GDI(Graphic Device Interface)

- USER, KERNEL와 함께 윈도우즈를 구성하는 핵심 모듈 중 하나이며, 출력장치 제어에 관여한다

- GDI의 목적은 다양한 그래픽장치 및 출력장치에 대해 하나의 공통된 인터페이스를 제공하는 것

GDI Object

- GDI를 통해 사용할 수 있는 각종 그리기 정보를 포함하는 객체

- 간단하게는 그래픽 출력에 사용되는 도구를 말하며, 펜, 브러시, 비트맵 등이 여기에 속한다

DC(Device Context)

- Windows OS에서 폰트, 색상, 굵기 등 화면 출력에 필요한 모든 정보를 가지는 데이터 ‘구조체’

- 그래픽 객체들의 속성과 그래픽 모드를 정의하는 자료의 집합체로 모든 출력장치에 관여한다

- 예를 들어, 모니터 화면에 출력을 하려면 해당 출력장치의 DC 핸들을 얻어서 제어해야 한다.

- 각각의 윈도우는 하나의 DC를 공유하는 것이 아니라 각각의 DC를 가지고 있는 것이다.

- 윈도우 프로그램은 윈도우의 화면에 출력되는 모든 내용을 DC를 이용하여 출력하게 된다.

- 운영체제는 프로그램이 DC를 요구할 때마다 적절한 초기값을 대입하여 그 핸들을 넘겨준다.

- 프로그램은 넘겨받은 DC핸들로 그림을 그리며, 정보가 정확할수록 정밀하게 그릴 수 있다.

Display DC(화면 출력용 DC)

1. Common DC : 작업영역에 그리기를 하는 가장 보편적인 DC이며, 화면 출력을 위해 사용된다.

- GetDC() & ReleaseDC()로 얻거나 해제하는 DC는 여기에 속한다.

2. Window DC : Common DC와 사용방법이 비슷하지만 원점이 되는 기준이 다르다..

(Common DC의 경우 원점이 작업영역의 좌상단이지만, Window DC의 경우 작업영역이 원점)

- GetWindowDC() & ReleaseDC()로 얻거나 해제하는 DC는 여기에 속한다

🡺 참고로 GetDC()로 화면 DC를 얻을 경우 지워진 영역(무효 영역)에 대해서는 보장해주지 않음

Memory DC(비트맵 출력용 DC)

- 디스플레이 DC와 동일한 특성을 갖지만 메모리 DC는 화면에 출력하지 않고 메모리 내부에서 출력을 한다는 점의 차이가 있다.

- 윈도우는 비트맵을 직접적으로 화면 DC로 출력하는 함수를 제공하지 않는다

- 비트맵은 대용량이므로 출력 속도가 형편없이 느리기 때문에 곧바로 출력할 경우 문제가 발생

- 메모리 DC에 먼저 그림을 그린 후 화면에 출력은 하지않고 메모리 DC에서 작업을 완료한 후 그 결과만 화면에 고속으로 복사해서 출력하는 방법을 사용한다 -> BitBlt() 이용

- 비트맵은 일종의 GDI 오브젝트지만 화면 DC에서는 선택할 수 없고, 메모리 DC에서만 비트맵을 선택할 수 있기 때문에 메모리 DC에서 먼저 비트맵을 읽어온 후 화면 DC로 복사하는 것이다.

무효영역(Update Region) - 윈도우의 화면 일부가 지워지거나 변경되어 다시 그려져야 하는 부분

만약 윈도우 전체가 다 변경되는 경우라면 무효영역을 작업영역과 동일 하지만 윈도우의 일부분만 변경되었다면 변경된 부분만 무효영역이 된다. 운영체제는 최대한 좁은 면적을 무효영역으로 만들어 그리기에 소요되는 시간을 최대한 적게 소모하게 한다.

클리핑영역 - 운영체제가 실제 그리기에 사용하는 영역으로 무효영역 & 가시영역의 교집합

클리핑영역은 일반적으로 무효영역 보다 좁다. 운영체제는 무효영역 중에서도 최소한의 영역만을 그리기 위해, 또 다른 윈도우 영역을 침범하지 않기 위해 클리핑영역을 사용한다.

윈도우에 무효영역이 발생하면 OS는 WM\_PAINT 메시지를 발생시키고, WM\_PAINT 메시지 함수 내부의 BeginPaint 함수가 호출되면서 무효영역을 유효영역으로 바꿔주게 된다.  
  
① 그리기를 위해 DC를 발급받는다. 이 DC핸들은 BeginPaint의 리턴값으로 반환되기도 한다

② 클리핑 영역을 조사하여 DC에 설정한다. 모든 출력함수는 DC에 설정된 클리핑 영역을 참고하여 영역 바깥으로 출력되는 것을 잘라낸다. 이때 무효영역, 윈도우의 Z 순서, 보이기 상태 등의 여러가지 정보를 참조한다.  
  
③ 무효영역을 없애고(=유효화) 다시 WM\_PAINT 메시지가 호출되지 않도록 한다. 만약 화면만 다시 그리고 무효영역을 그대로 내버려두면 운영체제는 계속해서 WM\_PAINT 메시지를 보낼 것이다.  
  
④ 다시 그려지는 영역에 캐럿이 있다면 그리기를 시작하기 전에 숨겨 캐럿이 파괴되지 않도록 한다. 이렇게 숨겨진 캐럿은 그리기를 끝내는 시점인 EndPaint에 의해 복구된다.  
  
⑤ 윈도우의 배경을 백그라운드 브러시로 지우기 위해 WM\_ERASEBKGND 메시지를 보내 처리한다. 또한 WM\_NCPAINT 메시지를 보내 비작업영역을 그리도록 한다.  
  
응용프로그램이나 운영체제에 의해 설정된 무효영역을 WM\_PAINT의 BeginPaint 함수에 의해 제거 됨으로써 새로운 무효영역이 생기기 전에는 WM\_PAINT 메시지가 전달되지 않는다.  
무효영역이 있으므로 WM\_PAINT가 발생하는 것인데, 만약에 WM\_PAINT 메시지 처리 루틴에서 무효영역을 없애지 않는다면 WM\_PAINT가 계속 발생 할 것이다.  
WM\_PAINT는 다른 메시지와는 달리 그릴게 없다고 하더라도 메시지 처리코드를 비워두어선 안된다.

WM\_PAINT 메시지는 모든 메시지 중에서 우선 순위가 가장 낮으며, 메시지를 보내는 경우는 두 가지 조건이 충족되었을 때이다.

첫째, 메시지 큐에 대기중인 메시지가 없어야 한다.  
사용자의 입력을 처리하는 일이 화면 그리기보다 더 중요하기 때문에 입력 처리상태에서는 그리기를 하지않는다.

둘째, 무효영역이 있어야 한다.   
아무리 한가해도 다시 그려져야 할 부분이 없으면 WM\_PAINT 메시지를 보낼 필요가 없다.  
WM\_PAINT를 큐에 넣는 주체는 운영체제이며 응용프로그램이 직접 WM\_PAINT 메시지를 큐에 넣지 않는다. 응용프로그램은 단순히 무효영역을 만들 뿐이며, 무효영역이 있으면 WM\_PAINT가 메시지 큐에 자동으로 들어간다. 운영체제는 두 개의 WM\_PAINT 메시지가 큐에 있으면 이 둘을 합쳐 하나로 만들되 두 메시지가 가진 무효영역의 합집합을 새로운 무효영역으로 설정한다.

InvalidateRgn, InvalidateRect - 프로그램 내부에서 윈도우의 모습을 변경시켰을 때 변경된 부분이 다시 그려질 수 있도록 강제로 무효화를 시켜주는 함수(WM\_PAINT가 무효화된 부분을 자동으로 다시 그림), 호출한다고 해서 즉시 그리기를 하지 않으며 무효영역을 모아서 한꺼번에 그린다.

화면이 무효화 되는 시점과 실제로 그려지는 시점은 비동기적이다. 무효영역이 있더라도 다른 메시지가 큐에 있다면 WM\_PAINT메시지는 대기중인 메시지가 처리되는 동안에는 전송되지 않는다.  
이럴 때 무효화되는 화면을 즉시 그리게 할 수 있는데 이 때 사용하는 함수가 UpdateWindow이다. 이 함수는 WM\_PAINT를 메시지 큐를 거치지 않고 곧바로 WndProc으로 보내므로 다른 메시지보다 우선해서 처리하도록 한다.