



WINAPI

-CAPTER 4-

SOULSEEK



목차

1. Window 생성과 파괴

2. Timer

1. WINDOW 생성과 파괴

1. WINDOW 생성과 파괴

WM_CREATE

- 윈도우가 생성되면서 받게 되는 메시지
- 윈도우가 생성되어서 보이기 전 초기화가 필요하다면 이용할 수 있다

WM_DESTROY

- 윈도우가 종료될 때 받게 되는 메시지
- 릴리즈가 필요하다면 이용할 수 있다

Switch(iMessage)

```
{  
    case WM_CREATE:  
        mem = (TCHAR *)malloc(1048576);  
        return 0;  
    case WM_DESTROY:  
        free(mem);  
        return 0;  
}
```

2. TIMER

2. TIMER

```
HDC hdc;
PAINTSTRUCT ps;
SYSTEMTIME st;
static TCHAR sTime[128];

switch (iMessage)
{
    case WM_CREATE:
        SetTimer(hWnd, 1, 1000, NULL);
        SendMessage(hWnd, WM_TIMER, 1, 0);
        return 0;
    case WM_TIMER:
        GetLocalTime(&st);
        wsprintf(sTime, TEXT("지금 시간은 %d:%d:%d입니다."),
            st.wHour, st.wMinute, st.wSecond);
        InvalidateRect(hWnd, NULL, TRUE);
        return 0;
    case WM_PAINT:
        hdc = BeginPaint(hWnd, &ps);
        TextOut(hdc, 100, 100, sTime, lstrlen(sTime));
        EndPaint(hWnd, &ps);
        return 0;
    case WM_DESTROY:
        KillTimer(hWnd, 1);
        PostQuitMessage(0);
        return 0;
}

return(DefWindowProc(hWnd, iMessage, wParam, lParam));
```

SetTimer(hWnd, nIDEvent, uElapsed, lpTimerFunc)

- 1번째 인수 : 윈도우 핸들
- 2번째 인수 : 타이머의 넘버
- 3번째 인수 : 타이머 메시지를 호출할 간격
- 4번째 인수 : 메시지가 발생했을 때 마다 호출될 콜백함수

KillTimer(hWnd, nIDEvent)

- 1번째 인수 : 윈도우 핸들
- 2번째 인수 : 타이머 넘버
- 타이머를 지정했다면 항상 파괴도 해주어야 한다

WM_TIMER

- **SetTimer**에 지정한 주기마다 발생하는 메시지
- 메시지가 발생하는 간격을 이용해서 주기적인 처리를 명령

SendMessage(hWnd, MSG, wParam, lParam)

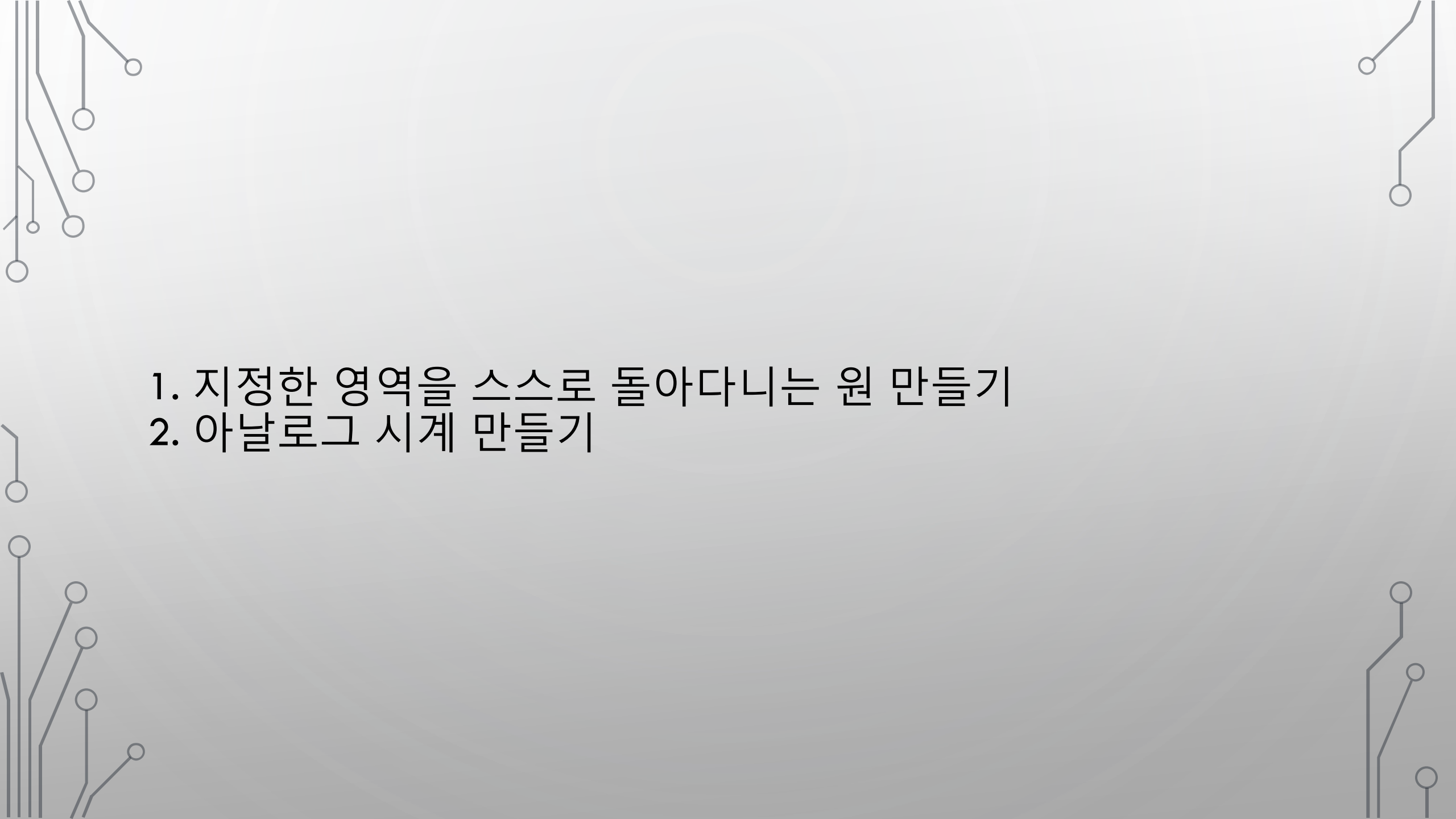
- 조건이나 상황과 상관없이 해당 메시지를 발생시킬 때 사용
- **MSG** 종류에 따라 **wParam**, **lParam**에 들어갈 값이 달라진다.

2. TIMER

```
switch (iMessage)
{
    case WM_CREATE:
        SetTimer(hWnd, 1, 100, TimeProc);
        SendMessage(hWnd, WM_TIMER, 1, 0);
        return 0;
    case WM_PAINT:
        hdc = BeginPaint(hWnd, &ps);
        Ellipse(hdc, x, y, x + 50, y + 50);
        EndPaint(hWnd, &ps);
        return 0;
    case WM_DESTROY:
        KillTimer(hWnd, 1);
        PostQuitMessage(0);
        return 0;
}
return(DefWindowProc(hWnd, iMessage, wParam, lParam));
}

void CALLBACK TimeProc(HWND hWnd, UINT uMsg, UINT idEvent, DWORD dwTime)
{
    x++;
    InvalidateRect(hWnd, NULL, TRUE);
}
```

- **SetTimer**에서 4번째 인수에 CALLBACK함수를 넣어주면 콜백으로 들어오게 된다.
- 연결된 타이머에서 인자 값을 받기때문에 CALLBACK함수의 인자는 신경 쓰지 않아도 된다.
- 타이머메시지가 올때마다 갱신을 해주는 코드가 필요하다면 CALLBACK에서 하는 것이 더 깔끔하다.

- 
- The background features a light gray gradient with faint, abstract circuit-like patterns in the corners. These patterns consist of thin gray lines and small circles, resembling a stylized electronic circuit board.
1. 지정한 영역을 스스로 돌아다니는 원 만들기
 2. 아날로그 시계 만들기