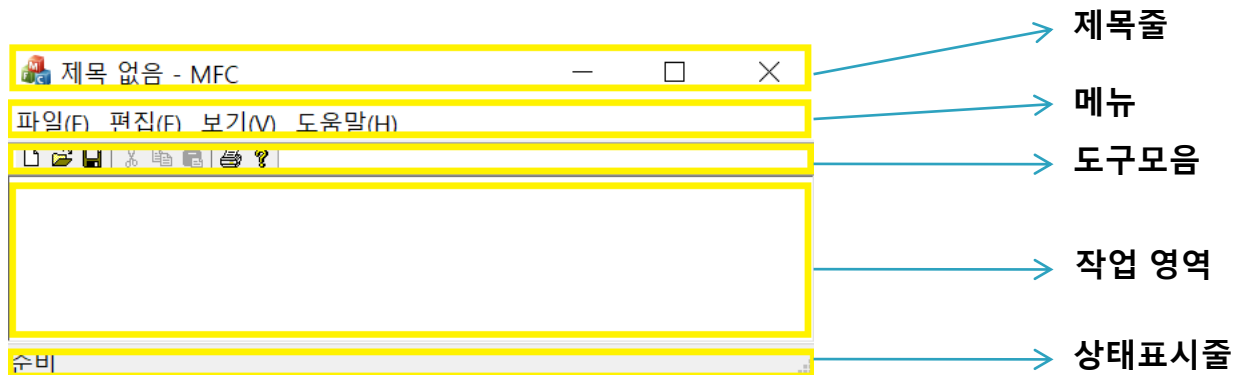


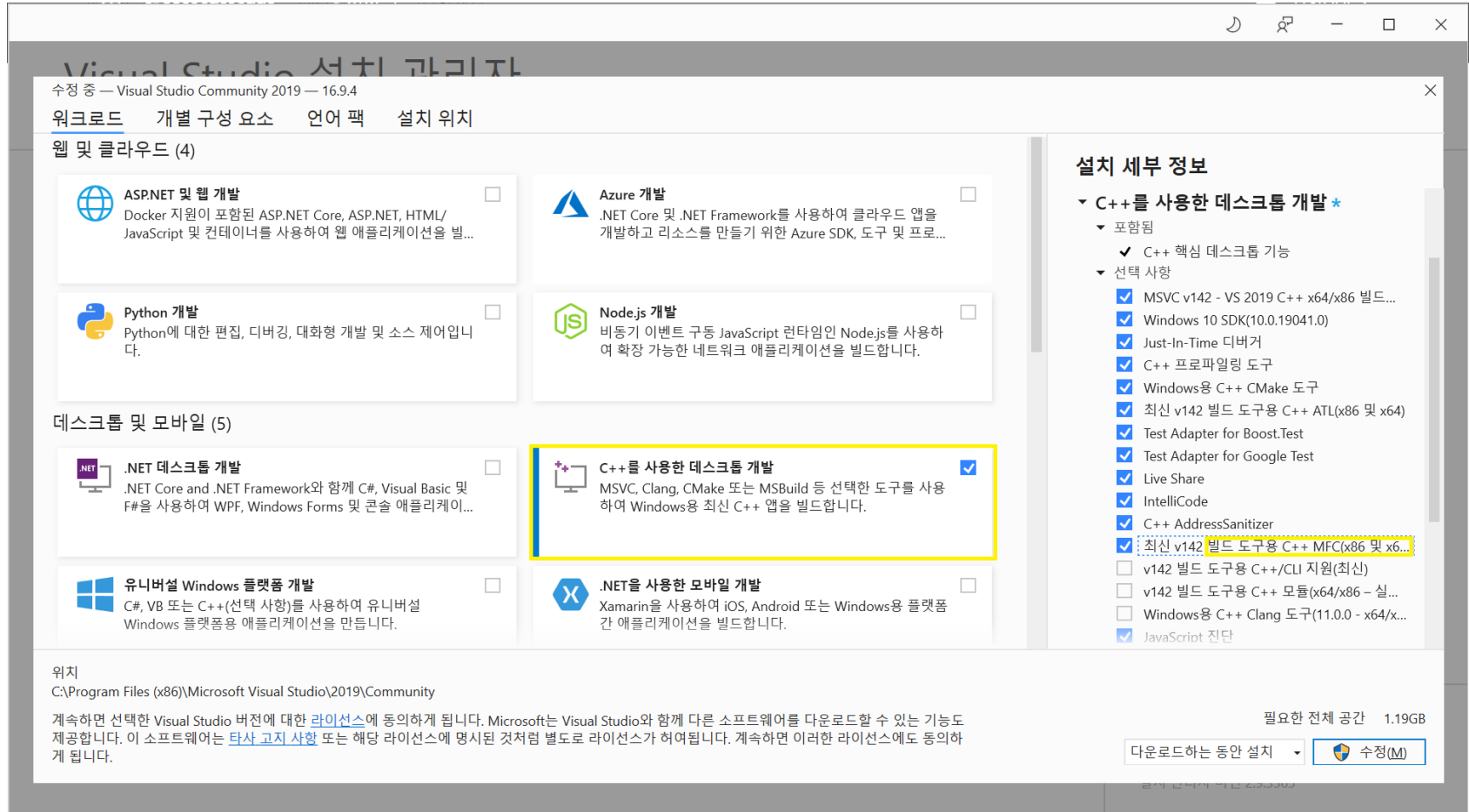
MFC

MFC란?

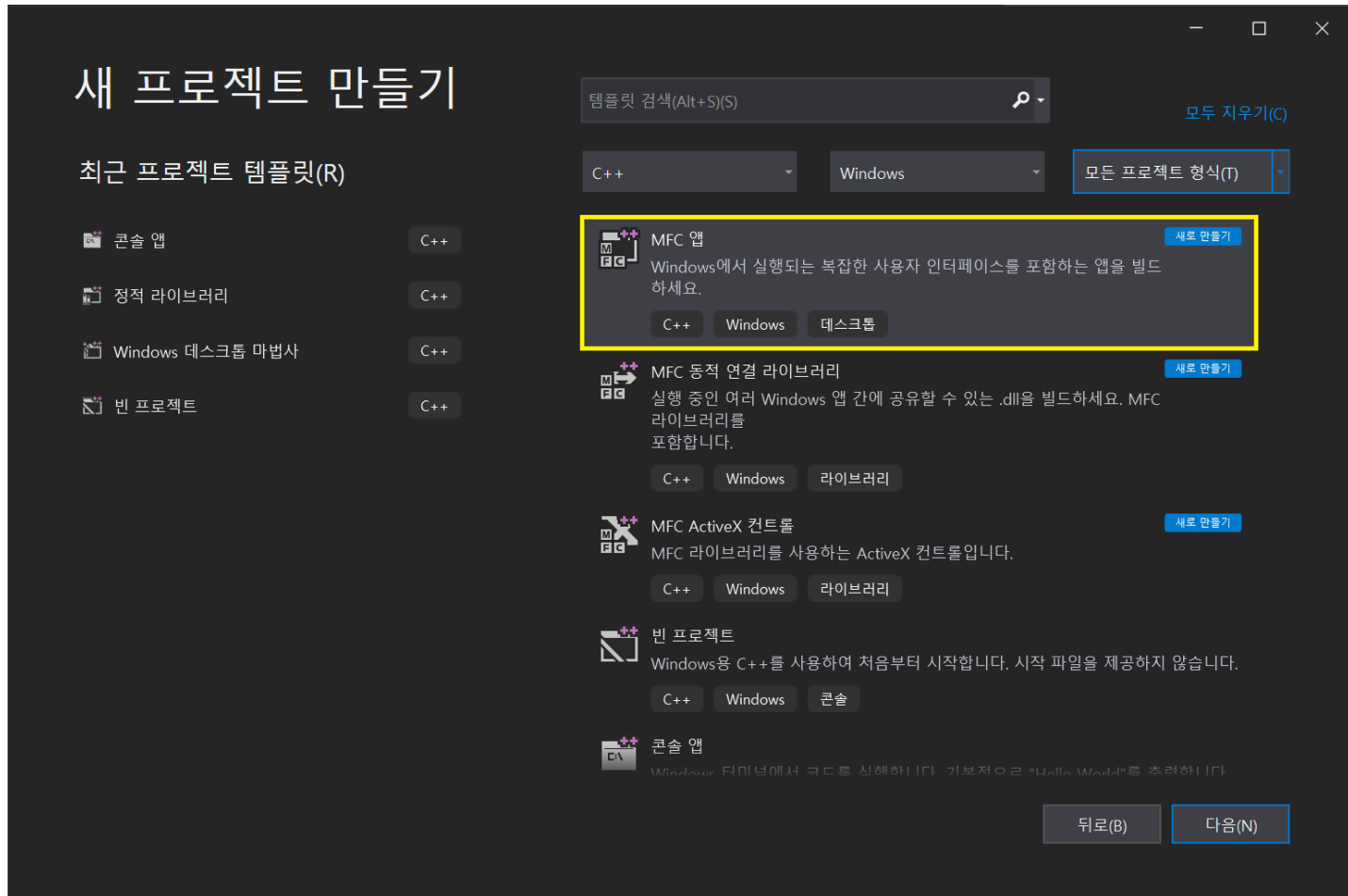
- ▶ Microsoft Foundation Class의 줄임말로 **마이크로소프트사에서 윈도우 애플리케이션을 제작**하라고 제공하는 **C++ 클래스 라이브러리**의 집합
- ▶ 특징으로는 **GUI(Graphic User Interface)**를 제공



MFC 프로젝트 만들기



MFC 프로젝트 만들기



MFC 프로젝트 만들기

MFC 애플리케이션
애플리케이션 종류 옵션

애플리케이션 종류

문서 템플릿 속성

사용자 인터페이스 기능

고급 기능

생성된 클래스

애플리케이션 종류(T)

단일 문서

애플리케이션 종류 옵션:

☐ 탭 문서(B)

☒ 문서/뷰 아키텍처 지원(V)

대화 상자 기반 옵션(O)

<없음>

복합 문서 지원

<없음>

문서 지원 옵션:

☐ 활성 문서 서버(A)

☐ 활성 문서 컨테이너(D)

☐ 복합 파일 지원(U)

프로젝트 스타일

MFC standard

비주얼 스타일 및 색(Y)

Windows Native/Default

☐ 비주얼 스타일 전환 사용(C)

리소스 언어(L)

ko-KR

MFC 사용

공유 DLL에서 MFC 사용

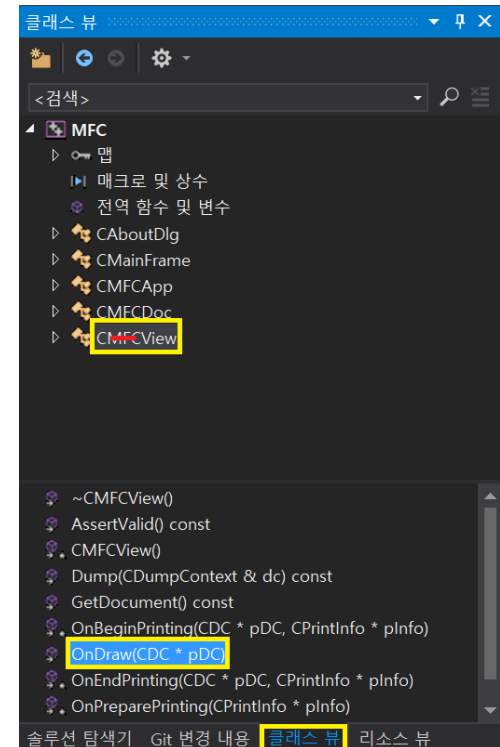
이전 다음 마침 취소

출력

- ▶ 화면의 출력은 View 클래스에서 담당한다
- ▶ 출력 할 내용은 OnDraw() 함수 내에 정의

```
void CMFCView::OnDraw(CDC* /*pDC*/)  
{  
    CMFCDoc* pDoc = GetDocument();  
    ASSERT_VALID(pDoc);  
    if (!pDoc)  
        return;
```

```
    // TODO: 여기에 원시 데이터에 대한 그리기 코드를 추가합니다.  
}
```



문자 출력

▶ DC를 가져오는 세 가지 방법

- **GetDC()** 함수로 DC를 사용 후, **ReleaseDC()** 함수로 해제 한다
- **CClientDC()** 함수로 DC를 사용 한다
- OnDraw()의 매개 변수의 **pDC**를 사용 한다(기본 적으로 주석 처리 되어 있으나 해제 후 사용할 수 있다)

```
void CMFCView::OnDraw(CDC* pDC)
{
    CMFCDoc* pDoc = GetDocument();
    ASSERT_VALID(pDoc);
    if (!pDoc)
        return;

    // TODO: 여기에 원시 데이터에 대한 그리기 코드를 추가합니다.
    CDC* pdc = GetDC();
    pdc->TextOutW(100, 50, _T("MFC Application"));
    ReleaseDC(pdc);

    CClientDC dc(this);
    dc.TextOutW(100, 100, _T("MFC Application"));

    pDC->TextOutW(100, 150, _T("MFC Application"));
}
```

문자 출력

- ▶ **BOOL TextOutW(int x, int y, const CString& str)**

- 화면에 **텍스트를 출력**해 준다
- x, y : 출력될 화면의 x, y 좌표
- str : 화면에 출력될 내용

- ▶ **_T()**

- **유니코드 플랫폼 환경**에서 사용된 **문자열 형태로 변환**하기 위하여 사용하는 매크로 함수

- ▶ **유니코드**

- 나라별로 서로 다른 언어 체계를 가지고 있지만, 국가별 모든 언어에 대해서 고유 번호를 제공하여 **어떤 플랫폼, 프로그램, 언어에 상관없이 모든 문자를 처리할 수 있도록 한 코드 체계**

그래픽 출력

```
void CMFCView::OnDraw(CDC* pDC)
{
    CMFCDoc* pDoc = GetDocument();
    ASSERT_VALID(pDoc);
    if (!pDoc)
        return;



    // TODO: 여기에 원시 데이터에 대한 그리기 코드를 추가합니다.
    CPoint pt1(100, 100), pt2(200, 200);
    pDC->MoveTo(pt1); // or pDC->MoveTo(100, 100);
    pDC->LineTo(pt2); // or pDC->LineTo(200, 200);

    CRect rc1(300, 100, 400, 200);
    pDC->Rectangle(&rc1); // or pDC->Rectangle(300, 100, 400, 200);

    CRect rc2(500, 100, 600, 200);
    pDC->Ellipse(&rc2); // or pDC->Ellipse(500, 100, 600, 200);
}
```

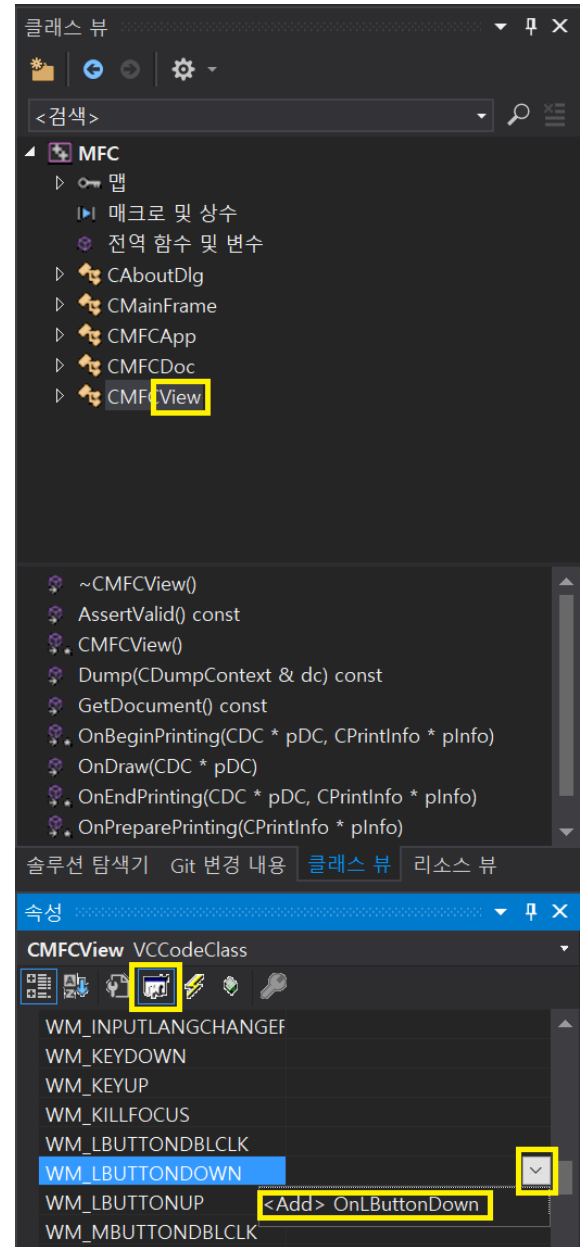
입력

▶ 마우스 왼쪽 버튼 클릭

1. [클래스 뷰]의 View 클래스를 선택
2. [속성]에서 [메시지] 아이콘  을 선택
3. 리스트에서 WM_LBUTTONDOWN을 선택
4.  을 선택하여 <Add>로 추가할 수 있다

```
void CMFCView::OnLButtonDown(UINT nFlags, CPoint point)
{
    // TODO: 여기에 메시지 처리기 코드를 추가 및/또는 기본값을 호출합니다.

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



입력

```
void CMFCView::OnLButtonDown(UINT nFlags, CPoint point)
{
    // TODO: 여기에 메시지 처리기 코드를 추가 및/또는 기본값을 호출합니다.
    CString str;
    str.Format(_T("[%d, %d]"), point.x, point.y);

    CClientDC dc(this);
    dc.TextOutW(point.x, point.y, str);

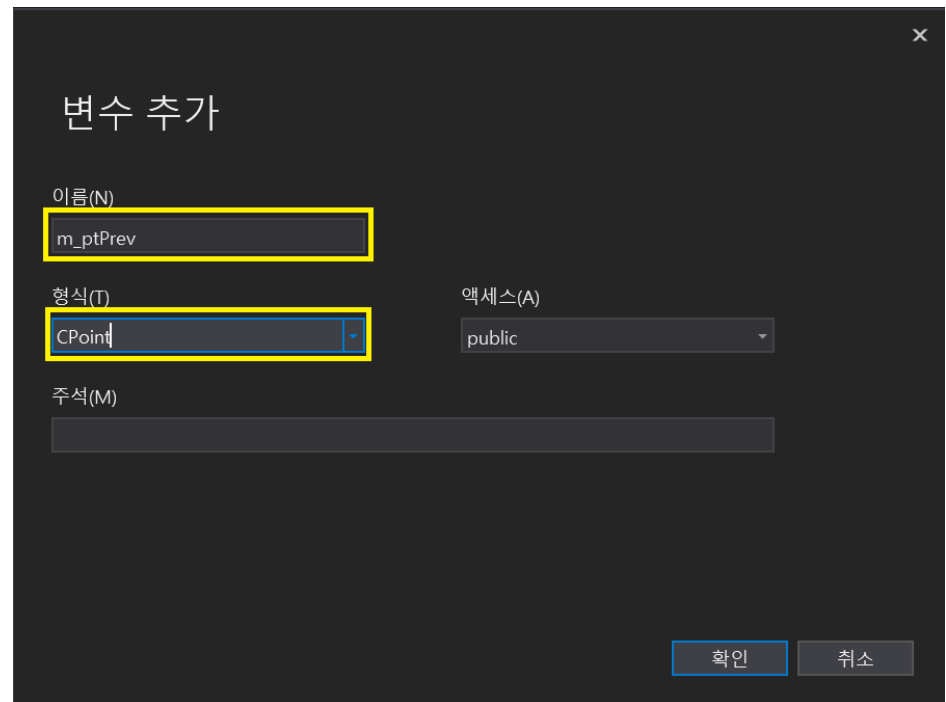
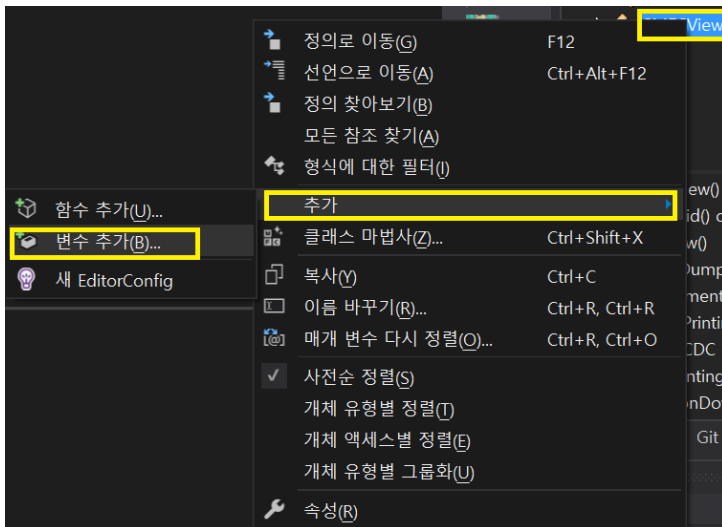
    CView::OnLButtonDown(nFlags, point);
}
```

- ▶ View 클래스는 작업 영역을 관리하는 클래스 이기 때문에 해당 영역을 넘어서면 작동하지 않는다

입력

▶ 변수 추가

- 기존과 같이 해당 클래스에 직접 추가
- 해당 클래스를 선택하여 [변수 추가] 항목을 이용하여 추가할 수 있다



마우스

```
void CMFCView::OnLButtonDown(UINT nFlags, CPoint point)
{
    // TODO: 여기에 메시지 처리기 코드를 추가 및/또는 기본값을 호출합니다.
    SetCapture();
    m_ptPrev = point;

    CView::OnLButtonDown(nFlags, point);
}
void CMFCView::OnLButtonUp(UINT nFlags, CPoint point)
{
    // TODO: 여기에 메시지 처리기 코드를 추가 및/또는 기본값을 호출합니다.
    ReleaseCapture();

    CView::OnLButtonUp(nFlags, point);
}
void CMFCView::OnMouseMove(UINT nFlags, CPoint point)
{
    // TODO: 여기에 메시지 처리기 코드를 추가 및/또는 기본값을 호출합니다.
    if (GetCapture() != this) return;

    CClientDC dc(this);
    dc.MoveTo(m_ptPrev);
    dc.LineTo(point);
    m_ptPrev = point;

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

마우스

▶ SetCapture()

- 마우스가 활성화된 윈도우 영역 밖으로 이동했을 경우의 지속적으로 마우스 메시지를 받기 위하여 사용
- SetCapture()를 사용 후 반드시 ReleaseCapture()를 호출 해야 한다

▶ ReleaseCapture()

- SetCapture() 사용이 끝났을 때 호출 한다

▶ GetCapture()

- 현재 마우스를 캡처하고 있는 윈도우의 포인터를 리턴 한다
- 현재 마우스가 활성화 윈도우 영역 밖에 있는지 아닌지 확인을 위하여 사용

키보드

- ▶ 키보드 입력에 따라 사각형을 이동시킨다

```
class CMFCView : public CView
{
    ...
public:
    CSize m_ViewSize; // 클라이언트 영역의 좌표를 저장할 변수.
    CPoint m_Pt; // 사각형을 출력할 좌표를 저장하기 위한 변수.
    ...
};

void CMFCView::OnDraw(CDC* pDC)
{
    CMFCDoc* pDoc = GetDocument();
    ASSERT_VALID(pDoc);
    if (!pDoc)
        return;

    // TODO: 여기에 원시 데이터에 대한 그리기 코드를 추가합니다.
    pDC->Rectangle(m_Pt.x - 10, m_Pt.y - 10, m_Pt.x + 10, m_Pt.y + 10);
}
```

키보드

```
void CMFCView::OnSize(UINT nType, int cx, int cy) // WM_SIZE : 윈도우 크기가 변경되면 호출.
{
    CView::OnSize(nType, cx, cy);
    // TODO: 여기에 메시지 처리기 코드를 추가합니다.
    m_ViewSize = CSize(cx, cy); // 클라이언트 영역의 크기를 저장.
    m_Pt = CPoint(cx * 0.5f, cy * 0.5f); // 사각형이 그려질 위치를 화면의 중앙으로 한다.
}
void CMFCView::OnKeyDown(UINT nChar, UINT nRepCnt, UINT nFlags) // WM_KEYDOWN
{
    // TODO: 여기에 메시지 처리기 코드를 추가 및/또는 기본값을 호출합니다.
    switch (nChar)
    {
    case VK_UP:
        m_Pt.y -= 10;
        if (0 > m_Pt.y) m_Pt.y = m_ViewSize.cy;
        break;

    case VK_DOWN:
        m_Pt.y += 10;
        if (m_ViewSize.cy < m_Pt.y) m_Pt.y = 0;
        break;

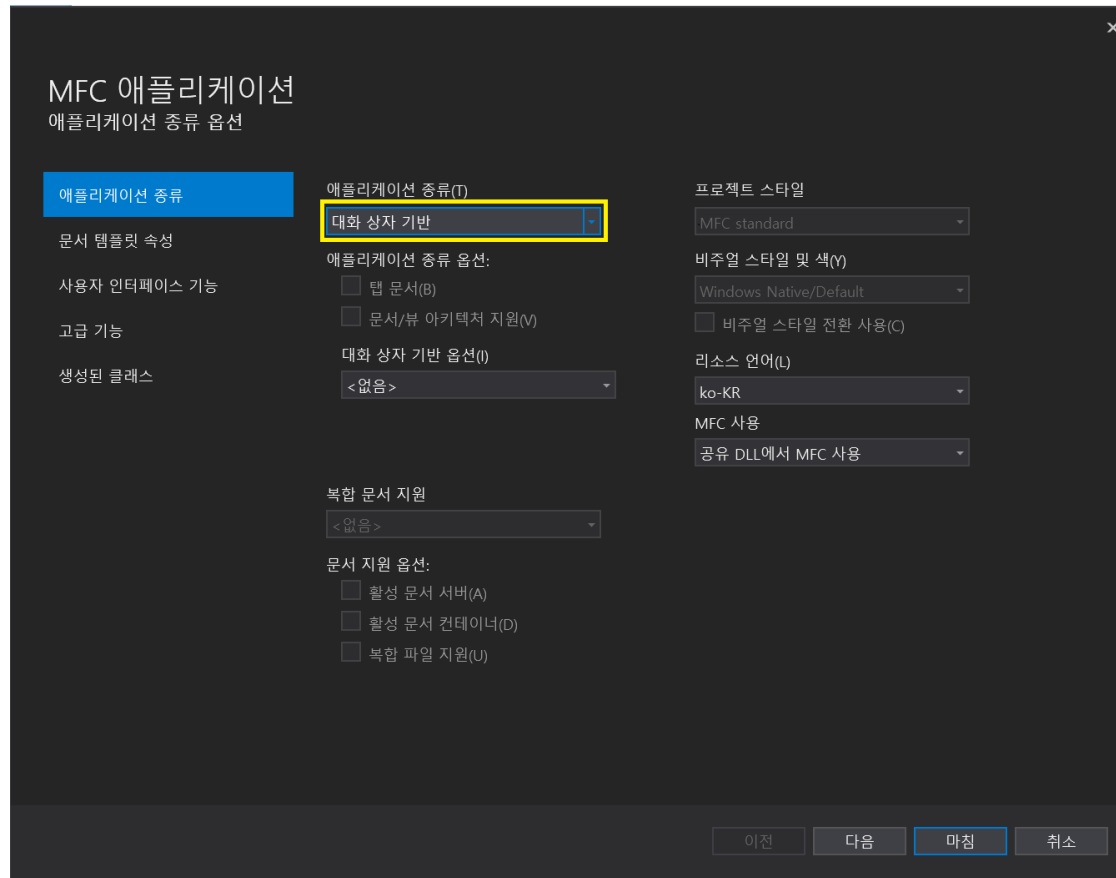
    case VK_LEFT:
        m_Pt.x -= 10;
        if (0 > m_Pt.x) m_Pt.x = m_ViewSize.cx;
        break;

    case VK_RIGHT:
        m_Pt.x += 10;
        if (m_ViewSize.cx < m_Pt.x) m_Pt.x = 0;
        break;

    }
    Invalidate(); // OnDraw() 함수 호출. 기본 bErase = 1
    CView::OnKeyDown(nChar, nRepCnt, nFlags);
}
```

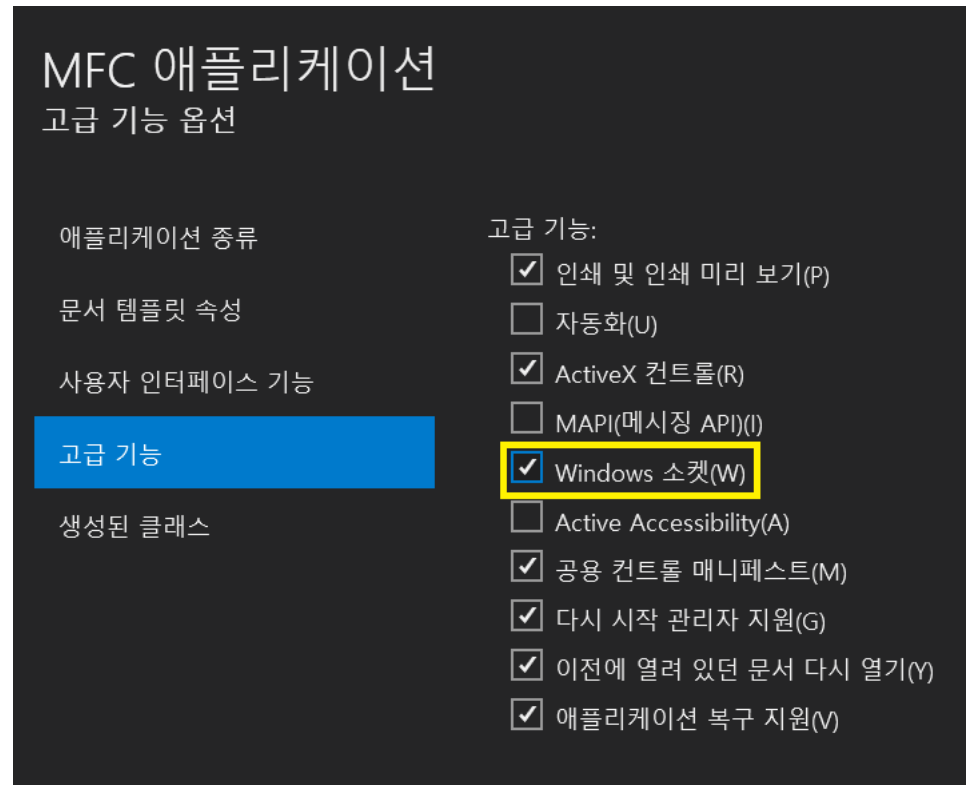

대화 상자(Dialog Box)

- ▶ 프로젝트를 [대화 상자 기반]으로 만든다



대화 상자(Dialog Box)

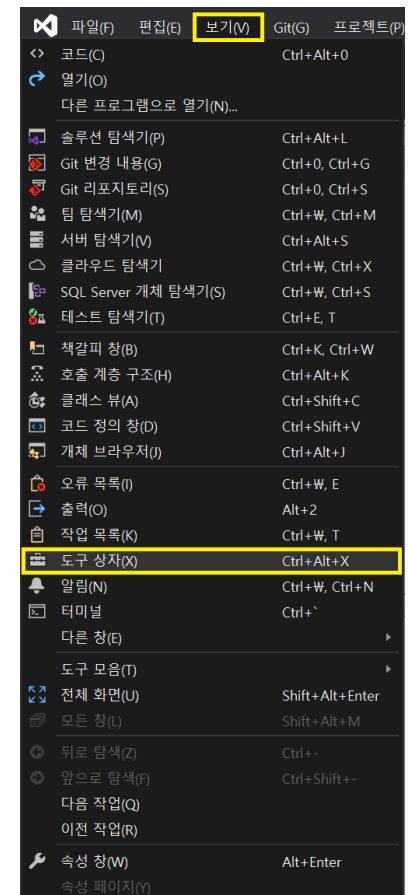
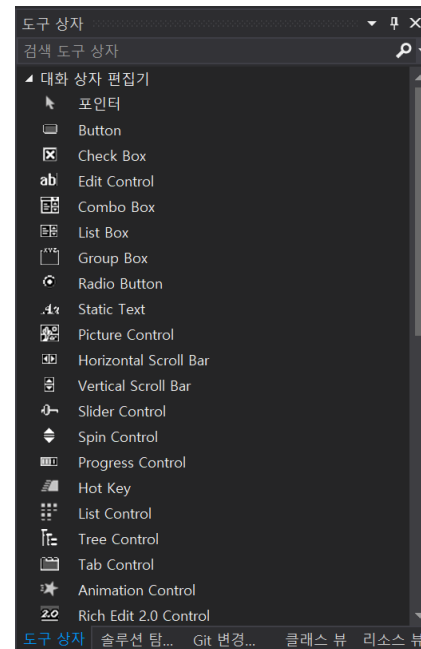
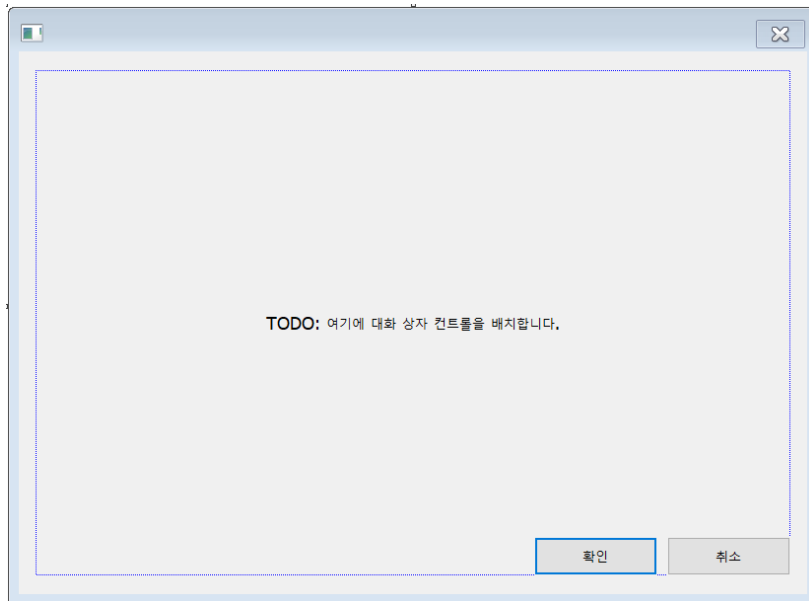
- ▶ Window Socket을 사용하기 위해서는 프로젝트 생성 때, 고급 기능에서 Window 소켓을 체크하여야 한다



대화 상자(Dialog Box)

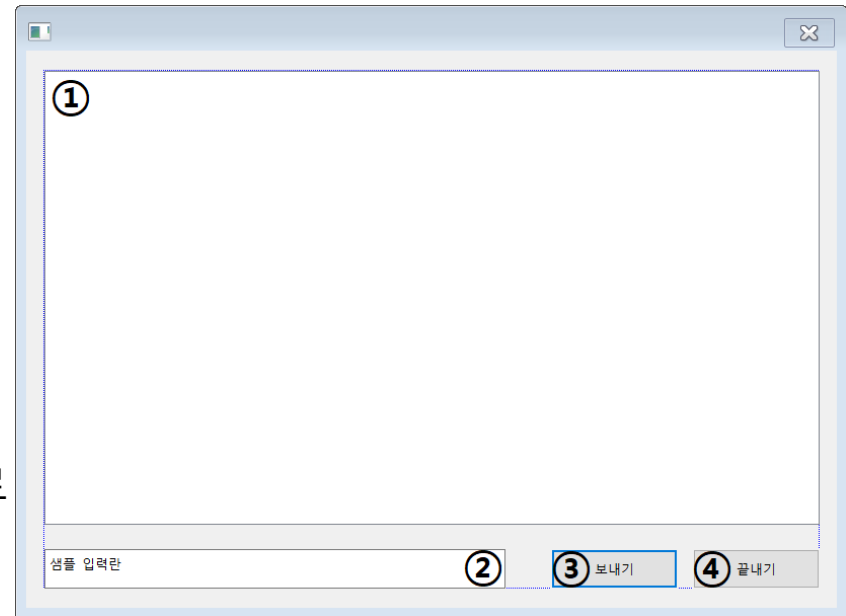
▶ 도구 상자

- 도구 상자에서 지원하는 여러 컨트롤들을 이용하여 대화 상자에 추가 할 수 있다
- 도구 상자는 [보기]의 [대화 상자]를 선택하면 추가할 수 있다



대화 상자(Dialog Box)

- ▶ Window 소켓을 추가하여 [대화 상자 기반]의 ClientSocket 프로젝트를 만든다
- ▶ 에디트 컨트롤(②)에 텍스트 작성
- ▶ 보내기 버튼(③)으로 내용을 입력
 - [확인] 버튼을 삭제하고 새로 만든다
- ▶ 입력된 내용(②)이 리스트 박스(①)에 출력
- ▶ 끝내기 버튼(④)을 눌러 애플리케이션을 종료



컨트롤 종류	번호	컨트롤 ID	캡션	비고
List Box	①	IDC_LIST_CHAT	(없음)	CListBox m_List / 정렬:False
Edit Control	②	IDC_EDIT_CHAT_TEXT	(없음)	(없음)
Button	③	IDC_BUTTON_SEND	보내기	기본 단추 : True
	④	IDCANCEL	끝내기	기본 단추 : False


대화 상자(Dialog Box)

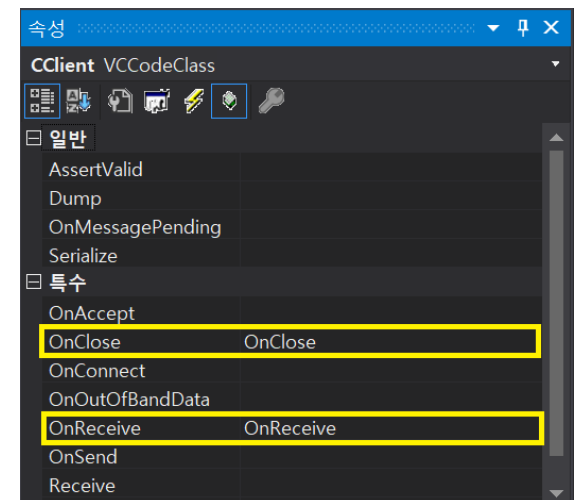
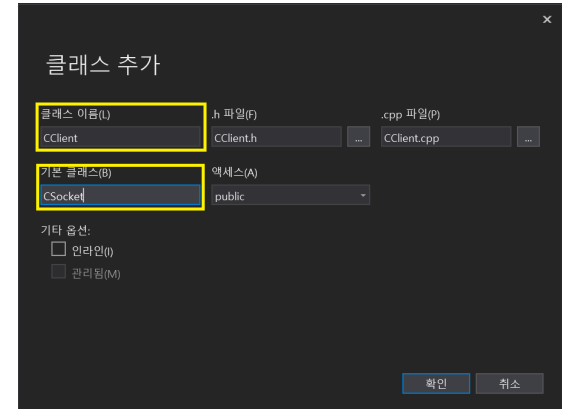
- ▶ **[변수 추가]**를 하면 해당 클래스에 자동으로 변수가 추가된다
 - 실제로 값을 쓰기 위해서는 DoDataExchange()에 DDX_Control(pDX, 컨트롤 ID, 변수명) 형식으로 등록 해야 한다([변수 추가]하기로 등록 할 경우 자동으로 등록된다)

```
class CClientSocketDlg : public CDialogEx
{
    ...
public:
    CListBox m_List;
};

void CClientSocketDlg::DoDataExchange(CDataExchange* pDX)
{
    CDialogEx::DoDataExchange(pDX);
    DDX_Control(pDX, IDC_LIST_CHAT, m_List);
}
```

대화 상자(Dialog Box)

- ▶ [클래스 뷰]에서 프로젝트를 마우스 우 클릭
- ▶ 메뉴의 [추가]-[클래스] 선택
- ▶ CSocket라는 기본 클래스를 지닌 CClient라는 클래스를 만든다
- ▶ [클래스 뷰]에서 생성한 CClient를 선택
- ▶ 속성 창에서 재정의의 를 선택여 OnClose와 OnRecieve를 추가



대화 상자(Dialog Box)

▶ 클라이언트 소켓이 닫히면 자동으로 호출된다

```
void CClient::OnClose(int nErrorCode)
{
    // ShutDown() 함수와 Close() 함수는 기본으로 있는 함수로, 그냥 추가하면 된다.
    ShutDown();
    Close();

    CSocket::OnClose(nErrorCode);

    AfxMessageBox(_T("ERROR : Disconnected From Server!!"));
}
```

▶ 서버에서 데이터를 받으면 자동으로 호출된다

```
#include "CClientSocketDlg.h"
void CClient::OnReceive(int nErrorCode)
{
    // TODO: 여기에 특수화된 코드를 추가 및/또는 기본 클래스를 호출합니다.
    // 서버에서 받은 데이터 처리를 여기서 하면 된다.
    char buf[MAX_BUFFER_SIZE + 1];
    ZeroMemory(buf, sizeof(buf));
    if (Receive(buf, sizeof(buf)) > 0)
    {
        // Dlg 클래스에 선언 해둔 멤버 변수 등을 가져와서 쓸 수 있다.
        CClientSocketDlg * pMain = (CClientSocketDlg *)AfxGetMainWnd();
        ...
    }

    CSocket::OnReceive(nErrorCode);
}
```

대화 상자(Dialog Box)

- ▶ 대화 상자 클래스에 새로 만든 **CClient** 객체를 추가한다

```
class CClientSocketDlg : public CDialogEx
{
    ...
public:
    ...
    CClient m_Client;
};
```

- ▶ 서버와의 연결시도는 OnInitDialog()에서 처리한다

```
BOOL CClientSocketDlg::OnInitDialog()
{
    ...

    // TODO: 여기에 추가 초기화 작업을 추가합니다.
    m_Client.Create(); // 클라이언트 소켓 생성.
    m_Client.Connect(_T("127.0.0.1"), 9000); // 서버의 IP 주소와 포트 번호를 이용하여 연결 시도.

    return TRUE; // 포커스를 컨트롤에 설정하지 않으면 TRUE를 반환합니다.
}
```


대화 상자(Dialog Box)

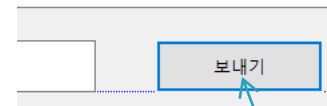
- ▶ 사용할 컨트롤을 더블 클릭을 하면 자동으로 해당 함수가 등록/추가 된다

```
class CClientSocketDlg : public CDialogEx
{
    ...
public:
    ...
    afx_msg void OnBnClickedButtonSend();
};

void CClientSocketDlg::OnBnClickedButtonSend()
{
    // 여기에서 서버로 보낼 데이터 처리를 하면 된다.
    CString msg;
    GetDlgItemText(IDC_EDIT_CHAT_TEXT, msg);           // Edit Control에 쓴 내용은 가져온다.
    if (!msg.IsEmpty())                                 // 가져온 내용이 있는지 확인.
    {
        m_List.AddString(msg);                          // 내용이 있다면 해당 문자를 List Box에서 출력.
        SetDlgItemText(IDC_EDIT_CHAT_TEXT, _T(""));    // Edit Control에 적혀있는 내용을 지운다.

        // CString으로 받은 문자열을 아래와 같이 버퍼에 넣을 수 있다.
        char buf[10];
        CW2A message(msg.GetString());
        strcpy_s(buf, sizeof(buf), message.m_szBuffer);

        // m_Client.Send() 함수를 이용하여 데이터를 전송할 수 있다.
        m_Client.Send(buf, sizeof(buf));
    }
}
```



Double Click!!