1. **问题概述：**

摄像头视频预览及图像采集：Media Foundation是微软推出的新一代多媒体处理框架，附件中给出的项目MFCaptureD3D是Windows SDKs中给出的用于摄像头视频预览的例子。它可以从摄像头视频流中获取每帧图像数据，将数据解码后以Directx 3D方式显示在指定控件上。这次作业在这个项目的基础上展开。

具体要求：

（1）阅读代码，理解其基本框架和实现原理，给出一个相应阅读报告。

（2）在该代码的基础上，增加一种图像处理方法（自己设计），使预览到的为处理后的视频图像。

（3）再增加一个功能，能够将当前预览图像保存为BMP格式文件。

2. **问题特点与分析：**

a) 图像的问题看似和以前遇到过的其他通过数学公式便能得到答案的问题不同，不同之处便在于我们不清楚它的处理方式，图像处理究竟是对图像的哪一种属性进行处理？它所对应的代码是哪一部分？怎么样修改这段代码才能修改图像最后呈现出的样子？

b) 项目所包含的文件数量以及每个文件所包含的函数数量都是很多的，从中如何将需要弄懂的函数和不需要弄懂的函数、需要修改的函数和不需要修改的函数区分出来是一个关键的问题。

c) 很明显图像处理是一个多线程的问题，一个线程取得摄像头权限之后就疯狂地将图像投射到窗口中来，另一个线程则监听键盘以及鼠标传来的消息，根据这些消息对图像进行一定的操作。

3. **设计思路**

a) 首先，通过 MFC 设计窗口在菜单栏中添加各个相机模式的选项，并通过他们独自的 id 在 OnCommand 函数中设置监听

b) 设置全球变量 flag，当相应的监听被触发时，修改全球变量 flag 的值。在 DrawFrame 函数中通过判断 flag 的值调用对应的函数执行操作，并将 lr.pBits 当做参数传递进函数中。

c) 在 WindowProc 中实现键盘监听，当按下 F1~F7 对应键时调整 flag 的值，使其能调用对应的图像处理函数，当按下方向键上下左右时，调节亮度

d) 设置全球变量 bf 以及 key，当按下方向键时，使 bf=1，key 为亮度调节的临时对应数值，在 DraeFrame 中的函数调用顺序为先判断 bf，在原图像调节完亮度的基础上再进行接下来调用函数进行图像处理

e) 在函数中，将指针的类型变为 RGBQUAD

```
typedef struct tagRGBQUAD {
        BYTE    rgbBlue;
        BYTE    rgbGreen;
        BYTE    rgbRed;
        BYTE    rgbReserved;
} RGBQUAD;
```

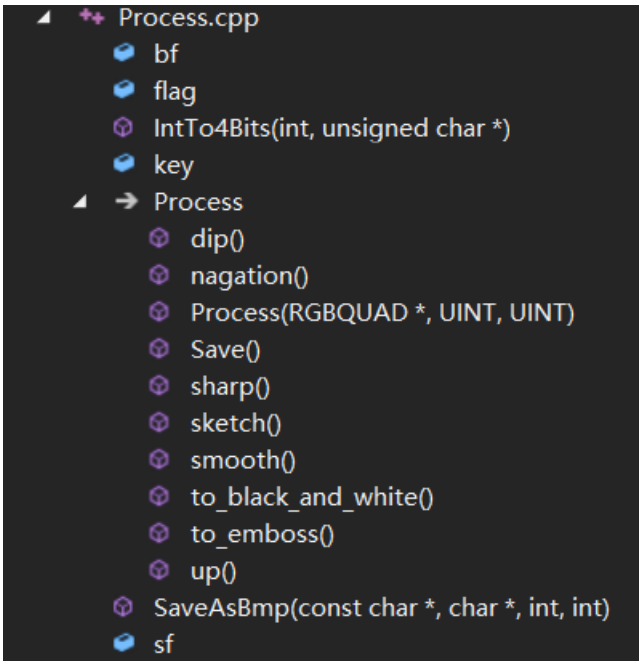
，这样便可以用 RGBQUAD[i]的方式调

用第 i 个像素，并用 RGBQUAD[i].rgbBlue 的方式调用这个元素中的各个属性。

f) 用老师所给的函数 Save()来保存图像

4. 具体实现

a) 设计一个类 Process



i. **flag**:用于判断调用图像处理的哪一种模式

ii. **bf**: 用于判断是否调节亮度

iii. **key**: 用于存储亮度调节的值

iv. **sf**:用于判断是否执行保存为 BMP 图像

v. **Process()**:构造函数，变量依次为图像数据、图像宽度、图像高度

vi. **to_black_and_white()**:处理为黑白图像

vii. **nagation()**:颜色翻转，深色变浅色，浅色变深色

viii. **to_emboss()**:浮雕

ix. **smooth()**:柔化，使色块的边界变得模糊

x. **dip()**:只有纯黑色或只有纯白色

xi. **sharp()**:锐化，即描边处理

xii. **sketch()**:素描处理

xiii. **up()**:调节亮度

xiv. **Save()**:保存图像

b) 在 WindowProc()以及 OnCommand()中设置键盘 F1~F7 监听、键盘上下左右方向键监听、键盘 ESC 键监听以及窗口菜单栏点击事件监听

c) 在 DrawFrame()中创造 Process 对象，将 lr.pBis,m_width,m_height 当做参数传入此对象的构造函数中。接下来判断 bf 的值，若为 1 则执行 up()函数进行亮度调节然后判断 flag 的值执行图像处理对应函数

d) 在 Process.cpp 中有各个函数对应的实现方法，其思想基本上都是操作像素 RGB 的各个值的大小来进行操作，如在 nagation（）操作中将 rgbBlue=255-rgbBlue,rgbGreen=255-rgbGreen,rgbRed=255-rgbRed,因为每个 rgb 只有从 0 到 255 这 256 个值，所以这样便完成了图像颜色的

翻转，同理，用类似的公式可以直接将所有的功能都实现。

e) 关于保存为 BMP 图像，只需要将老师所给的代码修改成以下这一段即可：

```
//修改图像数据
for(i=0;i<w*h*4;i+=4)
{
    data[i]=((i%254)-127);
    //data[i]=255;
    data[i+1]=0;
    data[i+2]=0;
    data[i+3]=0;
}
```

（修改前）

```
//修改图像数据
for (i = 0; i < w*h; i += 1)
{
    data[4 * i] = (pframe[i].rgbBlue);
    data[4 * i + 1] = (pframe[i].rgbGreen);
    data[4 * i + 2] = (pframe[i].rgbRed);
    data[4 * i + 3] = 0;
}
```

（修改后）

即将当前图像数据保存进 data 数组中，接下来再通过 SaveAsBmp 来保存图像即可。

5. **给分要求：三人均分**