윈도우 프로그래밍
Visual C++ MFC Programming

화면 출력

GDI와 디바이스 컨텍스트

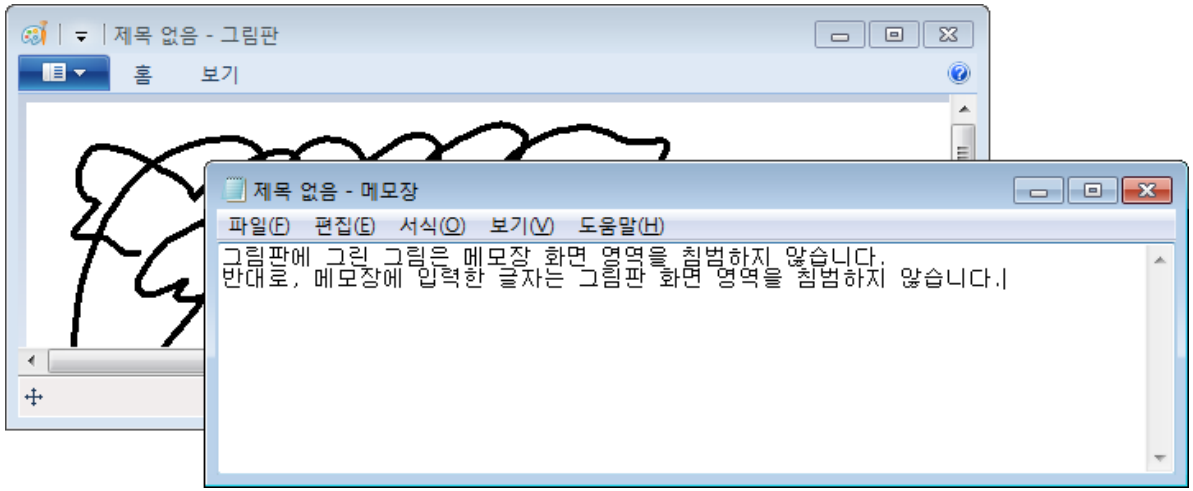
- 윈도우 운영체제에서 출력 시스템을 설계 시 고려 사항
  - 장치 변경에 따른 프로그램 수정 없음
    - 모니터, 비디오 카드, 프린터 등 출력에 사용되는 주변 장치가 변경되더라도 프로그램을 수정할 필요가 없어야 한다.
  - 프로그램 출력 영역 제약
    - 여러 프로그램이 화면을 분할해서 사용하므로 각각의 프로그램이 출력하는 영역에 제약을 가해야 한다.
    - 화면이나 기타 출력 장치에 직접 접근(Direct Access)하거나 독점해서 사용(Exclusive Use)하는 것을 운영체제 수준에서 막아야 한다.

GDI와 디바이스 컨텍스트

- GDI(Graphics Device Interface)
  - 윈도우 운영체제의 하위 시스템 중 하나로 DLL로 존재
  - 응용 프로그램의 요청을 받아 실제 출력 장치인 모니터나 프린터에 출력하는 역할을 담당

GDI와 디바이스 컨텍스트

- 윈도우 응용 프로그램이 화면에 출력 시 고려 사항
  - 클라이언트 영역에 출력하려면 출력 대상 윈도우의 위치를 알아야 한다.
  - 화면에 윈도우가 여러 개 있을 때, 출력 결과가 다른 윈도우 영역을 침범하지 않아야 한다.
  - 현재 출력할 화면이 다른 윈도우에 가려졌다면 출력할 수 없어야 한다.

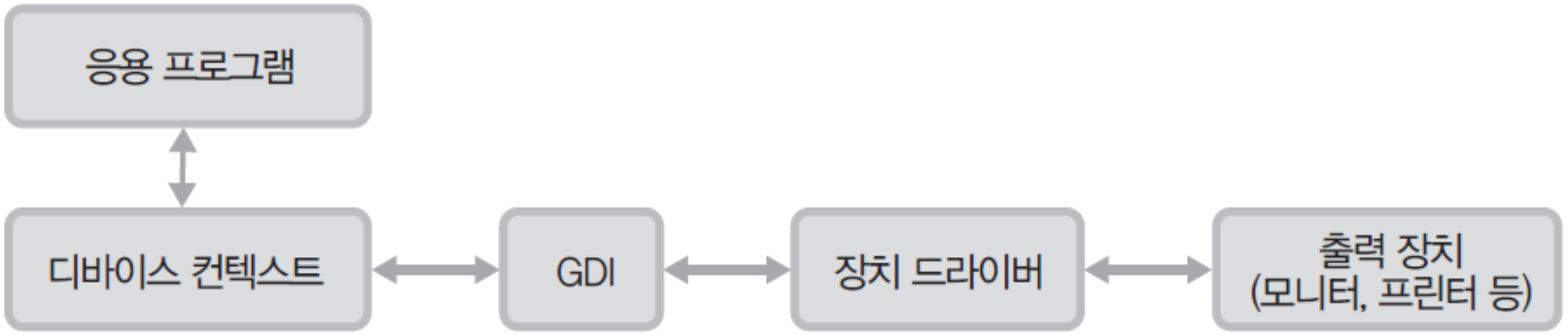


GDI와 디바이스 컨텍스트

- 디바이스 컨텍스트(DC, Device Context)
  - GDI가 생성하고 관리하는 데이터 구조체
  - 멀티태스킹 GUI 환경에서 발생할 수 있는 복잡한 상황들을 신경쓰지 않고 장치에 자유롭게 출력 가능

GDI와 디바이스 컨텍스트

- 윈도우 응용 프로그램의 출력 과정(1/2)



① 운영체제에 디바이스 컨텍스트를 요청한다.

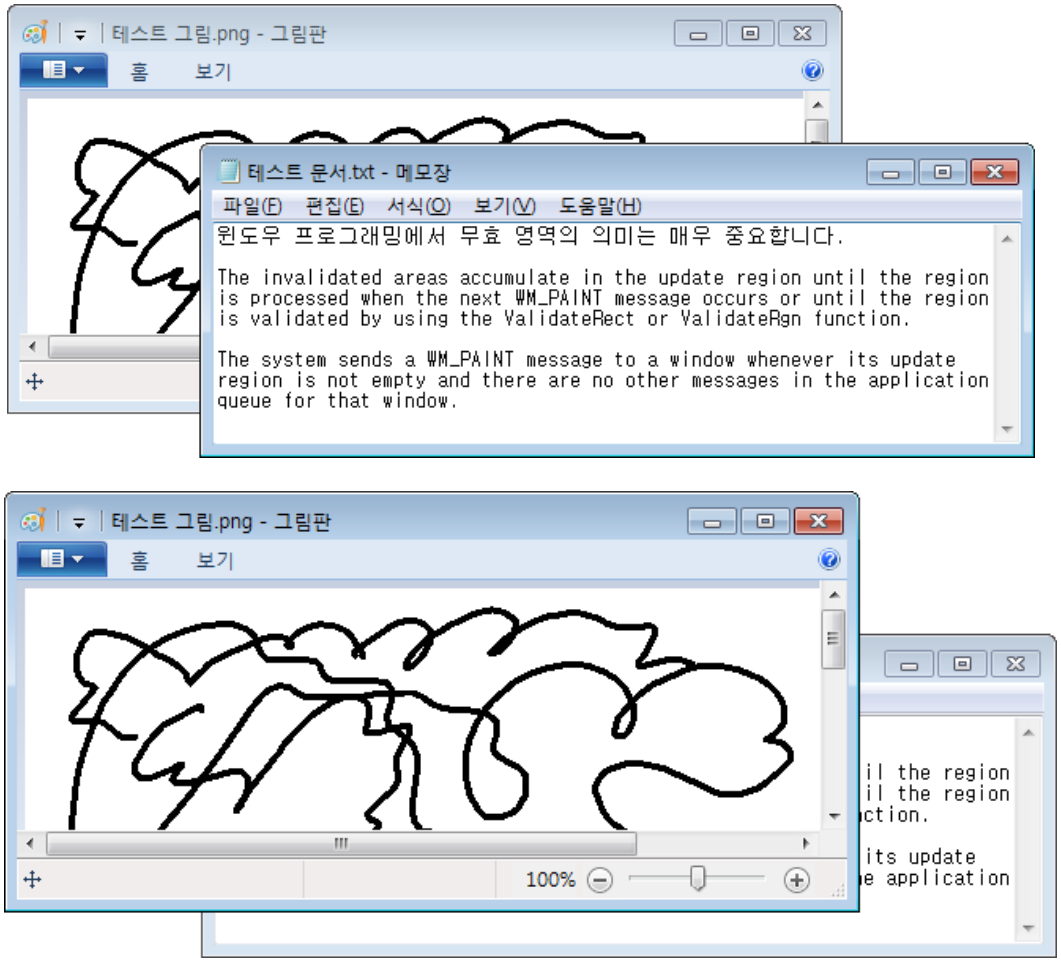
② 요청을 받은 운영체제의 GDI가 내부적으로 디바이스 컨텍스트를 만든 후, 디바이스 컨텍스트를 가리키는 핸들(HDC 타입)을 응용 프로그램에 돌려준다.

GDI와 디바이스 컨텍스트

- 윈도우 응용 프로그램의 출력 과정 (2/2)
  - ③ 응용 프로그램은 (필요하다면) 받은 디바이스 컨텍스트의 속성을 일부 변경한다. 그런 다음, 디바이스 컨텍스트 핸들을 윈도우 API 함수에 전달하여 출력(→장치 독립적)을 요청한다. 이 요청은 다시 운영체제의 GDI에 전달된다.
  - ④ GDI가 디바이스 컨텍스트에 포함된 정보를 토대로 다양한 상황을 고려하여 출력(→장치 의존적)한다. 이때 장치별 출력을 위해 장치 드라이버를 사용한다.

무효 영역의 개념

- 화면을 다시 그려야 하는 상황



무효 영역의 개념

- WM_PAINT 메시지 처리 방식
  - HelloSDK 프로그램

```
case WM_PAINT:
    hdc = BeginPaint(hwnd, &ps);
    TextOut(hdc, 100, 100, str, lstrlen(str));
    EndPaint(hwnd, &ps);
    return 0;
```

  - HelloMFC 프로그램

```
void CMainFrame::OnPaint()
{
    /* 생성자에서 ::BeginPaint() 호출! */
    CPaintDC dc(this);
    TCHAR *msg = _T("Hello, MFC");
    dc.TextOut(100, 100, msg, lstrlen(msg));
} /* 소멸자에서 ::EndPaint() 호출! */
```

무효 영역의 개념

- WM_PAINT 메시지 발생 상황
  - 윈도우가 생성될 때
  - 윈도우의 크기가 변경될 때
  - 윈도우가 최소화 또는 최대화되었을 때
  - 다른 윈도우가 가렸다가 드러날 때
- 무효 영역 생성 함수

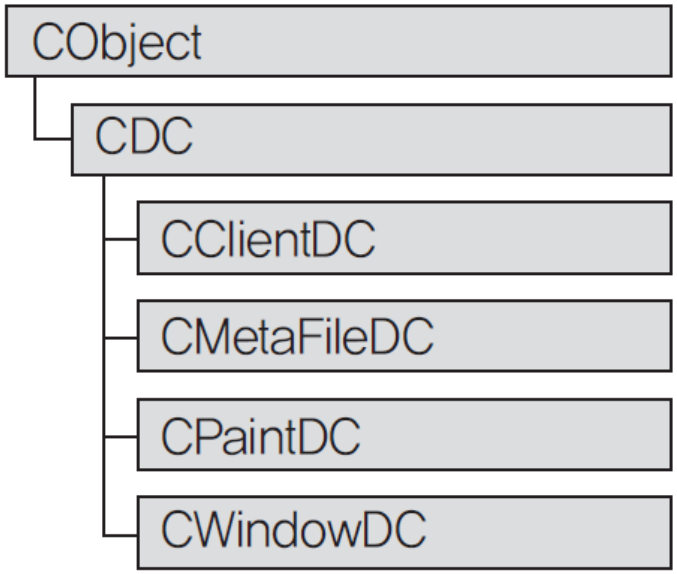
```
void CWnd::Invalidate(BOOL bErase = TRUE);
void CWnd::InvalidateRect(LPCRECT lpRect, BOOL bErase = TRUE);
void CWnd::InvalidateRgn(CRgn* pRgn, BOOL bErase = TRUE);
```

다양한 디바이스 컨텍스트 클래스

- SDK 프로그램 출력 과정
  - 운영체제에 디바이스 컨텍스트를 요청한다.
  - 디바이스 컨텍스트를 전달 인자로 사용해 API 함수를 호출하여 출력한다.
  - 디바이스 컨텍스트 사용이 끝났음을 운영체제에 알린다.
- MFC 프로그램 출력 과정
  - 디바이스 컨텍스트 객체를 생성한다.
  - 객체의 멤버 함수를 호출하여 출력한다.

다양한 디바이스 컨텍스트 클래스

- MFC 클래스 계층도



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	윈도우 전체 영역(클라이언트 영역 + 비클라이언트 영역)에 출력할 때
CMetaFileDC	메타파일(Metafile)에 출력할 때

CPaintDC 클래스

- CPaintDC
  - WM_PAINT 메시지 핸들러에서만 사용 가능
  - 클라이언트 영역에만 출력 가능

- 사용 예

```
void CChildView::OnPaint()
{
    CPaintDC dc(this);
    dc.Rectangle(0, 0, 200, 100);
    dc.Ellipse(200, 100, 500, 200);
}
```

** this 포인터: 뷰 객체의 주소값을 가지고 있다.

CClientDC 클래스

- CClientDC
  - WM_PAINT 메지시 핸들러를 제외한 곳에서 사용 가능
  - 클라이언트 영역에만 출력 가능

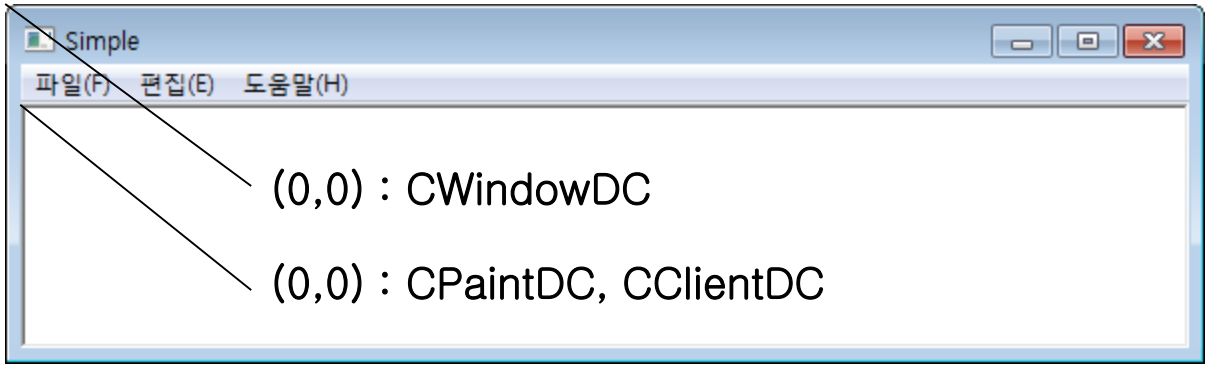
- 사용 예

```
void CChildView::OnLButtonDown(UINT nFlags, CPoint point)
{
    CClientDC dc(this);
    dc.Rectangle(point.x-20, point.y-20, point.x+20, point.y+20);
}
```

CWindowDC 클래스

- CWindowDC
  - 윈도우 전체(클라이언트+비클라이언트)영역에 출력
  - CPaintDC/CClientDC 클래스와 사용법 동일
  - 단, 좌표의 원점 위치가 다름

- 디바이스 컨텍스트별 원점 위치



그리기 함수

- 점 찍기

이름	기능
GetPixel()	화면 (x, y) 지점의 색을 얻는다.
SetPixel()	화면 (x, y) 지점에 특정 색상의 점을 찍고, 원래 점의 색을 리턴한다.
SetPixelV()	SetPixel( ) 함수와 출력은 같지만, 원래 점의 색을 리턴하지 않으므로 속도가 좀더 빠르다.

```
CClientDC dc(this);
dc.SetPixel(point.x, point.y, RGB(255, 0, 0));
```

그리기 함수

- 선 그리기
  - 현재 위치를 (x1,y1)으로 옮긴다.
    - 현재 위치는 디바이스 컨텍스트의 속성 중 하나
  - 현재 위치에서 (x2, y2)까지 선을 그린다.
    - 완료되면 현재 위치는 (x2,y2)로 자동 변경됨

이름	기능
MoveTo()	현재 위치를 (x, y) 위치로 옮긴다.
LineTo()	현재 위치에서 (x, y) 위치까지 선을 그리고, 현재 위치를 (x, y)로 변경한다.
Polyline()	POINT 구조체 배열로 전달된 점들을 차례로 이어서 선을 그린다.

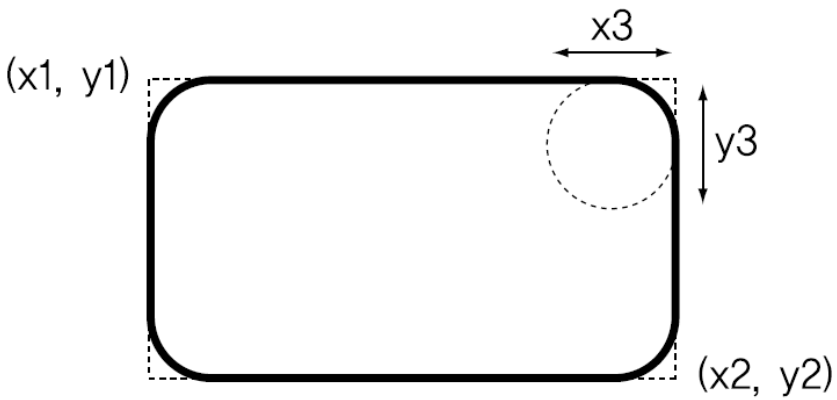
```
CClientDC dc(this);
dc.MoveTo(point.x, point.y);
dc.LineTo(point.x+100, point.y+200);
```

그리기 함수

- 도형 그리기

이름	기능
Rectangle()	직사각형을 그린다.
Ellipse()	직사각형에 내접하는 타원을 그린다.
RoundRect()	테두리가 둥근 직사각형을 그린다.
Polygon()	POINT 구조체 배열로 전달된 점들을 차례로 이어서 다각형을 그린다.

RoundRect(int x1, int y1, int x2, int y2, int x3, int y3);



텍스트 함수

- 텍스트 함수

이름	기능
TextOut()	(x, y) 위치에 문자열을 출력한다.
DrawText()	직사각형 영역 내부에 문자열을 출력한다.
SetTextColor()	글자의 색을 바꾼다.
SetBkColor()	글자의 배경색을 바꾼다.
SetTextAlign()	기준 위치에 대한 문자열 정렬 방식을 설정한다.

```
dc.SetTextColor(RGB(255, 0, 0));
dc.SetBkColor(RGB(0, 0, 255));
dc.TextOutW(point.x, point.y, _T("Hello PC"));
```

속성 함수

속성	초깃값	속성을 얻는 함수	속성을 변경하는 함수
텍스트 색상	검은색	GetTextColor()	SetTextColor()
배경 색상	흰색	GetBkColor()	SetBkColor()
배경 모드	OPAQUE	GetBkMode()	SetBkMode()
매핑 모드	MM_TEXT	GetMapMode()	SetMapMode()
그리기 모드	R2_COPYPEN	GetROP2()	SetROP2()
현재 위치	(0, 0)	GetCurrentPosition()	MoveTo()
펜	BLACK_PEN	SelectObject()	SelectObject()
브러시	WHITE_BRUSH	SelectObject()	SelectObject()
폰트	SYSTEM_FONT	SelectObject()	SelectObject()
비트맵	없음	SelectObject()	SelectObject()
팔레트	없음	SelectPalette()	SelectPalette()
리전	없음	SelectObject()	SelectObject()

그리기 모드

그리기 모드	연산	그리기 모드	연산
R2_NOP	D = D	R2_MERGENOTPEN	D = ~S | D
R2_NOT	D = ~D	R2_MASKNOTPEN	D = ~S & D
R2_BLACK	D = BLACK	R2_MERGEPEN	D = D | S
R2_WHITE	D = WHITE	R2_NOTMERGEPEN	D = ~(D | S)
R2_COPYPEN	D = S	R2_MASKPEN	D = D & S
R2_NOTCOPYPEN	D = ~S	R2_NOTMASKPEN	D = ~(D & S)
R2_MERGEPENNOT	D = ~D | S	R2_XORPEN	D = S ^ D
R2_MASKPENNOT	D = ~D & S	R2_NOTXORPEN	D = ~(S ^ D)

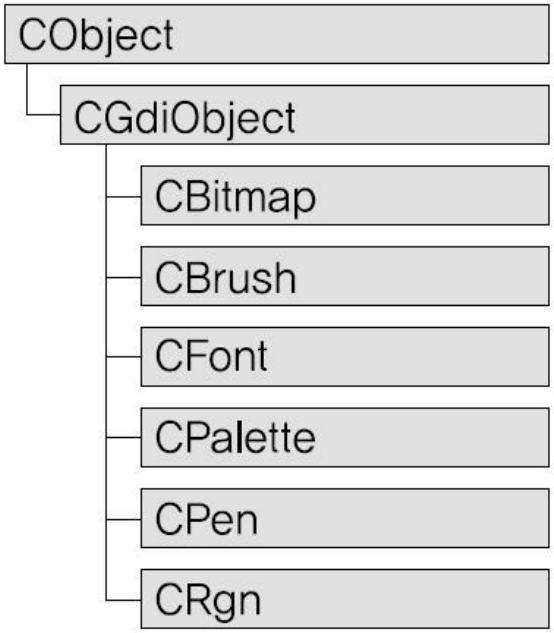
GDI 객체

- GDI 객체
  - GDI에서 출력할 때 사용하는 도구
- 종류

GDI 객체	용도	클래스 이름
펜	선을 그릴 때	CPen
브러시	면의 내부를 채울 때	CBrush
폰트	글자를 출력할 때	CFont
비트맵	그림을 출력할 때	CBitmap
팔레트	출력될 색의 집합을 다룰 때	CPalette
리전	다양한 형태의 면을 정의할 때	CRgn

GDI 객체

- MFC 클래스 계층도



GDI 객체

- GDI 객체 사용 방법

  ① GDI 객체를 스택에 생성한다.

  ② 생성된 GDI 객체를 CDC::SelectObject( ) 함수에 전달하여 디바이스 컨텍스트에 선택한다.

  ③ CDC 클래스 멤버 함수를 호출하여 출력한다.

  ④ 이전의 GDI 객체를 CDC::SelectObject( ) 함수에 전달하여 복원한다.

  ⑤ 함수가 끝나면 GDI 객체의 소멸자가 자동으로 호출되어 파괴된다.

펜

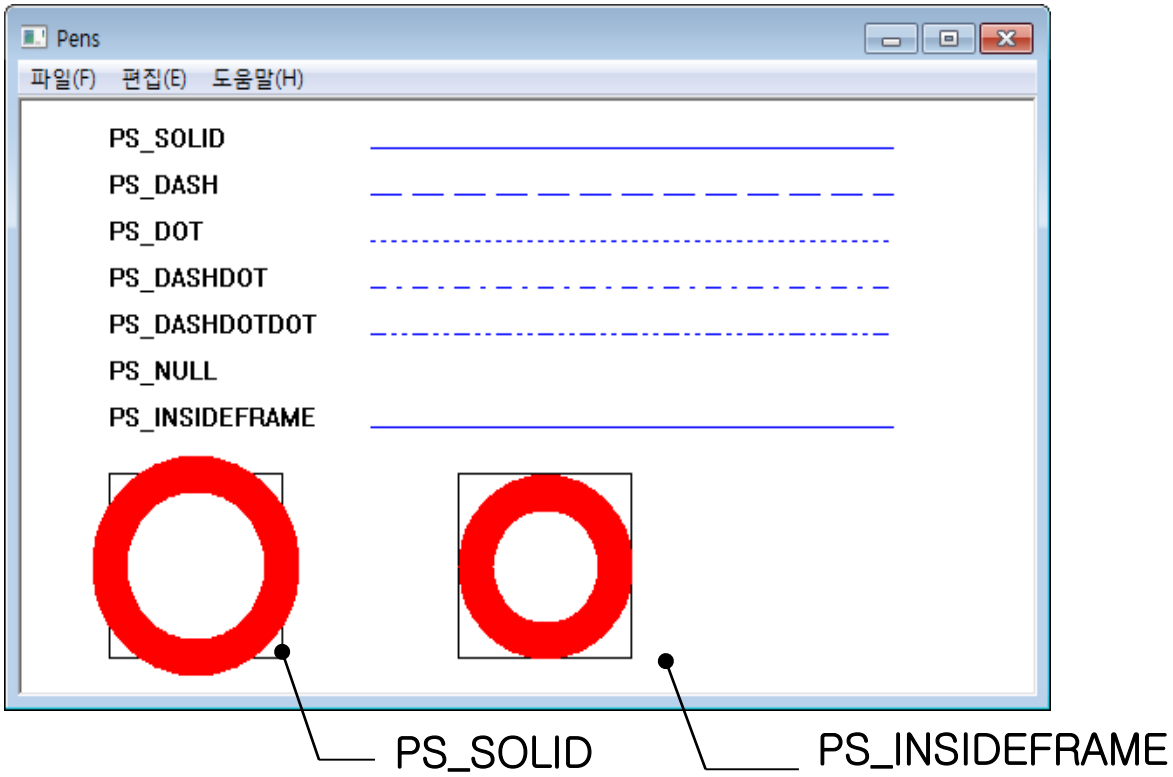
- 펜 생성 방법

```
// 방법 1
CPen pen(PS_SOLID, 2, RGB(255, 0, 0)); // 폭이 2인 빨간색 펜

// 방법 2
CPen pen;
pen.CreatePen(PS_SOLID, 2, RGB (255, 0, 0)); // 폭이 2인 빨간색 펜
```

펜

- 펜 스타일


Pens
파일(F) 편집(E) 도움말(H)
PS_SOLID
PS_DASH
PS_DOT
PS_DASHDOT
PS_DASHDOTDOT
PS_NULL
PS_INSIDEFRAME
PS_SOLID
PS_INSIDEFRAME

펜

- 펜 사용 예 1

```
CPaintDC dc(this); // 디바이스 컨텍스트 객체를 만든다.
CPen pen(PS_SOLID, 1, RGB(0, 0, 255)); // 펜 객체를 만든다.
CPen *pOldPen = dc.SelectObject(&pen); // 새로운 펜 선택, 이전 펜 저장
dc.Rectangle(100, 100, 200, 200); // 직사각형을 그린다.
dc.SelectObject(pOldPen); // 이전 펜 복원, 사용하던 펜 선택 해제
```

- 펜 사용 예 2

```
CPaintDC dc(this); // 디바이스 컨텍스트 객체를 만든다.
CPen pen(PS_SOLID, 1, RGB(0, 0, 255)); // 펜 객체를 만든다.
dc.SelectObject(&pen); // 새로운 펜을 선택한다.
dc.Rectangle(100, 100, 200, 200); // 직사각형을 그린다.
```

브러시

- 브러시 종류

종류	생성 예
솔리드(Solid, 속이 채워짐)	CBrush brush(RGB(255, 0, 0));
해치(Hatch, 교차된 평행선 무늬)	CBrush brush(HS_DIAGCROSS, RGB(255, 0, 0));

패턴(Pattern,
비트맵의 반복 무늬)
	CBitmap bitmap; //비트맵 객체 생성
	bitmap.LoadBitmap(IDB_BITMAP1); //비트맵 로드
	CBrush brush(&bitmap);

브러시

- 브러시 사용 예 1

```
CPaintDC dc(this); // 디바이스 컨텍스트 객체를 만든다.
CBrush brush(RGB(255, 0, 0)); // 브러시 객체를 만든다.
CBrush *pOldBrush = dc.SelectObject(&brush); // 새로운 브러시 선택, 이전 브러시 저장
dc.Ellipse(100, 100, 200, 200); // 원을 그린다.
dc.SelectObject(pOldBrush); // 이전 브러시 복원, 사용하던 브러시 선택 해제
```

- 브러시 사용 예 2

```
CPaintDC dc(this); // 디바이스 컨텍스트 객체를 만든다.
CBrush brush(RGB(255, 0, 0)); // 브러시 객체를 만든다.
dc.SelectObject(&brush); // 새로운 브러시를 선택한다.
dc.Ellipse(100, 100, 200, 200); // 원을 그린다.
```

폰트

- 폰트 생성 방법
  - 폰트 객체 생성
  - 폰트 객체에 대해 Create*() 함수 호출

```
font.CreateFont(...);
또는 font.CreateFontIndirect(...);
또는 font.CreatePointFont(...);
또는 font.CreatePointFontIndirect(...);
```

폰트

- 폰트 사용 예

```
CPaintDC dc(this); // 디바이스 컨텍스트 객체를 만든다.
CFont font; // 폰트 객체를 생성한다.
font.CreatePointFont(400, _T("Arial ” )); // 크기가 40(= 400/10) 포인트인 Arial 폰트
dc.SelectObject(&font); // 새로운 폰트를 선택한다.
dc.TextOut(10, 10, CString("Hello")); // 텍스트를 출력한다.
```