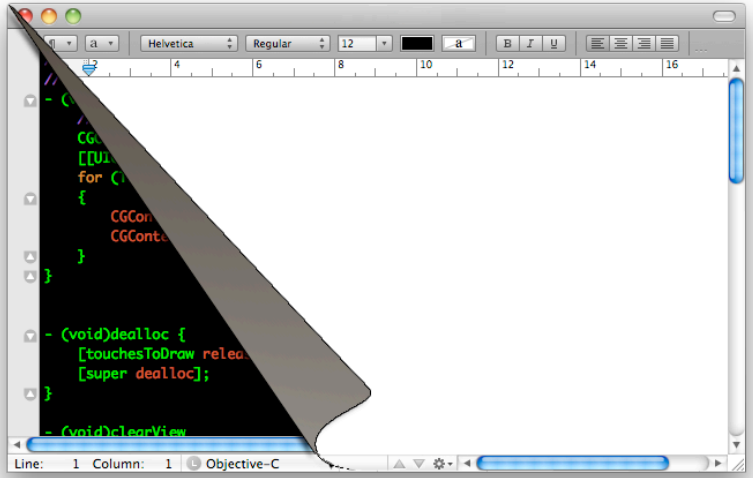


Designing Interactive Systems II

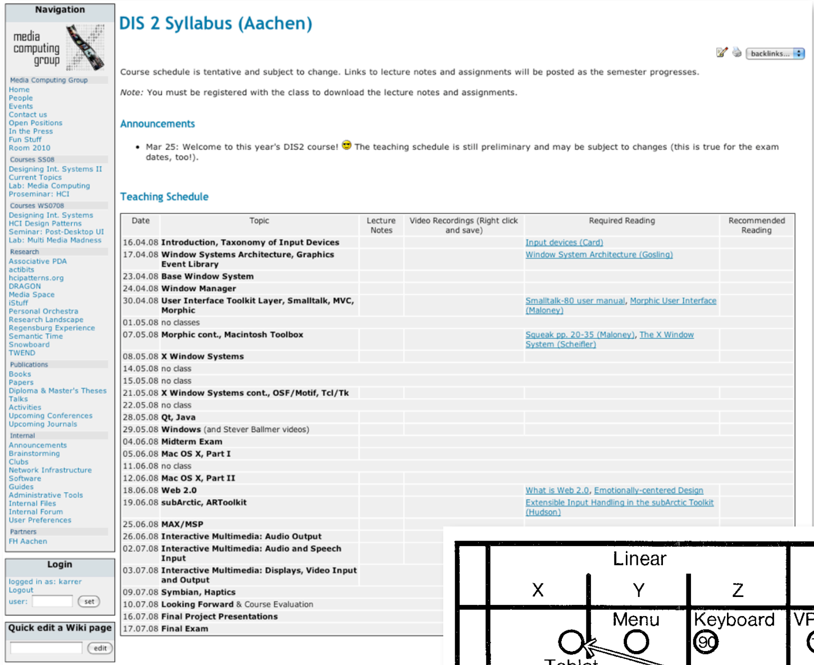
Computer Science Graduate Program SS 2011

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

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

Today

- Class syllabus
- About our group
- Device technology



DIS 2 Syllabus (Aachen)

Course schedule is tentative and subject to change. Links to lecture notes and assignments will be posted as the semester progresses.

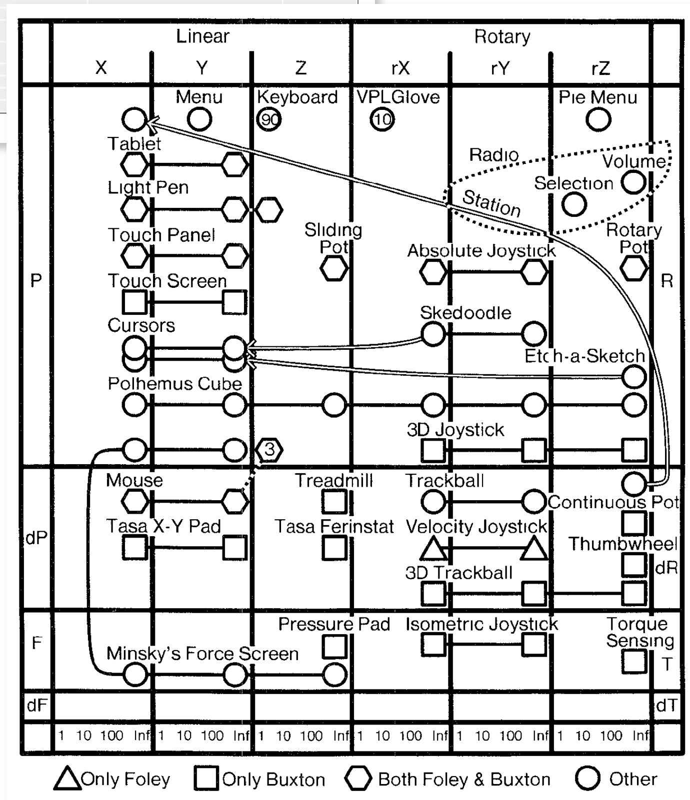
Note: You must be registered with the class to download the lecture notes and assignments.

Announcements

- Mar 25: Welcome to this year's DIS2 course! The teaching schedule is still preliminary and may be subject to changes (this is true for the exam dates, too!).

Teaching Schedule

Date	Topic	Lecture Notes	Video Recordings (Right click and save)	Required Reading	Recommended Reading
16.04.08	**Introduction, Taxonomy of Input Devices**			Input devices (Card)	
17.04.08	**Window Systems Architecture, Graphics Event Library**			Window System Architecture (Gosling)	
23.04.08	**Base Window System**				
24.04.08	**Window Manager**				
30.04.08	**User Interface Toolkit Layer, Smalltalk, MVC, Morphic**			Smalltalk-80 user manual, Morphic User Interface (Maloney)	
01.05.08	no classes				
07.05.08	**Morphic cont., Macintosh Toolbox**			Squeak pp. 20-35 (Maloney), The X Window System (Scheifler)	
08.05.08	**X Window Systems**				
14.05.08	no class				
15.05.08	no class				
21.05.08	**X Window Systems cont., OSF/Motif, Tcl/Tk**				
22.05.08	no class				
28.05.08	**Qt, Java**				
29.05.08	**Windows** (and Stever Ballmer videos)				
04.06.08	**Midterm Exam**				
05.06.08	**Mac OS X, Part I**				
11.06.08	no class				
12.06.08	**Mac OS X, Part II**				
18.06.08	**Web 2.0**			What is Web 2.0, Emotionally-centered Design	
19.06.08	**subArctic, ARToolkit**			Extensible Input Handling in the subArctic Toolkit (Hudson)	
25.06.08	**MAX/MSP**				
26.06.08	**Interactive Multimedia: Audio Output**				
02.07.08	**Interactive Multimedia: Audio and Speech Input**				
03.07.08	**Interactive Multimedia: Displays, Video Input and Output**				
09.07.08	**Symbian, Haptics**				
10.07.08	**Looking Forward** & Course Evaluation				
16.07.08	**Final Project Presentations**				
17.07.08	**Final Exam**				

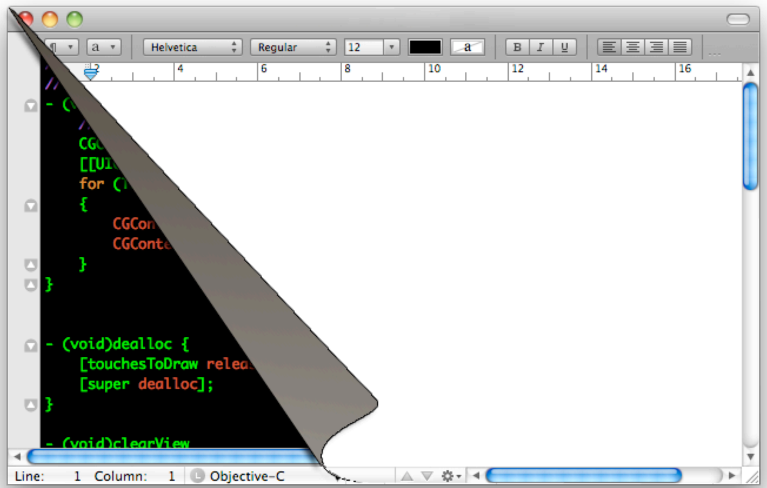


Administrivia

- Format: V3/Ü2
- Lecture: Wednesday, 9:00–12:00
- Lab: Monday, 14:15–15:45
- 6 credit points
- Final grade:
  - 50% midterm exam + 50% final exam
  - Use good exercise grades to gain points for the exams
- Lecture recordings on iTunes U

Topics

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

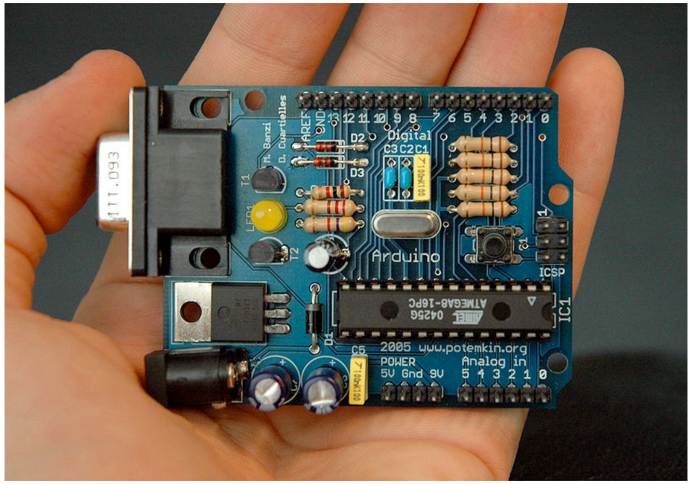


Class Syllabus

- 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 (14:15–15:45)
    - Part I: Implementing your own simple window system
    - Part II: Development using several existing GUI toolkits
    - Part III: Working with iPhone, Arduino, etc.
- The Fab Lab:
    - Easy prototyping of
        - Embedded circuits
        - Physical components



DIS 2 Team

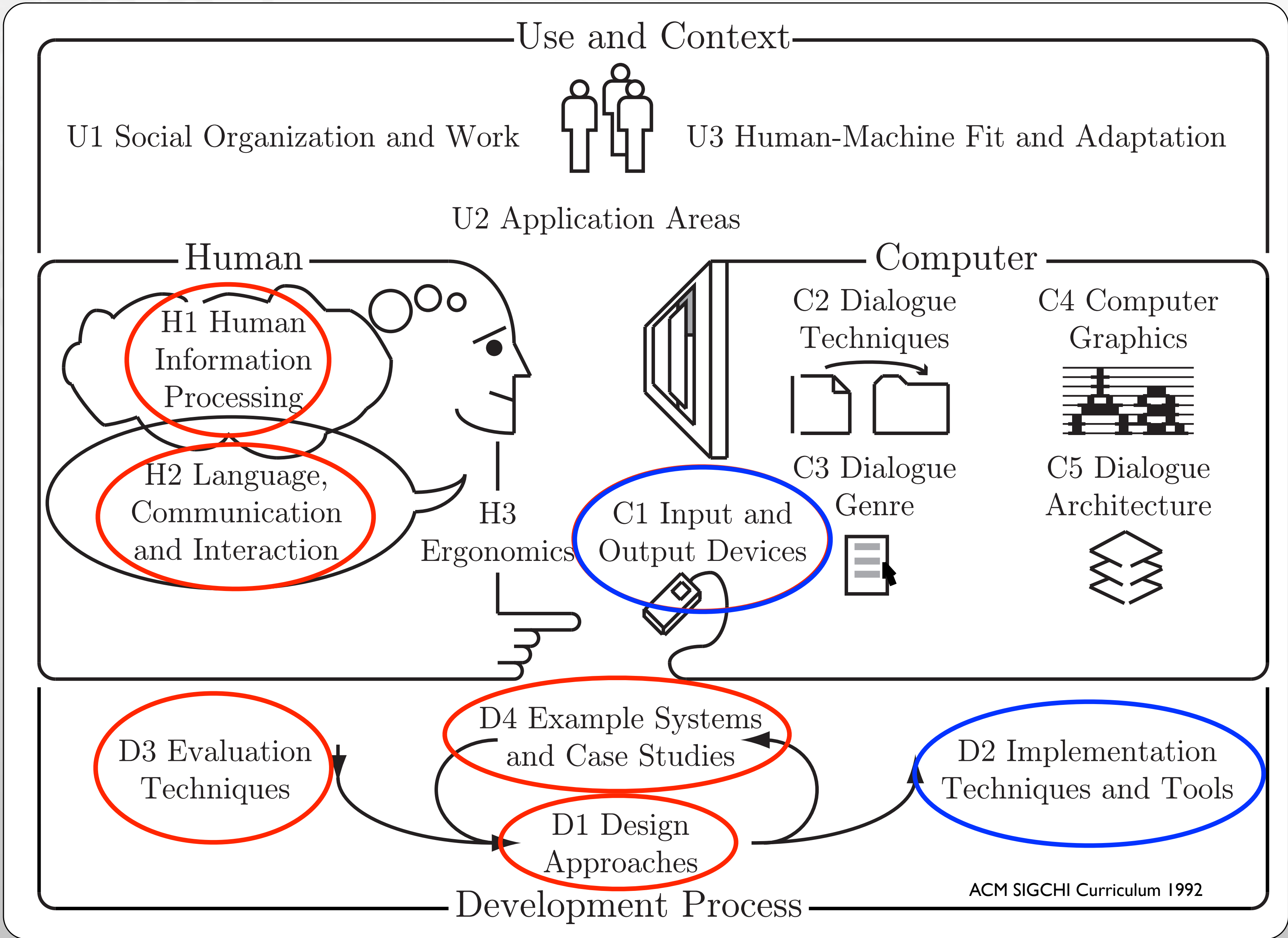
- Prof. Dr. Jan Borchers

- Dipl.-Inform. Moritz Wittenhagen

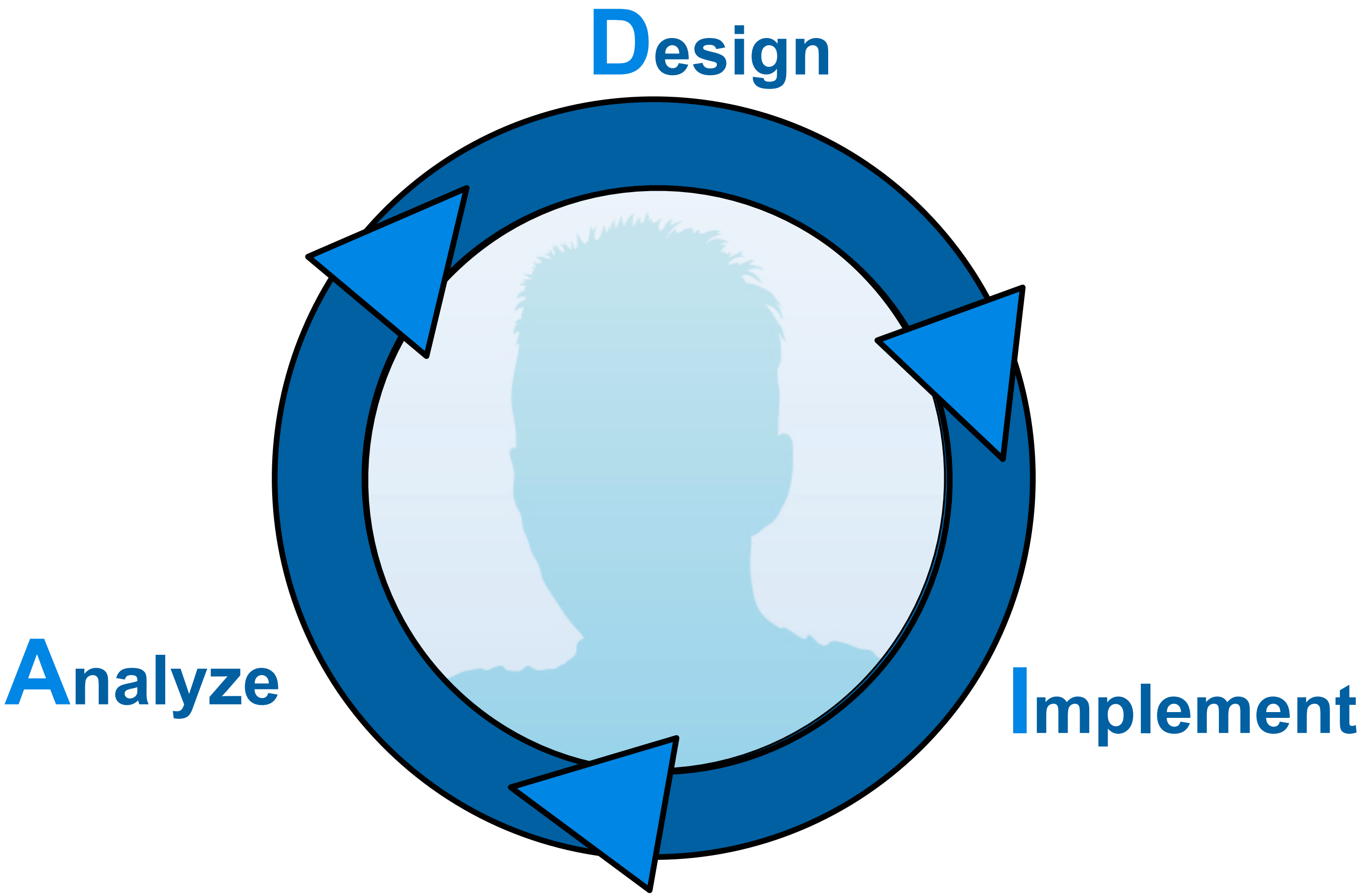
- Dipl.-Inform. Florian Heller

Current Research Projects

How DIS I and DIS II Cover HCI



Iterative Design—the DIA Cycle

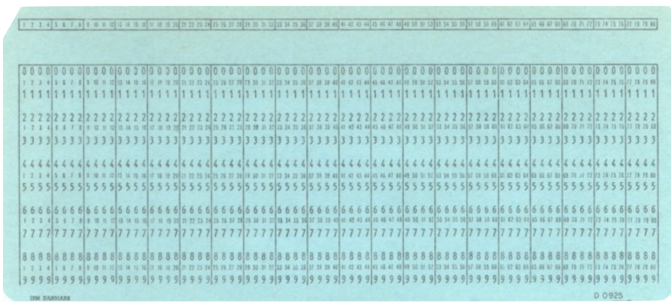


A Brief History of User Interfaces

(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?)

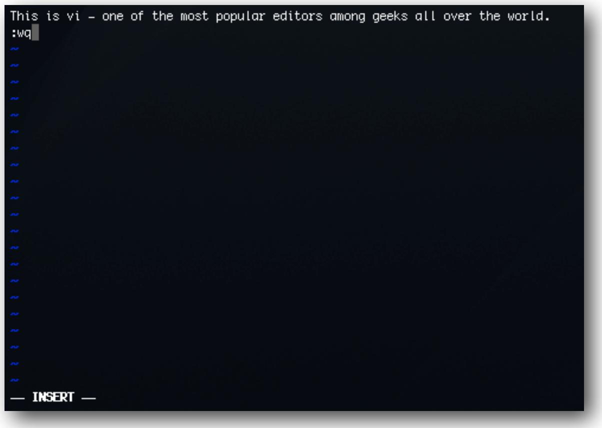
    specify job and parameters



A Brief History of User Interfaces

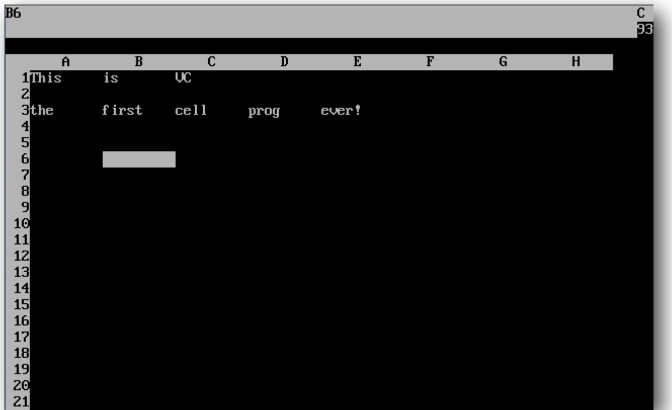
- **Time-sharing Systems**
  - Command-line based interaction with simple terminal
  - Shorter turnaround (per-line), but similar program structure
  - → Applications read arguments from the command line, return results
  - Example: still visible in Unix commands

- **Full-screen textual interfaces**
  - Shorter turnaround (per-character)
  - Interaction starts to feel "real-time" (e.g. vi)
  - → Applications receive UI input and react immediately in main "loop" (threading becomes important)



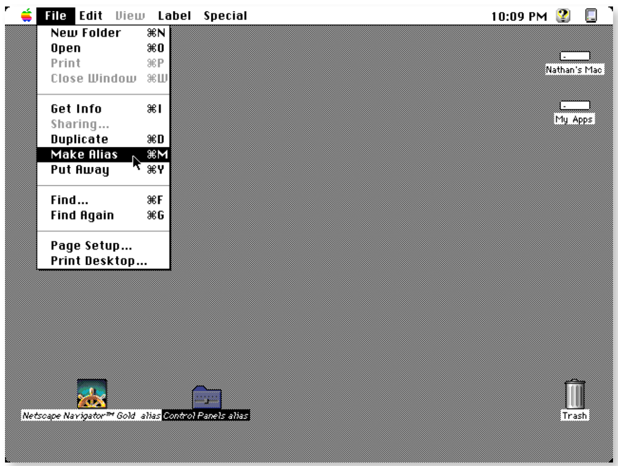
A Brief History of User Interfaces

- 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

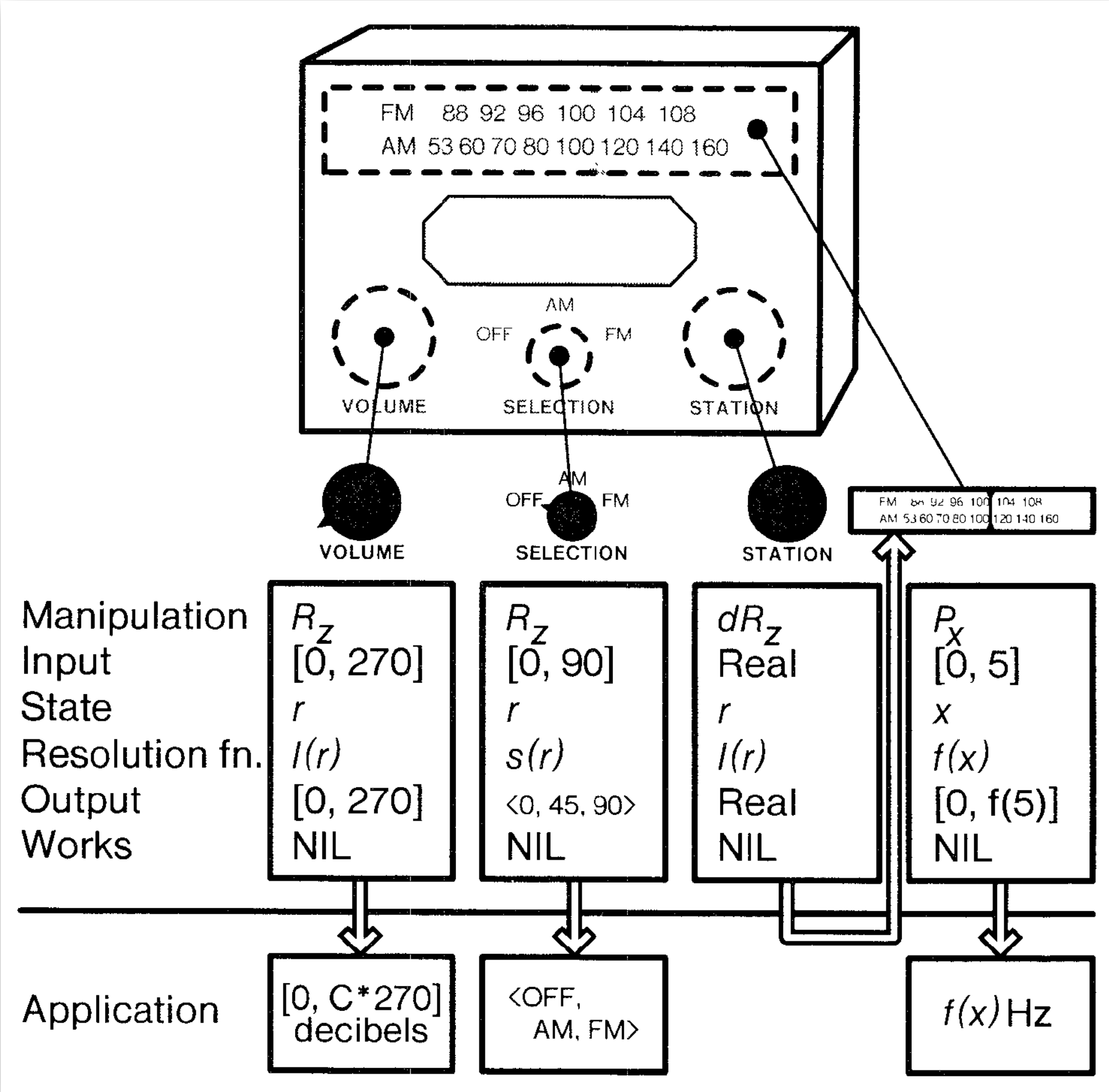
- Card, Mackinlay, Robertson 1991
- Goal: Understand input device design space
  - Insight in space, grouping, performance reasoning, new design ideas
- Idea: Characterize input devices according to physical/mechanical/spatial properties
- Morphological approach
  - Device designs = points in parameterized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

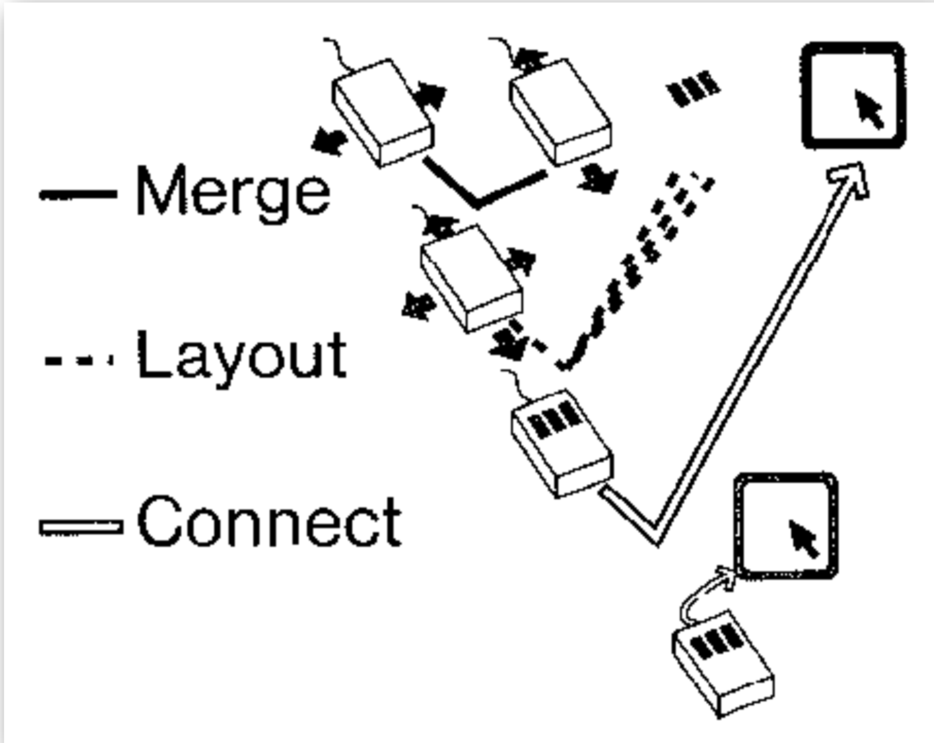
P, dP	R, dR
F, dF	T, dT

Radio Example

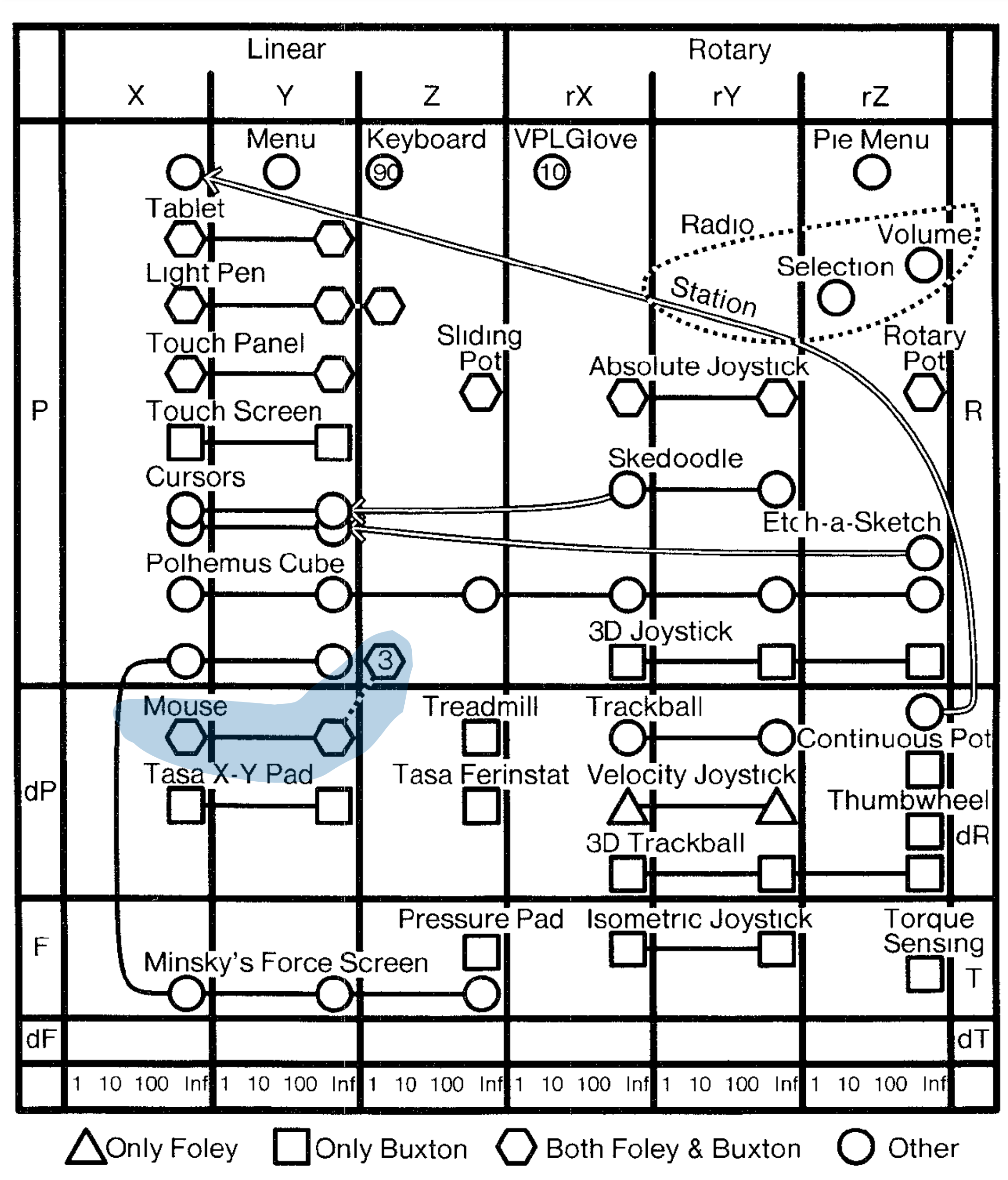


Composition

- 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 := {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

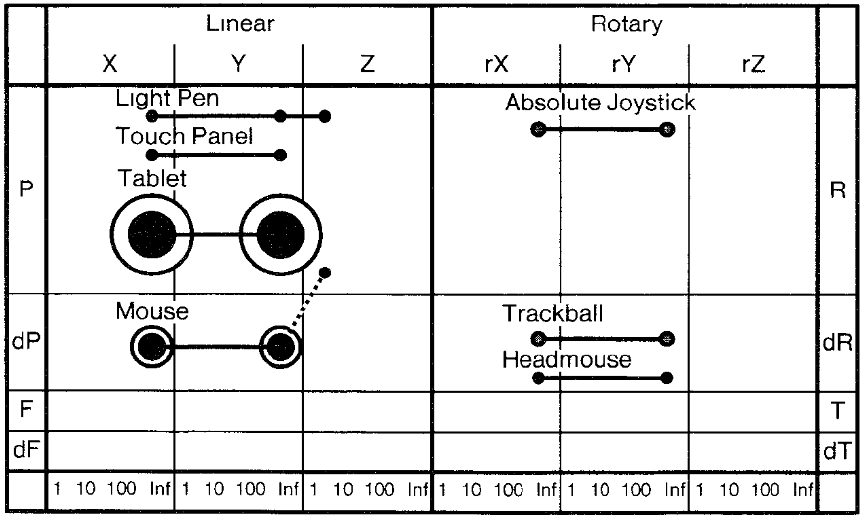
Testing Points

- Evaluate mappings according to
  - Expressiveness (conveys meaning exactly)
  - Effectiveness (felicity)
- Visual displays easily express unintended meanings
- For input devices, expressiveness suffers if $|In| \neq |Out|$
  - $|In| < |Out|$: 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,...

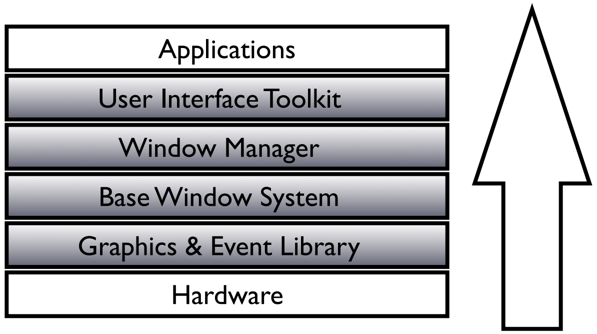
Example: Device Footprint



- 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

Now: Window Systems
Part I

- Window System Requirements
- 4-Layer Model
- Graphics and Event Library



Window Systems: Basic Tasks

- Basic window system tasks:
  - Input handling: Pass user input to appropriate application
  - Output handling: Visualize application output in windows
  - Window management: Manage and provide user controls for windows
  - *This is roughly what our Simple Reference Window System will be implementing*

Window Systems: Requirements

- Independent of hardware and operating system
- Legacy (text-based) software support (virt. terminals)
- No noticeable delays (few ms) for basic operations (edit text, move window); 5+ redraws/s for cursor
- Customizable look&feel for user preferences
- Applications doing input/output in parallel
- Small resource overhead per window, fast graphics
- Support for keyboard and graphical input device
- Optional: Distribution, 3-D graphics, gesture, audio,...

In-Class Exercise: Window Systems Criteria

- In groups of 2, brainstorm criteria that you would look at when judging a new window system
- We will compile the answers in class afterwards

Window Systems: Criteria

- Availability (platforms supported)
- Productivity (for application development)
- Parallelism
  - external: parallel user input for several applications possible
  - internal: applications as actual parallel processes
- Performance
  - Basic operations on main resources (window, screen, net), user input latency—up to 90% of processing power for UI
- Graphics model (RasterOp vs. vector)

Window Systems: Criteria

- Appearance (Look & Feel, exchangeable?)
- Extensibility of WS (in source code or at runtime)
- Adaptability (localization, customization)
  - At runtime; e.g., via User Interface Languages (UILs)
- Resource sharing (e.g., fonts)
- Distribution (of window system layers over network)
- API structure (procedural vs. OO)
- API comfort (number and complexity of supplied toolkit, support for new components)

Window Systems: Criteria

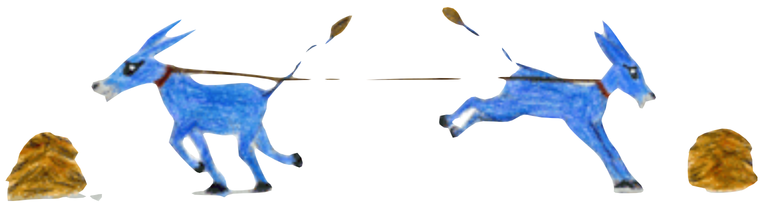
- Independence (of application and interaction logic inside programs written for the WS)
- IAC (inter-application communication support)
  - User-initiated, e.g., Cut&Paste

Technique	Selection	Clipboard	DDE	OLE
Duration	short	short	medium	long
Data types	special	special	special	any
Directed	yes	no	yes	no
Relation	1:1	m:1:n	1:1	m:n
Abstraction	low	low	medium	high

Window Systems: Conflict

- WS developer wants: elegant design, portability
- App developer wants: Simple but powerful API
- User wants: immediate usability+malleability for experts

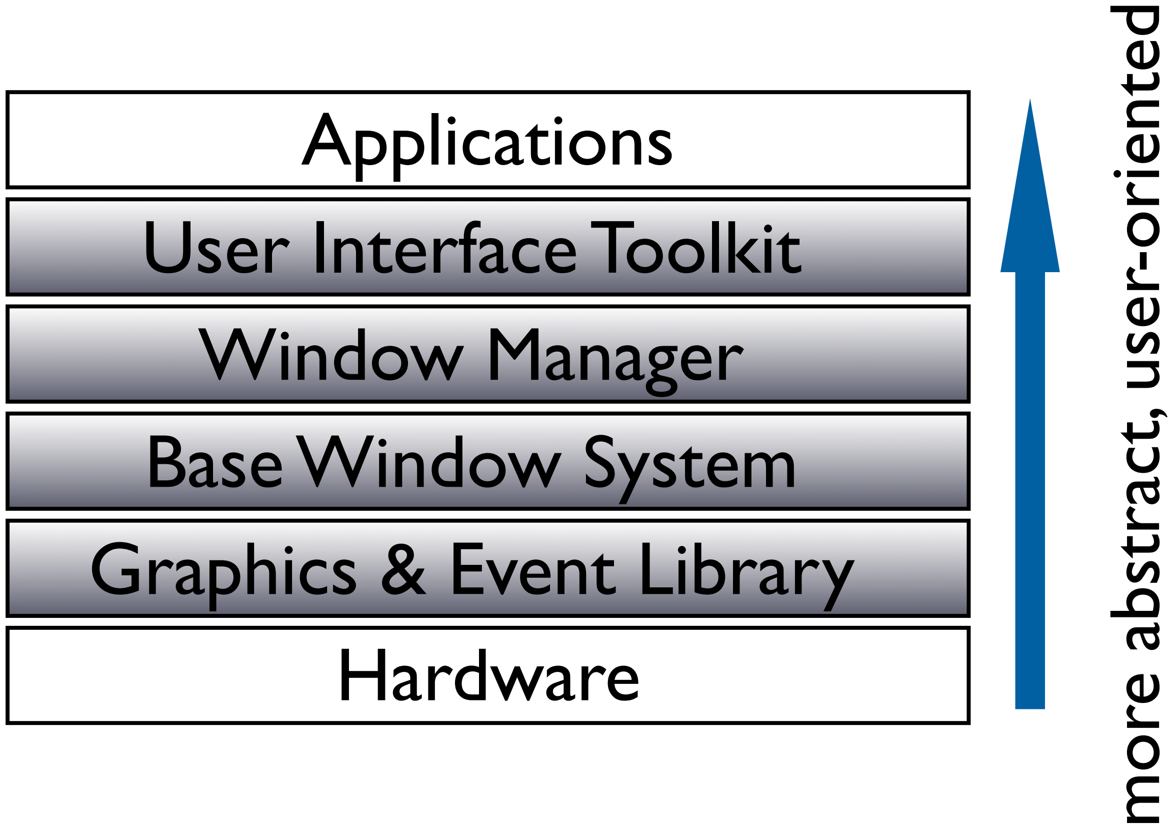
- Partially conflicting goals



- Architecture model shows if/how and where to solve
- Real systems show sample points in tradeoff space

The 4-Layer Model of Window System Architectures

- Layering of virtual machines
- Idealized system model
- Where is the OS?
- Where is the user?
    - Physical vs. abstract communication
    - See ISO/OSI model



The 4-Layer Model of Window System Architectures

- UI Toolkit (a.k.a. Construction Set)
    - Offers standard user interface objects (widgets)
- Window Manager
    - Implements user interface to window functions
- Base Window System
    - Provide logical abstractions from physical resources (e.g., windows, mouse actions)
- Graphics & Event Library (implements graphics model)
    - High-performance graphics output functions for apps, register user input actions, draw cursor

What to do next

- Register in CAMPUS
- For next lab(!), read:
  - Stuart K. Card, Jock D. Mackinlay and George G. Robertson: "A morphological analysis of the design space of input devices", ACM Transactions on Information Systems, 9(2), 99-122, 1991
  - Window System Architecture chapter from Gosling's NeWS book (James Gosling, David S. H. Rosenthal, and Michelle J. Arden, "The NeWS Book", Springer-Verlag, 1989, Chapter 3)
- See the L2P course room for all materials