**• 게임 처리를 위한 WinMain 함수의 메시지 루프.**

**일반 어플리케이션을 위한 WinMain 메시지 루프.**

while (GetMessage(&msg, NULL, 0, 0))

{

if (!TranslateAccelerator(msg.hwnd, hAccelTable, &msg))

{

TranslateMessage(&msg);

DispatchMessage(&msg);

}

}

**게임을 위한 WinMain 메시지 루프.**

while( 1 )

{

if ( PeekMessage(&msg, NULL, 0, 0, PM\_REMOVE) )

{

if (msg.message == WM\_QUIT)

break;

TranslateMessage(&msg);

DispatchMessage(&msg);

}

else

{

**// 게임 처리 함수 호출 !!**

}

}

윈도우 프로그래밍에선 위와 같이 메인 루프가 GetMessage() 함수를 사용하여 현재 처리될 메시지를 검색하며, GetMessage() 함수는 현재 윈도우에 처리할 메시지가 없다면 CPU 를 사용하지 않고 무한 대기상태가 되며, WM\_MOVE, WM\_CLOSE, WM\_PAINT 등… 윈도우 메시지가 발생했을 때 GetMessage() 함수를 빠져 나와서 DispatchMessage() 함수를 통해서 윈도우 프로시저가 호출되어 처리된다.

그러나 게임은 일정 시간마다 매번 화면을 업데이트 시켜줘야만 한다. 이를 위해서 WM\_TIMER 를 사용할 수도 있지만, WM\_TIMER 의 경우 반응시간이 좋지 않은 관계로 사용하지 않으며 위의 소스처럼 while() 루프를 변경해서 사용한다. PeekMessage() 함수를 사용해서 현재 윈도우에 입력된 메시지를검색한다. PeekMessage() 함수는 GetMessage() 함수와는 다르게 메시지가 있는지 없는지 확인을 하고, 현재 메시지큐에 메시지가 있던 없던 PeekMessage() 함수를 빠져 나온다. 리턴값으로 현재 처리할 메시지가 있는지 없는지 확인 할 수 있으며 처리할 윈도우 메시지가 있다면 이를 처리하도록 하고, 처리할 메시지가 없다면 게임처리 함수를 호출하도록 한다.

이렇게 해서 윈도우 메시지를 처리 해주면서 게임처리를 위한 함수도 매번 호출 해줄 수 있게 된다.