

Working with Win32

CS193W
Dr. Patrick Young

Announcements

Readings

- Petzold
- Online Documentation
 - PlatformSDK : User Interface Services

Practice Assignments

- Practice 14: Working with Win32

About Win32

- Object Oriented
 - centered around Window Class
 - Written on top of C (not C++)

Outline of Lecture

- Window Class
- Main Program
- Examples and Details

Object-Oriented Programming

- objects combine
 - procedural information (functions)
 - data

MFC Programming

- based on document/views
- data stored in
 - document
 - actual document data
 - view
 - viewing information (e.g., viewing settings, scroll position)

MFC Programming

- procedural information
 - how to draw
 - how to respond to mouse clicks
 - how to respond to menu items
- mostly on view (or document)
- usually in response to window message*

* even OnDraw is often caused by a WM_PAINT message

Window Classes (WndClass)

- each window based on a WndClass
 - defines behavior of window
 - may define data
- Win32 includes pre-defined WndClasses
 - controls pre-defined classes
 - dialog boxes, desktop

WndProc

- each WndClass includes a Window Procedure
 - defines behavior of windows based on class
 - responsible for handling messages
- no other procedural information in window class

WndProc

```
LRESULT CALLBACK WndProc(HWND hWnd, UINT message,  
                          WPARAM wParam, LPARAM lParam);
```

- handle to window receiving message
- message ID
 - WM_CREATE, WM_LBUTTONDOWN
- wParam
- lParam

WndProc Essentially Switch

```
LRESULT CALLBACK WndProc(HWND hWnd, UINT message,  
                          WPARAM wParam, LPARAM lParam)
```

```
{...  
  switch (message)  
  {  
    case WM_COMMAND: ...  
    case WM_CREATE: ...  
    case WM_PAINT: ...  
    case WM_LBUTTONDOWN: ...  
    case WM_DESTROY: ...  
  }  
}
```

Data Supported

- data can be allocated for
 - shared by all windows in class
 - provided individually for each window
- example:
 - need to store x and y position for squares

Data Supported

- access functions very primitive
 - data treated as sequential bytes in memory
 - can access individual 16 or 32-bit chunks via offsets

Accessing Data

- ```
LONG SetWindowLong(HWND hWnd,
 int nIndex, LONG dwNewLong);
```
- handle to window to modify
  - offset in bytes to desired long
  - new value
- similarly
    - GetWindowLong, SetWindowWord, GetWindowWord
    - SetClassLong, GetClassLong, SetClassWord, GetClassWord

## Other WndClass Info

- Window Class also defines
  - default cursor
  - icons
  - background brush
  - menu
  - class styles
    - CS\_HREDRAW, CS\_VREDRAW, CS\_DBLCLKS

## WndClass Registration

- need to register WndClass before use
  - create WNDCLASSEX structure
  - call RegisterClassEx
- refer to WndClass by string
  - provide lpzClassName in WNDCLASSEX
  - use lpzClassName when creating windows

## WNDCLASSEX

```
typedef struct _WNDCLASSEX {
 UINT cbSize; LPCTSTR lpzMenuName;
 UINT style; LPCTSTR lpzClassName;
 WNDPROC lpfnWndProc; HICON hIconSm;
 int cbClsExtra; ;
 int cbWndExtra; ;
 HANDLE hInstance; ;
 HICON hIcon; ;
 HCURSOR hCursor; ;
 HBRUSH hbrBackground; ;
} WNDCLASSEX;
```

## WNDCLASSEX Notes

- `cbClsExtra` and `cbWndExtra`
  - define amount space to provide for class and window data (for `GetWindowLong`, `SetWindowLong`, etc.)
- `hInstance`
  - handle to “application instance”
  - provided when program executes

## WinMain

- main procedure of Win32 program
  - equivalent of `main` in C programs
  - register window class(es)
  - create and display window
  - enter message loop

## WinMain

```
int WINAPI WinMain(HINSTANCE hInstance,
HINSTANCE hPrevInstance,
LPSTR lpCmdLine, int nCmdShow);
```

- `hInstance` identifies instance of application
  - multiple copies of application can run concurrently
- `hPrevInstance` no longer used
- `lpCmdLine` access to command line if executed from prompt
- `nCmdShow` identifies initial status of window
  - `SW_SHOWNORMAL`, `SW_SHOWMINIMIZED`

## Creating a Window

```
HWND CreateWindow(
LPCTSTR lpClassName,
LPCTSTR lpWindowName,
DWORD dwStyle,
int x, int y,
int nWidth, int nHeight,
HWND hWndParent,
HMENU hMenu,
HANDLE hInstance,
LPVOID lpParam
);
```

## CreateWindow Notes

- `lpWindowName`
  - caption of window
  - used for text of buttons
- `hMenu`
  - `NULL` to use `WndClass`' menu
- `lpParam`
  - user defined data accessible from `lParam` of `WM_CREATE` message

## Create and Display Main Wnd

```
WNDCLASSEX wndClass;
// initialize wndClass structure here ...

RegisterClassEx(&wndClass);

hWnd = CreateWindow(szWindowClass, szTitle,
WS_OVERLAPPEDWINDOW, CW_USEDEFAULT,
0, CW_USEDEFAULT, 0, NULL, NULL, hInstance, NULL);

ShowWindow(hWnd, nCmdShow);
UpdateWindow(hWnd);
```

## Message Loop

- after main window created/updated
- enter “infinite” message loop
  - retrieve messages from queue
  - process messages
  - dispatch to appropriate WndProc

## Message Loop

```
MSG msg;
while (GetMessage (&msg, NULL, 0, 0) > 0) {
 TranslateMessage (&msg);
 DispatchMessage (&msg);
}
```

## Message Loop

```
BOOL GetMessage(LPMSG lpMsg, HWND hWnd,
 UINT wMsgFilterMin, UINT wMsgFilterMax);
```

```
GetMessage (&msg, NULL, 0, 0);
```

- message structure to store message info
- HWND if only want messages to one window
- MsgFilter's limit actual message retrieved
- returns 0 on WM\_QUIT, > 0 for normal operation, -1 for error

## Message Loop

```
TranslateMessage (&msg);
DispatchMessage (&msg);
```

- translate message used to convert keydowns to chars
  - MW\_KEYDOWN and WM\_CHAR (ASCII)
- generates WM\_CHAR message

\* does not change message, generates a new one

## Message Loop

```
DispatchMessage (&msg);
```

- calls appropriate WndProc for window corresponding to msg

## Accelerators

```
while (GetMessage(&msg, NULL, 0, 0))
{
 if (!TranslateAccelerator(msg.hwnd, hAccelTable, &msg))
 {
 TranslateMessage(&msg);
 DispatchMessage(&msg);
 }
}
```

## TranslateAccelerator

- grabs message before it's dispatched
- translates to appropriate WM\_COMMAND message

## Overview

- now know basics of Win32
- go over details and some examples
  - Squares Example

## Squares WndClass

```
WNDCLASSEX wcxex;
...
wcxex.cbClsExtra = 0;
wcxex.cbWndExtra = 8;
...
return RegisterClassEx(&wcxex);
```

## Squares WndProc

```
LRESULT CALLBACK WndProc(HWND hWnd, UINT message,
 WPARAM wParam, LPARAM lParam)
{...
 switch (message)
 {
 case WM_COMMAND: ...
 case WM_CREATE: ...
 case WM_PAINT: ...
 case WM_LBUTTONDOWN: ...
 case WM_DESTROY: ...
 }
}
```

## Squares WndProc

```
case WM_COMMAND:
 wmlId = LOWORD(wParam);
 wmEvent = HIWORD(wParam);
 switch (wmlId) {
 case IDM_ABOUT:
 ...
 break;
 case IDM_EXIT:
 DestroyWindow(hWnd);
 break;
 }
 break;
```

## Squares WndProc

```
case WM_CREATE:
 SetWindowLong(hWnd,0,40);
 SetWindowLong(hWnd,4,40);
 break;
```

## Squares WndProc

```
case WM_LBUTTONDOWN:
 xPos = LOWORD(lParam);
 yPos = HIWORD(lParam);
 SetWindowLong(hWnd, 0, xPos);
 SetWindowLong(hWnd, 4, yPos);
 InvalidateRect(hWnd, NULL, TRUE);
 break;
```

## GDI Win32 Calls

- very similar to MFC calls

```
pDC->LineTo(100,100);

LineTo(hdc, 100, 100);
```

## Squares WndProc

```
case WM_PAINT:
 hdc = BeginPaint(hWnd, &ps);
 xPos = GetWindowLong(hWnd, 0);
 yPos = GetWindowLong(hWnd, 4);
 Rectangle(hdc, xPos, yPos, xPos+20, yPos+20);
 EndPaint(hWnd, &ps);
 break;
```

## Squares WndProc

```
case WM_DESTROY:
 PostQuitMessage(0);
 break;

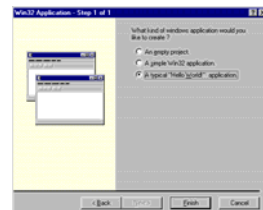
default:
 return DefWindowProc(hWnd,
 message, wParam, lParam);
```

## Windows Subclassing

- can subclass either
  - window classes
  - individual windows
- provide a new WndProc
  - messages go to new WndProc
  - remember old WndProc address
  - pass some messages to old WndProc
    - CallWindowProc

## Visual C++ Support

- can create Win32 program outlines



## Visual C++ Support

- choose “A Typical Hello World App”
  - includes registering window class
  - creating and showing window
  - message loop
  - WndProc filled in
- with “A Simple Win32 App”
  - empty WinMain

## Things to Notice on Squares

- Win32 Object-Oriented, but ...
  - limited ability to define behavior and state
  - Requires more to remember and type more
- No Serialization
  - need to actually work with file
  - need to add dialogs
- No Concept of Documents or Views
- No Support for Multiple Windows
- No CScrollView or other Support Classes
- No Collection Classes