More Win32 API
- More Mouse Stuff
- GetSystemMetrics()
- WM_PAINT Messages
- GetClientRect()
- TextOut()
- wsprintf()
- Using Fonts
- GetTextMetrics()
- Example Program

Mouse Messages
- Client Area Mouse Messages—
  - Mouse msgs generated when mouse moves over the window’s client area
  - or when pressed/released within window’s client area
  - 21 messages in all
  - WM_MOUSEMOVE: Sent to window under cursor when mouse moved
    - lParam = mouse cursor X,Y position
    - wParam: Mouse notification code
      - MK_LBUTTON, MK_RBUTTON, MK_SHIFT, MK_CONTROL

Client Area Mouse Messages—
- Mouse msgs generated when mouse moves over the window’s client area
- or when pressed/released within window’s client area
- 21 messages in all
- WM_MOUSEMOVE: Sent to window under cursor when mouse moved
  - lParam = mouse cursor X,Y position
  - wParam: Mouse notification code
    - MK_LBUTTON, MK_RBUTTON, MK_SHIFT, MK_CONTROL

Input Focus
- Window whose caption line is highlighted has "input focus"
- Only this window will receive keyboard input
- Run 2 instances of Winapp2
  - Note: keyboard accelerators only work with instance that has input focus
- Input focus not significant for mouse input
- Good since mouse is used to activate a window

When a window gains (loses) input focus:
- WM_SETFOCUS (WM_KILLFOCUS) message sent to window
- Common responses:
  - highlight an edit area, change a caption, etc.
- SetFocus(hWnd)
  - Give a window (or a control) the input focus
- Response to receiving input focus depends on window style

Nonclient Area Mouse Messages
- Mouse actions in other parts of window
  - WM_NC* messages sent
    - (* = MOUSEMOVE, etc.)
    - wParam: HT* hit test code
      - non-client area where action occurred
      - lParam: mouse cursor position
      - Usually not processed by applications
      - Could use to generate other messages
        - e.g., WM_NCLBUTTONDOWN + coordinates => WM_COMMAND
Capturing the Mouse
- To limit mouse to interacting with just one program
- e.g., screen capture program
- Application that does this has "captured" the mouse
- Only it will receive mouse messages.
- Use: SetCapture(hWnd);
- Release with: ReleaseCapture(void);

Getting information on user interface items
- Use:
  GetSystemMetrics(nIndex)
  - nIndex specifies which item
  - See online help

WM_PAINT Messages
- Sent any time client area is invalidated (exposed)
- Should redraw everything in exposed area
- Use BeginPaint(hWnd,&ps) to get a DC
- ps is a pointer to a PAINTSTRUCT
  - contains info about area to be redrawn
- Use EndPaint(hWnd,&ps) to release the DC

typedef struct PAINTSTRUCT
{
  HDC hdc;        // device context handle
  BOOL fErase;   // should background be redrawn? T/F
  RECT rcPaint;  // rectangular area to update
  BOOL fRestore;  // reserved for use by Windows
  BOOL fIncUpdate;           // reserved
  BYTE rgbReserved[16];  // reserved
} PAINTSTRUCT;

WM_PAINT Message
- If you want to keep stuff already drawn in your window after it’s exposed:
  - You need to keep track of everything drawn
  - Then redraw in response to WM_PAINT

Forcing a WM_PAINT
- InvalidateRect(hWnd,&rect,bErase);
  - parameters:
    - window to be invalidated
    - rectangular area (NULL => entire client area)
    - background erased (TRUE/FALSE)
- Causes a WM_PAINT message to be placed on the queue
- This could be done in response to mouse & other messages
Determining Client Area

- GetClientRect(hWnd,&rect)
  - rect pointer will contain (0,0,width,height)
  - You may need to know this
    - for animations

Displaying Text

- TextOut(hdc,x,y,lpTxt,cbTxt);
  - x,y: position on client area of window
  - lpTxt: string to be displayed
  - cbTxt: length of the string
  - current DC text color & bkgnd color used
  - current DC font is used
  - can use lstrlen() to get cbTxt
    - for example:
      char cBuf[] = "Hello, World";
      TextOut(hdc, 0, 0, cBuf, lstrlen(cBuf));

Displaying Numeric Values

- Must format values into a string
- Can use wsprintf()
- See online help
- Example:
  char cBuf[50];
  int num = 19;
  wsprintf(cBuf, "The number is: %d ", num);
  TextOut(hdc, 10, 10, cBuf, lstrlen(cBuf));

Using and Changing Fonts

- FONT: Typeface, style, size of characters in a character set
- Three basic kinds of fonts--
  - Stock fonts--built into Windows, always available
  - Logical or GDI fonts--defined in separate .fon (stroke or raster) or .fot/.ttf (TrueType) font resource files in \windows\system and stored on disk
  - Device fonts--native to the output device (e.g., built-in printer fonts).

Some Stock Fonts

<table>
<thead>
<tr>
<th>Font</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANSI_FIXED_FONT</td>
</tr>
<tr>
<td>ANSI_VAR_FONT</td>
</tr>
<tr>
<td>DEVICE_DEFAULT_FONT</td>
</tr>
<tr>
<td>OEM_FIXED_FONT</td>
</tr>
<tr>
<td>SYSTEM_FONT</td>
</tr>
<tr>
<td>SYSTEM_FIXED_FONT</td>
</tr>
</tbody>
</table>

Windows Stock Fonts

Some Stroke Fonts

<table>
<thead>
<tr>
<th>Font</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modern AaBbCcDdEe</td>
</tr>
<tr>
<td>Roman AaBbCcDdEe</td>
</tr>
<tr>
<td>Script AaBbCcDdEe</td>
</tr>
</tbody>
</table>

Windows Stroke Fonts
Some Raster Fonts

<table>
<thead>
<tr>
<th>Font</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Courier AaBbCcDdEe</td>
<td>Windows Raster Fonts</td>
</tr>
<tr>
<td>MS Serif AaBbCcDdEe</td>
<td></td>
</tr>
<tr>
<td>MS Sans Serif AaBbCcDdEe</td>
<td></td>
</tr>
</tbody>
</table>

Some True Type Fonts

<table>
<thead>
<tr>
<th>Font</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Courier New AaBbCcDdEe</td>
<td>Windows TrueType Fonts</td>
</tr>
<tr>
<td>Courier New Bold AaBbCcDdEe</td>
<td></td>
</tr>
<tr>
<td>Courier New Italic AaBbCcDdEe</td>
<td></td>
</tr>
<tr>
<td>Times New Roman AaBbCcDdEe</td>
<td></td>
</tr>
<tr>
<td>Times New Roman Bold AaBbCcDdEe</td>
<td></td>
</tr>
<tr>
<td>Times New Roman Italic AaBbCcDdEe</td>
<td></td>
</tr>
<tr>
<td>Arial AaBbCcDdEe</td>
<td></td>
</tr>
<tr>
<td>Arial Bold AaBbCcDdEe</td>
<td></td>
</tr>
<tr>
<td>Arial Italic AaBbCcDdEe</td>
<td></td>
</tr>
</tbody>
</table>

Using Stock Fonts

- **GetStockObject()**
  - returns handle to the desired font
  - can be selected into a DC

```c
HDC    hDC;
HFONT  hFont;
hDC = GetDC(hWnd);
hFont = GetStockObject(ANSI_VAR_FONT);
SelectObject(hDC,hFont);
```

Using Logical Fonts

- Obtain a handle to the font data resource and select it into the DC
  - Just like a stock font, except it's loaded from separate file (.fon, .fot/.ttf).
  - Use CreateFont() instead of GetStockObject() to load and get a font handle.
  - CreateFont() makes new fonts by interpolating data in a font file
    - New sizes, bold/underlined, rotated/distorted
    - Called logical since they come from program logic not just from a file

```c
hFont = CreateFont(Ht, Width, Escapement, Orientation, Weight, Italic, Underline, StrikeOut, CharSet, OutputPrecision, ClipPrecision, Quality, PitchAndFamily, Facename);
```

CreateFont()

```c
hFont = CreateFont(36, 0, 3000, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,"Roman")
```

Escapement & Orientation

<table>
<thead>
<tr>
<th>Escapement</th>
<th>Orientation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1800</td>
<td>0</td>
</tr>
<tr>
<td>1800</td>
<td>0</td>
</tr>
<tr>
<td>7100</td>
<td>7100</td>
</tr>
<tr>
<td>7100</td>
<td>7100</td>
</tr>
</tbody>
</table>

Character Escapement & Orientation
Determining Character Sizes

- With CreateFont(), may not get what you want
- Use GetTextMetrics(hDC,lpTextmetric)
  - See online help

\begin{center}
\begin{tabular}{c}
\hline
\textbf{Font Measurements} \tabularnewline
\hline
\end{tabular}
\end{center}

\begin{itemize}
\item User types ==> blue text in client area
\item Can change font from menu
\item Backspace editing feature
\item cBuf[] builds text string as it's input
\item WM_CHAR message received ==> character tested & appended to cBuf if:
  \begin{itemize}
  \item Character is alphanumeric \texttt{IsCharAlphaNumeric()}
  \item Or character is punctuation \texttt{IsAnsiPunc()} returns TRUE
  \item And cBuf[] hasn't been filled
\end{itemize}
\end{itemize}

\begin{itemize}
\item To display, force a WM_PAINT message
  \begin{itemize}
  \item InvalidateRect()
  \end{itemize}
\item Response: draw cBuf[] string
\item Also string will be redrawn automatically after exposure (resizing, uncovering)
\end{itemize}

\begin{itemize}
\item WM_CHAR for printable characters
\item WM_KEYDDOWN for Backspace
\item IsAnsiPunc() -- a helper function that tests ranges of ANSI codes for punctuation characters
\item WM_CREATE when program starts -->
  \begin{itemize}
  \item Use CreateFont() to create new Roman font & save handle in hFont
  \end{itemize}
\item WM_COMMAND to choose font from a popup menu (set nFontChoice variable)
\end{itemize}

\begin{itemize}
\item WM_PAINT message:
  \begin{enumerate}
  \item Get a DC with BeginPaint()
  \item Change color to blue
  \item Check value of nFontChoice
  \item SelectObject() to select chosen font into DC
  \item TextOut() to output the cBuf[] string
  \item Release DC with EndPaint()
  \end{enumerate}
\item Note use of static variables to "remember" variable values from one WndProc() callback to another
\end{itemize}