Interaction Design - part 2

- Input devices
  - Pointing, speech UIs
- Errors
  - Recognition, diagnosis, and recovery
- Optimizing execution sequences
- Discussion of “Design and Prototype” phase of group project

Action Execution

- Physical actions of dragging, clicking, typing, etc.
  - design goal is to minimize
  - make things “hang together” as movements, not just (abstractly) as plans
- Crucial role of *input device*—analogous to icons, etc.: different devices have different affordances:
  - keyboard?
  - mouse?
  - joystick?
  - trackball?
  - data glove?
<table>
<thead>
<tr>
<th>Device</th>
<th>Input Characteristics</th>
<th>Sample Applications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Button</td>
<td>Simple discrete input</td>
<td>Command execution or attribute specification</td>
</tr>
<tr>
<td>Keyboard</td>
<td>Spatial array, small finger movement, allows combination keypresses, discrete</td>
<td>Open-ended continuous symbolic input</td>
</tr>
<tr>
<td>Mouse</td>
<td>Grasped with hand, one or more buttons, large arm movement, analog</td>
<td>Pointing and selecting in a 2D space</td>
</tr>
<tr>
<td>Trackball</td>
<td>Grasped and rolled with hand, constrained movement in horizontal plane, one or more buttons, analog</td>
<td>Panning (rolling over) large maps or other 2D surfaces</td>
</tr>
<tr>
<td>Joystick</td>
<td>Grasped with hand, pushed or twisted, one or more buttons, constrained movement in three dimensions, analog</td>
<td>Setting direction of movement in virtual space, continuous zooming</td>
</tr>
<tr>
<td>Data glove</td>
<td>Tracking of finger and hand position in three dimensions</td>
<td>Grabbing and positioning objects in virtual space</td>
</tr>
</tbody>
</table>

What makes a pointing device “good”?

- *Fitts Law*: time to select target is a regular function of distance and size of the target
  - (but of course not all targets can be big and large :-)

- Articulatory directness
  - Jumping the Gulf of Execution
  - Directness of mapping from task semantics to device operation actions (e.g., twist to rotate)

- Feedback
  - Continuous feedback (proprioceptive (positional) and kinesthetic (dynamic))
  - Target acquisition feedback
Speech Input and Output

- Speech I/O inherently linear, relatively slow
  - trades off with familiarity, naturalness
  - may address with restricted vocabularies—commands
- Speech recognition accuracy still limited
  - depends on speaker (even mood of speaker), amount of pre-training
- Synthetic speech output quality also limited
  - biggest challenge is *prosody* (intonation contours)
  - many systems use digitized natural speech snippets
  - BUT useful for alerts, warnings (*why?*)
- Biggest benefit: parallel processing, multi-modal
  - also critical for hands-busy, heads-up tasks

Designing for Errors

- Key point: people always make errors
  - “Read everything before doing anything!”
  - *Why do people make errors?*
- Carefully analyze physically challenging actions
  - Some errors episodes are useful
  - Some are just annoying
  - Some are disastrous
- Overlearned procedures (e.g., from other systems) lead to *intrusions* (*slips*, not *mistakes*)
  - most common form is typos, transposition of letters
  - e.g., hitting delete before I get the text selected
  - e.g., making a menu selection before menu pops-up
Slips versus Mistakes

<table>
<thead>
<tr>
<th>Type of Error</th>
<th>Example Situation</th>
<th>Design Approach</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mistake: asking for non-existent function or object</td>
<td>Mistyping the name of a command so that its function can not be executed</td>
<td>Represent (e.g., in lists, icons) what is available</td>
</tr>
<tr>
<td>Mistake: over-generalizing an earlier experience</td>
<td>In a listserv, using “reply” when intending to reply only to the sender of a message</td>
<td>Present through training or documentation a more complete set of examples</td>
</tr>
<tr>
<td>Slip: doing something that is appropriate, but not in current mode</td>
<td>Trying to input text into a document while the Font dialog box is open</td>
<td>Minimize modes and when necessary mark well with status and feedback cues</td>
</tr>
<tr>
<td>Slip: making a request that is interpreted as something else</td>
<td>Using keyboard short-cut to turn off underline before adding space (in PowerPoint reverses existing underline)</td>
<td>Improve consistency of low-level controls within and across applications</td>
</tr>
<tr>
<td>Slip: completing an automated (but inappropriate) action</td>
<td>Deleting a text selection before the selection has not been correctly specified</td>
<td>Predict locus of such errors and increase the amount of feedback (or alerts) provided</td>
</tr>
</tbody>
</table>

Supporting Error Management

- Disabling inappropriate commands
  - graying out cut and copy when no text range is selected
- Blocking inappropriate commands
  - cut/copy with no currently open file => nil
  - cut/copy with no text range selected => BEEP
- Confirmation prompt
  - Do you really want to reformat your disk?
- Undo!
  - reversibility: (oops!) Back in a Web browser
  - predicting, supporting right level and depth of undo
  - what are the issues here?
Optimizing Execution Sequences

- Feedback and good defaults are essential!
  - especially in long, costly, or tedious transactions
- Consider implications of longterm use
  - focus on actions for frequent choices, fast-paths
  - BUT, be careful to note when you are
    → violating overall consistency, or favoring one task at expense of other important or common tasks
- Customization: users define their own sequences
  - e.g., mapping commands to key combinations
    (must have a rightward delete)
  - can be critical when supporting users with special needs

New Zealand research topic:
how to help end-users create macros to ease the tedium of standard (slow) GUI techniques
Project Phase 2: Design & Prototype

- Metaphor and technology exploration (brainstorm)
- Design four scenarios that **transform** problem scenarios
  - one scenario shown in all four phases (problem, activity, information, interaction)
  - three others shown as problem-interaction design pairs.
- Design claims: at least six, 1-2 from each scenario
- Prototype screenshots, demo scripts
  - Beth will walkthrough
  - *May schedule live demonstration with Beth*
## Example: Metaphors for VSF

<table>
<thead>
<tr>
<th>Phase</th>
<th>Design Ideas</th>
</tr>
</thead>
<tbody>
<tr>
<td>Activity Design: <em>Cocktail party</em></td>
<td>Informal discussions among visitors, as they move from group to group</td>
</tr>
<tr>
<td>Information Design: <em>Documentary</em></td>
<td>Movie or animated sequences of screens and audio; timeline visualization</td>
</tr>
<tr>
<td>Interaction Design: <em>Public lecture</em></td>
<td>Constant stream of visual/auditory output; visitor relatively passive, may be writing</td>
</tr>
</tbody>
</table>

(note: OK to use same metaphor, but “ideas” must be clearly appropriate for the different phases)

## Example: Technology for VSF

<table>
<thead>
<tr>
<th>Phase</th>
<th>Design Ideas</th>
</tr>
</thead>
<tbody>
<tr>
<td>Activity Design: <em>Threaded discussion</em></td>
<td>Visitors view exhibits, post comments by topic; students/others may then reply</td>
</tr>
<tr>
<td>Information Design: <em>MOOsburg chat</em></td>
<td>Sequential list of text messages; each is identified with name, perhaps color</td>
</tr>
<tr>
<td>Interaction Design: <em>MOOsburg map</em></td>
<td>Click on a location to go there; pan to see more map; annotate or ‘mark up’ map</td>
</tr>
</tbody>
</table>

(note: OK to use same technology, but “ideas” must be clearly appropriate for the different phases)
Document One Scenario Transformation

- Problem scenario: Mr. King coaches Sally (p. 68-69)
  - describes current problem situation (you already did these)

- Activity scenario: Mr. King coaches Sally (p. 97)
  - makes “key move” to design world & envisionment, but still just emphasizing the activity

- Information scenario: Mr. King coaches Sally (p. 145)
  - adds just the information/presentation details, along with user’s anticipation, reaction to what he/she sees

- Interaction scenario: Mr. King coaches Sally (p. 183)
  - all the UI details and user experience, i.e. the full “story”

Add Remaining Design Scenarios

- Just a pairing that shows the starting state
  - i.e., the problem scenario from Phase 1

- And the final state
  - i.e., the fully detailed user interaction scenario

- You may want/need to go through intermediate scenarios, but not required to document these
1-2 Claims for Each Scenario

<table>
<thead>
<tr>
<th>Scenario &amp; feature</th>
<th>Positive and negative consequences</th>
</tr>
</thead>
</table>
| Mr. King coaches: integrating products of common tools | + builds on exhibitors’ existing skills  
+ extends diversity of fair and its services 
-but may lead to confusion about what is and is not part of the fair |
| Mr. King coaches: nested components in layers | + simplifies browsing of main content 
- but viewers may never realize that the lower-level layers exist |

(repeated for two more scenarios; the claims may address activity, information, or interaction features)

Prototype & Screenshots

- Use whatever prototyping platform you want
  - HTML with some JavaScript probably enough for most
  - can also use Visual Basic, etc. if team has the skills
- Create three independent “scenario machines”
  - i.e. able to enact each scenario from fixed start state
  - no processing of data required, but do include simple error messages, e.g. if wrong link/button pressed
- Take and submit screenshots enough to document key states of each scenario
  - think of as a storyboard telling system side of story; include captions that connect to scenario narration
Demo Your Prototype

- Remote or Face-to-face walkthrough
  - After March 25, arrange by contacting Beth Yost
  - set up time and place convenient for all
- Plan for about 30 minutes, demo the scenario as well as any error messages
  - may also want to provide access after the demo
  - Beth will take notes as needed, then later fill out an evaluation check-sheet
  - counts as 40% of the grade for this phase