

Window Programming

Visual C++ MFC Programming

Lecture 06

김예진

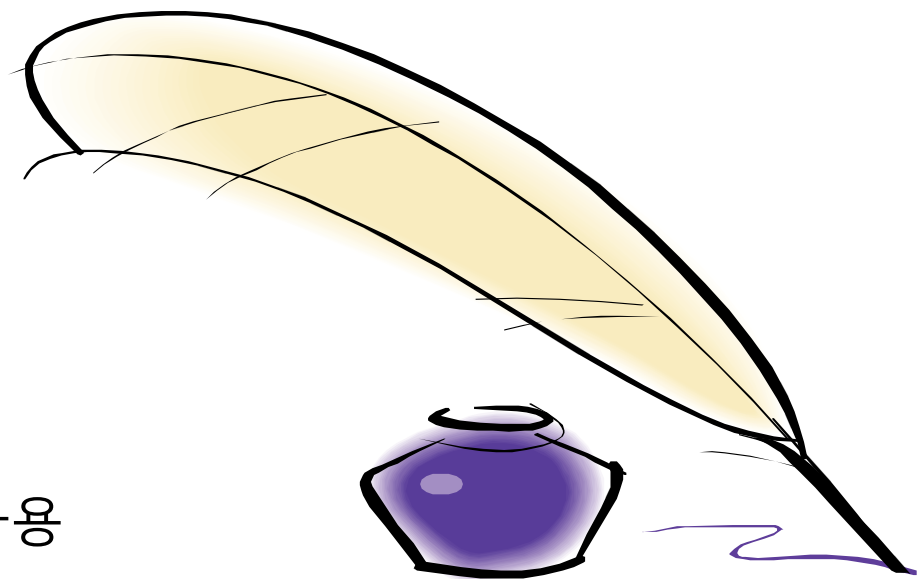
Dept. of Game Software

Notices

- 03/20: HW 1 (Due: 03/26) → Avg: 8.85

Plan

- MFC로 그림 그리기 2
 - Pen & Brush
 - 연습 1: CPen 사용
 - 연습 2: CBrush 사용
 - Font
 - 연습 3: CFont 사용
 - Polygon
 - 연습 4: CRgn 사용
 - Bitmap
 - 연습 5: CBitmap 사용
 - 연습 6: Bitmap 출력
 - Double Buffering
 - 연습 7: Double buffering 사용



GDI를 이용하여 그림을 그리는 것

- 기본값:



- 색을 바꾸거나 특성을 바꿀 수 있을까?

- Pen의 특성을 바꾼다 : 선
- Brush의 특성을 바꾼다: 면



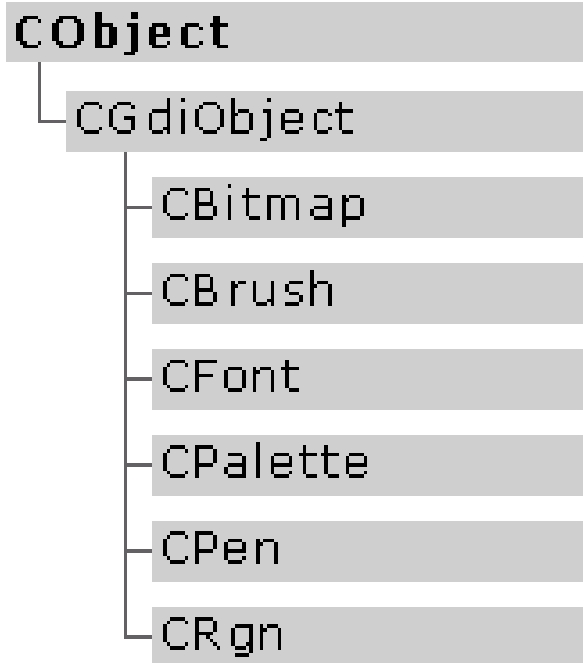
GDI 객체 (1/3)

- 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	문자를 출력할 때	CFont
비트맵	픽셀의 집합으로 이루어진 그림을 다룰 때	CBitmap
팔레트	출력될 색의 집합을 다룰 때	CPalette
영역	다양한 형태의 면을 정의할 때	CRgn

GDI 객체 (2/3)

- 클래스 계층도



GDI 객체 (3/3)

- 그림을 그리는 순서 (일반적인 경우)
 1. Pen을 고름
 2. Pen을 줍
 3. 그림을 그림
 4. Pen을 내려놓음



GDI 객체 (3/3)

- 그림을 그리는 순서 (GDI객체사용)
 1. 사용할 객체(pen)를 정의
 2. DC에 이 객체를 지정 (CDC::SelectObject())
(이전에 가지고 있던 객체를 임시저장)
 3. 그림을 그림
 4. 사용할 객체를 선택해제
(이전에 가지고 있던 객체로 환원)



펜: CPen

- 생성 방법

// 방법 1

CPen pen(PS_SOLID, 2, RGB(255, 0, 0));

// 생성자

// 방법2

CPen pen;

pen.CreatePen (PS_SOLID, 2, RGB (255, 0, 0));

// 초기화함수

스타일 폭 색

- 펜 스타일

PS_SOLID

PS_DASH

- - - - -

PS_DOT

.....

PS_DASHDOT

- . - . - .

PS_DASHDOTDOT

- . - . - .

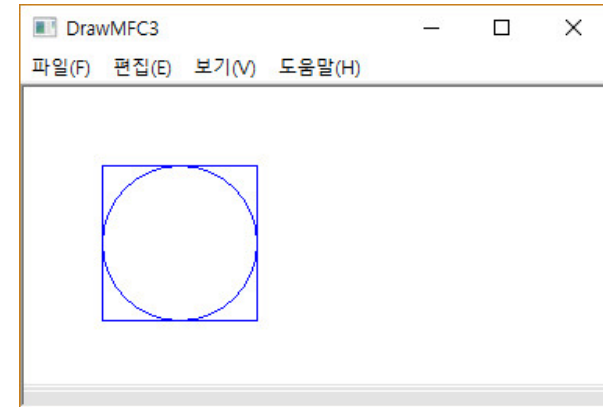
PS_NULL

PS_INSIDEFRAME

연습 1: CPen 사용

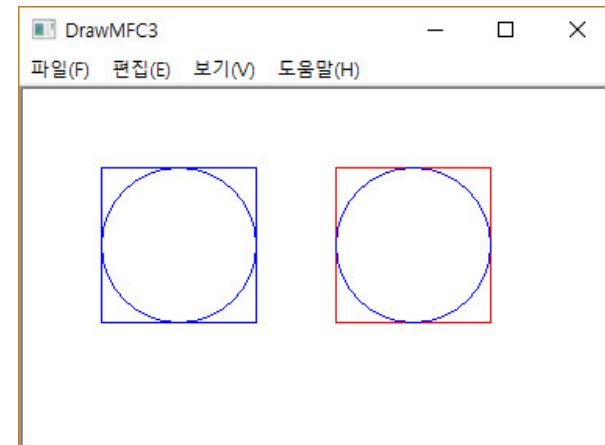
- 사용 예 1: blue 사용

```
CPen pen1(PS_SOLID, 1, RGB(0, 0, 255));  
dc.SelectObject(&pen1);  
dc.Rectangle(50, 50, 150, 150);  
dc.Ellipse(50, 50, 150, 150);
```



- 사용 예 2: blue 저장 후 red 사용

```
CPen pen2(PS_SOLID, 1, RGB(255, 0, 0));  
CPen *pOldPen = dc.SelectObject(&pen2);  
dc.Rectangle(200, 50, 300, 150);  
dc.SelectObject(pOldPen);  
dc.Ellipse(200, 50, 300, 150);
```



브러시: CBrush

- 브러시란? : 면을 어떻게 채우는가를 정의하는 것
- 브러시종류 (생성자에 따라 결정됨)

브러시 종류	생성 예
솔리드(Solid, 속이 채워짐)	<code>CBrush brush(RGB(255, 0, 0));</code>
해치(Hatch, 교차된 평행선 무늬)	<code>CBrush brush(HS_DIAGCROSS, RGB(255, 0, 0));</code>
패턴(Pattern, 비트맵 의 반복 무늬)	<code>CBitmap bitmap; bitmap.LoadBitmap(IDB_BITMAP1); CBrush brush(&bitmap);</code>

연습 2: CBrush 사용

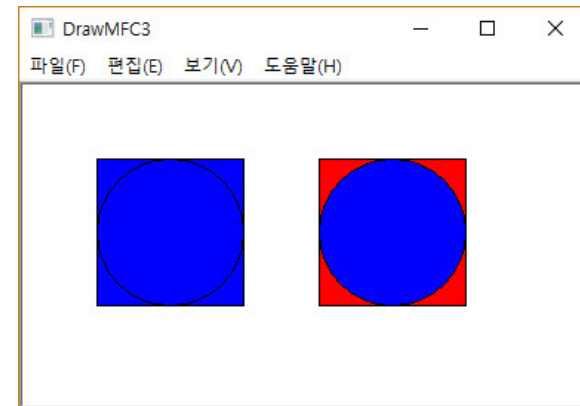
- 사용 예 1: blue 사용

```
CBrush brush1(RGB(0, 0, 255));  
dc.SelectObject(&brush1);  
dc.Rectangle(50, 50, 150, 150);  
dc.Ellipse(50, 50, 150, 150);
```



- 사용 예 2: blue 저장 후 red 사용

```
CBrush brush2(RGB(255, 0, 0));  
CBrush *pOldBrush = dc.SelectObject(&brush2);  
dc.Rectangle(200, 50, 300, 150);  
dc.SelectObject(pOldBrush);  
dc.Ellipse(200, 50, 300, 150);
```



폰트: CFont

- 생성 방법
 - CFont 객체 생성
 - CFont 객체에 대해 CreateFont() 함수를 호출

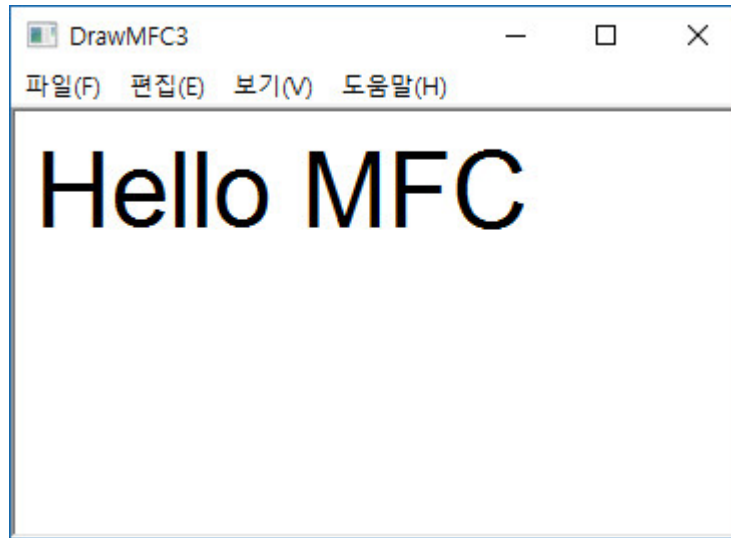
```
CFont font;  
font.CreateFont(...);  
// font.CreateFontIndirect(...);  
// font.CreatePointFont(...);  
// font.CreatePointFontIndirect(...);
```

연습 3: CFont 사용

- 사용 예

```
CFont font;  
font.CreatePointFont(400, _T("Arial"));  
dc.SelectObject(&font);  
dc.TextOut(10, 10, _T("Hello MFC"));
```

크기 글씨체이름



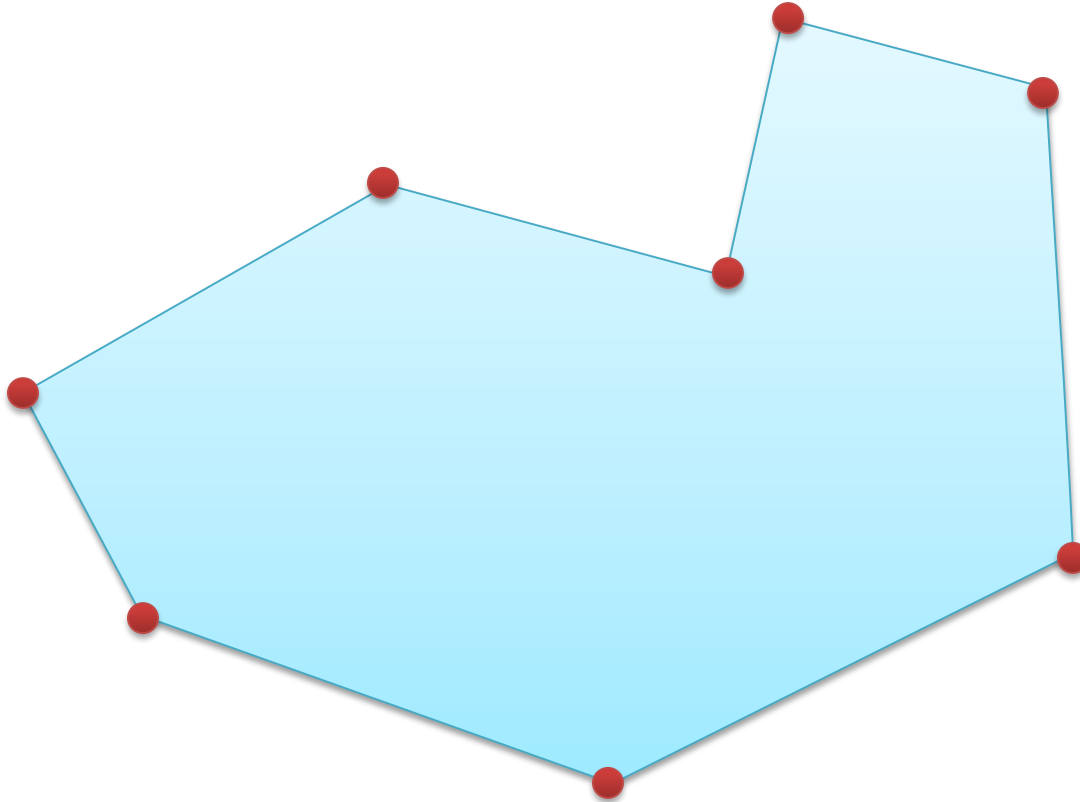
미리 정의된 내장 객체

이름	용도
BLACK_PEN	폭이 1 픽셀인 검정색 펜
WHITE_PEN	폭이 1 픽셀인 흰색 펜
NULL_PEN	투명 펜
BLACK_BRUSH	검정색 브러시
DKGRAY_BRUSH	어두운 회색 브러시
GRAY_BRUSH	회색 브러시
LTGRAY_BRUSH	밝은 회색 브러시
HOLLOW_BRUSH 또는 NULL_BRUSH	투명 브러시
SYSTEM_FONT	윈도우 운영체제가 사용하는 폰트 예) 메뉴, 대화상자, ...

- CDC::SelectStockObject() 함수를 사용하여 디바이스 컨텍스트에 선택한다.

CRgn: 사각형(원)이 아닌 영역 객체

- 다각형 (Polygon)



점들의 집합으로 정의됨

CRgn: 사각형(원)이 아닌 영역 객체

- CPoint
 - 2차원 위치 정보를 표현하는 클래스
 - 멤버변수: x, y

```
// 위치 정보를 멤버변수에 할당
CPoint pt1;
pt1.x = 100;
pt1.y = 200;

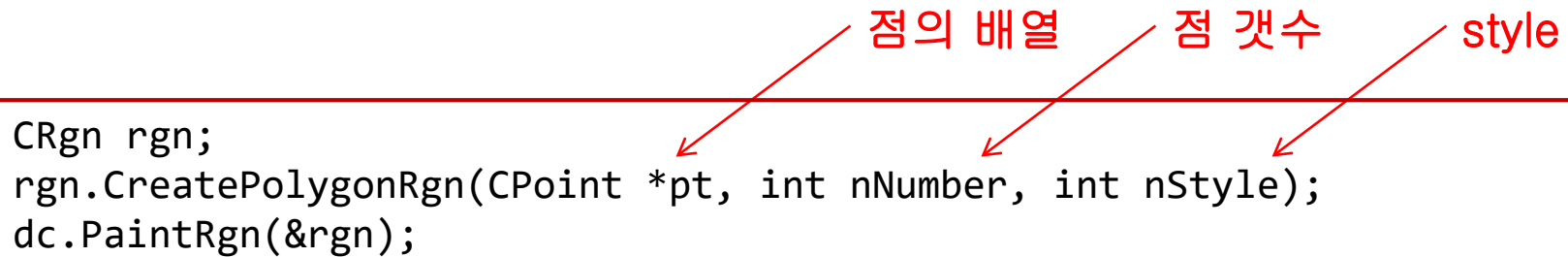
// 생성자를 통한 할당
CPoint pt2(300, 200);

// 할당연산자를 통한 할당
CPoint pt3;
pt3 = pt2;

// Copy 생성자를 통한 할당
CPoint pt4(pt1);
```

CRgn: 사각형(원)이 아닌 영역 객체

- 사용방법



The diagram shows the `CRgn.CreatePolygonRgn` method signature with three red arrows pointing to its parameters from labels above. The label '점의 배열' (Array of points) points to the `CPoint *pt` parameter. The label '점 갯수' (Number of points) points to the `int nNumber` parameter. The label 'style' points to the `int nStyle` parameter.

```
CRgn rgn;  
rgn.CreatePolygonRgn(CPoint *pt, int nNumber, int nStyle);  
dc.PaintRgn(&rgn);
```

nStyle: ALTERNATE or WINDING

연습 4: CRgn 사용

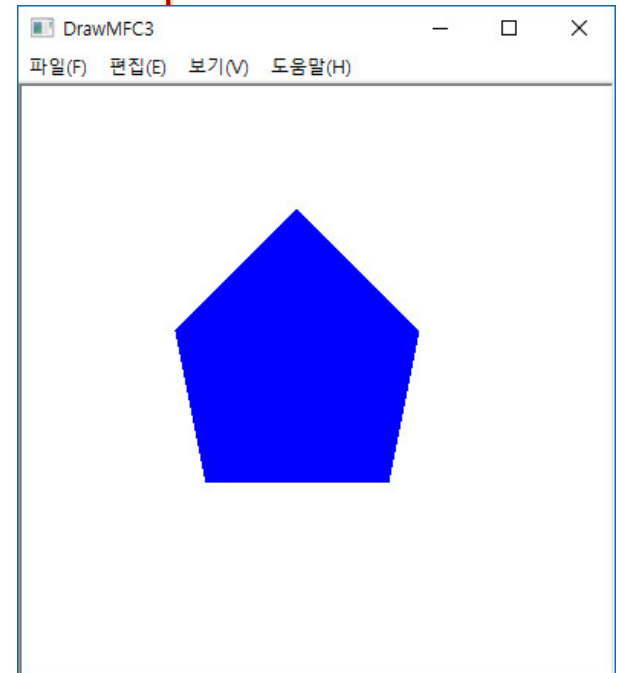
- Blue 오각형을 그리시오.

```
CRgn    rgn;
CPoint  ptVertex[5];
ptVertex[0] = CPoint(180,80);
ptVertex[1] = CPoint(100,160);
ptVertex[2] = CPoint(120,260);
ptVertex[3] = CPoint(240,260);
ptVertex[4] = CPoint(260,160);

rgn.CreatePolygonRgn( ptVertex, 5, ALTERNATE);
CBrush  brush(RGB(0,0,255));
dc.SelectObject(&brush);
dc.PaintRgn(&rgn);

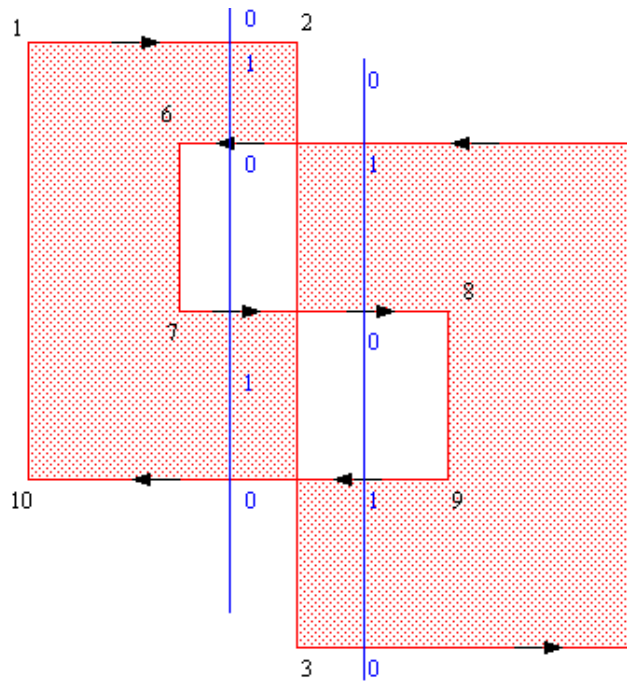
// or

dc.FillRgn(&rgn, &brush);
```

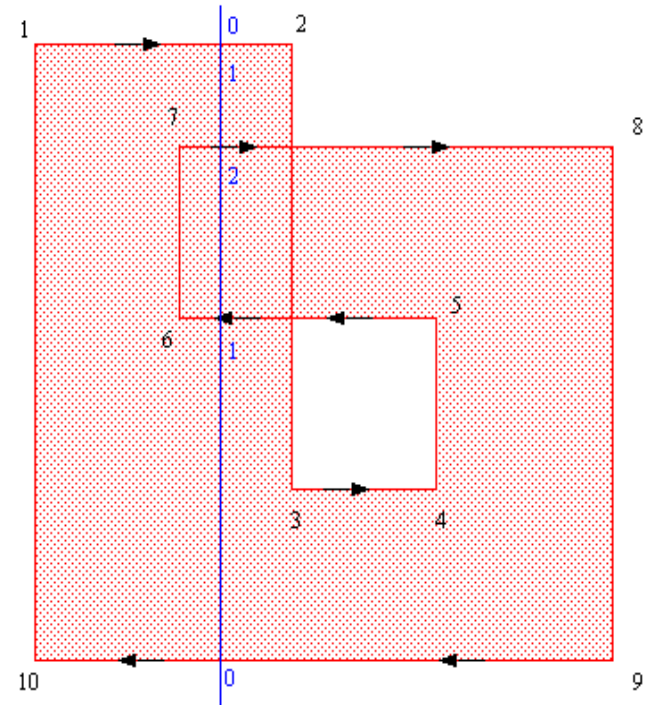


CRgn: 사각형(원)이 아닌 영역 객체

- ALTERNATE or WINDING ?
 - 면의 내부를 채우는 알고리즘의 선택

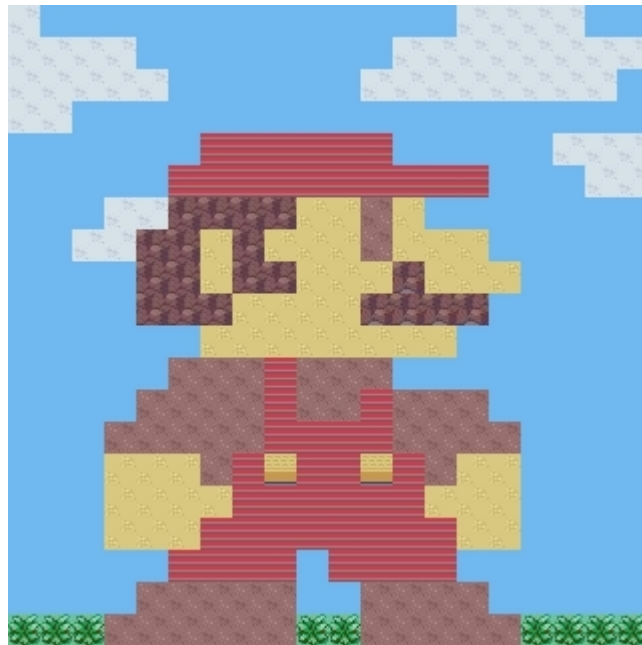


Alternate filling



Winding filling

비트맵 (Bitmap)



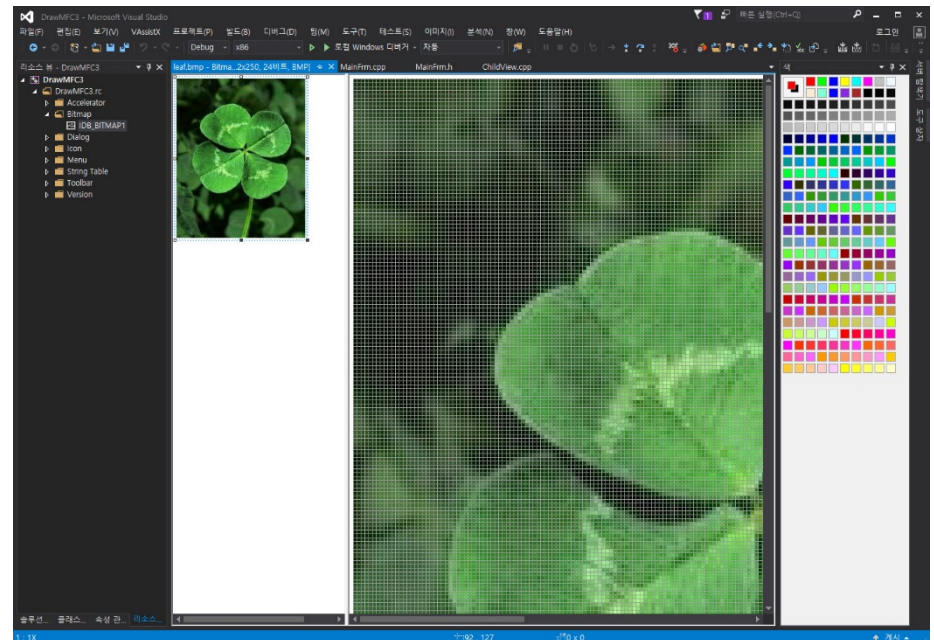
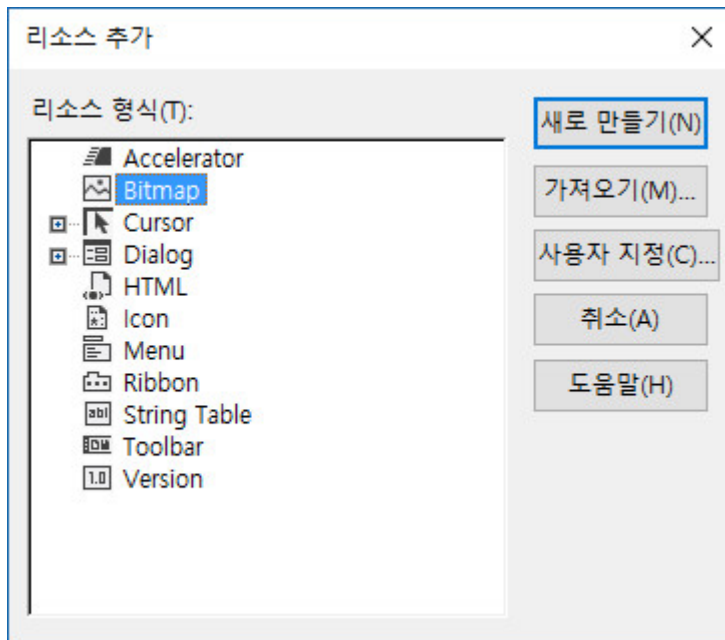
비트맵이란?

- 그림을 dot(pixel)로 표현 하는 것

ASCII : 65	0 to 7
	00 00000000
	00 00000000
	08 00001000
	1C 00011100
	36 00110110
	63 01100011
	63 01100011
	7F 01111111
	63 01100011
	63 01100011
	63 01100011
	63 01100011
	00 00000000
	00 00000000
	00 00000000
	00 00000000

비트맵이란?

- 간단한 비트맵 만들기
 - 리소스 뷰에서 리소스 추가를 사용



연습 5: CBitmap 사용

- 사용법: Brush에 로딩

```
CBitmap bitmap;  
bitmap.LoadBitmap(IDB_BITMAP1);  
  
CBrush brush(&bitmap);  
dc.SelectObject(&brush);  
dc.Rectangle(0, 0, 200, 200);
```

리소스 아이디



CBitmap

- 비트맵 정보

```
int CBitmap::GetBitmap (BITMAP* pBitMap) ;

struct BITMAP {
    int bmType;
    int bmWidth;           // 비트맵의 폭(픽셀 단위)
    int bmHeight;          // 비트맵의 높이(픽셀 단위)
    int bmWidthBytes;
    BYTE bmPlanes;
    BYTE bmBitsPixel;
    LPVOID bmBits;
};
```

연습 5: Cbitmap 사용

- 비트맵 정보 출력

```
// 이어서
CBitmap bitmap;
bitmap.LoadBitmap(IDB_BITMAP1);
BITMAP bmpinfo;
bitmap.GetBitmap(&bmpinfo);

// CString: VC++에서 지원하는 문자열 클래스
CString str;
// Format(..): printf와 같은 기능을 하는 멤버 함수
str.Format(_T("가로 = %d, 세로 = %d\n"),
           bmpinfo.bmWidth, bmpinfo.bmHeight);

dc.TextOut(100, 100, str);
```



비트맵을 직접 출력한다면...

- 한 점씩 그리지 않고 한꺼번에 메모리로 보낸다
(Block) (Transfer)
- 절차:
 - 원래의 도화지
 - 또 다른 도화지 준비
 - 그 도화지에 그림 그리기
 - 원래의 도화지로 그린거 오려붙이기

비트맵을 직접 출력한다면...

- 한 점씩 그리지 않고 한꺼번에 메모리로 보낸다
(Block) (Transfer)

- 절차:

- 원래의 도화지 `CPaintDC dc;`
- 또 다른 도화지 준비 `CDC::CreateCompatibleDC(..);`
- 그 도화지에 그림 그리기 `CDC::SelectObject(...);`
- 원래의 도화지로 그린거 오려붙이기 `dc.BitBlt(...);`

↑
Bit Block Transfer (BitBlt)

연습 6: Bitmap 출력

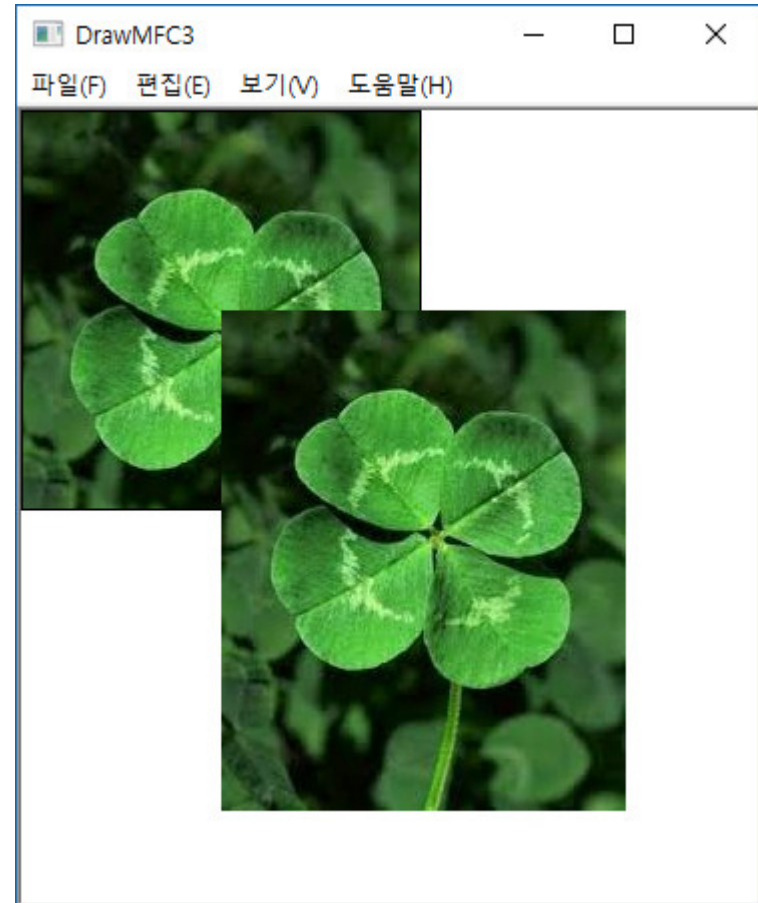
- (100, 100)에 비트맵의 원래 크기로 올려 붙이시오.

```
CPaintDC dc(this);

CBitmap bitmap;
bitmap.LoadBitmap(IDB_BITMAP1);
BITMAP bmpInfo;
bitmap.GetBitmap(&bmpInfo);

CDC memDc;
memDc.CreateCompatibleDC(&dc);
memDc.SelectObject(&bitmap);

dc.BitBlt(100, 100,
    bmpInfo.bmWidth, bmpInfo.bmHeight,
    &memDc, 0, 0, SRCCOPY);
```

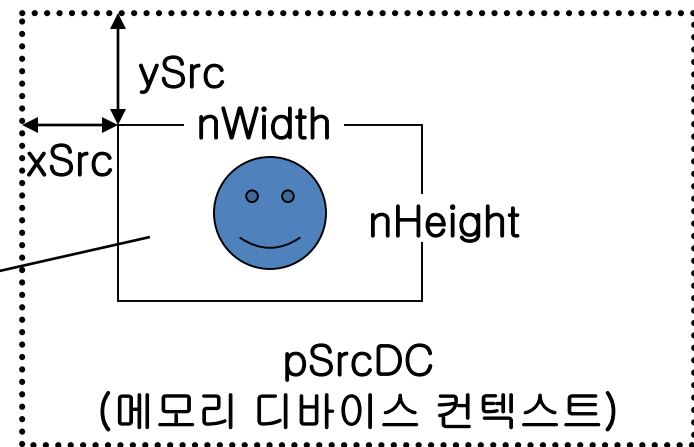
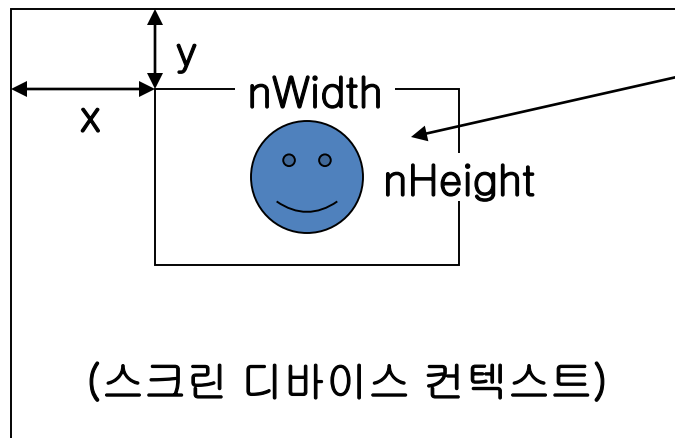


CBitmap

- 비트맵 출력 함수

```
BOOL BitBlt (int x, int y, int nWidth, int nHeight, CDC* pSrcDC,  
             int xSrc, int ySrc, DWORD dwRop);
```

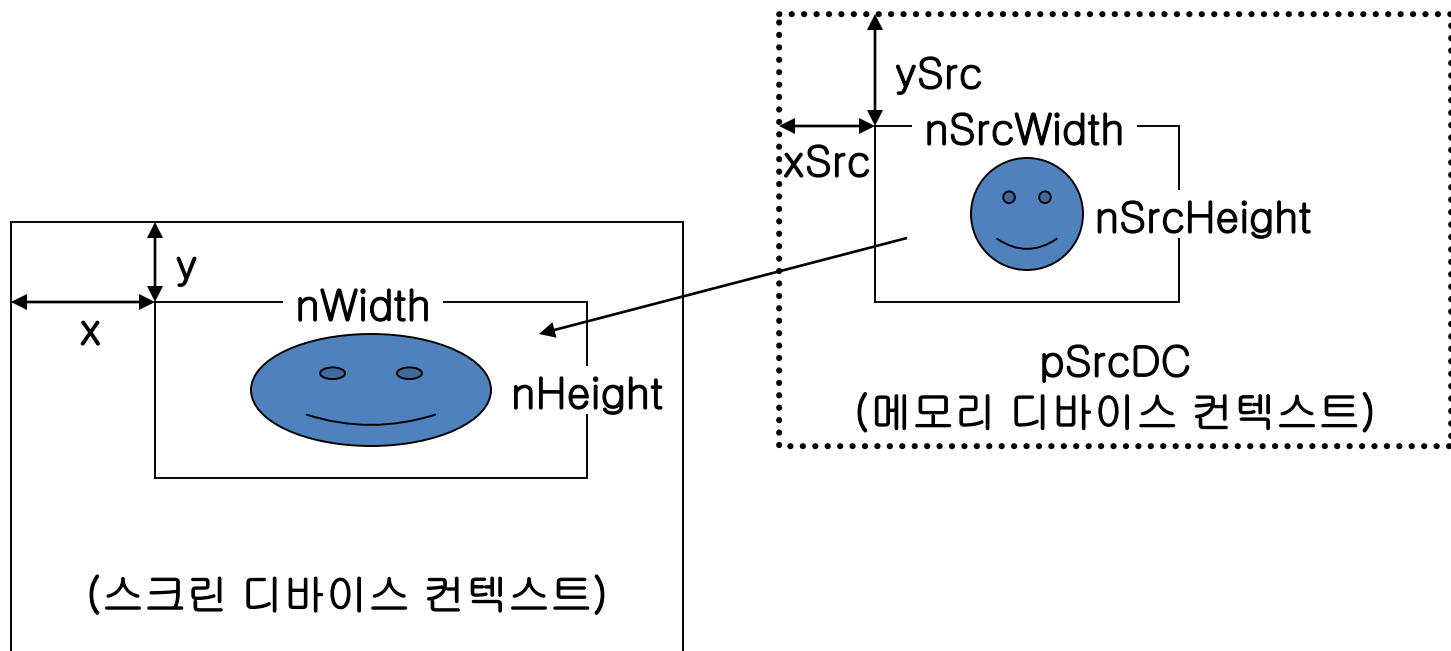
그림을 복사하는 방법:
SRCCOPY



CBitmap

- 비트맵 출력 함수 (cont'd)

```
BOOL StretchBlt (int x, int y, int nWidth, int nHeight, CDC* pSrcDC, int  
xSrc, int ySrc, int nSrcWidth, int nSrcHeight, DWORD dwRop) ;
```



Memory에 그림그리고 넘겨주기

- 절차:

1. 메모리DC 만들기 (CreateCompatibleDC)
2. DC와 Bitmap을 연결 (SelectObject)
3. 그림 그리기
4. 그려진 그림을 화면DC로 BitBlt

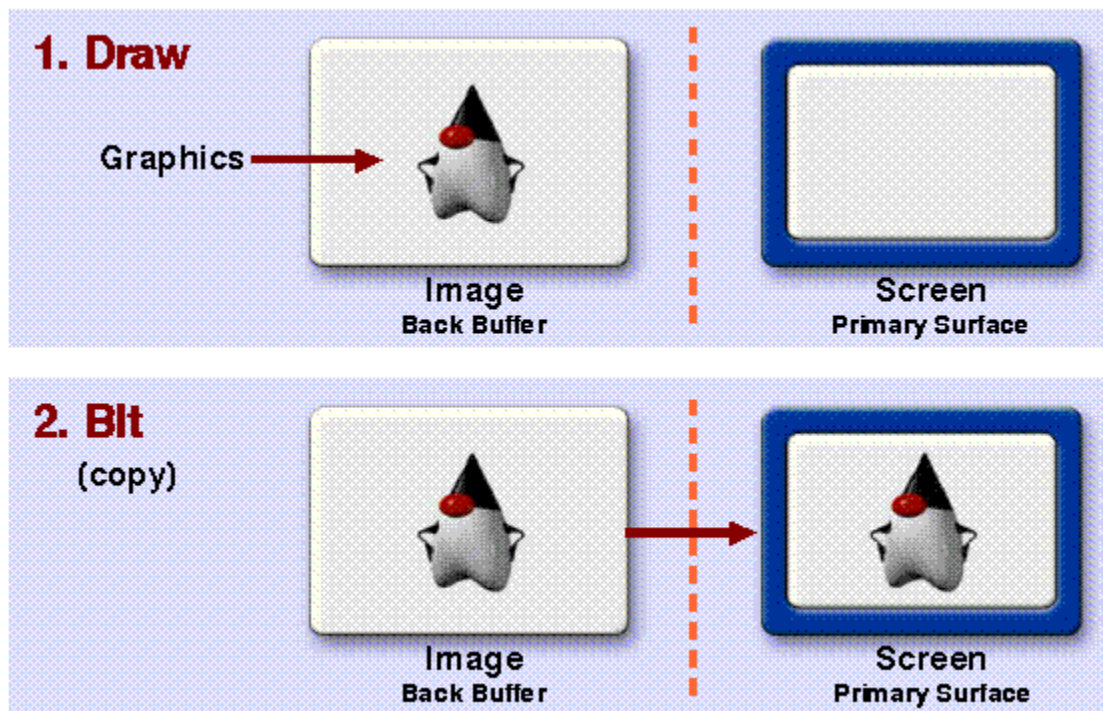
좀 더 진보된 그리기 방법:
더블 버퍼링
(Double Buffering)

여태까지 그림 그리는 방법은...

- 그림을 다시 그릴 때 그리는 과정이 보인다.
 - 특히 느린 컴퓨터를 사용할 때..
 - Why?
- 그림을 다시 그릴 때 깜박거린다.
 - 원래 내용을 무효화(invalidate) 한다.
 - = 먼저 하얗게 지운다

Double Buffering

- 두 개의 Device Context(도화지)를 이용함.
 1. 그림을 그리는 용도: Back buffer
 2. 그림을 보여주는 용도: Front screen buffer



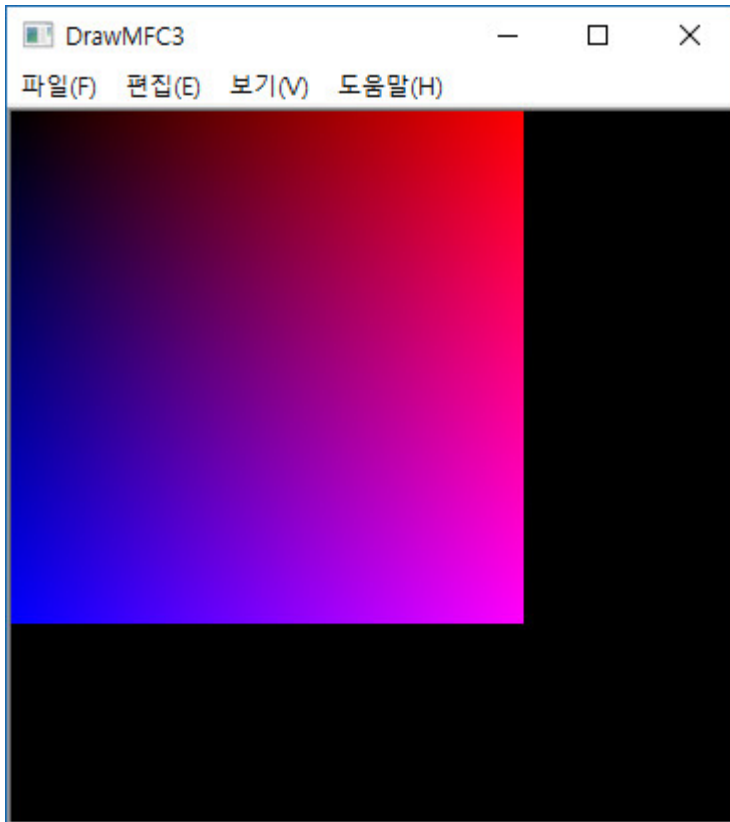
Memory에 그림그리고 넘겨주기

한꺼번에

- 절차:
 - 1. 메모리DC 만들기 (CreateCompatibleDC)
 - 2. 그림그릴 메모리 만들기 (CreateCompatibleBitmap)
 - 3. DC와 Bitmap을 연결 (SelectObject)
 - 4. 그림 그리기
 - 5. 그려진 그림을 화면DC로 BitBlt
- ➔ 그림을 다른 도화지에 그리고, 도화지를 빠르게 바꿔 치기:
더블 버퍼링(Double Buffering)

연습 7: Double Buffering 사용

- Blue와 red 사이의 색상이 점점 변하는 사각형을 출력하시오. 이 때, 빠른 출력을 위해 double buffering을 사용하시오.



```
for(int x = 0 ; x < 256 ; x++)  
    for(int y = 0 ; y < 256 ; y++)  
        dc.SetPixelV(x, y, RGB(x, 0, y));
```

윈도우 사이즈를 조절하면?

연습 7: Double Buffering 사용

- 빠른 출력을 위해 double buffering 사용

```
CRect rect;
GetClientRect(rect);

CDC memDC;          // 가상 DC
memDC.CreateCompatibleDC(&dc);

CBitmap bitmap;     // 그림을 저장할 공간 마련
bitmap.CreateCompatibleBitmap(&dc, rect.Width(), rect.Height());
memDC.SelectObject(&bitmap);

for(int x = 0 ; x < 256 ; x++)
    for(int y = 0 ; y < 256 ; y++)
        memDC.SetPixelV(x, y, RGB(x, 0, y));

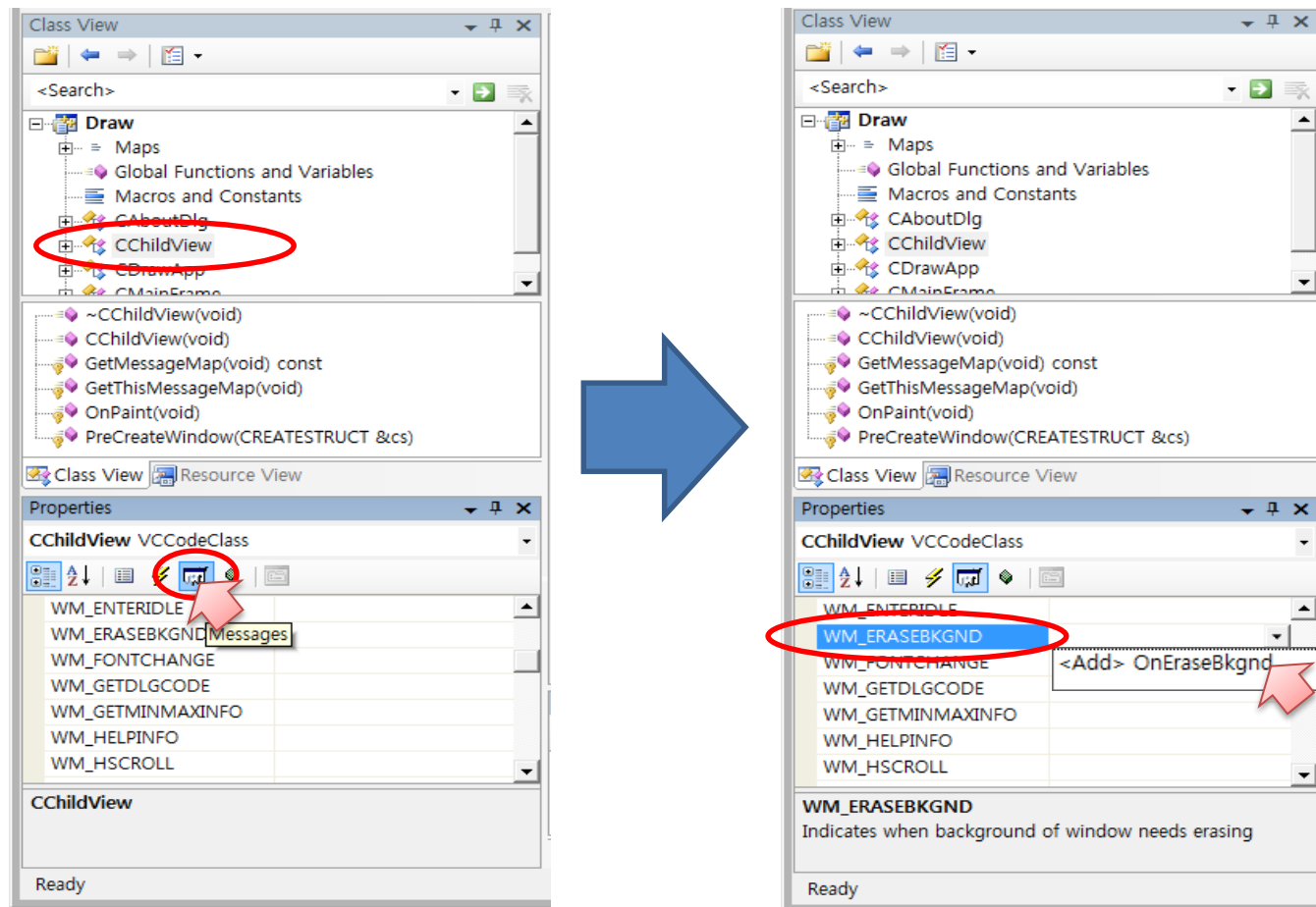
dc.BitBlt(0, 0, rect.Width(), rect.Height(), &memDC, 0, 0, SRCCOPY);
```

Double Buffering using bitmap

- 그림이 빨리 그려진다!
- 그러나, 여전히 깜박인다?
 - 화면을 매번 깨끗이 지우기 때문!
 - “화면을 지울 필요가 있다”라는 메시지를 받아서 지우지 말자고 하자.
- 해결책:
WM_ERASEBKGDND 메시지 핸들러를 만들어 아무 일도 하지 말자고 한다.

WM_ERASEBKGND 핸들러 추가

- WM_ERASEBKGND: 화면을 지울 때(하얗게 칠할 때) 불리는 메시지



WM_ERASEBKGND 핸들러 추가

- WM_ERASEBKGND: 화면을 지울 때(하얗게 칠할 때) 불리는 메시지
- 화면을 지우지 않게 하기 위해 아무 일도 하지 않고 그냥 return

```
BOOL CChildView::OnEraseBkgnd(CDC* pDC)
{
    // TODO: Add your message handler code here and/or call default

    // return CWnd::OnEraseBkgnd(pDC);
    return true;
}
```

Q & A