Input

Input devices
Text entry
Positional input
MacBook Wheel (The Onion)
- https://youtu.be/9BnLbv6QYcA
- https://www.student.cs.uwaterloo.ca/~cs349/videos/macbook_wheel.mp4
iPod Wheel
Dimensions to Classify Computer Input

- Sensing Method
- Continuous vs. Discrete
- Degrees of Freedom (DOF)
- Type of Data Managed
Specific vs. General Input Devices

- Specific input devices optimized for specific tasks
  - Problems?

- General input devices adapted to many task
  - Problems?
Text Input

QWERTY

Physical vs. virtual keyboards
Typewriters and QWERTY

- Original design intended for typing on paper
- QWERTY not designed to slow typing down → designed to space “typebars” to reduce jams and speed typing up

1874 QWERTY patent drawing

http://www.daskeyboard.com
QWERTY Properties

- Standard for Latin-script alphabets

- Properties
  - Alternate hands when typing, for improved efficiency
  - Computer version adds function keys, cursor keys, meta keys
  - Can be modified for different locales (e.g. pound key, accents)
  - Variants also exist: AZERTY (French), QWERTZ (Czech)
QWERTY Problems?

- Common combinations
  - awkward finger motions. (eg: tr)
  - a jump over home row. (eg: br)
  - are typed with one hand. (e.g. was, were)
- Most typing with the left hand (thousands of words left vs. hundreds right)
- About 16% of typing uses lower row, 52% top row, 32% home row

```
<table>
<thead>
<tr>
<th>~</th>
<th>!</th>
<th>@</th>
<th>#</th>
<th>$</th>
<th>%</th>
<th>^</th>
<th>&amp;</th>
<th>*</th>
<th>(</th>
<th>)</th>
<th>-</th>
<th>+</th>
<th>=</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tab</td>
<td>Q</td>
<td>W</td>
<td>E</td>
<td>R</td>
<td>T</td>
<td>Y</td>
<td>U</td>
<td>I</td>
<td>O</td>
<td>P</td>
<td>{</td>
<td>}</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Caps Lock</td>
<td>A</td>
<td>S</td>
<td>D</td>
<td>F</td>
<td>G</td>
<td>H</td>
<td>J</td>
<td>K</td>
<td>L</td>
<td>:</td>
<td>;</td>
<td>&quot;</td>
<td>Enter</td>
</tr>
<tr>
<td>Shift</td>
<td></td>
<td>Z</td>
<td>X</td>
<td>C</td>
<td>V</td>
<td>B</td>
<td>N</td>
<td>M</td>
<td>&lt;</td>
<td>&gt;</td>
<td>?</td>
<td>/</td>
<td>Shift</td>
</tr>
<tr>
<td>Ctrl</td>
<td>Win Key</td>
<td>Alt</td>
<td></td>
<td>Alt</td>
<td>Win Key</td>
<td>Menu</td>
<td>Ctrl</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```
Dvorak Optimizations

- The most common letters and digraphs should be the easiest to type. Thus, about 70% of keyboard strokes are on home row.

- The least common letters should be on the bottom row, which is the hardest row to reach.

- The right hand should do more of the typing, because most people are right-handed.

- Is it actually more efficient? If so, at what cost?

<table>
<thead>
<tr>
<th>~</th>
<th>`</th>
<th>!</th>
<th>@</th>
<th>#</th>
<th>$</th>
<th>%</th>
<th>^</th>
<th>&amp;</th>
<th>*</th>
<th>(</th>
<th>)</th>
<th>{</th>
<th>}</th>
<th>Backspace</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tab</td>
<td>&quot;</td>
<td>,</td>
<td>&lt;</td>
<td>&gt;</td>
<td>P</td>
<td>Y</td>
<td>F</td>
<td>G</td>
<td>C</td>
<td>R</td>
<td>L</td>
<td>?</td>
<td>+</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Caps Lock</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Shift</th>
</tr>
</thead>
<tbody>
<tr>
<td>:</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Ctrl</th>
</tr>
</thead>
<tbody>
<tr>
<td>Win Key</td>
</tr>
</tbody>
</table>
There is a lot of value in standardization...

Once I've used a computer for a while, no one else will ever use it again.
**Key Codes (see Events)**

- Pressing a key generates a key code i.e. unique numeric value passed to app.

<table>
<thead>
<tr>
<th>ASCII control characters</th>
<th>ASCII printable characters</th>
<th>Extended ASCII characters</th>
</tr>
</thead>
<tbody>
<tr>
<td>00 NULL (Null character)</td>
<td>32 space 64 @ 96 `, 128 Ç</td>
<td>128 Ç 160 á 192 L 224 Ó</td>
</tr>
<tr>
<td>01 SOH (Start of Header)</td>
<td>33 ! 65 A 97 a 129 ü 161 i 193 ‼ 225 ß</td>
<td></td>
</tr>
<tr>
<td>02 STX (Start of Text)</td>
<td>34 &quot; 66 B 98 b 130 é 162 ó 194 ‡ 226 Ő</td>
<td></td>
</tr>
<tr>
<td>03 ETX (End of Text)</td>
<td>35 # 67 C 99 c 131 à 163 ū 195 † 227 Õ</td>
<td></td>
</tr>
<tr>
<td>04 EOT (End of Trans.)</td>
<td>36 $ 68 D 100 d 132 ä 164 ň 196 ‡ 228 ö</td>
<td></td>
</tr>
<tr>
<td>05 ENQ (Enquiry)</td>
<td>37 % 69 E 101 e 133 à 165 Ñ 197 ‡ 229 Ő</td>
<td></td>
</tr>
<tr>
<td>06 ACK (Acknowledgement)</td>
<td>38 &amp; 70 F 102 f 134 â 166 Â 198 ã 230 μ</td>
<td></td>
</tr>
<tr>
<td>07 BEL (Bell)</td>
<td>39 ' 71 G 103 g 135 ç 167 ° 199 ò 231 p</td>
<td></td>
</tr>
<tr>
<td>08 BS (Backspace)</td>
<td>40 ( 72 H 104 h 136 ê 168 ã 200 ‡ 232 b</td>
<td></td>
</tr>
<tr>
<td>09 HT (Horizontal Tab)</td>
<td>41 ) 73 I 105 i 137 ê 169 © 201 ‡ 233 Ú</td>
<td></td>
</tr>
<tr>
<td>10 LF (Line feed)</td>
<td>42 * 74 J 106 j 138 è 170 ™ 202 † 234 Õ</td>
<td></td>
</tr>
<tr>
<td>11 VT (Vertical Tab)</td>
<td>43 + 75 K 107 k 139 ê 171 ¾ 203 † 235 Õ</td>
<td></td>
</tr>
<tr>
<td>12 FF (Form feed)</td>
<td>44 , 76 L 108 l 140 î 172 ¾ 204 † 236 y</td>
<td></td>
</tr>
<tr>
<td>13 CR (Carriage return)</td>
<td>45 - 77 M 109 m 141 î 173 ï 205 † 237 Ý</td>
<td></td>
</tr>
<tr>
<td>14 SO (Shift Out)</td>
<td>46 . 78 N 110 n 142 Ä 174 « 206 † 238 Œ</td>
<td></td>
</tr>
<tr>
<td>15 SI (Shift In)</td>
<td>47 / 79 O 111 o 143 Ä 175 » 207 † 239 Œ</td>
<td></td>
</tr>
<tr>
<td>16 DLE (Data link escape)</td>
<td>48 0 80 P 112 p 144 É 176 ¹ 208 ‡ 240 ¶</td>
<td></td>
</tr>
<tr>
<td>17 DC1 (Device control 1)</td>
<td>49 1 81 Q 113 q 145 æ 177 ² 209 ‡ 241 ±</td>
<td></td>
</tr>
<tr>
<td>18 DC2 (Device control 2)</td>
<td>50 2 82 R 114 r 146 Æ 178 ¾ 210 † 242 §</td>
<td></td>
</tr>
<tr>
<td>19 DC3 (Device control 3)</td>
<td>51 3 83 S 115 s 147 Ő 179 ½ 211 ‡ 243 %</td>
<td></td>
</tr>
<tr>
<td>20 DC4 (Device control 4)</td>
<td>52 4 84 T 116 t 148 Õ 180 ½ 212 † 244 ¶</td>
<td></td>
</tr>
<tr>
<td>21 NAK (Negative acknowl.)</td>
<td>53 5 85 U 117 u 149 ô 181 Â 213 ‡ 245 ¥</td>
<td></td>
</tr>
<tr>
<td>22 SYN (Synchronous idle)</td>
<td>54 6 86 V 118 v 150 ų 182 ¥ 214 ‡ 246 $</td>
<td></td>
</tr>
<tr>
<td>23 ETOB (End of trans. block)</td>
<td>55 7 87 W 119 w 151 ų 183 ¥ 215 ‡ 247 ¥</td>
<td></td>
</tr>
<tr>
<td>24 CAN (Cancel)</td>
<td>56 8 88 X 120 x 152 ÿ 184 © 216 ‡ 248 $</td>
<td></td>
</tr>
<tr>
<td>25 EM (End of medium)</td>
<td>57 9 89 Y 121 y 153 Ô 185 ¶ 217 ‡ 249 %</td>
<td></td>
</tr>
<tr>
<td>26 SUB (Substitute)</td>
<td>58 : 90 Z 122 z 154 Ŭ 186 ‹ 218 ‡ 250 .</td>
<td></td>
</tr>
<tr>
<td>27 ESC (Escape)</td>
<td>59 ; 91 [ 123 { 155 ø 187 ‹ 218 ‡ 251 .</td>
<td></td>
</tr>
<tr>
<td>28 FS (File separator)</td>
<td>60 &lt; 92 \ 124</td>
<td>156 £ 188 ‹ 219 ‡ 252 ‹</td>
</tr>
<tr>
<td>29 GS (Group separator)</td>
<td>61 = 93 ] 125 } 157 Ø 189 ç 219 ‡ 253 %</td>
<td></td>
</tr>
<tr>
<td>30 RS (Record separator)</td>
<td>62 &gt; 94 ^ 126 ~</td>
<td></td>
</tr>
</tbody>
</table>
Unicode

- The world needs more than 255 characters!
  - (Extended) ASCII was limited to 255 characters (i.e. 8 bits).

- **Unicode** is a superset of ASCII, that has replaced it in common use
  - Values 0-127 have the same meaning in both (e.g. ‘A’ == 65)
  - Uses multiple bytes to store character information, which greatly increases the range of values
  - Denoted as UTF-xx where xx is the minimum number of bits.

- UTF-8 is the standard method of encoding characters
  - minimum 8 bits.
  - Capable of encoding all 1,112,064 code points in Unicode (characters, control codes, other meaningful characters)
Mechanical Design of Keyboards

- To increase portability of devices, keyboards are frequently downsized
  - Smaller, low-profile keys
  - Shorter travel distance
  - Sometimes fewer keys

- All interfere with typing, or reduce efficiency!
“Virtual” Keyboards

- Touch screen or other flat surface
- Problems:
  - small keys reduce accuracy
  - no mechanical feedback makes it hard to tell if key was pressed
  - no tactile feedback makes it hard to find the home row
  - resting of hands difficult
- Advantage:
  - portable, no extra hardware
  - customizable keys (e.g. new language, symbols, emojis)
  - customizable layout or functionality (e.g. swipe, thumb layout)
Chording Keyboards

- Englebart’s NLS Keyboard
  - Multiple keys together produce letter
  - No hand “targeting”, potentially very fast
  - Can be small and portable
  - One handed

- Thad Starner’s Twiddler
  - for wearable computing input
Alternatives for Text Entry: Text Recognition and Gestures

- Graffiti/Unistroke Gestures
  - map single strokes to “enter letter” commands
- Natural Handwriting recognition
  - dictionary-based classification algorithms
Alternatives for Text Entry: Predictive Text

- Use language characteristics to predict input
  - Given characters typed so far, what are the most likely intended words?
  - Given words typed so far, what is the most likely word to follow?

- A variation is used for T9, nine-key text entry
  - Given an ambiguous set of characters, what is the most likely word

- Possible Problems
  - “collisions” between common words
  - entering words not in dictionary difficult
  - hard to focus on typing and monitoring prediction
Alternatives for Text Entry: Gestural Text Input

ShapeWriter
http://www.shuminzhai.com/shapewriter_research.htm

8Pen Keyboard
http://www.8pen.com/
## Text Input Expert-User Input Rates

<table>
<thead>
<tr>
<th>Device</th>
<th>Input Rates</th>
</tr>
</thead>
<tbody>
<tr>
<td>Qwerty Desktop</td>
<td>80+ WPM typical, record: 150 WPM for 50 minutes</td>
</tr>
<tr>
<td>Qwerty Thumb</td>
<td>60 WPM typical with training (Clarkson et al., CHI 2005)</td>
</tr>
<tr>
<td>Soft Keyboards</td>
<td>45 WPM</td>
</tr>
<tr>
<td>T9</td>
<td>45 WPM possible for experts (Silverberg et al., CHI 2000)</td>
</tr>
<tr>
<td>Gestural</td>
<td>~30 WPM 8Pen, ShapeWriter claims 80 WPM (expert)</td>
</tr>
<tr>
<td>Handwriting</td>
<td>33 WPM (Wilkund et al., Human Factors Society, 1987)</td>
</tr>
<tr>
<td>Graffiti 2</td>
<td>9 WPM (Koltringer, Grechenig, CHI 2004)</td>
</tr>
</tbody>
</table>
Positional Input

Positional devices
Direct vs. indirect
Absolute vs. relative
Properties of Positional Input Devices

- **Force vs. Displacement Sensing**
  - (most) joysticks = force
  - mouse = displacement

- **Position vs. Rate Control**
  - (most) joysticks = rate
  - mouse = position

- **Absolute vs. Relative Positioning**
  - touchscreen = absolute
  - mouse = relative

- **Direct vs. Indirect Contact**
  - direct = touchscreen
  - indirect = mouse

- **DOF (Dimensions) Sensed**
Force vs. Displacement Sensing

- **Isometric** devices measure force
  - elastic isometric vs. “pure” isometric devices
  - e.g. joysticks

- **Isotonic** devices measure displacement
  - e.g. mouse

Integrated, elastic pointing device.

Most joysticks are elastic, and spring back to neutral/middle.

A wii remote thumb stays where it is positioned.
Position vs. Rate Control Transfer Function

- force sensing should be mapped to rate (e.g. joystick, pedal)
- displacement sensing should be mapped to position (e.g. mouse)

[Diagram showing the relationship between rate and force/speed, and position and displacement.]
Position Control & Managing Coordinates

- Cartesian coordinates vs. computer coordinates

Positional control (displacement sensing) devices are more common for desktop interaction, and report screen coordinates.
- Clicking: point
- Dragging: series of points
Direct vs. Indirect Input

- **Indirect**: the position of the cursor is controlled by some external device.

- **Direct**: the position of the cursor is controlled by direct contact with the screen.
Absolute vs. Relative Position

- **Absolute position** is a direct mapping of input device position to a display input position. Examples?

- **Relative position** maps changes in input device position to changes in display input position. Examples?
Combinations

Absolute Direct

Relative Indirect
Clutching and Relative Positioning

- **Scenario**: you’re moving the mouse and you hit the edge of the desk before you finish positioning the mouse. What do you do?
- To make relative position work, you need to **clutch** (i.e. repeatedly move to achieve the target)
- Clutching is **one** solution to making relative positioning work
Control-Display Gain (CD Gain)

- Second solution for relative input: manipulate the movement itself.
- Ratio of *display pointer* movement to *device control* movement
  - the ratio is a scale factor (the “gain”)
  - usually expressed as velocity, works for rate control and position control

\[
\text{CDgain} = \frac{V_{\text{pointer}}}{V_{\text{device}}}
\]

Graphs showing different CD gains:
- **CDgain = 1**
- **CDgain = \(\frac{1}{2}\)**
- **CDgain = 2**
Pointer Acceleration Manipulated CD Gain (*aka Mouse Acceleration*)

- Dynamically change CD Gain based on device velocity; can reduce the need to clutch

(Casiez et al. 2008)
Hybrid Absolute and Relative Pointing

- [http://youtu.be/FZmOB1g5KjM](http://youtu.be/FZmOB1g5KjM)