //默认构造图像
    Ex1(const char*);               //现有图像
    ~Ex1();
    void display();                 //第一题
    void change_color();            //第二题
    void blue_circle1();            //第三题不使用CImg的函数调用
    void blue_circle2();            //第三题使用CImg的函数调用
    void yellow_circle1();          //第四题不使用CImg的函数调用
    void yellow_circle2();          //第四题使用CImg的函数调用
    void drawLine1();               //第五题不使用CImg的函数调用
    void drawLine2();               //第五题不使用CImg的函数调用
    void save(const char*);    //第六题
private:
    CImg<unsigned char> img;
};
```

1. 读入 1.bmp 文件，并用 CImg.display()表示。

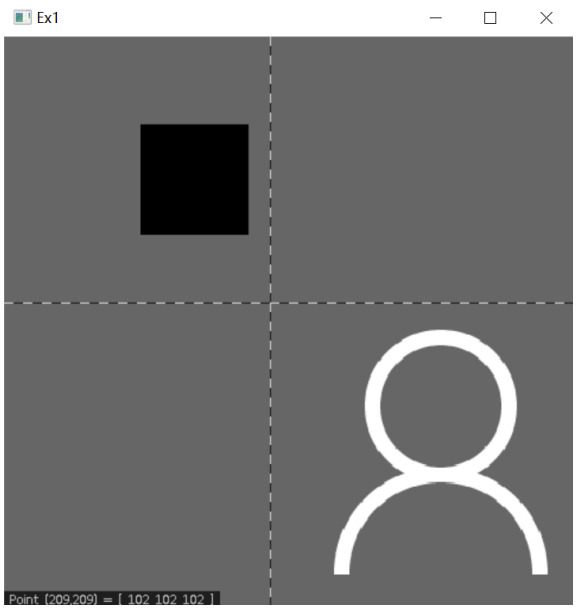
这个没什么好说的：

```
void Ex1::display() {
    img.display("Ex1");
}
```

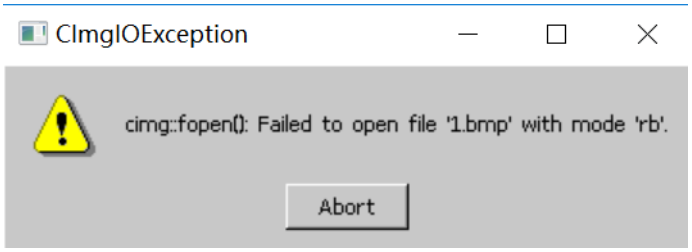
编译运行后，会先弹出控制台，显示这个图片的参数：

```
Ex1: this = 0075FCEC, size = (448,448,1,3) [588 Kio], data = (unsigned char*)02710040..027A303F (non-shared) = [ 102 102 102 102 102 102 102 102 ... 102 102
102 102 102 102 102 102 ], min = 0, max = 255, mean = 103.318, std = 33.5957, coords_min = (107,69,0,0), coords_max = (339,230,0,0).
```

然后 1.bmp 会显示出来：



如果读入一个不存在的文件或者 1.bmp 与代码不在同一个文件路径里，那么程序会报错：



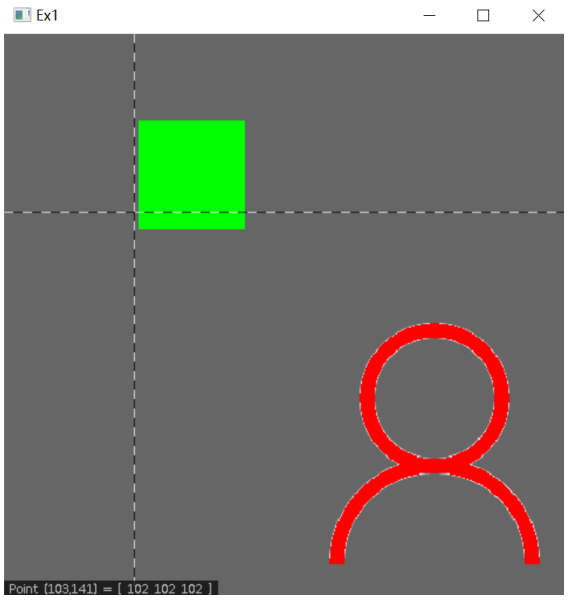
2. 把 1.bmp 文件的白色区域变成红色，黑色区域变成绿色。

利用 cimg_forXY 对逐个像素的颜色判断然后修改成对应的颜色即可。

常用颜色的 RGB 值可以通过百度百科得到，代码如下：

```
void Ex1::change_color() {
    cimg_forXY(img, x, y) {
        if (img(x, y, 0) == 0 && img(x, y, 1) == 0 && img(x, y, 2) == 0) {
            img(x, y, 0) = 0;
            img(x, y, 1) = 255;
            img(x, y, 2) = 0;
        }
        else if (img(x, y, 0) == 255 && img(x, y, 1) == 255 && img(x, y, 2) == 255)
            img(x, y, 0) = 255;
            img(x, y, 1) = 0;
            img(x, y, 2) = 0;
        }
    }
}
```

具体的效果如下：



3. 在图上绘制一个圆形区域，圆心坐标(50, 50)，半径为 30，填充颜色为蓝色。

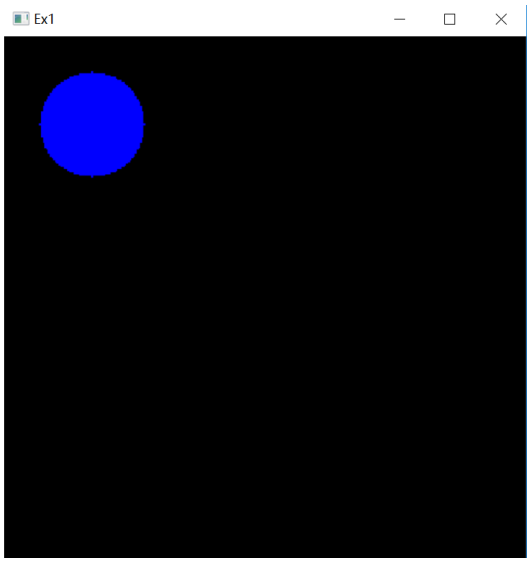
首先不调用 CImg 相关函数，那么直接判断每个像素点与(50, 50)之间的距离。如果距离小于 30，则填充为蓝色即可：

```
void Ex1::blue_circle1() {
    cimg_forXY(img, x, y) {
        double dx = (x - 50) * (x - 50);
        double dy = (y - 50) * (y - 50);
        double distance = sqrt(dx + dy);
        if (distance <= 30.0) {
            img(x, y, 0) = 0;
            img(x, y, 1) = 0;
            img(x, y, 2) = 255;
        }
    }
}
```

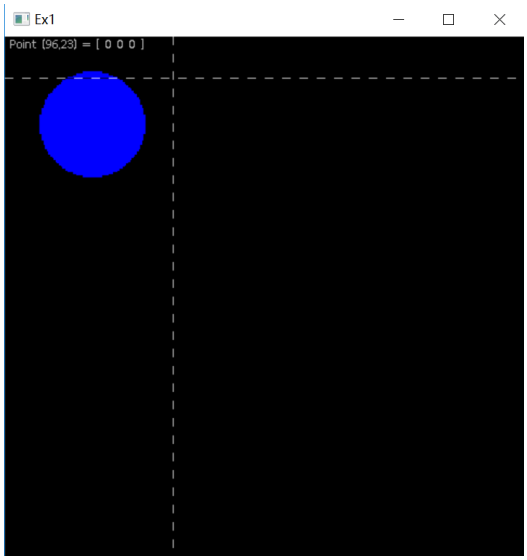
调用 CImg 相关函数的话比较简单：

```
void Ex1::blue_circle2() {
    unsigned char blue[] = { 0, 0, 255 };
    img.draw_circle(50, 50, 30, blue);
}
```

具体效果：首先是不调用 CImg 相关函数，显示的图像如下：



调用 CImg 相关函数的图像如下：

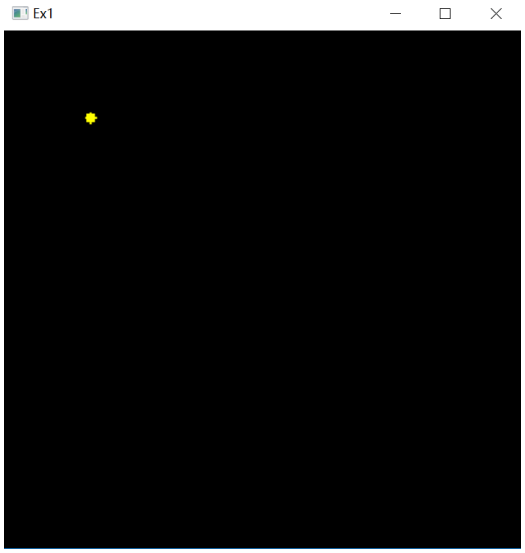


从视觉上看，两者看不出多大的差别。

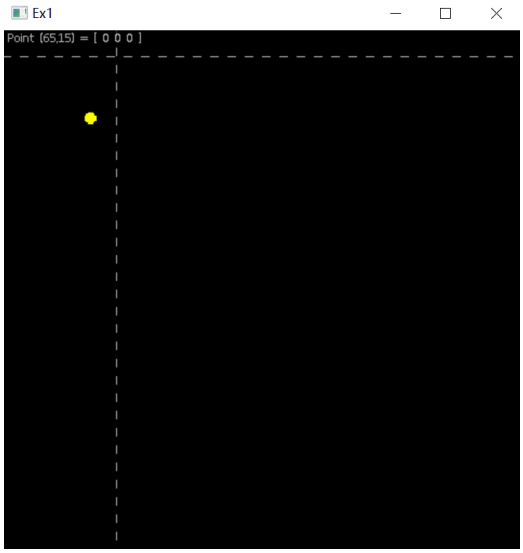
4. 在图上绘制一个圆形区域，圆心坐标(50, 50)，半径为 3，填充颜色为黄色。

实现思路与第 3 题一样，就不贴代码上来了。

具体效果：首先是不调用 CImg 相关函数，显示的图像如下：



调用 CImg 相关函数的图像如下：



从视觉上看，两种方法实现的效果都不是太好。因为像素点的坐标为整数值，半径越小说明采样的像素点越少，而第 4 题圆的半径只有 3，能用的像素点非常有限，所以会出现锯齿形状，绘制效果不好。值得一提的是，虽然两者效果都不是很好，但是调用 CImg 相关函数的效果还是要好了不少，估计 CImg 的函数里有更加好的方法来画圆（看起来有点复杂，我没有仔细去看怎么实现的）。

5. 在图上绘制一条长为 100 的直线段，起点坐标为(0, 0)，方向角为 35 度，直线的颜色

为蓝色。

首先是不调用 CImg 相关函数。直线的斜率是确定的，我们可以通过题目的条件获取直线的终点坐标：

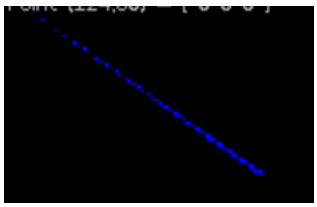
```
double dx = 100 * cos(35 * M_PI / 180);
double dy = 100 * sin(35 * M_PI / 180);
```

其中 M_PI 是圆周率的宏定义。由于 vs 的 cmath 标准库里没有包含这个宏定义，所以需要在头文件中自己定义。

然后我先想通过坐标比值与 tan 值近似相等（由于精度问题，不可能完全相等）的方式筛选出相应的像素点来染色：

```
if(abs((double)y / (double)x - tan(35 * M_PI / 180)) <= 0.01 && x <= dx && y <= dy) {
```

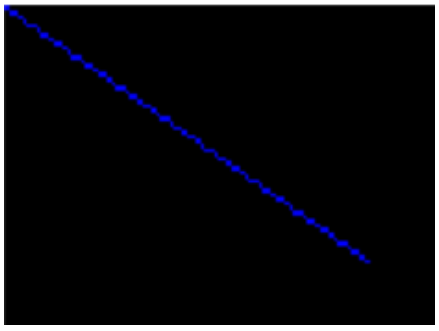
但是效果并不好，因为随着坐标的增大，比值符合范围的区域越大，这就导致了直线越来越粗的情况：



所以这里不用比值而是用乘法的方式来进行约束，试了几次发现 0.5 的误差绘制出的直线相对比较好：

```
if(abs(y - x * tan(35 * M_PI / 180)) < 0.5 && x <= dx && y <= dy) {
    img(x, y, 0) = 0;
    img(x, y, 1) = 0;
    img(x, y, 2) = 255;
}
```

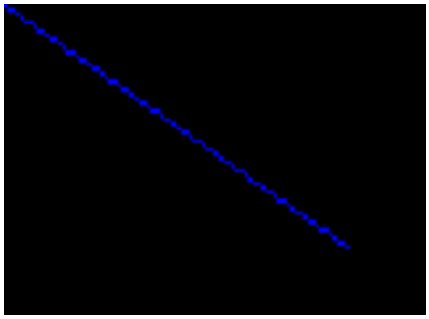
效果如下：



调用 CImg 相关函数的代码如下：

```
void Ex1::drawLine2() {
    unsigned char blue[] = { 0, 0, 255 };
    img.draw_line(0, 0, 100 * cos(35 * M_PI / 180), 100 * sin(35 * M_PI / 180), blue);
}
```

结果如下：



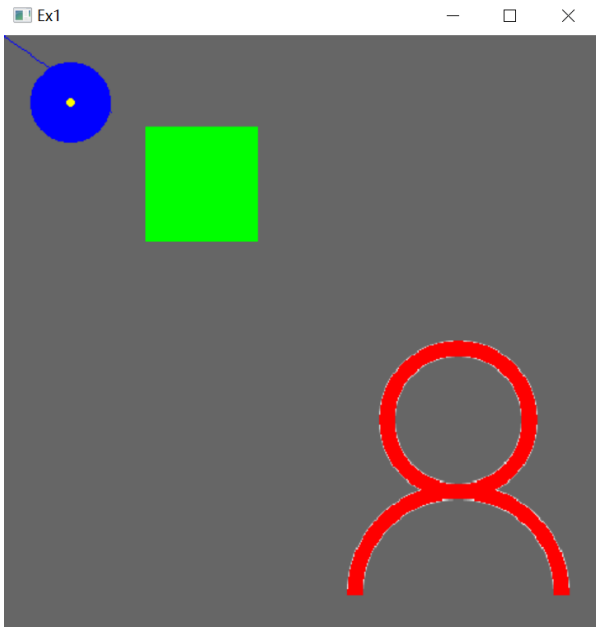
感觉差别并不大，还算比较直但是都有毛刺。

6. 把上面的操作保存为 2.bmp。

这个没什么好说的。

```
void Ex1::save(const char* str) {
    img.save(str);
}
```

所有操作完成后的效果图如下：



测试代码也没什么好说的，直接调用类的方法即可（想测试调用 CImg 相关函数方法的最终效果，改一下调用的函数即可）：

```
#include "Ex1.h"

int main() {
    Ex1 SrcImg("1.bmp");
    SrcImg.change_color();
    SrcImg.blue_circle1();
    SrcImg.yellow_circle1();
    SrcImg.drawLine1();
    SrcImg.save("2.bmp");
    SrcImg.display();

    return 0;
```

总结：这次作业主要还是对 CImg 有一个初步的了解，主要的难点在画直线的操作。总的来说，这为我能够攻克以后更难的作业项目打好了基础。

至于能得多少分，95 应该。。可以吧？（不敢打 100）