공간 영역 필터링 - Dialog Box 기반 구현

http://deios.kr

이번 시간에는 Dialog Box기반으로 OpenCV를 이용하는 방법에 대해서 알아보겠습니다.

공간 필터링(Spatial Filtering)이란 영상에 있는 공간 주파수 대역을 제거하거나 강조하는 필터 처리입니다. 사용되는 필터의 계수에 따라 특정 주파수를 제거하거나 강조하게 되며, 필터 마스크(Filter Mask)또는 회선 마스크의 가중치 선택이 공간 필터의 행동을 결정하게 됩니다. 영상처리에 사용되는 회선 마스크는 홀수 차원의 정방형 마스크가 사용되게 됩니다.

필터는 입력되는 신호의 일부 성분을 제거하거나 일부 특성을 변경하려고 설계된 하나의 시스템이며 두 종류로 구분됩니다. 첫 번째는 필터의 길이가 한정된 유한임펄스응답(FIR ; Finite Impulse Response)필터이고, 두 번째는 필터의 길이가 무한정한 무한임펄스응답(IIR ; Infinite Impulse Respones)필터입니다. FIR필터는 필터를 설계하기가 쉬우며 따라서 신호도 쉽게 처리할 수 있는 장점을 가지고 있습니다. IIR필터는 필터를 설계하기가 어려우나 필터의 특성은 더 우수합니다. 영상처리에서는 FIR 필터를 더 많이 사용합니다.

컨벌루션(회선;Convolution)이란 선형 시불변 시스템에 입력되는 신호가 어떤 신호를 출력하는지 결정하는 식을 의미합니다.

영상처리는 가로방향과 세로 방향의 처리가 필요한 2차원 처리이며, 필터링을 이용한 영상처리는 2차원의 컨벌루션을 수행하는 것입니다.

OpenCV에서 마스크 모양의 커널(Kernel)을 회선하는 함수는 cvFilter2D입니다. 이 함수를 레퍼런스에서 찾아보면 다음과 같습니다.

cvFilter2D

Convolves an image with the kernel.

```
void cvFilter2D(
            const CvArr* src,
            CvArr* dst,
            const CvMat* kernel,
            CvPoint anchor=cvPoint(-1,-1));
```

src The source image

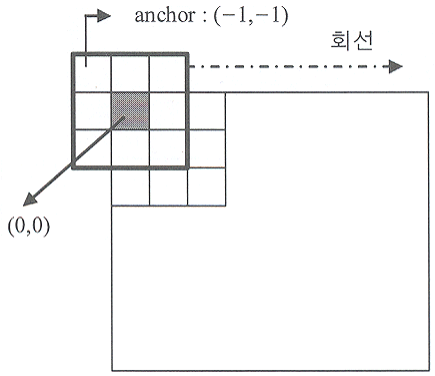
dst The destination image

kernel Convolution kernel, a single-channel floating point matrix. If you want to apply different kernels to different channels, split the image into separate color planes using cvSplit and process them individually

anchor The anchor of the kernel that indicates the relative position of a filtered point within the kernel. The anchor shoud lie within the kernel. The special default value (-1,-1) means that it is at the kernel center

The function applies an arbitrary linear filter to the image. In-place operation is supported. When the aperture is partially outside the image, the function interpolates outlier pixel values from the nearest pixels that are inside the image.

cvFilter2D함수는 src영상을 받아 kernel을 가지고 필터링 한 후 dst로 출력해 줍니다. 여기서 anchor은 어느 지점부터 필터링을 시작할 지를 결정해 주는 값이며 기본으로 (-1, -1)위치가 지정됩니다. 이는 공간영역 필터링에서 주변에 테두리가 생기지 않게 됩니다.



이번에 구현할 필터 마스크(Kernel)은 다음과 같습니다.

1. soften filter - 부드러운 효과

$$\frac{1}{18}\times\begin{matrix}1&1&1\\1&10&1\\1&1&1\end{matrix}$$

2. Enhance focus filter - 물체를 둘러싼 것을 초점을 맞춘 효과

$$\frac{1}{3}\times\begin{matrix}-1&0&-1\\0&7&0\\-1&0&-1\end{matrix}$$

3. Enhance detail filter - 물체를 둘러싼 것을 상세히 드러냄

$$\frac{1}{5}\times\begin{matrix}0&-1&0\\-1&9&-1\\0&-1&0\end{matrix}$$

4. Blur light filter - 블러링 단점인 잡음 미제거 보안 효과

$$\frac{1}{14}\times\begin{matrix}1&1&1\\2&2&2\\1&1&1\end{matrix}$$

5. Gaussian filter - 블러링 단점인 잡음 미제거 보안 효과

$$\frac{1}{16}\times\begin{matrix}1&2&1\\2&4&2\\1&2&1\end{matrix}$$

먼저 DeiosImgUtil Class를 작성하도록 하겠습니다. 이 DeiosImgUtil Class에는 이미지의 너비가 4의 배수가 되도록 조정하는 멤버 함수와 이미지를 HBITMAP으로 변환하는 멤버 함수가 들어갑니다.

Class를 만든 후 Resize4Bitmap함수와 IplImage2Bitmap함수를 추가합니다. 원형은 다음과 같습니다.

```
IplImage *Resize4Bitmap(int resize_height, int resize_width, IplImage * image)
HBITMAP IplImage2Bitmap(IplImage * image)
```

다음과 같이 작성합니다.

```
// 정성환 외 1명/컴퓨터 비전 실무 프로그래밍/홍릉과학출판사/2007/324p~325p
IplImage *CDeiosImgUtil::Resize4Bitmap(int resize_height, int resize_width, IplImage * image){
	int height = image->height;
	int width = image->width - (image->width%4);

	int sHeight = height;
	int sWidth = width;

	if( (height > resize_height) && (width > resize_width) ){
		sHeight = resize_width;
		width = (int) (width*(resize_height/(double)height));
		sWidth = width - (width%4);
	}else if( height > resize_height) sHeight = resize_height;
	else if( width > resize_width) sWidth = resize_width - (resize_width%4);

	IplImage *resize_image = cvCreateImage(cvSize(sWidth, sHeight), IPL_DEPTH_8U,
image->nChannels);

	cvResize(image, resize_image, CV_INTER_CUBIC);

	return resize_image;
}
```

```
// 정성환 외 1명/컴퓨터 비전 실무 프로그래밍/홍릉과학출판사/2007/325p~326p
HBITMAP CDeiosImgUtil::IplImage2Bitmap(IplImage *image ){
	HDC hDC=::CreateCompatibleDC(0);
	BYTE tmp[sizeof(BITMAPINFO)+255*4 ];
	BITMAPINFO *bmi = (BITMAPINFO *) tmp;
	HBITMAP hBmp;

	int height = image->height;
	int width = image->width;
	int widthStep = width +(width % 4);
	int nChannels = image->nChannels;
	int bpp = 8*nChannels;
	int bmpDataSize = height*widthStep*nChannels;

	memset( bmi, 0, sizeof(BITMAPINFO) );
	bmi->bmiHeader.biSize = sizeof(BITMAPINFOHEADER);
	bmi->bmiHeader.biHeight = height; // 영상의 높이
	bmi->bmiHeader.biWidth = width; //영상의 너비
	bmi->bmiHeader.biPlanes = 1; // 비트 플레인 수 (항상 1임)
	bmi->bmiHeader.biBitCount = bpp; // 한 화소당 비트 개수
```

```
	bmi->bmiHeader.biCompression = BI_RGB; // BI_RGB: 압축하지 않음.
	bmi->bmiHeader.biSizeImage = bmpDataSize; // 영상의 크기
	bmi->bmiHeader.biClrUsed = 0;

	if( bpp == 8 ){
		for( int i=0; i<256; i++ ){
			bmi->bmiColors[i].rgbBlue = i;
			bmi->bmiColors[i].rgbGreen = i;
			bmi->bmiColors[i].rgbRed = i;
		}
	}

	hBmp = ::CreateDIBSection( hDC, bmi, DIB_RGB_COLORS, NULL, 0, 0 );
	::DeleteDC(hDC);

	::SetBitmapBits( hBmp, image->imageSize, image->imageData );

	return hBmp;
}
```

이번에는 해당 기능을 구현할 Filter Class를 만듭니다.
Filter Class에는 오버로딩 된 ApplyFilter2D멤버 함수가 작성됩니다.
원형은 다음과 같습니다.

```
IplImage * ApplyFilter2D(IplImage * src, float *kernel, int kHeight, int kWidth);
IplImage * ApplyFilter2D(IplImage * src, float **kernel, int kHeight, int kWidth);
IplImage * ApplyFilter2D(IplImage * src, CvMat kernel);
```

다음과 같이 코드를 작성합니다.

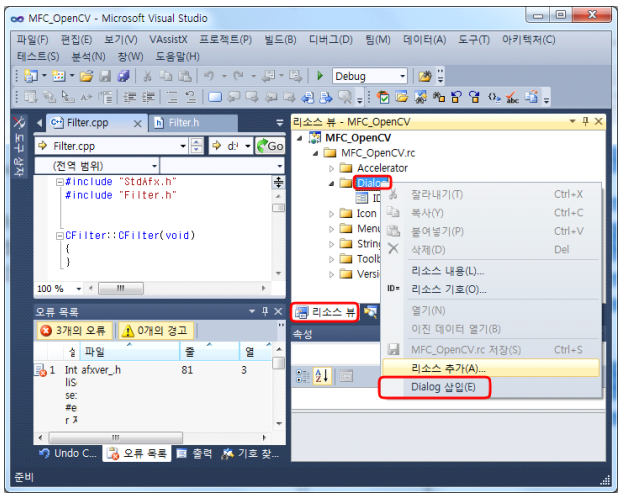
```
// 정성환 외 1명/컴퓨터 비전 실무 프로그래밍/홍릉과학출판사/2007/295p
IplImage * CFilter::ApplyFilter2D(IplImage * src, float *kernel, int kHeight, int kWidth){
	CvMat mat_kernel = cvMat( kWidth, kHeight, CV_32FC1, kernel );

	IplImage *dst = ApplyFilter2D( src, mat_kernel );

	cvReleaseData( &mat_kernel );

	return dst;
}
```

```
// 정성환 외 1명/컴퓨터 비전 실무 프로그래밍/홍릉과학출판사/2007/295p~296p
IplImage * CFilter::ApplyFilter2D(IplImage * src, float **kernel, int kHeight, int kWidth){
	int cnt = -1;

	float *single_kernel = (float *)calloc( kHeight*kWidth, sizeof(float) );
```

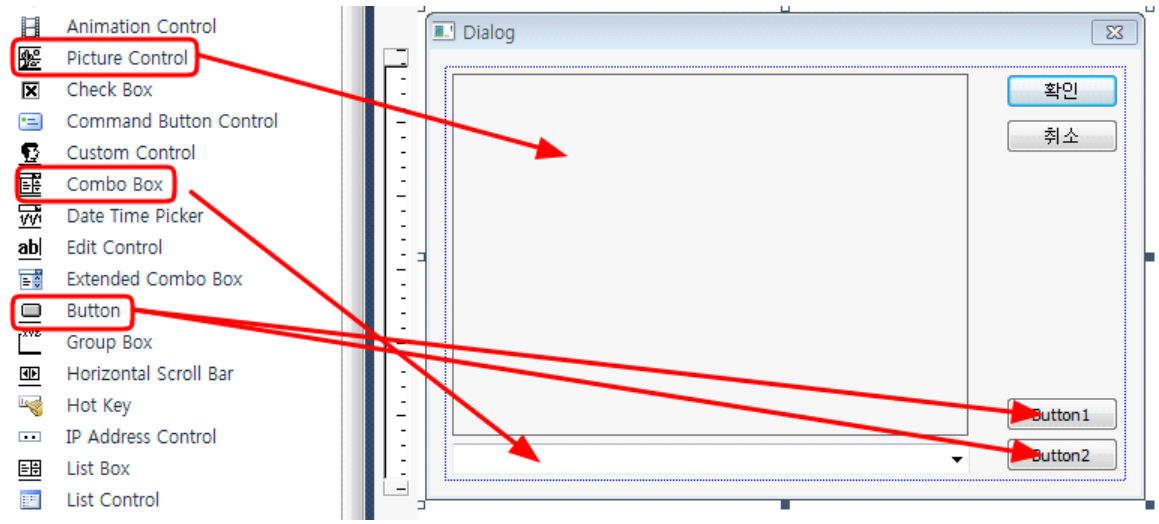
```
        for(int i = 0; i < kHeight ; i++)
                for(int j = 0; j<kWidth ; j++)
                        single_kernel[++cnt] = kernel[i][j];

        IplImage *dst = ApplyFilter2D( src, single_kernel, kHeight, kWidth );

        free(single_kernel);

        return dst;
}
```

```
// 정성환 외 1명/컴퓨터 비전 실무 프로그래밍/홍릉과학출판사/2007/296p
IplImage * CFilter::ApplyFilter2D(IplImage * src, CvMat kernel){
        IplImage *dst = cvCreateImage(cvGetSize(src), IPL_DEPTH_8U, src->nChannels);

        cvFilter2D(src, dst, &kernel, cvPoint(-1, -1));

        return dst;
}
```

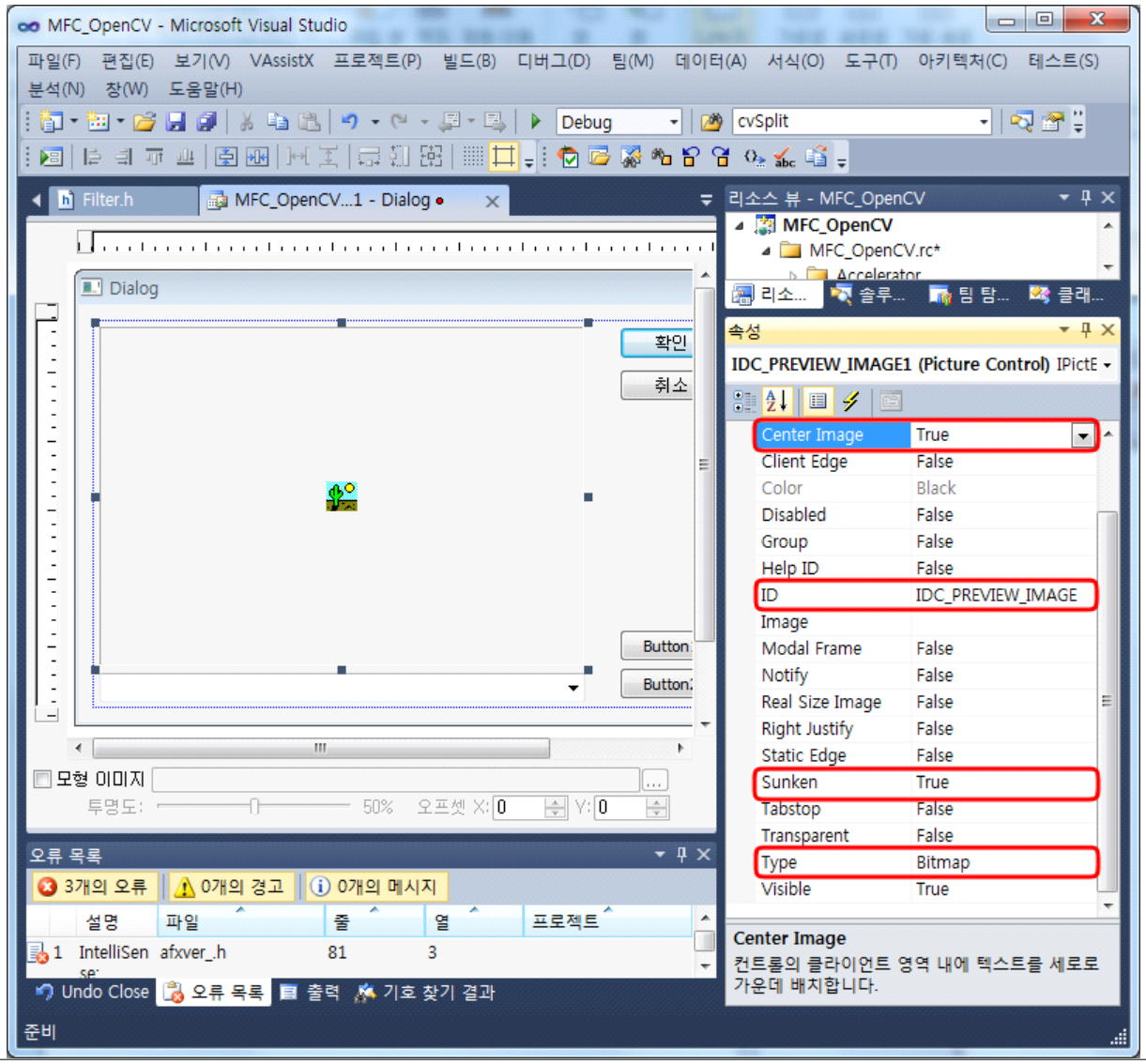
이제 UI를 설계해 보겠습니다. [리소스뷰] -> [Dialog] -> [Dialog 삽입]을 선택합니다.



다음과 같이 컨트롤 들을 배열합니다.



Picture Control의 속성을 다음과 같이 지정합니다.



ID : IDC_PREVIEW_IMAGE

Type : Bitmap
Sunken : True
Center Image : True

비슷한 방법으로 Combo Box의 속성을 다음과 같이 지정합니다.

ID : IDC_COMBO_FILTER_TYPE
Type : Drop List

이번에는 2개의 Button중 첫 번째 Button의 속성을 설정하겠습니다.

ID : IDC_PREVIEW_BUTTON
Caption : 미리보기

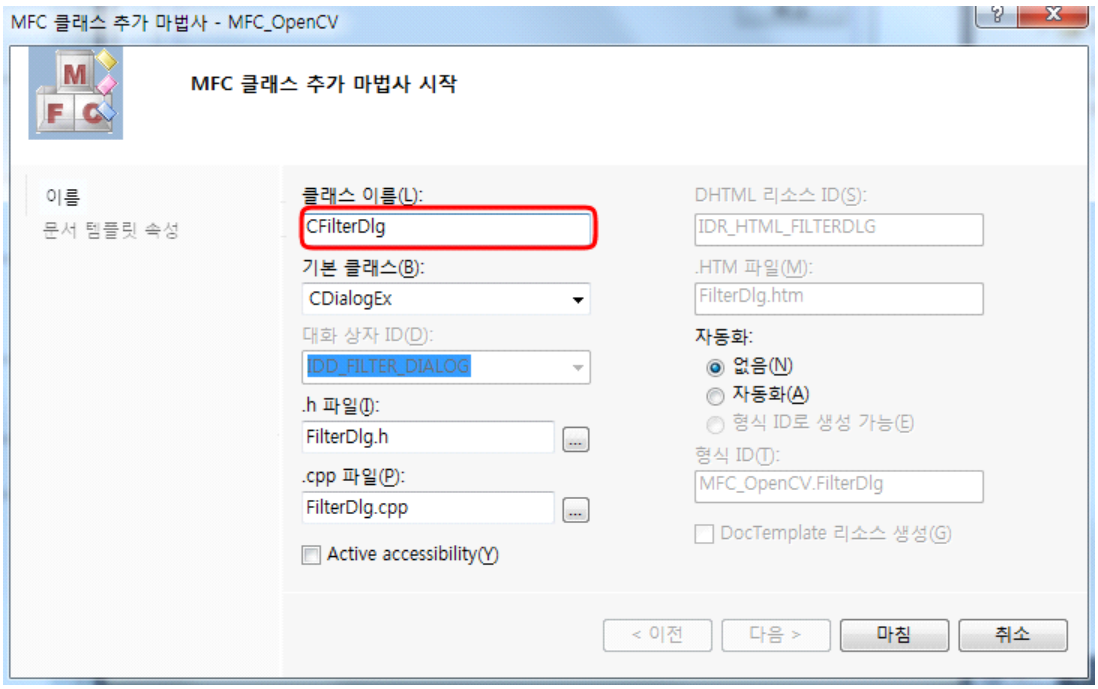
2번째 Button의 속성을 설정하겠습니다.

ID : IDC_INITIAL_BUTTON
Caption : 초기화

이번에는 Dialog의 ID를 지정해 줍니다.

ID : IDD_FILTER_DIALOG

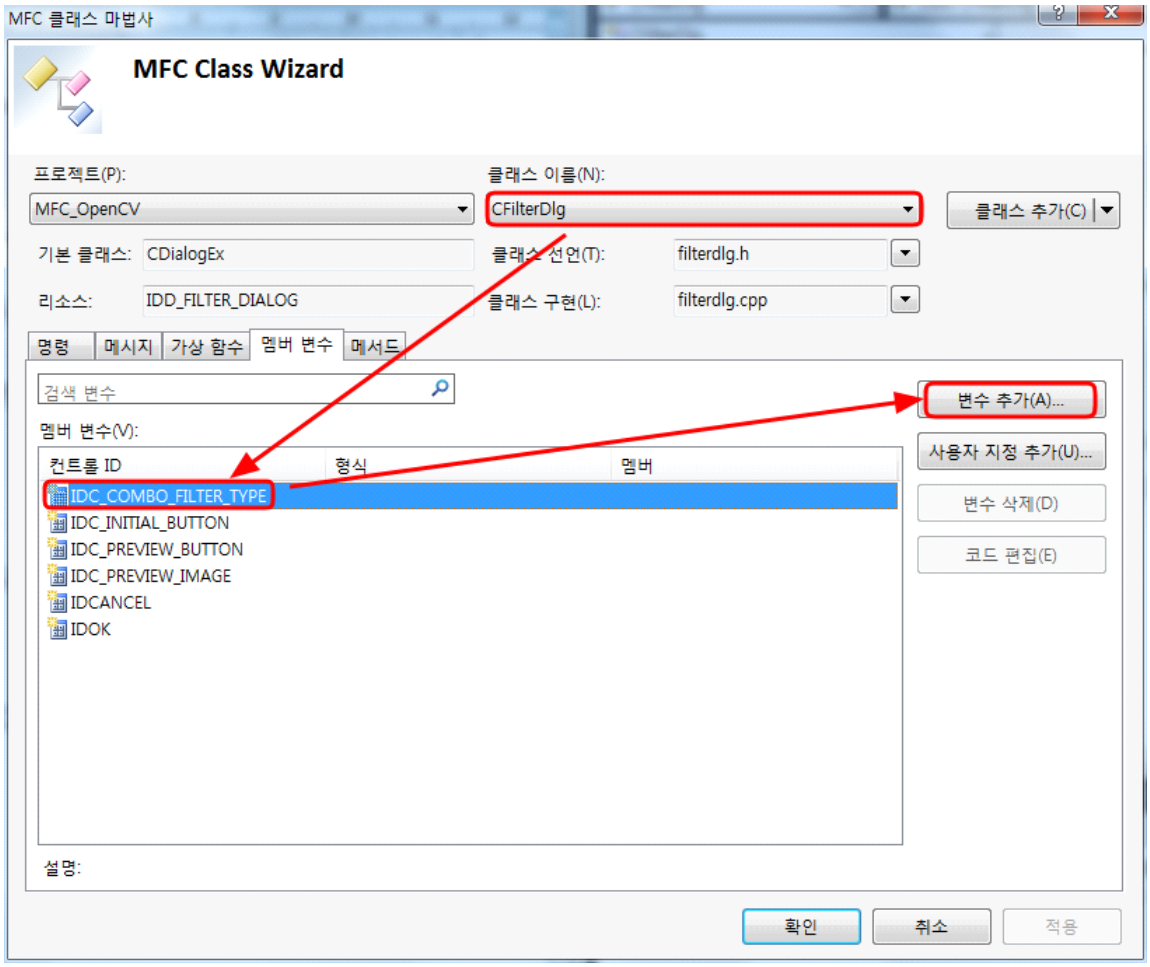
이제 만들어진 Dialog에 대응하는 Class를 추가합니다.
[Shift + Alt + C]버튼을 눌러 Class 추가 마법사를 실행합니다. 클래스 이름은 FilterDlg로 합니다.



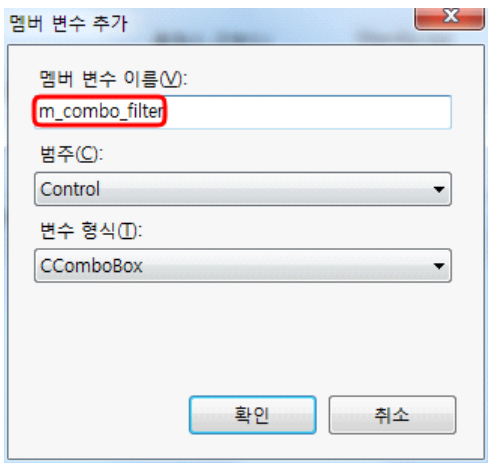
이번에는 Dialog의 각 Control에 대응하는 멤버 변수를 추가해 보겠습니다.

모든 Control의 멤버 변수를 추가할 필요는 없으며, 입력과 출력을 필요로 하는 Control의 멤버 변수만 추가하면 됩니다.
[Ctrl + Shift + X]키를 눌러 Class 마법사를 실행합니다.

CFilterDlg Class의 IDC_COMBO_FILTER_TYPE Control의 멤버 변수를 추가합니다.
클래스 이름과 멤버 변수를 확인 후, 변수 추가 버튼을 클릭합니다.



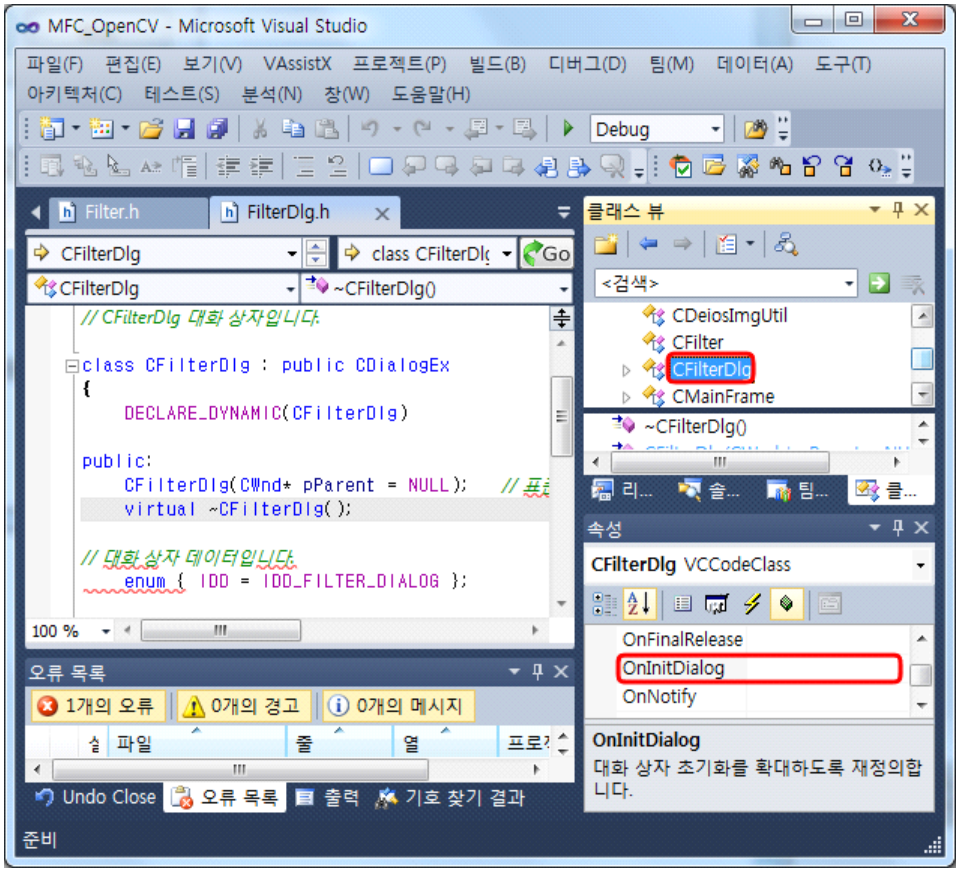
멤버 변수 이름을 m_combo_filter로 지정합니다.



동일한 방법으로 다음과 같은 멤버 변수를 추가합니다.

IDC_PREVIREW_IMAGE : m_preview_image

이번에는 Dialog 초기화 코드를 작성하겠습니다.



마찬가지 방법으로 OnDestroy멤버 함수도 추가해 줍니다.

이제 FilterDlg.h 파일을 열어 다음과 같은 멤버 변수를 추가합니다.

```
CvvImage d_org_cvvImage; // 원 영상
CvvImage d_cvvImage; // 필터링 결과 후의 영상

int picture_height; // Picture 컨트롤의 높이 기준
int picture_width; // Picture 컨트롤의 너비 기준(4의 배수)

float soften_mask[9]; // For Soften filtering
float ef_mask[9]; // For Enhance focus filtering
float ed_mask[9]; // For Enhance detail filtering
float bl_mask[9]; // For Blur light filtering
float bb_mask[9]; // For Blur blending filtering
```

이번에는 각종 헤더 파일을 include 하겠습니다.

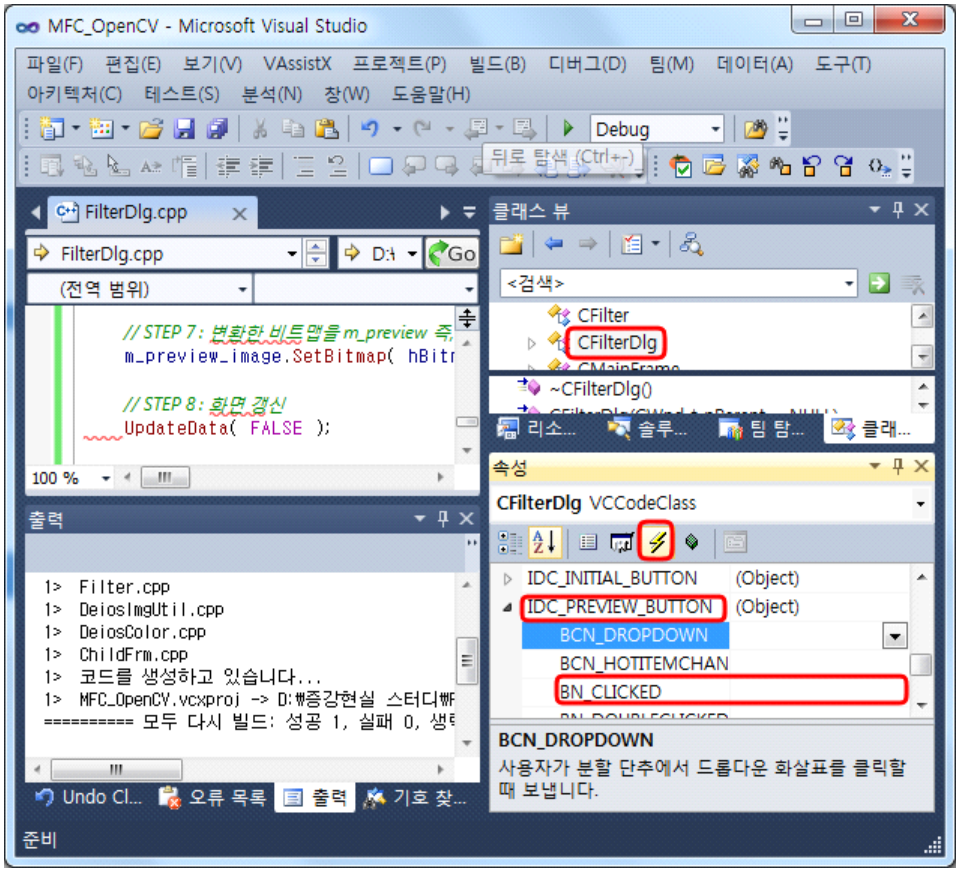
```
#include "MainFrm.h"
#include "ChildFrm.h"
#include "MFC_OpenCVDoc.h"
#include "Filter.h"
#include "DeiosImgUtil.h"
```

OnInitDialog멤버 함수에 다음과 같이 코딩합니다.

```
// 정성환 외 1명/컴퓨터 비전 실무 프로그래밍/홍릉과학출판사/2007/317p~320p
BOOL CFilterDlg::OnInitDialog(){
	CDialogEx::OnInitDialog();

	// TODO:  여기에 추가 초기화 작업을 추가합니다.

	// STEP 1 : Picture 컨트롤 크기 초기화
	picture_height = 228; // Picture 컨트롤의 높이 기준
	picture_width = 328; // Picture 컨트롤의 너비 기준(4의 배수)

	// STEP 2 : 필터 마스크 초기화

	// STEP 2-1 : Soften filter - 부드러운 효과를 냄.
	float _mask1[] = {	1.0f/18.0f,  1.0f/18.0f, 1.0f/18.0f,
						1.0f/18.0f, 10.0f/18.0f, 1.0f/18.0f,
						1.0f/18.0f,  1.0f/18.0f, 1.0f/18.0f };
	memcpy( soften_mask, _mask1, sizeof(float)*9 );

	// STEP 2-2 : Enhance focus filter - 물체를 둘러싼 것을 초점을 맞춘 효과를 냄.
	float _mask2[] = {	-1.0f/3.0f,  0.0f/3.0f, -1.0f/3.0f,
						0.0f/3.0f,  7.0f/3.0f,  0.0f/3.0f,
						-1.0f/3.0f,  0.0f/3.0f, -1.0f/3.0f };
	memcpy( ef_mask, _mask2, sizeof(float)*9 );

	// STEP 2-3 : Enhance detail filter - 물체를 둘러싼 것을 상세히 드러내는 효과를 냄.
	float _mask3[] = {	0.0f/5.0f, -1.0f/5.0f,  0.0f/5.0f,
						-1.0f/5.0f,  9.0f/5.0f, -1.0f/5.0f,
						0.0f/5.0f, -1.0f/5.0f,  0.0f/5.0f };
	memcpy( ed_mask, _mask3, sizeof(float)*9 );

	// STEP 2-4 : Blur light filter - 블러링 단점인 잡음 미제거 보완 효과를 냄.
	float _mask4[] = {	1.0f/14.0f, 1.0f/14.0f, 1.0f/14.0f,
						2.0f/14.0f, 2.0f/14.0f, 2.0f/14.0f,
						1.0f/14.0f, 1.0f/14.0f, 1.0f/14.0f };
	memcpy( bl_mask, _mask4, sizeof(float)*9 );

	// STEP 2-5 : Blur blending filter - 블러링 단점인 잡음 미제거 보완 효과를 냄.
	float _mask5[] = {	1.0f/16.0f, 2.0f/16.0f, 1.0f/16.0f,
						2.0f/16.0f, 4.0f/16.0f, 2.0f/16.0f,
						1.0f/16.0f, 2.0f/16.0f, 1.0f/16.0f };
	memcpy( bb_mask, _mask5, sizeof(float)*9 );

	// STEP 3 : 콤보 박스에 항목 추가
```

```
	CString strFilterList[] = { _T( "Soften filter" ), _T( "Enhance focus filter" ), _T( "Enhance detail filter" ), _T( "Blur light filter" ),_T( "Blur blending filter" )};

	int length = sizeof(strFilterList) / sizeof(CString);
	for(int i = 0; i< length ; i++)
		m_combo_filter.AddString(strFilterList[i]);

	CMainFrame *pFrame = (CMainFrame *)AfxGetMainWnd();
	CChildFrame *pChild = (CChildFrame *)pFrame->GetActiveFrame();
	CMFC_OpenCVDoc *pDoc = (CMFC_OpenCVDoc *)pChild->GetActiveDocument();

	// STEP 5 : 원 영상을 복사
	d_org_cvvImage.CopyOf( pDoc->m_CvvImage.GetImage(), pDoc->m_CvvImage.GetImage()->nChannels*8 );
	d_cvvImage.CopyOf( pDoc->m_CvvImage.GetImage(), pDoc->m_CvvImage.GetImage()->nChannels*8 );

	// STEP 6 : 원 영상을 비트맵으로 변환하기 위한 영상 크기 재조절
	CDeiosImgUtil cUtil;

	// (주의!) BITMAP의 특성 : 가로의 길이가 4의 배수가 아니면 찌그러짐
	IplImage *image = d_org_cvvImage.GetImage();
	IplImage *resize_image = cUtil.Resize4Bitmap( picture_height, picture_width, image );

	HBITMAP hBitmap = cUtil.IplImage2Bitmap( resize_image );
	cvReleaseImage( &resize_image );

	// STEP 7 : 변환한 비트맵을 m_preview 즉, picture 컨트롤에 뿌려준다.
	m_preview_image.SetBitmap( hBitmap );

	// STEP 8 : 화면 갱신
	UpdateData( FALSE );

	return TRUE;  // return TRUE unless you set the focus to a control
	// 예외: OCX 속성 페이지는 FALSE를 반환해야 합니다.
}
```

이번에는 미리보기 Button에 해당하는 멤버 함수를 작성해 보겠습니다.
FilterDlg Class의 이벤트 영역에서 IDC_PREVIEW_BUTTON을 선택 한 후 BN_CLICKED를 선택하여 메시지 처리기를 추가합니다.



다음과 같이 코딩합니다.

```
// 정성환 외 1명/컴퓨터 비전 실무 프로그래밍/홍릉과학출판사/2007/328p~330p
void CFilterDlg::OnBnClickedPreviewButton()
{
	// TODO: 여기에 컨트롤 알림 처리기 코드를 추가합니다.
	float kernel[9] = {0., };
	int kernel_height = 3;
	int kernel_width = 3;
	int kernel_length = kernel_height*kernel_width;

	// STEP 1 : 콤보 박스에서 선택한 항목을 가져오기
	CString str;
	int nIndex = m_combo_filter.GetCurSel();
	if( nIndex == CB_ERR ) return;
	m_combo_filter.GetLBText( nIndex, str );

	// STEP 2 : 콤보 박스에서 선택한 필터링 수행 하기 위한 커널 복사
	if( str == "Soften filter" )
		memcpy( kernel, soften_mask, sizeof(float)*kernel_length );
	else if( str == "Enhance focus filter" )
		memcpy( kernel, ef_mask, sizeof(float)*kernel_length );
```

```
	else if( str == "Enhance detail filter" )
		memcpy( kernel, ed_mask, sizeof(float)*kernel_length );
	else if( str == "Blur light filter" )
		memcpy( kernel, bl_mask, sizeof(float)*kernel_length );
	else if( str == "Blur blending filter" )
		memcpy( kernel, bb_mask, sizeof(float)*kernel_length );

	// STEP 3 : 원 영상을 비트맵으로 변환하기 위한 영상 크기 재조절
	CDeiosImgUtil cUtil;
	IplImage *image = d_cvvImage.GetImage();
	IplImage *resize_image = cUtil.Resize4Bitmap( picture_height, picture_width, image
);

	// STEP 4 : 공간 영역 필터링 수행
	CFilter cFilter;
	IplImage *filtered_image = cFilter.ApplyFilter2D( resize_image, kernel,
kernel_height, kernel_width );
	IplImage *filtered_image2 = cFilter.ApplyFilter2D( d_cvvImage.GetImage(), kernel,
kernel_height, kernel_width );

	d_cvvImage.CopyOf(filtered_image2);

	// STEP 5 : 수행한 결과를 비트맵으로 변환.
	HBITMAP hBitmap = cUtil.IplImage2Bitmap( filtered_image );
	//d_cvvImage.CopyOf( filtered_image, filtered_image->nChannels*8 );

	// STEP 6 : 변환한 비트맵을 m_preview 즉, picture 컨트롤에 뿌려줌.
	m_preview_image.SetBitmap( hBitmap );
	m_preview_image.Invalidate();

	// STEP 7 : 메모리 해제
	cvReleaseImage( &resize_image );
	cvReleaseImage( &filtered_image );
	cvReleaseImage( &filtered_image2 );

	// STEP 8 : 화면 갱신
	UpdateData( FALSE );
}
```

초기화 Button에 대응하는 멤버 함수를 추가하고 다음과 같이 코딩합니다.

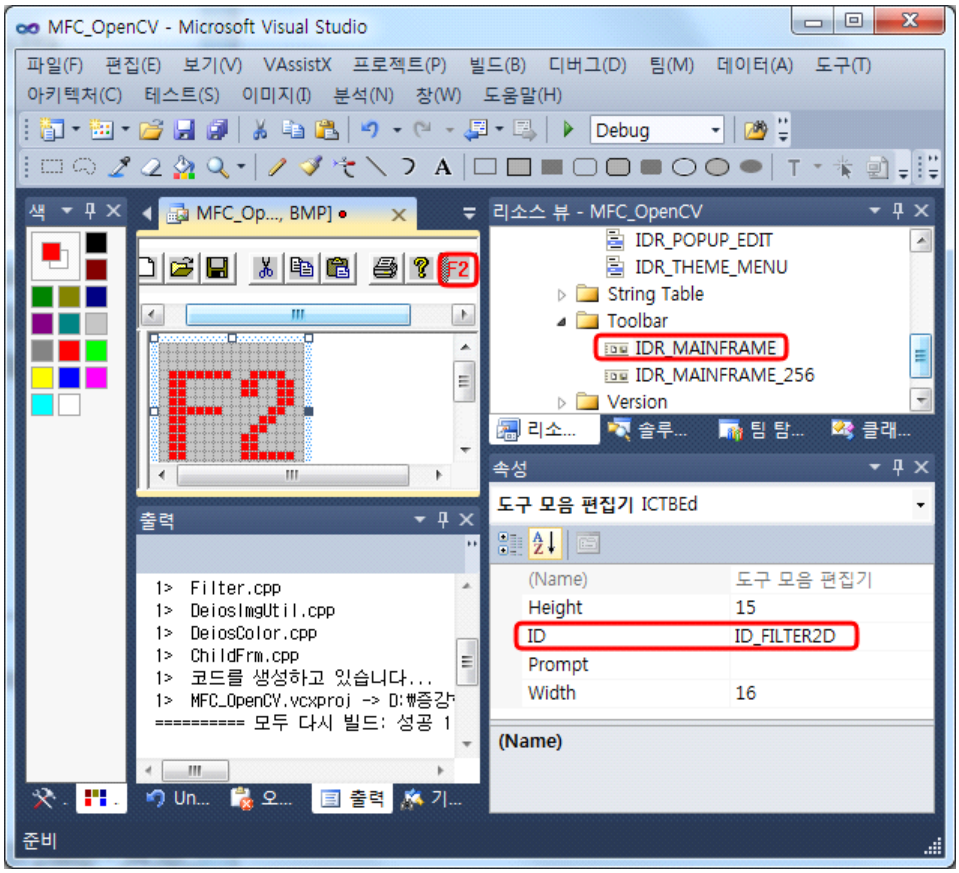
```
// 정성환 외 1명/컴퓨터 비전 실무 프로그래밍/홍릉과학출판사/2007/332p
void CFilterDlg::OnBnClickedInitialButton()
{
	// TODO: 여기에 컨트롤 알림 처리기 코드를 추가합니다.
```

```
	// STEP 1 : 원 영상을 가져옴.
	CDeiosImgUtil cUtil;
	IplImage *image = d_org_cvvImage.GetImage();
	IplImage *resize_image = cUtil.Resize4Bitmap( picture_height,
		picture_width,
		image );

	// STEP 2 : 비트맵으로 변환.
	HBITMAP hBitmap = cUtil.IplImage2Bitmap( resize_image );
	d_cvvImage.CopyOf( resize_image, resize_image->nChannels*8 );

	// STEP 3 : 변환한 비트맵을 m_preview 즉, picture 컨트롤에 뿌려줌.
	m_preview_image.SetBitmap( hBitmap );
	m_preview_image.Invalidate();

	// STEP 4 : 메모리 해제
	cvReleaseImage( &resize_image );

	// STEP 5 : 화면 갱신
	UpdateData( FALSE );
}
```

마지막으로 확인 Button에 해당하는 메시지 처리기를 구현하겠습니다.
다음과 같이 코딩합니다.

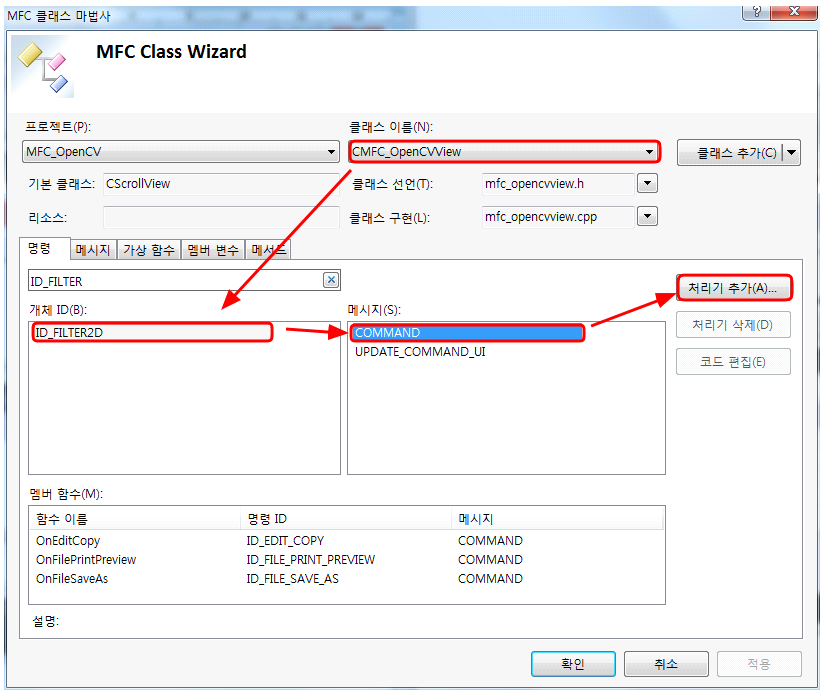
```
void CFilterDlg::OnBnClickedOk()
{
	// TODO: 여기에 컨트롤 알림 처리기 코드를 추가합니다.
	CMainFrame *pFrame = (CMainFrame *)AfxGetMainWnd();
	CChildFrame *pChild = (CChildFrame *)pFrame->GetActiveFrame();
	CMFC_OpenCVDoc *pDoc = (CMFC_OpenCVDoc *)pChild->GetActiveDocument();

	pDoc->m_CvvImage.CopyOf( d_cvvImage );
	UpdateData(TRUE);
	CDialogEx::OnOK();
}
```

이제 Dialog를 보여주는 작업만 남았습니다. 이전 시간에 Menu에 추가해 봤으니, 이번 시간에는 Toolbar에 추가해 보겠습니다.

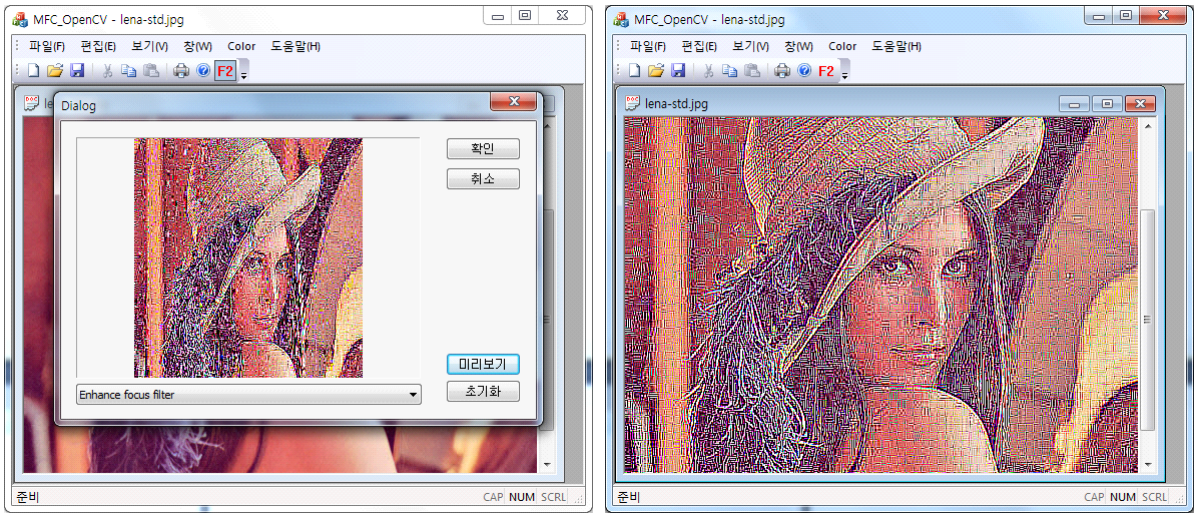


IDR_MAINFRAME_256에도 똑같이 설정합니다.

View Class에서 해당하는 멤버 함수를 추가합니다.



```
void CMFC_OpenCVView::OnFilter2d()
{
	// TODO: 여기에 명령 처리기 코드를 추가합니다.
	CFilterDlg dlg;
	dlg.DoModal();

	Invalidate(FALSE);
}
```



Reference

정성환 외 1명/컴퓨터 비전 실무 프로그래밍/홍릉과학출판사/2007/288p~343p,571p

신종홍 외 2명/디지털 영상처리 입문/한빛미디어/2010/430p~462p