Windows Programming

Lecture 08

Brief History of Win32

- **1983** Windows is announced
- **1984** Work begins on Word for Windows 1.0
- **November 1985:** Windows 1.0 launched0
- **April 1987:** Windows 2.0

Brief History of Win32

- **1988:** Windows/286 + /386 Windows goes up to version 2.04 then splits into Windows/286 and Windows/386, the latter supporting multiple DOS boxes.

- **November 1989:** Winword 1.0 finally ships: four years after the original scheduled date.

- **May 1990 :** Windows 3.0 ships. It operates in 3 modes
  - real (8086 mode as for Windows 2.x)
  - protected ('286)
  - enhanced ('386, with multiple DOS boxes and virtual memory

Brief History of Win32

- **May 1990 :** Windows 3.0 ships It operates in 3
- **Late 1991:** Windows 3.1. Multimedia extensions become part of the standard build
- **Late 1992:** Preliminary Win32 API published as NT beta released Win32 on NT offers pre-emptive multitasking
- **Summer 1993:** NT 3.1 Launches As well as x86 CPUs, NT becomes available for MIPS and Alpha CPUs.

Brief History of Win32

- **Summer 1994:** NT3.5 launches
- **August 1995:** Windows 95 ships
- **September 1995:** Windows NT3.51 Still regarded by many as the most solid version of NT for servers
- **Summer 1996:** NT4.0 MIPS and PowerPC support is dropped: Alpha becomes the only non-x86 CPU to support Win32 .

Brief History of Win32

- **June 1998:** Win98 Ships IE4 is built in
- **September 1998:** Visual Studio 6.0
- **Feb 2000:** Windows 2000 The most significant windows release since Win95. Windows 2000 finally grows up.

WYSIWYG
(what you see is what you get)

Pronounced wizzy-wig, stands for what you see is what you get. A WYSIWYG application is one that enables you to see on the display screen exactly what will appear when the document is printed. This differs, for example, from word processors that are incapable of displaying different fonts and graphics on the display screen even though the formatting codes have been inserted into the file. WYSIWYG is especially popular for desktop publishing.

Windows 3.0 operates in 3 modes

- 8086 mode
- Protected / 286 mode
- Enhanced / 386 mode

- **Supports Multiple dos boxes**
- **virtual memory**

Multitasking

- The ability to execute more than one *task* at the same time, a task being a program.

- In multitasking, only one CPU is involved, but it switches from one program to another so quickly that it gives the appearance of executing all of the programs at the same time.

Multitasking

- There are two basic types of multitasking:
  - *Preemptive*
  - *cooperative*

Multitasking

- In preemptive multitasking, the operating system parcels out CPU *time slices* to each program

- In cooperative multitasking, each program can control the CPU for as long as it needs it. If a program is not using the CPU, however, it can allow another program to use it temporarily.

Windows Components

- Kernel
- GDI
- User

Kernel Main Windows Component

- Heart of the Operating System
- Kernel32.dll

Kernel Responsibilities

- Process Management
- File Management
- Memory Management ...

Kernel

The kernel is the inner core of the Windows CE operating system. The kernel is responsible for scheduling and synchronizing threads, processing exceptions and interrupts, loading applications, and managing virtual memory.

GDI Graphics Device Interface

- GDI32.dll

GDI Responsibilities

- Management of Device Contexts.
- Drawing of primitive shapes.
- Drawing elements e.g. Pens, Brushes, Bitmaps, Palettes, etc.
- .....

GDI
(Graphics Devise interface)

The Windows subsystem responsible for displaying text and images on display devices and printers. The GDI processes graphical function calls from a Windows-based application. It then passes those calls to the appropriate device driver, which generates the output on the display hardware. By acting as a buffer between applications and output devices, the GDI presents a device-independent view of the world for the application while interacting in a device-dependent format with the device

User

Manages all user interface elements (Dialogs, Menus, Text, cursors, Controls, Clipboard, etc)

User Main Windows Component

- User32.dll

User Responsibilities

- Management of user interface elements
- Dialogs, Menus, Text, Cursors, Controls, Clipboard, etc.

Clipboard

- Manage by user32.dll
- Used for copy, paste and etc.
- Temporary storage area

Clipboard

- The Clipboard is a part of Windows that stores material which you have cut or copied from a document. You put material on the Clipboard by selecting it, and then choosing the **Cut** or **Copy** commands (the Cut command removes the selection from the original; the Copy command leaves the selection intact in the original document).

- To **Paste** the Clipboard material in a new place, position the cursor in the appropriate place (by clicking the left mouse button once where you want the material to appear) and select the Paste option from the Edit menu or from the Toolbar.

Clipboard

- Material stays on the Clipboard until it is replaced with something else. You can paste the material as many times as you like. But once you cut or copy something new, the old material on the Clipboard will be lost.

- Once you know how to use Clipboard, then the standard procedure is to open the document from which you wish to remove (or copy) something. Use the **Copy** or **Cut** commands. Then move to the second application, position the cursor where you want the transferred material to appear, and select the **Paste** command.

- As mentioned before, it is usually more prudent to use the **Copy** command. Then, if anything goes wrong, the material is still in the original document. If you use the **Cut** command and you lose the material on the Clipboard, you will have to redo the work.

Handles in windows

- Operating system maintains a list of opened files.
- DOS keeps record of them with a number.
- Windows keeps record of them with handles.

HANDLE , HWND , HINSTANCE
are all windows typedefs

HANDLE

- **General handle for anything**
- **For example,**
  - **handle of opened windows,**
  - **Handle of brushes**
  - **Handle of running processes/programs**

HWND

- **Handle to a Window**
- **Returns HWND type constant**

HINSTANCE

- **Handle to an Instance of an Application**

WIn32 typedefs

HANDLE	A number of Operating system's maintained list of elements.
HWND	Handle to a Window
HINSTANCE	Handle to an Instance of an Application

Our first Win32 programm

```
#include <windows.h>

int WINAPI WinMain( HINSTANCE hInstance,
                    HINSTANCE hPrevInstance,
                    LPSTR     lpCmdLine,
                    int       nCmdShow)
{
      MessageBox(NULL, "This is our first Windows
      Programming Application.", "Virtual
      University", MB_OK);

      return 0;
}
```

Arguments to WinMain()

```
int WINAPI WinMain(

HINSTANCE hInstance,
        // handle to current instance
HINSTANCE hPrevInstance,
        // handle to previous instance
LPSTR lpCmdLine,
        // pointer to command tail
int nCmdShow
        // show state
);
```

The MessageBox() API function

```
int MessageBox(

HWND hWnd,
        // handle to owner window
LPCTSTR lpText,
        // text in message box
LPCTSTR lpCaption,
        // message box title
UINT uType
            // message box style
);
```

The MessageBox() styles

WINUSER.H included in Windows.h

- `#define MB_OK 0x00000000L`
- `#define MB_OKCANCEL 0x00000001L`
- `#define MB_ABORTRETRYIGNORE 0x00000002L`
- `#define MB_YESNOCANCEL 0x00000003L`
- `#define MB_YESNO 0x00000004L`
- `#define MB_RETRYCANCEL 0x00000005L`

Example 2: Command Line arguments

```
#include <windows.h>

int WINAPI WinMain(HINSTANCE hInstance,
                   HINSTANCE hPrevInstance,
                   LPSTR     lpCmdLine,
                   int       nCmdShow)
{
    MessageBox(NULL, GetCommandLine(), "Virtual
    University", MB_OK);

    return 0;
}
```

The MessageBox() returns

WINUSER.H included in Windows.h

- `#define IDOK 1`
- `#define IDCANCEL 2`
- `#define IDABORT 3`
- `#define IDRETRY 4`
- `#define IDIGNORE 5`
- `#define IDYES 6`
- `#define IDNO 7`

WIn32 Data types

- **BOOL** A Boolean value.
- **BYTE** An 8-bit integer that is not signed.
- **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.

WIn32 Data types

- **LPTSTR** A 32-bit pointer to a character string that is portable for Unicode and DBCS.
- **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.
  WPARAM 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.