Lecture 5:
Input Basics

GUI Input Modalities

• How can a user provide input to the GUI?
  - keyboard
  - mouse / joystick
  - touch pad / screen
  - pen / stylus
  - speech
  - other...

Keyboard input

• Short history of the keyboard
  - typewriter patented by Christopher Latham Sholes in 1868
  - Sholes patent sold to Remington & Sons in 1873
    • Remington adopted the QWERTY layout
  - companies held contests to see whose typewriters & typists could enter the fastest
  - why did the QWERTY keyboard win out?
    • people disagree… seemingly part marketing, part luck

Keyboard input

- So how fast can people type anyway?

![Distribution of All Typing Speed Scores](Typing_Speed_How_Fast_is_Average.png)

- Of course, this is touch typing normal text...

Mouse input

- Short history of the mouse
  - Douglas Engelbart invented the mouse in 1963 in a larger project to “augment human intellect”
  - and thereby invented “point and click”

Direction manipulation

- Why is the mouse so effective?
  - at least part of the answer lies in the facilitation of “direction manipulation”

- Direction manipulation involves...
  - visual representation of the manipulated objects
  - physical actions instead of text entry
  - immediately visible impact of the operation

- Actually, “direct manipulation” is a misnomer
  - obviously it’s not direct, but indirect
    - your hand might move 3”; the cursor moves 10” (dependent on your display size)
    - but the key is that it feels direct — because of the real-world metaphor

Example: “Hand” dragging
Direction manipulation

• Example: Desktop folder & file manipulation

Related input

• Trackball
  - nice for games, small footprint
  - why aren’t they mainstream? metaphor?

• Touchpad / trackpad
  - small & thin — perfect for laptops
  - integrates notions of mouse & pen
  - metaphor is maintained

• Trackpoint (IBM)
  - pencil-eraser nub
  - some love ’em, some hate ’em

Pen input

• Pen-based interfaces in mobile computing

Pen input

• Pen input has lots of advantages…
  - users can hold a familiar input device
  - users can write in a familiar input language
  - users can carry with them

  … and some major disadvantages:

Eat up Martha

Beat up Martin
Pen input recognition

- Handwriting
  - very general, well-developed human skill
  - thus, make use of what users can already do!
  - but hard to recognize (for people & machines)
- Gestures
  - gesture alphabets
  - Palm Pilot graffiti
  - editing gestures
  - easier to recognize

Off-line vs. on-line recognition

- Off-line recognition
  - examine static output of handwriting, i.e., the end result of the writing
  - like OCR, but much harder!

- On-line recognition
  - examine dynamic movement of handwriting, i.e., the strokes, pen up/downs involved
  - Which is more “informed”? more useful?

Recognition techniques

- Neural networks
  - neurally-inspired computational models
  - input: bitmap, or “vectorized” strokes
  - output: probably characters
  - best for off-line recognition

- Hidden Markov models (HMMs)
  - powerful probabilistic models
  - input: vectorized strokes
  - output: full recognition of chars, words, etc.
  - best for on-line recognition

On-line feature extraction

- On-line strokes -> feature vectors
  - basic features: pen up/down, direction, velocity
  - useful features: curvature, reversal, ...
  - other features: ascender/descender? ...

And hybrid methods!
On-line recognition

- Hidden Markov models (HMMs)
  - probabilistic models for dynamic behavior
- Set of N states with
  - $a(i,j)$ = probability of state transition $i \rightarrow j$
  - $b(o,i)$ = probability of seeing $o$ in state $i$
    - can be discrete or continuous prob. distributions

Hidden Markov models

- Let’s say we have
  - $M$ = HMM representing predicted behavior
  - $O$ = observation vector sequence $O$
- Three problems
  - evaluation: find $Pr(O|M)$
  - decoding: find the state sequence $Q$ that maximizes $Pr(O|M,Q)$
  - training: adjust parameters of $M$ to increase $Pr(O|M)$

Hidden Markov models

- HMM decoding (Viterbi algorithm)
  - find best state sequence through HMM, maximizing the probability of the sequence
  - assuming $O = < x x y x y >$, try to decode...

Hidden Markov models

- HMM training (Baum-Welch / EM algorithm)
  - re-adjust $a(i,j), b(o,i)$ to increase $Pr(O|M)$
  - iterative procedure
  - allows for fine-tuning of HMM parameters for particular observation sets
    - what’s in the set? everything? a single person? a specific group of people?
      (e.g., R/L-handed for writing, male/female for speech)
  - susceptible to local minima!
    - and this is often a problem!
Hidden Markov models

- Composing HMMs
  - we can add "sub-" HMMs into larger HMMs, creating a model hierarchy at different levels
- For instance, we can create three levels
  - strokes
  - letters
  - words

On-line recognition

- Stroke HMMs with states
  - up-down loop
    - s1: up, + curvature, hi velocity
    - s2: down, + curvature, hi velocity
  - up-down cusp
    - s1: up, + curvature, hi velocity
    - s2: 0 velocity
    - s3: down, + curvature, hi velocity
  - up-down rhamphoid
    - s1: up, - curvature, hi velocity
    - s2: 0 velocity
    - s3: down, + curvature, hi velocity

On-line recognition

- Letter HMMs based on stroke HMMs
  - a
    - ramphoid — cusp
      ("o" ?)
  - m
    - cusp — cusp
      ("w" ?)
  - j
    - loop — down-up loop
      ("g" ?)

On-line recognition

- Word HMMs based on letter HMMs
  - basic idea is straightforward
  - but it’s deceptively tricky — why??
On-line recognition

- Putting it all together
  - compacting states
  - taking word frequencies into account
  - where do frequencies come from?

Handwriting issues

- Vocabulary size
- Individual variability
  - writer dependent / ind.? adaptive?
- Signal segmentation
  - isolated words, continuous
- Speed-accuracy tradeoff
- Printed vs. handwritten?
  - often some combination of the two!!
- Dotting i’s, crossing t’s, … for on-line recog.
- Mixing language with graphics & gestures

Speech input

- Another up & coming input modality
- A few benefits of speech...
  - like handwriting, very natural
  - hands-free
    - you can avoid carpal tunnel
    - you can drive while writing email (ok, benefit??)
- … and a few drawbacks:
  - some people hate talking
  - some people hate other people talking
  - and again, the recognition problem

Speech recognition

- Limited speech recognition
  - only allow small sets of words/phrases
    - e.g., “one” - “nine”
- Full speech recognition
  - again, general, well-developed skill
  - full standard vocabulary (1000-10000+ words)
  - specialized vocabularies (research, medical, …)
  - “editing” vocabularies (back, delete, …)
- Same basic ideas for recognition
  - convert to recognizable signal (transforms)
  - recognize using hybrid methods along with hierarchy of phonemes, words, etc.
Speech issues

- Speech has many of the same issues as pen and handwriting input
  - vocabulary size
  - individual variability
    - speaker dependent, adaptive, independent
  - signal segmentation
    - isolated words, continuous

Research Question of the Day

- How quickly can users enter input given the various input modalities?
- There’s lots of research out there about such questions
- Let’s focus on one: using a laptop trackpad
- What’s faster for button pressing?
  - pressing a button with the thumb?
  - lifting off and tapping the trackpad?
  - some other method?


Research Question of the Day

- MacKenzie & Oniszczak (1998) compared:
  - “button”: pressing a button with the thumb
  - “lift & tap”: lifting off and tapping the trackpad
  - “tactile”; sensing pressure on trackpad
    - shown to right...

Research Question of the Day

- With a simple questionnaire, people showed some interesting opinions:

<table>
<thead>
<tr>
<th>Selection Technique</th>
<th>Speed Perception</th>
<th>Accuracy Perception</th>
<th>Overall Preference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Button</td>
<td>1</td>
<td>15</td>
<td>5</td>
</tr>
<tr>
<td>Lift-and-tap</td>
<td>15</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>Tactile</td>
<td>19</td>
<td>15</td>
<td>17</td>
</tr>
</tbody>
</table>

Figure 10. Questionnaire results. (Note: Scores are totals of participants’ ratings; higher scores are better.)
Research Question of the Day

- Here are the quantitative results:

![Graph showing movement time vs. error rate for different input methods.]

- And when measuring “throughput,” which combines speed & accuracy:

![Bar chart showing throughput for different input methods.]

Research Question of the Day

- How does this throughput compare with other input modalities?
  - trackpad: 1.0 - 1.5 bps (what we just saw)
  - trackball: 2.0 - 3.5 bps
  - mouse: 3.0 - 4.5 bps

Multimodality

- Today’s desktops mostly use mouse & keys
- We’ve already started expanding the horizon
  - speech input is more common
  - pen tablets work & feel better
- Off the desktop, it’s even more important
  - virtual worlds
    - allow for pointing, grabbing, etc.
    - “direct manipulation” becomes even more direct
  - in-vehicle devices
    - visual & manual channels are busy
    - can interface exploit other channels?
    - what are the cognitive implications?
Other input modalities

- Eye-movement input

Redundancy

- Redundancy is a great thing
- Use many input devices together
- Redundancy is a great thing
- Allow the user to choose between them
- Redundancy is a great thing
- And not choose once, but continuously
- Redundancy is a great thing
- With freedom comes speed & usability
- Redundancy is a great thing

Other input modalities

- Sign-language input

Redundancy

- Command examples of redundancy
  - mouse & keyboard
    - e.g., menu commands & keyboard shortcuts
  - commands in different places
    - e.g., menu commands & toolbar items
  - mouse & eye ??
Input in Swing

• Swing monitors many types of input
  • Most common input
    - mouse: click, motion, drag
    - keyboard: keypresses, “focus”
    - (file: read, write)
  • Other inputs
    - speech: through speech recognition API
    - pen: through 3rd-party APIs and toolkits
      • e.g., SATIN developed at UC Berkeley

So how do I write the code?

• Swing prefers not to give you input directly
  • e.g., mouse position, keystroke
  • this is a good thing!
  • you can still read such input; it’s just not preferable
  • instead (as you know) it’s based on events that are processed in the context of a GUI component
    • e.g., mouse click as an event on a button
    • e.g., keystroke as an event in a text component
    • e.g., keystroke as a shortcut to a menu command