15장 프로젝트 _웨이브렛 변환

- 웨이브렛 변환의 개념
- 웨이브렛 변환을 위한 MFC 설정하기
- 파일 입.출력 설정하기
- 웨이브렛 변환을 위한 대화상자 설정하기
- 순방향 웨이브렛 변환 구현하기
- 역방향 웨이브렛 변환 구현하기

한빛미디어(주)

학습목표

- ✓ 웨이브렛 변환의 개념을 소개한다.
- ✓ 순방향 웨이브렛 변환 프로그램을 실습한다.
- ✓ 역방향 웨이브렛 변환 프로그램을 실습함다.

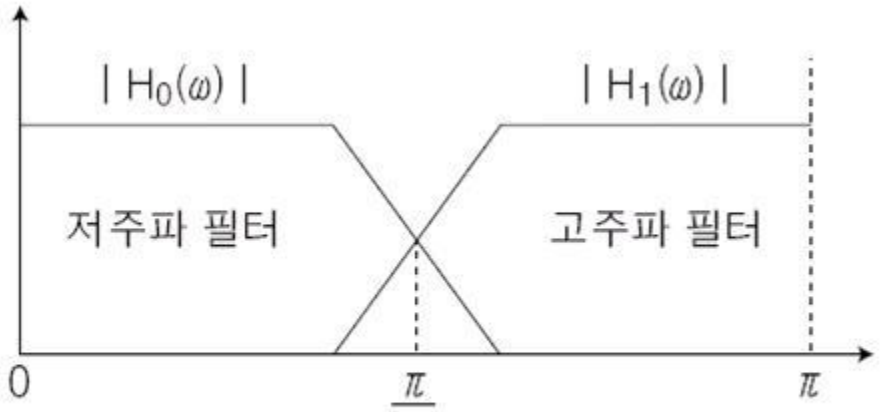
Section 01 웨이브렛 변환의 개념

- 웨이브렛은 기본 함수로 **sine, cosine** 함수 외에 웨이브렛 모함수를 사용하는데, 각 주파수 영역에 따라 변화하는 다양한 기저 함수를 생성하여 사용함
  - 시간-주파수에 국부적인 성질이 있음.

- 푸리에 변환은 시간과 주파수 정보를 동시에 파악할 수 없지만 웨이브렛에서는 이 둘을 동시에 파악 가능

필터 뱅크를 이용한 이산 웨이브렛 변환 수행

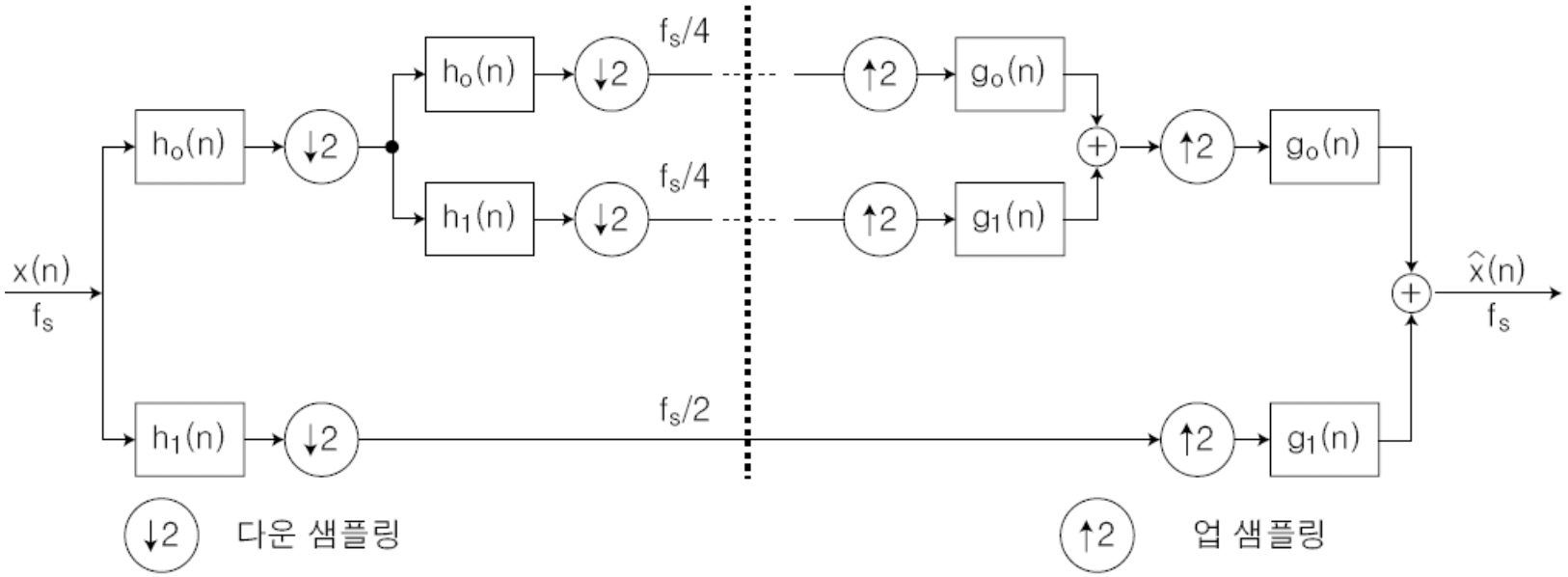
- 이산 웨이브렛 변환은 저주파 통과 필터와 고주파 통과 필터로 구성된 필터 뱅크로 수행됨.
- 이때, 사용되는 필터는 특수하게 설계된 것으로, 직교 특성, 선형 특성, 고주파와 저주파 부분을 정확하게 분할하는 특성이 있음.



[그림 15-1] 웨이브렛 필터의 주파수 응답

필터 뱅크를 이용한 이산 웨이브렛 변환 수행(계속)

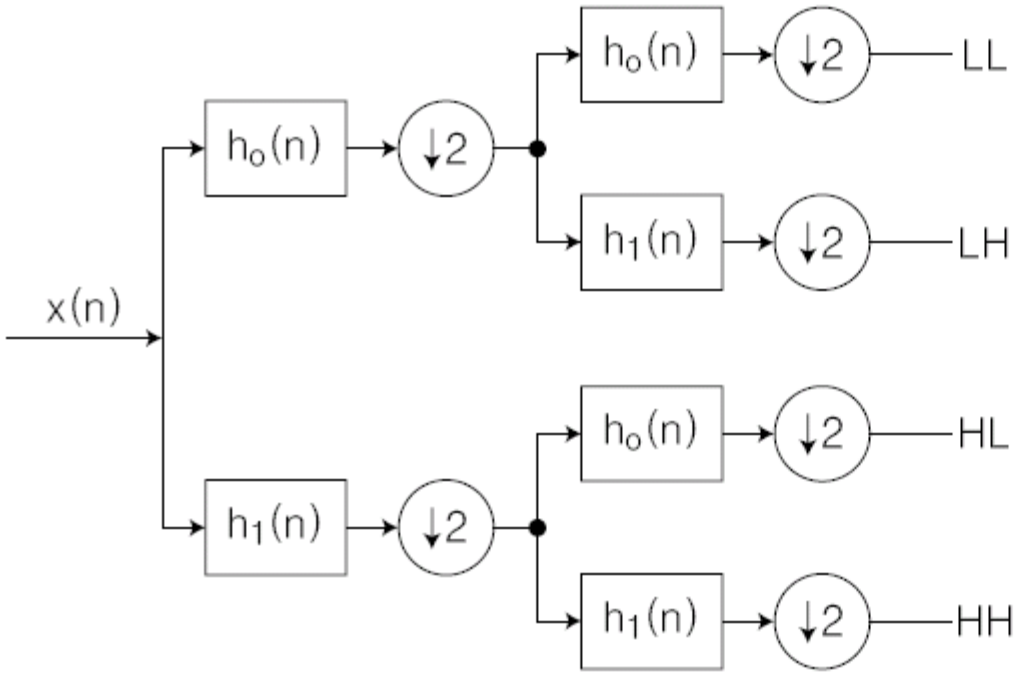
- 필터 뱅크의 동작을 이용한 웨이브렛 변환과 역변환의 수행 과정을 나타낸 것. 필터링은 컨벌루션으로 수행됨.
- 웨이브렛 변환 과정에서는 각각 필터링한 뒤 데이터의 크기를 절반으로 줄이는 다운 샘플링이 수행되고, 웨이브렛 역변환 과정에서는 각 필터링한 뒤 데이터의 크기를 원 상태로 보상해 주려고 두 배로 증가시키는 업 샘플링이 수행됨.



[그림 15-2] 필터 뱅크를 이용한 웨이브렛 변환과 역변환

필터 뱅크를 이용한 2차원 이산 웨이브렛 변환

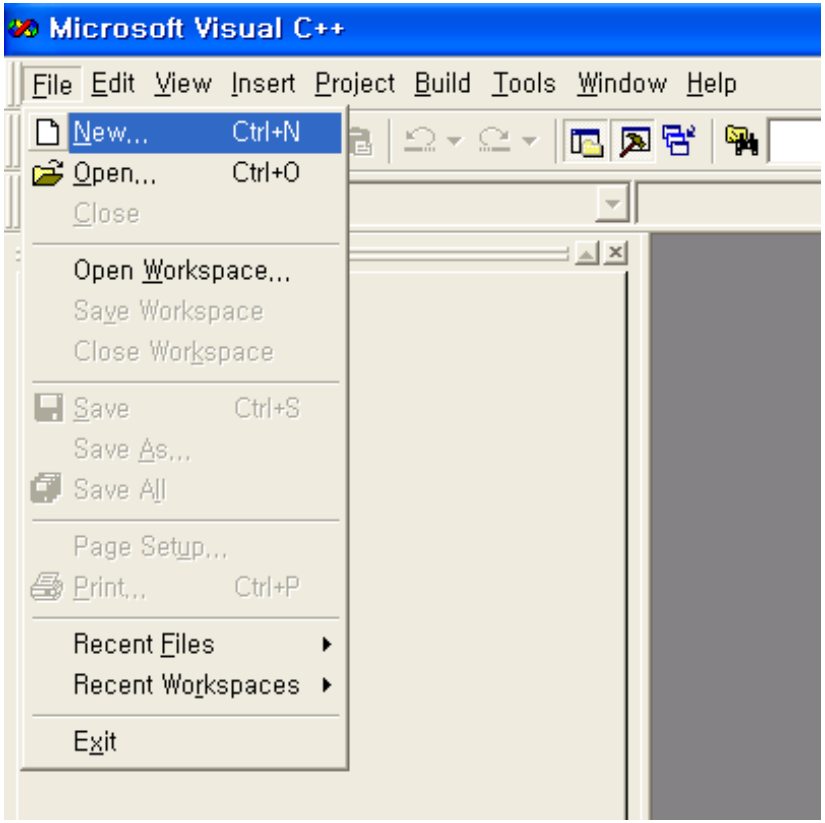
- 웨이브렛을 영상에 적용하려면 2차원 처리를 수행해야 함.
- 분리성을 만족함.
- 먼저 세로 방향으로 웨이브렛을 수행한 뒤 가로 방향으로 웨이브렛을 수행하면 2차원 웨이브렛 변환이 가능



[그림 15-3] 2차원 웨이브렛 변환

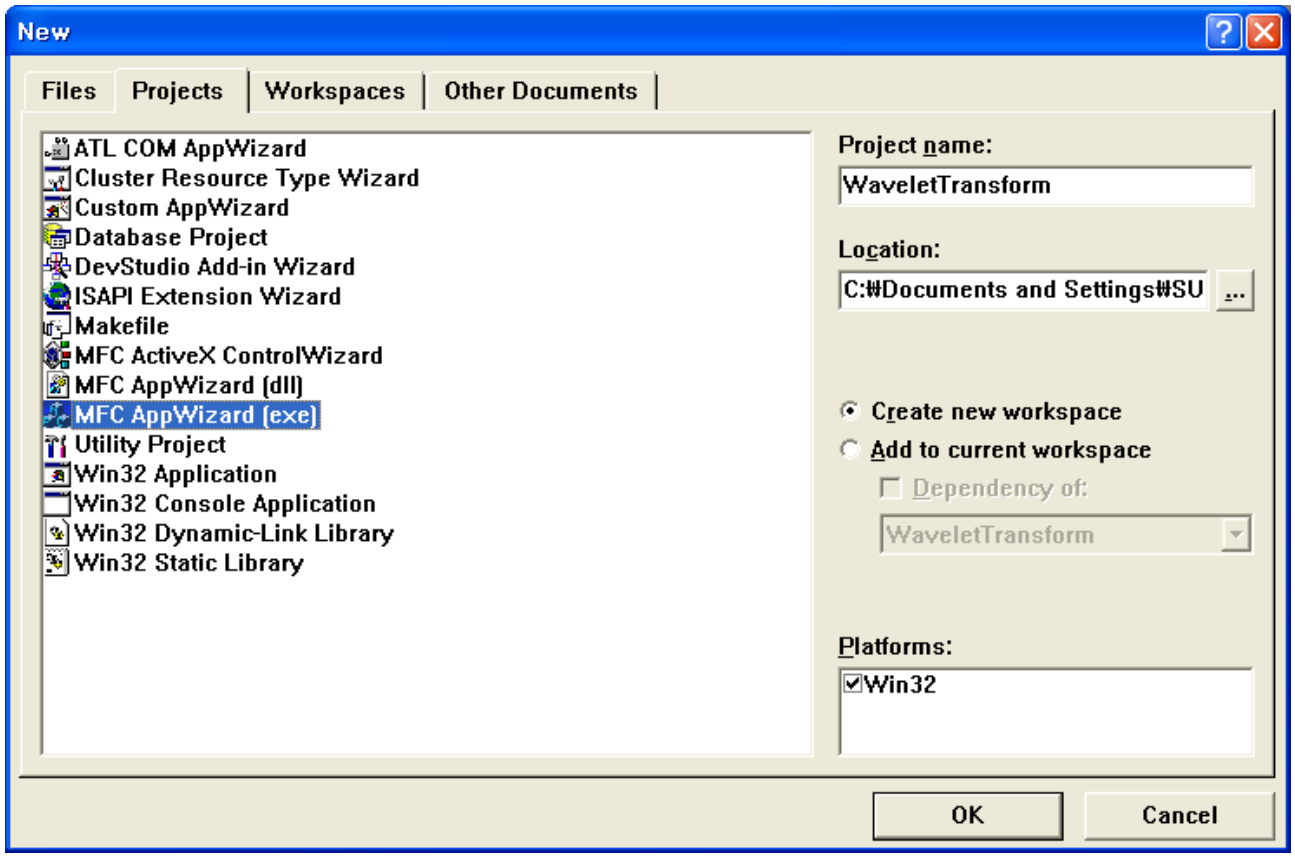
Section 02 웨이브렛 변환을 위한 MFC 설정하기

❶ **Visual C++ 프로그램을 실행 → [File]-[New] 메뉴 클릭**



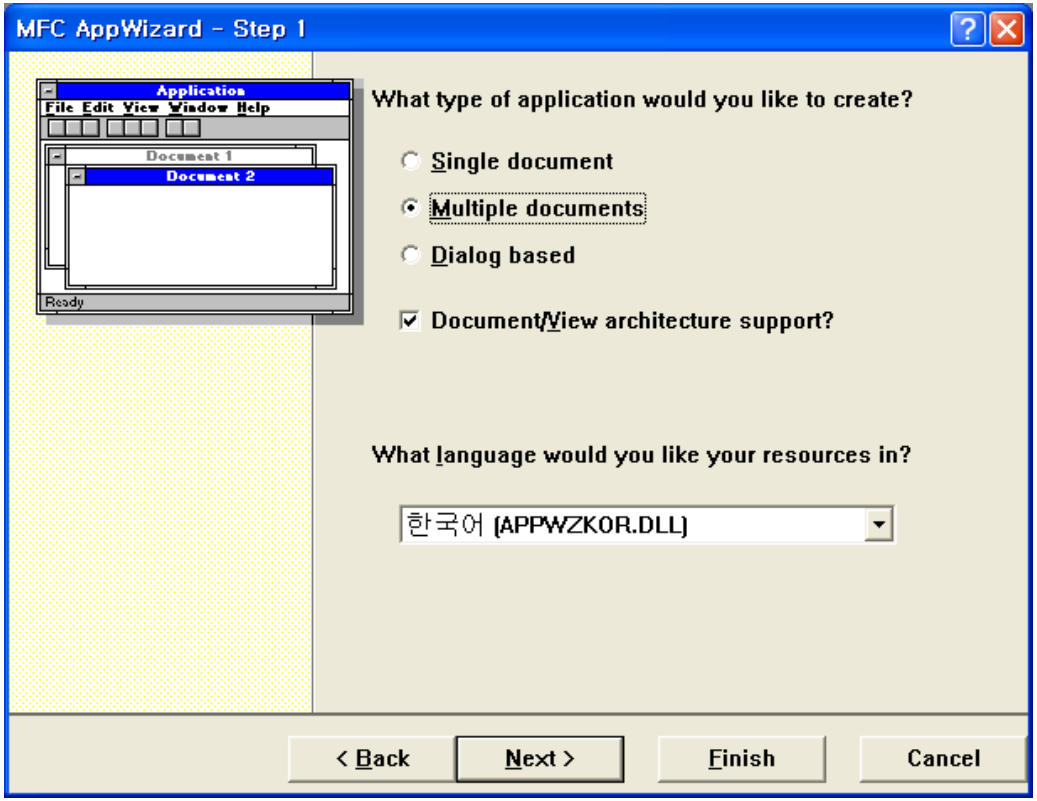
❷ **[New] 대화상자의 [Projects] 탭에서 MFC AppWizard [exe] 항목을 선택 → Project name란에 다음과 같이 입력**

Project name	WaveletTransform



❸ **[MFC AppWizard] 대화상자에서 MDI 기반에서 작성하기 위해 MDI(multiple documents) 항목을 선택 → [Finish] 버튼 클릭해 기본 환경 설정 완료**

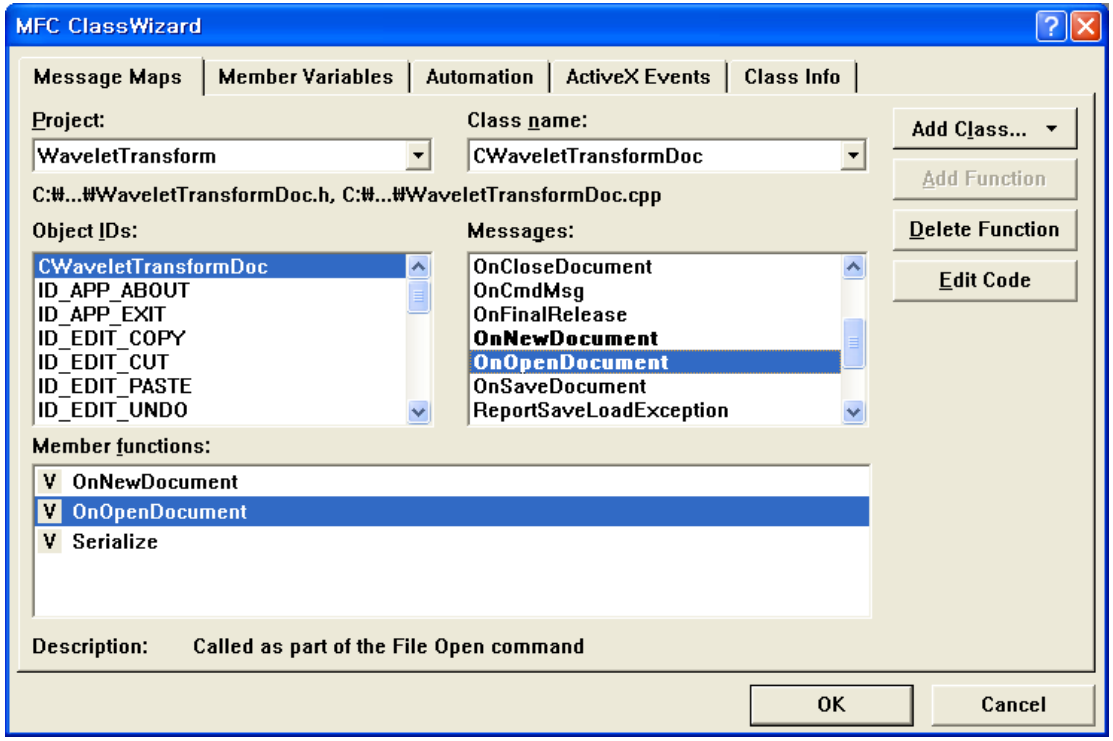
Type of application	Multiple documents
language	한국어 [APPWZKOR.DLL]



Section 03 파일 입·출력 설정하기

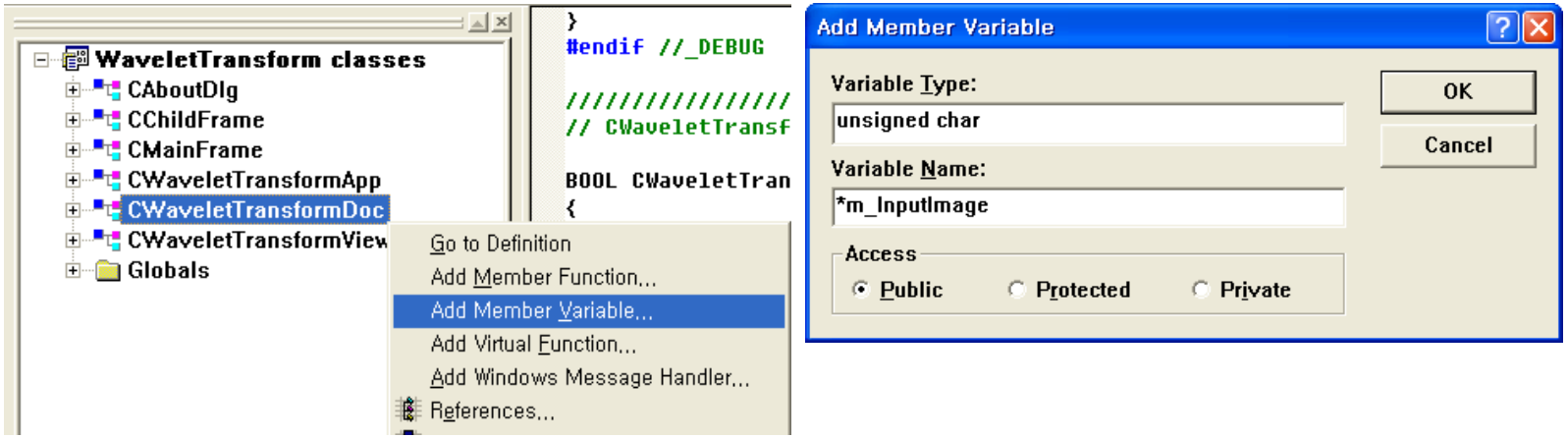
❶ Visual C++ 프로그램에서 [View]-[ClassWizard] 메뉴 클릭

❷ [MFC ClassWizard] 대화상자의 Class name 항목에서 CWaveletTransformDoc를 선택 → Message 항목에서 OnOpenDocument 더블 클릭→ CWaveletTransformDoc에 OnOpenDocument 함수 추가



Section 03 파일 입·출력 설정하기(계속)

❸ 함수에서 사용할 입력 영상 버퍼 변수, 출력 영상 버퍼 변수, 영상의 크기와 관련된 변수 추가



Variable Type	Variable Name	Access	설명
unsigned char	*m_InputImage	Public	입력 영상 버퍼
unsigned char	*m_OutputImage	Public	출력 영상 버퍼
unsigned char	**m_ArrangeImage	Public	변환 정렬 영상 버퍼
int	m_Height	Public	영상의 세로축 크기
int	m_Width	Public	영상의 가로축 크기
int	m_Size	Public	영상의 전체 크기

❹ **BOOL CWaveletTransformDoc::OnOpenDocument(LPCTSTR lpszPathName)** 에 다음 프로그램 추가(1)

```
BOOL CWaveletTransformDoc::OnOpenDocument(LPCTSTR lpszPathName)
{
   if(!CDocument::OnOpenDocument(lpszPathName))
   return FALSE;

   CFile File;

   File.Open(lpszPathName, CFile::modeRead | CFile::typeBinary);
   if(File.GetLength() == 256*256){ // RAW 파일의 크기 결정

      m_Height = 256;
      m_Width = 256;
   }
   else if(File.GetLength() == 512*512){ // RAW 파일의 크기 결정
      m_Height = 512;
      m_Width = 512;
   }
   else{
      AfxMessageBox("Not Support Image Size");
      return 0;
   }
```

❹ **BOOL CWaveletTransformDoc::OnOpenDocument(LPCTSTR lpszPathName)에 다음 프로그램 추가(2)**

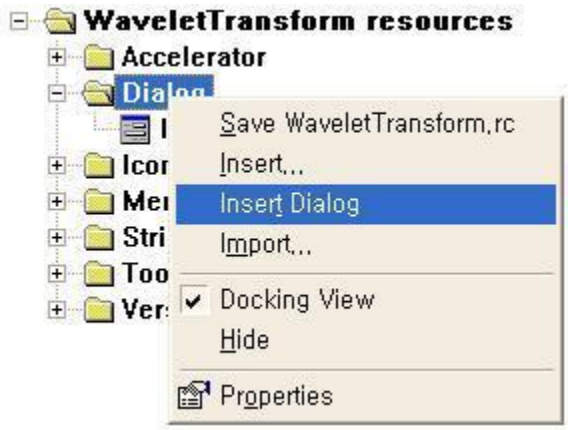
```
    m_Size = m_Height * m_Width;
    m_InputImage = new unsigned char [m_Height * m_Width];
    // 입력 영상 저장 공간
    m_OutputImage = new unsigned char [m_Height * m_Width];
    // 출력 영상 저장 공간

    for(int i = 0 ; i<m_Height * m_Width ; i++){
        m_InputImage[i] = 255; // 초기화
        m_OutputImage[i] = 255; // 초기화
    }

    File.Read(m_InputImage, m_Height * m_Width); // 파일 읽기
    File.Close();

    return TRUE;
}
```

❺ 입력된 영상을 확인하기 위해 **CWaveletTransformView::OnDraw(CDC* pDC)** 에 다음 프로그램 추가

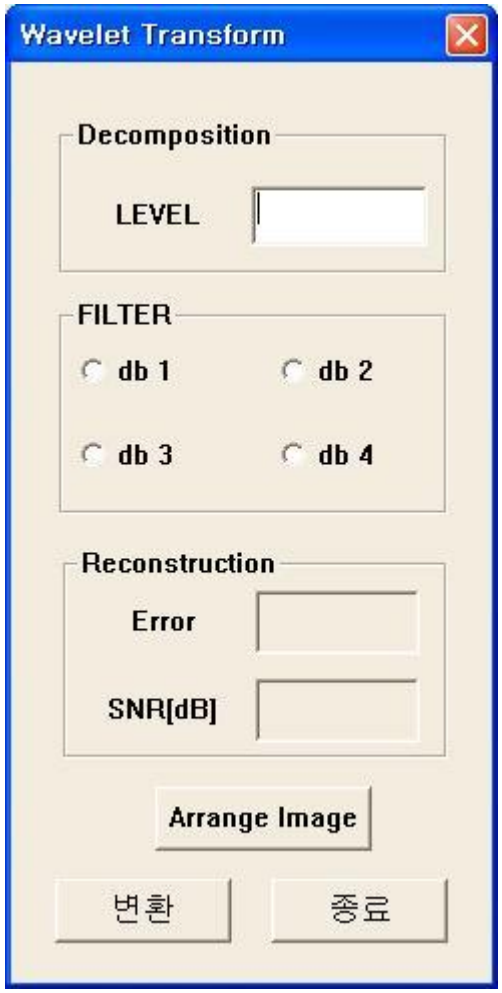
```
void CWaveletTransformView::OnDraw(CDC* pDC)
{
   CWaveletTransformDoc* pDoc = GetDocument();
   ASSERT_VALID(pDoc);
   // TODO: add draw code for native data here

   int i, j;
   int R, G, B;

   for(i=0 ; i<pDoc->m_Height ; i++){
   for(j=0 ; j<pDoc->m_Width ; j++){
      R = pDoc->      m_InputImage[i*pDoc->m_Width + j];
      B = G = R;
      pDC->SetPixel(j+5, i+5, RGB(R, G, B));
      // 입력 영상을 화면에 출력
   }
}
   for(i=0 ; i<pDoc->m_Height ; i++){
      for(j=0 ; j<pDoc->m_Width ; j++){
         R = pDoc->m_OutputImage[i*pDoc->m_Width + j];
         B = G = R;
         pDC->SetPixel(j+pDoc->m_Width+10, i+5, RGB(R, G, B));
         // 복원된 영상을 화면에 출력
      }
   }
}
```

Section 04 웨이브렛 변환을 위한 대화상자 설정하기

❶ **ResourceView 창에서 [WaveletTransform resources]-[Dialog] 폴더에서 마우스 오른쪽 버튼을 클릭 → 바로가기 메뉴 [Insert Dialog] 클릭**



❷ 추가된 대화상자를 오른쪽과 같이 편집



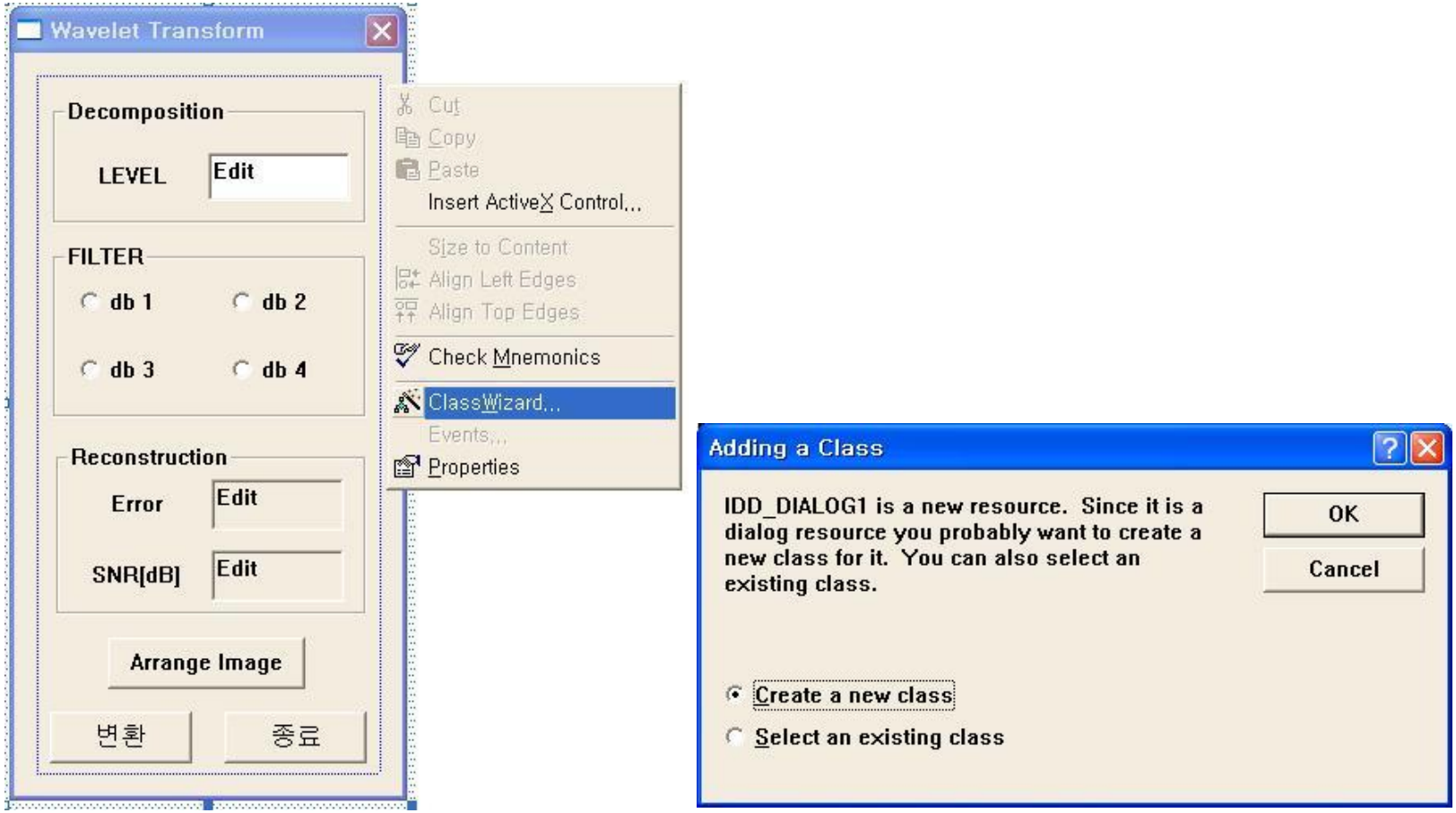
Section 04 웨이브렛 변환을 위한 대화상자 설정하기(계속)

❸ 대화상자 속성을 다음과 같이 정의

	ID	Caption	Option
Dialog Properties	IDD_DIALOG1	Wavelet Transform	
Group Box Properties	IDC_STATIC	Decomposition	
Group Box Properties	IDC_STATIC	FILTER	
Group Box Properties	IDC_STATIC	Reconstruction	
Text Properties	IDC_STATIC	LEVEL	
Text Properties	IDC_STATIC	Error	
Text Properties	IDC_STATIC	SNR[dB]	
Edit Properties	IDC_EDIT1		Tab stop
Edit Properties	IDC_EDIT2		Tab stop
Edit Properties	IDC_EDIT3		Tab stop
Radio Button Properties	IDC_RADIO1	db 1	Group
Radio Button Properties	IDC_RADIO2	db 2	
Radio Button Properties	IDC_RADIO3	db 3	
Radio Button Properties	IDC_RADIO4	db 4	
Push Button Properties	IDC_BUTTON_ARRANGE	Arrange Image	
Push Button Properties	IDC_BUTTON_UPDATE	변환	
Push Button Properties	IDC_BUTTON_END	종료	

Section 04 웨이브렛 변환을 위한 대화상자 설정하기(계속)

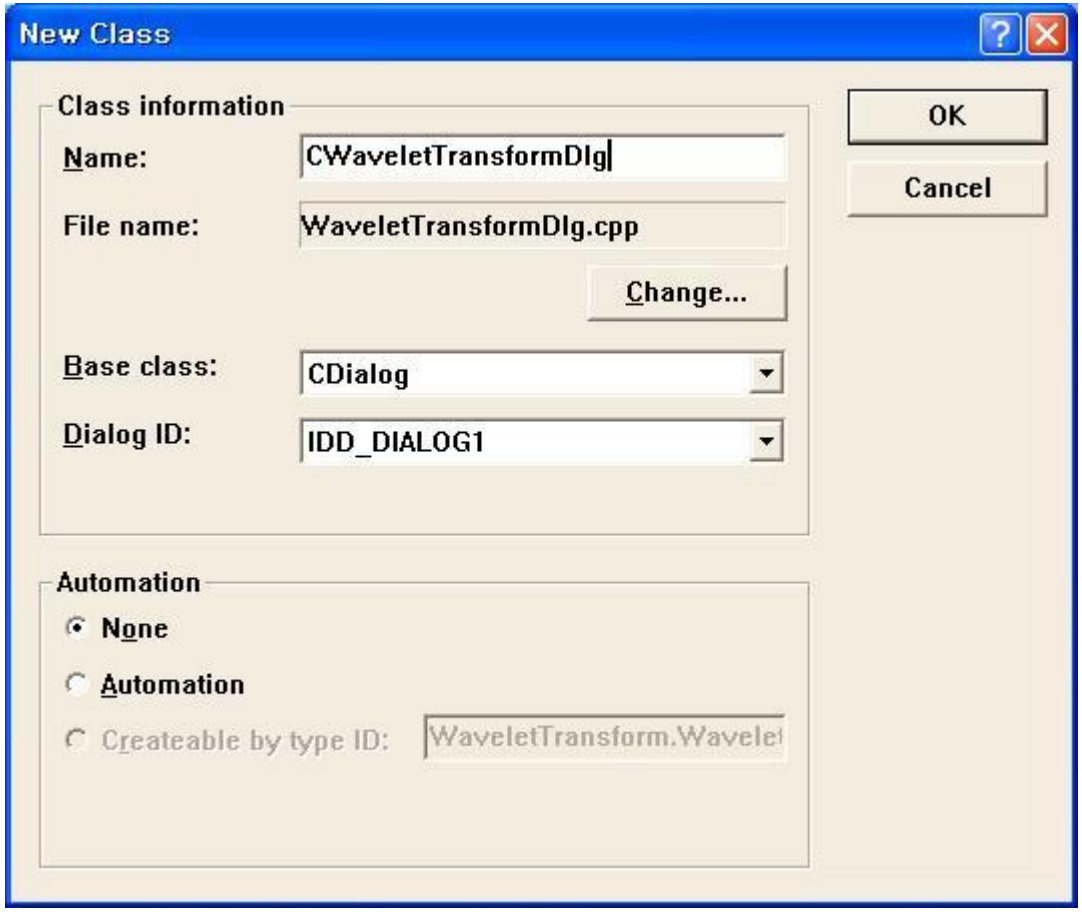
❹ 대화상자 위에서 마우스 오른쪽 버튼을 눌러 **[ClassWizard]** 메뉴를 클릭
→ **[ClassWizard]** 대화상자에서 **[Add Class]** 버튼 클릭
→ **[Adding a Class]** 대화상자에서 **Create a new class** 항목 선택



Section 04 웨이브렛 변환을 위한 대화상자 설정하기(계속)

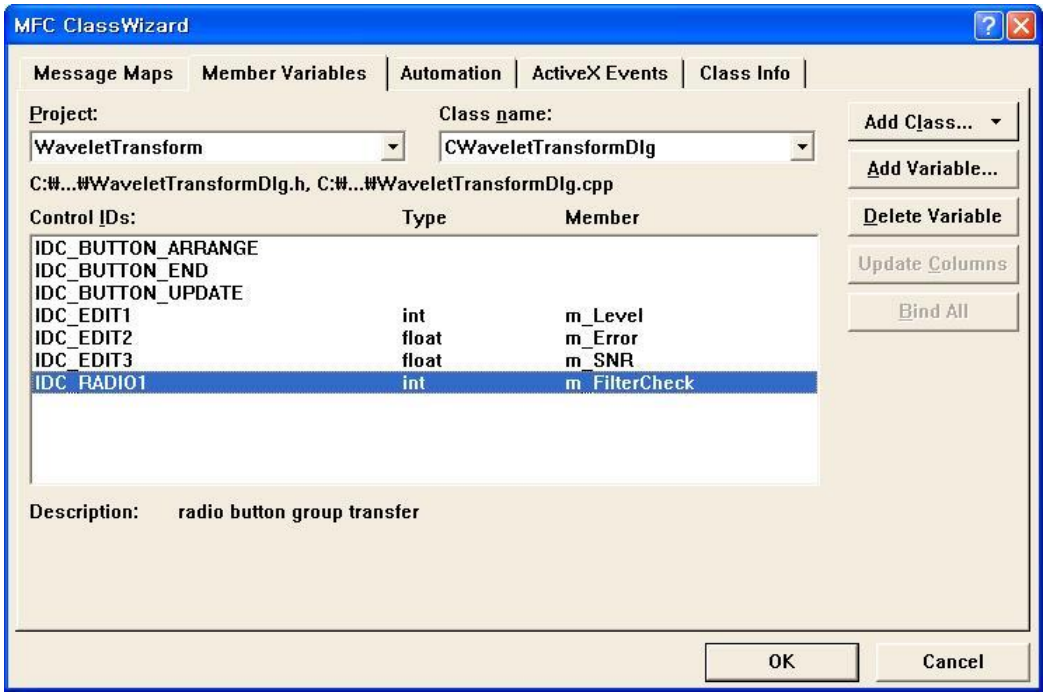
❺ 다음과 같이 대화상자를 클래스에 등록

Class Name	CWaveletTransformDlg



❻ **[Class Wizard] 대화상자에서 [Add Variable] 버튼 클릭 → [Add Member Variable] 대화상자에서 CWaveletTransforDlg 대화상자를 선택 → 변수 추가**

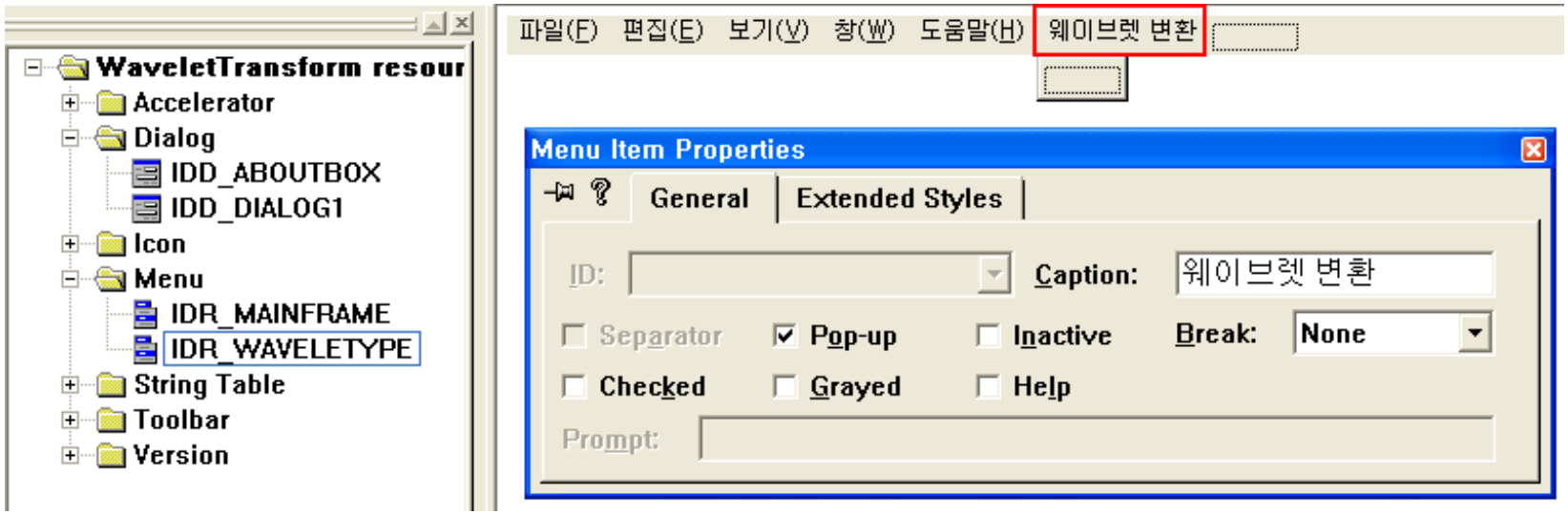
Control IDs	Member variable name	Variable Type
IDC_EDIT1	m_Level	int
IDC_EDIT2	m_Error	float
IDC_EDIT3	m_SNR	float
IDC_RADIO1	m_FilterCheck	int



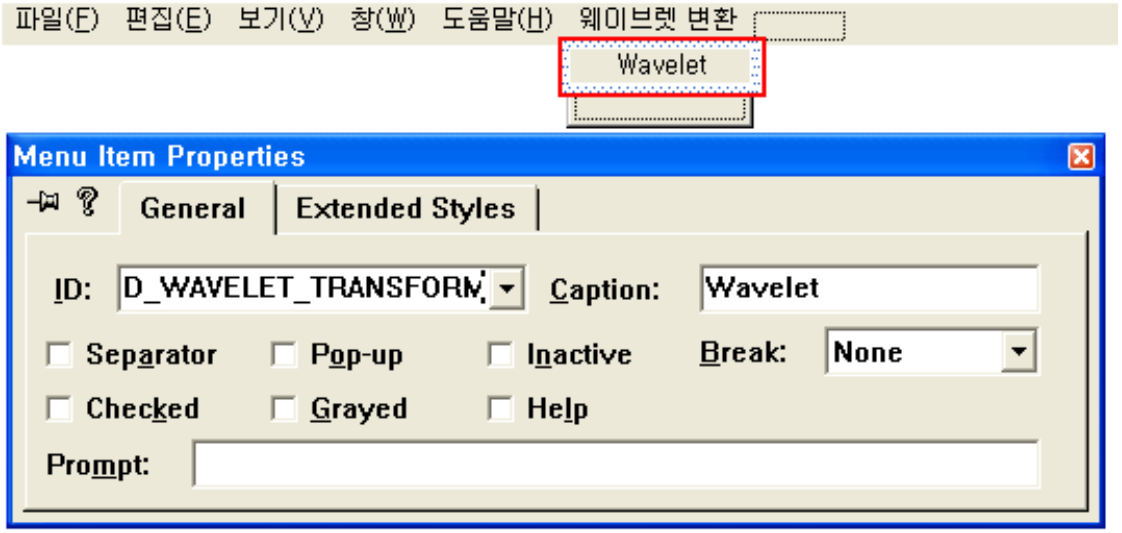
Section 05 순방향 웨이브렛 변환 구현하기

❶ **ResourceView 창에서 [Menu]-[IDR_IMAGETYPE] 더블클릭해 메뉴 추가**

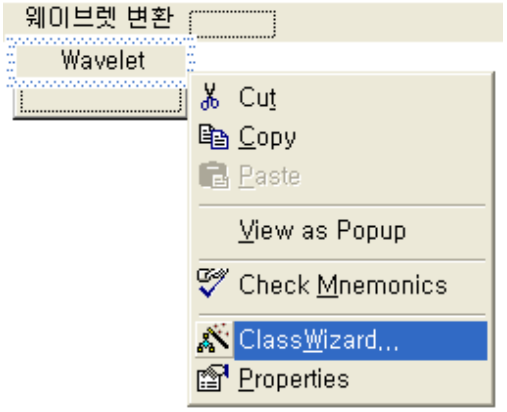
❷ 메뉴를 추가할 부분을 더블클릭→ **[Menu Item Properties]** 대화상자에서 다음과 같이 캡션 추가



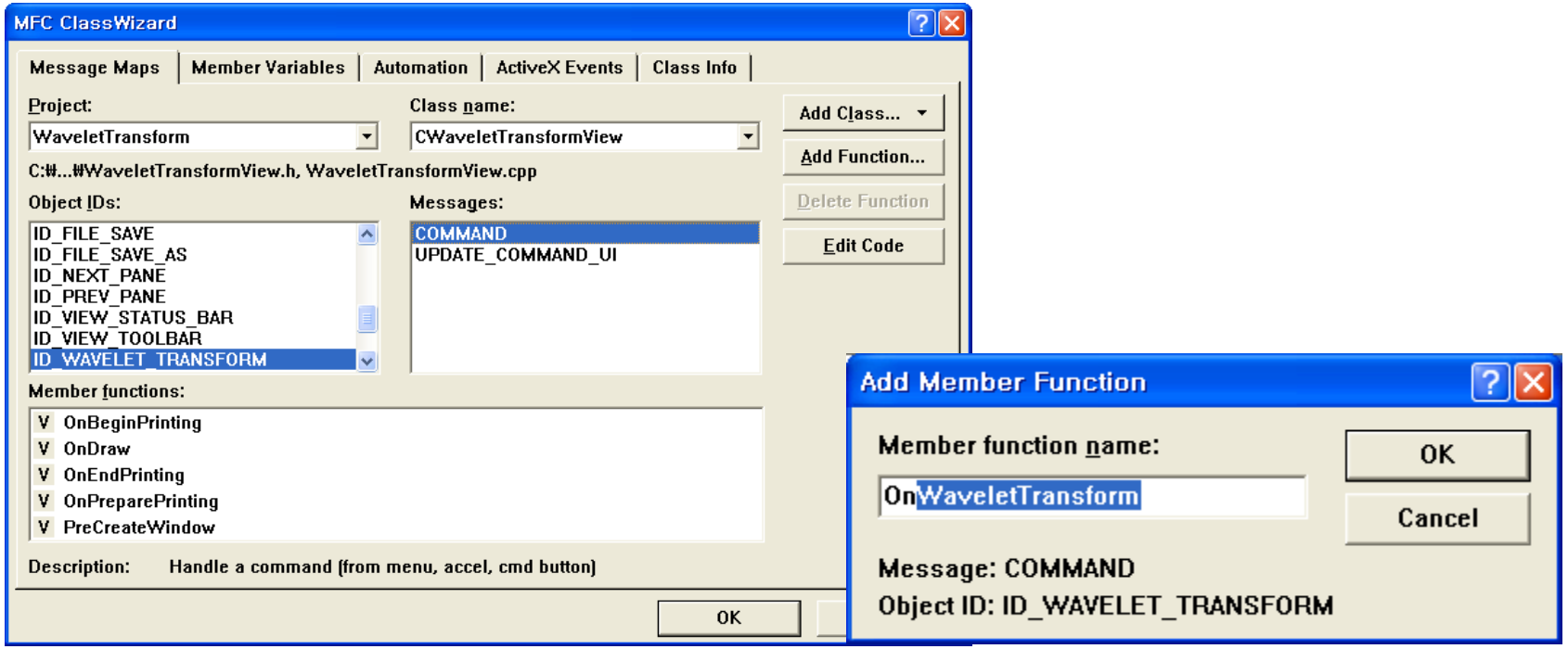
❸ 프로그램을 실행하는 실제 메뉴를 추가하려고 ❷처럼 메뉴가 들어갈 위치를 더블클릭→ **[Menu Item Properties]** 대화상자에서 다음과 같이 속성 지정



❹ 해당 메뉴 위에서 마우스 오른쪽 버튼을 눌러 바로가기 메뉴 **[ClassWizard]** 클릭



❺ **[MFC Class Wizard] 대화상자의 [Message Maps] 탭에서 Object IDs 항목과 Messages 항목을 다음과 같이 지정하고 [Add Function] 버튼 클릭**
→ [Add Member Function] 대화상자에서 [OK] 버튼을 눌러 함수를 추가
→ [Edit Code] 버튼을 눌러 함수로 이동

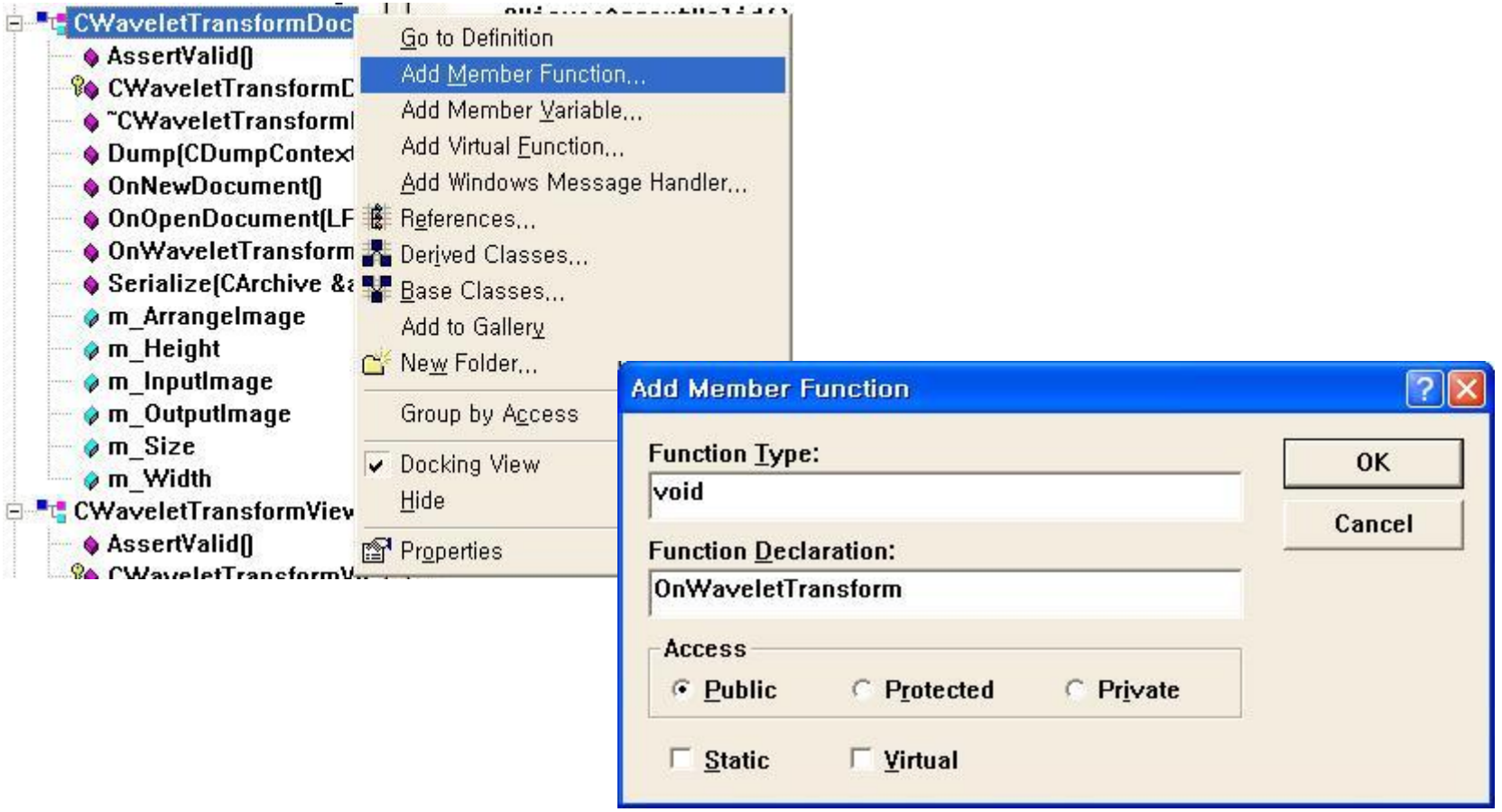


Member function name	OnWaveletTransform

❻ CWaveletTransformView::OnWaveletTransform에 다음과 같이 프로그램을 작성. CWaveletTransformView는 CWaveletTransformDoc를 호출하는 역할을 하고, 웨이브렛 변환은 CWaveletTransformDoc에서 일어나게 됨

```
void CWaveletTransformView::OnWaveletTransform()
{
    CWaveletTransformDoc* pDoc = GetDocument();
    ASSERT_VALID(pDoc);

    pDoc->OnWaveletTransform(); // Doc 클래스에서 생성해야 할 함수 이름

    Invalidate(TRUE);
}
```

❼ **ResoureViess 창의 CWaveletTransformDoc 폴더 위에서 마우스 오른쪽 버튼 클릭 후 바로 가기 메뉴 [Add Member Function] 클릭→ [Add Member Function] 대화상자에서 OnWaveletTransform 함수 추가**



Section 05 순방향 웨이브렛 변환 구현하기(계속)

❽ CWaveletTransformDoc::OnWaveletTransform에서는 입력된 영상과 미리 만든 대화상자를 이용하여 웨이브렛 변환 실행. Wavelet Transform 대화상자를 호출할 수 있도록 [File View]-[Header File]-[WaveletTransformDoc.h] 파일을 클릭하여 다음 부분 추가

```
// aveletTransformDoc.h : interface of the CWaveletTransformDoc class
/////////////////////////////////////////////////////////////////////
#if !defined(AFX_WAVELETTRANSFORMDOC_H__4A38BAEB_
E3CB_44C9_B2CB_057CD4D9FF0A__INCLUDED_)
#define AFX_WAVELETTRANSFORMDOC_H__4A38BAEB_E3CB_44C9_
B2CB_057CD4D9FF0A__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

#include "WaveletTransformDlg.h" // 대화상자 헤더 선언

class CWaveletTransformDoc : public CDocument
{
protected: // create from serialization only
   CWaveletTransformDoc();
   DECLARE_DYNCREATE(CWaveletTransformDoc)
// Attributes

public:
// Operations
public:
   CWaveletTransformDlg *pDlg;
```

❾ CWaveletTransformDoc 클래스의 생성자와 소멸자를 다음과 같이 작성

```
CWaveletTransformDoc::CWaveletTransformDoc()
{
    // TODO: add one-time construction code here
    pDlg = new    CWaveletTransformDlg(this);
}
    CWaveletTransformDoc::~CWaveletTransformDoc()
{
    delete pDlg;
}
```

Section 05 순방향 웨이브렛 변환 구현하기(계속)

⑩ 생성된 함수 CWaveletTransformDoc::OnWaveletTransform을 다음과 같이 작성
여기서 WaveletTransform 대화상자를 호출하게 됨.

```
void CWaveletTransformDoc::OnWaveletTransform()
{
    if(pDlg->GetSafeHwnd() == NULL)
        pDlg->Create(IDD_DIALOG1); // 모달리스 대화상자 이용

    pDlg->ShowWindow(SW_SHOW);
}
```

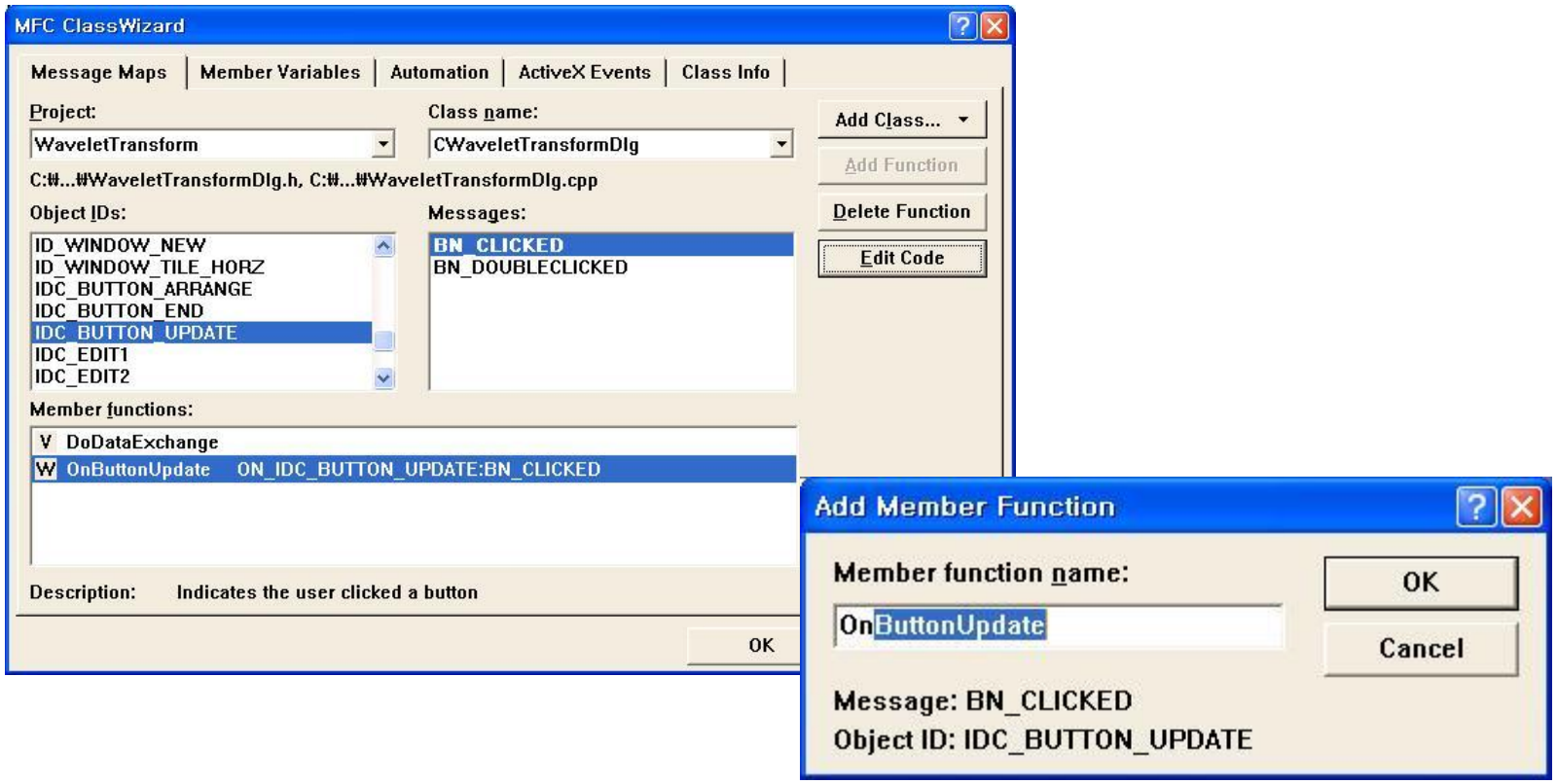
⓫ CWaveletTransformDlg 클래스로 이동. 대화상자를 이용하여 변환에 사용되는 변수를 입력받게 됨. [File View]-[Header Files]-[WaveletTransformDlg.h] 파일로 이동하여 다음 프로그램 추가

```
/////////////////////////////////////////////////////////////////
//    CWaveletTransformDlg dialog
class CWaveletTransformDoc;

class CWaveletTransformDlg : public CDialog
{
// Construction
public:
   CWaveletTransformDlg(CWnd* pParent = NULL); // standard
      constructor
   CWaveletTransformDlg(CWaveletTransformDoc *pDoc, CWnd*
      pParent = NULL);
                          :
// Implementation
protected:
   CWaveletTransformDoc *m_pDoc;
```

⑫ [File View]-[Source Files]-[WaveletTransformDlg.cpp] 파일로 이동하여 다음과 같이 프로그램 수정

```
#include "WaveletTransformDoc.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////
//    CWaveletTransformDlg dialog
   CWaveletTransformDlg::CWaveletTransformDlg(CWnd* pParent /*=NULL*/)
   : CDialog(CWaveletTransformDlg::IDD, pParent)
{
   m_pDoc = NULL;
}
   CWaveletTransformDlg::CWaveletTransformDlg(CWaveletTransform
      Doc *pDoc, CWnd* pParent /*=NULL*/)
         : CDialog(CWaveletTransformDlg::IDD, pParent)
   {
      m_pDoc = pDoc; // Doc 클래스 참조
      m_Level = 1; // Wavelet 분해 레벨 초기화
      m_Error = 0.0f; // MSE 초기화
      m_SNR = 0.0f; // SNR 초기화
      m_FilterCheck = 0; // Filter Check 초기화
   }
```

❸ **[MFC Class Wizard] 대화상자의 Wavelet Transform 대화상자에서 [변환] 버튼 클릭→ 웨이브렛 변환이 일어나도록 다음과 같이 함수 추가**



Class name	CWaveletTransformDlg
Member function name	OnButtonUpdate

```
void CWaveletTransformDlg::OnButtonUpdate()
{
    // TODO: Add your control notification handler code here
    UpdateData(TRUE);
    m_pDoc->m_Level = m_Level;
    // 대화상자에서 입력받은 레벨 값을 Doc 클래스에 전달
    m_pDoc->OnWaveletEncode(); // 웨이브렛 변환
    UpdateData(FALSE); // 결과 업데이트
}
```

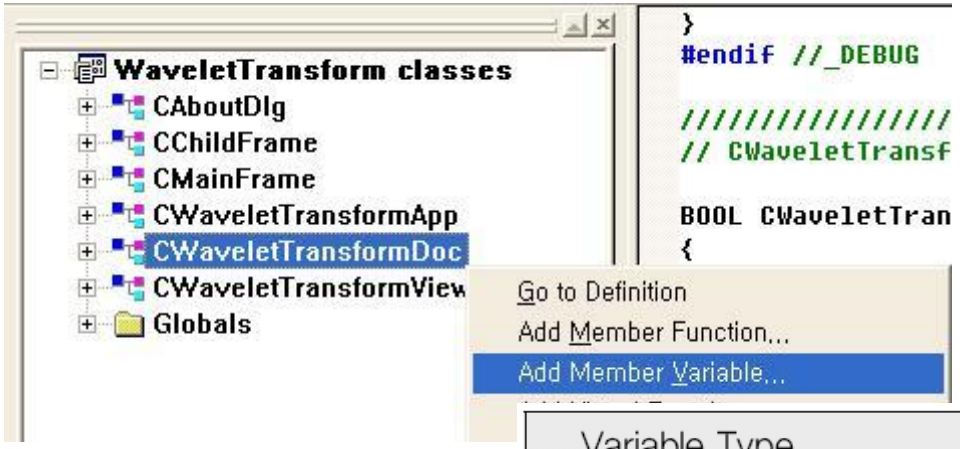
14 대화상자의 변환 버튼이 클릭되면 변환이 실행되고 실행 코드는 **CWaveletTransformDoc** 클래스에서 있게 됨. **CWaveletTransformDoc** 클래스로 이동하여 **OnWaveletEncode** 함수 추가

Function Type	void
Function Declaration	OnWaveletEncode
Access	Public

⑮ 수학 함수를 사용하기 위해 **CWaveletTransforDoc.cpp**의 위쪽에 다음과 같이 선언

```
// WaveletTransformDoc.cpp:implementation of the CWaveletTransformDoc class
//
#include "stdafx.h"
#include "WaveletTransform.h"

#include "WaveletTransformDoc.h"
#include "math.h" // 수학 함수를 위한 헤더 선언
```

16 **OnWaveletEncode 함수에서 사용되는 전역 변수를 선언하기 위해 [WaveletTransform Classes]-[CWaveletTransformDoc] 폴더 위에서 마우스 오른쪽 버튼 클릭→[Add Member Variable] 클릭→ [Add Member Variable] 대화상자를 이용해 변수 추가**



Variable Type	Variable Name	Access
**m_tempInput	double	Public
**m_tempOutput	double	Public
m_Level	int	Public
m_FilterTap	int	Public
*m_FilterH0	double	Public
*m_FilterH1	double	Public
*m_FilterG0	double	Public
*m_FilterG1	double	Public

17 **CWaveletTransformDoc 폴더 위에서 마우스 오른쪽 버튼 클릭 →[Add Member Function] 클릭→ [Add Member Function] 대화상자에서 OnWaveletEncode 함수에서 사용되는 함수를 선언하고 작성**

함수 타입	함수명	설명
void	OnWaveletEncode	
void	OnFilterTapGen( )	필터 탭 생성 함수
void	OnFilterGen(double* m_H0, double* m_H1, double* m_G0, double* m_G1)	필터 생성 함수
double*	OnDownSampling(double *m_Target, int size)	다운 샘플링 함수
double*	OnConvolution(double* m_Target, double* m_Filter, int size, int mode)	1차원 컨벌루션 함수
unsigned char**	OnMem2DAllocUnsigned(int height, int width)	2차원 메모리 할당
double**	OnMem2DAllocDouble(int height, int width)	2차원 메모리 할당
double**	OnScale(double **m_Target, int height, int width)	정규화 함수

① **OnDownSampling** 함수 추가하기

```
double* CWaveletTransformDoc::OnDownSampling(double *m_Target,
    int size)
{
    int i;
    double* m_temp;

    m_temp = new double [size / 2];

    for(i=0 ; i<size/2 ; i++)
        m_temp[i] = m_Target[2*i]; // 다운 샘플링 처리

    return m_temp;
}
```

② **OnConvolution 함수 추가하기(1)**

```
double* CWaveletTransformDoc::OnConvolution(double *m_Target,
      double *m_Filter, int size, int mode)
{ // Circular Convolution을 위한 함수
   int i, j;
   double *m_temp, *m_tempConv;

   double m_sum = 0.0;

   m_temp = new double [size + m_FilterTap - 1];
   m_tempConv = new double [size]; // Convolution 결과 출력 배열

   switch(mode){
   case 1 : // Circular Convolution을 위한 초기화
      for(i=0 ; i<size ; i++)
         m_temp[i] = m_Target[i];

   for(i=0 ; i<m_FilterTap-1 ; i++)
         m_temp[size + i] = m_Target[i];
   break;
```

② **OnConvolution 함수 추가하기(2)**

```
case 2 :
      for(i=0 ; i<m_FilterTap-1 ; i++)
         m_temp[i] = m_Target[size - m_FilterTap + i + 1];
      for(i=m_FilterTap-1 ; i<size + m_FilterTap-1 ; i++)
         m_temp[i] = m_Target[i - m_FilterTap + 1];
   break;
   }

   for(i=0 ; i<size ; i++){
      for(j=0 ; j<m_FilterTap ; j++){
         m_sum += (m_temp[j+i] * m_Filter[m_FilterTap-j-1]);
         // Convolution 연산
      }
      m_tempConv[i] = m_sum;
      m_sum = 0.0;
   }
   return m_tempConv; // 연산 결과를 반환
}
```

③ **OnFilterTapGen 함수 추가하기**

```
void CWaveletTransformDoc::OnFilterTapGen()
{ // Filter Tap 선택
   switch(pDlg->m_FilterCheck)
   {
   case 0 : m_FilterTap = 2;
      break;
   case 1 : m_FilterTap = 4;
      break;
   case 2 : m_FilterTap = 6;
      break;
   case 3 : m_FilterTap = 8;
      break;
   default : AfxMessageBox("Wrong Filter Tap");
   return;
   }
}
```

④ OnFilterGen 함수 추가하기(1)

```
void CWaveletTransformDoc::OnFilterGen(double *m_H0,
            Double *m_H1, double *m_G0, double *m_G1)
{ // 필터 계수 값
   int i;
   switch(m_FilterTap)
   {
   case 2 :
      m_H0[0] = 0.70710678118655;
      m_H0[1] = 0.70710678118655;
      break;
   case 4 :
      m_H0[0] = -0.12940952255092;
      m_H0[1] = 0.22414386804186;
      m_H0[2] = 0.83651630373747;
      m_H0[3] = 0.48296291314469;
      break;
   case 6 :
      m_H0[0] = 0.03522629188210;
      m_H0[1] = -0.08544127388224;
      m_H0[2] = -0.13501102001039;
      m_H0[3] = 0.45987750211933;
      m_H0[4] = 0.80689150931334;
      m_H0[5] = 0.33267055295096;
      break;
```

④ OnFilterGen 함수 추가하기(2)

```
    case 8 :
       m_H0[0] = -0.01059740178500;
       m_H0[1] = 0.03288301166698;
       m_H0[2] = 0.03084138183599;
       m_H0[3] = -0.18703481171888;
       m_H0[4] = -0.02798376941698;
       m_H0[5] = 0.63088076792959;
       m_H0[6] = 0.71484657055254;
       m_H0[7] = 0.23037781330886;
       break;
    default :
       AfxMessageBox("Wrong Filter");
       return;
    }
    // H0 필터 계수를 이용해, H1, G0, G1 필터 계수 생성
    for(i=0 ; i<m_FilterTap ; i++)
       m_H1[i] = pow(-1,i+1) * m_H0[m_FilterTap - i - 1];

    for(i=0 ; i<m_FilterTap ; i++)
       m_G0[i] = m_H0[m_FilterTap - i - 1];

    for(i=0 ; i<m_FilterTap ; i++)
       m_G1[i] = pow(-1, i) * m_H0[i];
}
```

⑤ OnMem2DAllocUnsigned 함수 추가하기

```
unsigned char** CWaveletTransformDoc::OnMem2DAllocUnsigned
      (int height, int width)
{ // unsigned char 형태의 2차원 배열 할당
   int i, j;
   unsigned char** temp;

   temp = new unsigned char *[height];

   for(i=0 ; i<height ; i++) // 2차원 배열 할당
      temp[i] = new unsigned char [width];

   for(i=0 ; i<height ; i++){
      for(j=0 ; j<width ; j++){
         temp[i][j] = 0; // 2차원 배열 초기화
      }
   }
   return temp;
}
```

⑥ OnMem2DAlloc double 함수 추가하기

```
double** CWaveletTransformDoc::OnMem2DAlloc double(int height, int width)
{ // double 형태의 2차원 배열 할당
   int i, j;
   double** temp;

   temp = new double *[height];

   for(i=0 ; i<height ; i++)
      temp[i] = new double [width];

   for(i=0 ; i<height ; i++){
      for(j=0 ; j<width ; j++){
      temp[i][j] = 0;
      }
   }
   return temp;
}
```

⑦ OnScale 함수 추가하기(1)

```
double** CWaveletTransformDoc::OnScale(double **m_Target,
          int height, int width)
{ // 정규화 함수 : 필터링된 값을 0~255 사이의 값으로 정규화
   int i, j;
   double min, max;
   double **temp;

   temp = OnMem2DAlloc   double(height, width);

   min = max = m_Target[0][0];

   for(i=0 ; i<height ; i++){
      for(j=0 ; j<width ; j++){
         if(m_Target[i][j] <= min){
            min = m_Target[i][j]; // 최소값
         }

         if(m_Target[i][j] >= max){
            max = m_Target[i][j]; // 최대값
         }
      }
   }
```

⑦ **OnScale 함수 추가하기(2)**

```
    max = max - min;
    for(i=0 ; i<height ; i++){
        for(j=0 ; j<width ; j++){
            temp[i][j] = (m_Target[i][j] - min) * (255. / max);
            // 정규화 처리
        }
    }
    return temp;
}
```

18 OnWaveletEncode 함수 작성(1)

```
void CWaveletTransformDoc::OnWaveletEncode()
{ // Wavelet encode 함수
   if(m_Level <= 0 || (pow(2, m_Level+3) > (double) m_Width)||(pow(2, m_Level+3) > (double) m_Height)){
      AfxMessageBox("Not support decomposition level");
      return;
      // 최대 분해 레벨이 512*512이면 6레벨로 제한
   }

   int i, j, k, width, height;
   double *m_Conv1, *m_Conv2, *m_Conv3, *m_Conv4;
   // Convolution을 위한 버퍼
   double *m_Down1, *m_Down2, *m_Down3, *m_Down4;
   // 다운 샘플링을 위한 버퍼
   double *m_Hor, *m_Ver1, *m_Ver2;
   double **m_L, **m_H, **m_LL, **m_LH, **m_HL, **m_HH, **m_SLL, **m_SLH, **m_SHL, **m_SHH;

   m_tempInput = OnMem2DAlloc   double(m_Height, m_Width);
   m_tempOutput = OnMem2DAlloc   double(m_Height, m_Width);
   m_ArrangeImage = OnMem2DAllocUnsigned(m_Height, m_Width);

   for(i=0 ; i<   m_Height ; i++){
      for(j=0 ; j<   m_Width ; j++){
         m_tempInput[i][j]
            = (double)      m_InputImage[i*   m_Width + j];
         // 1차원 입력을 2차원 배열로 변환
      }
   }
```

18 OnWaveletEncode 함수 작성(2)

```
OnFilterTapGen(); // 필터 tap 생성

m_FilterH0 = new double [m_FilterTap]; // 필터 계수를 위한 배열
m_FilterH1 = new double [m_FilterTap]; // 필터 계수를 위한 배열
m_FilterG0 = new double [m_FilterTap]; // 필터 계수를 위한 배열
m_FilterG1 = new double [m_FilterTap]; // 필터 계수를 위한 배열

OnFilterGen(m_FilterH0, m_FilterH1, m_FilterG0, m_FilterG1);
// 필터 계수 생성

width= m_Width;
height= m_Height;

for(k=0 ; k<m_Level ; k++){
   m_L = OnMem2DAlloc   double(height, width/2); //
   m_H = OnMem2DAlloc   double(height, width/2); //
   m_LL = OnMem2DAlloc   double(height/2, width/2);
   // LL 저장을 위한 배열
   m_LH = OnMem2DAlloc   double(height/2, width/2);
   // LH 저장을 위한 배열
   m_HL = OnMem2DAlloc   double(height/2, width/2);
   // HL 저장을 위한 배열
   m_HH = OnMem2DAlloc   double(height/2, width/2);
   // HH 저장을 위한 배열

   m_Hor = new double [width]; // 횡 입력을 위한 배열
```

18 OnWaveletEncode 함수 작성(3)

```
        for(i=0 ; i<height ; i++){
        for(j=0 ; j<width ; j++){
        m_Hor[j] = m_tempInput[i][j];
        // 입력 배열을 1차원 배열에 할당
    }

        m_Conv1 = OnConvolution(m_Hor, m_FilterH0, width, 1);
        // Convolution 처리
        m_Conv2 = OnConvolution(m_Hor, m_FilterH1, width, 1);
        // Convolution 처리
        m_Down1 = OnDownSampling(m_Conv1, width); // 다운 샘플링
        m_Down2 = OnDownSampling(m_Conv2, width); // 다운 샘플링

        for(j=0 ; j<width/2 ; j++){// 다운 샘플링 결과를 저장
            m_L[i][j] = m_Down1[j];
            m_H[i][j] = m_Down2[j];
        }
    }

    m_Ver1 = new double[height];

    m_Ver2 = new double[height];
```

⑱ OnWaveletEncode 함수 작성(4)

```
for(i=0 ; i<width/2 ; i++){
   for(j=0 ; j<height ; j++){
      m_Ver1[j] = m_L[j][i]; // 열 방향으로 1차원 배열에 할당
      m_Ver2[j] = m_H[j][i];
   }

   m_Conv1 = OnConvolution(m_Ver1, m_FilterH0, height, 1);
   // Convolution 처리
   m_Conv2
      = OnConvolution(m_Ver1, m_FilterH1, height, 1);
   m_Conv3
      = OnConvolution(m_Ver2, m_FilterH0, height, 1);
   m_Conv4
      = OnConvolution(m_Ver2, m_FilterH1, height, 1);

   m_Down1 = OnDownSampling(m_Conv1, height); // 다운 샘플링
   m_Down2 = OnDownSampling(m_Conv2, height);
   m_Down3 = OnDownSampling(m_Conv3, height);
   m_Down4 = OnDownSampling(m_Conv4, height);

   for(j=0 ; j<height/2 ; j++){
      m_LL[j][i] = m_Down1[j]; // 결과 저장
      m_LH[j][i] = m_Down2[j];
      m_HL[j][i] = m_Down3[j];
      m_HH[j][i] = m_Down4[j];
   }
}
```

18 OnWaveletEncode 함수 작성(6)

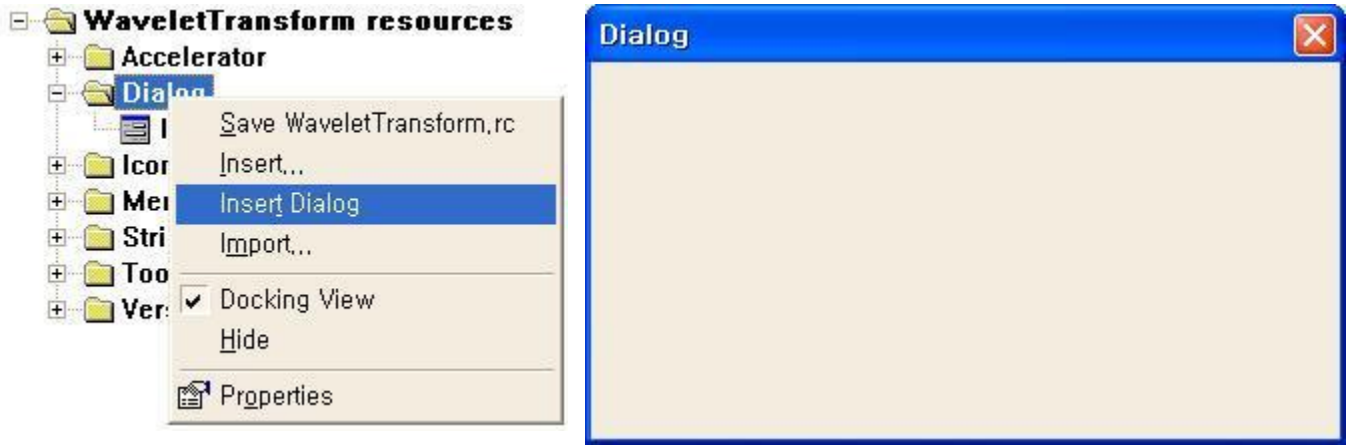
```
m_SLL = OnScale(m_LL, height/2, width/2); // 처리 결과를 정규화
m_SLH = OnScale(m_LH, height/2, width/2);
m_SHL = OnScale(m_HL, height/2, width/2);
m_SHH = OnScale(m_HH, height/2, width/2);

for(i=0 ; i<height/2 ; i++){
   for(j=0 ; j<width/2 ; j++){
      m_tempOutput[i][j] = m_LL[i][j];
      m_tempOutput[i][j+(width/2)] = m_HL[i][j];
      m_tempOutput[i+(height/2)][j] = m_LH[i][j];
      m_tempOutput[i+(height/2)][j+(width/2)] =
         m_HH[i][j];
      // 처리 결과를 정렬
      m_ArrangeImage[i][j]
         = (unsigned char)m_SLL[i][j];
      m_ArrangeImage[i][j+(width/2)]
         = (unsigned char)m_SHL[i][j];
      m_ArrangeImage[i+(height/2)][j]
         = (unsigned char)m_SLH[i][j];
      m_ArrangeImage[i+(height/2)][j+(width/2)]
         = (unsigned char)m_SHH[i][j];
      // 정규화 과정을 거친 정렬 영상
      }
   }
   width = width / 2;
   // 분해를 계속하기 위해 영상의 가로축 크기를 반으로 줄임
   height = height / 2;
   // 분해를 계속하기 위해 영상의 세로축 크기를 반으로 줄임
```

⑱ OnWaveletEncode 함수 작성(7)

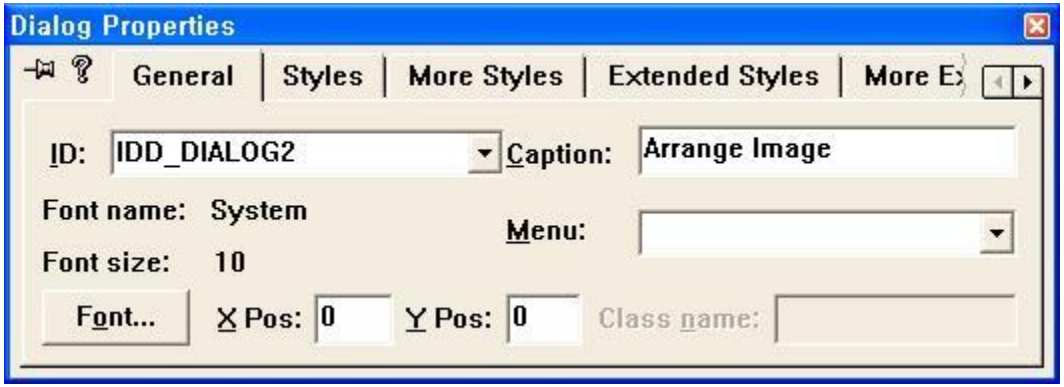
```
        m_tempInput = OnMem2DAlloc   double(height, width);

        for(i=0 ; i<height ; i++){
           for(j=0 ; j<width ; j++){
              m_tempInput[i][j] = m_LL[i][j];
              // LL 값을 새로운 입력으로 할당
           }
        }
     }

     delete [] m_Conv1, m_Conv2, m_Conv3, m_Conv4;
     delete [] m_Down1, m_Down2, m_Down3, m_Down4;
     delete [] m_Hor, m_Ver1, m_Ver2;
     for(i=0 ; i<height ; i++){ // 메모리 해제
        delete[] m_LL[i];
        delete[] m_LH[i];
        delete[] m_HL[i];
        delete[] m_HH[i];
        delete[] m_SLL[i];
        delete[] m_SLH[i];
        delete[] m_SHL[i];
        delete[] m_SHH[i];
        delete[] m_L[i];
        delete[] m_H[i];
     }
     delete m_L, m_H, m_LL, m_LH, m_HL, m_HH, m_SLL, m_SLH,
        m_SHL, m_SHH;

     UpdateAllViews(NULL);
}
```

정렬 영상 출력

❶ [Resource View]창의 [WaveletTransform resources-Dialog] 폴더에서 마우스 오른쪽 버튼 클릭→ [Insert Dialog] 클릭해 대화상자 하나 삽입([OK], [Cancel] 버튼은 삭제)



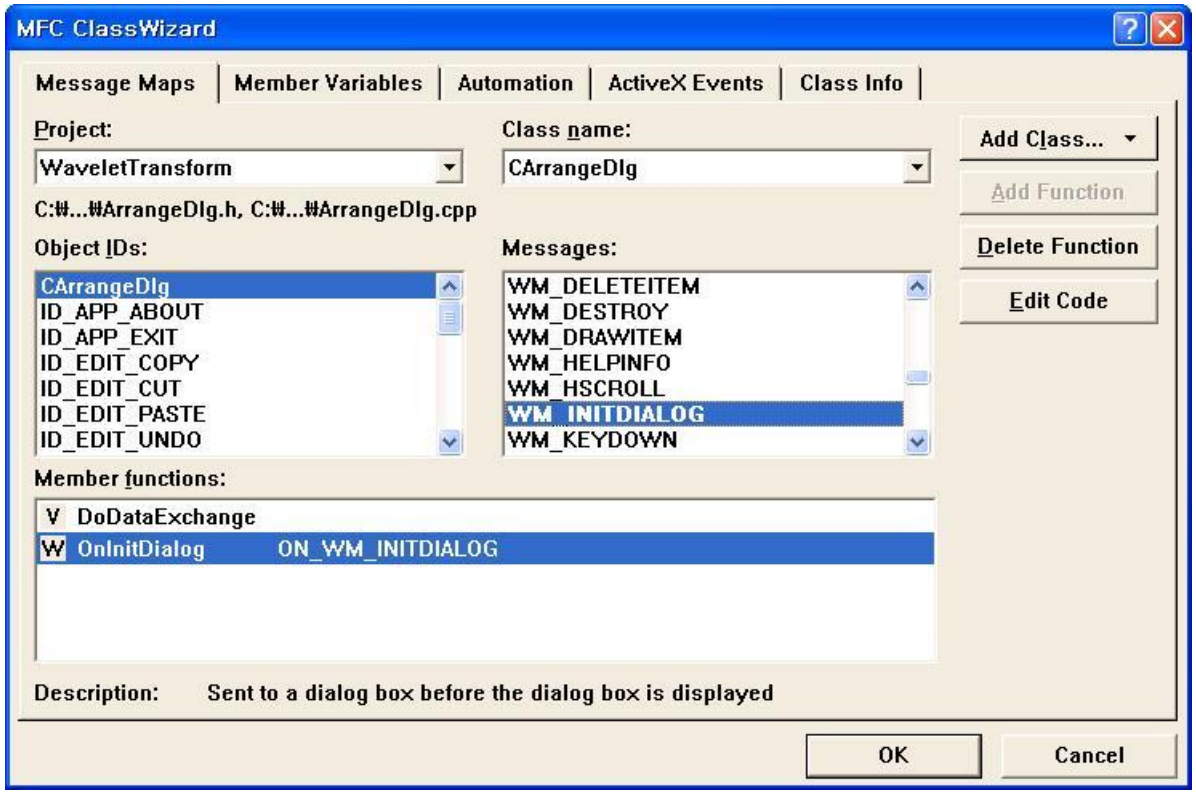
→[Dialog Properties] 대화상자에서 캡션 Arrange Image로 설정



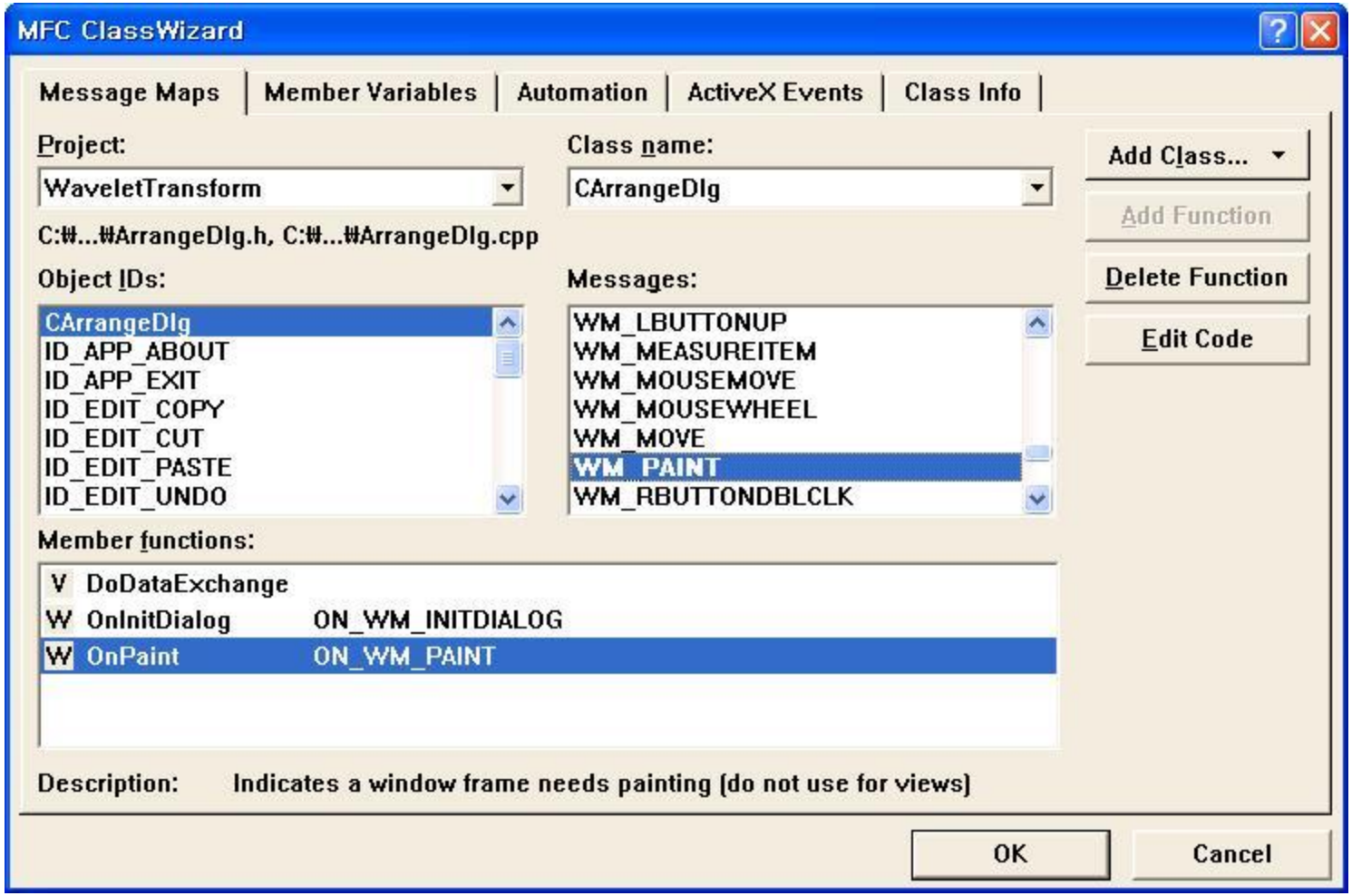
❷ 대화상자를 클래스에 추가

Class name	CArrangeDlg

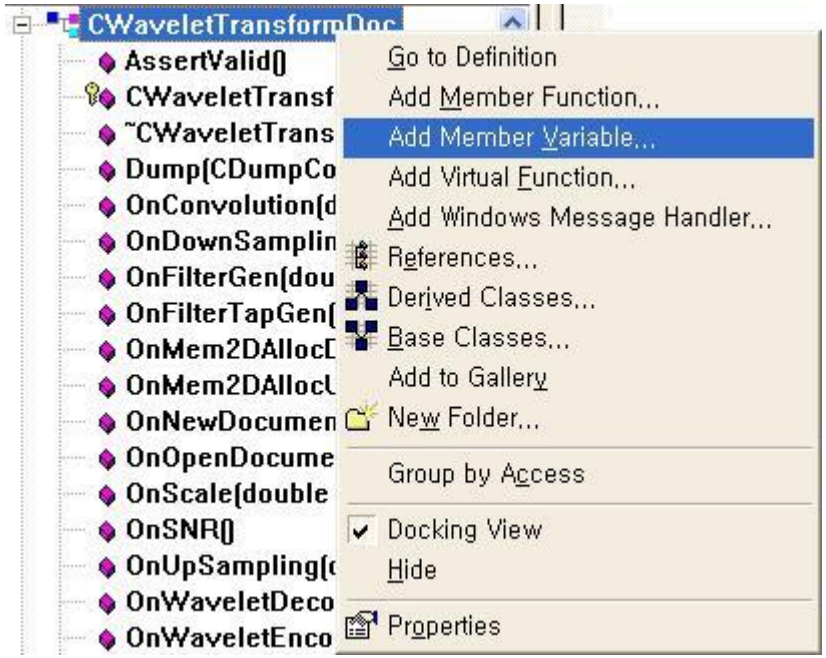
❸ **[MFC ClassWizard] 대화상자의 [Message Maps] 탭에서 Messages 항목의 WM_INITDIALOG를 클릭하여 OnInitDialog 함수 추가. 대화상자를 초기화하는 역할을 함.**



❹ **Messages** 항목의 **WM_PAINT**지를 클릭하여 **OnPain** 함수 추가. 이곳은 대화상자에 그림을 그리는 역할 수행



❺ CWaveletTransformDoc에서 CArrangeDlg 클래스에 출력할 영상과 크기를 저장하는 변수 선언



Variable Type	Variable Name	Access	설명
unsigned char	**m_tempImage	Public	입력 영상 버퍼
int	Height	Public	영상의 세로축 크기
int	Width	Public	영상의 가로축 크기

❻ 생성된 함수에 다음 프로그램 추가

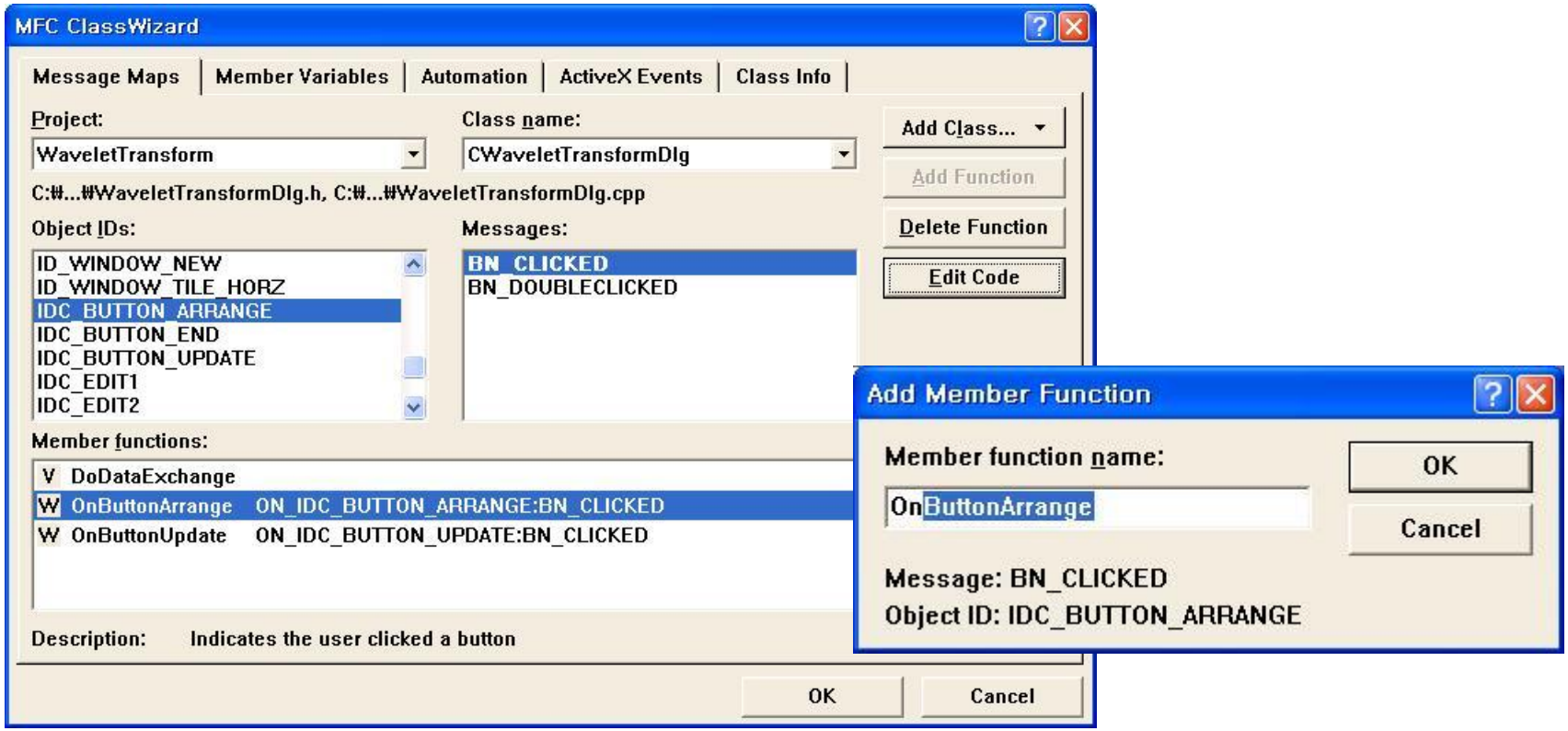
① OnInitDialog 함수 추가하기

```
BOOL CArrangeDlg::OnInitDialog()
{
   CDialog::OnInitDialog();

   CRect rect, rectC;
   GetWindowRect(&rect);
   GetClientRect(&rectC);

   int cx, cy;

   CSize sizeImg;
   sizeImg.cx = Width;
   sizeImg.cy = Height;

   cx = sizeImg.cx + rect.Width() - rectC.Width() + 4;
   // 정렬 영상 출력을 위한 대화상자 크기 조절
   cy = sizeImg.cy + rect.Height() - rectC.Height() + 4;

   SetWindowPos(this, 0, 0, cx, cy, SWP_NOZORDER);

   return TRUE;
   // return TRUE unless you set the focus to a control
   // EXCEPTION: OCX Property Pages should    return FALSE
}
```

② OnPaint 함수 추가하기

```
void CArrangeDlg::OnPaint()
{
   CPaintDC dc(this); // device context for painting
   // TODO: Add your message handler code here
   // Do not call CDialog::OnPaint() for painting messages
   int i, j;
   unsigned char R, G, B;

   for(i=0 ; i<Height ; i++){
      for(j=0 ; j<Width ; j++){
         R = m_tempImage[i][j];

         G = B = R;
         dc.SetPixel(j, i, RGB(R, G, B));
         // 정규화된 정렬 영상을 화면에 출력
      }
   }
}
```

❼ 정렬된 영상은 **Wavelet Transform(IDD_DIALOG1)** 대화상자에서 **[Arrange Image]** 버튼을 클릭하면 영상이 출력되게 함. **[MFC ClassWizard]** 대화상자에서 **[Arrange Image]** 버튼에 함수 추가



Class name	CWaveletTransformDlg
Member function name	OnButtonArrange

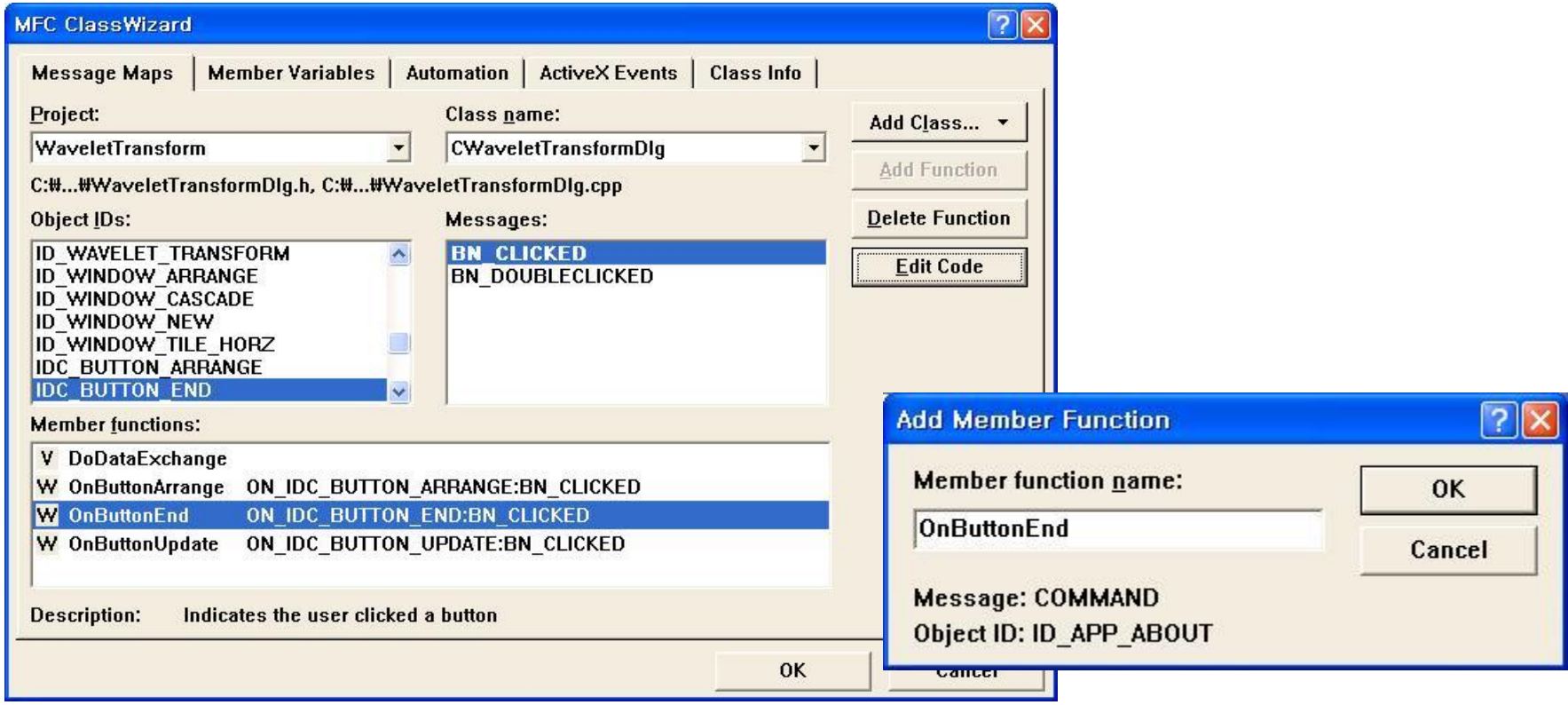
❽ ArrangeDlg를 사용하기 위해 CWaveletTransformDlg.cpp 파일 위쪽에 ArrangeDlg.h 선언

```
#include "stdafx.h"
#include "WaveletTransform.h"
#include "WaveletTransformDlg.h"

#include "WaveletTransformDoc.h"

#include "ArrangeDlg.h"
```

❾ OnButtonArrange 함수 추가

```
void CWaveletTransformDlg::OnButtonArrange()
{
    // 버튼을 누르면 정렬 영상 출력
    CArrangeDlg dlg;

    dlg.Width = m_pDoc->m_Width;
    dlg.Height = m_pDoc->m_Height;
    dlg.m_tempImage = m_pDoc->m_ArrangeImage;

    UpdateData(TRUE);

    dlg.DoModal(); // 정렬 영상을 위한 대화상자 출력
}
```

⑩ **Wavelet Transform(IDD_DIALOG1) 대화상자의 [종료] 버튼에 변환을 종료하는 프로그램 추가**



Class name	CWaveletTransformDlg
Member function name	OnButtonEnd

⓫ OnButtonEnd 함수 추가

```
void CWaveletTransformDlg::OnButtonEnd()
{
    CDialog::OnOK();
}
```

Section 06 역방향 웨이브렛 변환 구현하기

❶ 역방향 웨이브렛 변환은 순방향 변환이 일어난 뒤 변환된 값을 사용하여 역방향 변환을 하게 됨. 따라서 순방향에서 사용되었던 함수와 대화상자가 재사용됨.

❷ **CWaveletTransformDlg::OnButtonArrange** 함수에 다음 프로그램 추가

```
void CWaveletTransformDlg::OnButtonUpdate()
{
    // TODO: Add your control notification handler code here
    UpdateData(TRUE);
    m_pDoc->m_Level = m_Level;

    m_pDoc->OnWaveletEncode();

    m_pDoc->OnWaveletDecode(); // 웨이브렛 역변환
    m_pDoc->OnSNR(); // 신호 대 잡음비

    UpdateData(FALSE);
}
```

Section 06 역방향 웨이브렛 변환 구현하기(계속)

❸ 순방향 웨이브렛 변환처럼 **CWaveletTransformDoc** 클래스에 **OnWaveletDecode** 함수를 추가하여 역방향 웨이브렛 변환을 수행할 수 있도록 한. 역방향 웨이브렛 변환하는 데 필요한 함수는 **CWaveletTransformDoc** 클래스에 추가

함수 타입	함수명	설명
void	OnWaveletDecode	
double*	OnUpSampling(double *m_Target, int size)	업 샘플링 함수
void	OnSNR( )	신호 대 잡음비

① OnUpSampling 함수 추가하기

```
double* CWaveletTransformDoc::OnUpSampling(double *m_Target,
    int size)
{ // 업 샘플링을 위한 함수
    int i;
    double* m_temp;

    m_temp = new double[size * 2];

    for(i=0 ; i<size*2 ; i++)
        m_temp[i] = 0.0; //초기화

    for(i=0 ; i<size ; i++)
        m_temp[2*i] = m_Target[i]; // 업 샘플링 처리

    return m_temp;
}
```

Section 06 역방향 웨이브렛 변환 구현하기(계속)

② OnSNR 함수 추가하기

```
void CWaveletTransformDoc::OnSNR()
{ // 신호 대 잡음비를 위한 함수
   double OrgSum, ErrSum, MeanErr, MeanOrg;
   int i;

   OrgSum = 0.0;
   ErrSum = 0.0;

   // calculate mean squared error
   for(i=0 ; i<m_Size ; i++){
      // 오류의 에너지 계산
      ErrSum += ((double) m_InputImage[i] - m_Recon[i]) *
         ((double) m_InputImage[i] - m_Recon[i]);
   }
   MeanErr = ErrSum / m_Size; // 오류 에너지 평균

   for(i=0 ; i<m_Size ; i++){
      // 신호의 에너지 계산
      OrgSum += ((double) m_InputImage[i]) *
         ((double) m_InputImage[i]);
   }
   MeanOrg = OrgSum / m_Size; // 신호 에너지 평균

   pDlg->m_Error = (float)MeanErr; // 오류 출력
   pDlg->m_SNR = (float)(10 * (double)log10(MeanOrg / MeanErr));
   // 신호 대 잡음비 계산
}
```

❹ OnWaveletDecode 함수 작성(1)

```
void CWaveletTransformDoc::OnWaveletDecode()
{
   int i, j, k;
   int width, height;
   double *tempLL, *tempLH, *tempHL, *tempHH, *tempL, *tempH;
   double **L, **H;
   double *Up1, *Up2, *Up3, *Up4;
   double *Conv1, *Conv2, *Conv3, *Conv4;
   double **R;

   width= m_Width / (int)(pow(2, m_Level));
   height= m_Height / (int)(pow(2, m_Level));

   m_Recon = new double [m_Width * m_Height];

   for(k=m_Level ; k>0 ; k--){
      if(width >    m_Width || height >m_Height){ // 분해 종료
         return;
      }

      tempLL = new double [height];
      tempLH = new double [height];
      tempHL = new double [height];
      tempHH = new double [height];
```

❹ OnWaveletDecode 함수 작성(2)

```
L = OnMem2DAlloc   double(height*2, width);
H = OnMem2DAlloc   double(height*2, width);

tempL = new double [width];
tempH = new double [width];

R = OnMem2DAlloc   double(height*2, width*2);

for(i=0 ; i<width ; i++){
   for(j=0 ; j<height ; j++){
   // 정렬 영상에서 처리하려는 열을 분리
      tempLL[j] = m_tempOutput[j][i];
      tempLH[j] = m_tempOutput[j + height][i];
      tempHL[j] = m_tempOutput[j][i + width];
      tempHH[j] = m_tempOutput[j + height][i + width];
   }

   Up1 = OnUpSampling(tempLL, height); // 업 샘플링
   Up2 = OnUpSampling(tempLH, height);
   Up3 = OnUpSampling(tempHL, height);
   Up4 = OnUpSampling(tempHH, height);

   Conv1 = OnConvolution(Up1, m_FilterG0, height*2, 2);
   // 컨벌루션 연산
   Conv2 = OnConvolution(Up2, m_FilterG1, height*2, 2);
   Conv3 = OnConvolution(Up3, m_FilterG0, height*2, 2);
   Conv4 = OnConvolution(Up4, m_FilterG1, height*2, 2);
```

❹ OnWaveletDecode 함수 작성(3)

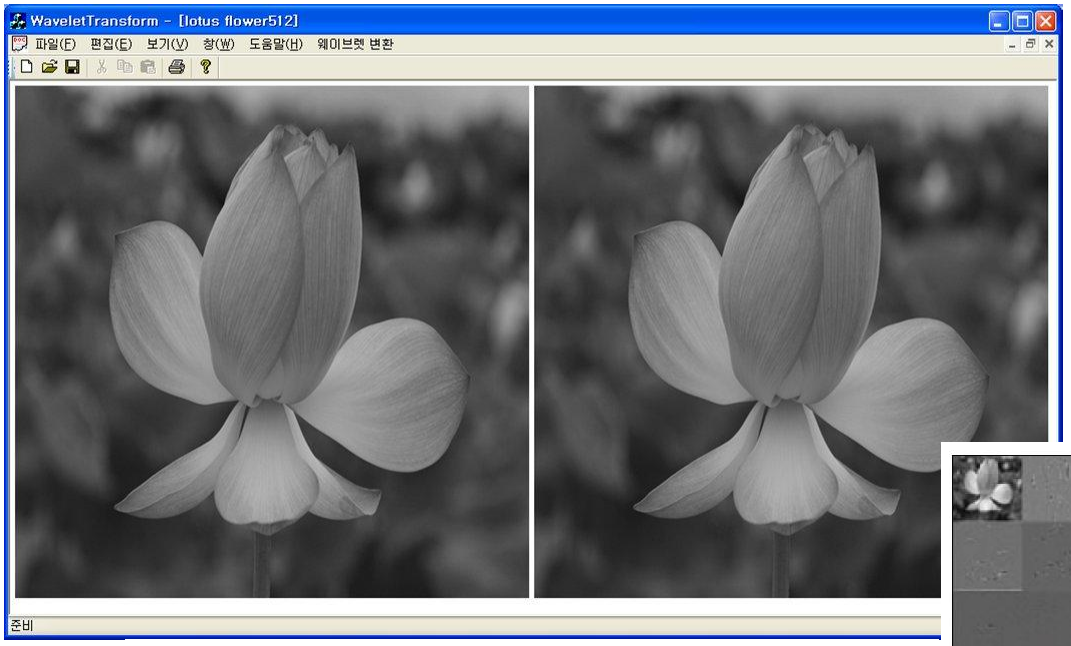
```
        for(j=0 ; j<height*2 ; j++){
           L[j][i] = Conv1[j] + Conv2[j];
           H[j][i] = Conv3[j] + Conv4[j];
        }
     }
     for(i=0 ; i<height*2 ; i++){
        for(j=0 ; j<width ; j++){
           tempL[j] = L[i][j]; // 횡 데이터 분리
           tempH[j] = H[i][j];
        }

        Up1 = OnUpSampling(tempL, width); // 업 샘플링
        Up2 = OnUpSampling(tempH, width);

        Conv1 = OnConvolution(Up1, m_FilterG0, width*2, 2);
        // 컨벌루션 연산
        Conv2 = OnConvolution(Up2, m_FilterG1, width*2, 2);
        for(j=0 ; j<width*2 ; j++){
          R[i][j] = Conv1[j] + Conv2[j];
        }
    }

    for(i=0 ; i<height*2 ; i++){
        for(j=0 ; j<width*2 ; j++){
           m_tempOutput[i][j] = R[i][j];
           // 복원 데이터를 다시 정렬
        }
    }
```

❹ OnWaveletDecode 함수 작성(4)

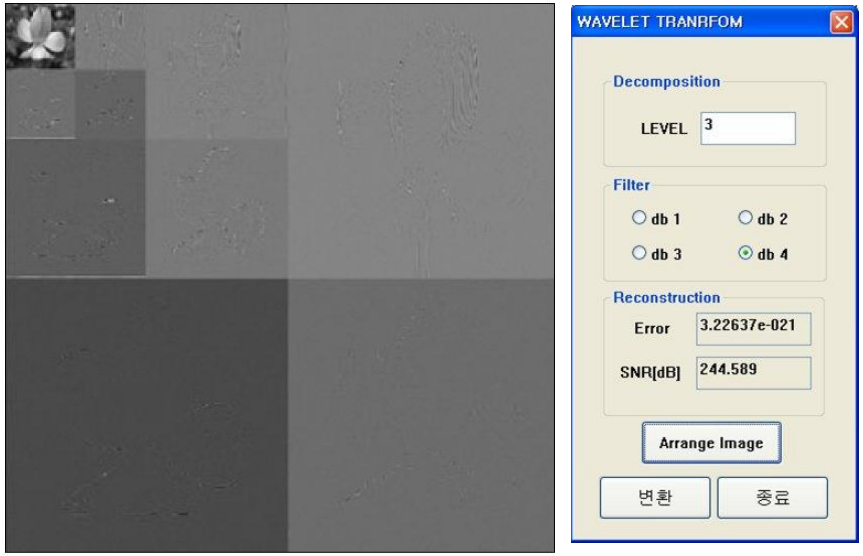
```
      height = height * 2; // 영상의 크기를 두 배 확장
      width = width * 2;
   }

   for(i=0 ; i<   m_Height ; i++){
      for(j=0 ; j<   m_Width ; j++){
         m_Recon[i*   m_Width + j] = R[i][j];
         m_OutputImage[i*   m_Width + j]
            = (unsigned char)R[i][j];
         // 최종 복원된 결과를 출력
      }
   }

   UpdateAllViews(NULL);

   delete [] tempLL, tempLH, tempHL, tempHH, tempL, tempH;
   // 메모리 해제
   delete [] Up1, Up2, Up3, Up4;
   delete [] Conv1, Conv2, Conv3, Conv4;

   for(i=0 ; i<   m_Height ; i++){ // 메모리 해제
      delete[] L[i];
   delete[] H[i];
   delete[] R[i];
   }

   delete L, H, R;
}
```

Section 06 역방향 웨이브렛 변환 구현하기(계속)

❺ 결과 영상



(a) 입력 영상 (b) 복원 영상



(c) 3단계 분해된 영상

2차원 웨이브렛 변환 결과

Thank you