Notes on Winsock (1)

- An API used for TCP/IP to communicate with Windows applications.
- winsock.dll, wsock32.dll

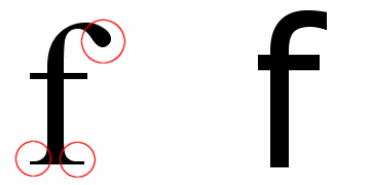
Application using Winsock
→
winsock.dll/wsock32.dll
→
TCP/IP
→
Modem
→
Network

Notes on Winsock (2)

- A socket is a network connection between two computers.
- To create a socket, you will need:
  - The **IP address** of the computer you are connecting to.
  - The **Port** you want to connect to.
- You can check out:
  - http://world.std.com/~jimf/papers/sockets/winsock.html
  - http://www.district86.k12.il.us/central/activities/computerclub/Tutorials/Winsock/Index.htm

Font Lingo (1)

- Type face: a font design.
- Type family: a group of fonts belonging to the same type face but include italics, bold, etc… variants.
- Serif:
- Sans-Serif:



Font Lingo (2)

- Fixed-width font (mono spaced):
  courier new
  wwwwwww
  iiiiiii
- Proportional-width font:
  times new roman
  wwwwwww
  iiiiiii
- Point – a unit to measure the size of a font. a point is approximately 1/72 th of an inch.
- Point size – the height of a font measured in points.
- Weight – the darkness of a font (how bold it is).

Font Families in Windows

Decorative	Novelty fonts. E.g. Old English.
Modern	Mono spaced with or without serifs. E.g. Courier New.
Roman	Proportional font with serifs. E.g. Times New Roman.
Swiss	Proportional font without serifs. E.g. Arial.
Script	Looks like handwriting. E.g. Script.
Dontcare	No font family info is known or does not matter when creating the font.

Font Technologies

- Raster: Bitmapped fonts, designed for a specific resolution of a device.
- Vector: Uses scalable lines and curves. This makes them resolution independent.
- TrueType: Similar to Vector fonts but optimised for fast drawing speed.
- OpenType: Similar to TrueType but shape definitions may contain PostScript data too.

Character Sets

- Each font has a specific character set.
- A character set defines which shapes/printable characters (letters, punctuation, symbols, etc…) are defined in the font.
- Each character is identified by a number.
- Most character sets are supersets of ASCII (retaining the meanings for the fonts numbered from 32 to 127).

Common Character Sets (1)

- The **Windows character set** – very similar to the ASCII set from 32 up to 255. The first character is conveniently the space/blank.
- The above character sets uses 1 byte to represent a character (i.e. 2^8=255 chars). This is enough for most western languages including diacriticals. Eastern languages are not catered for so the **Unicode** standard was adopted that uses 16 bits to identify a character.

Common Character Sets (2)

- The **OEM character set** is the font used in full screen text mode/console applications. Characters 32 to 255 are similar to ASCII and the Windows character set but the lower 32 characters are used for custom symbols (see http://www.ascii-table.org/).
- The **Symbol character set** is a font containing symbols (e.g. math symbols).
- **Vendor Specific**.

Mapping Modes

- In Windows, mapping modes define how values describing the sizes of objects are interpreted. For example:

Mapping Modes (not all)	Each unit is equivalent to…
MM_HIMETRIC	0.001 millimeters
MM_HIENGLISH	0.001 inches
MM_TEXT	1 pixel
MM_TWIPS	1/1440 of an inch

- To select a mapping mode for a device context, you use the **SetMapMode** API call.

CreateFont

- A font instance may be created using the **CreateFont** GDI function.
- Arguments to the function include:
  - The desired height of the font,
  - The average width of the font,
  - Weight,
  - …
  - Face name.
- Usually the height parameter is passed in Point Sizes and the width is set to 0 for the font engine to automatically choose the best value.
- The function however does not expect the height value in point sizes but in **logical units** depending on the current mapping mode.
- To use point sizes, IF you are in the MM_TEXT mapping mode you can use the following expression to convert the desired point size to logical units:

  MulDiv(**PointSize**, GetDeviceCaps(hDC, LOGPIXELSY), 72) * -1

- Muldiv divides two 32-bit numbers and divides the resulting 64-bit number by another 32-bit number.

Printing Text

- Once the font has been created, it is selected as the default font of the device context using the **SelectObject** API call.
- Now a function like **TextOut** can be called to draw text in that font on the specified DC.
- Once we are ready from the font, it's memory is freed using **DeleteObject**.

Using Controls

- As mentioned earlier on, a lot of basic Windows controls are windows themselves. It is no surprise that these controls are created using the **CreateWindow** API call.
- However, in this case we would use predefined windows class names. Some include:
  - BUTTON
  - COMBOBOX
  - EDIT
  - LISTBOX
  - STATIC (a label)

Creating a Button

```
hwndButton = CreateWindow("BUTTON",
                          "OK",
                          <button styles>,
                          ...
                          hWnd, // parent window
                          101,  // Control ID
                          ...
```

- Button styles include:
  - BS_PUSHBUTTON (normal button)
  - BS_DEFPUSHBUTTON (like above but has a heavy border and responds to the enter key even if not focused)
  - WS_CHILD is used to tell Windows that the button is the child of a container window.

Handling Controls in the Callback

- When a button event occurs (like a click) the parent window receives a WM_COMMAND message.
- The LoWord of the wParam tells us the ID of the control that generated the command message.
- The HiWord of the wParam tells us the notification message for the control (e.g. BN_CLICKED if a button is clicked).
- The lParam contains the window handle of the control.