

Windows Programming Overview

Seong Jong Choi
chois@mmlab.net

[Multimedia Lab.](#)
Dept. of Electrical and Computer Eng.
University of Seoul
Seoul, Korea

2. Contents

- Windows NT Programming Perspective
- How Windows NT and Your Program Interact
- The Windows NT API
- The Components of a Window
- Some Windows NT Application Basics
- The Windows Function
- A Windows NT Skeleton
- Using a Definition File
- Compiling the Skeleton Program
- Naming Conventions

2.1 Main Objectives

- Introduction Windows Programming
 - It Discusses how a program must **interact** with Windows NT and what **rules** must be followed by every Windows NT application.
 - It develops an **application skeleton** that will be used as a basic for all other Windows NT programs.

2.2 Windows NT Programming Perspective

- **What** the program does, not **how** the user must **interact** with it.
 - Understanding that not every program that runs under Windows NT will necessarily present the user with a Windows style interface.
- Windows NT is graphics oriented , GUI
 - But don` t worry about what type of graphics device.

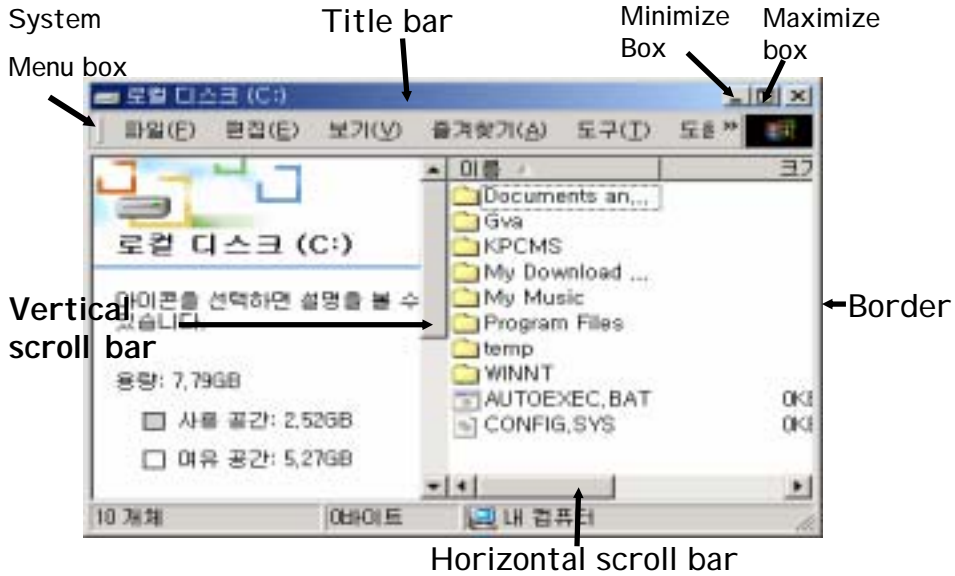
2.2 Windows NT Programming Perspective - cont'd

- The Desktop Model
- The Mouse
 - The Windows NT interface was designed for the mouse , it allows the use of the keyboard.
- Icons and Graphic Images
- Menus and Dialog Boxes

2.3 How Windows NT and Your Program Interact

- Windows NT calls your Program.
 - **Message-Based** interaction
 - A Windows NT Program wait until it is sent a message by windows.
 - Once a message is received, your program is expected to take an appropriate action
 - Interrupt-driven programs

2.5 The Components of a Window



2005-12-02

Windows Skeleton

7

An Example: A Minimal Windows Skeleton 1/2

```
// A minimal Windows NT skeleton

#define STRICT
#include <windows.h>

LRESULT CALLBACK WindowFunc(HWND, UINT, WPARAM, LPARAM);

char szWinName[] = "MyWin"; //name of window class

int WINAPI WinMain(
    HINSTANCE hThisInst,
    HINSTANCE hPrevInst,
    LPSTR lpszArgs, //pointer to command line*/
    int nWinMode //show state of window*/
) //hPrevInst will always be NULL (but 16-bit windows, Non-zero)
{
    HWND hwnd; //Handle to a window.
    MSG msg;
    WNDCLASS wcl;

    //Define a window class
    wcl.hInstance = hThisInst; //handle to this instance
    wcl.lpszClassName = szWinName; //window class name
    wcl.lpfnWndProc = WindowFunc; //window function
    wcl.style = 0; //default style

    wcl.hIcon = LoadIcon(NULL, IDI_APPLICATION); //icon style
    wcl.hCursor = LoadCursor(NULL, IDC_ARROW); //cursor style
    wcl.lpszMenuName = 0; //no menu

    wcl.cbClsExtra = 0; //no extra
    wcl.cbWndExtra = 0; //information needed

    //make the window light gray
    wcl.hbrBackground = (HBRUSH)GetStockObject(LTGRAY_BRUSH);

    //Register the window class
    if (!RegisterClass(&wcl)) return 0;
}
```

2005-12-02

Windows Skeleton

8

An Example: A Minimal Windows Skeleton 2/2

```
//Now that a window class has been registered,
//a window can be created
hwnd = CreateWindow(
    szWinName,           //
    "Windows NT Skeleton", //
    WS_OVERLAPPEDWINDOW, //
    CW_USEDEFAULT,      //
    CW_USEDEFAULT,      //
    CW_USEDEFAULT,      //
    CW_USEDEFAULT,      //
    NULL,               //
    NULL,               //
    hInst,              //
    NULL);              //

/////Display the window
ShowWindow(hwnd, nWinMode);
UpdateWindow(hwnd);

/////Create the message loop
while(GetMessage(&msg, NULL, 0, 0))
{
    TranslateMessage(&msg); //allow use of keyboard
    DispatchMessage(&msg); //return control to windows
}

return msg.wParam;
// end of WinMain()

//This function is called by Window NT and is passed
//message from the message queue.
LRESULT CALLBACK WindowFunc(HWND hwnd, //
    UINT message, //
    WPARAM wParam, //
    LPARAM lParam)
{
    switch(message){
        case WM_DESTROY: //terminate the program
            PostQuitMessage(0); //
            break;
        default:
            return DefWindowProc(hwnd, message, wParam, lParam); //
    }
    return 0;
}
```

2005-12-02

Windows Skeleton

9

WinMain()

- WinMain() is the entry-point function.
- WinMain() contains the following steps:
 1. Define and register the window class
 2. create the main window
 3. enter the message loop

2005-12-02

Windows Skeleton

10

1. Define and register the window class

- Every window must have a window class.
- A window class defines
 - Window style
 - icon
 - cursor
 - Menu
 - window procedure

```
////////Define a window class
WNDCLASS wcl;

wcl.hInstance = hThisInst;           //handle to this instance
wcl.lpszClassName = szWinName;      //window class name
wcl.lpfnWndProc = WindowFunc;       //window function
wcl.style = 0;                       //default style

wcl.hIcon = LoadIcon(NULL, IDI_APPLICATION); //icon style
wcl.hCursor = LoadCursor(NULL, IDC_ARROW); //cursor style
wcl.lpszMenuName = 0;                //no menu

wcl.cbClsExtra = 0;                  //no extra
wcl.cbWndExtra = 0;                  //information needed

//make the window light gray
wcl.hbrBackground = (HBRUSH)GetStockObject(LTGRAY_BRUSH);

////////Register the window class
if(!RegisterClass(&wcl)) return 0;
```

2. Create the Main Window

```
//Now that a window class has been registered,
//a window can be created
hwnd = CreateWindow(    szWinName,           //
                       "Windows NT Skeleton", //
                       WS_OVERLAPPEDWINDOW, //
                       CW_USEDEFAULT,       //           x,y
                       CW_USEDEFAULT,       //           가 ,
                       CW_USEDEFAULT,       //
                       NULL,                //           child window
                       NULL,                //
                       hThisInst,           //
                       NULL);

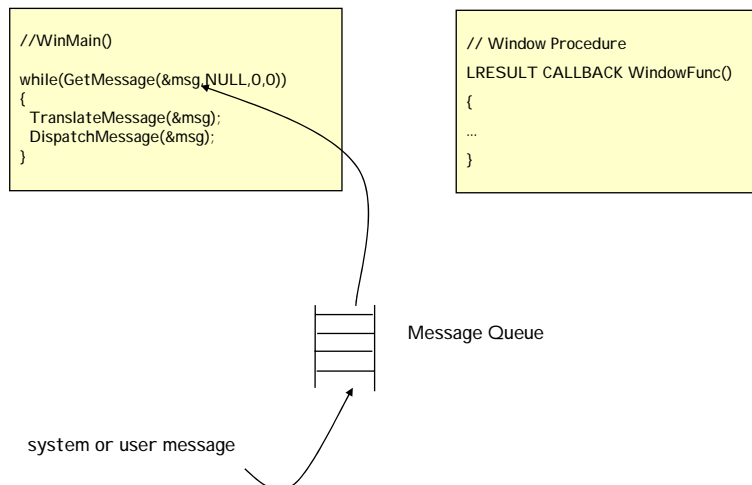
////////display the window
ShowWindow(hwnd, nWinMode);
UpdateWindow(hwnd);
```

- The first parameter is the name of the window class that we registered.
- the system does not display a window until the application calls the ShowWindow function.

3. Enter the Message Loop

```
//////Creat the message loop
while(GetMessage(&msg,NULL,0,0))
{
    TranslateMessage(&msg);           //allow use of keyboard
    DispatchMessage(&msg);           //return control to windows
}
```

- `GetMessage()` read messages from the application message queue
 - If the function retrieves the `WM_QUIT` message, the return value is zero.
 - Else, the return value is nonzero.
- `DispatchMessage()` sends each message to the appropriate window procedure.
- `TranslateMessage()` translates virtual-key message into character messages.



The Window Procedure

```
HRESULT CALLBACK WindowFunc(HWND hwnd, //
    UINT message, //
    WPARAM wParam, //
    LPARAM lParam)
{
    switch(message){
    case WM_DESTROY: //terminate the program
        PostQuitMessage(0); // "0"
        break;
    default:
        return DefWindowProc(hwnd, message, wParam, lParam); // default ..
    }
    return 0;
}
```

- Every window must have a window procedure
- **DefWindowProc()** calls the default window procedure to provide default processing for any window messages that an application does not process.

Windows Defined Data Type

- **BOOL** A Boolean value.
- **BSTR** A 32-bit character pointer.
- **BYTE** An 8-bit integer that is not signed.
- **COLORREF** A 32-bit value used as a color value.
- **DWORD** A 32-bit unsigned integer or the address of a segment and its associated offset.
- **LONG** A 32-bit signed integer.
- **LPARAM** A 32-bit value passed as a parameter to a window procedure or callback function.
- **LPCSTR** A 32-bit pointer to a constant character string.
- **LPSTR** A 32-bit pointer to a character string.
- **LPCTSTR** A 32-bit pointer to a constant character string that is portable for Unicode and DBCS.
- **LPTSTR** A 32-bit pointer to a character string that is portable for Unicode and DBCS.

Windows Defined Data Type

- **LPVOID** A 32-bit pointer to an unspecified type.
- **LRESULT** A 32-bit value returned from a window procedure or callback function.
- **UINT** A 16-bit unsigned integer on Windows versions 3.0 and 3.1; a 32-bit unsigned integer on Win32.
- **WNDPROC** A 32-bit pointer to a window procedure.
- **WORD** A 16-bit unsigned integer.
- **LPARAM** A value passed as a parameter to a window procedure or callback function: 16 bits on Windows versions 3.0 and 3.1; 32 bits on Win32.

2.10 Naming Convention

Prefix	Data Type
b	Boolean (one byte)
c	Character(one byte)
dw	Long unsigned integer
f	16-bit bitfield
h	Handle
l	Long integer
lp	Long pointer

2.10 Naming Convention – cont'd

Prefix	Data Type
n	Short integer
p	Short pointer
pt	Long integer holding screen coordinates
w	Short unsigned integer
sz	Pointer to null-terminated string
lpsz	Long pointer to null-terminated string
rgb	Long integer holding RGB color values