



WINAPI

-CAPTER 2-

SOULSEEK

목차

1. DC

2. WM_PAINT

3. 문자열 출력

4. 점, 선, 도형 출력

1. DC

1. DC

- 출력에 필요한 모든 정보를 가지는 데이터 구조체이며 GDI 모듈에 의해 관리된다.
- 출력에 사용되는 최소한의 정보만 제공하면 나머지는 DC에 의해 정해진 동작을 한다.
- 해당 윈도우의 핸들을 이용한 타겟 윈도우를 지정해준다.

```
LRESULT CALLBACK WndProc(HWND hWnd, UINT iMessage, WPARAM wParam, LPARAM lParam)
{
    HDC hdc;
    switch (iMessage)
    {
        case WM_DESTROY:
            PostQuitMessage(0);
            return 0;
        case WM_LBUTTONDOWN:
            hdc = GetDC(hWnd);
            TextOut(hdc, 100, 100, TEXT("Beautiful Korea"), 15);
            ReleaseDC(hWnd, hdc);
            return 0;
    }
    return(DefWindowProc(hWnd, iMessage, wParam, lParam));
}
```

1. GetDC()

- **Window** 핸들 값을 받아서 적당한 **DC**을 얻어온 후 반환한다.
- 어떤 메시지에서도 모두 사용 가능하며, **Window**의 상태가 바뀌어서 변화된 정보는 **DC**에 출력하여 적용시켜주지 않는다.
- 한번 받아온 **DC**는 반듯이 해제해주어야 하는데 **GetDC**는 **ReleaseDC()**로 해제해 줘야 한다.

2. TextOut()

- 화면에서 해당 문자열을 출력시키는 함수.
- **DC**의 핸들과 좌표 값과 출력문자열의 정보만으로 문자열을 출력할 수 있다.

3. WM_LBUTTONDOWN

- 마우스 좌클릭을 했을 때 호출되는 메시지.

2. WM_PAINT

2. WM_PAINT

```
LRESULT CALLBACK WndProc(HWND hWnd, UINT iMessage, WPARAM wParam, LPARAM lParam)
{
    HDC hdc;
    PAINTSTRUCT ps;
    switch (iMessage)
    {
        case WM_DESTROY:
            PostQuitMessage(0);
            return 0;

        case WM_PAINT:
            hdc = BeginPaint(hWnd, &ps);
            TextOut(hdc, 100, 100, TEXT("Beautiful Korea"), 15);
            EndPaint(hWnd, &ps);
            return 0;
    }
    return(DefWindowProc(hWnd, iMessage, wParam, lParam));
}
```

1. PAINTSTRUCT

- 그리기 전용 구조체이고 **BeginPaint**에 사용한다.

2. BeginPaint()

- 윈도우 핸들값과 그리기 전용 구조체를 받아서 **그리기용 DC**를 생성한 후 반환한다.
- **WM_PAINT**라는 메시지에서만 사용이 가능하며 창의 정보가 계속 변하여도 그린 부분이 복원될 필요성을 알려줘서 **다시 그린다**.
- 받아서 사용한 **DC**는 **EndPaint()**함수로 **해제**해 줘야한다.

3. 문자열 출력

3. 문자열 출력

```
case WM_PAINT:
```

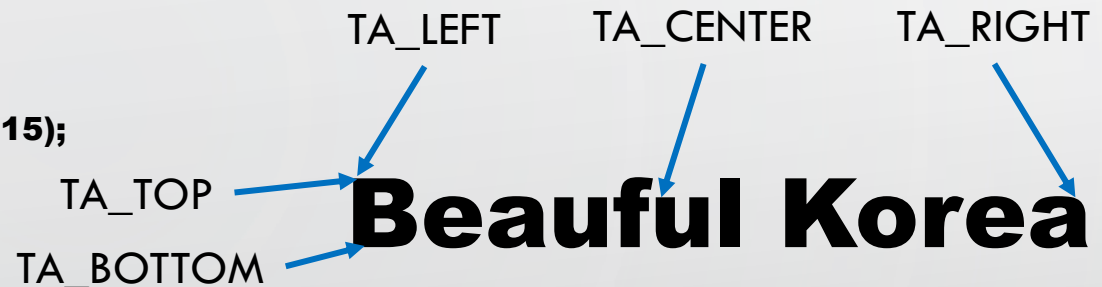
```
hdc = BeginPaint(hWnd, &ps);
```

```
SetTextAlign(hdc, TA_CENTER);
```

```
TextOut(hdc, 100, 100, TEXT("Beautiful Korea"), 15);
```

```
EndPaint(hWnd, &ps);
```

```
return 0;
```



- **SetTextAlign(HDC, UINT)**

- **TEXT** 출력 중심점을 설정한다.
- 기본 설정은 **TA_TOP | TA_LEFT**

값	설명
TA_TOP	지정된 좌표가 상단이 중심인 좌표가 된다.
TA_BOTTOM	지정된 좌표가 하단이 중심인 좌표가 된다.
TA_CENTER	지정된 좌표가 수평중앙이 중심인 좌표가 된다.
TA_RIGHT	지정된 좌표가 수평 오른쪽이 중심인 좌표가 된다.
TA_LEFT	지정된 좌표가 수평 왼쪽이 중심인 좌표가 된다.
TA_UPDATECP	지정된 좌표 대신 CP를 사용하여 문자 출력 후 CP를 변경한다.
TA_NOUPDATECP	CP를 사용하지 않고 지정한 좌표로 출력하며 CP를 변경하지 않는다

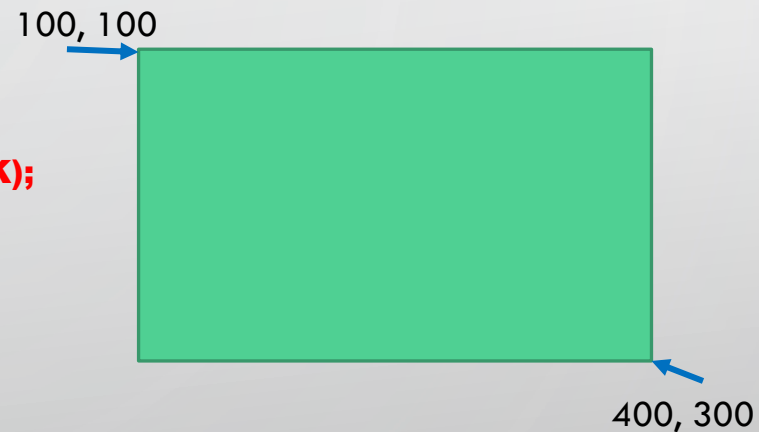
3. 문자열 출력

```
RECT rt = { 100, 100, 400, 300 };
TCHAR *str = TEXT("So keep your eyes on me now"
"무엇을 보든 좋아할 거야 닿을 수 없는 Level"
"나와 대결 원한 널 확신해");
switch (iMessage)
{
    case WM_PAINT:
        hdc = BeginPaint(hWnd, &ps);
        DrawText(hdc, str, -1, &rt, DT_CENTER | DT_WORDBREAK);
        EndPaint(hWnd, &ps);
    return 0;
}
```

값	설명
DT_LEFT	수평 왼쪽 정렬한다.
DT_RIGHT	수평 오른쪽 정렬한다.
DT_CENTER	수평 중앙 정렬한다.
DT_BOTTOM	사각영역의 바닥에 문자열을 출력한다.
DT_VCENTER	사각영역의 수직 중앙에 문자열을 출력한다.
DT_WORDBREAK	사각영역의 오른쪽 끝에서 자동 개행되도록 한다.
DT_SINGLELINE	한 줄로 출력한다.
DT_NOCLIP	사각영역의 경계를 벗어나도 문자열을 자르지 않고 그대로 출력

1. RECT

- 4개의 숫자를 기준으로 시작좌표 x, y에서 뒤에 두 숫자의 크기 만큼을 가지는 사각형 범위를 말한다.



2. DrawText

- 사각형을 그린 뒤 해당 사각형 내부에서 **Text** 출력
- 첫번째 인자 : **DC** 핸들
- 두번째 인자 : 출력 문자열
- 세번째 인자 : 문자열 길이 **NULL** 문자열이면 **-1**
- 네번째 인자 : **Left, right, top, bottom** 값을 가지는 **Rect** 구조체를 받아 해당 사각형 영역 안에서 **Text** 출력 **NULL**을 받을 시 전체영역을 다시 그린다.
- 다섯번째 인자 : **DrawText** 함수 옵션 설정.

4. 점, 선, 도형 출력

4. 점, 선, 도형 출력

case WM_PAINT:

hdc = BeginPaint(hWnd, &ps);

for(int i = 0; i < 100; i++)

SetPixel(hdc, 10 + (i * 3), 10, RGB(255, 0, 0));

MoveToEx(hdc, 50, 50, NULL);

LineTo(hdc, 300, 90);

Rectangle(hdc, 50, 100, 200, 180);

Ellipse(hdc, 220, 100, 400, 200);

EndPaint(hWnd, &ps);

return 0;

- **SetPixel()**

- 해당 위치에 점을 찍는다.
- 첫번째 인자 : **DC** 핸들값
- 두번째, 세번째 인자 : **x, y**좌표값
- 네번째 인자 : 색상값 **RGB(레드, 그린, 블루) - 0 ~ 255**의 넘버

4. 점, 선, 도형 출력

- **MoveToEx()**

- 선의 시작시점을 설정한다.
- 첫번째 인자 : **DC** 핸들 값
- 두번째, 세번째 인자 : **x, y**좌표 값
- 네번째 인자 : 선 이전 좌표 값(통상 **NULL**)

MoveToEx() x = 50, y = 50

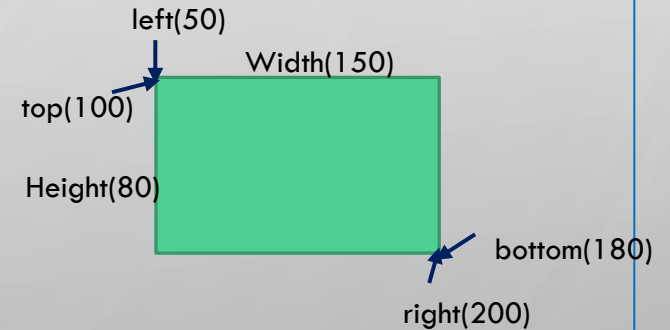
- **LineTo()**

- 선의 종료지점을 설정한다.
- 첫번째 인자 : **DC** 핸들 값
- 두번째, 세번째 인자 : **x, y**좌표 값

LineTo() x = 50, y = 50

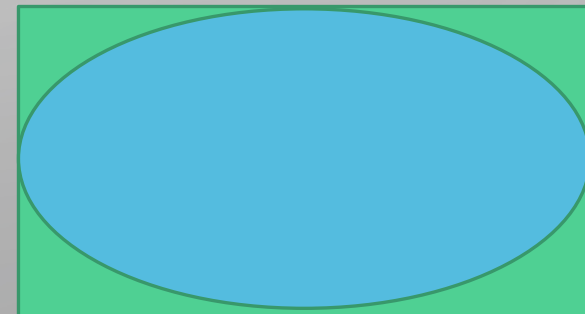
- **Rectangle()**

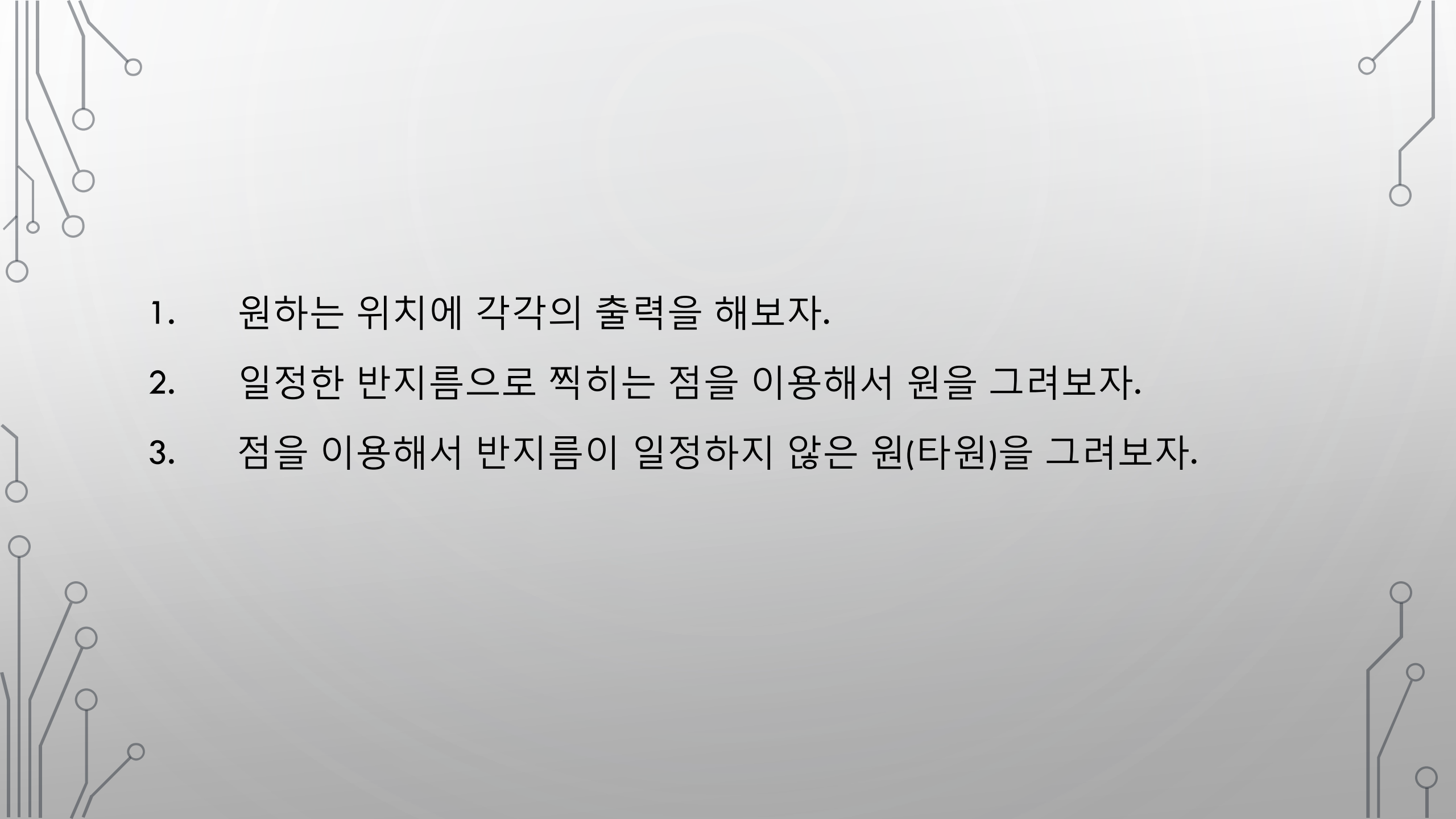
- 사각형을 그리는 함수
- 첫번째 인자 : **DC** 핸들 값
- **2, 3, 4, 5**번째 인자 값 : **left, top, right, bottom**



- **Ellipse()**

- 사각형 범위를 잡은 뒤 원형을 그리는 함수
- 첫번째 인자 : **DC** 핸들 값
- **2, 3, 4, 5**번째 인자 값 : **left, top, right, bottom**



- 
1. 원하는 위치에 각각의 출력을 해보자.
 2. 일정한 반지름으로 찍히는 점을 이용해서 원을 그려보자.
 3. 점을 이용해서 반지름이 일정하지 않은 원(타원)을 그려보자.