Windows Programming

Lecture 12

An Overview of Previous Lecture

- Hierarchy of windows
- Threads
- Parent and Child windows
- Owner and Owned windows
- Window styles
- Parts of a typical application window
- Common messages

An Overview of Previous Lecture

- Handling `WM_SYSCOMMAND` message to swap the function of minimize and maximize buttons.
- `DefWindowProc()` sending `WM_CLOSE` message after processing the `WM_SYSCOMMAND` message.

Pre-defined Window Classes

- Pre-registered or System Window Classes
- Window procedures of System Window Classes are pre-written

System Window Classes

- Till now, we registered a window class before creating a window. A number of window classes are pre-registered / pre-coded in Windows, and their window procedures also are pre-written
- No call to **`RegisterClass()`** before creating such a window

System Window Classes

There are two types of System Window Classes

1. Those which can be used by the user processes.
2. Those which can only be used by the system.

Using ATOM returned by `RegisterClass()`

Instead of the class-name string, we can use the ATOM value returned by

`RegisterClass()`
in
`CreateWindow()`

- User defined window classes are automatically unregistered when application using those classes terminates.

- System classes cannot be unregistered by user processes.

System/Pre-defined Window Classes

(to be used by user processes only)

- **`Button`**
  - **`pushbutton`**
  - **`autocheckbox`**
  - **`radio etc.`**
- **`ComboBox`**
- **`Edit`**
- **`ListBox`**
- **`ScrollBar`**
- **`Static`**

System/Pre-defined Window Classes

(to be used by system only)

ATOM values

#32768	The class for a menu
#32769	The class for the desktop window
#32770	The class for a dialog box
#32771	The class for the task switch window

Styles of system Window class BUTTON

```
BS_AUTOCHECKBOX
BS_AUTORADIOBUTTON
BS_PUSHBUTTON
BS_RADIOBUTTON
BS_DEFPUSHBUTTON
```

… … …

Example: System Window Classes

```
hWnd = CreateWindow(
            "BUTTON",
            "Virtual University",
            BS_RADIOBUTTON | WS_VISIBLE |
            WS_OVERLAPPEDWINDOW |   WS_CAPTION,
            50, 50, 200, 100, NULL, NULL, hInstance,
            NULL);

while(GetMessage(&msg, NULL, 0, 0) > 0)
{
    if(msg.message == WM_LBUTTONUP)
    {
          DestroyWindow(hWnd);
          PostQuitMessage(0);
    }
    DispatchMessage(&msg);
}
```

Is it possible to get information about a window after it is created

GetWindowLong( )

The **GetWindowLong()** function retrieves information about the specified window.

GetWindowLong( )

```
LONG GetWindowLong(
   HWND  hWnd,          // handle to window
   int   nIndex         // offset of value to retrieve
);
```

GWL_WNDPROC
GWL_HINSTANCE
GWL_STYLE

SetWindowLong( )

The **SetWindowLong()** function changes an attribute of the specified window.

SetWindowLong( )

```
LONG SetWindowLong(
   HWND hWnd,              // handle to window
   int nIndex,             // offset of value to set
   LONG dwNewLong          // new value
);
```

GWL_WNDPROC
GWL_HINSTANCE
GWL_STYLE

Sub-classing

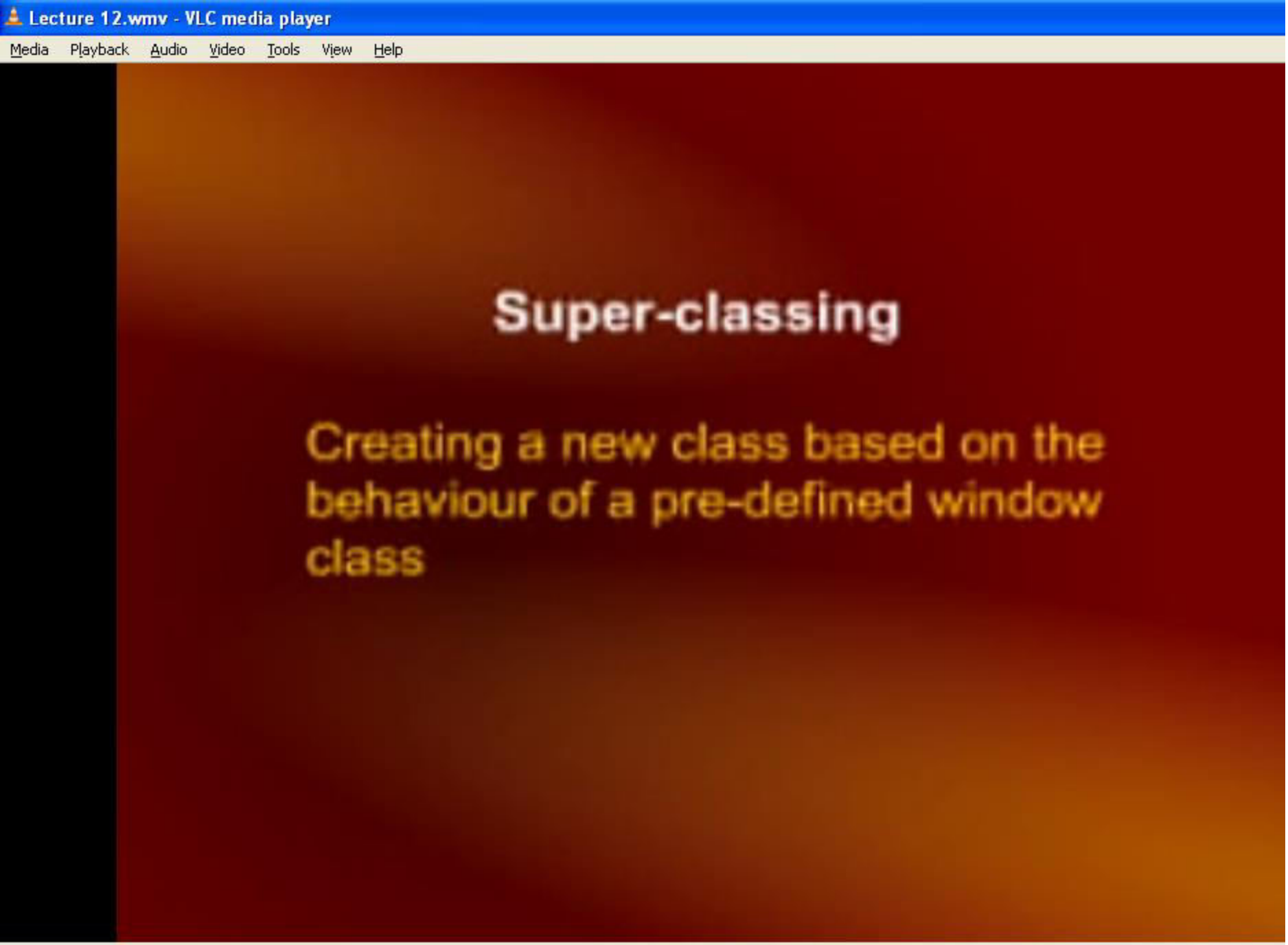
Substituting the window procedure for a window class with another.

Sub-classing allows you to change the behavior of an existing window, typically a control, by inserting a message map to intercept the window's messages. For example, suppose you have a dialog box with an edit control that you want to accept only non-numeric characters. You could do this by intercepting **WM_CHAR** messages destined for the edit control and discarding any messages indicating that a numeric character has been entered.

Sub-classing

Sub-classing is of two types:

- Just for one window
- For all windows in the application created after substitution of the window procedure


Lecture 12.wmv - VLC media player
Media Playback Audio Video Tools View Help
Super-classing
Creating a new class based on the behaviour of a pre-defined window class

Super-classing

Super-classing defines a class that adds new functionality to a predefined window class, such as the button or list box controls.

Information About a Window Class

- **`GetClassLong()`**
- **`SetClassLong()`**

GetClassLong()

The **GetClassLong()** function retrieves the specified 32-bit (long) value from the **WNDCLASS** structure associated with the specified window.

GetClassLong()

```
LONG GetClassLong(
   HWND hWnd,        // handle to window
   int nIndex        // offset of value to retrieve
);
```

GCL_WNDPROC
GCL_STYLE

`SetClassLong()`

returns the previous value of the specified 32-bit integer

SetClassLong()

The **SetClassLong()** function replaces the specified 32-bit (long) value at the specified offset into the extra class memory or the **WNDCLASS** structure for the class to which the specified window belongs.

SetClassLong()

```
LONG SetClassLong(
    HWND hWnd,          // handle to window
    int nIndex,         // offset of value to set
    LONG dwNewLong      // new value
);
```

Difference Between `SetWindowLong()` and `SetClassLong()`

- In `SetWindowLong()`, behavior of a single window is modified.
- In `SetClassLong()`, behavior of the window class is modified

Type 1-Subclassing

Sub-classing

```
WNDPROC oldWindowProc;

   hWnd = CreateWindow("BUTTON", "Virtual University",
                       BS_AUTOCHECKBOX | WS_VISIBLE |
                       WS_OVERLAPPEDWINDOW,
                       50, 50, 200, 100,
                       NULL, NULL, hInstance, NULL);

   oldWindowProc = (WNDPROC)SetWindowLong (hWnd,
                                           GWL_WNDPROC,
                                           (LONG)myWindowProc);

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

   return msg.wParam;
```

Sub-classing

```
LRESULT CALLBACK myWindowProc (HWND hWnd, UINT message, WPARAM
   wParam, LPARAM lParam)
{
   switch (message)
   {
        case WM_LBUTTONDOWN:
               MessageBox (hWnd, "Left mouse button pressed.",
               "Message", MB_OK);
               DestroyWindow(hWnd);
               break;

        case WM_DESTROY:
               PostQuitMessage(0);
               break;

        default:
               return CallWindowProc(oldWindowProc, hWnd,
               message, wParam, lParam);
   }
   return 0;
}
```

Type 2-Subclassing

Example: Sub-classing

```
HWND hWnd; WNDPROC oldWindowProc;

hWnd = CreateWindow("BUTTON",...
... ... ... ... ...
oldWindowProc =
     (WNDPROC)SetClassLong(hWnd,
      GCL_WNDPROC, (LONG)myWindowProc);

DestroyWindow(hWnd);

MessageBox(NULL, "Virtual University",
                 "Message", MB_OK);
```

Example: Sub-classing

```
HWND hWnd; WNDPROC oldWindowProc;

hWnd = CreateWindow("BUTTON",...
... ... ... ... ...
oldWindowProc =
    (WNDPROC)SetClassLong(hWnd,
     GCL_WNDPROC, (LONG)myWindowProc);

DestroyWindow(hWnd);

MessageBox(NULL, "Virtual University",
                 "Message", MB_OK);
```

Sub-classing: The Window Procedure

```
LRESULT CALLBACK myWindowProc (HWND hWnd, UINT message, WPARAM
   wParam, LPARAM lParam)
{
   switch (message)
   {
        case WM_LBUTTONDOWN:
                MessageBeep(0XFFFFFFFF);
                break;

        default:
                return CallWindowProc(oldWindowProc, hWnd,
                message, wParam, lParam);
   }
   return 0;
}
```