Programming For Engineers

WIN32GUI Lessons

by
Wan Azhar Wan Yusoff[1], Ahmad Fakhri Ab. Nasir[2]
Faculty of Manufacturing Engineering
wazhar@ump.edu.my[1], afakhri@ump.edu.my[2]

0.0 Chapter's Information

- Expected Outcomes
  - To use Windows GUI programming.

- Contents

  1.0 Introduction

  2.0 Windows Definitions

  3.0 Basic Windows Program

1.0 Introduction

- We should know basic Windows Programming technology especially the usage of:
  a. Use typedef struct. Windows use typedef struct instead of struct. Therefore, the data type has no "struct" in front of the structure data type. Instead it uses all capital letters to represent the struct. Example:

```
typedef struct
{
    int age;
    double cpa;
} STUDENT;

STUDENT ali;
```

1.0 Introduction

b. Use typedef for new data type name. For example, instead of using unsigned int, it uses #define unsigned int UINT. Then when we need unsigned int, we use UINT.

```
typedef unsigned int UNIT;
UINT a;
```

c. Use macro #define to create constant name. For example, the pie number:

```
#define PI 3.14
double area, radius;
area = 2*Pi*radius;
```

1.0 Introduction

d. Use data type casting. Example casting from double to integer:

```
double a = 3.14;
int b;
b = (int) a;
```

2.0 Windows Definitions

- Many definitions are stated in windef.h, winNT.h and BaseTsd.h. On the right is an example from windef.h file.

```
typedef WINBOOL BOOL;
#else
#define BOOL WINBOOL
#endif
typedef unsigned char BYTE;
#endif /* ndef XFree86Server */
typedef BOOL *PBOOL,*LPBOOL;
typedef unsigned short WORD;
typedef float FLOAT;
typedef FLOAT *PFLOAT;
typedef BYTE *PBYTE,*LPBYTE;
typedef int *PINT,*LPINT;
typedef WORD *PWORD,*LPWORD;
typedef long *LPLONG;
typedef DWORD *PDWORD,*LPDWORD;
typedef CONST void *PCVOID,*LPCVOID;
typedef int INT;
typedef unsigned int UINT,*PUINT,*LPUINT;

#include <winnt.h>

typedef UINT_PTR WPARAM;
typedef LONG_PTR LPARAM;
typedef LONG_PTR LRESULT;
#ifndef _HRESULT_DEFINED
typedef LONG HRESULT;
#define _HRESULT_DEFINED
#endif
#ifndef XFree86Server
typedef WORD ATOM;
#endif /* XFree86Server */
```

3.0 Basic Windows Program

- We will develop basic window program line by line. Then, we will inspect what each line means. Type the one page windows frame program using console. We will use printf() if We want to look at window variable. Make sure to include stdio.h as usual.
- Present each windows basic typedef and typedef struct. Look at Microsoft Help for WINDOW Data Type page.
  a. HINSTANCE - instance of object handle
  b. LPSTR - long pointer string
  c. WPARAM - word pararneter. This is a 2-byte integer.
  d. LPARAM - long parameter. This is a 4-byte integer.
  e. UNIT - unsigned int
  f. MSG - structure data type to store user response.
  g. WNDCLASSEX - structure data type to create windows frame

3.0 Basic Windows Program

h. LRESULT - long return value of a function. This is a 4-byte integer.

i. CALLBACK - this is macro #define to _stdcall()

j. WINAPI - this is macro #define to _stdcall()

- Present windows constants.
  a. WM_CLOSE - number to indicate closing window
  b. WM_DESTROY - number to indicate closing window
  c. NULL - zero
  d. lDI_APPLICATION - integer number identification for image for program application.
  e. IDC_ARROW - integer number identification for control object arrow.

3.0 Basic Windows Program

f. COLOR_WINDOW - integer number to represent color.
g. nCmdShow - integer number to represent window type.

- Functions to s.etup Windows Frame
  a. LoadIcon() - function to load program icon.
  b. LoadCurso() - function to load program cursor.
  c. RegisterClassEx(WNDCLASSEX);
  d. CreateWindowEx()
  e. ShowWindow()
  f. UpdateWindow()
  g. GetMessage()
  h. TranslateMessage()

3.0 Basic Windows Program

i. DispatchMessage()
j. DestroyWindow()
k. PostQuitMessage()
l. DefWindowProc()

- In WinMain() function, we need 3 variables:
  a. WNDCLASSEX is a struct data type for creating window frame.
  b. HWND is int** is a pointer to integer number for identification. Handle means the identity of object. Every object must have number (handle) so that windows can recognize each object.
  c. MSG is the struct data type for user message.

3.0 Basic Windows Program

```
#LRESULT CALLBACK myProgram(HWND hwnd, UINT msg, WPARAM wparam, LPARAM lparam);
// (1) Create window main function
int WINAPI WinMain(HINSTANCE hInstance,     //unsigned int for our program number
                   HINSTANCE hPrevInstance, //unsigned int for old Windows OS. Now always zero.
                   LPSTR lpCmdline,         //long pointer string
                   int nCmdShow)            //number for default window style
{
    // (2) Create window variables
    WNDCLASSEX myClass;     //WNDCLASSEX is struct for window frame.
    HWND myWindowHandle;    //Our window frame number.
    MSG userMessage;        //Message number from user is a struct data type
    // (3) Design window basic properties
    myClass.cbSize = sizeof(WNDCLASSEX);                    //memory size of window frame.
    myClass.style = 0; myClass.lpfnWndProc = myProgram;     //The procedure for action based on user response
    myClass.cbClsExtra = 0;                                 //Extra bytes at the end of myClass
    myClass.cbWndExtra = 0; myClass.hInstance = hInstance;  //window handle same as winMain() handle
    myClass.hIcon = LoadIcon(NULL, IDI_APPLICATION);        //IDI is ID image
    myClass.hIconSm = LoadIcon(NULL, IDI_APPLICATION);
    myClass.hCursor = LoadCursor(NULL, IDC_ARROW);          //IDC is ID control
    myClass.hbrBackground = (HBRUSH)(COLOR_WINDOW+1);
    myClass.lpszMenuName = NULL;                            //Menu object name
    myClass.lpszClassName = "wawyWindowClass";              //Our window name
    // (4) Register windows
    RegisterClassEx(&myClass);
    // (5) Create window - different from design window form
    myWindowHandle = CreateWindowEx(WS_EX_CLIENTEDGE,
                                    "wawyWindowClass",
                                    "My Basic Window",
                                    WS_OVERLAPPEDWINDOW,CW_USEDEFAULT,CW_USEDEFAULT,
```

3.0 Basic Windows Program

```
                                    300,120, NULL, NULL, hInstance, NULL);
    // (6) Display and update window
    ShowWindow(myWindowHandle, nCmdShow);
    UpdateWindow(myWindowHandle);
    // (7) Get user response from event-driven Loop
    while(GetMessage(&userMessage, NULL, 0, 0) > 0)
    {
        TranslateMessage(&userMessage);     // (8) Translate OS message to program message
        DispatchMessage(&userMessage);      // (9) Send program message to processing function
    }
    return 0;
}
// (10) Receive message from DispatchMessage()
LRESULT CALLBACK myProgram(HWND hwnd, UINT msg, WPARAM wparam, LPARAM lparam)
{
    switch(msg)         // Go to appropriate function based on user message number
    {
        case WM_CLOSE:
            DestroyWindow(hwnd);        //When user click close window - WinMain0 exit while loop
            break;
        case WM_DESTROY:
            PostQuitMessage(0);         // when user want to exit winMain0 while loop
            break;
        default:                        //Back to WinMain0 function
            return DefWindowProc(hwnd, msg, wparam, lparam);
    }
    return 0;
}
```