CS160: User Interface Design

Due Today

1. Program Assignment 3
2. Competitive Analysis

Group Brainstorm

Mean: 15.3
Mode: 15
Median: 15
StdDev: 1.73
Range: 12-19
"What has definitely been covered in the field of skateboarding apps is instructional videos and skatepark/shop/spots locators (that use GPS to get directions). No apps really give a skater a personalized profile where they can keep track of the tricks they have accomplished[...]" (Annette Trujillo)
## Contextual Inquiry and Task Analysis

### Assigned Today

**Programming Assignment 4**  
**Topic:** BART application  
**Due:** Wed 3/3/10

### Competitive Analysis

<table>
<thead>
<tr>
<th>Input</th>
<th>Data Flow</th>
<th>Graphical Syntax</th>
<th>Multisegment Support</th>
<th>Score on Use</th>
<th>Labelled players</th>
<th>Task Feature</th>
<th>Game Price</th>
<th>Game Demo</th>
<th>Due Date</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>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Nathaniel Baldwin

Wei Wu
Programming Assignment 4

Overview
For your next optional individual programming assignment, we will build on our in-class discussion of user interfaces to WiSP. Your task is to design and implement a phone app for the WiSP station that allows users to view and edit their WiSP balance. The app should display the current balance, allow users to add funds, and offer a list of upcoming departures for the nearest train station.

XML Parsing

XML documents can be parsed using various languages and frameworks. Here are a few examples:

- **Python**: Using the `xml.etree.ElementTree` module.
  ```python
  import xml.etree.ElementTree as ET
  tree = ET.parse('example.xml')
  root = tree.getroot()
  # Further processing...
  ```

- **Java**: Using the `org.w3c.dom` API or `org.xml.sax.SAXParser`.
  ```java
  import java.io.StringReader;
  import javax.xml.parsers.ParserConfigurationException;
  import javax.xml.parsers.SAXParser;
  import javax.xml.parsers.SAXParserFactory;
  import org.xml.sax.InputSource;
  import org.xml.sax.SAXException;
  import org.xml.sax.XMLReader;
  import org.xml.sax.XMLReaderFactory;
  import org.xml.sax.helpers.DefaultHandler;
  import org.xml.sax.helpers.XMLReaderFactoryImpl;

  public class XMLParser {    
    public void parse(String xmlString) throws SAXException, ParserConfigurationException {    
      SAXParserFactory factory = SAXParserFactory.newInstance();    
      SAXParser parser = factory.newSAXParser();    
      XMLReader reader = XMLReaderFactoryImpl.createXMLReader();    
      DefaultHandler handler = new DefaultHandler();    
      parser.parse(new InputSource(new StringReader(xmlString)), handler);    
    }
  }
  ```

- **JavaScript**: Using the `DOMParser` object.
  ```javascript
  const parser = new DOMParser();
  const xmlDoc = parser.parseFromString(xmlString, 'text/xml');```

These examples provide a starting point for parsing XML documents in different programming languages.
Questions:

What (low-level) tasks are the users trying to accomplish with an input device?

How can we think about the space of possible input devices?

What interaction techniques are encouraged/discouraged by a particular device?

Important Tasks

Text Entry
Pointing/Marking
- Target acquisition
- Steering/positioning
- Freehand drawing
- Drawing lines
- Tracing and digitizing
- …
Text Entry: Keystroke Devices

Array of Discrete Inputs
Many variants of form and key layout
Can be one-handed or two
Wide range of sizes
Two-hand full keyboard is relatively standardized. Less standardization on others: Command keys, generic function keys, cursor movement, numeric keypad...

Take advantage of procedural memory
Power law of practice

\[ T_n = T_1 n^{-a} + c \]

Keyboards

Key Layouts

<table>
<thead>
<tr>
<th>QWERTY</th>
<th>DVORAK</th>
</tr>
</thead>
<tbody>
<tr>
<td>!@#$%^&amp;*</td>
<td>!@#$%^&amp;*</td>
</tr>
<tr>
<td>`12345678</td>
<td>`12345678</td>
</tr>
<tr>
<td>987654321</td>
<td>987654321</td>
</tr>
<tr>
<td>QWERTY</td>
<td>QWERTY</td>
</tr>
<tr>
<td>12345678</td>
<td>12345678</td>
</tr>
<tr>
<td>QWERTY</td>
<td>QWERTY</td>
</tr>
</tbody>
</table>
Mobile Text Entry: Keypads

Multi-tap mappings
Multiple presses per letter

Ambiguity resolution
One press per letter; dictionary lookup

Chording
Multiple keys pressed simultaneously
\(2^n\) combinations for \(n\) keys

Mobile Text Entry: Touch / Stylus

Soft Keyboards
Benefits! Drawbacks!

Mobile Text Entry: Handwriting Recog.
Mobile Text Entry: Touch / Stylus

Custom symbol sets improve recognition accuracy; appropriate for indirect (eyes-free) input

Stroke Entry Methods (e.g., Swype, ShapeWriter)

Which is fastest?

<table>
<thead>
<tr>
<th>Technique</th>
<th>Novice</th>
<th>Expert</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard Keyboard</td>
<td>30 wpm</td>
<td>70 wpm</td>
</tr>
<tr>
<td>Soft Keyboard</td>
<td>9 wpm</td>
<td>43 wpm</td>
</tr>
<tr>
<td>T9</td>
<td>11 wpm</td>
<td>46 wpm</td>
</tr>
<tr>
<td>Handwriting</td>
<td>16 wpm</td>
<td>27 wpm</td>
</tr>
<tr>
<td>Multi-press</td>
<td>7 wpm</td>
<td>27 wpm</td>
</tr>
</tbody>
</table>

Standard Keyboard: 30 wpm (novice) - 70 (experts); 150 wpm for stenographers

Soft keyboards: 9 wpm (novice) – 43 wpm (expert)

T9: 11 wpm (expert)

Handwriting: 16-20 wpm

Multi-press: 7 wpm (novice) - 27 wpm (expert)
What about Speech Recognition?

Dictation is faster than typing (~100 wpm)

What about Speech Recognition?

Dictation is faster than typing (~100 wpm), BUT:

Speech is different from written language:
Speaking in well-formed, complete, print-ready sentences is cognitively challenging

High cost of correcting errors through speech channel alone

Social awkwardness?
Sensing: Rotary Encoder

Sensing: Fwd Rotation

Sensing: Backwd Rotation

Solution: Use two out-of-phase detectors

Oops!
Sensing: Rotary Encoder

Transformation

\[ cx_t = \max(0, \min( \text{sw}, cx_{t+1} + dx \cdot cd )) \]

\[ cy_t = \ldots \]

cx: cursor x position in screen coordinates at time t
sw: screen width

dx: mouse x movement delta in screen coordinates

cd: control-display ratio

Device Abstraction

Click, DoubleClick, MouseUp, MouseDown, MouseMove ...
What about optical mice?

Source: http://spritesmods.com/?art=mouseeye

What is sensed?

<table>
<thead>
<tr>
<th>Physical Properties Used by Input Devices</th>
<th>Linear</th>
<th>Rotary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Position</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Absolute</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Relative</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Force</td>
<td>Force F</td>
<td>Force T</td>
</tr>
<tr>
<td>Rotation</td>
<td>Rotation R</td>
<td>Rotation T</td>
</tr>
<tr>
<td>Delta Force</td>
<td>Delta Force F</td>
<td>Delta Force T</td>
</tr>
<tr>
<td>Delta Rotation</td>
<td>Delta Rotation R</td>
<td>Delta Rotation T</td>
</tr>
</tbody>
</table>

C.D Ratio

For one unit of movement in physical space, how far does the cursor travel in display space?

Q: What is the C.D ratio for direct touch screen input?

Device Acquisition Time


Other device properties:

Indirect vs. Direct

Direct: Input and output space are unified

C.D Ratio

Q: What is the C.D ratio for direct touch screen input?
Trackball, Trackpad

Trackpoint
Indirect, force sensing, velocity control
Nonlinear transfer function

Mobile Pointing
D-Pad (see: arrow keys)
Trackball
Direct touch (see: Trackpad)
Stylus

Which is faster?
Which is faster?

Experiment: Mice are fastest!

Fitts’ Law

Time $T_{pos}$ to move the hand to target size $S$ which is distance $D$ away is given by:

$$T_{pos} = a + b \log_2 (D/S + 1)$$

*Index of Difficulty (ID)*

Only relative precision matters
**Fitts’ Law**

Time $T_{\text{pos}}$ to move the hand to target size $S$ which is distance $D$ away is given by:

$$T_{\text{pos}} = a + b \log_2 (D/S + 1)$$

- **Device Characteristics**
  - $a$: start/stop time
  - $b$: speed

---

**Bandwidth of Human Muscle Groups**

- **Neck (headmouse)**: 27 bits
- **Arm**: 17 bits
- **Wrist**: 23 bits
- **Finger**: 38 bits

---

**Why is the mouse fastest?**

Why these results?

- Time to position mouse proportional to Fitts’ Index of Difficulty $I_d$.
  - (i.e. how well can the muscles direct the input device)

Therefore speed limit is in the eye-hand system, not the mouse.

Therefore, mouse is a near optimal device.

---

**Fitts’ Law Example**

Which will be faster on average?

- **Pie menu (bigger targets & less distance)**
Fitts’ Law in Windows & Mac OS

- Windows 95: Missed by a pixel
- Windows XP: Good to the last drop

The Apple menu in Mac OS X v10.4 Tiger

Source: screenshot, Internet Explorer, Microsoft, 2007: Apple

Fitts’ Law in Microsoft Office 2007

- Larger, labeled controls can be clicked more quickly
- Mini Toolbar: Close to the cursor

Source: screenshot, Internet Explorer, Microsoft, 2007

Everything is best for something and worst for something else.
- Bill Buxton

3-State Model of Input (Buxton)

<table>
<thead>
<tr>
<th>State</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Out Of Range: The device is not in its physical tracking range.</td>
</tr>
<tr>
<td>1</td>
<td>Tracking: Device motion moves only the cursor.</td>
</tr>
<tr>
<td>2</td>
<td>Dragging: Device motion moves objects on the screen.</td>
</tr>
</tbody>
</table>

(Tables from Hackley Reading)
(Multi-) Touch

**Strengths**

- Direct input allows maximal screen space for mobile devices (ocular centrism).
- More degrees of freedom.
- “Virtual input devices” are adaptable.
- No extra pieces to lose or break (stylus!)

**Challenges** (from Butoni)

- No tactile feedback.
- Requires free use of (both) hands and eyes.
- “Fat Finger” problems – precision & occlusion

**Terminology** (from Butoni)

- Touch-tablets vs Touch screens
- Single-finger vs multi-finger
- Multi-person vs multi-touch
- Points vs Postures
- Hands and fingers vs Objects
Posture-based Interaction

The "Fat Finger" Problem

Graphics: Patrick Baudisch, nanoTouch
A Software Solution

A Hardware Solution: Use the Backside

Hybrids: Keyboards on Interactive Tables
Hybrids: Multi-touch on Mice

FTIR Mouse

Applies the principle of Focused Infrared Reflectance to illuminate a sensor. Trigger and user a sensor to track multiple points of touch on its curved transduced surface.

Next Time

Model-View-Controller, Event-driven UIs
Basics of Event Handling
Developing User Interfaces. Dan Olsen, Ch.4.
Don’t forget to read and submit comment!

Finish up your Contextual Inquiry, get started on Programming Assignment IV!