



WINAPI

-CAPTER 3-

SOULSEEK



목차

1. KEYBOARD

2. MOUSE

3. MESSAGEBOX

1. KEYBOARD

1. KEYBOARD

WM_CHAR

- 키보드 입력이 발생했을 경우 보내는 메시지.
- 키보드에서 입력된 문자 키에만 반응한다.
- WM_PAINT에서 따로 그려주는 명령을 수행해야 표현이 가능하다.
- wParam으로 입력 값을 알려준다.

```
TCHAR str[256];
LRESULT CALLBACK WinProc(HWND hWnd, UINT iMessage, WPARAM wParam, LPARAM lParam)
{
    HDC hdc;
    PAINTSTRUCT ps;
    int len;

    switch(iMessage)
    {
        case WM_CHAR:
            len = lstrlen(str);
            str[len] = (TCHAR)wParam;
            str[len + 1] = 0;
            InvalidateRect(hWnd, NULL, TRUE);
            return 0;
        case WM_PAINT:
            hdc = BeginPaint(hWnd, &ps);
            TextOut(hdc, 100, 100, str, lstrlen(str));
            EndPaint(hWnd, &ps);
            return 0;
        case WM_DESTROY:
            PostQuitMessage(0);
            return 0;
    }
    return(DefWindowProc(hWnd, iMessage, wParam, lParam));
}
```

- 강제로 윈도우에 WM_PAINT 메시지를 발생시킨다.
- 1번째 인수 : hWnd
- 2번째 인수 : 다시 그릴 Rect의 크기 NULL일 경우 윈도우 전체
- 3번째 인수 : 지정된 범위에 대한 처리 TRUE일 경우 지우고 다시 그리기, FALSE 경우 지우지 않고 그리기

1. KEYBOARD

WM_KEYDOWN

- 키보드가 입력되면 전달되는 메시지.
- 입력된 모든 키의 정보를 알 수 있다.
- OS에서 설정한 가상 키코드의 값을 알려주며 키보드 종류와 상관없이 범용적인 값을 가진다.
- 영문일 경우 일반 문자열은 대문자와 비교할 수 있다.(ex if(wParam == 'Z'))

가상 키 코드	값(16진수)	키
VK_LEFT	0x25	좌측 커서 키
VK_RIGHT	0x27	우측 커서 키
VK_UP	0x26	위쪽 커서 키
VK_DOWN	0x28	아래쪽 커서 키
VK_SPACE	0x20	Space
VK_ESCAPE	0x1B	Esc
VK_RETURN	0x0D	Enter

1. KEYBOARD

```
HDC hdc;
PAINTSTRUCT ps;
int x = 100;
int y = 100;

switch(iMessage)
{
    case WM_KEYDOWN:
        switch(wParam)
        {
            case VK_LEFT:
                x -= 1;
                break;
            case VK_RIGHT:
                x += 1;
                break;
            case VK_UP:
                y -= 1;
                break;
            case VK_DOWN:
                y += 1;
                break;
        }

        InvalidateRect(hWnd, NULL, TRUE);
        return 0;
    case WM_PAINT:
        hdc = BeginPaint(hWnd, &ps);
        TextOut(hdc, x, y, TEXT("A"), 1);
        EndPaint(hWnd, &ps);
        return 0;
    case WM_DESTROY:
        PostQuitMessage(0);
        return 0;
}
```

키 코드로 메시지를 받아서 좌표 값을 설정 한 후 다시 지웠다가 그리는 것을 이용해 그려진 것이 좌표 이동하는 것처럼 보이게 하였다.

2. MOUSE

2. MOUSE

- Key를 체크하는 별도의 메시지 대신 Mouse의 3개의 버튼의 상태에 따른 체크를 한다.
- IParam으로 Mouse의 현재 좌표를 알 수 있다. HIWORD : Y좌표, LOWORD : X좌표
- WM_MOUSEMOVE로 이동체크 이동할 때마다 메시지가 발생한다.

버튼	누름	놓음	더블클릭
Left	WM_LBUTTONDOWN	WM_LBUTTONUP	WM_LBUTTONDBLCLK
Right	WM_RBUTTONDOWN	WM_RBUTTONUP	WM_RBUTTONDBLCLK
Center	WM_MBUTTONDOWN	WM_MBUTTONUP	WM_MBUTTONDBLCLK

Switch(iMessage)

```
{
    case WM_LBUTTONDOWN:
        x = LOWORD(IParam);
        y = HIWORD(IParam);
        InvalidateRect(hWnd, NULL, TRUE);
    return 0;
    case WM_PAINT:
        hdc = BeginPaint(hWnd, &ps);
        Ellipse(hdc, x, y, x + 100, y + 100);
        EndPaint(hWnd, &ps);
    return 0;
}
```

- 좌클릭을 했을 때 좌표 값을 받아서 그 위치에 원을 그린다.

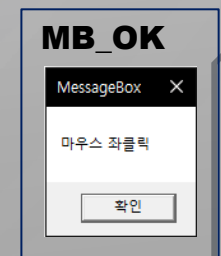
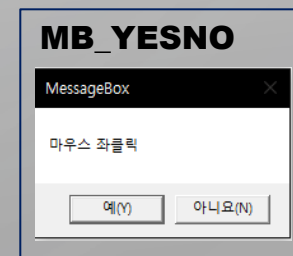
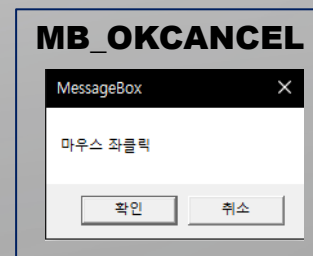
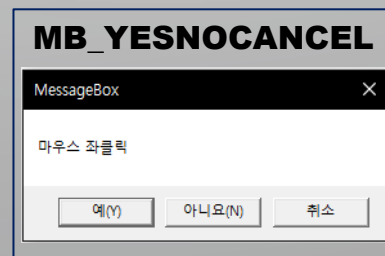
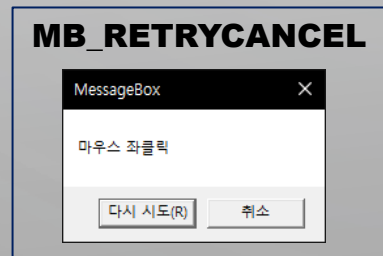
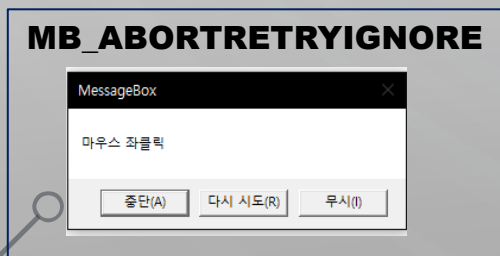
3. MESSAGEBOX

3. MESSAGEBOX

MessageBox(hWnd, lpText, lpCaption, uType)

- 팝업박스로 특정 상황을 알려주고 내부버튼의 상태에 따라 동작을 전달 할 수 있다.
- **1번째 인자 : hWnd**
- **2번째 인자 : 표시 할 문자열.**
- **3번째 인자 : 메시지 박스의 이름.**
- **4번째 인자 : 메시지 박스 버튼 옵션.**

값	설명
MB_ABORTRETRYIGNORE	Abort, Retry, Ignore 세개의 버튼을 보여준다.
MB_OK	OK 버튼을 보여준다.
MB_OKCANCEL	OK, Cancel 버튼을 보여준다.
MB_RETRYCANCEL	Retry, Cancel 버튼을 보여준다.
MB_YESNO	Yes, No 버튼을 보여준다.
MB_YESNOCANCEL	Yes, No, Cancel 버튼을 보여준다.



3. MESSAGEBOX

```
case WM_LBUTTONDOWN:
```

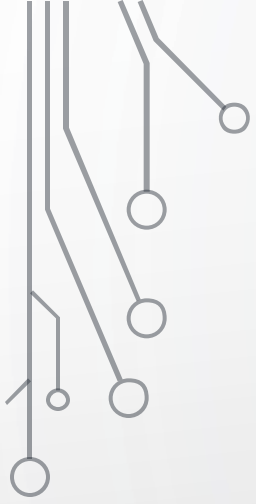


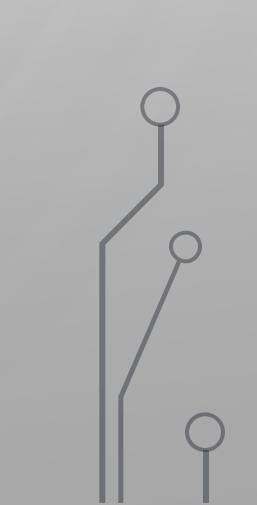
```
    MessageBox(hWnd, TEXT("마우스 좌클릭"), TEXT("MessageBox"), MB_OK);  
    return 0;
```

MessageBox의 버튼 처리

값	설명
IDABORT	Abort 버튼을 눌렀을 때
IDCANCEL	Cancel 버튼을 눌렀을 때
IDIGNORE	Ignore 버튼을 눌렀을 때
IDNO	NO 버튼을 눌렀을 때
IDOK	OK 버튼을 눌렀을 때
IDRETRY	Retry 버튼을 눌렀을 때
IDYES	Yes 버튼을 눌렀을 때

```
case WM_LBUTTONDOWN:
```

```
    if(MessageBox(hWnd, TEXT("마우스 좌클릭"), TEXT("MessageBox"), MB_OK) == IDOK)  
    {  
  
    }  
    else  
    {  
  
    }  
    return 0;
```

- 
- 
- 
- 
1. left, right, up, down 버튼으로 조작하는 원을 만들어보자.
 2. Mouse포인터를 중심으로 따라다니는 원을 만들어 보자.
 3. Mouse포인터를 중심으로 따라다니는 원을 Rect영역을 만들어서 그 안에서 벗어나지 않고 따라다니는 원을 만들어보자.
 4. MessageBox를 이용해서 원을 사각형, 사각형을 원으로 바꾸는 것을 만들어보자.