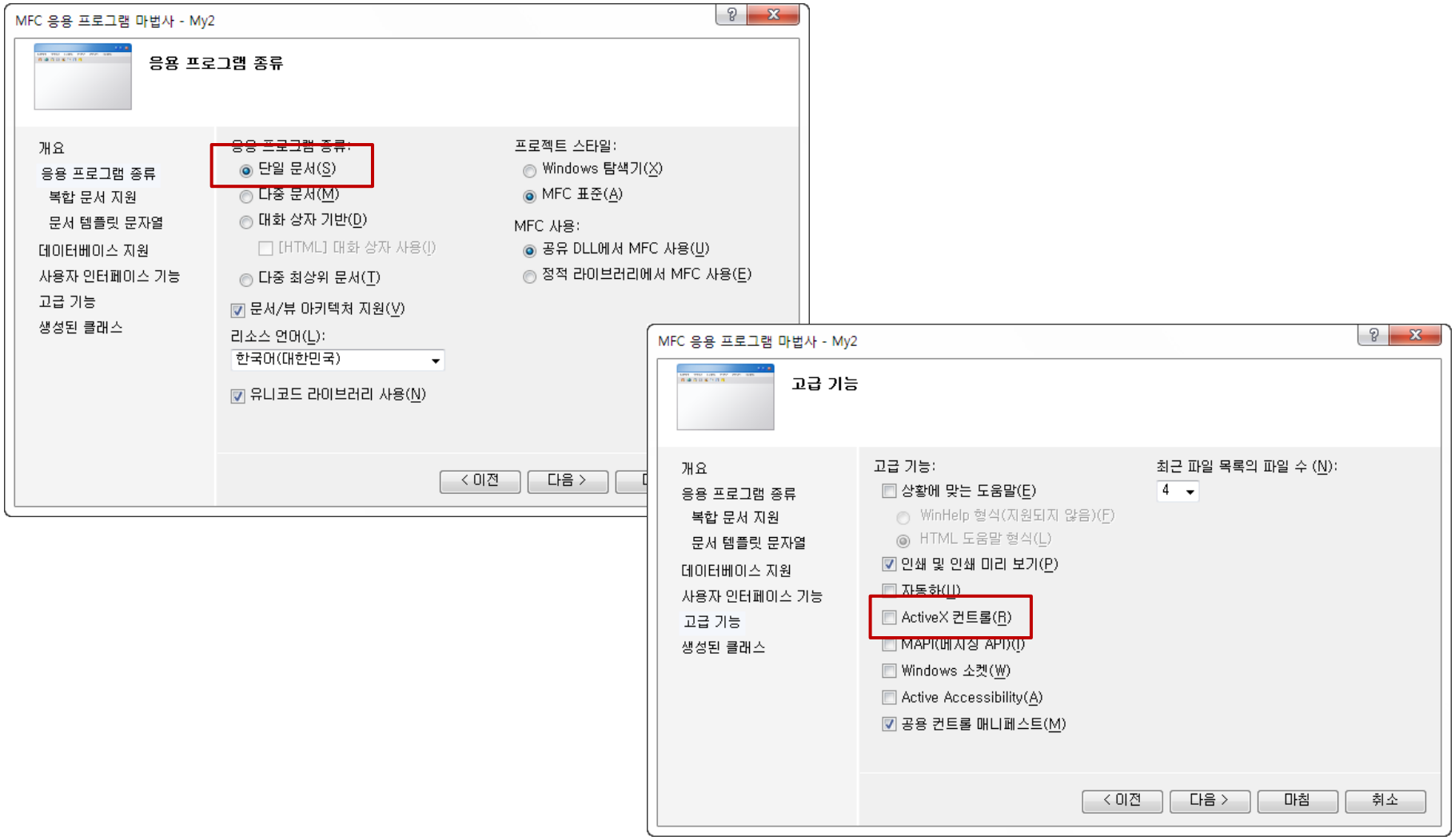
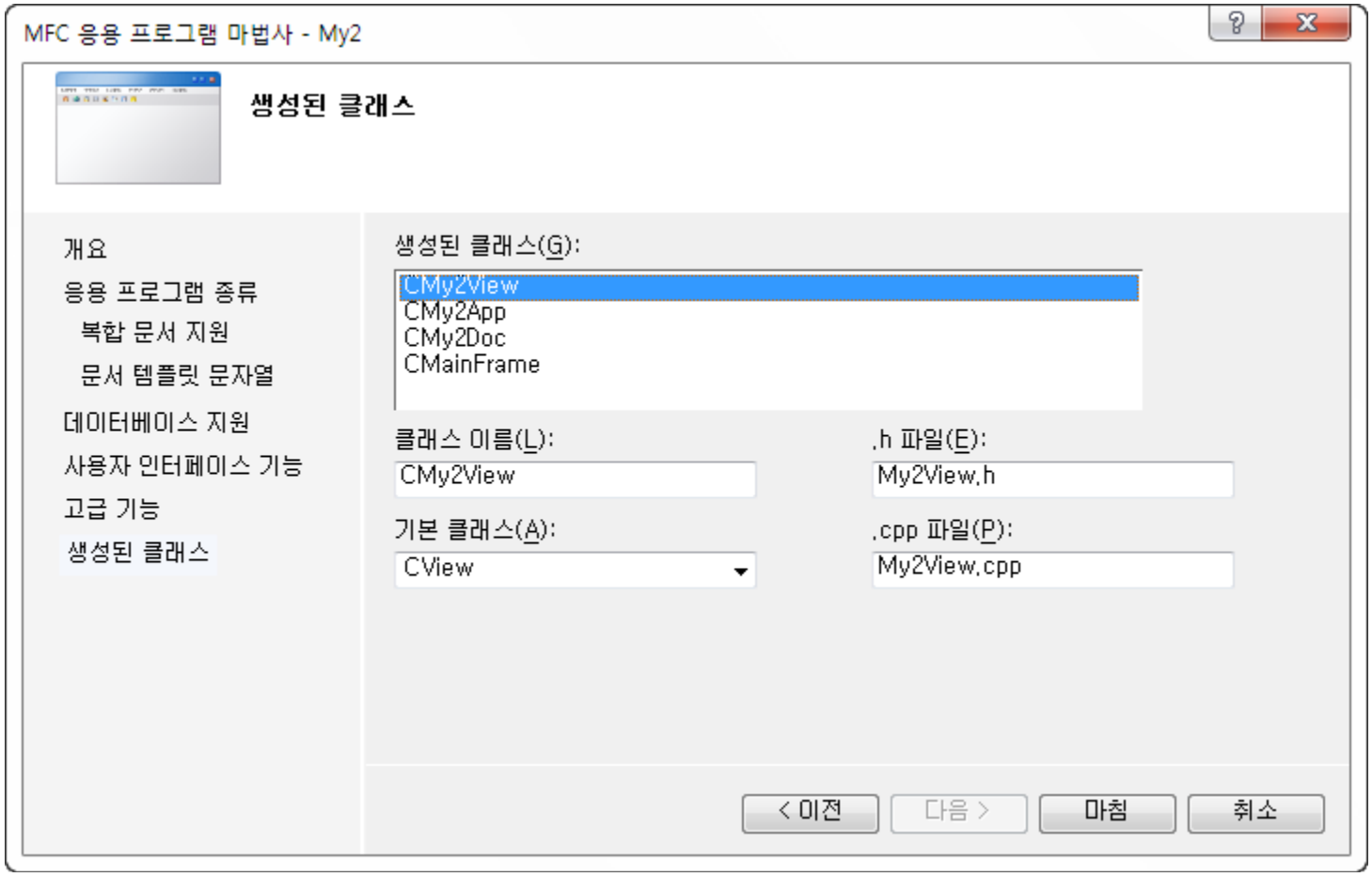
6장
MFC 메시지 처리

김성영교수
금오공과대학교
컴퓨터공학부

마법사를 사용한 MFC 응용 프로그램 생성 – 1



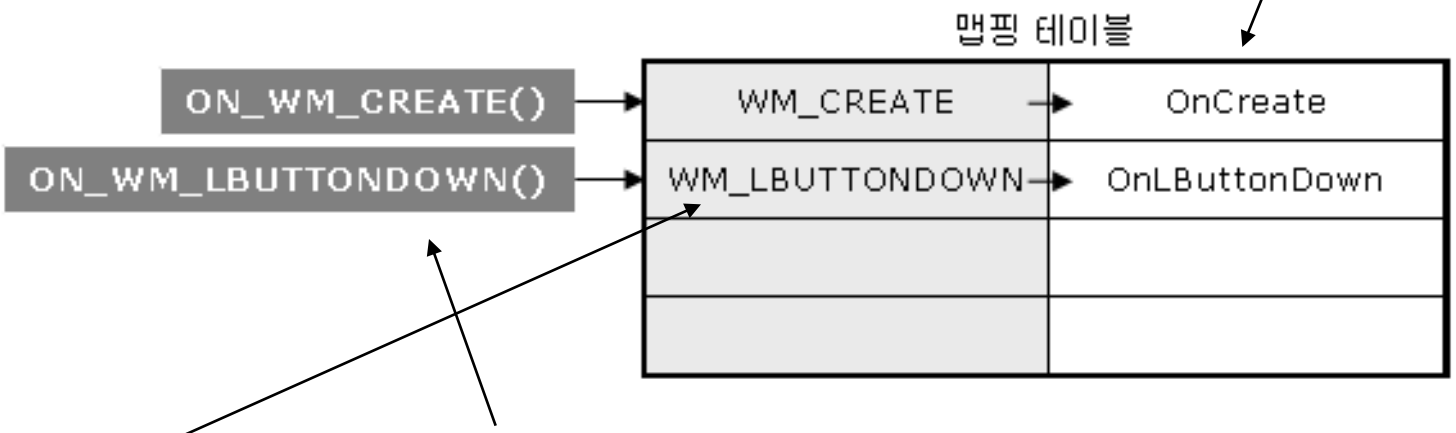
마법사를 사용한 MFC 응용 프로그램 생성 – 2



메시지 맵 관련 매크로

- 메시지 맵 – 메시지와 메시지 핸들러를 매핑하는 테이블
- MFC에서 제공하는 매핑 매크로(3가지 유형)

- 첫 번째 유형



메시지에 대한 매핑 매크로를 알면 메시지 처리 가능

- **매핑 하려면 먼저 매핑 테이블 있어야 되지**

구현파일(.CPP)

```
BEGIN_MESSAGE_MAP(A,B)   //A는 부모 클래스 명, B는 파생  클래스 명
        ON_WM_CREATE()
        ON_WM_LBUTTONDOWN()
END_MESSAGE_MAP()
```

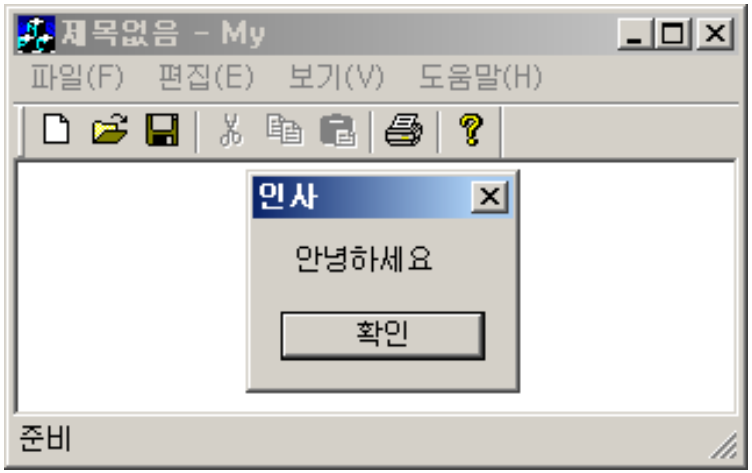
정의파일(.H)

```
afx_msg void OnCreate()(LPCREATESTRUCT);    // 메시지 핸들러 prototype
afx_msg void OnLButtonDown(UINT, CPoint);    // 메시지 핸들러 prototype
DECLARE_MESSAGE_MAP()          // 매핑 테이블 마련
```

메시지처리 유형 1

실습 6.1

마우스 왼쪽 버튼 눌림 **이벤트 처리**를 해보자.



어느 클래스에서 처리?

MyView.h

```
// MyView.h : interface of the CMyView class
//
class CMyView : public CView
{
…
// Implementation
public:
        virtual ~CMyView();
#ifdef _DEBUG
        virtual void AssertValid() const;
        virtual void Dump(CDumpContext& dc) const;
#endif
protected:
// Generated message map functions
protected:
        //{{AFX_MSG(CMyView)
        //}}AFX_MSG
        afx_msg void OnLButtonDown( UINT, CPoint );      // 추가한 부분
        DECLARE_MESSAGE_MAP()
};
```

MyView.cpp

```
////////////////////////////////////////////////////////////////////////////
// CMyView

IMPLEMENT_DYNCREATE(CMyView, CView)

BEGIN_MESSAGE_MAP(CMyView, CView)
        // Standard printing commands
        ON_COMMAND(ID_FILE_PRINT, CView::OnFilePrint)
        ON_COMMAND(ID_FILE_PRINT_DIRECT, CView::OnFilePrint)
        ON_COMMAND(ID_FILE_PRINT_PREVIEW, CView::OnFilePrintPreview)
        ON_WM_LBUTTONDOWN()
END_MESSAGE_MAP()
```

MyView.cpp

```
////////////////////////////////////////////////////////////////////////////
// CMyView message handlers
void CMyView::OnLButtonDown( UINT nFlags, CPoint point )
{
        ::MessageBox( ___①___, "안녕하세요", "인사", MB_OK );
        _____②_____;
}
```

①에 입력해야하는 코드는?

②에 입력해야하는 코드는?

실습 6.2

마우스 왼쪽 버튼 클릭 시 메시지 박스 대신 마우스의 좌표를 출력하자.

MyView.cpp

```
void CMyView::OnLButtonDown( UINT nFlags, CPoint point )
{




        CView::OnLButtonDown( nFlags, point );
}
```

문자열 클래스 CString 을 이용한 문자열 처리

MyView.cpp

```
void CMyView::OnLButtonDown( UINT nFlags, CPoint point )
{




        CView::OnLButtonDown( nFlags, point );
}
```

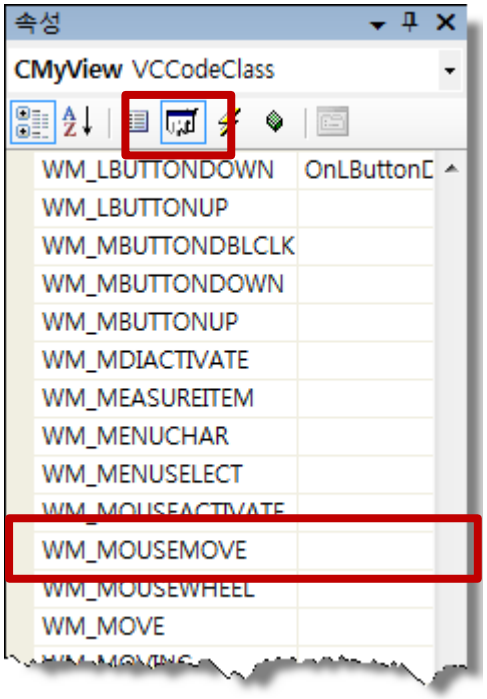
실습 6.3

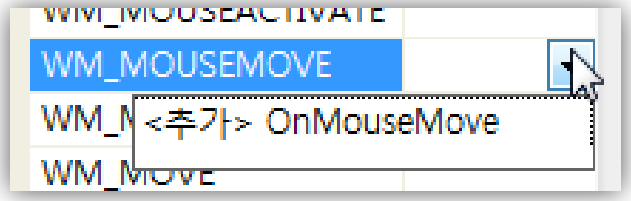
마우스를 움직일 때마다 마우스의 좌표를 출력하자.

- ON_WM_MOUSEMOVE MSDN 검색
- 정의파일(.h)
  - afx_msg void OnMoudseMove(UINT, CPoint) 프로토타입 삽입
- 구현파일(.cpp)
  - ON_WM_MOUSEMOVE 매핑 매크로 삽입
  - OnMouseMove() 메시지 핸들러 구현

Visual Studio 위저드를 이용한 메시지 핸들러 등록

- 속성 창 선택
- 클래스 뷰에서 CMyView 클래스 선택
- WM_MOUSEMOVE 선택 후에 메시지 핸들러 등록





MyView.h

```
afx_msg void OnMouseMove( UINT nFlags, CPoint point );
DECLARE_MESSAGE_MAP()
```

MyView.cpp

```
BEGIN_MESSAGE_MAP(CMyView, CView)
        . . .
        ON_WM_LBUTTONDOWN()
        ON_WM_MOUSEMOVE()
END_MESSAGE_MAP()
. . .
void CMyView::OnMouseMove( UINT nFlags, CPoint point )
{
        // TODO: 여기에 메시지 처리기 코드를 추가 및/또는 기본값을 호출합니다.

        CView::OnMouseMove(nFlags, point);
}
```

HDC 대신 이에 대한 클래스인 CDC 클래스와 관련 함수 사용

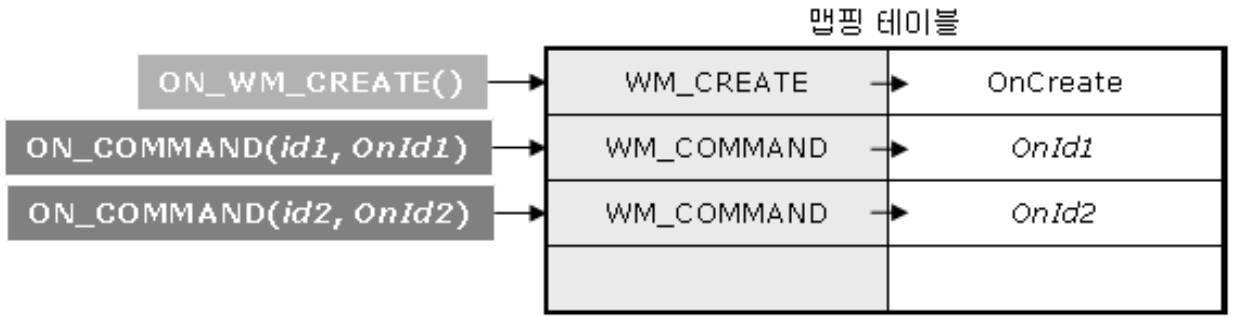
MyView.cpp

```
void CMyView::OnMouseMove( UINT nFlags, CPoint point )
{
	CString strPos;
	strPos.Format( "%04d %04d", point.x, point.y );

	CDC* pDC;
	pDC = this->GetDC( );		// this는 CMyView 객체를 가리킴 (생략 가능)
	pDC->TextOut( 0, 0, strPos );	// CDC 객체의 멤버함수 TextOut 함수 이용
	this->ReleaseDC( pDC );		// CDC 객체 반환

	CView::OnMouseMove( nFlags, point );
}
```

- GetDC( ), ReleaseDC( )는 첫 번째 인자로 HWND요구 → CWnd의 멤버함수
- TextOut( )함수는 첫 번째 인자로 HDC요구 → CDC의 멤버함수

메시지처리 유형 2

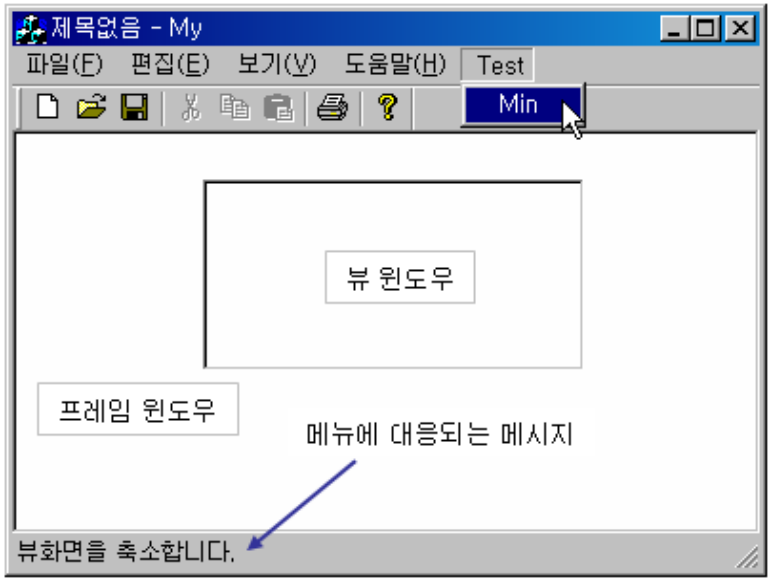
- WM_COMMAND 메시지에 대한 처리
  - 메뉴항목, 단축 키, 컨트롤 윈도우의 이벤트에 대응하여 발생
- 매핑 매크로에 이벤트를 발생시키는 주체와 대응하는 메시지 핸들러 함께 지정



- **id1**: 이벤트 발생 주체의 식별자
- **OnId1**: 사용자가 정한 핸들러 이름

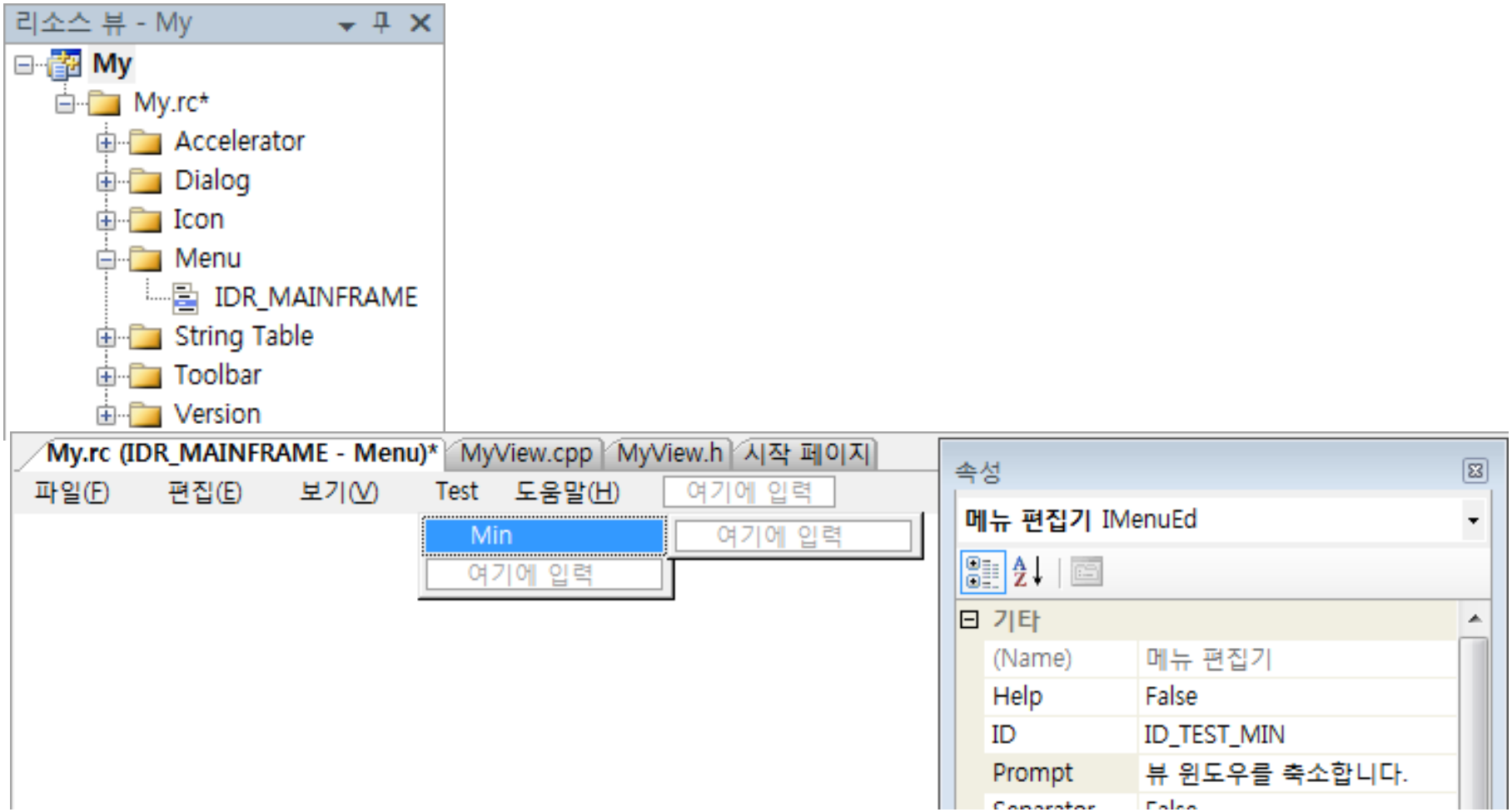
실습 6.4

메뉴를 클릭하여 뷰 윈도우를 축소하여 화면의 중앙으로 옮기자.

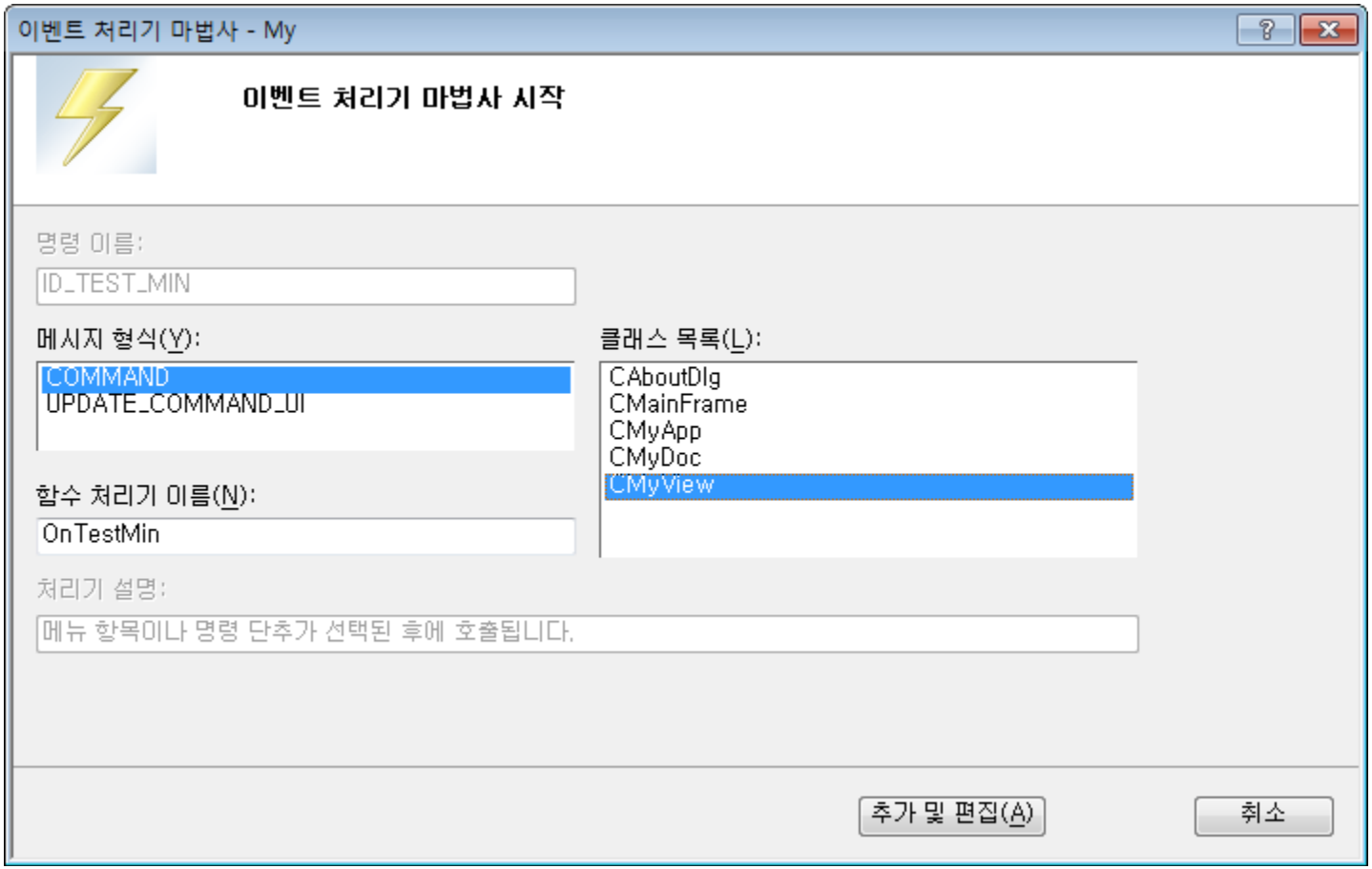


- 메뉴, 툴바, 상태바를 포함하고 있는 윈도우는 CMainFrame 클래스 객체로부터 생성한 프레임 윈도우
- 화면가운데 사각형 윈도우는 CMyView 클래스 객체로부터 생성한 뷰 윈도우

메뉴 항목 추가


리소스 뷰 - My
My
My.rc*
Accelerator
Dialog
Icon
Menu
IDR_MAINFRAME
String Table
Toolbar
Version
My.rc (IDR_MAINFRAME - Menu)*
MyView.cpp
MyView.h
시작 페이지
파일(F)
편집(E)
보기(V)
Test
도움말(H)
여기에 입력
Min
여기에 입력
여기에 입력
속성
메뉴 편집기 IMenuEd
기타
(Name)
메뉴 편집기
Help
False
ID
ID_TEST_MIN
Prompt
뷰 윈도우를 축소합니다.

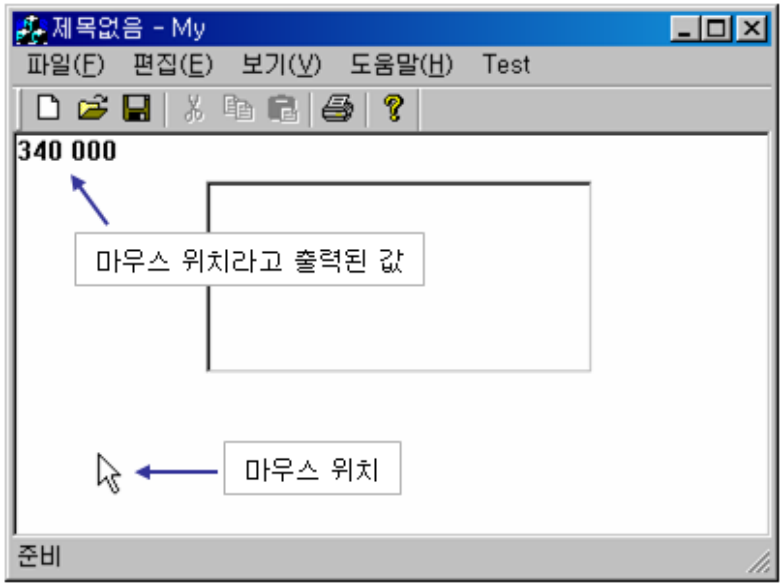
이벤트 처리기 (메시지 핸들러) 추가

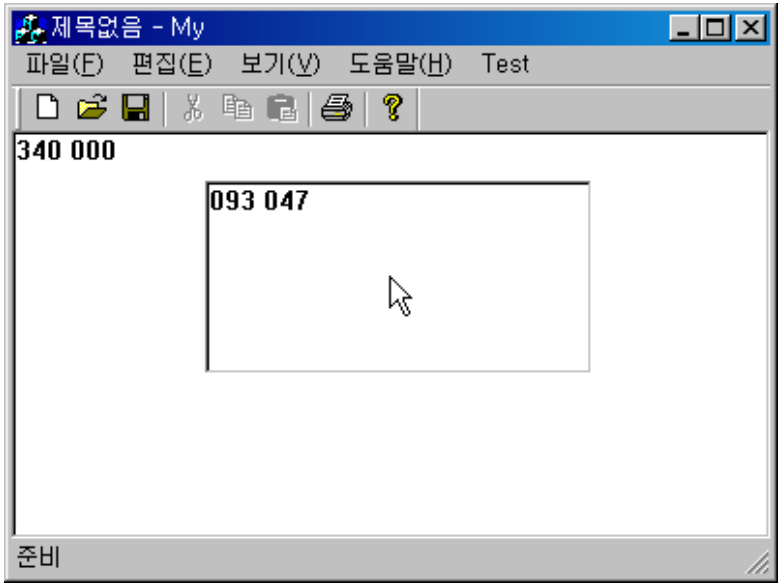


이벤트 처리기 (메시지 핸들러) 코드 입력

MyView.cpp

```
void CMyView::OnTestMin()
{
        this->MoveWindow( 100, 50, 200, 100 ); // this 생략 가능
/*
        CReat rt;
        rt.left = 100;  rt.top = 100;  rt.right = 200;  rt.bottom = 200;
        MoveWindow( rt );
*/
}
```





- WM_MOUSEMOVE 이벤트를 CMyView 클래스에서만 처리
  → 뷰 윈도우 안에서만 마우스 움직임에 대한 처리
- CMainFrame 클래스에 동일 핸들러 구현
  → 프레임 윈도우에서도 같은 처리
- 프레임 윈도우가 화면 갱신을 하지 않는 이상 출력된 것은 그대로 남아있음

실습 6.5

줄어든 뷰 윈도우를 다시 원상복구하자.

- 메뉴에 Max 메뉴 항목 추가하고 이에 대한 처리
- MoveWindow() 함수 호출
- 윈도우의 크기는 프레임 윈도우의 클라이언트 영역크기로 설정
  - 윈도우의 클라이언트 영역 크기를 알아오는 함수

BOOL GetClientRect(HWND *hWnd*, LPRECT *lpRect*);

MyView.cpp

```
void CMyView::OnTestMax( )
{
        CRect rt;
        GetClientRect( &rt );
        MoveWindow( rt );
}
```

- **뷰 윈도우의 클라이언트 영역 크기를 알아내고 움직이도록 하므로 변화 없음**
- **this -> 생략, 여기서 this는 CMyView클래스의 객체 포인터**

MyView.cpp

```
void CMyView::OnTestMax()
{
        CRect rt;
        CMainFrame* pFrame;

        pFrame = _____;        //프레임 윈도우 객체의 포인터 할당 필요

        pFrame->GetClientRect( &rt );
        this->MoveWindow(rt);
}
```

```
CMyApp  theApp;   // my.h 파일에 전역변수로 선언됨
```

- **애플리케이션 객체 theApp는 전역변수 이므로 어디서든 사용가능**

MyView.cpp

```
void CMyView::OnTestMax( )
{
        CRect rt;
        CWnd* pFrame;  // CMainFrame을 CWnd로 대치
        pFrame = theApp.m_pMainWnd;
        pFrame->GetClientRect( &rt );
        this->MoveWindow( rt );
}
```

- **theApp가 다른 파일에서 선언 되었기 때문에 에러 발생**

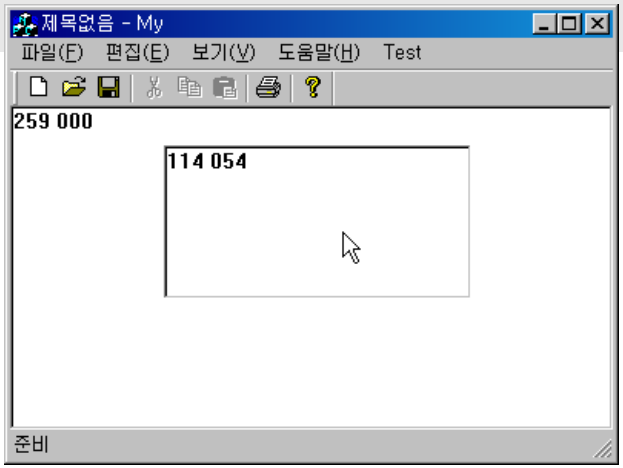
- theApp의 이름이 바뀌었을 경우 다음의 전역함수 사용

```
CWinApp* AfxGetApp( );                  // 어플리케이션 객체의 주소 반환
CWnd* AfxGetMainWnd( );                 // 프레임 윈도우 객체의 주소 반환
HINSTANCE AfxGetInstanceHandle( );      // 인스턴스 핸들 반환
```

- **전역함수를 써서 다음과 같이 변경**

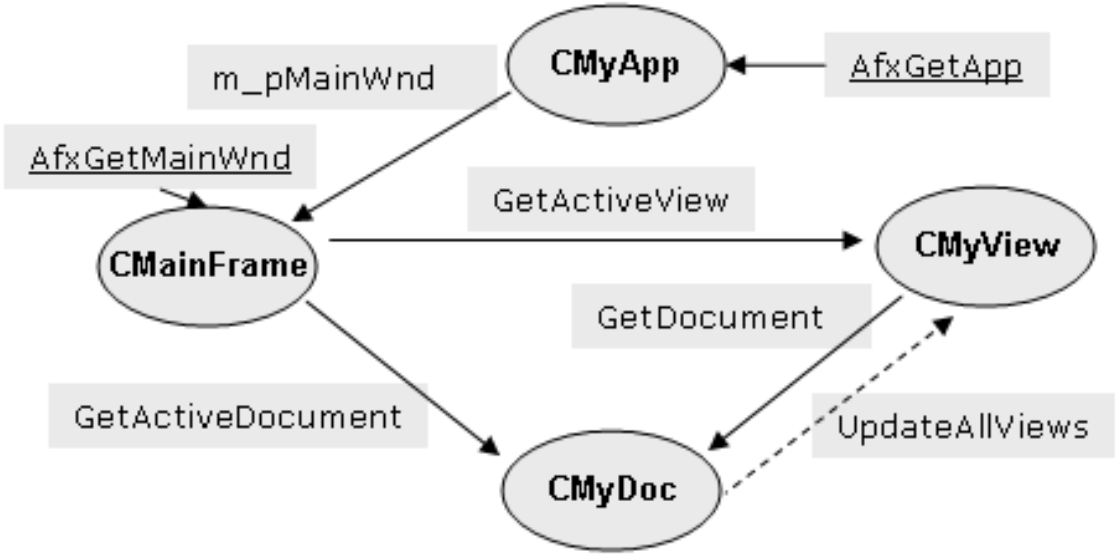
MyView.cpp

```
void CMyView::OnTestMax()
{
        CRect rt;
        CWnd* pFrame;

        pFrame = AfxGetMainWnd( );
        pFrame->GetClientRect( &rt );
        this->MoveWindow( rt );
}
```



주요 클래스간의 통신

4개의 주요 클래스 상호간의 객체 포인터 획득 관계



CView* GetActiveView( );

CDocument* GetActiveDocument( );

void UpdateAllViews(CView* *pSender*, **LPARAM** *lHint* **= 0L, CObject*** *pHint* **= NULL);**

CDocument* GetDocument( );

도큐먼트에서 뷰의 객체에 대한 주소를 알고 싶으면?

- 프레임 윈도우 객체를 거치면 가능

```
void CMyDoc::OnTest ()
{
        CFrameWnd* pFrame;   // CMainFrame 대신 사용
        pFrame = (CFrameWnd*)AfxGetMainWnd( );

        CView* pView;              //  CMyView 대신 사용
        pView = (CView*)pFrame->GetActiveView( );

        CDC* pDC;
        pDC = pView->GetDC();
        pDC->TextOut( 0, 0, "도큐먼트에서 직접 뷰에 출력하기" );
        pView->ReleaseDC( pDC );
}
```

각 파생 클래스의 고유 멤버변수나 멤버함수를 사용한다면?

```
#include “MainFrm.h”  // CMainFrame
#include “MyDoc.h”    // CMyView.h에서 CMyDoc 클래스를 사용
#include “MyView.h”   // CMyView
. . .
void CMyDoc::OnTest ()
{
        CMainFrame* pFrame;
        pFrame = (CMainFrame*)AfxGetMainWnd( );

        CMyView* pView;
        pView = (CMyView*)pFrame->GetActiveView( );

        // CMyView의 함수라고 가정
        pView->DoSomethingSpecificToCMyView( );
}
```

실습 6.6

실습 6.5에서 툴바가 사라지는 문제를 해결하자.

- 프레임 윈도우의 클라이언트 영역에서 툴바 윈도우의 높이를 고려하여 뷰 윈도우의 크기를 복원
- 툴바 윈도우의 크기
  - CWnd 클래스의 GetWindowRect()함수를 이용하여 획득
  - GetClientRect()함수를 사용해도 무방

```
Void  CWnd::GetWindowRect( LPRECT lpRect );
```

MainFrm.h

```
class CMainFrame : public CFrameWnd
{
protected: // create from serialization only
        CMainFrame( );
        DECLARE_DYNCREATE( CMainFrame )


… 생략…

// 멤버접근함수를 작성해야 하나, 여기서는 단순히 변수를 public으로 설정함
public:
        CStatusBar  m_wndStatusBar;
        CToolBar    m_wndToolBar;

…생략…
```

MyView.cpp

```
#include "MainFrm.h"
#include "MyDoc.h"
#include "MyView.h“
. . .
void CMyView::OnTestMax ()
{
        CRect rt;
        CMainFrame* pFrame;
        CToolBar* pToolBar;

        pFrame = (CMainFrame*)AfxGetMainWnd( );
        pToolBar = &( pFrame->m_wndToolBar );

        pFrame->GetClientRect( &rt );

        CRect rtTool;
        pToolBar->GetClientRect( &rtTool );

        rt.top = rt.top + rtTool.Height( );      // 툴바 윈도우 높이만큼 빼줌
        this->MoveWindow( rt );
}
```

툴바 윈도우는 부모 윈도우에 도킹될 수 있도록 구현

MainFrm.cpp

```
int CMainFrame::OnCreate( LPCREATESTRUCT lpCreateStruct )
{
        . . .
        m_wndToolBar.EnableDocking( CBRS_ALIGN_ANY );
        EnableDocking( CBRS_ALIGN_ANY );
        DockControlBar( &m_wndToolBar );

        return 0;
}
```

툴바 윈도우의 프레임 윈도우에 대한 도킹 여부 확인 필요

- MSDN을 참조하여 CToolBar 및 부모 클래스의 기능 확인

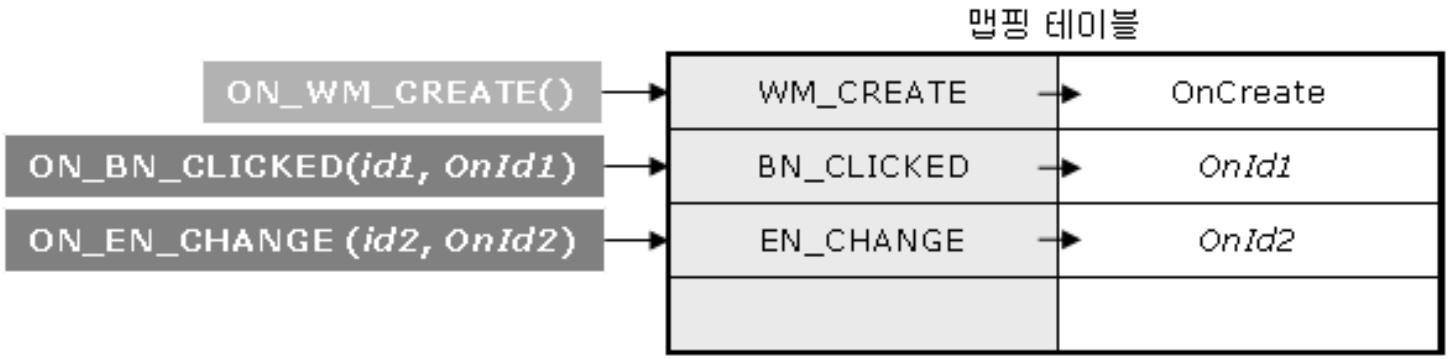
Lab.

실습 6.5에서 상태바가 사라지는 문제를 해결하자.

- 프레임 윈도우의 클라이언트 영역에서 툴바 및 상태바 윈도우의 높이를 고려하여 뷰 윈도우의 크기를 복원

메시지처리 유형 3

- 노티피케이션 코드에 대한 처리
- 노티피케이션 코드의 매핑 매크로는 WM_COMMAND와 동일

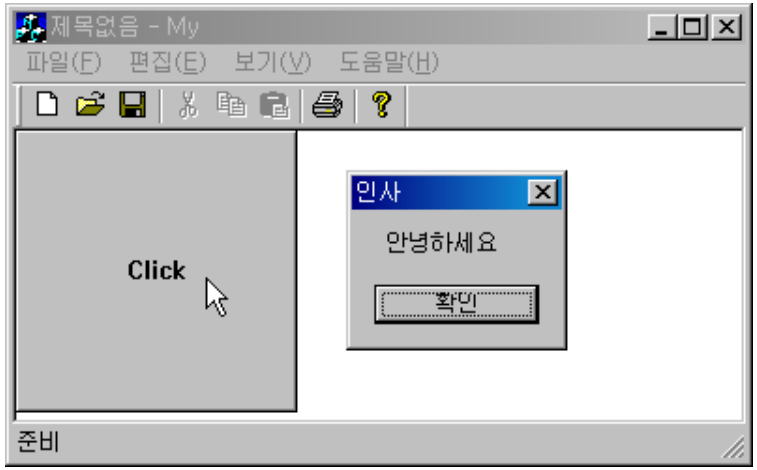


- **id1**: 이벤트 발생 주체의 식별자
- **OnId1**: 사용자가 정한 핸들러 이름

- 매핑 매크로 명칭은 ON으로 시작함
  - 버튼기호 BN + 노티피케이션의 코드 CLICKED

실습 6.7

버튼 윈도우를 추가하고 클릭하면 메시지 박스를 출력하자.



- 버튼 윈도우를 클래스화해 놓은 CButton 클래스를 활용
- Create( ) 함수를 이용하여 생성

BOOL Create(LPCTSTR *lpszCaption*, **DWORD** *dwStyle*,
const RECT& *rect*, **CWnd*** *pParentWnd*, **UINT** *nID* **)**

뷰 객체에 대한 WM_CREATE 메시지 핸들러 추가

MyView.h

```
DECLARE_MESSAGE_MAP()
afx_msg int OnCreate( LPCREATESTRUCT lpCreateStruct );
```

MyView.cpp

```
BEGIN_MESSAGE_MAP( CMyView, Cview )
        ON_WM_CREATE()
END_MESSAGE_MAP()
. . .
int CMyView::OnCreate( LPCREATESTRUCT lpCreateStruct )
{
        if ( CView::OnCreate(lpCreateStruct) == -1 )
                return -1;
        // TODO: Add your specialized creation code here
}
```

뷰 클래스에 버튼 객체를 멤버 변수로 추가

MyView.h

```
class CMyView : public CView {
protected: // serialization에서만 만들어집니다.
        CMy2View();
        DECLARE_DYNCREATE(CMy2View)

// 특성입니다.
public:
        CMy2Doc* GetDocument() const;
private:
        CButton* m_pBtn;

 . . .
}
```

버튼 윈도우 생성

MyView.cpp

```
CMyView::CMyView( ) {
        m_pBtn = NULL;
}
CMyView::~CMyView( ) {
        if ( m_pBtn ) {
                m_pBtn->DestroyWindow();
                delete m_pBtn;
        }
}
int CMyView::OnCreate( LPCREATESTRUCT lpCreateStruct ) {
        if (CView::OnCreate(lpCreateStruct) == -1)
                return -1;

        m_pBtn = new CButton();
        m_pBtn->Create( _T("Click"),              // 캡션
                        WS_VISIBLE | WS_CHILD,// 스타일
                        CRect( 0, 0, 150, 150 ),          // 위치 및 크기
                        this,                             // 뷰 클래스를 부모 클래스로 설정
                        888                               // 윈도우 식별자
                        );

        return 0;
}
```

버튼을 눌렀을 때 메시지 박스 출력

MyView.h

```
DECLARE_MESSAGE_MAP()
afx_msg int OnCreate( LPCREATESTRUCT lpCreateStruct );
afx_msg void OnBtnClick();
```

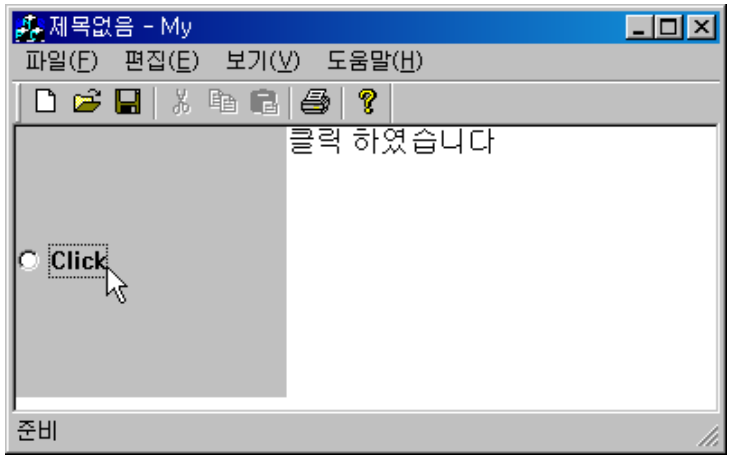
MyView.cpp

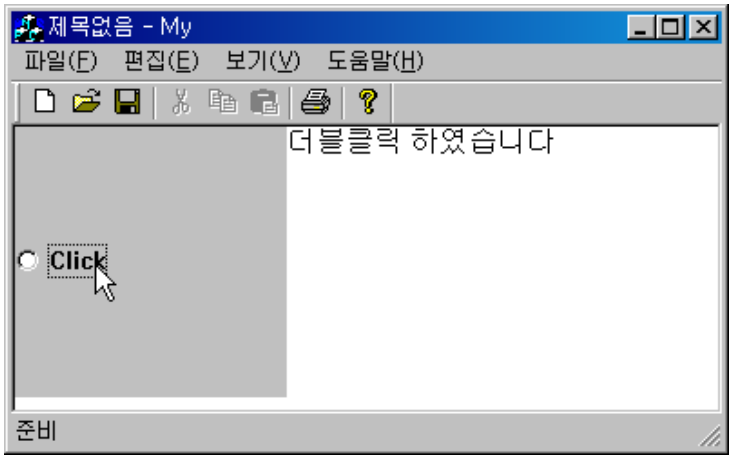
```
BEGIN_MESSAGE_MAP( CMyView, Cview )
        . . .
        ON_WM_CREATE()
        ON_BN_CLICKED( 888, OnBtnClick )
        END_MESSAGE_MAP()

void CMyView::OnBtnClick( )
{
        MessageBox( _T("안녕하세요"), _T("인사") );
}
```

- 위저드를 사용할 수 없으므로 관련 코드를 직접 입력

실습 6.8

더블클릭을 처리하자.





- 더블클릭은 라디오 버튼에서만 사용 가능
- 뷰 윈도우에 메시지 출력

MyView.h

```
DECLARE_MESSAGE_MAP( )
afx_msg int OnCreate( LPCREATESTRUCT lpCreateStruct );
afx_msg void OnBtnClick( );
afx_msg void OnBtnDblClick( );
```

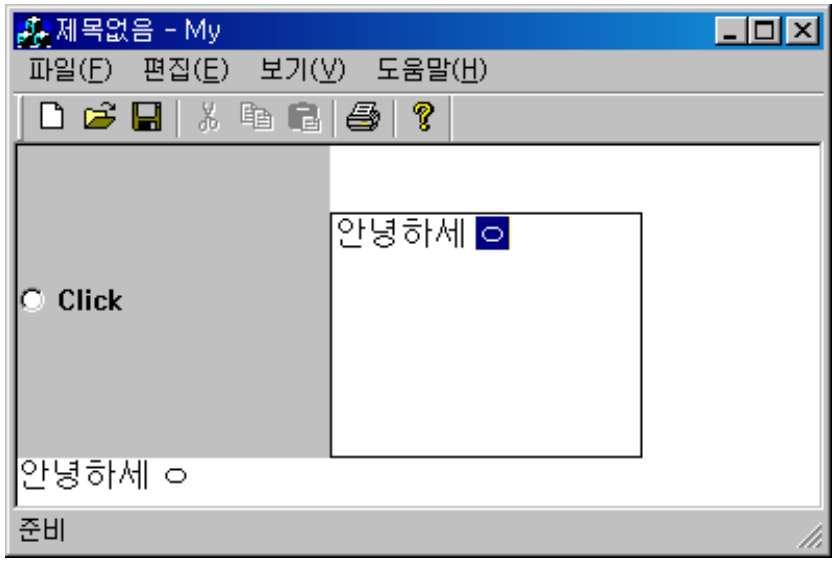
MyView.cpp

```
BEGIN_MESSAGE_MAP( CMyView, Cview )
        . . .
        ON_WM_CREATE()
        ON_BN_CLICKED( 888, OnBtnClick )
        ON_BN_DOUBLECLICKED( 888, OnBtnDblClick )
        END_MESSAGE_MAP()

int CMyView::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
        if (CView::OnCreate(lpCreateStruct) == -1)
                return -1;

        m_pBtn = new CButton();
```

```
        m_pBtn->Create( "Click",
                        WS_VISIBLE | WS_CHILD | BS_RADIOBUTTON,
                        CRect( 0, 0, 150, 150 ),
                        this,
                        888
                        );
        return 0;
}
void CMyView::OnBtnClick( )
{
        CDC* pDC;
        pDC = GetDC( );
        pDC->TextOut( 150, 0, _T("클릭 하였습니다.     ") );
        ReleaseDC( pDC );
}
void CMyView::OnBtnDblClick()
{
        CDC* pDC;
        pDC = GetDC( );
        pDC->TextOut( 150, 0, _T("더블클릭 하였습니다.") );
        ReleaseDC( pDC );
}
```

실습 6.9

에디트 윈도우에 문자를 입력하면 뷰 윈도우에 동일하게 출력하자.



- EN_CHANGE 노티피케이션 코드 활용
- 에디트 윈도우의 입력 내용 읽기/쓰기
  - CWnd 클래스의 GetWindowText( ), SetWindowText( ) 사용

MyView.h

```
class CMyView : public CView
{
. . .
private:
        CButton* m_pBtn;
        CEdit* m_pEdt;
. . .
protected:
        DECLARE_MESSAGE_MAP( )
public:
        afx_msg int OnCreate( LPCREATESTRUCT lpCreateStruct );
        afx_msg void OnBtnClick( );
        afx_msg void OnBtnDblClick( );
        afx_msg void OnEnChange( );
};
```

MyView.cpp

```
BEGIN_MESSAGE_MAP( CMyView, Cview )
        . . .
        ON_WM_CREATE( )
        ON_BN_CLICKED( 888, OnBtnClick )
        ON_BN_DOUBLECLICKED( 888, OnBtnDblClick )
        ON_EN_CHANGE( 999, OnEnChange )
END_MESSAGE_MAP( )

CMyView::CMyView( )
{
        m_pBtn = NULL;
        m_pEdt = NULL;
}

CMyView::~CMyView( )
{
        if ( m_pBtn ) {
                m_pBtn->DestroyWindow( );
                delete m_pBtn;     }
        if ( m_pEdt ) {
                m_pEdt->DestroyWindow( );
                delete m_pEdt;     }
}
```

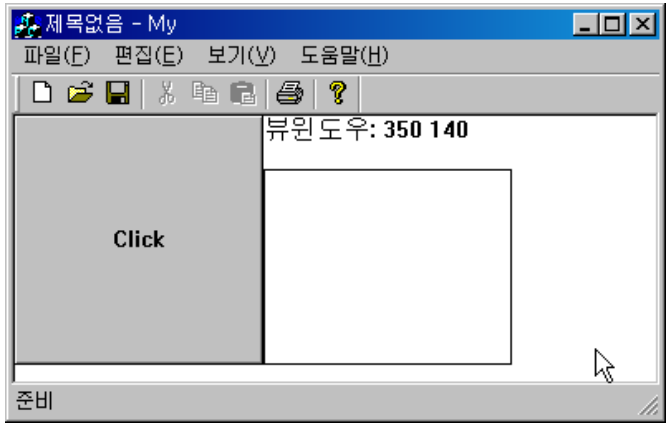
```
int CMyView::OnCreate( LPCREATESTRUCT lpCreateStruct )
{
        if ( CView::OnCreate(lpCreateStruct) == -1 )
                return -1;

        m_pBtn = new CButton( );
        m_pBtn->Create( _T("Click"),
                        WS_VISIBLE | WS_CHILD | BS_RADIOBUTTON,
                        CRect( 0, 0, 150, 150 ),
                        this,
                        888
                        );

        m_pEdt = new CEdit( );
        m_pEdt->Create(                         // 4개의 인자 사용
                        WS_VISIBLE | WS_CHILD | WS_BORDER,
                        CRect( 150, 32, 300, 150 ),
                        this,
                        999
                        );

        return 0;
}
```

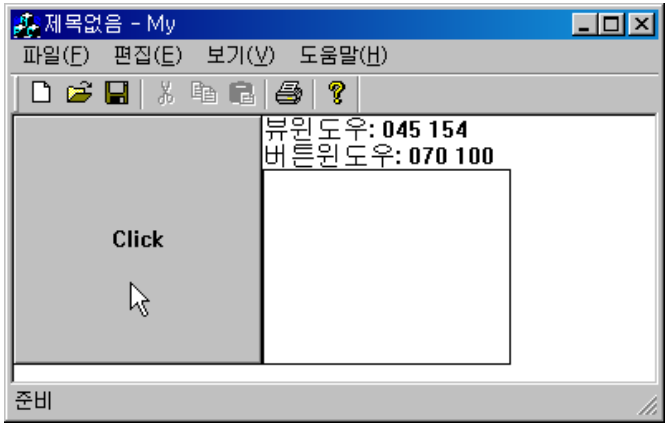
```
void CMyView::OnBtnClick( )
{
        m_pEdt->SetWindowText( _T("") );
}
void CMyView::OnBtnDblClick( )
{
        CDC* pDC;
        pDC = GetDC( );
        pDC->TextOut( 310, 0, _T("더블클릭 하였습니다") );
        ReleaseDC(pDC);
}
void CMyView::OnEnChange( )
{
        CString strMsg;
        m_pEdt->GetWindowText( strMsg );

        CDC* pDC;
        pDC = GetDC( );
        pDC->TextOut( 0, 150, strMsg );
        ReleaseDC( pDC );
}
```

메시지처리 유형 4

실습 6.10

뷰 및 버튼 윈도우에서의 마우스 좌표를 뷰 윈도우에 출력하자.





- 뷰 윈도우에서의 마우스 움직임 처리
  - CMyView::OnMouseMove() 핸들러에서 처리

뷰 윈도우에서의 마우스 움직임 처리 (WM_MOUSEMOVE)

MyView.h

```
afx_msg void OnMouseMove(UINT, CPoint);
```

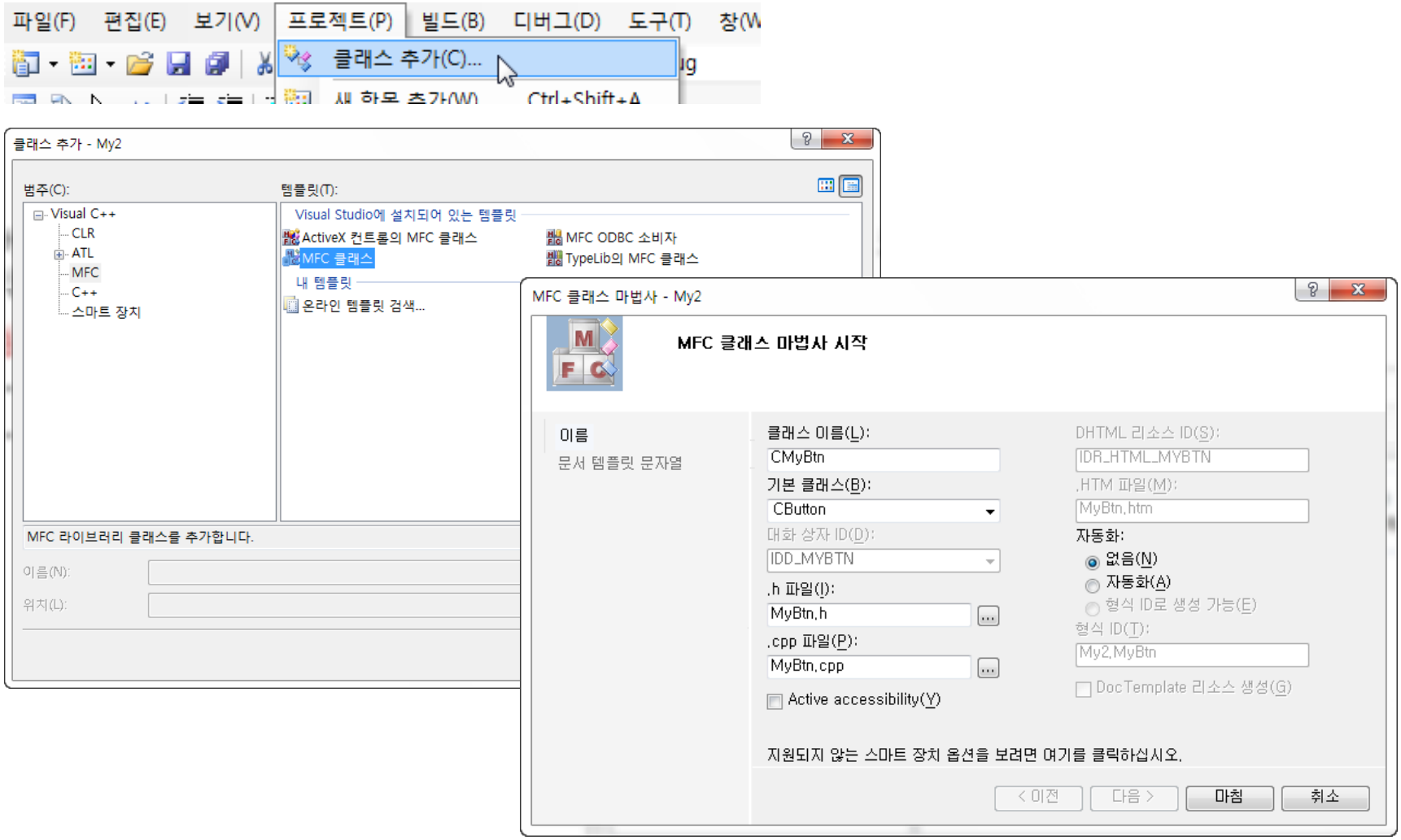
MyView.cpp

```
BEGIN_MESSAGE_MAP( CMyView, CView )
        . . .
        ON_WM_MOUSEMOVE()
END_MESSAGE_MAP()
. . .
void CMyView::OnMouseMove( UNIT nFlags, CPoint point )
{
        CString strPos;
        strPos.Format( "뷰윈도우: %04d %04d", point.x, point.y );

        CDC* pDC;
        pDC = GetDC( );
        pDC->TextOut( 150, 0, strPos );
        ReleaseDC( pDC );

        CView::OnMouseMove( nFlags, point );
}
```

버튼 윈도우에서의 마우스 움직임 처리

- 버튼 윈도우 상에서 마우스 움직임 처리를 위한 핸들러는 코드상에서 제공되지 않음
- 상속을 통해 해결
  - CButton 클래스를 상속하여 파생 클래스 CMyBtn을 정의
  - OnMouseMove() 핸들러 재정의

버튼 윈도우에서의 마우스 움직임 처리



MyBtn.h

```
class CMyBtn: public Cbutton {
        . . .
        afx_msg void OnMouseMove( UINT, CPoint );
};
```

MyBtn.cpp

```
BEGIN_MESSAGE_MAP( CMyBtn, CButton )
        ON_WM_MOUSEMOVE()
END_MESSAGE_MAP()
. . .
void CMyBtn::OnMouseMove( UNIT nFlags, CPoint point )
{
        CString strPos;
        strPos.Format( _T("버튼윈도우: %04d %04d"), point.x, point.y );

        CDC* pDC;
        pDC = GetDC( );
        pDC->TextOut( 150, 16, strPos );
        ReleaseDC( pDC );

        CButton::OnMouseMove( nFlags, point );
}
```

MyView.h

```
class CMyBtn;
class CMyView: public CView {
        . . .
private:
        CMyBtn *m_pBtn;
        CEdit     *m_pEdt;
};
```

MyView.cpp

```
#include "MyBtn.h"
. . .
void CMyView::OnCreate( LPCREATESTRUCT lpCreateStruct )
{
        . . .
        m_pBtn = new CMyBtn( );
        m_pBtn->Create( _T("Click"),
                        WS_VISIBLE | WS_CHILD | BS_RADIOBUTTON,
                        CRect( 0, 0, 150, 150 ), this, 888
                        );
        . . .
}
```

버튼 윈도우에서의 마우스 좌표를 뷰 윈도우에 출력

MyBtn.cpp

```
void CMyBtn::OnMouseMove( UINT nFlags, CPoint point )
{
        CWnd* pView;
        pView = GetParent( );

        CString strPos;
        strPos.Format( _T("버튼윈도우: %04d %04d"), point.x, point.y );

        CDC* pDC;
        pDC = pView->GetDC( );
        pDC->TextOut( 150, 16, strPos );
        ReleaseDC( pDC );

        CButton::OnMouseMove( nFlags, point );
}
```

메시지 맵 벗어나기

- API에서의 메시지 처리 함수(윈도우 프로시저) 프로토타입

```
LRESULT WndProc( HWND, UINT, WPARAM, LPARAM )
```

- 첫 번째 인자가 HWND이므로, CWnd 클래스 멤버함수에 정의

```
virtual LRESULT WindowProc(
    UINT message, WPARAM wParam, LPARAM lParam
)
```

- CMyBtn 클래스는 CWnd 클래스의 자식 클래스이므로

  WindowProc() 함수를 재정의하여 메시지 맵 코드 삭제 가능

MYBTN.H

```
public:
        LRESULT WindowProc( UINT, WPARAM, LPARAM );
```

MYBTN.CPP

```
LRESULT CMyBtn::WindowProc( UINT msg, WPARAM wParam, LPARAM lParam )
{
        switch( msg ) {
        case WM_MOUSEMOVE:
                CWnd* pView;
                pView = GetParent( ); //뷰 객체 획득

                CString strPos;
                strPos.Format( "%04d %04d", LOWORD(lParam), HIWORD(lParam) );

                CDC* pDC;
                pDC = pView->GetDC( );
                pDC->TextOut( 150, 16, strPos );
                pView->ReleaseDC( pDC );

                break;
        }
        return CButton::WindowProc( msg, wParam, lParam );
}
```

사용자 메시지 처리 (메시지처리 유형 5)

Lab.

SendMessage( )를 사용하여 뷰 객체에서 화면 출력을 처리하자.

- **CWnd::SendMessage( )**

  #define WM_MYMOVE WM_USER+1

- **사용자 메시지 매핑 매크로**

  ON_MESSAGE(WM_MYMOVE, OnMyMove);

  http://msdn.microsoft.com/ko-kr/library/k35k2bfs(v=VS.80).aspx

- **메시지 핸들러 프로토타입**

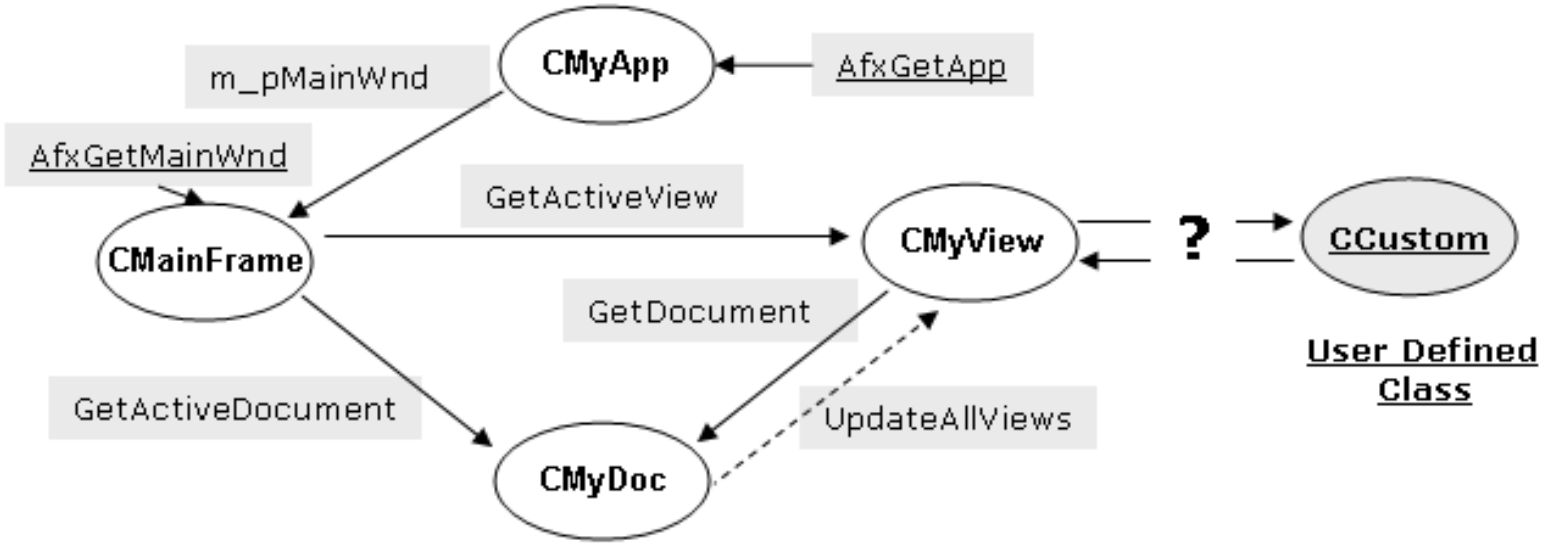
  afx_msg LRESULT OnMyMove(WPARAM, LPARAM);

  - wParam 생성: MAKEWPARAM 매크로
  - lParam 생성: MAKELPARAM 매크로

  http://msdn.microsoft.com/ko-kr/library/ms632664(v=VS.85).aspx

MFC 프로그램의 일반적 유형

- 주요 객체 네 개와 프로그래머가 필요로 하는 객체로 구성
- 새로 추가한 객체와의 통신 요구



- 기존 어느 객체와도 통신이 가능

도큐먼트 객체에 대한 접근 예제

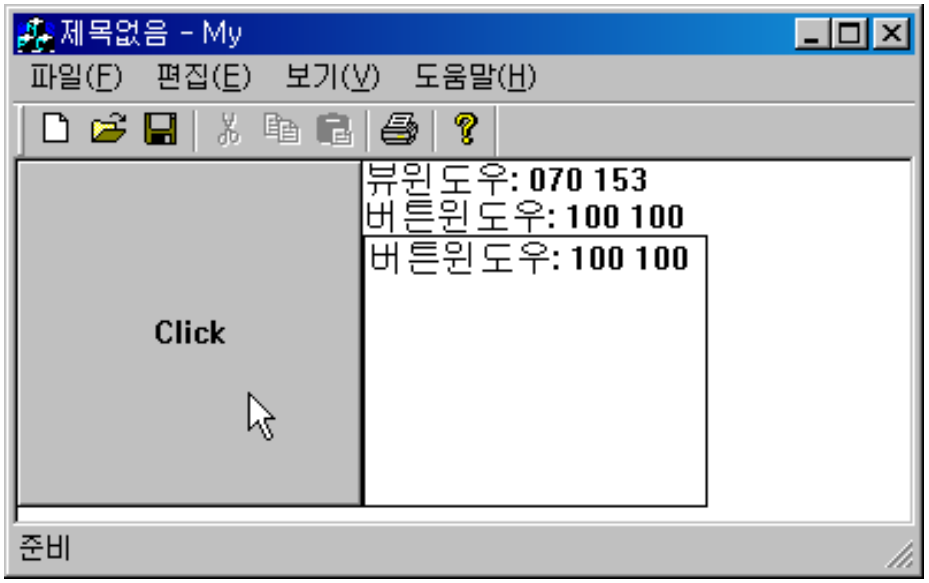
- 프레임 윈도우 객체를 통해 도큐먼트 객체에 접근
- 유사한 방법으로 뷰 객체에 접근 가능

Custom.cpp

```
#include “mydoc.h”
void CCustom::DocumentAccessing()
{
        CFrameWnd* pFrame;
        pFrame = ( CFrameWnd* )AfxGetMainWnd( );
        CMyDoc* pDoc;
        pDoc = ( CMyDoc* )pFrame->GetActiveDocument( );
        pDoc->DoSomething( );
}
```

실습 6.11

실습 6.10을 변경하여 버튼 윈도우에서 마우스를 움직일 때 에디트 윈도우에도 좌표를 출력하자.



- 뷰 객체의 멤버로 있는 에디드 객체를 얻어와야 함
- 뷰 객체의 주소 타입은 CMyView*

MyBtn.cpp

```
#include "mainfrm.h"
#include "mydoc.h"
#include "myview.h"
. . .
void CMyBtn::OnMouseMove( UINT nFlags, CPoint point ) {
        CMainFrame* pFrame = ( CMainFrame* )AfxGetMainWnd( );
        CMyView* pView = ( CMyView* )( pFrame->GetActiveView() );
        CEdit* pEdit = pView->m_pEdt;

        CString strPos;
        strPos.Format( "버튼윈도우: %04d %04d", point.x, point.y );

        m_pEdit->SetWindowText( strPos );

        CDC* pDC;
        pDC = pView->GetDC( );
        pDC->TextOut( 150, 16, strPos );
        pView->ReleaseDC( pDC );

        CButton::OnMouseMove( nFlags, point );
}
```

실습 6.11의 문제점

- CMyBtn 클래스의 범용성 떨어짐
  - CMyView 클래스 사용을 위해 3개의 기본 헤더파일 사용

MyBtn.h

```
class CMyBtn: public CButton
{
. . .
public:
        CEdit* m_pEdit;
private:
        void SetWnd( CEdit* pEdit );
. . .
```

MyBtn.cpp

```
#include "mainfrm.h"
#include "mydoc.h"
#include "myview.h"
 . . .
void CMyBtn::SetWnd( CEdit* pEdit )
{
        m_pEdit = pEdit;
}
```

```
void CMyBtn::OnMouseMove( UINT nFlags, CPoint point )
{
        CWnd* pView;
        pView = GetParent( );

        CString strPos;
        strPos.Format( "버튼윈도우: %04d %04d", point.x, point.y );

        m_pEdit->SetWindowText( strPos );

        CDC* pDC;
        pDC = pView->GetDC( );
        pDC->TextOut( 150, 16, strPos );
        pView->ReleaseDC( pDC );

        CButton::OnMouseMove( nFlags, point );
}
```

MyView.cpp

```
int CMyView::OnCreate( LPCREATESTRUCT lpCreateStruct )
{
        if (CView::OnCreate(lpCreateStruct) == -1)
                return -1;

        m_pBtn = new CMyBtn();

        m_pBtn->Create( "Click", WS_VISIBLE|WS_CHILD,
                        CRect(0,0,150,150), this, 888 );

        m_pEdt = new CEdit( );

        m_pEdt->Create( WS_VISIBLE|WS_CHILD|WS_BORDER,
                        CRect(150,32,300,150), this, 999 );

        m_pBtn->SetWnd(m_pEdt);  //버튼 윈도우에게 에디트 윈도우 객체 전달

        return 0;
}
```