게임프로그래밍

입력을 위한 윈도우 API

박종승

Dept. of CSE, Incheon Nat. Univ.
jong@inu.ac.kr
http://ecl.inu.ac.kr

입력을 위한 윈도우 API

- 윈도우 메시지 큐를 통한 입력
  - 윈도우 메시지 프로시저에서 윈도우 메시지 큐의 메시지를 순차적으로 접근함
  - 키보드 입력 중에서 문자 입력에만 한정할 경우에는 WM_CHAR 메시지를 사용하면 됨
    - 인자 wParam는 입력된 문자 코드(ASCII 코드)임

```
LRESULT CALLBACK WndProc(HWND hWnd,UINT iMessage,WPARAM wParam,LPARAM lParam)
{
        static char buf[80];
        switch (iMessage) {
        case WM_CHAR:
                wsprintf(buf, "%c 문자가 눌러졌습니다", wParam);
                InvalidateRect(hWnd, NULL, TRUE);
                return 0;
        /* 이하 생략 */
        }
}
```

입력을 위한 윈도우 API'

- 윈도우 메시지 큐를 통한 입력'
  - 특수 키를 포함한 일반적인 키보드 입력의 처리를 위해서는 WM_KEYDOWN 메시지를 사용하면 됨
    - 인자 wParam는 입력된 가상 키 코드 값을 전달해줌

```
LRESULT CALLBACK WndProc(HWND hWnd, UINT iMessage, WPARAM wParam, LPARAM lParam)
{
        static char buf[80];
        switch (iMessage) {
        case WM_KEYDOWN:
                switch (wParam) {
                case VK_LEFT:
                        wsprintf(buf, "%s 키가 눌러졌습니다", "VK_LEFT");
                        InvalidateRect(hWnd, NULL, TRUE);
                        return 0;
                /*이하 생략*/
        }
}
```

입력을 위한 윈도우 API”

- 키보드 가상 키
  - 가상 키(virtual key) 코드는 키보드 하드웨어에 무관하게 키 입력 값을 응용 프로그램에 전달하기 위해 사용되는 코드임
  - 256개의 키들에 대한 가상 키 코드가 정의되어 있음

키	윈도우 가상키 (심볼 VK_*XXX*)	가상키 상수값
영문자	심볼 없음, 'A' ~ 'Z' (ASCII 동일)	0x41 ~ 0x5A
숫자	심볼 없음, '0' ~ '9' (ASCII 동일)	0x30 ~ 0x39
백스페이스 키	VK_BACK (ASCII 동일)	0x08
스페이스바 키	VK_SPACE (ASCII 동일)	0x20
Tab 키	VK_TAB (ASCII 동일)	0x09
Enter 키	VK_RETURN (ASCII 동일)	0x0D
Esc 키	VK_ESCAPE (ASCII 동일)	0x1B
Shift 키	VK_SHIFT	0x10
Ctrl 키	VK_CONTROL	0x11
기능 키 F1~F12	VK_F1 ~ VK_F12	0x70 ~ 0x7B
위쪽, 아래쪽 화살표 키	VK_UP, VK_DOWN	0x26, 0x28
왼쪽, 오른쪽 화살표 키	VK_LEFT, VK_RIGHT	0x25, 0x27
Insert, Delete 키	VK_INSERT, VK_DELETE	0x2D, 0x2E
Home, End 키	VK_HOME, VK_END	0x24, 0x23
Page Up, Page Down 키	VK_PRIOR, VK_NEXT	0x21, 0x22

입력을 위한 윈도우 API"'

- 윈도우 메시지 큐를 통한 마우스 입력
  - 누름 : WM_(L/R/M)BUTTONDOWN
  - 놓음 : WM_(L/R/M)BUTTONUP
  - 더블클릭 : 누름 : WM_(L/R/M)BUTTONDBLCLK
  - 움직임 : WM_MOUSEMOVE

```
LRESULT CALLBACK WndProc(HWND hWnd, UINT iMessage, WPARAM wParam, LPARAM lParam)
{
        static char buf[80];
        switch (iMessage) {
        case WM_LBUTTONDOWN:
                x=LOWORD(lParam);
                y=HIWORD(lParam);
                wsprintf(buf, "마우스 왼쪽 버튼이 눌러졌습니다. 위치=(%d,%d)", x, y);
                InvalidateRect(hWnd, NULL, TRUE);
                return 0;
        case WM_MOUSEMOVE:
                x=LOWORD(lParam);
                y=HIWORD(lParam);
                wsprintf(buf, "마우스가 이동하였습니다. 위치=(%d,%d)", x, y);
        /*이하 생략*/
        }
}
```

Win32의 키보드 입력 함수

02.WinKeyboardExample

- GetAsyncKeyState 함수
  - Win32 API에서의 비동기식 키보드 입력 함수
    - 윈도우 메시지 큐가 아니라 하드웨어 상태값을 읽어오는 함수임
  - 함수의 리턴값 (SHORT 형)
    - 함수 호출 시점에 해당 키가 눌러진 상태이면 최상위 비트(MSB)가 1로 지정됨.
      - 즉 값이 0x0000 또는 0x0001 이라면 호출 시점에 안눌린 상태이고, 0x8000 또는 0x8001 이라면 호출 시점에 눌린 상태임.
    - 직전의 GetAsyncKeyState 함수 호출 이후에 해당 키가 눌러진 적이 있으면 최하위 비트(LSB)가 1로 지정됨.
      - 즉 값이 0x0000 또는 0x8000은 이전에 누른 적이 없는 경우이고, 0x0001과 0x8001은 이전에 누른 적이 있는 경우임.
  - 호출 시점에서의 눌러진 상황에만 관심이 있는 경우 :
    - 예를 들어서 오른쪽 화살표 키가 눌러진 동안에는 계속 오른쪽으로 이동시키는 목적인 경우

```
if (GetAsyncKeyState(VK_RIGHT) & 0x8000) { /* 오른쪽키가 눌러졌음 */ }
```

마우스 커서 위치 얻기

03.WinMouseExample

- 마우스 커서와 관련한 좌표
  - 스크린 좌표 : 스크린 전체 영역에 대한 좌표
    - 원점 (0,0)의 위치는 스크린의 왼쪽 위 모서리
  - 클라이언트 좌표 : 윈도우 창 내부 영역 내에서의 상대적인 좌표
    - 원점 (0,0)의 위치는 원도우 창 내부의 왼쪽 위 모서리
- 마우스 커서와 관련한 윈도우 API 함수
  - SetCursorPos : 스크린 좌표계로 주어진 좌표에 커서를 위치시킴
  - GetCursorPos : 커서의 위치를 스크린 좌표계의 좌표로 받아옴
  - ScreenToClient : 스크린 좌표계의 커서 좌표를 클라이언트 좌표계의 좌표로 변환함
  - ClientToScreen : 클라이언트 좌표계의 좌표를 스크린 좌표계의 좌표로 변환함
- 예제: 마우스 커서의 현재의 스크린 좌표를 얻고 이를 클라이언트 좌표로 변환하는 코드

```
POINT pos;
GetCursorPos(&pos);
ScreenToClient(hWnd, &pos);
```