CS530: Rapid Prototyping

Ideas to windows

- How do we go from ideas to windows?
  - Prototyping...
    - rapid initial development, sketching & testing many designs to determine the best (few?) to continue with further design & implementation

- Why do we prototype?
  - get feedback on our design faster
  - experiment with alternative designs
  - fix problems before code is written
  - --> saves time! saves money!
  - keep the design centered on the user

Progressive refinement

- Applications are iteratively refined
  - from less to more detail
  - from coarse to fine granularity

- Designers create representations of app, web sites, etc. at multiple levels of detail

Levels of prototyping

- **Fidelity** refers to the level of detail
  - High fidelity
    - prototypes look like the final product
  - Low fidelity
    - artists renditions with many details missing

Thanks to James Landay for the foundation of these slides!
Hi-fi prototyping (➔ implementation)

- IDE = Integrated Development Environment
  - provides editor, compiler, debugger, etc.
  - also provides “builder” for GUIs
- Example: JBuilder
  - creating a new …?

Hi-fi prototyping (➔ implementation)

- specifying the main application frame
- when you click on “Finish”, what does this do?

Hi-fi prototyping (➔ implementation)

- creation of the frame class(es) / file(s)

Hi-fi prototyping (➔ implementation)

- refining the design
  (for a JButton)
Hi-fi prototyping (→ implementation)

- And continuing with implementation...
  - compiling and running the application
  - debugging
  - final production of the application
  (or applet, servlet, ...)

Issues with hi-fi prototyping

- Advantages
  - don’t need to produce layouts from scratch
  - initializes all the basic component code, you fill in the functionality later

- Disadvantages
  - must give specific instance of a general idea
    - e.g., exact widgets, fonts, alignments, colors
  - designers, evaluators focus on details instead of overall structure & functionality
  - takes longer!

Why use low-fidelity prototyping?

- Traditional methods take too long
  - sketches → prototype → evaluate → iterate
- Can simulate the prototype
  - sketches → evaluate → iterate
  - sketches act as prototypes
    - designer "plays computer"
    - other design team members observe & record
- Kindergarten implementation skills (!?)
  - allows non-programmers to participate

Paper sketches
Storyboards

• What are “storyboards”?
  • used in film & animation
  • provide a “script” of important events
  • leave out the details, concentrate on key interactions

“Luke & Leia coming toward camera. Behind them, Biker #3…”

“Elements: Luke Leia…”

Storyboards

• Storyboards for designing GUIs
  • create / pin-up lots of screens
  • specify user interaction by associating screens

“Wizard of Oz” technique

• Faking the interaction
  • name from (surprise) “The Wizard of Oz”
  • “the man behind the curtain”
• Much more important for hard-to-implement features
  • e.g., speech & handwriting recognition
• Example: How might you evaluate a natural language flight-reservation system using a Wizard-of-Oz technique?
  • materials/setup? input/output?
Low-fidelity prototyping

• Advantages of low-fi/paper prototyping
  • support brainstorming
  • do not require specification of details
  • designers feel comfortable sketching

• Drawbacks
  • do not evolve easily
  • lack support for “design memory”
  • force manual translation to electronic format
  • do not allow end-user interaction

• Can we do better?

Research Question of the Day

• Quickly sketch this…

Research Question of the Day

• Add behavior…

Research Question of the Day

• Transform it to this…
Research Question of the Day

- **SILK = Sketching Interfaces Like Krazy!**
- Designer sketches ideas rapidly with electronic pad and pen
  - SILK recognizes widgets
  - easy editing with gestures
- Designer or end-user tests interface
  - widgets behave
  - specify additional behavior visually
- Automatically transforms to a “finished” GUI

[Landay, CHI 1996]

Research Question of the Day

- **SILK: Screens**
  - screen = rough sketch of single screen state, including layout & components
  - features for handling screens
  - editing: use strokes to delete, move, group, ...
  - history: save, restore, annotate
  - widget inference / recognition

Research Question of the Day

- **SILK: Specifying behaviors**
  - behavior = association of a screen object with another screen
  - specifying a behavior
  - put relevant screens in the “storyboard”
  - draw arrow from object to screen

Research Question of the Day

- **SILK: Component recognition**
  - Infer / recognize components based on stroke-based gestures
  - scrollbar = long thin box + small inner box
  - Difficulties
    - differ’nt strokes for differ’nt folks
    - different stroke(s) may produce same drawing, but may be recognized very differently!
  - requires that system be trained for specific user
Research Question of the Day

- SILK: Component composition
  - When a component is recognized...
  - does the new component contain or is it contained by another component?
  - is the new component near another component?
  - is the new component in a sequence of components of the same type?