Window Programming

Visual C++ MFC Programming

Lecture 04

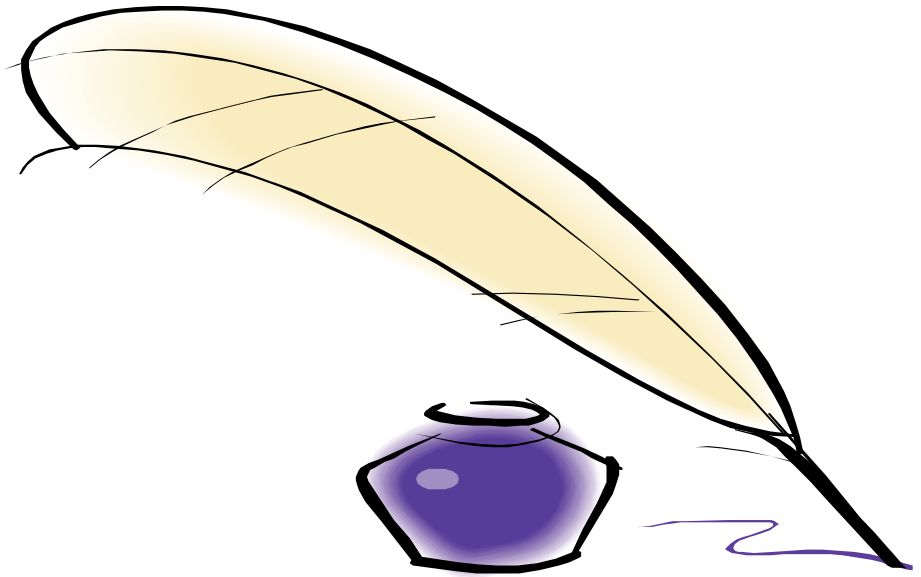
김예진
Dept. of Game Software

Announcement

- 03/16: HW #1 on ClassNet (Due: 03/24), Avg: 8.1

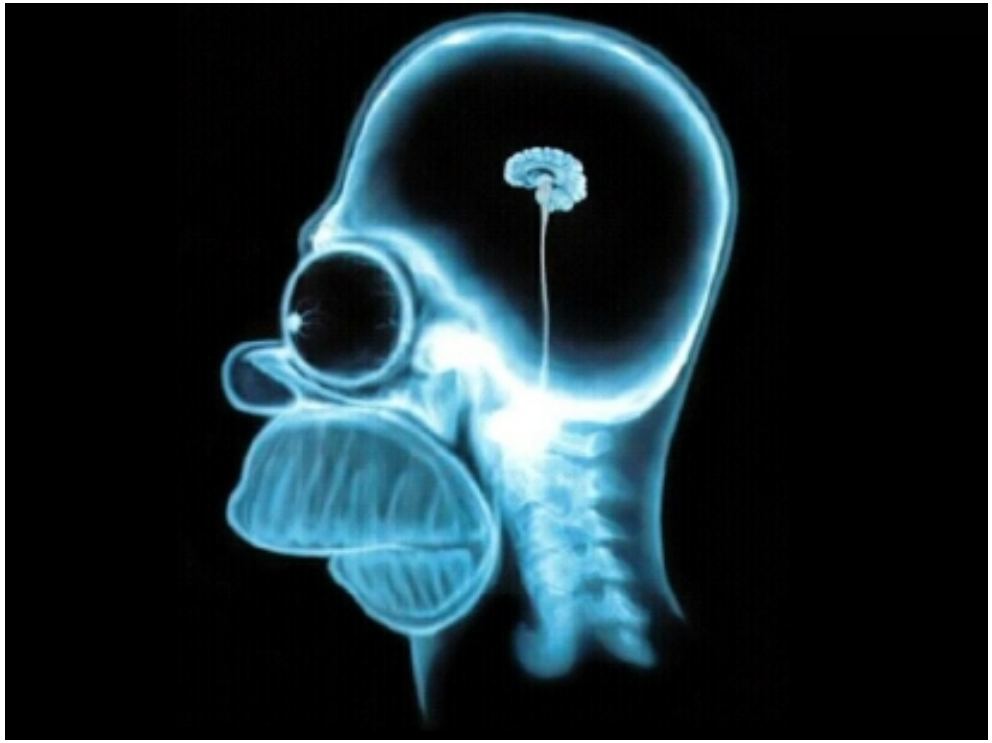
Plan

- MFC 프로그램 시작하기
  - MFC 기초 클래스
  - MFC 프로그램 구조
  - MFC와 Device Context



Look into the codes

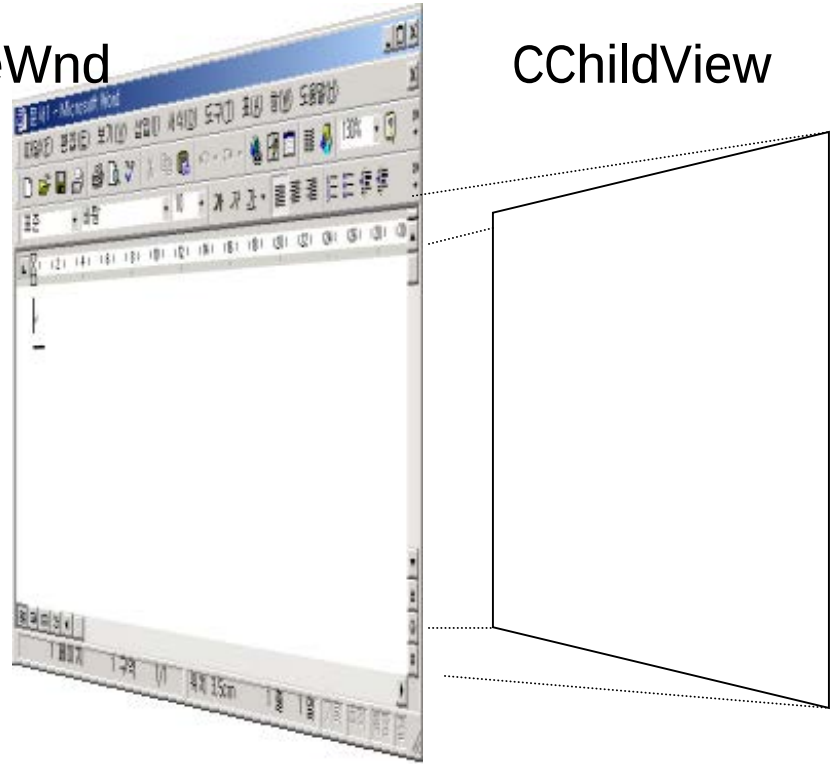
classes



VC++ Framework

CFrameWnd

CChildView



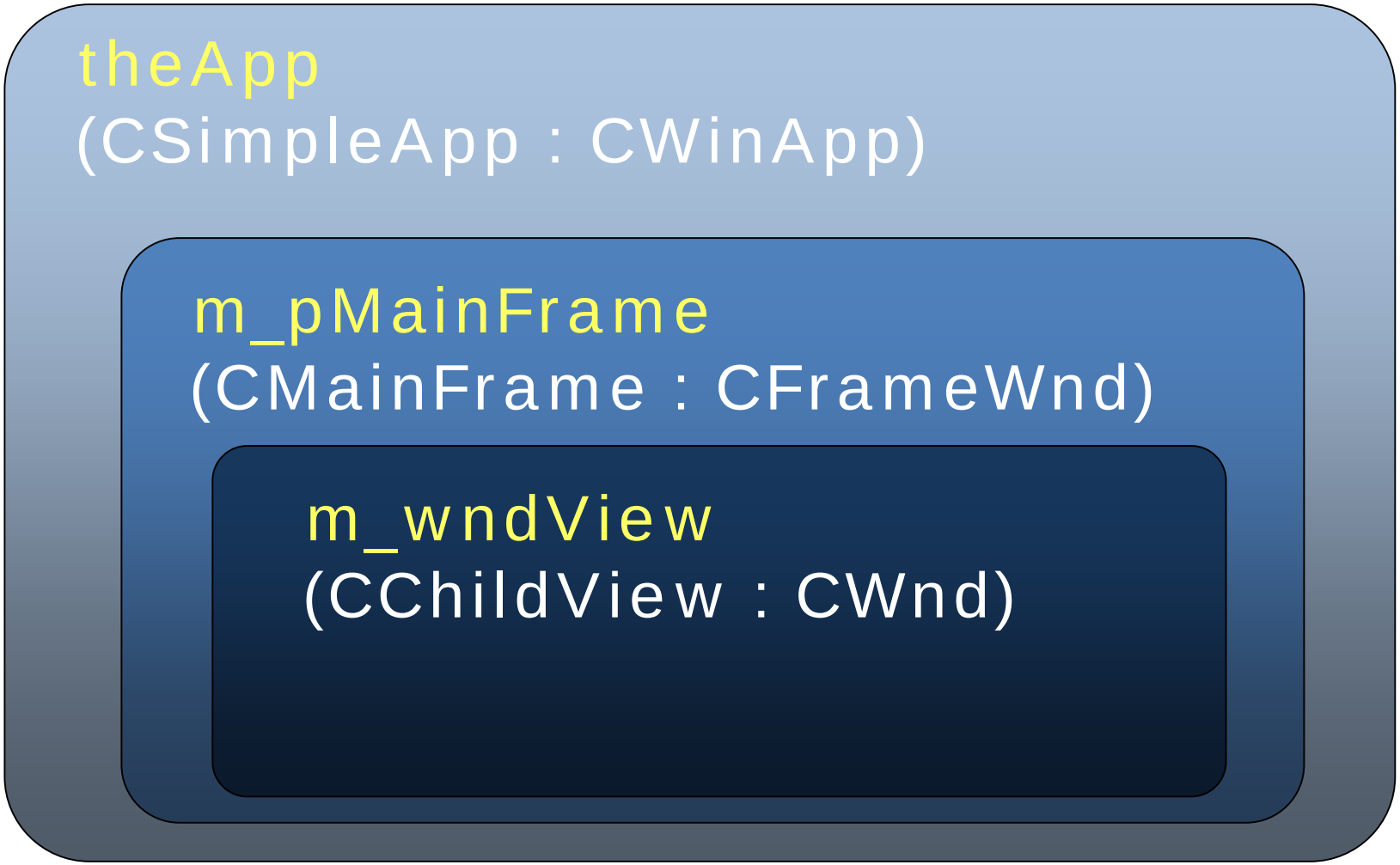
윈도우의 프레임(틀)을 관리

데이터를 보여주는 윈도우

CWinApp

- 위의 두 오브젝트를 묶어 주고, 프로그램을 구동 시킴(눈에 안보임)
- 메시지 루프를 돌림

프로그램 내부 구조


theApp
(CSimpleApp : CWinApp)
m_pMainFrame
(CMainFrame : CFrameWnd)
m_wndView
(CChildView : CWnd)

응용 프로그램 실행순서

```
CSimpleApp   theApp;

WinMain()                              // MFC 내부에 숨겨짐
{

    theApp.InitInstance();             // 초기화

    theApp.Run();                      // 메시지 루프

    theApp.ExitInstance();             // 종료

}
```

MFC: 미리 약속된 맴버 함수들이 필요한 순간에 순서대로 호출됨.

➔ 함수들의 재정의를 통해 원하는 작업을 수행하게 함.

응용 프로그램 클래스 (1/4)

```
// SimpleMFC.h
class CSimpleMFC : public CWinApp
{
    public:
        CSimpleMFC();                       // 클래스 생성자
        virtual BOOL InitInstance();        // 초기화
        afx_msg void OnAppAbout();          // About 메시지 핸들러
        DECLARE_MESSAGE_MAP()               // 1개이상의 메시지 핸들러
};
```

응용 프로그램 클래스 (2/4)

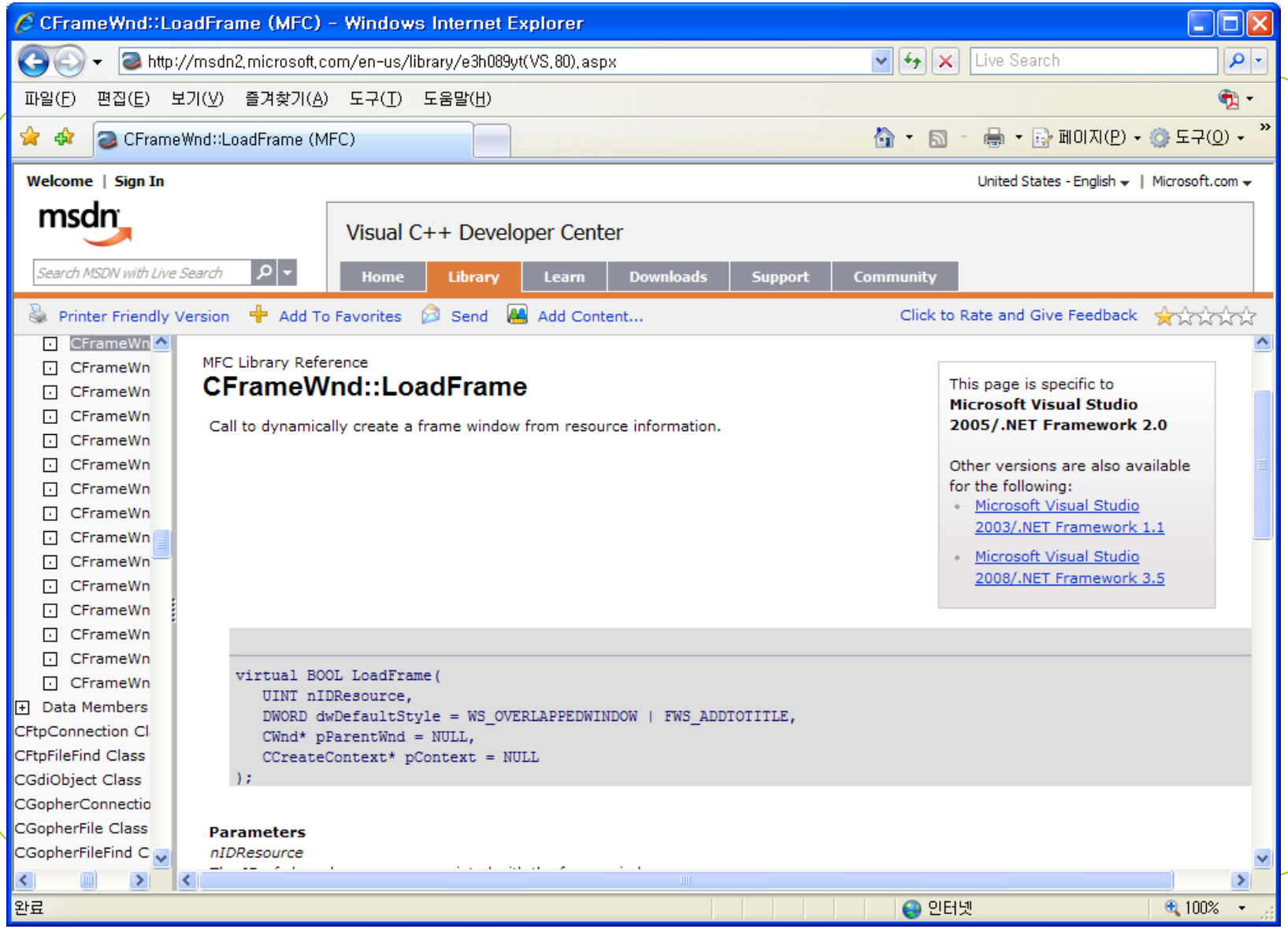
```
#include "stdafx.h"
#include "simple.h"
#include "MainFrm.h“

// Simple.cpp
BEGIN_MESSAGE_MAP(CSimpleApp, CWinApp)
    ON_COMMAND(ID_APP_ABOUT, OnAppAbout)
END_MESSAGE_MAP()

CSimpleApp::CSimpleApp()
{
}

CSimpleApp theApp;        // 응용프로그램 자신에 해당하는 전역객체
```

MFC 기본 header file을 모아놓음

응용 프로그램 클래스 (3/4)

```
BOOL CSimpleApp::InitInstance()
{
    SetRegistryKey(_T("Local AppWizard-Generated
    Applications"));

    CMainFrame* pFrame = new CMainFrame;
    m_pMainWnd = pFrame;

    pFrame->LoadFrame(IDR_MAINFRAME,
        WS_OVERLAPPEDWINDOW | FWS_ADDTOTITLE, NULL,
        NULL);

    pFrame->ShowWindow(SW_SHOW);
    pFrame->UpdateWindow();

    return TRUE;
}
```

응용 프로그램 클래스 (3/4)



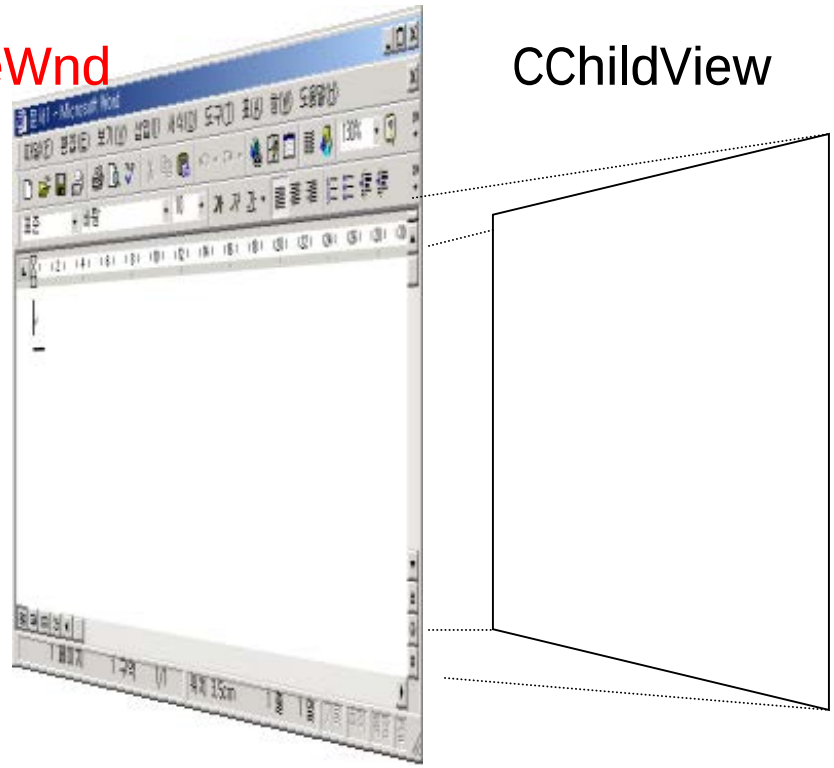
응용 프로그램 클래스 (4/4)

```
// 대화상자 관련 클래스 선언 및 정의 부분 – 생략
// ...

void CSimpleApp::OnAppAbout()
{
    CAboutDlg aboutDlg;
    aboutDlg.DoModal();
}
```

응용 프로그램 실행순서

```
CSimpleApp theApp;

WinMain()                          // MFC 내부에 숨겨짐
{

    theApp.InitInstance();         // 초기화

    theApp.Run();                  // 메시지 루프

    theApp.ExitInstance();         // 종료

}
```

VC++ Framework



- 위의 두 오브젝트를 묶어 주고, 프로그램을 구동 시킴(눈에 안보임)
- 메시지 루프를 돌림

프레임 윈도우 클래스 (1/5)

```
// MainFrm.h
class CMainFrame : public CFrameWnd
{
      public:
         CMainFrame();

      protected:
         DECLARE_DYNAMIC(CMainFrame);

      public:
         virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
         virtual BOOL OnCmdMsg(UINT nID, int nCode, void* pExtra,
            AFX_CMDHANDLERINFO* pHandlerInfo);
         virtual ~CMainFrame();
         CChildView m_wndView;

      protected:
         afx_msg int OnCreate(LPCREATESTRUCT lpCreateStruct);
         afx_msg void OnSetFocus(CWnd *pOldWnd);
         DECLARE_MESSAGE_MAP()
};
```

프레임 윈도우 클래스 (1/5)

- 미리 정의된(약속된) 함수들의 재정의:

```
virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
```

  - 프로그램 실행 초기에 해야될 일을 정의함

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

  - 프로그램 창이 생기기 직전에 해야할 일을 정의함

On으로 시작하는 함수 이름의 경우: Message Handler

예) OnCreate ← WM_CREATE 메시지의 메시지 핸들러
OnPaint ← WM_PAINT 메시지의 메시지 핸들러

프레임 윈도우 클래스 (2/5)

```
// MainFrm.cpp
IMPLEMENT_DYNAMIC(CMainFrame, CFrameWnd)

BEGIN_MESSAGE_MAP(CMainFrame, CFrameWnd)
    ON_WM_CREATE()
    ON_WM_SETFOCUS()
END_MESSAGE_MAP()

CMainFrame::CMainFrame()
{
}

CMainFrame::~CMainFrame()
{
}
```

프레임 윈도우 클래스 (3/5)

```
int CMainFrame::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
    if (CFrameWnd::OnCreate(lpCreateStruct) == -1)
        return -1;
    if (!m_wndView.Create(NULL, NULL, AFX_WS_DEFAULT_VIEW,
        CRect(0, 0, 0, 0), this, AFX_IDW_PANE_FIRST, NULL))
    {
        TRACE0("Failed to create view window\n");
        return -1;
    }

    return 0;
}
```

프레임 윈도우 클래스 (4/5)

```
BOOL CMainFrame::PreCreateWindow(CREATESTRUCT& cs)
{
    if( !CFrameWnd::PreCreateWindow(cs) )
        return FALSE;
    cs.dwExStyle &= ~WS_EX_CLIENTEDGE;
    cs.lpszClass = AfxRegisterWndClass(0);
    return TRUE;
}
```

```
struct CREATESTRUCT{
    LPVOID lpCreateParams;
    HINSTANCE hInstance;
    HMENU hMenu;
    HWND hwndParent;
    int cy;
    int cx;
    int y;
    int x;
    LONG style;
    LPCTSTR lpszName;
    LPCTSTR lpszClass;
    DWORD dwExStyle;
};
```

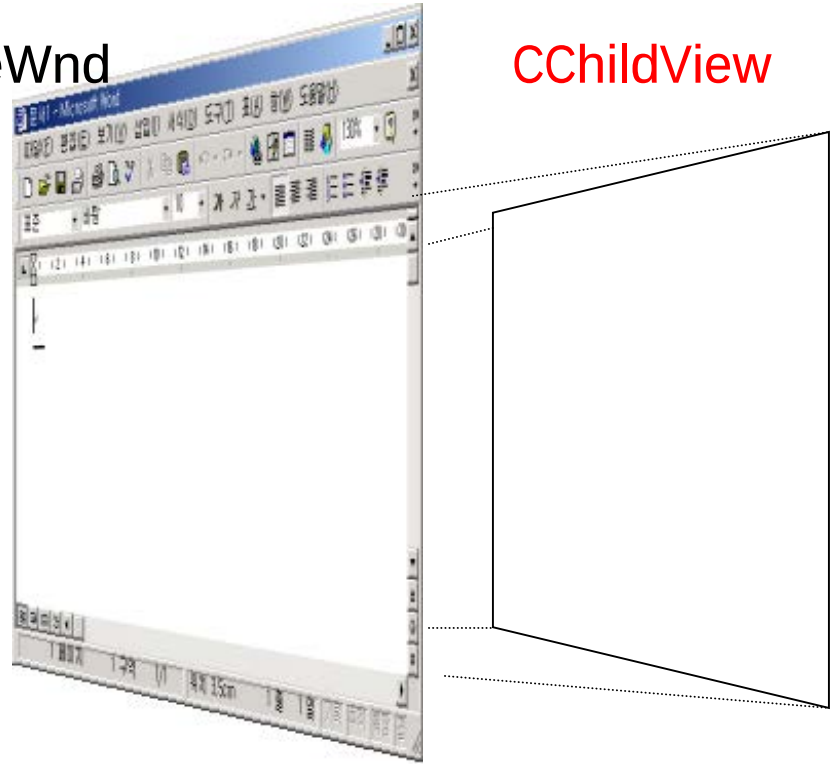
프레임 윈도우 클래스 (5/5)

```
void CMainFrame::OnSetFocus(CWnd* pOldWnd)
{
    m_wndView.SetFocus();
}


BOOL CMainFrame::OnCmdMsg(UINT nID, int nCode, void*
    pExtra, AFX_CMDHANDLERINFO* pHandlerInfo)
{
    if (m_wndView.OnCmdMsg(nID, nCode, pExtra,
    pHandlerInfo))
        return TRUE;

    return CFrameWnd::OnCmdMsg(nID, nCode, pExtra,
    pHandlerInfo);
}
```

VC++ Framework

CFrameWnd

CChildView



윈도우의 프레임(틀)을 관리

데이터를 보여주는 윈도우

CWinApp

- 위의 두 오브젝트를 묶어 주고, 프로그램을 구동 시킴(눈에 안보임)
- 메시지 루프를 돌림

뷰 클래스 (1/4)

```
// ChildView.h
class CChildView : public CWnd
{
    public:
        CChildView();

    protected:
        virtual BOOL PreCreateWindow(CREATESTRUCT& cs);

    public:
        virtual ~CChildView();

    protected:
        afx_msg void OnPaint();
        DECLARE_MESSAGE_MAP()
};
```

뷰 클래스 (2/4)

```
// ChildView.cpp
CChildView::CChildView()
{
}

CChildView::~CChildView()
{
}

BEGIN_MESSAGE_MAP(CChildView,CWnd )
    ON_WM_PAINT()
END_MESSAGE_MAP()
```

뷰 클래스 (3/4)

```
BOOL CChildView::PreCreateWindow(CREATESTRUCT& cs)
{
    if (!CWnd::PreCreateWindow(cs))
        return FALSE;

    cs.dwExStyle |= WS_EX_CLIENTEDGE;
    cs.style &= ~WS_BORDER;
    cs.lpszClass = AfxRegisterWndClass (
        CS_HREDRAW|CS_VREDRAW|CS_DBLCLKS,
        ::LoadCursor(NULL, IDC_ARROW),
        HBRUSH(COLOR_WINDOW+1), NULL);

    return TRUE;
}
```

Style: http://msdn2.microsoft.com/en-us/library/ms632600(VS.85).aspx

ExStyle: http://msdn2.microsoft.com/en-us/library/ms632680(VS.85).aspx

System Color: http://msdn2.microsoft.com/en-us/library/ms724371.aspx

뷰 클래스 (4/4)

```
void CChildView::OnPaint()
{
    CPaintDC dc(this);
    dc.TextOut(100, 100, _T("안녕하세요.“));
}
```

요약

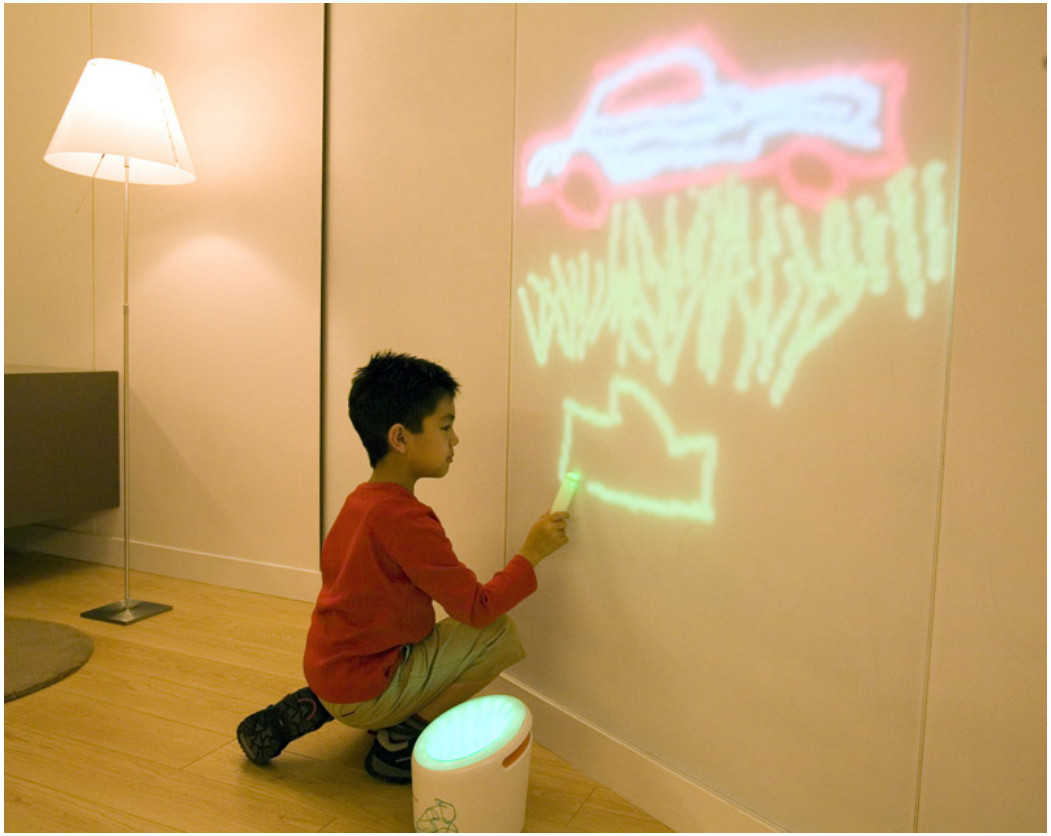
클래스 종류	베이스 클래스 이름	핵심 함수 - 주 역할
응용 프로그램 클래스	CWinApp	InitInstance() - 프레임 윈도우를 생성한다. Run() - 메시지 루프를 제공한다.
프레임 윈도우 클래스	CFrameWnd	OnCreate() - 뷰를 생성한다.
뷰 클래스	CWnd	OnPaint() - 화면에 출력한다.

API와 MFC 의 구조 대비:

- 책 page 54의 그림 참조

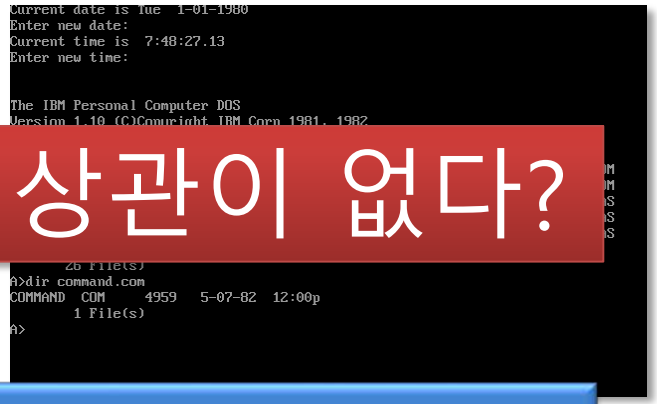
MFC 화면출력

How to draw with MFC



도스 시절 게임하기…

- In 1980s, before the birth of the “windows 95”
  - MS-DOS (Disk Operating System)



도스는 그래픽스 카드와 상관이 없다?

Device-Independent 개념의 시작

CGA

CGA용 게임

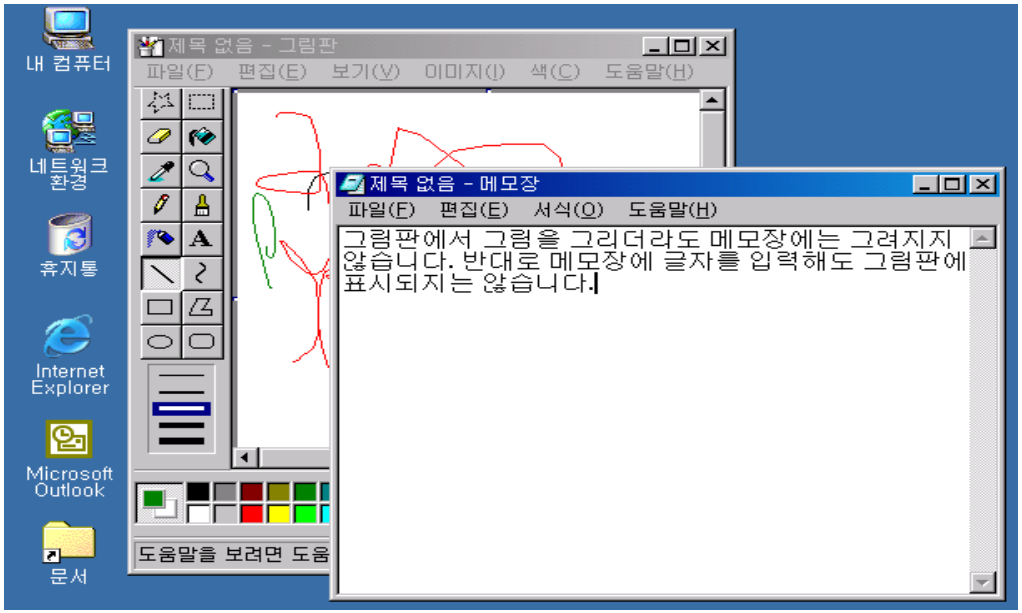
게임회사가 그래픽스 카드 드라이버를 각각 제공

GDI와 디바이스 컨텍스트 (1/5)

- 출력 시스템을 설계할 때 고려할 사항
  - Device-independent
    - 모니터, 비디오 카드, 프린터 등 출력에 사용되는 주변 장치가 변경되는 경우에도 프로그램을 수정할 필요가 없어야 한다.
  - Multi-tasking
    - 화면이나 기타 출력 장치를 직접 접근(Direct Access)하거나 독점해서 사용하는 것을 운영체제 차원에서 막아야 한다.

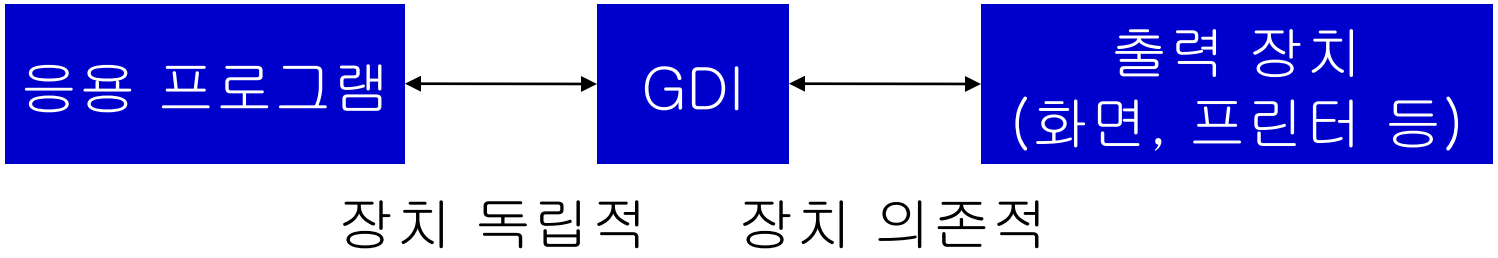
GDI와 디바이스 컨텍스트 (2/5)

- 화면 출력할 때 고려할 사항
  - 클라이언트 영역에 출력하려면 화면에 해당하는 윈도우 위치를 알아야 한다.
  - 화면에 여러 개의 윈도우가 있을 때 출력 결과가 다른 윈도우의 영역을 침범하지 않아야 한다.
  - 현재 출력할 화면이 다른 윈도우에 가려졌다면 출력을 할 수 없어야 한다.



GDI와 디바이스 컨텍스트 (3/5)

- GDI(Graphics Device Interface)
  - 운영체제의 하위 시스템 중 하나
  - 응용 프로그램의 요청을 받아서 실제 출력 장치에 대한 출력을 담당



GDI와 디바이스 컨텍스트 (4/5)

- 디바이스 컨텍스트(DC, Device Context)
  - GDI가 생성하고 관리하는 데이터 구조체
  - 멀티태스킹(멀티스레딩) GUI 환경에서 발생할 수 있는 여러 상황을 고려한 출력 가능

CDC: 디바이스 컨텍스트 클래스

화면에 관한 종합패키지:
(화면의 상태, 그림에 대한 정보
및 그림을 그리는 함수를 제공)

GDI와 디바이스 컨텍스트 (5/5)

- 윈도우 응용 프로그램의 출력 과정
  1. 응용 프로그램: 디바이스 컨텍스트 요청
  2. 운영체제(GDI): 디바이스 컨텍스트 생성 및 핸들(인덱스) 제공
  3. 응용프로그램: 디바이스 컨텍스트에 그림 그리기
  4. 운영체제(GDI): 그림 그리는 상황 검토 후 출력

GDI와 디바이스 컨텍스트 (5/5)

- 윈도우 응용 프로그램의 출력 과정

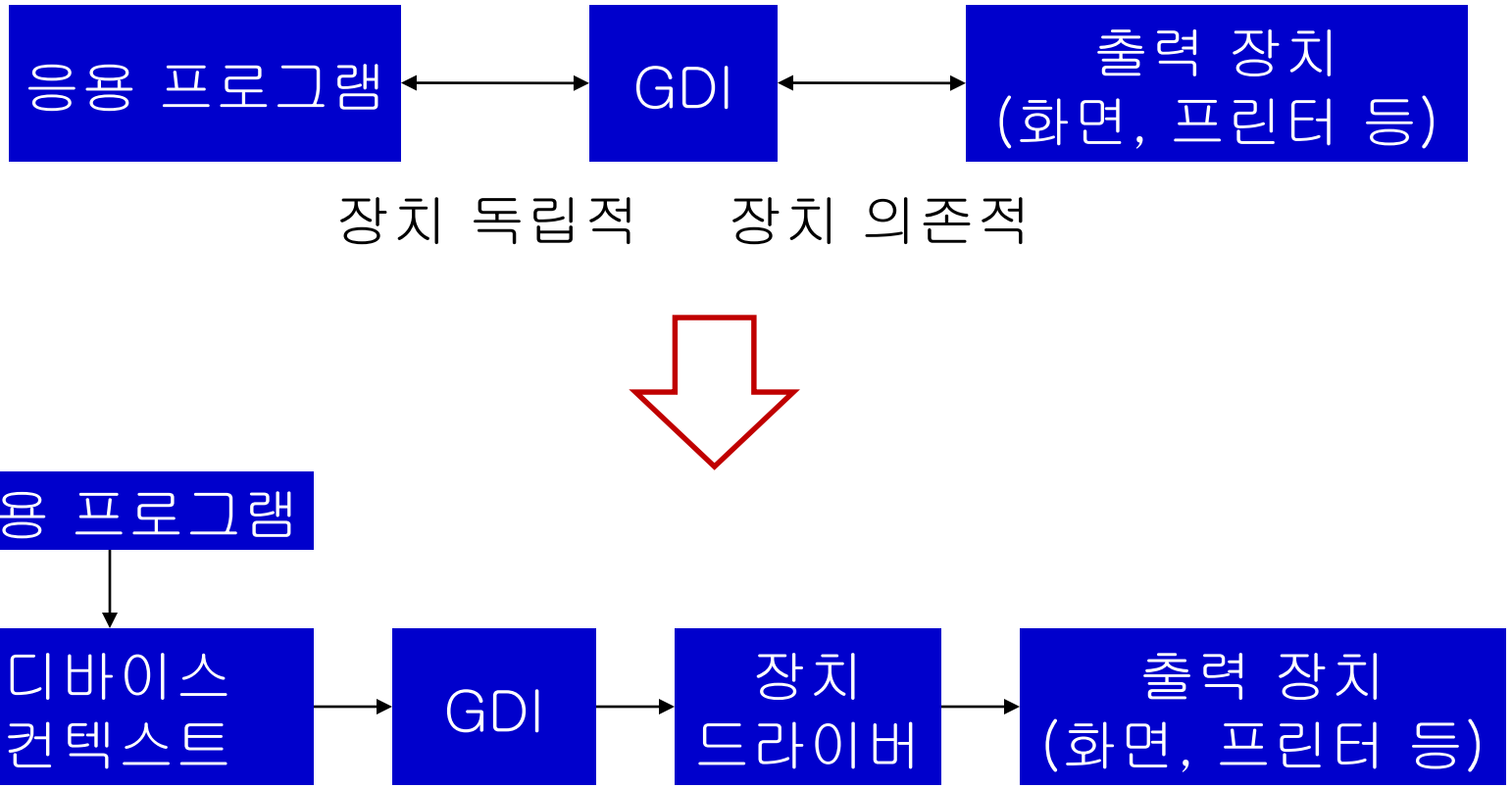


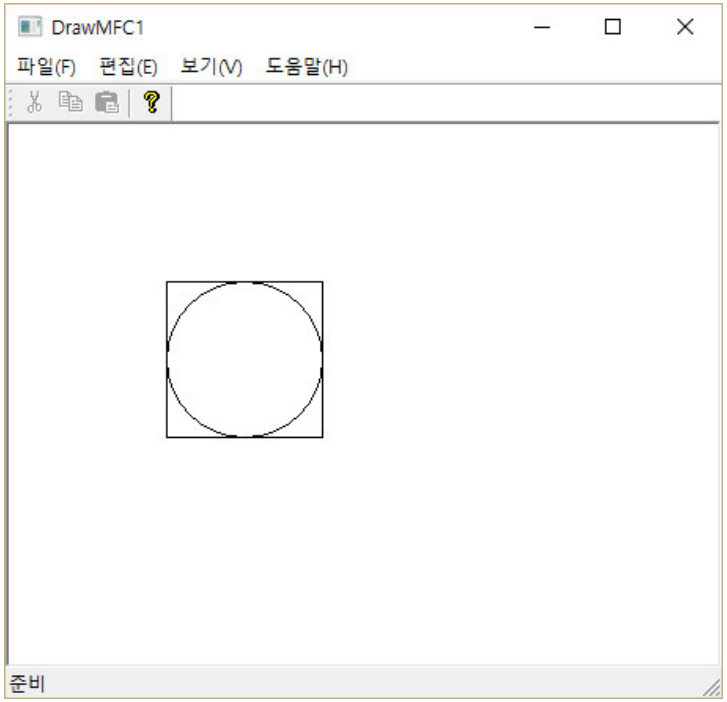
그림 그리는 부분은 어디에?

1. WM_PAINT 메시지 핸들러 (OnPaint) 함수 속

2. 아무데나...

연습: 도형 그리기

- 다음과 같이 사각형과 원을 추가
  - AppWizard로 New project 생성
  - CChildView의 WM_PAINT 메시지 핸들러 수정




```
void CChildView::OnPaint()
{
        CPaintDC dc(this);
        dc.Rectangle(100, 100, 200, 200);
        dc.Ellipse(100, 100, 200, 200);
}
```

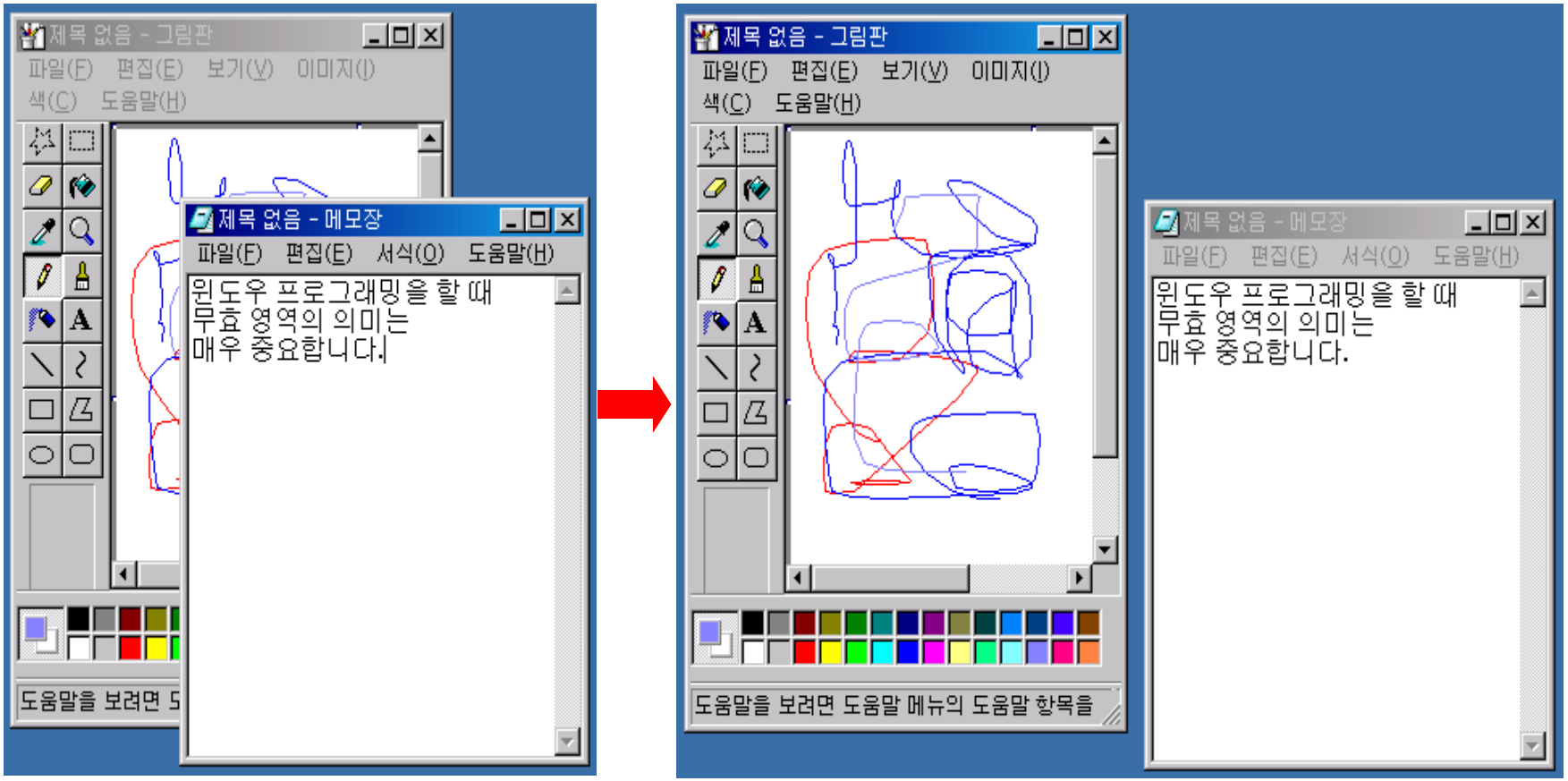
WM_PAINT??

- WM: Windows Message
  - MFC에서 제공하는 기본 메시지
  - 화면 내 창이나 마우스, 키보드과 관련된 메시지

- WM_PAINT
  - 화면을 다시 그릴 필요가 있다고 판단될 경우 발생되는 메시지
  - "무효영역(invalid region)이 생겼다!"

무효 영역 (1/3)

- 화면을 다시 그려야 하는 상황



무효 영역 (2/3)

- WM_PAINT 메시지 발생 상황
  - 윈도우가 생성될 때
  - 윈도우의 크기가 변경될 때
  - 최소화 또는 최대화 되었을 때
  - 다른 윈도우가 가렸다가 드러날 때
- 무효 영역 생성을 강제로 발생

```
void CWnd::Invalidate (BOOL bErase = TRUE);
void CWnd::InvalidateRect (LPCRECT lpRect, BOOL bErase = TRUE);
```

무효 영역 (3/3)

- WM_PAINT 메시지 처리
  - MFC를 사용할 때

```
void CMainFrame::OnPaint()
{
    CPaintDC dc(this);
    dc.TextOut(100, 100, L“Hello, MFC”);
}
```

  - 디바이스 컨텍스트 객체 생성 (this pointer 이용)
    - CPaintDC dc(this)
  - 객체의 멤버 함수를 호출하여 출력
    - dc.TextOut(…)

출력 방법 비교

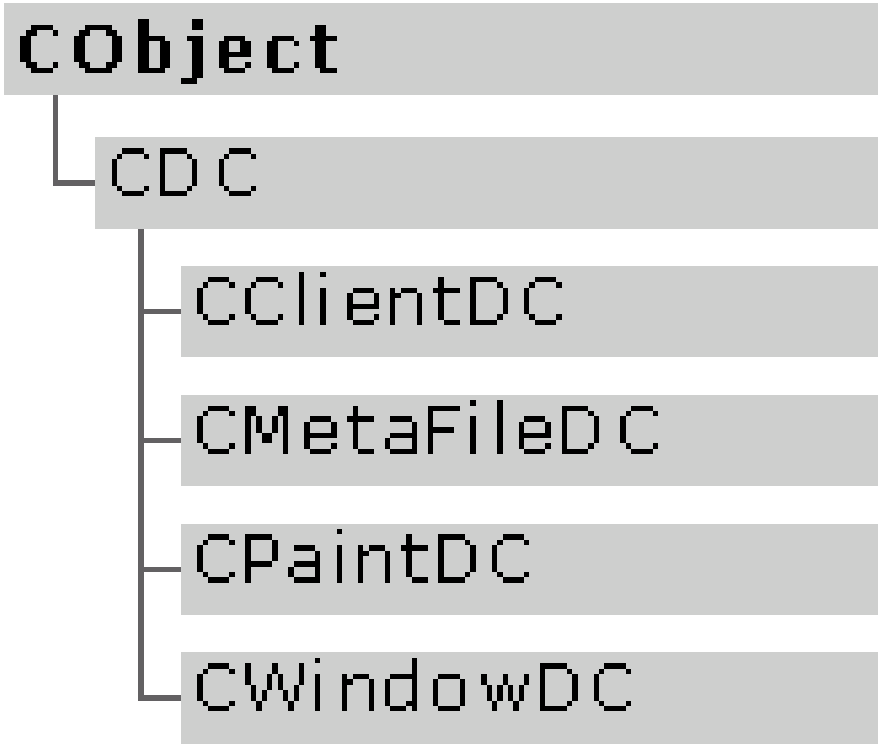
- MFC → 클래스의 사용으로 간단하다.
  - 디바이스 컨텍스트 객체 생성 (this pointer 이용)
    - CPaintDC dc(this)
  - 객체의 멤버 함수를 호출하여 출력
    - dc.TextOut(…)

- Win32 API → 훨씬 복잡하다.
  - 윈도우 운영체제에게 디바이스 컨텍스트를 요청
    - BeginPaint(hwnd, &ps);
  - 얻어낸 디바이스 컨텍스트 핸들을 사용하여 출력
    - TextOut(hdc, 100, 100, “Hello, MFC”);
  - 운영체제에게 디바이스 컨텍스트 사용이 끝났음을 알림
    - EndPaint(hwnd, &ps);

다양한 디바이스 컨텍스트 클래스

클래스 이름	용도
CPaintDC	클라이언트 영역에 출력할 때 (WM_PAINT 메시지 핸들러에서만 사용)
CClientDC	클라이언트 영역에 출력할 때 (WM_PAINT 메시지 핸들러를 제외한 다른 모든 곳에서 사용)
CWindowDC	윈도우의 전체 영역(클라이언트 영역 + 비 클라이언트 영역)에 출력할 때
CMetaFileDC	메타 파일(Metafile)에 출력할 때

클래스 계층도



CPaintDC 클래스

- 오로지 WM_PAINT 메시지 핸들러에서만 사용
  - OnPaint()

- CPaintDC 사용 예

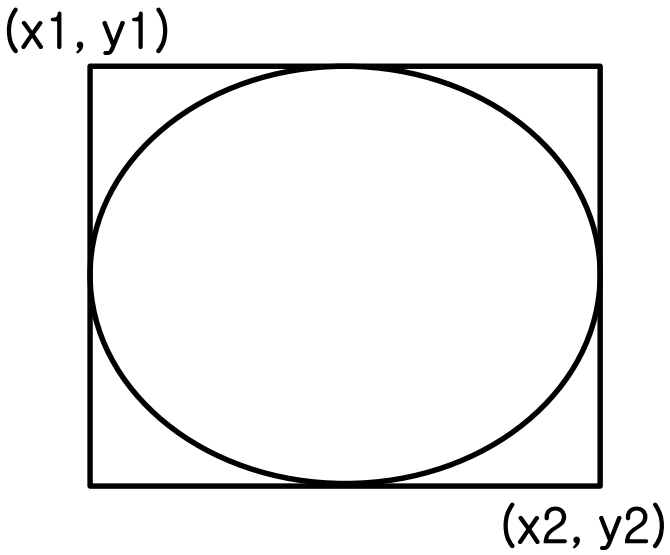
```
void CChildView::OnPaint()
{
    CPaintDC dc(this);
    // ...
}
```

DC의 그리기 관련 맴버 함수

- 도형 그리기

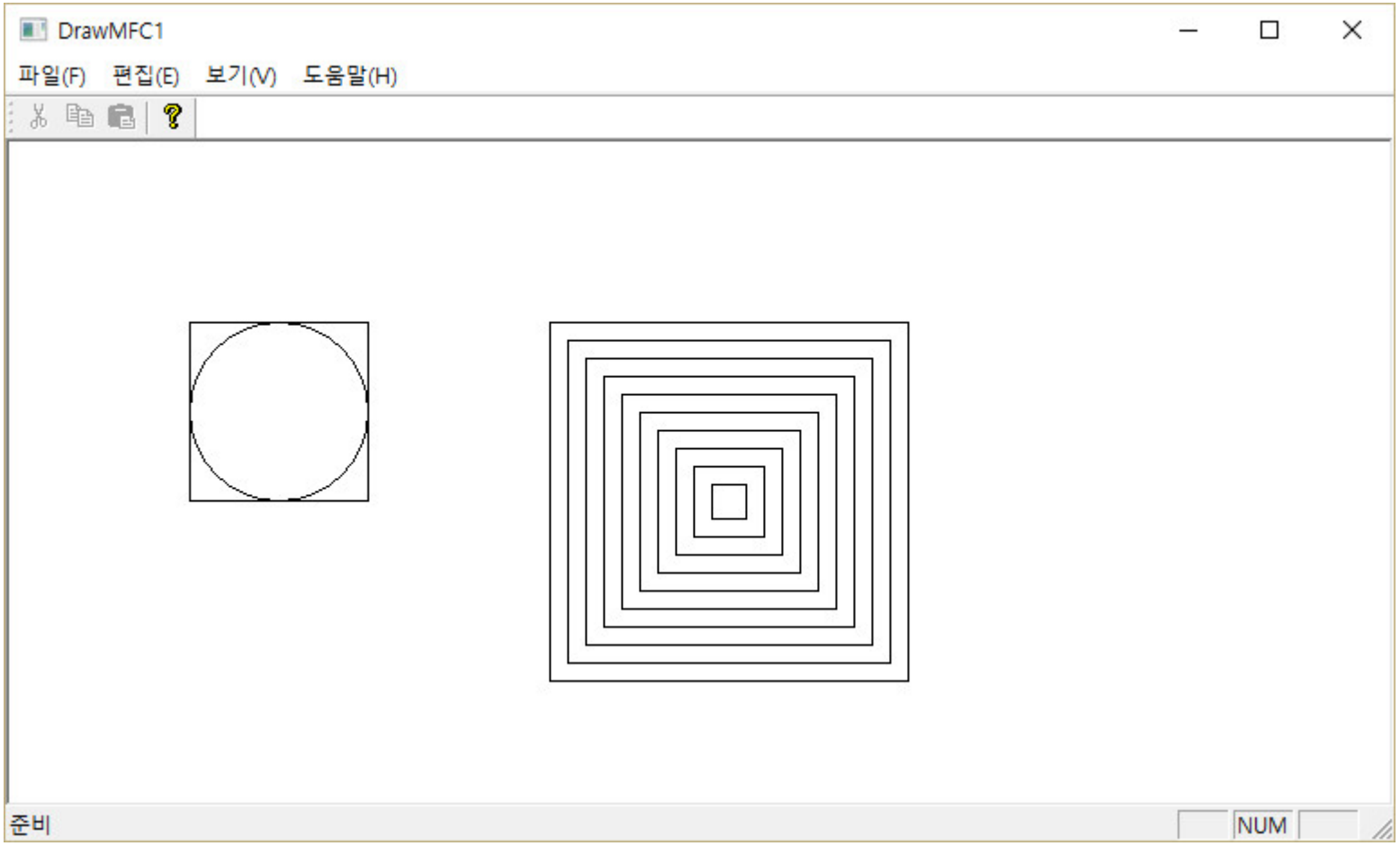
이름	기능
Rectangle()	사각형을 그린다.
Ellipse()	사각형에 내접하는 타원을 그린다.

```
dc.Rectangle (x1, y1, x2, y2);
dc.Ellipse (x1, y1, x2, y2);
```



연습: CPaintDC 사용

- 다음과 같이 점점 작아지는 사각형을 추가



연습: CPaintDC 사용

```
void CMainWindow::OnPaint ()
{
    CPaintDC dc(this);

    … // 기존 사각형과 원을 그린 부분

    // 점점 작아지는 사각형 그리기
    for(int i = 0; i < 10; i++)
    {
        int dd = i*10;
        dc.Rectangle(100 + dd, 100 + dd, 300 - dd, 300 - dd);
    }
}
```

CClientDC 클래스

- WM_PAINT 메시지 핸들러 이외의 부분에서 사용
- CClientDC 사용 예

```
void CChildView::OnLButtonDown(UINT nFlags, CPoint point)
{
   CClientDC dc(this);
   // ...
}
```

CClientDC 사용

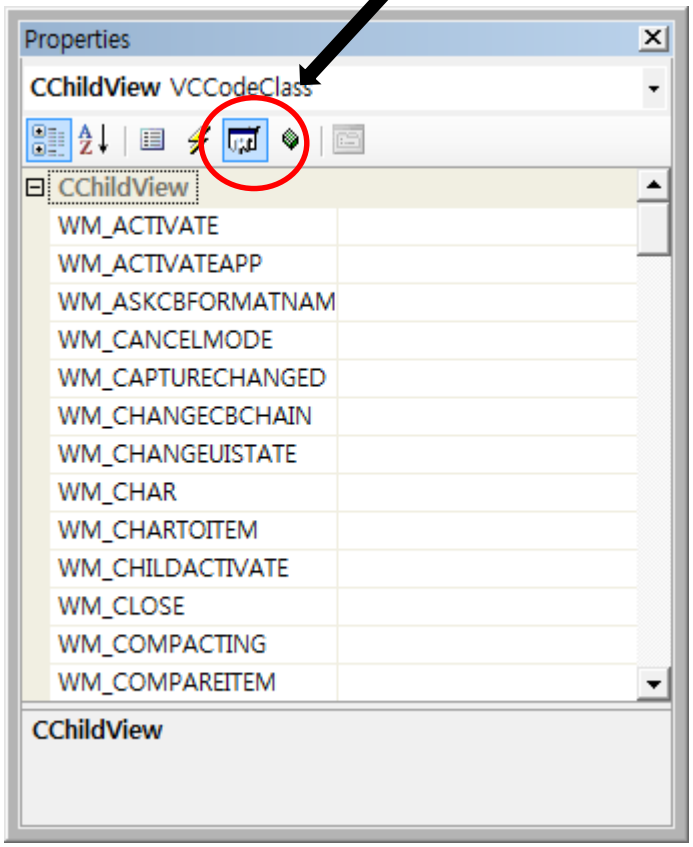
- 마우스 버튼이 눌렸을 때 그림을 그려보자!

1. 마우스 버튼이 눌리는 메시지 핸들러 추가
2. CClientDC 객체 생성
3. 생성된 DC객체를 이용하여 그리기

마우스 버튼 핸들러 추가 1

- CChildView의 속성창을 이용하여 추가

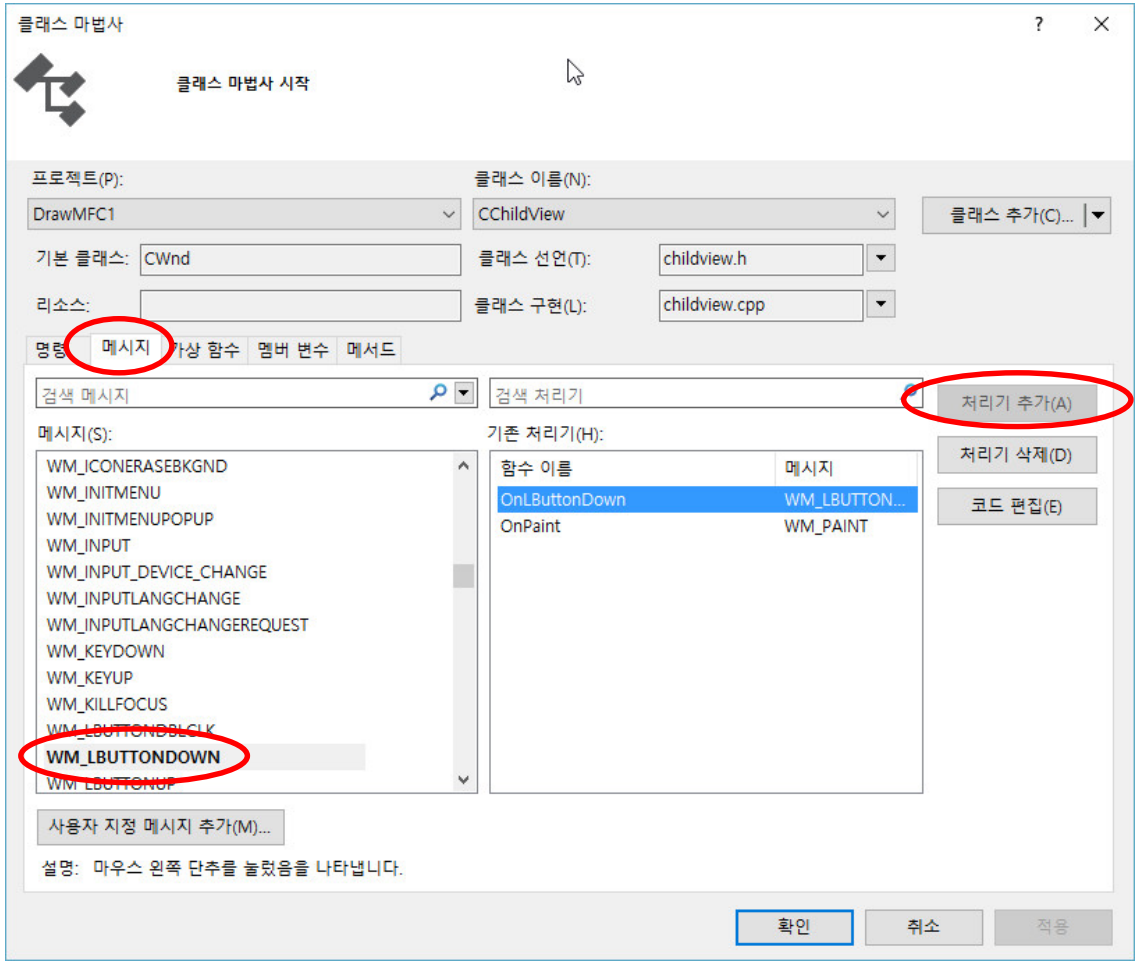
윈도우 메시지 핸들러 추가 버튼



마우스 버튼 메시지:
WM_LBUTTONDOWN
WM_RBUTTONDOWN
WM_MBUTTONDOWN
…

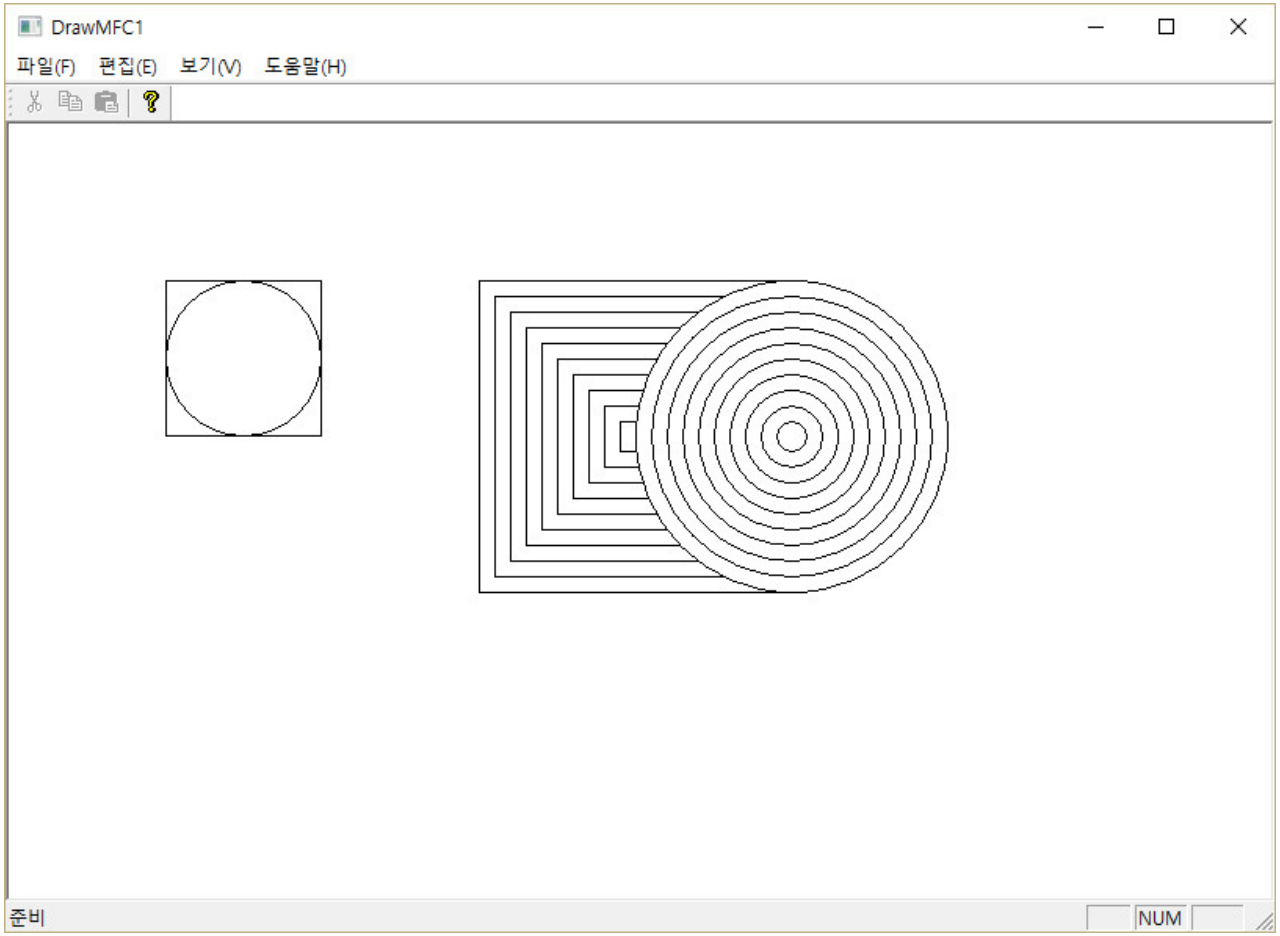
마우스 버튼 핸들러 추가 2

- ClassWizard를 이용하여 추가
  - 클래스 뷰(Class View) → CChildView → 클래스 마법사 (*RMB)



연습: CClientDC 사용 1

- 다음과 같이 마우스 버튼(왼쪽)을 누르면 점점 작아지는 원을 그려보자

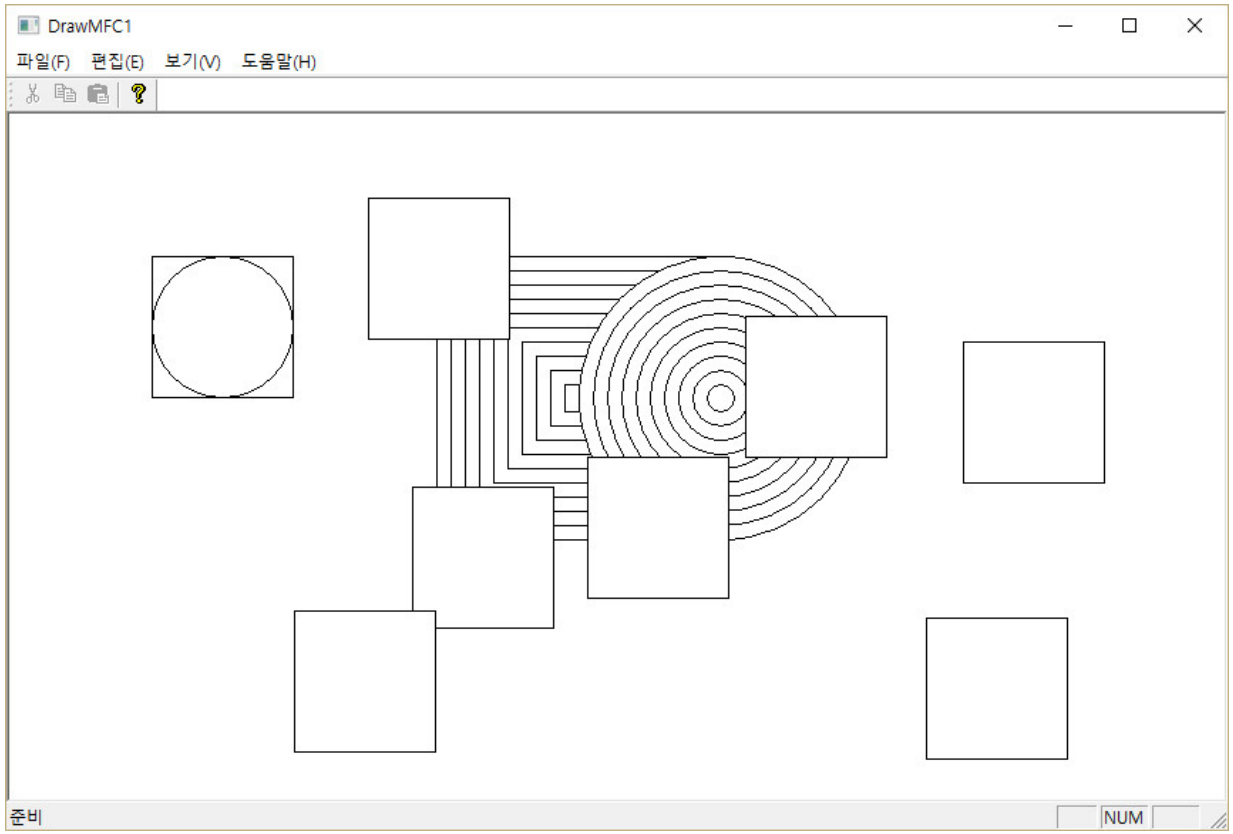


연습: CClientDC 사용 1

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

    // 점점 작아지는 원 그리기
    For (int i = 0; i < 10; i++)
    {
        int dd = i*10;
        dc.Ellipse(400 + dd, 100 + dd, 600 - dd, 300 - dd);
    }

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

연습: CClientDC 사용 2

- 다음과 같이 마우스 버튼(오른쪽)을 누른 위치에 사각형을 그려보자

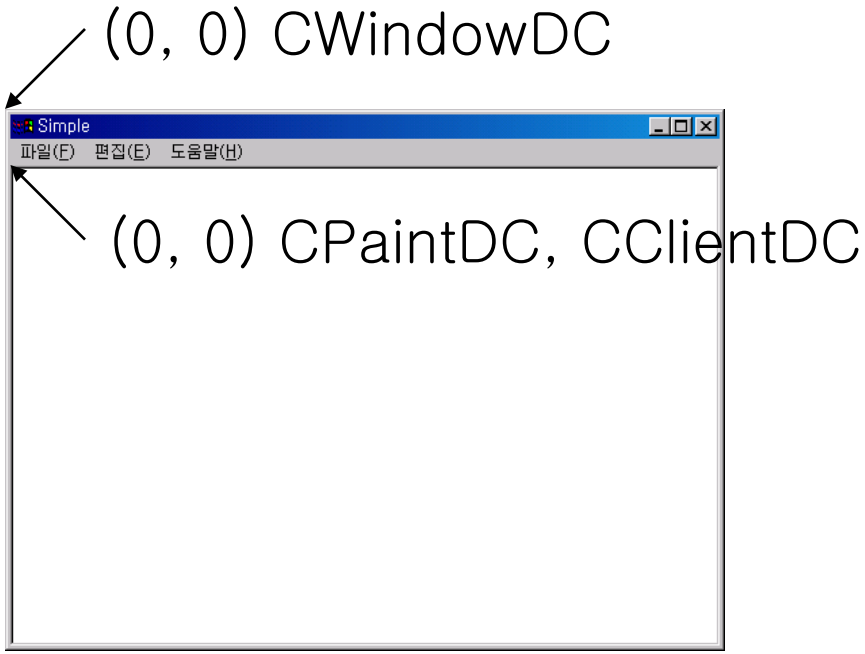


연습: CClientDC 사용 2

```
void CChildView::OnRButtonDown(UINT nFlags, Cpoint point)
{
    // TODO: 여기에 메시지 처리기 코드를 추가 및/또는 기본값을 호출합니다.
    CClientDC dc(this);

    // 점점 작아지는 원 그리기
    dc.Rectangle(point.x, point.y, point.x + 100, point.y + 100);

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

CWindowDC 클래스

- CWindowDC 사용 방법
  - CPaintDC, CClientDC 클래스와 동일
- 원점 위치



Q & A