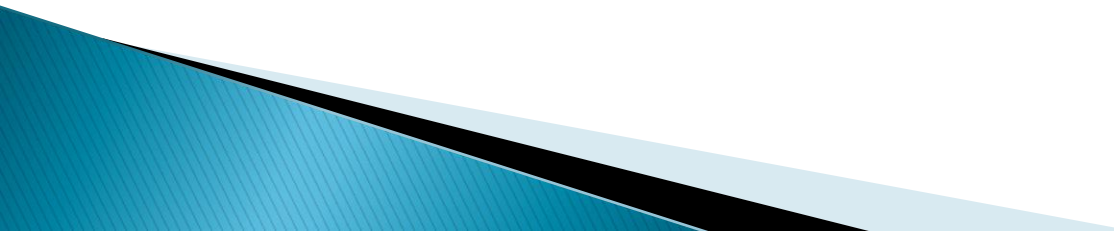


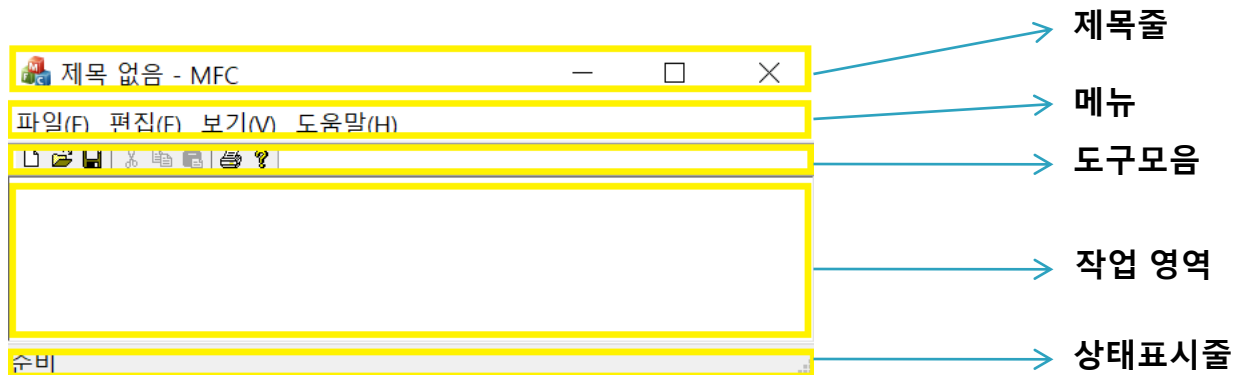
MFC

목차

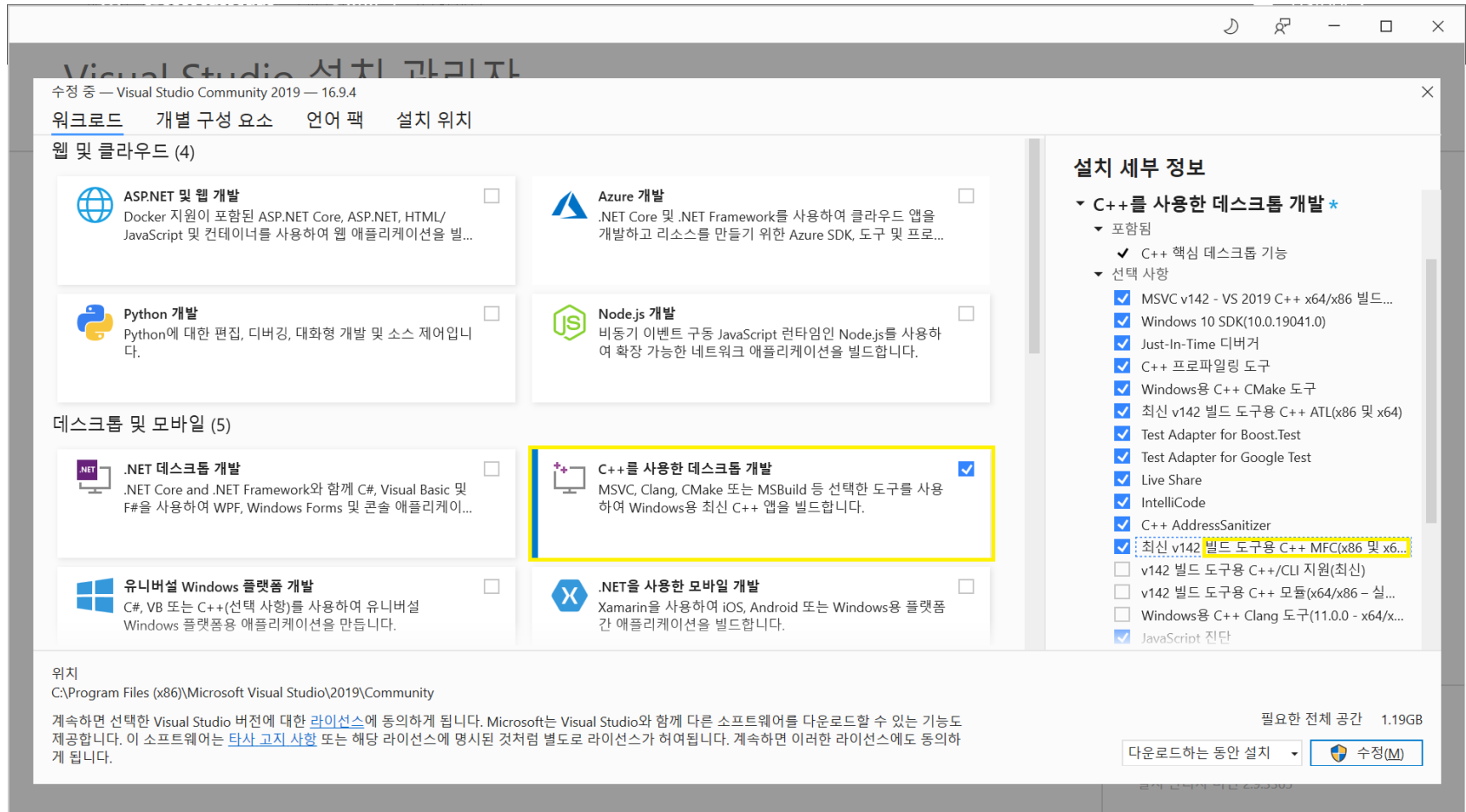
1. MFC란? . . . 3p
 2. MFC 프로젝트 만들기 . . . 4p
 3. OnDraw . . . 7p
 4. 윈도우 메시지 . . . 10p
 5. 대화 상자(Dialog Box) . . . 17p
 6. nonblocking Socket 채팅 클라이언트 . . . 20p
- 

MFC란?

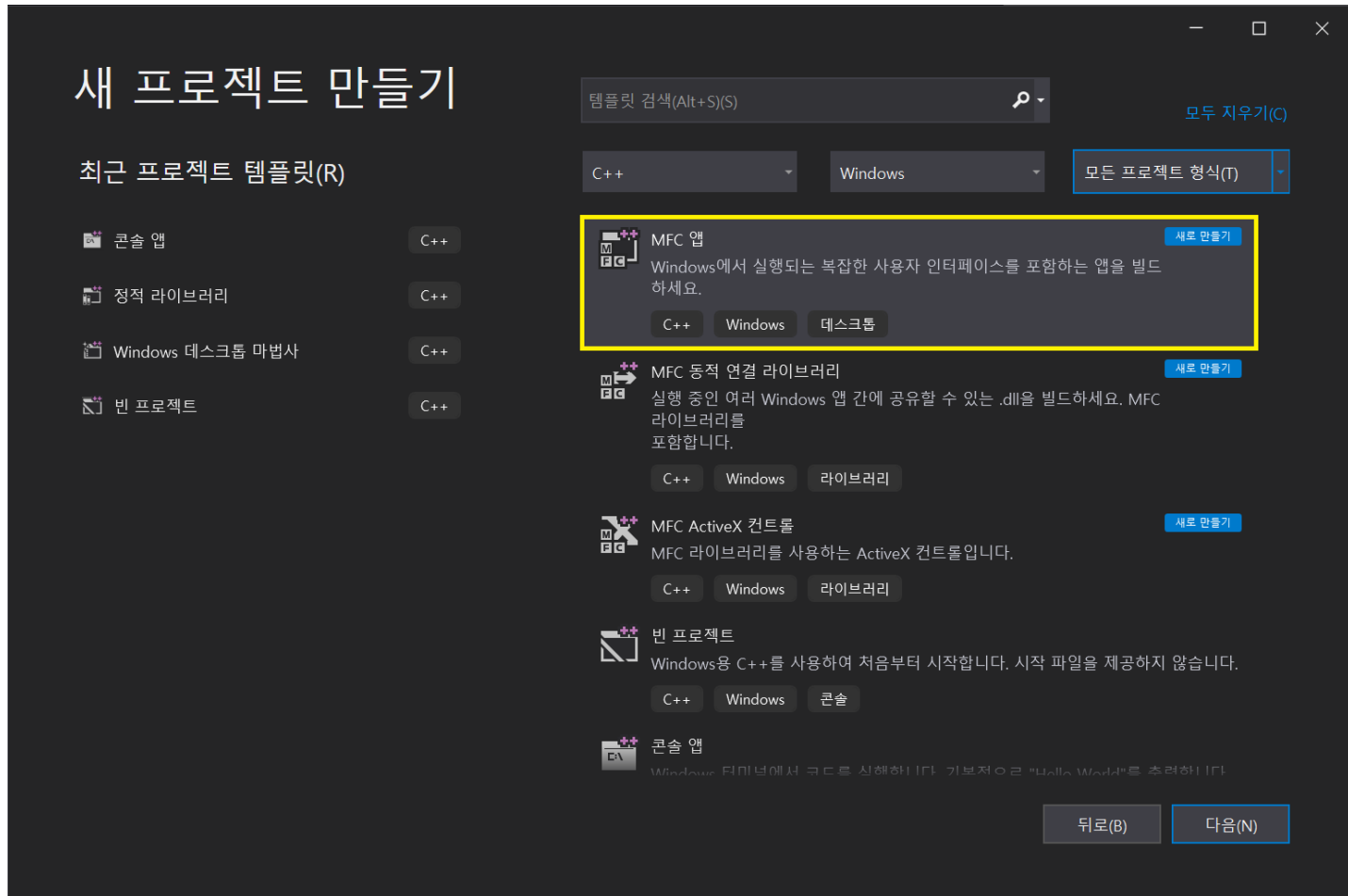
- ▶ Microsoft Foundation Class의 줄임말로 **마이크로소프트사에서 윈도우 애플리케이션을 제작**하라고 제공하는 **C++ 클래스 라이브러리**의 집합
- ▶ 특징으로는 **GUI(Graphic User Interface)**를 제공



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(V)
Windows Native/Default

☐ 비주얼 스타일 전환 사용(C)

리소스 언어(L)
ko-KR

MFC 사용
공유 DLL에서 MFC 사용

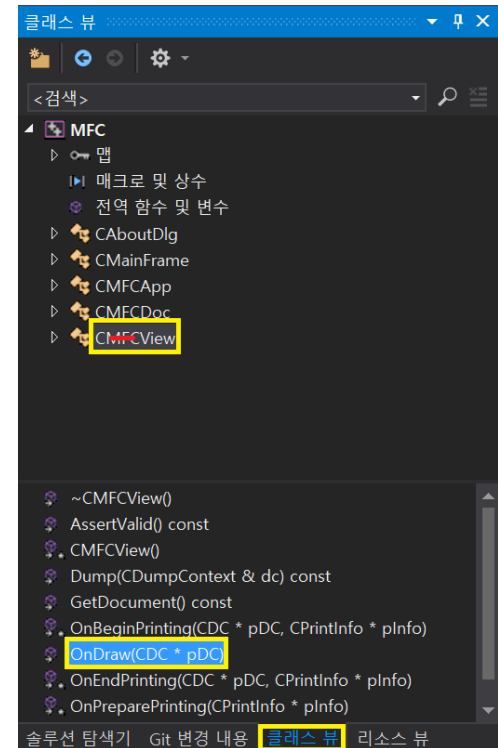
이전 다음 마침 취소

OnDraw

▶ View Class

- 화면의 출력 담당
- 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.  
}
```



OnDraw

▶ 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 Print1"));
    ReleaseDC(pdc);

    CClientDC dc(this);
    dc.TextOutW(100, 100, _T("MFC Application Print2"));

    pDC->TextOutW(100, 150, _T("MFC Application Print3"));
}
```


OnDraw

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

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

- ▶ **_T()**

- **유니코드 플랫폼 환경에서 사용될 문자열 형태로 변환**하기 위하여 사용하는 매크로 함수

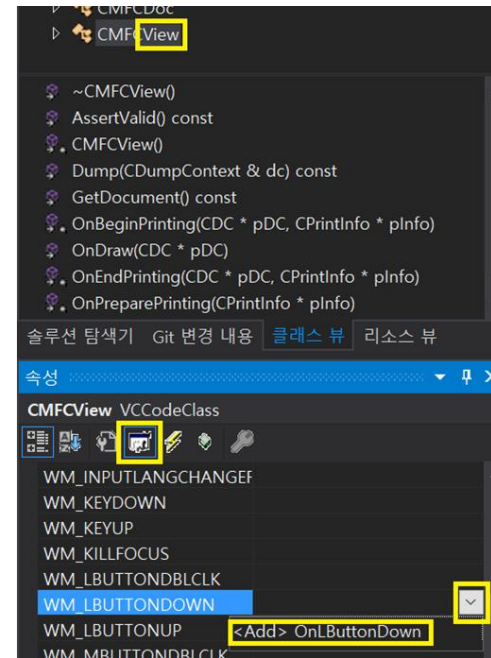
- ▶ **유니코드**

- 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**

윈도우 메시지

▶ WM_LBUTTONDOWN

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



```
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);
```

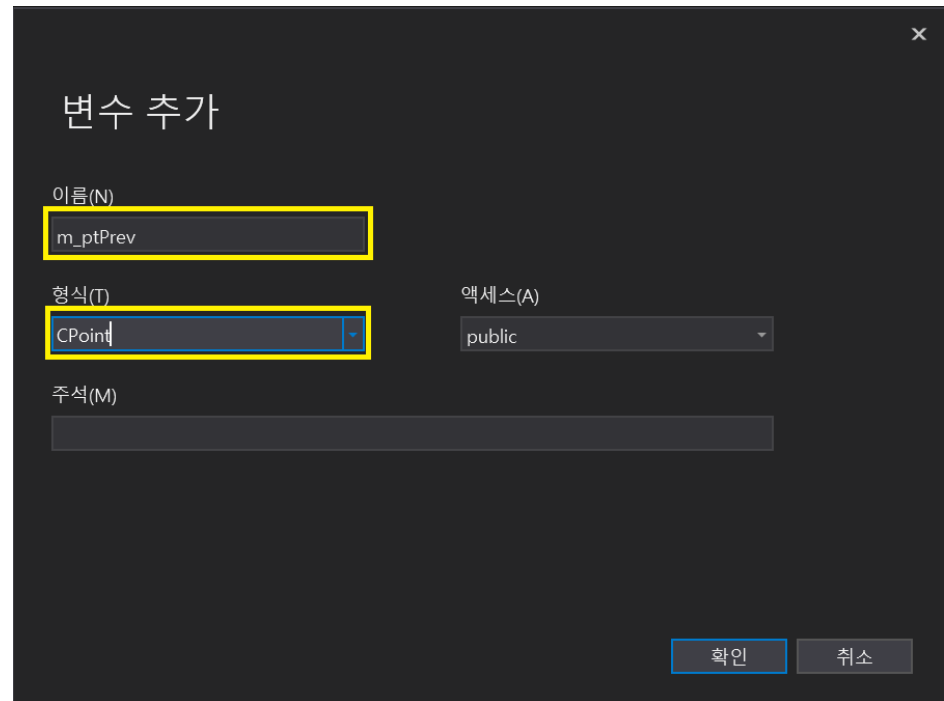
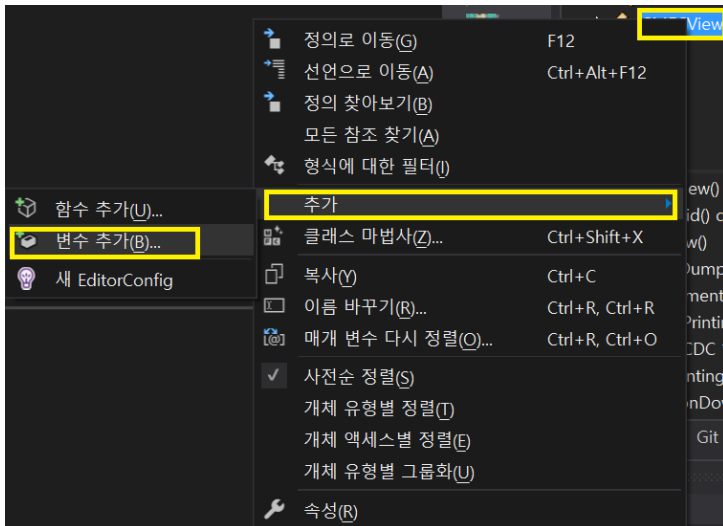
```
}
```

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

윈도우 메시지

▶ 변수 추가

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



윈도우 메시지

▶ 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

윈도우 메시지

```
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);
}
```

윈도우 메시지

▶ WM_KEYDOWN

- 키보드 입력 시 호출
- 방향키에 따라 사각형을 이동하는 예제 작성

▶ WM_SIZE

- 윈도우의 크기가 변경되면 호출

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

윈도우 메시지

```
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);
}

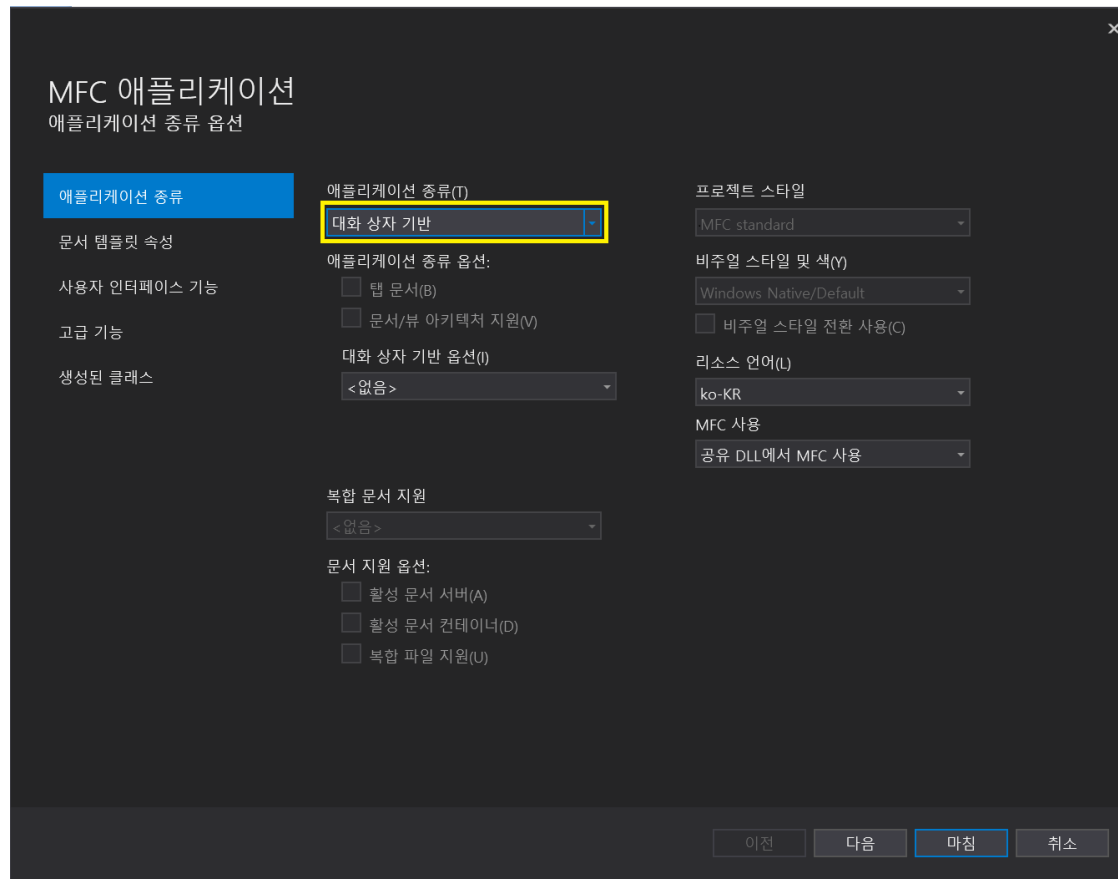
void CMFCView::OnSize(UINT nType, int cx, int cy) // WM_SIZE
{
    CView::OnSize(nType, cx, cy);
    // TODO: 여기에 메시지 처리기 코드를 추가합니다.
    m_ViewSize = CSize(cx, cy); // 클라이언트 영역의 크기를 저장.
    m_Pt = CPoint(cx * 0.5f, cy * 0.5f); // 사각형이 그려질 위치를 화면의 중앙으로 한다.
}
```

윈도우 메시지

```
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)

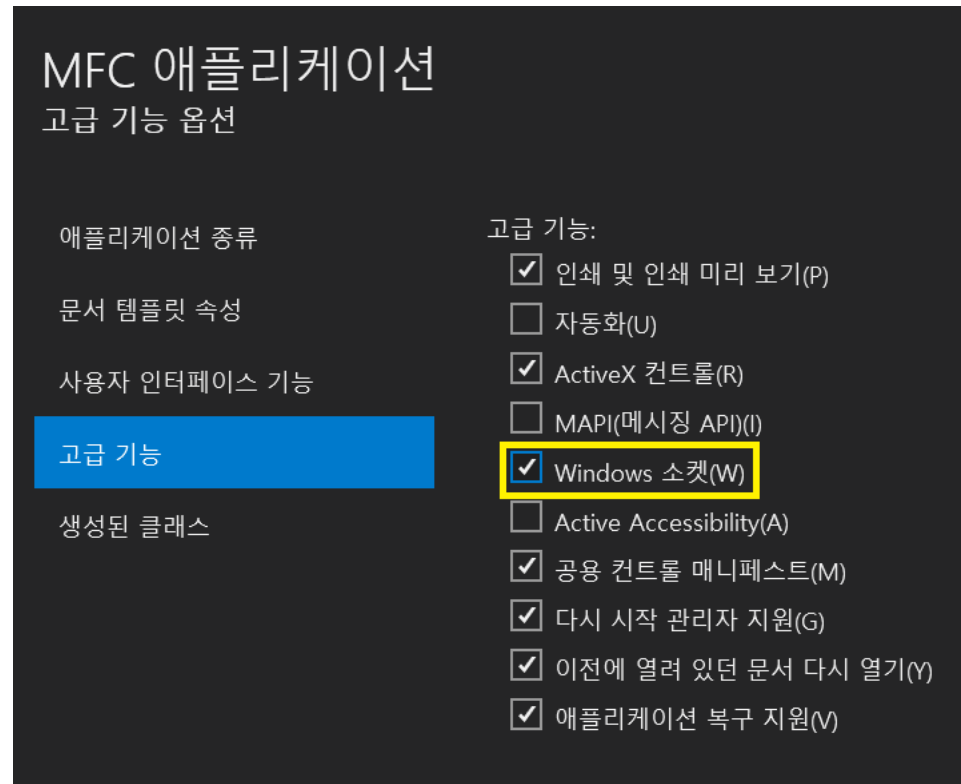
▶ [대화 상자 기반] 프로젝트 만들기



대화 상자(Dialog Box)

▶ Window Socket

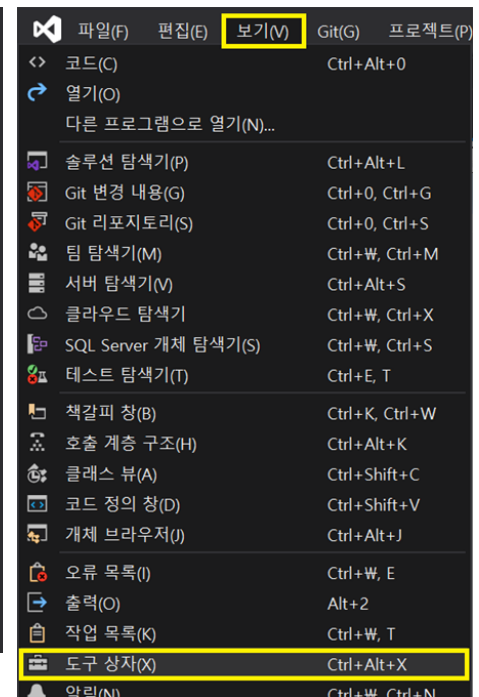
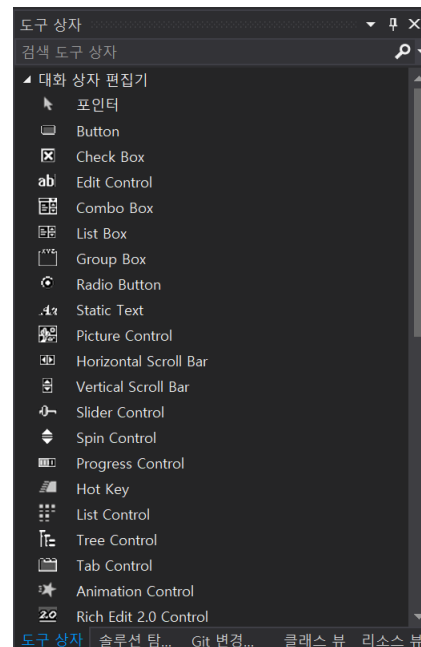
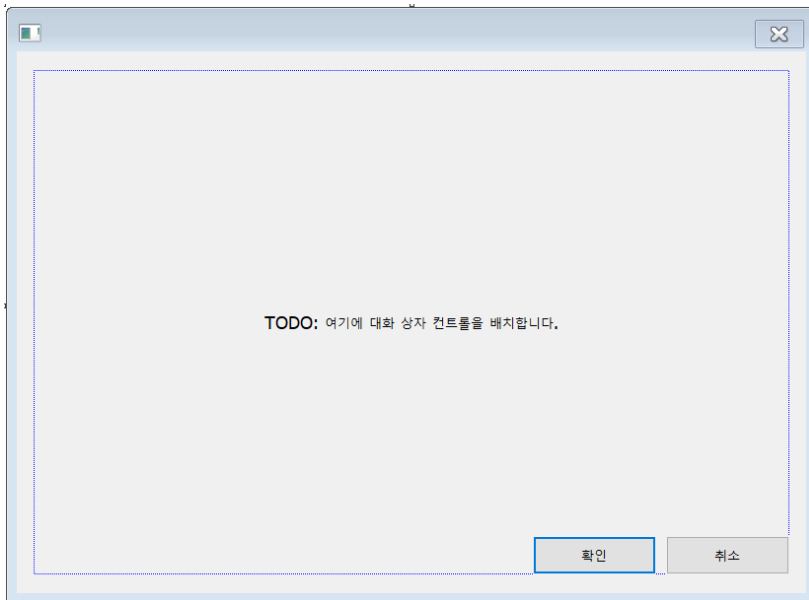
- MFC에서는 **Nonblocking Socket**을 사용
- Socket을 사용하기 위해서 [고급 기능] => [Windows 소켓] 항목을 **체크**



대화 상자(Dialog Box)

▶ 도구 상자

- [도구 상자]는 [보기]>[도구 상자]를 선택하여 프로젝트에 추가
- [도구 상자]에서 지원하는 여러 [컨트롤]들을 이용하여 [대화 상자]에 추가



nonblocking Socket 채팅 클라이언트

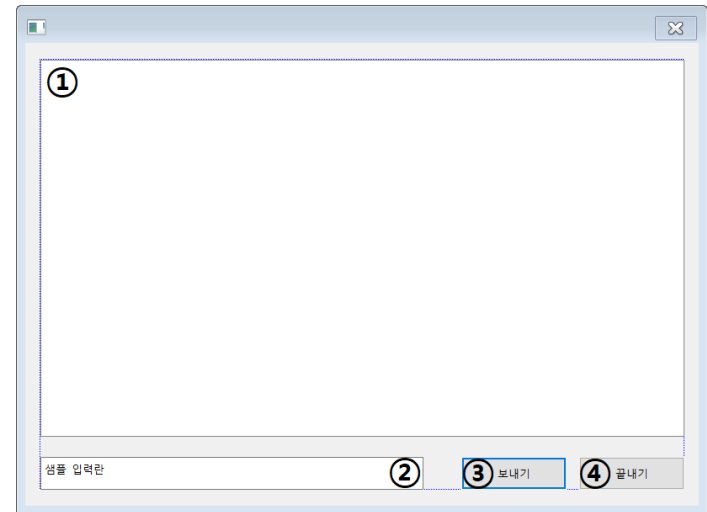
▶ 채팅 클라이언트 [대화 상자] 구성

1. Static Text(커트룰) 삭제 *(DOTO: 여기에 대화 상자 컨트롤을 배치합니다.)*

2. 리스트 박스(①) 추가
3. 속성에서 컨트롤 ID를 IDC_LIST_CHAT으로 변경
4. 속성에서 정렬을 False로 변경
5. 마우스 오른쪽 클릭, [변수 추가]를 선택
6. 에디트 커트룰(②) 추가
7. 속성에서 컨트롤 ID를 IDC_EDIT_CHAT_TEXT로 변경

8. [확인] 버튼을 삭제
9. 보내기 버튼(③) 추가
10. 속성에서 컨트롤 ID를 IDC_BUTTON_SEND로 변경
11. 속성에서 기본 단추를 True로 변경
12. 속성에서 캡션을 [보내기]로 변경

13. 취소 버튼(④) 선택
14. 속성에서 기본 단추를 False로 변경
15. 속성에서 캡션을 [끝내기]로 변경



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


nonblocking Socket 채팅 클라이언트

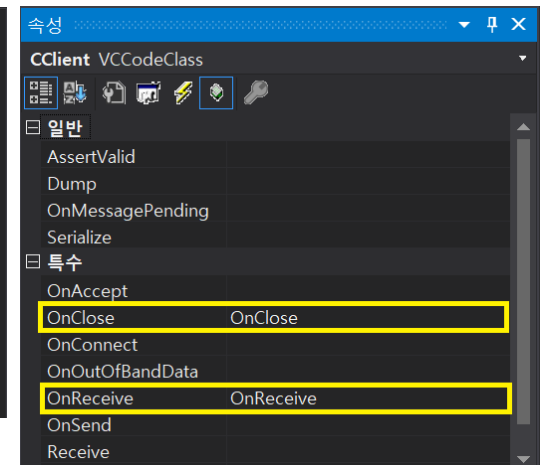
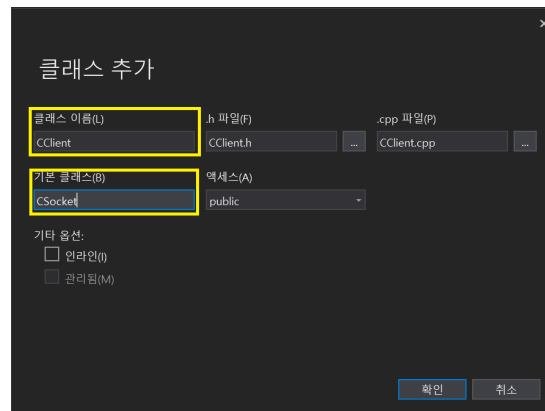
▶ 버퍼 사이즈 매크로 추가

- 프로젝트명(ClientSocket)Dlg.h 파일에 코드 작성

```
#define MAX_BUFFER_SIZE 256
```

▶ Client Class 만들기

1. [클래스 뷰]에서 프로젝트를 마우스 우 클릭
2. 메뉴의 [추가]-[클래스] 선택
3. **CSocket**라는 기본 클래스를 지닌 **CClient**라는 클래스를 만든다
4. [클래스 뷰]에서 생성한 CClient를 선택
5. 속성 창에서 재정의()를 선택여 **OnClose**와 **OnRecieve**를 추가



nonblocking Socket 채팅 클라이언트

```
#include "CClientSocketDlg.h"

void CClient::OnClose(int nErrorCode) // 클라이언트 소켓이 닫히면 자동으로 호출
{
    // ShutDown() 함수와 Close() 함수는 기본으로 있는 함수로, 그냥 추가하면 된다.
    ShutDown();
    Close();

    CSocket::OnClose(nErrorCode);

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

void CClient::OnReceive(int nErrorCode) // 서버에서 데이터를 받으면 자동으로 호출
{
    // TODO: 여기에 특수화된 코드를 추가 및/또는 기본 클래스를 호출합니다.
    // 서버에서 받은 데이터를 여기서 하면 된다.
    char buf[MAX_BUFFER_SIZE + 1];
    ZeroMemory(buf, sizeof(buf));
    if (Receive(buf, sizeof(buf)) > 0)
    {
        // TODO : 서버에서 받은 데이터 처리.

        CClientSocketDlg * pMain = (CClientSocketDlg *)AfxGetMainWnd(); // Dlg 클래스에 선언 해둔 멤버 변수 등을 가져와서 쓸 수 있다.
        // pMain->m_List.AddString(buf); // 서버에서 받은 데이터를 리스트 박스 컨트롤에 출력.
    }

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

nonblocking Socket 채팅 클라이언트

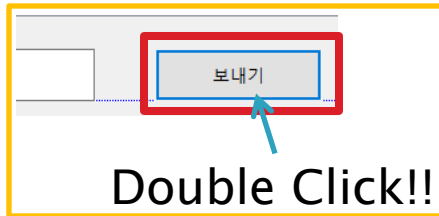
```
// CClientSocketDlg.h
class CClientSocketDlg : public CDialogEx
{
    ...
public:
    ...
    CClient m_Client; // CClient 변수 추가.
};

// CClientSocketDlg.cpp
BOOL CClientSocketDlg::OnInitDialog() // 대화 상자 초기화 함수.
{
    ...

    // 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.
}
```

nonblocking Socket 채팅 클라이언트



```
void CClientSocketDlg::OnBnClickedButtonSend() // 보내기 버튼 더블 클릭 시, 자동 추가.
{
    // 함수명은 컨트롤 ID를 참조하여 자동으로 정해진다.
    // TODO : 서버로 보낼 데이터 처리.
    CString msg;
    GetDlgItemText(IDC_EDIT_CHAT_TEXT, msg); // Edit Control에 쓴 내용은 가져온다.
    if (!msg.IsEmpty()) // 가져온 내용이 있는지 확인.
    {
        SetDlgItemText(IDC_EDIT_CHAT_TEXT, _T("")); // Edit Control에 적혀있는 내용을 지운다.

        // CString으로 받은 문자열을 아래와 같이 버퍼에 넣을 수 있다.
        char buf[MAX_BUFFER_SIZE + 1];
        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));
    }
}
```