Designing Interactive Systems II

Computer Science Graduate Programme SS 2010

Prof. Dr. Jan Borchers
Media Computing Group
RWTH Aachen University

http://hci.rwth-aachen.de/dis2

Today

• Class syllabus
• About our group
• Device technology

Topics

• What makes a UI tick?
• Technical concepts, software paradigms and technologies behind HCI and user interface development

Administrivia

• New format: V3/Ü2
• Lecture: Wednesday, 9:00–12:00
• Lab: Monday, 15:30–17:00
• 6 credit points (8 with additional work if needed)
• Final grade:
  • 20% weekly assignments  25% midterm exam
  • 20% final project  35% final exam
• Requires MPO 2010
• Lecture recordings on iTunes U
**Class Syllabus**

- **Part I:** Key concepts of UI systems
  - Device technologies
  - Window System Architecture Model
- **Part II:** Comparing seminal window systems
  - Mac, X/KDE, Java/Swing, Windows, NeXT/OS X,…
  - Paradigms & problems, designing future UI systems
  - Overview of UI prototyping tools
- **Part III:** UIs Beyond The Desktop
  - Think beyond today's GUI desktop metaphor
  - UIs for Mobile, Physical Computing, Ubicomp, Multimedia

**The Lab**

- **Lab session on Mondays (15:30–17:00)**
  - Part I: Implementing your own simple reference window system
  - Part II: Development using several existing GUI toolkits (such as Java/Swing, Interface Builder)
  - Part III: Working with iPhone, Quartz Composer, Arduino, etc.
  - The Fab Lab:
    - Easy prototyping of
      - Embedded circuits
      - Physical components

**DIS 2 Team**

- Prof. Dr. Jan Borchers
- Dipl.-Inform. Moritz Wittenhagen
- Dipl.-Inform. Florian Heller

**How DIS I and DIS II Cover HCI**
### A Brief History of User Interfaces

**Iterative Design—the DIA Cycle**

1. **Design**
2. **Prototype/Implement**
3. **Analyze/Test/Evaluate**

**A Brief History of User Interfaces**

(Done in DIS I to understand the new interaction metaphors, reviewed here to understand the new programming paradigms)

- **Batch-processing**
  - No interactive capabilities
  - All user input specified in advance (punch cards, ...)
  - All system output collected at end of program run (printouts, ...)
  - Applications have no user interface component distinguishable from File I/O
  - Job Control Languages (example: IBM3090-JCL, anyone?): specify job and parameters

- **Time-sharing Systems**
  - Command-line based interaction with simple terminal
  - Shorter turnaround (per-line), but similar program structure
  - Applications read arguments from the command line, return results
  - Example: still visible in Unix commands

- **Menu-based systems**
  - Discover “Read & Select” over “Memorize & Type” advantage
  - Still text-based!
  - Example: VisiCalc
  - Applications have explicit UI component
  - But: choices are limited to a particular menu item at a time (hierarchical selection)
  - Application still “in control”
A Brief History of User Interfaces

- Graphical User Interface Systems
  - From character generator to bitmap display (Alto/Star/Lisa..)
  - Pointing devices in addition to keyboard
  - Event-based program structure
    - Most dramatic paradigm shift for application development
    - User is "in control"
    - Application only reacts to user (or system) events
    - Callback paradigm
  - Event handling
    - Initially application-explicit
    - Later system-implicit

Design Space of Input Devices

- Card, Mackinlay, Robertson 1991
- Goal: Understand input device design space
  - Insight in space, grouping, performance reasoning, new design ideas
  - Idea: Characterize input devices according to physical/mechanical/spatial properties
  - Morphological approach
  - device designs = points in parameterized design space
  - combine primitive moves and composition operators

Primitive Movements

- Input device maps physical world to application logic
- Input device := <M, In, S, R, Out, W>
  - Manipulation operator
  - Input domain
  - Device State
  - Resolution function In->Out
  - Output domain
  - Additional work properties

Radio Example
Composition

• Merge
  • Result = Cartesian product
  • E.g., mouse coordinates: $X \oplus Y = \{(x, y)\}$

• Layout
  • Spatial collocation
  • E.g., mouse $(x, y)$ & buttons
  • How different from merge?

• Connect
  • Chaining
  • E.g., mouse output & cursor
  • Virtual devices

Design Space (excerpt)

Complete space $\equiv \{\text{all possible combinations of primitives and composition operators}\}$

Mouse = one point!

In-Class Group Exercise: SpaceBall

• Place the SpaceBall into the design space
  • Ball mounted on a plate with 12 buttons
  • Detects precise amount of pushing and twisting in all directions without moving
  • Auto-zeroes physically

Is This Space Complete?

• No – it focuses on mechanical movement
  • Voice
  • Other senses (touch, smell, ...)
• But: Already proposes new devices
  • Put circles into the diagram and connect them
Testing Points

- Evaluate mappings according to
  - Expressiveness (conveys meaning exactly)
  - Effectiveness (felicity)
- Visual displays easily express unintended meanings
- For input devices, expressiveness suffers if |In| ≠ |Out|
  - |In| < |Out|: Cannot specify all legal values
  - |In| > |Out|: Can specify illegal values

Effectiveness

- How well can the intention be communicated?
- Various figures of merit possible
  - Performance-related
    - Device bandwidth (influences time to select target, ergonomics and cognitive load)
    - Precision
    - Error (% missed, final distance, statistical derivatives)
    - Learning time
    - Mounting / grasping time
  - Pragmatic
    - Device footprint, subjective preferences, cost,...

Example: Device Footprint

- Circle size := device footprint
  - Black: with 12” monitor
  - White: with 19” monitor
- What do we see?
  - Tablet, mouse expensive
  - Worse with larger displays
- But:
  - Mouse Acceleration alleviates this (model of C:D ratio?)
  - Higher resolution mice

What to do next

- Register in CAMPUS by Monday 12:00
- For next class, read:
- See the L2P course room for all materials