

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

1

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?

2

Common User Input Devices

Device	Input Characteristics	Sample Applications
Button	Simple discrete input	Command execution or attribute specification
Keyboard	Spatial array, small finger movement, allows combination keypresses, discrete	Open-ended continuous symbolic input
Mouse	Grasped with hand, one or more buttons, large arm movement, analog	Pointing and selecting in a 2D space
Trackball	Grasped and rolled with hand, constrained movement in horizontal plane, one or more buttons, analog	Panning (rolling over) large maps or other 2D surfaces
Joystick	Grasped with hand, pushed or twisted, one or more buttons, constrained movement in three dimensions, analog	Setting direction of movement in virtual space, continuous zooming
Data glove	Tracking of finger and hand position in three dimensions	Grabbing and positioning objects in virtual space

3

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

4

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

5

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

6

Slips versus Mistakes

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

7

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?

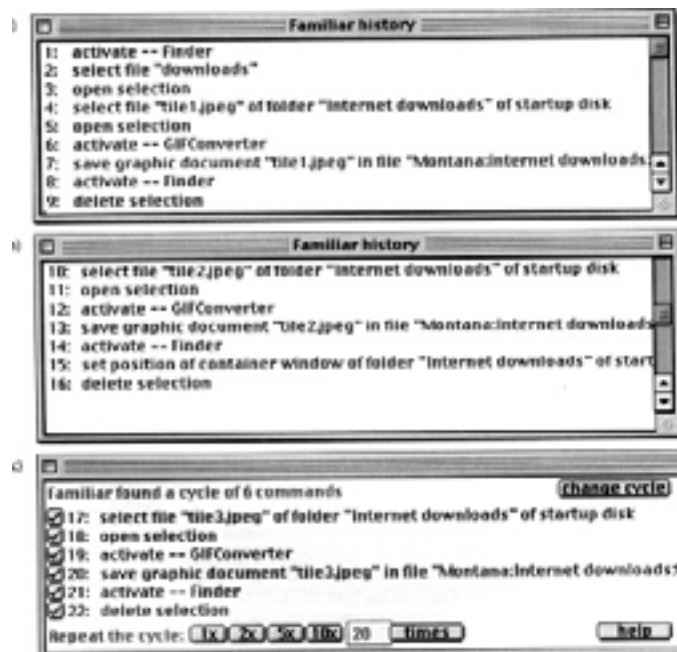
8

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

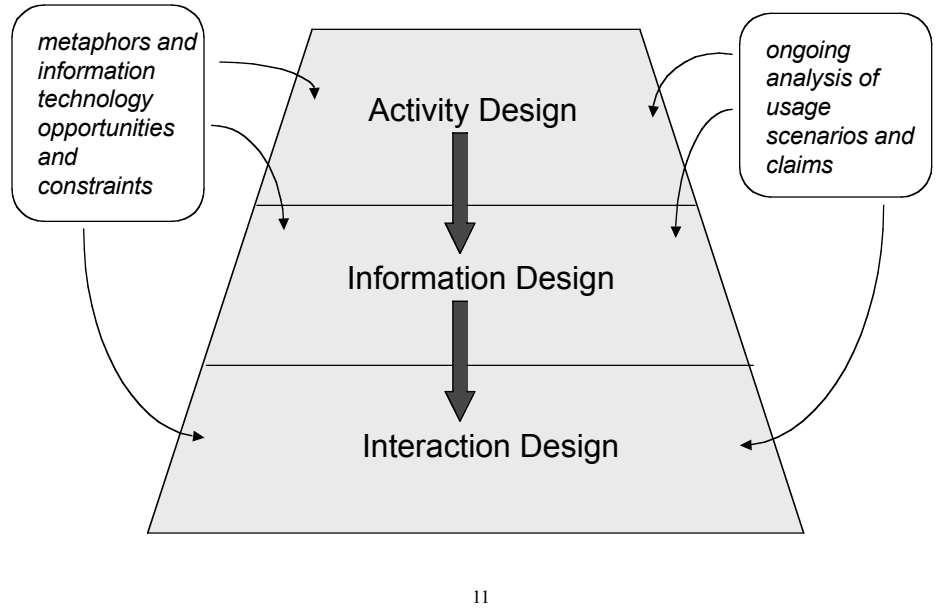
9

New Zealand research topic:
how to help end-users create macros to ease the tedium of standard (slow) GUI techniques



10

SBD: From Activity to Interaction



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

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

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

13

Example: Technology for VSF

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

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

14

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”

15

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

16

1-2 Claims for Each Scenario

Scenario & feature	Positive and negative consequences
<i>Mr. King coaches:</i> 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
<i>Mr. King coaches:</i> 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)

17

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

18

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