深 圳 大 学 实 验 报 告

课程名称：数字图像处理

实验项目名称：图像特效显示实验

学院：计算机与软件学院

专业：计算机科学与技术

指导教师：吴惠思 教授

报告人：林浩晟 学号：2022280310

实验时间：2025 年 3 月 10 日

实验报告提交时间：2025 年 3 月 11 日

教务部制

一、实验目的：

1. 掌握图像特效显示原理
2. 掌握开/关门特效显示实现方法
3. 掌握百叶窗特效显示实现方法
4. 掌握淡入/出特效显示实现方法

二、实验原理：

1、显示图像部分(矩形)区域函数

2、实现开/关门特效显示函数

- 由中间向左右(开门)显示
- 由左右向中间(关门)显示
- 由中间向上下(开门)显示
- 由上下向中间(关门)显示

3、实现百叶窗特效显示函数

- 向右显示
- 向左显示
- 向下显示
- 向上显示

4、实现淡入/淡出特效显示函数

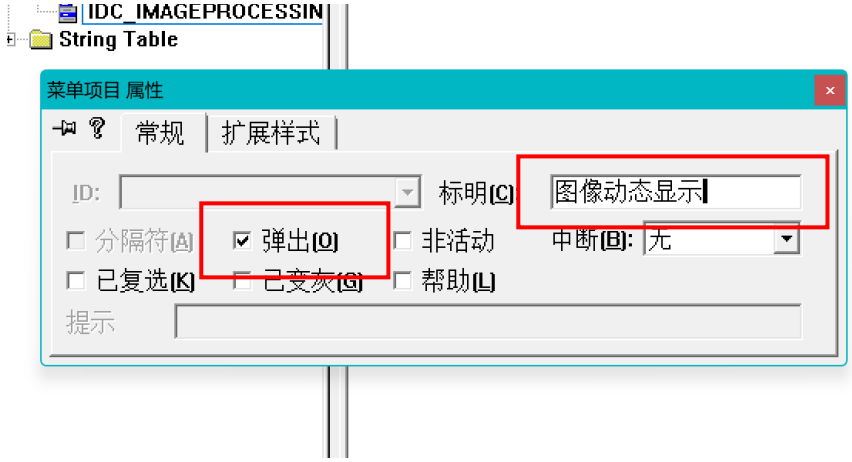
三、实验用品：

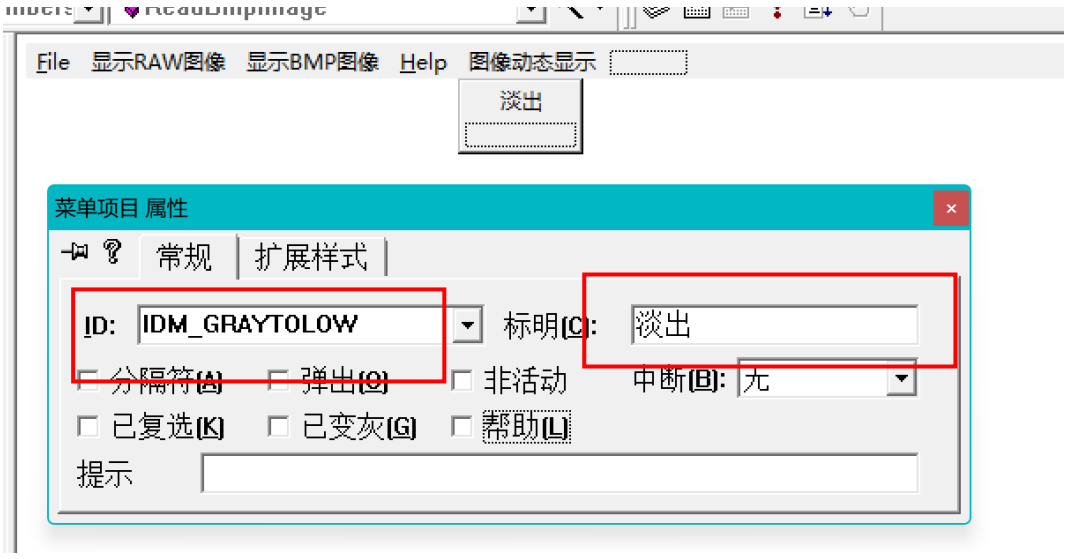
计算机、visual C++6.0

四、实验过程及内容：

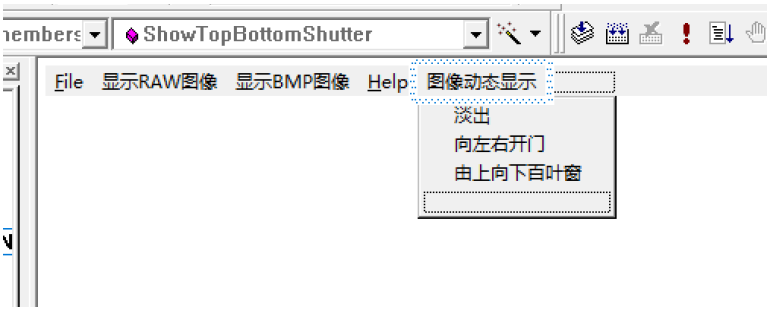
1. 按顺序点击

2. 按照指引操作；





同理，将其余选项添加，如下：



3. 在 Imageprocessing.cpp 中添加相应的代码；

```
case WM_CREATE:
    hWinDC = GetWindowDC(hWnd);
    break;

case WM_COMMAND:
    wmId = LOWORD(wParam);
    wmEvent = HIWORD(wParam);
    // Parse the menu selections:
    switch (wmId)
    {
    case IDM_GRAYTOLOW:                          //淡出
        ShowGrayToLow(OrgImage, IMAGEWIDTH, IMAGEHEIGHT, XPOS, YPOS);
        break;
    case IDM_LEFTRIGHTOPEN:                      //向左右开门
        ShowLeftRightOpen(OrgImage, IMAGEWIDTH, IMAGEHEIGHT, XPOS, YPOS);
        break;
    case IDM_TBSHUTTER:                          //由上向下百叶窗
        ShowTopBottomShutter(OrgImage, IMAGEWIDTH, IMAGEHEIGHT, XPOS, YPOS);
        break;
    case IDM_SHOWRAWIMAGE:
        OpenImageFileDlg("打开图像文件");
        ReadImage(ImgDlgFileName, OrgImage, IMAGEWIDTH, IMAGEHEIGHT);
        ShowImage(OrgImage, IMAGEWIDTH, IMAGEHEIGHT, XPOS, YPOS);
        break:
```

4. 随后，按照 PPT 的代码提示完善函数；淡出函数如下，PPT 中的函数多了一个右括号，正确如下：

```
void ShowGrayToLow(char* Image, int wImage, int hImage, int xPos, int yPos)
{
    int i, j, m;        int r, g, b, gray;
    for (m = 1; m < GRAYCHANGENUM; m++)
    {
        for (i = 0; i < hImage; i++)
        {
            for (j = 0; j < wImage; j++)
            {
                gray = (BYTE)Image[i * wImage + j];
                r = g = b = gray * (GRAYCHANGENUM - m) / GRAYCHANGENUM;
                SetPixel(hWinDC, j + yPos, i + xPos, RGB(r, g, b));
            }
        }
    }
}
```

5. 以下为由中间向左右(开门)显示函数，同时写上了注释；

```
void ShowLeftRightOpen(char* Image, int wImage, int hImage, int xPos, int yPos)
{
    int i;
    RECT ShowRect;
    InvalidateRgn(hWind, NULL, TRUE);  // 清空窗口
    UpdateWindow(hWind);

    ShowRect.top = 0;
    ShowRect.bottom = hImage;

    // 从中间向左右逐渐打开
    for (i = 0; i < wImage / 2; i++) {
        ShowRect.left = wImage / 2 - i;            // 左边界逐渐向左扩展
        ShowRect.right = wImage / 2 - i + 1;        // 显示一列
        ShowImageRect(Image, wImage, hImage, xPos, yPos, ShowRect);  // 左开门

        ShowRect.left = wImage / 2 + i;            // 右边界逐渐向右扩展
        ShowRect.right = wImage / 2 + i + 1;        // 显示一列
        ShowImageRect(Image, wImage, hImage, xPos, yPos, ShowRect);  // 右开门

        Sleep(1);  // 控制速度，可以根据需要调整
    }
}
```

其中 hWind 为窗口句柄，应为全局变量，并且在 InitInstance 函数中赋值；

```
HWND hWind;
HDC  hWinDC;
int ImageWidth, ImageHe
```

```
BOOL InitInstance(HINSTANCE hInstance, int nCmdShow)
{
    HWND hWnd;

    hInst = hInstance; // Store instance handle in our global variable

    hWnd = CreateWindow(szWindowClass, szTitle, WS_OVERLAPPEDWINDOW,
        CW_USEDEFAULT, 0, CW_USEDEFAULT, 0, NULL, NULL, hInstance, NULL);

    hWind = hWnd;

    if (!hWnd)
    {
        return FALSE;
    }

    ShowWindow(hWnd, nCmdShow);
    UpdateWindow(hWnd);

    return TRUE;
}
```

ShowImageRect 函数定义如下，与 ShowImage 函数类似，用于部分显示图像。

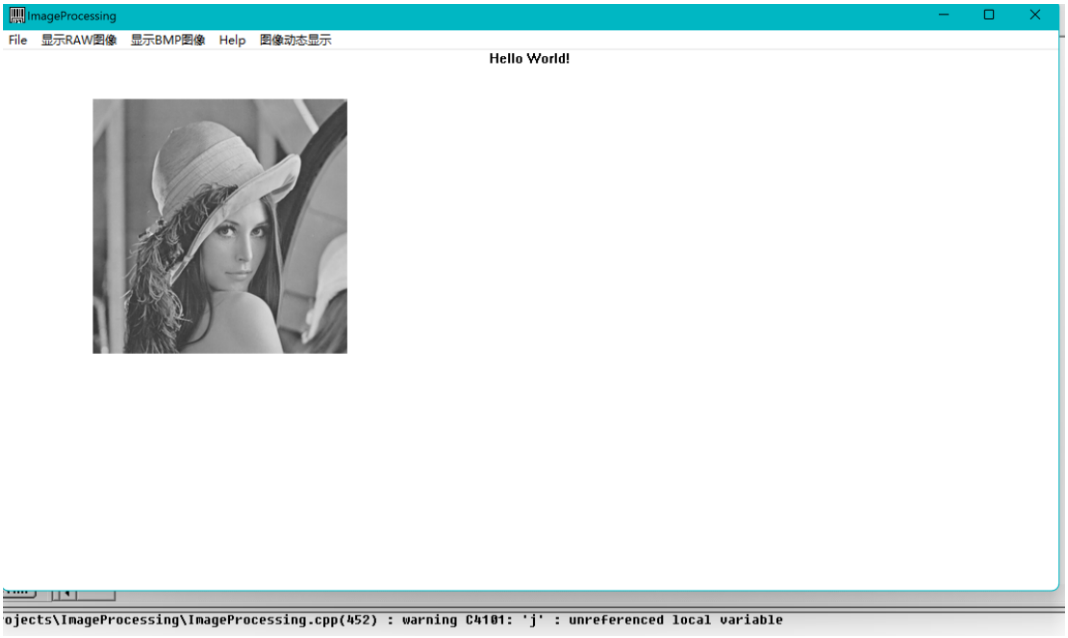
```
void ShowImageRect(char* Image, int wImage, int hImage, int xPos, int yPos, RECT ShowRect)
{
    if (!hWinDC) return;

    int rectWidth = ShowRect.right - ShowRect.left;
    int rectHeight = ShowRect.bottom - ShowRect.top;

    for (int i = ShowRect.top; i < ShowRect.bottom; i++) {
        for (int j = ShowRect.left; j < ShowRect.right; j++) {
            BYTE pixel = Image[i * wImage + j];
            SetPixel(hWinDC, j + xPos, i + yPos, RGB(pixel, pixel, pixel));
        }
    }
}
```

6. 以下为由上向下百叶窗显示函数，同时写上了注释，与由中间向左右(开门)显示函数类似，这里不做解释。

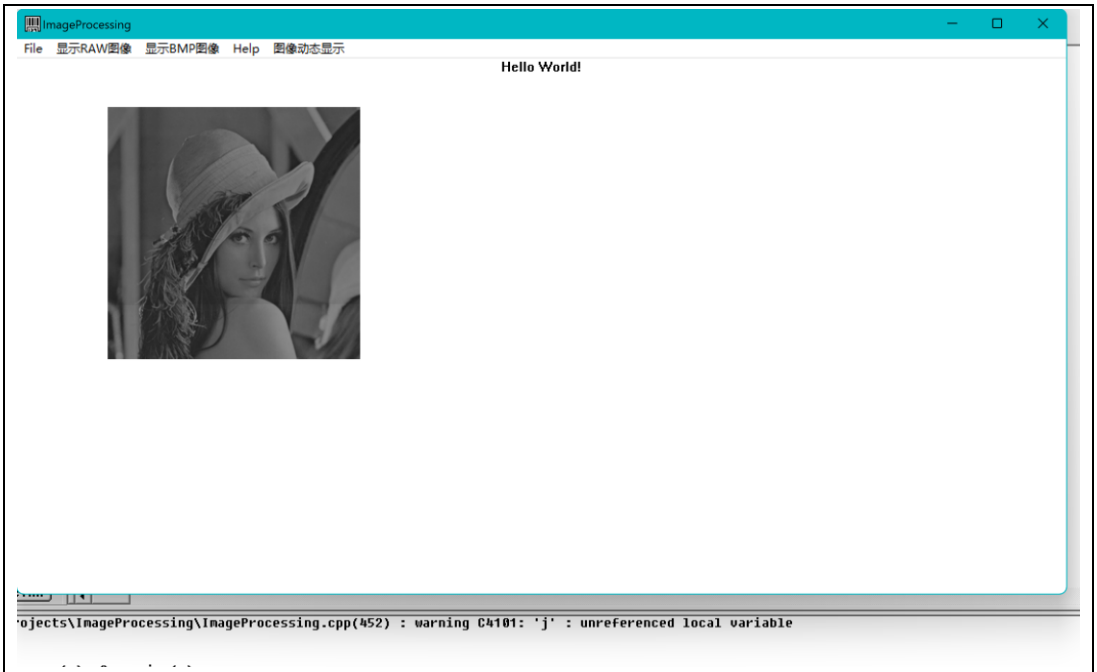
```
void ShowTopBottomShutter(char *Image, int wImage, int hImage, int xPos, int yPos)
{
    int i, j;
    RECT ShowRect;

    // 清空窗口，更新显示
    InvalidateRgn(hWind, NULL, TRUE);
    UpdateWindow(hWind);

    // 设置显示区域宽度
    ShowRect.left = 0;
    ShowRect.right = wImage; // 显示区域宽度为图像的宽度

    // 从上到下逐渐显示图像
    for (i = 0; i < hImage; i++) {
        ShowRect.top = i;           // 从图像的第 i 行开始
        ShowRect.bottom = i + 1;    // 显示一行

        // 调用 ShowImageRect 函数显示当前行
        ShowImageRect(Image, wImage, hImage, xPos, yPos, ShowRect);

        // 控制显示的速度，可以根据需要调整时间
        Sleep(1);
    }
}
```

7. 实验结果

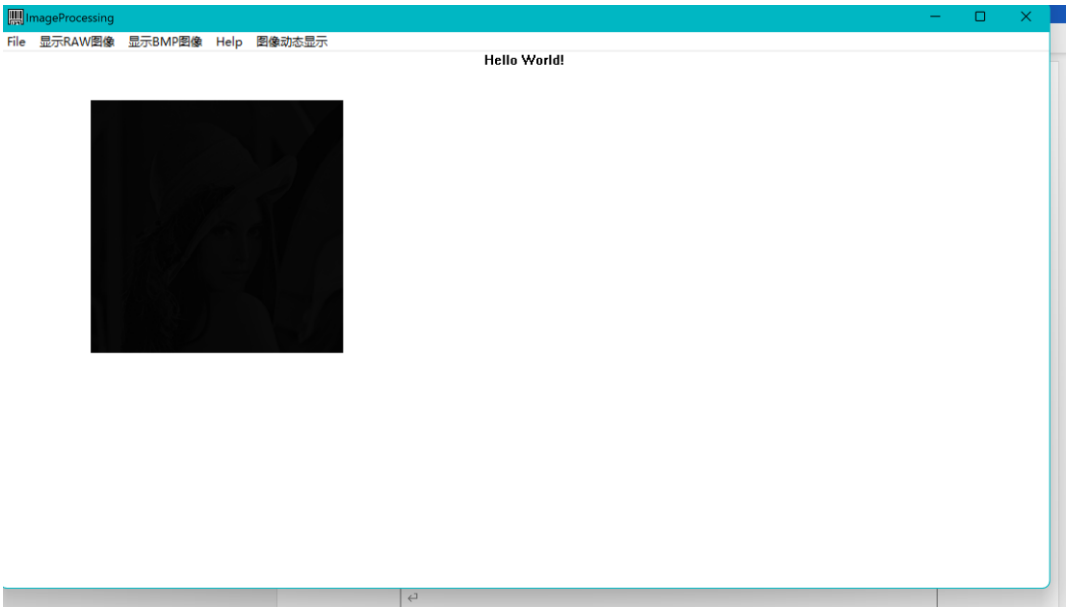
①打开图片，可以发现图片正常显示；



②点击淡出后，可以发现图片渐渐变黑；



③最后变成了黑色。



④点击向左右开门如图：

⑤点击由上向下百叶窗如图：

8. 为.bmp 类型文件制作特效动画的代码；首先是线性变黑函数；使用 3 层循环，第一层控制图像淡出的渐变步数，每次循环，图像的 RGB 值都会根据 m 逐渐衰减，后两层循环获取每行每列的像素信息，同时需要让行指针从末尾开始，防止图片倒转；获取像素后计算出当前图像的颜色衰减比例，最后将经过衰减后的颜色值应用到图像上；

```
void ShowGrayToLow1(char* Image, int wImage, int hImage, int xPos, int yPos)
{
    int i, j, m;
    int rowSize = (wImage * 3 + 3) & ~3; // 计算含填充的行字节数

    for (m = 1; m < GRAYCHANGENUM; m++)
    {
        for (i = 0; i < hImage; i++)
        {
            int invertedY = hImage - 1 - i; // 行序翻转
            char* row = Image + invertedY * rowSize; // 行指针

            for (j = 0; j < wImage; j++)
            {
                int offset = j * 3; // 每像素3字节
                BYTE b = row[offset];
                BYTE g = row[offset + 1];
                BYTE r = row[offset + 2];

                // 各通道同步衰减
                float ratio = (GRAYCHANGENUM - m)/(float)GRAYCHANGENUM;
                SetPixel(hWinDC, j + xPos, i + yPos,
                        RGB(r * ratio, g * ratio, b * ratio));
            }
        }
        Sleep(1);
    }
}
```

9. 由于.bmp 的由中间向左右(开门)显示函数与由上向下百叶窗显示函数和.raw 文件类似，这里不再赘述，下面是.bmp 的 ShowImageRect 函数；用两层循环获取.bmp 图片的 RBG 值，同时进行倒置处理，这样子显示出来是正的。

```
void ShowImageRect1(char* Image, int wImage, int hImage, int xPos, int yPos, RECT ShowRect)
{
    if (!hWinDC) return;

    for (int i = ShowRect.top; i < ShowRect.bottom; i++) {
        for (int j = ShowRect.left; j < ShowRect.right; j++) {
            // 处理BMP倒置行序
            int invertedY = hImage - 1 - i;
            int offset = (invertedY * wImage + j) * 3;

            BYTE b = Image[offset];     // Blue
            BYTE g = Image[offset + 1]; // Green
            BYTE r = Image[offset + 2]; // Red

            SetPixel(hWinDC, j + xPos, i + yPos, RGB(r, g, b));
        }
    }
}
```

10. 实验结果

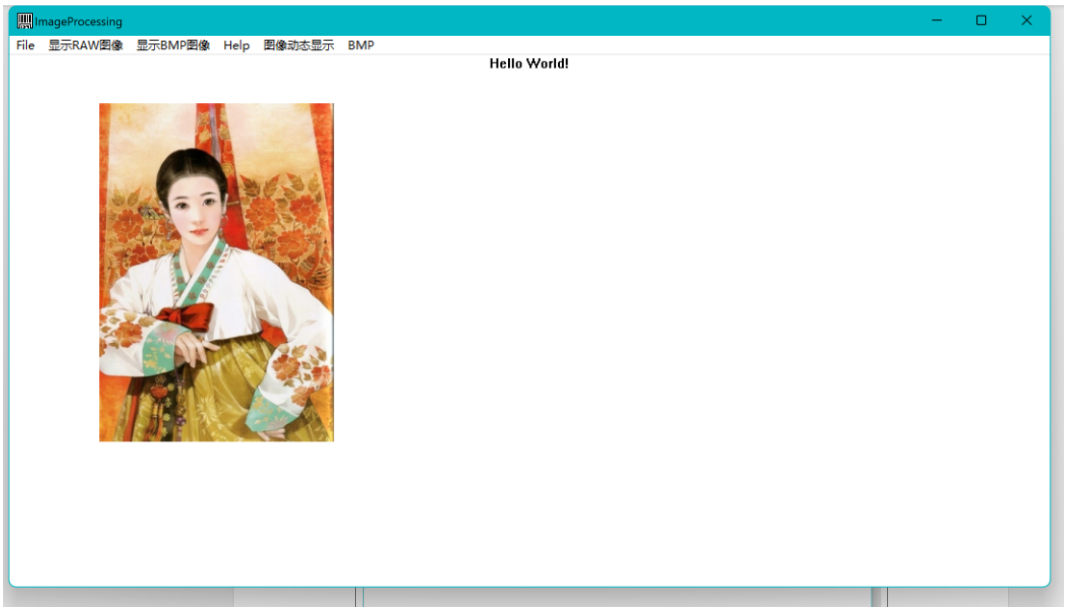
可以发现图片渐渐变黑

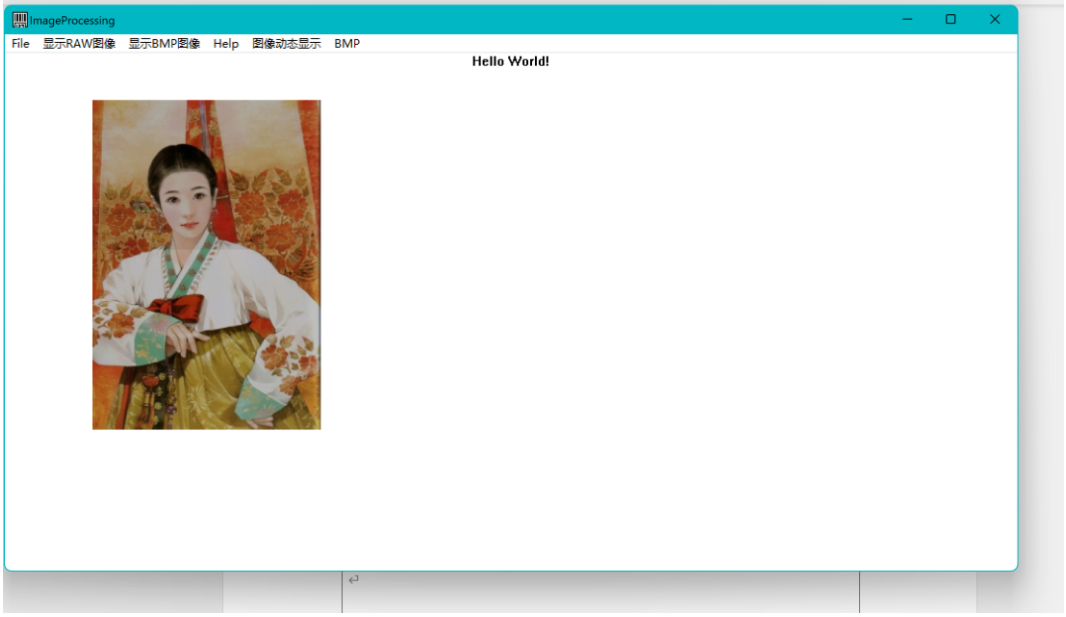
最后变成黑色

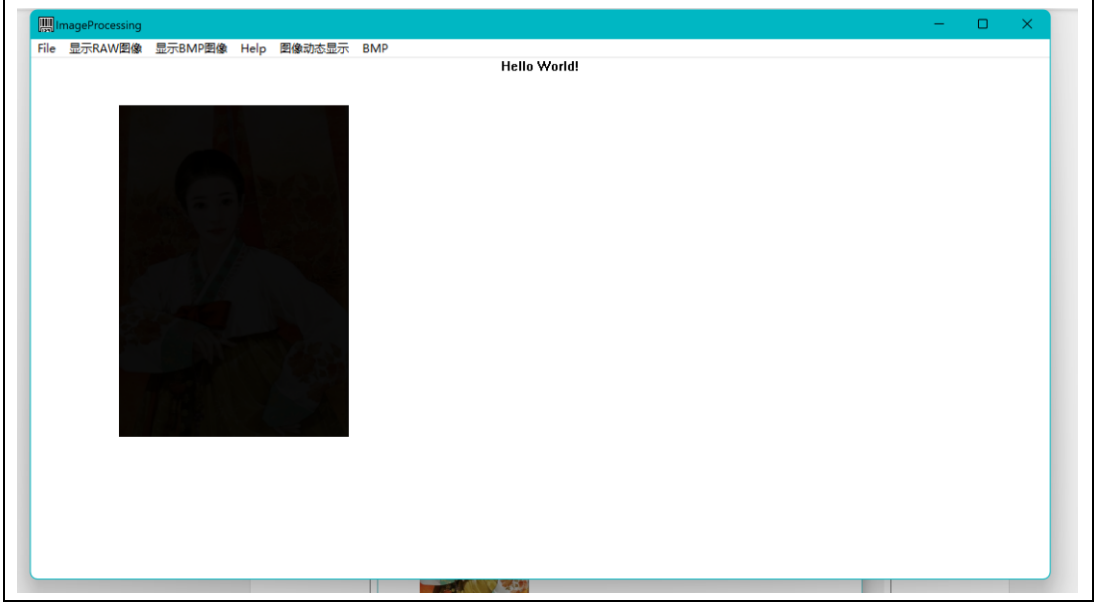
由中间向左右(开门)显示

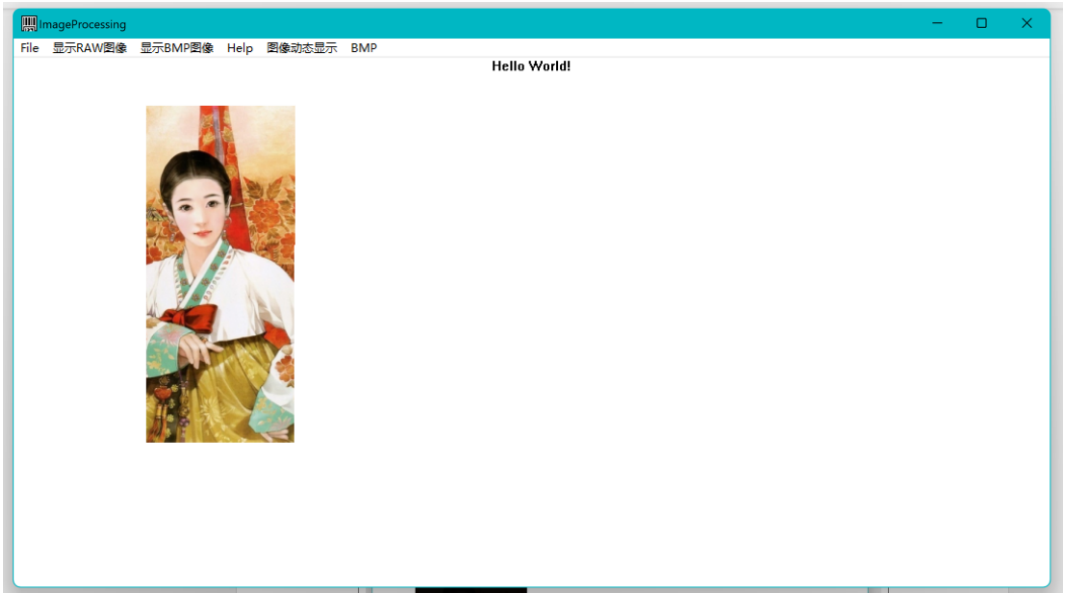
由上向下百叶窗显示

下面为 girl.bmp 的演示，这里不再赘述


ImageProcessing
File 显示RAW图像 显示BMP图像 Help 图像动态显示 BMP
Hello World!


ImageProcessing
File 显示RAW图像 显示BMP图像 Help 图像动态显示 BMP
Hello World!


ImageProcessing
File 显示RAW图像 显示BMP图像 Help 图像动态显示 BMP
Hello World!

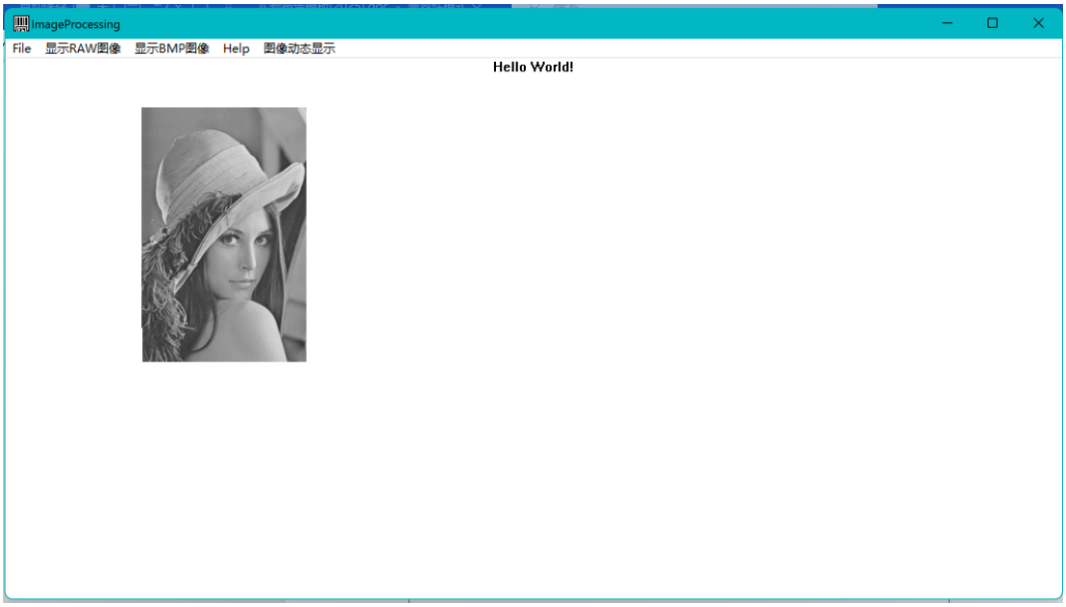

ImageProcessing
File 显示RAW图像 显示BMP图像 Help 图像动态显示 BMP
Hello World!


ImageProcessing
File 显示RAW图像 显示BMP图像 Help 图像动态显示 BMP
Hello World!

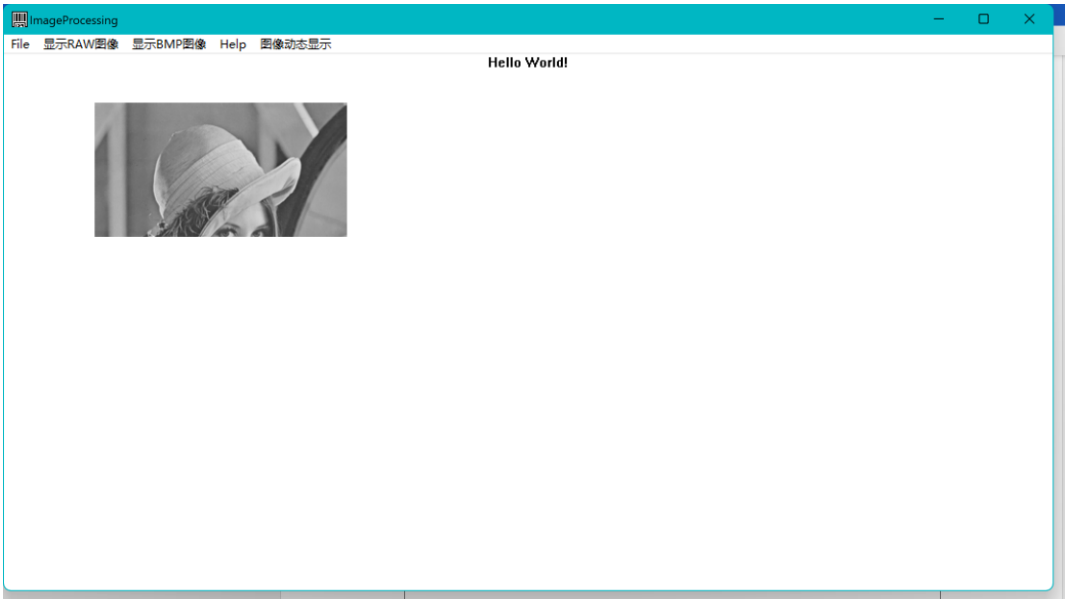
五、实验现象及数据处理：

lena256.raw

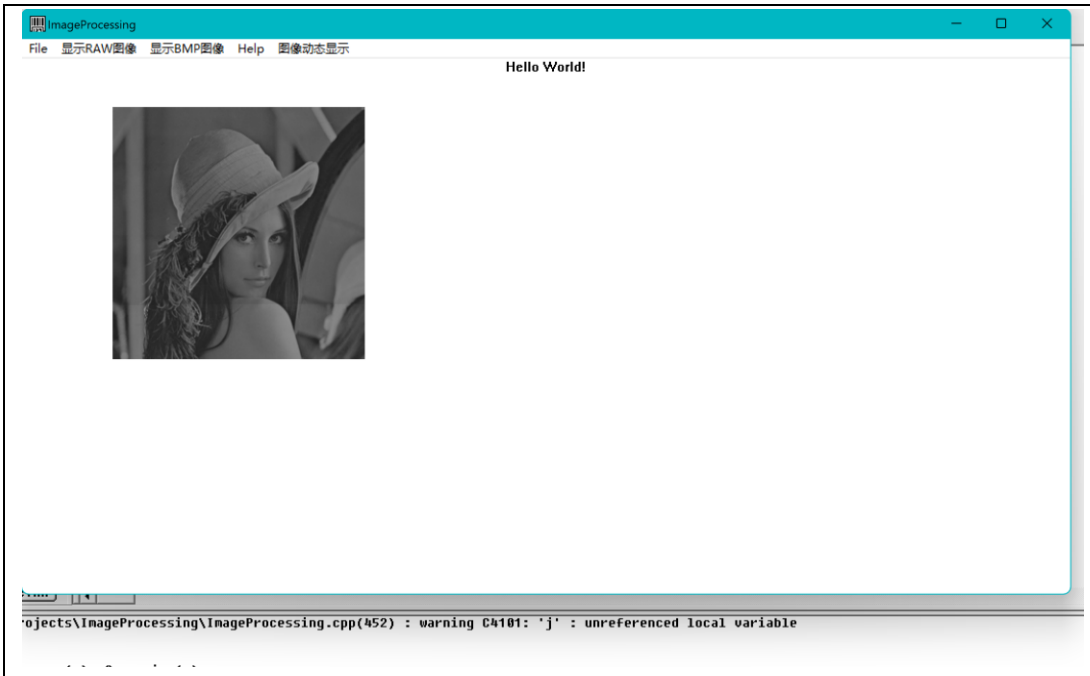
由中间向左右(开门)显示



由上向下百叶窗显示



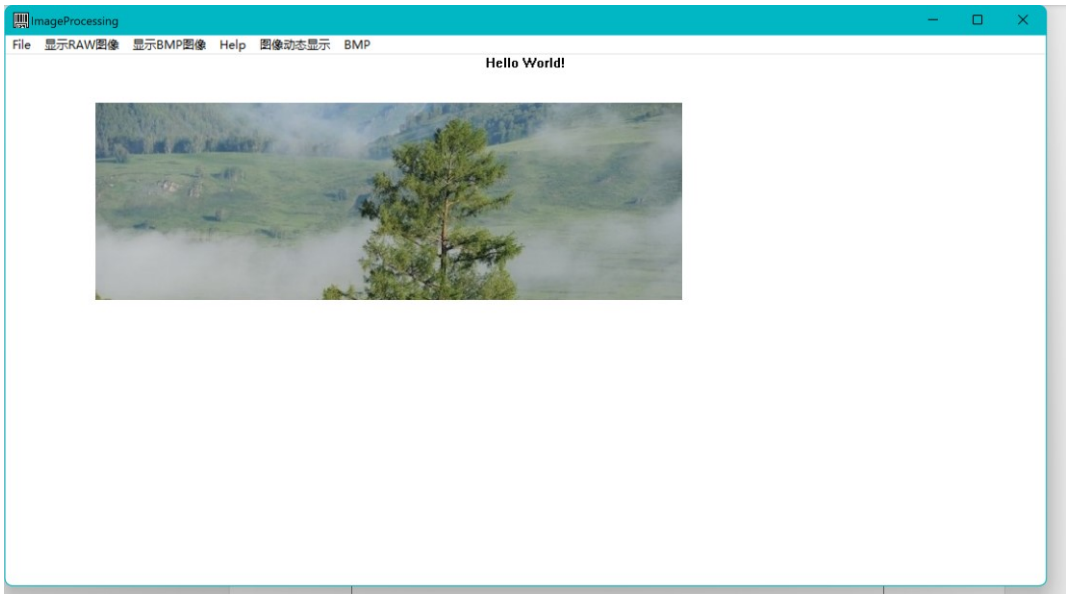
淡入显示



tree.bmp

由中间向左右(开门)显示

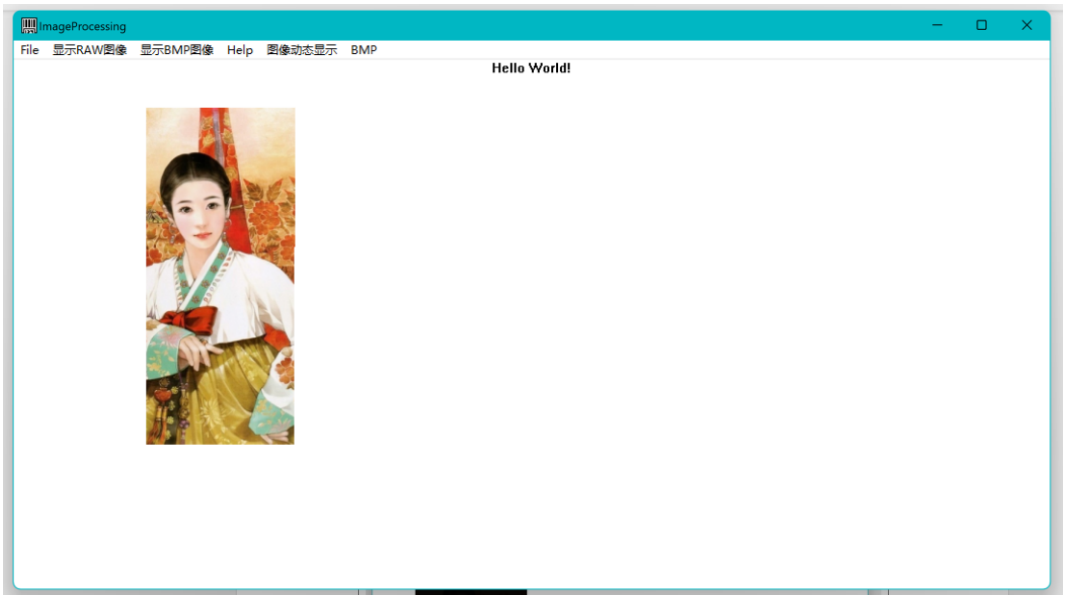
由上向下百叶窗显示



淡入显示

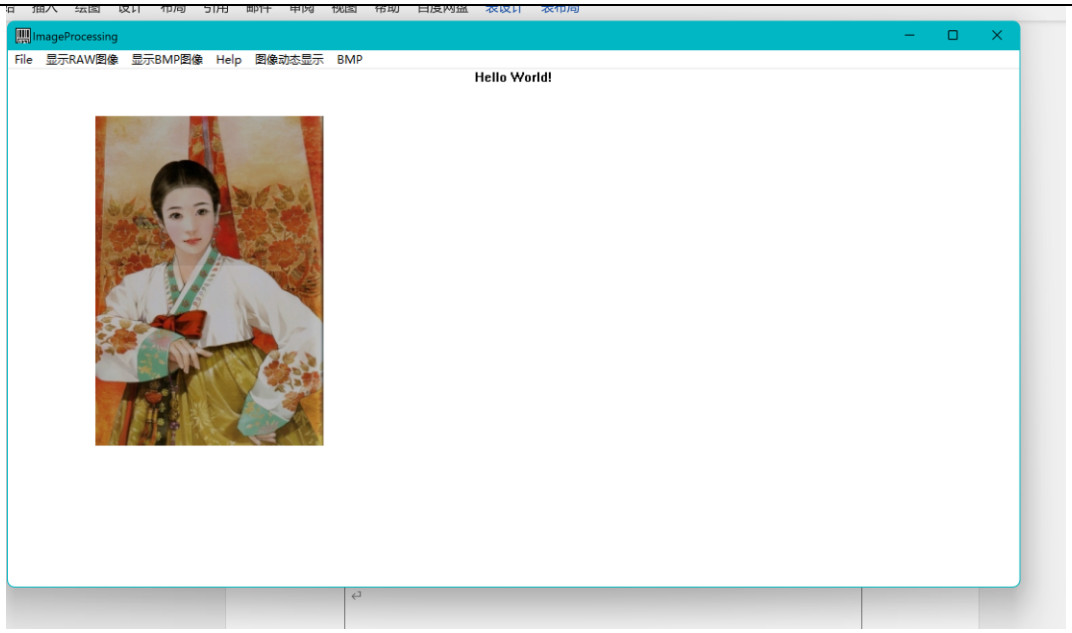
girl.bmp

由中间向左右(开门)显示



由上向下百叶窗显示

淡入显示



六、实验结论：

实验成功完成！实验中提供的三张图片都通过 visual6.0C++成功完成了动态效果。

实验感想：

通过本次实验，我深入理解了图像特效的实现原理与编程逻辑。在 Visual C++ 6.0 环境下，针对.raw 和.bmp 格式的图像，完成了开门、百叶窗、淡入三种动态效果。开门效果通过逐列分割像素并双向填充实现，让我体会到图像分块处理的高效性；百叶窗特效采用分段刷新机制，强化了对位图数据逐行操作的理解；淡入效果则通过 Alpha 通道渐变实现，加深了对色彩空间混合计算的掌握。

思考题：
指导教师批阅意见： **成绩评定：** **指导教师签字：** **年　　月　　日**
备注：

注：1、报告内的项目或内容设置，可根据实际情况加以调整和补充。

2、教师批改学生实验报告时间应在学生提交实验报告时间后 10 日内。