

Parallel and Concurrent Programming

Lecture 6
Windows Messaging & Events

Event-driven Programming

- A programming paradigm in which the flow of the program is determined by sensor outputs or user actions (mouse clicks, key presses) or messages from other programs.

Batch Programming

- In batch programming, the flow is determined by the programmer.
- Batch programming is the style taught in beginning programming classes.
- The more complex event-driven programming is the standard architecture of modern interactive programs.

Batch Program Version

```
read a number (from the keyboard) and store it in variable A[0]
read a number (from the keyboard) and store it in variable A[1]
print A[0]+A[1]
```

Event-driven Version

```
set counter K to 0
repeat {
  if a number has been entered (from keyboard) {
    store in A[K] and increment K
    if K equals 2 print A[0]+A[1]
    reset K to 0
  }
}
```

Event Handlers

- Because the code for checking for events and the main loop do not depend on the application, many programming frameworks take care of their implementation and expect the user to provide **only the code for the event handlers**.
- In this next example there may be a call to event handler called **OnKeyEnter()** that includes an argument with a string of characters, corresponding to what the user typed before hitting the ENTER key.
- If we want to add two numbers we need to use storage outside the event handler, so the implementation might look like the following.

A Trivial Event Handler

```
Declare globally the counter K and the integer T.  
OnKeyEnter(character string S)  
{  
    convert S to a number N  
    if K is zero store N in T and increment K  
    otherwise add N to T, print the result and reset K to zero  
}
```

Windows Events

- The event object is a kernel object that stays **nonsignaled until a condition is met**.
- The programmer has the control over setting the event object to a signaled or a nonsignaled state.
- This is unlike a mutex or semaphore where the operating system governs the signaled and nonsignaled state of the object.

Windows Events

- The concept of an event object is very similar to a condition variable, giving the programmer maximum flexibility to define complex synchronization objects.
- There are two types of events: manual-reset events and auto-reset events.
 - A **manual-reset event** can be returned to a nonsignaled state only when reset by the programmer.
 - An **auto-reset event** is set back to the nonsignaled state after the completion of a wait

Windows Messaging

- Microsoft Foundation Class (MFC) Library supports a message driven programming model.
- Its central component is the Message Map.
- Generally implemented as macros in Win32 applications.

Windows Messaging

- The Microsoft Windows operating system requires user-interactive processes that wish to run on the operating system to construct a message loop for responding to events.
- In this operating system, a **message** is equated to an **event** created and imposed upon the operating system.
- An event can range from user interaction, network traffic, system processing, timer activity, and inter-process communication among others.

Windows Messaging

- The "heart" of most Win32 applications is the WinMain function, which calls GetMessage(), in a loop.
- GetMessage blocks until a message, or "event", is received. After some optional processing, it will call DispatchMessage(), which dispatches the message to the relevant handler, also known as WindowProc.

Windows Messaging

```
// Macro calls to create message handling infrastructure
//
BEGIN_MESSAGE_MAP( MFC_Window, CFrameWnd)
//Macro to map the left button click to the handler
ON_WM_LBUTTONDOWN()
//Macro to map the left button click to the handler
ON_WM_LBUTTONUP()
END_MESSAGE_MAP()
```

Event Handler

```
// Message handler framework provided.
// Programmer inserts code for actions taken
//
void MFC_Window::OnLButtonDown(UINT nFlags, CPoint point)
{
    // TODO: Add your message handler code here

    CFrameWnd::OnLButtonDown(nFlags, point);
    m_StartPoint = point;
}

See example application: Windows_Messaging_Example
```

Event Loop

- The event loop, message dispatcher, message loop or **message pump** is a programming construct that waits for and dispatches events or messages in a program.
- It works by polling some internal or external "event provider", which generally blocks until an event has arrived, and then calls the relevant event handler ("dispatches the event").
- The event loop almost always operates asynchronously with the message originator.

Event Loop

- When the event loop forms the central control flow construct of a program, as it often does, and is thus at the highest level of control within the program, it may be termed the **main loop** or main **event loop**.
- Message pumps are said to 'pump' messages from the program's message queue (assigned and usually owned by the underlying operating system) into the program for processing.
- An event loop is a tool of inter-process communication.
- The event loop is a specific implementation technique of systems that use message passing.

PeekMessage Function

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

PeekMessage Function

- The PeekMessage function dispatches incoming sent messages, checks the thread message queue for a posted message, and retrieves the message (if any exist).
- PeekMessage retrieves messages associated with the window identified by the hWnd parameter (if any).
- See PeekMessage Example.

References

- <http://www.microsoft.com/msj/0795/dilascia/dilascia.aspx>
- http://en.wikipedia.org/wiki/Event_loop
- <http://www.wehlo.com/Code/msgthreads/index.htm>
- <http://www.codeproject.com/KB/dialog/messagehandling.aspx>
- http://wiki.winehq.org/List_Of_Windows_Messages
- http://en.wikipedia.org/wiki/Event-driven_programming
- <http://www.wehlo.com/Code/msgthreads/index.htm>
- [http://msdn2.microsoft.com/en-us/library/ms686915\(VS.85\).aspx](http://msdn2.microsoft.com/en-us/library/ms686915(VS.85).aspx)
- <http://softwarecommunity.intel.com/articles/eng/3009.htm>