Designing Interactive Systems 2

Lecture 1: Introduction, History, Design Space of Input Devices

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

hci.rwth-aachen.de/dis2

RWTH AACHEN UNIVERSITY

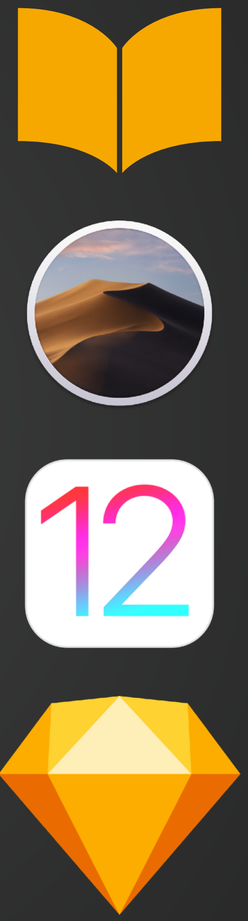
Class Syllabus

- Part 1
  Key Concepts

- Part 2
  Usage and Design of UI Toolkits and Design Systems

- Part 3
  UIs Beyond the Desktop

- Part 4
  Prototyping Process



Administrivia

- Format: V3/Ü2
- 6 Credit points
- Class times
  - **Lecture** on Wednesdays (8:30 — 11:00), Room 2222
  - **Lab** on Mondays (14:30 — 16:00), Room 2222

Team

Prof. Dr.
Jan Borchers

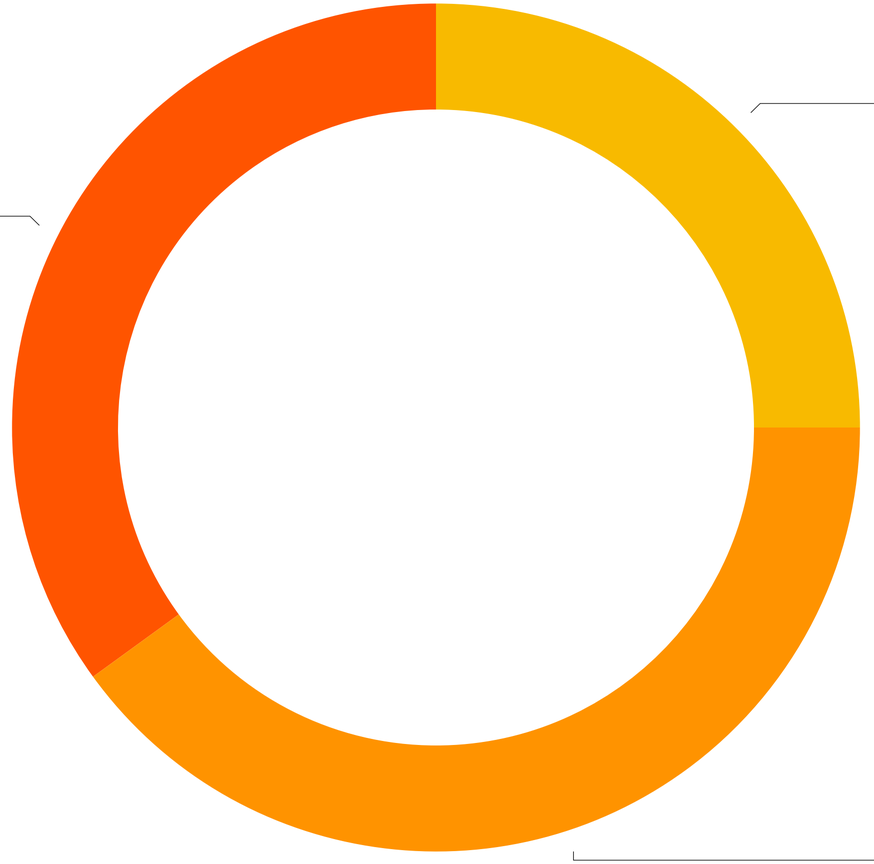
Sebastian
Hueber

hueber@cs.rwth-aachen.de
E-Mail Subject: [DIS2]

Anke
Brocker

brocker@cs.rwth-aachen.de
E-Mail Subject: [DIS2]

Your Final Grade



25%
Midterm Exam (60 min)
May, 17th

35%
Final Exam (60 min)
July, 18th
or August, 9th

40%
Weekly Assignments

Weekly Assignments

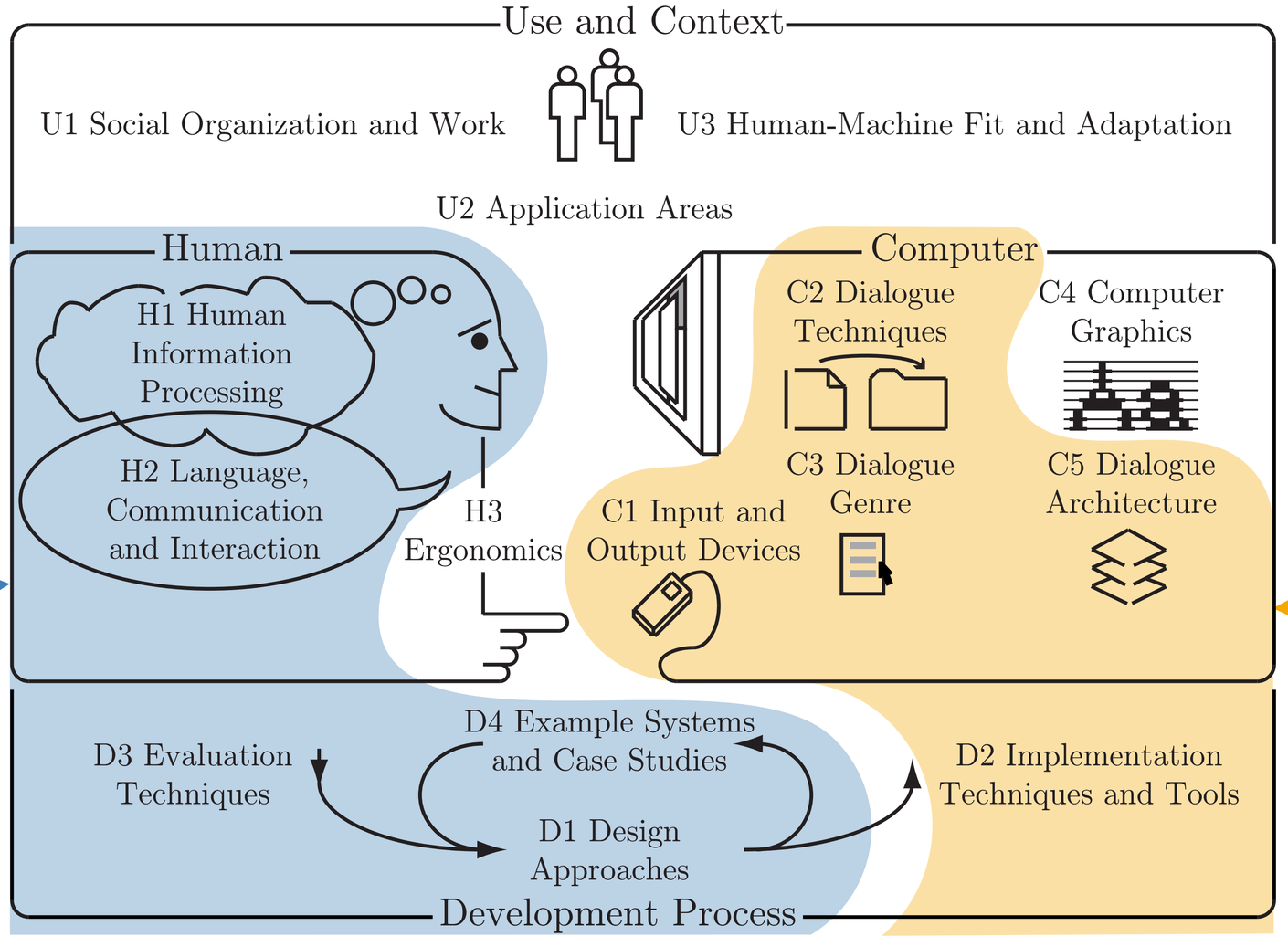
- We have a strict grading policy:
  - **Late submissions** will be graded 5.0 without feedback
  - **Team size** is 2 students (3 only by permission). If you hand in a solution without a team partner: 5.0 without feedback
  - If your code does **not compile**: 5.0 without feedback
- For some assignments you will need a **Mac**
  - No Mac? Visit http://www-rbi.informatik.rwth-aachen.de/Pool+Helpdesk/
- Submission via **Moodle**

Website

- All information about this course can be found online

- hci.rwth-aachen.de/dis2

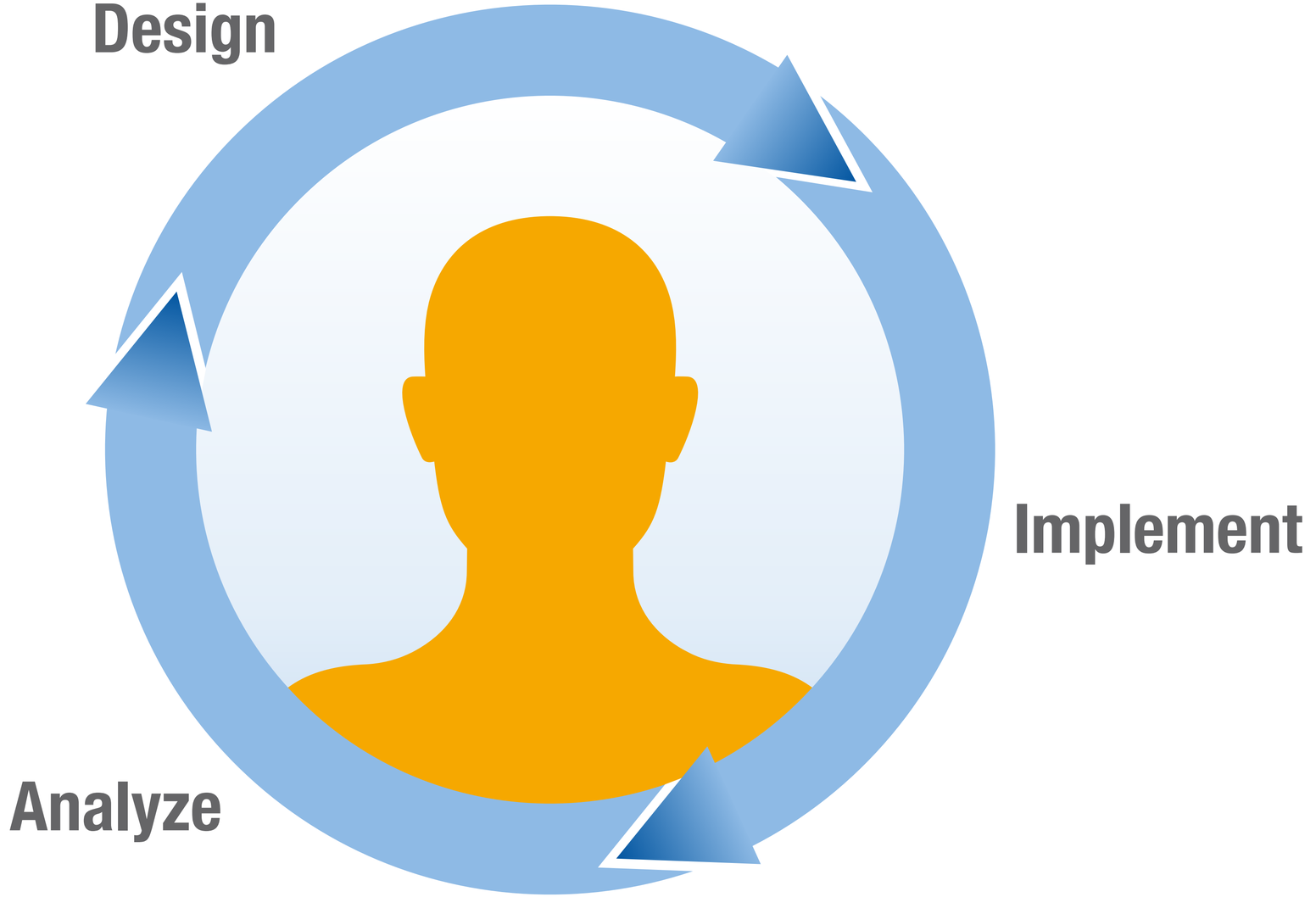
How DIS1 and DIS2 Cover HCI



DIS 1

DIS 2

DIA Cycle

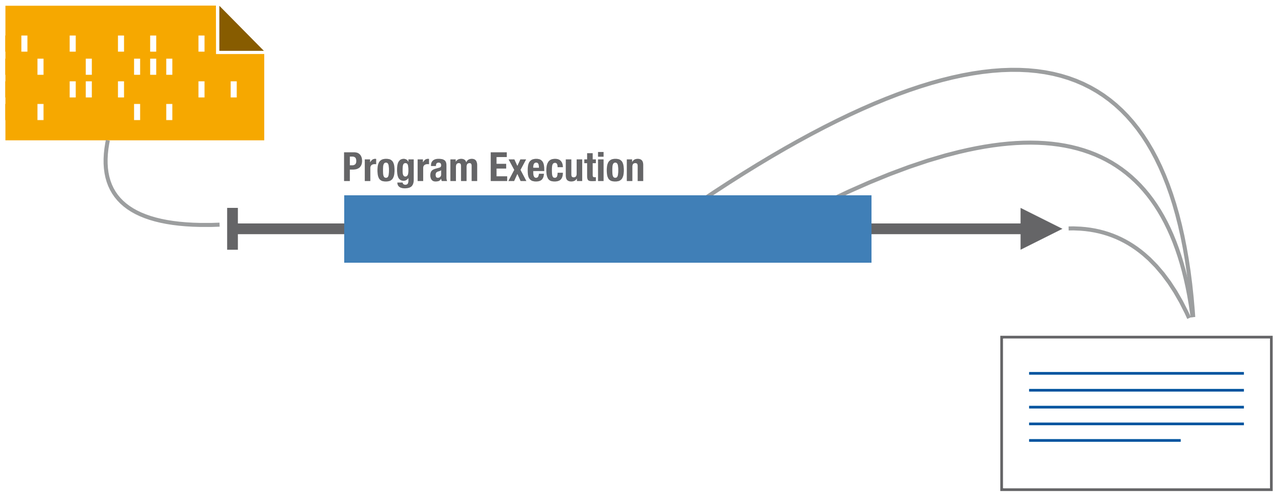


CHAPTER 1

History of User Interface Programming Paradigms

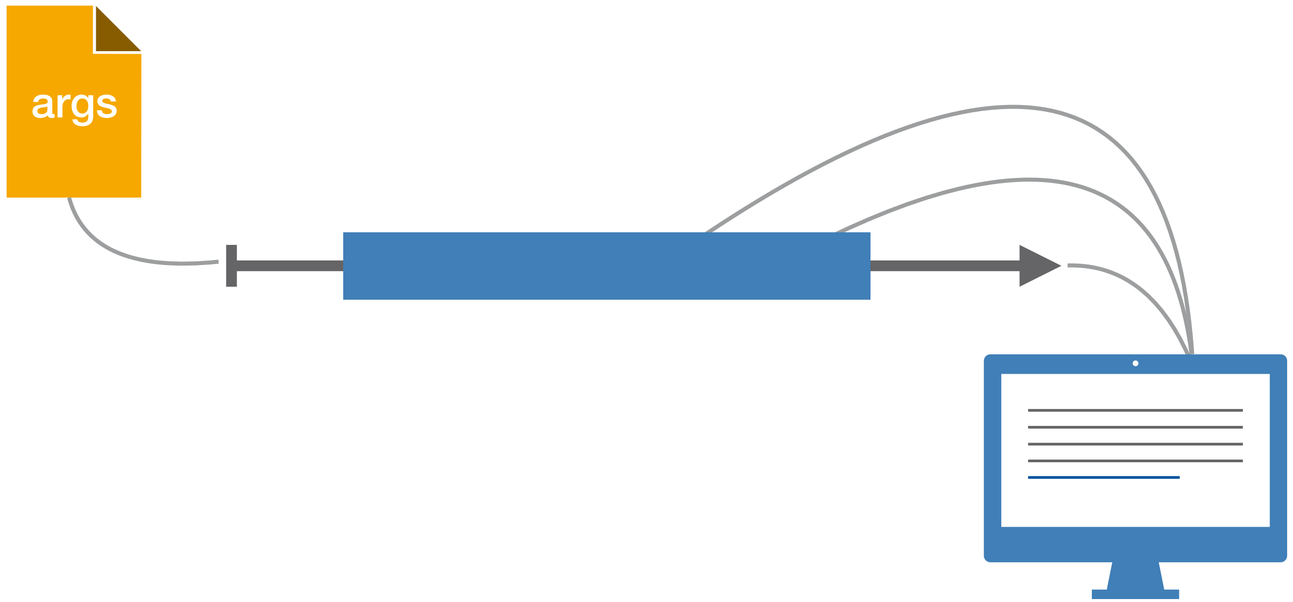
Batch Processing

- Prepare data on punch cards
- Wait for result as printout offline



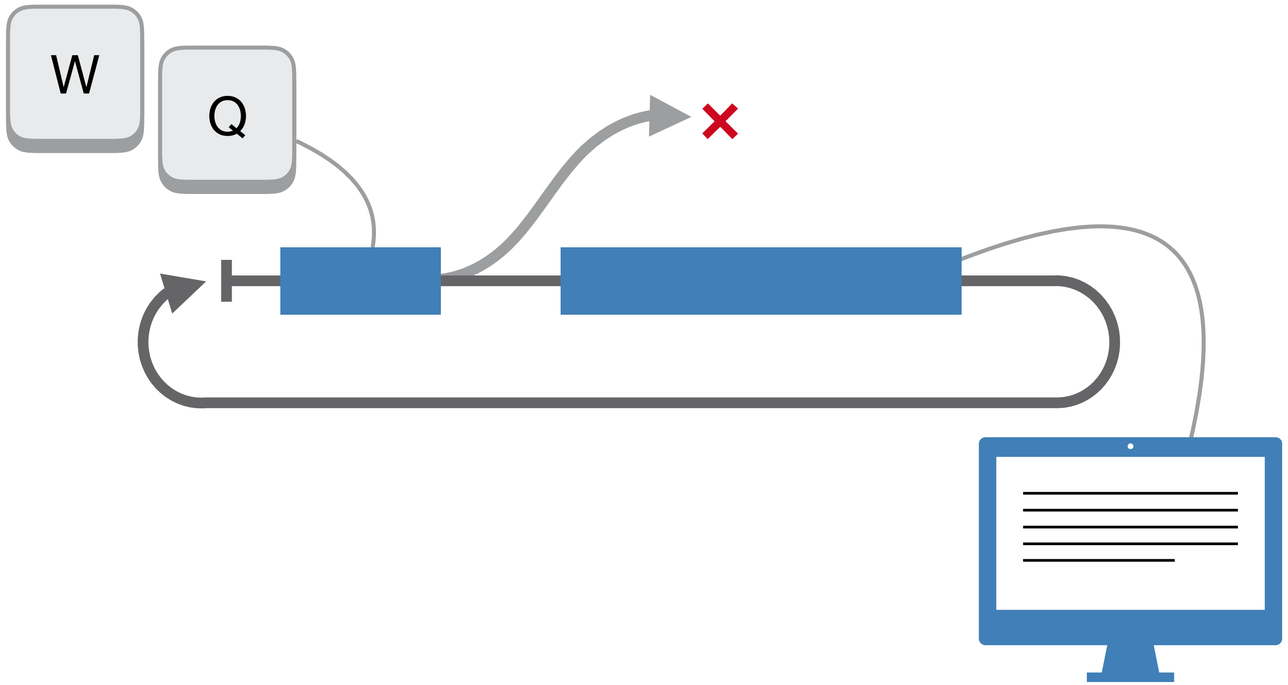
Time-sharing Systems

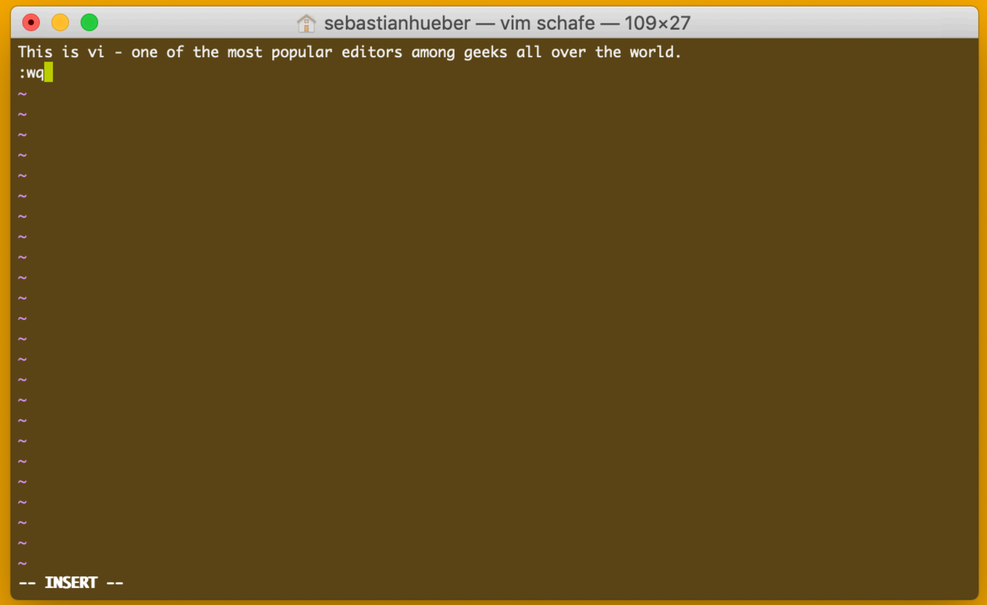
- Command-line based interaction
- Shorter turnaround (per-line)



Full-screen textual UIs

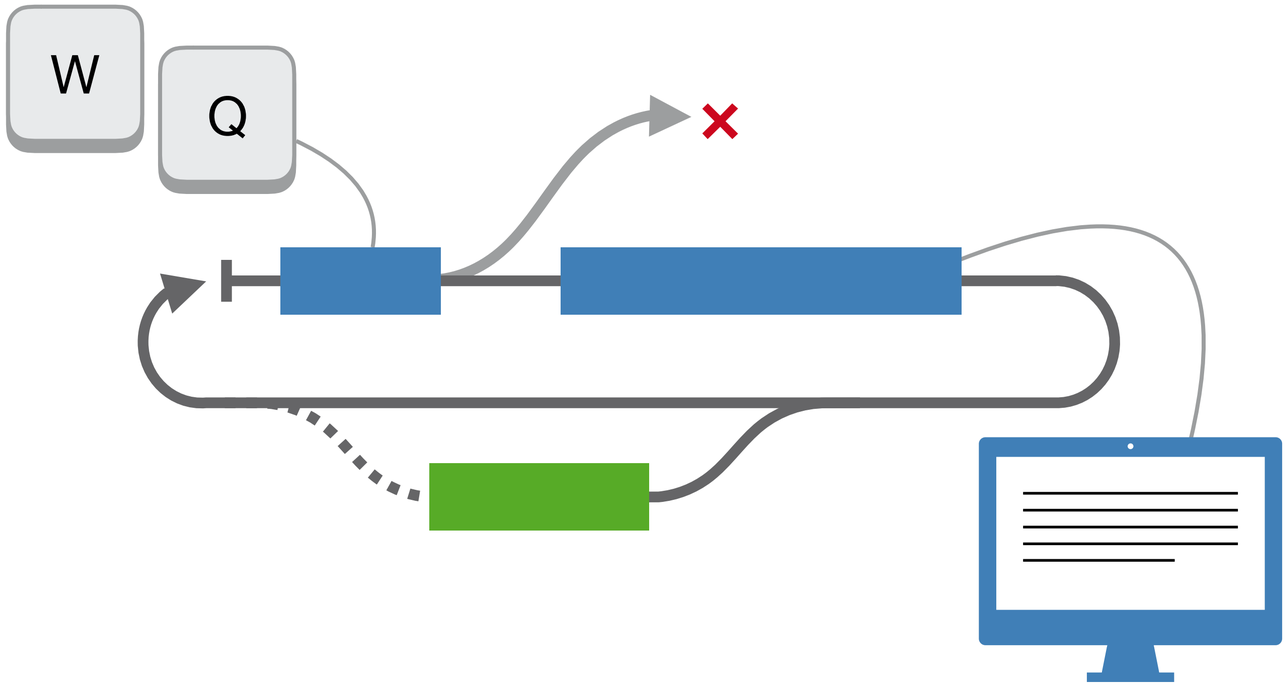
- Turnaround per character
- Interaction starts to feel “real-time”

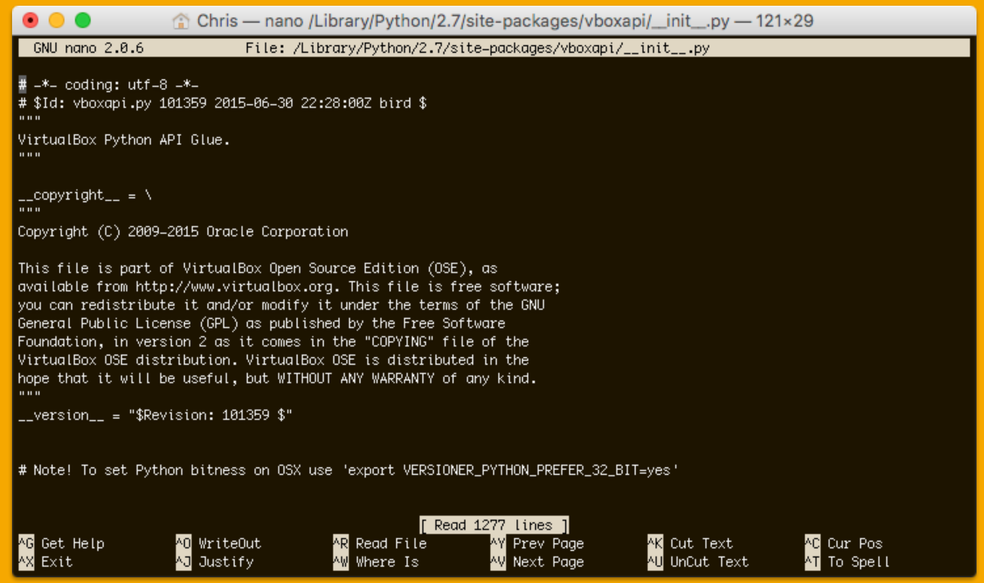




Menu-based Systems

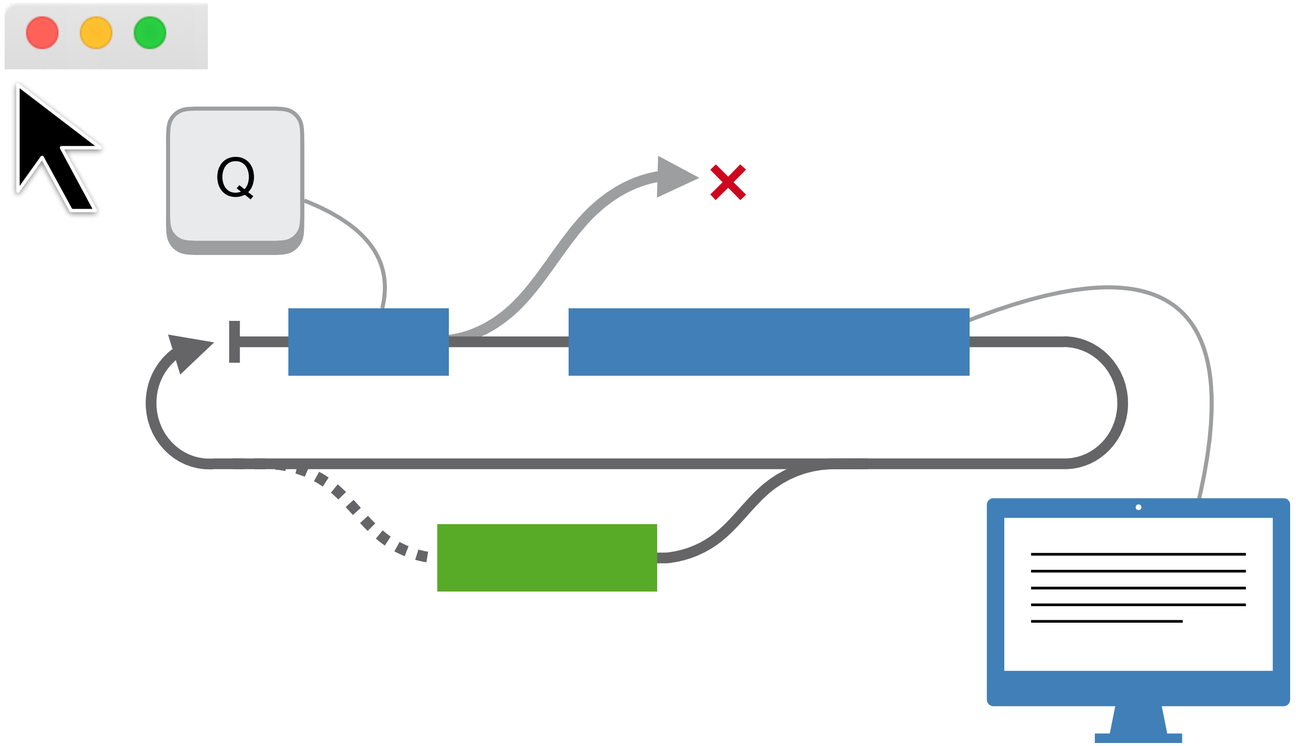
- Discover functionalities instead of memorizing them
- **Threading** becomes important





Graphical User Interface

- Event-based program structure
- Pointing devices in addition to keyboard



CHAPTER 2

Design Space of Input Devices

Design Space of Input Devices

- Card, Mackinlay, Robertson 1991
- Categorization of input devices according to physical, mechanical and spatial properties
- **Why?**
  - Compare input devices
  - Identify new input modalities

Movement Primitives

Manipulation operator

Device state

Output domain

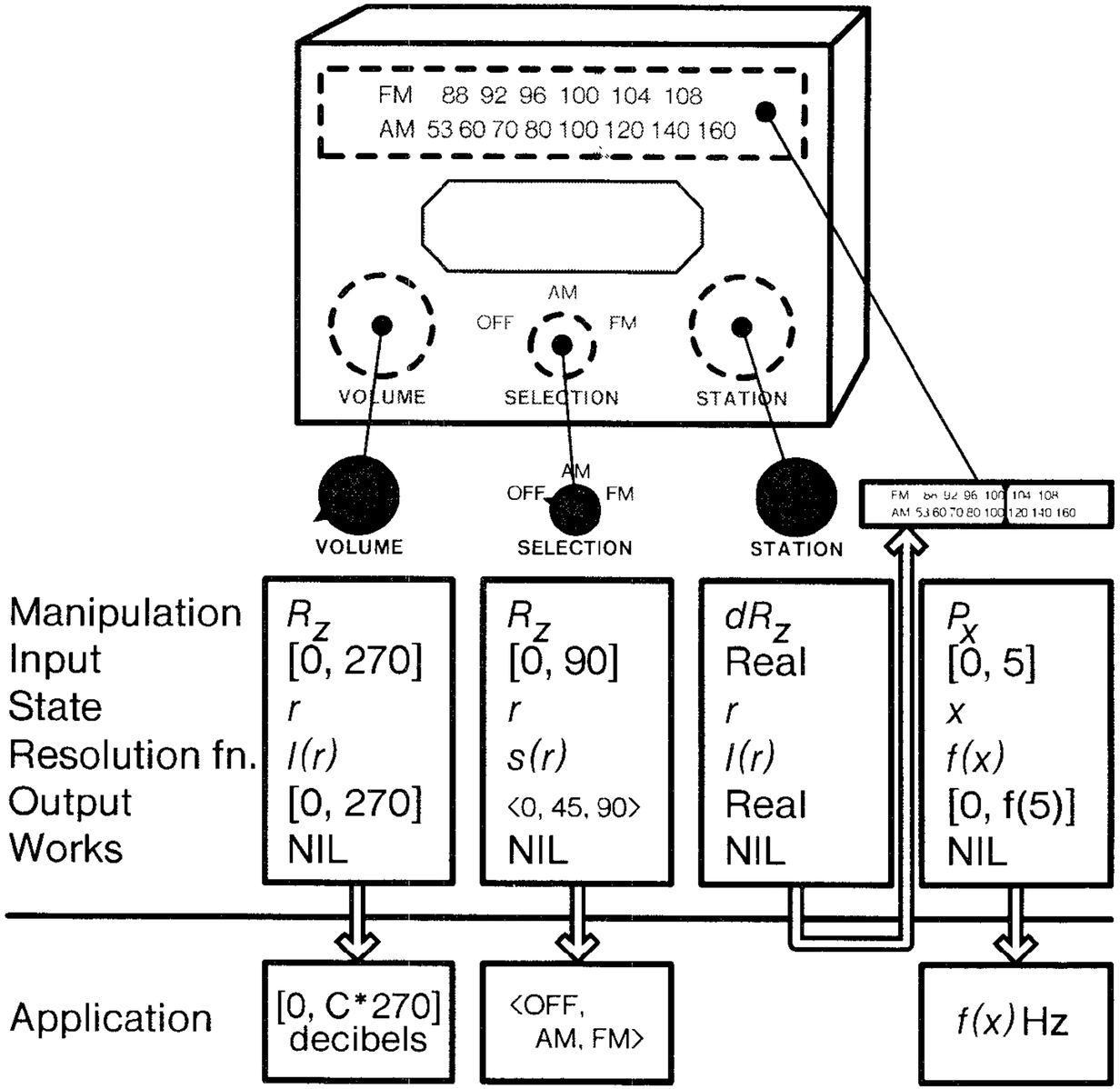
Input device := <M, In, S, R, Out, W>

Input domain

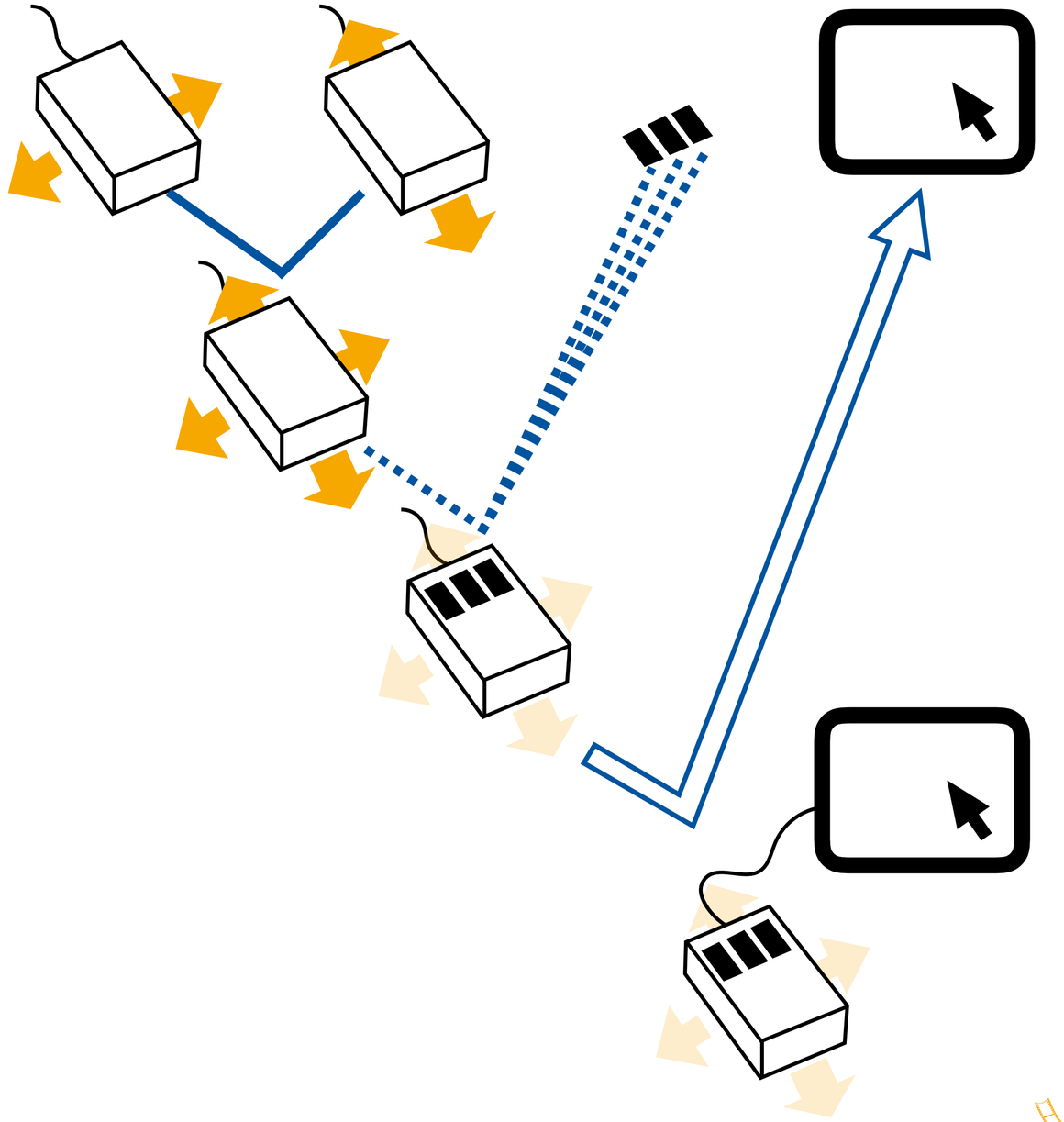
Resolution function

Work properties

Example



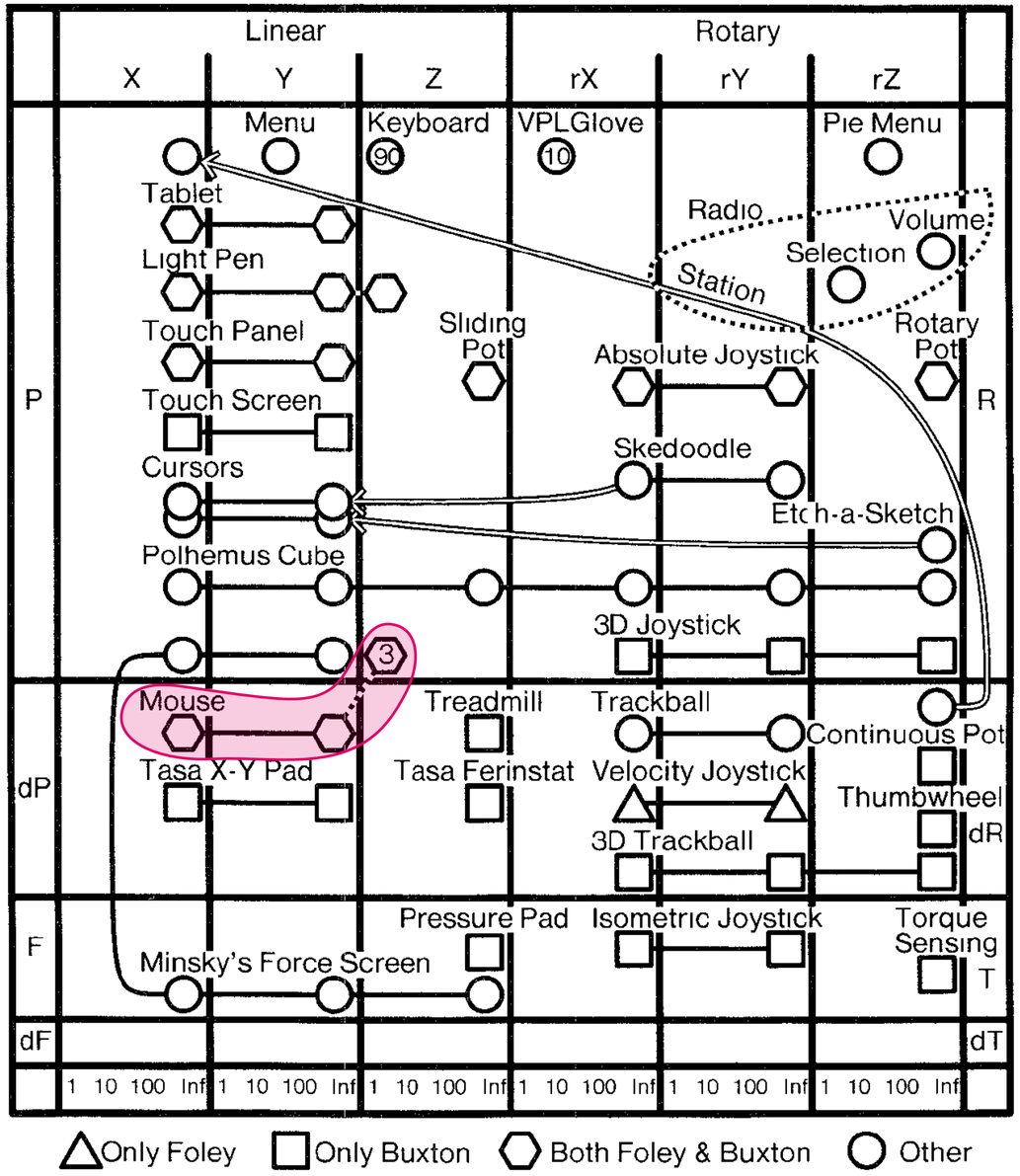
Compositions



Merge

Layout

Connect



In-Class Exercise

- Plot out the input capabilities of the Ferrari Racing Controller on the Card Design Space of Input Devices.

- The controller consists of a **steering wheel** with **8 buttons** and a **rotary switch** with 5 states, as well as **2 pedals**.

- Assume that the steering wheel can only have one full rotation.

In-Class Exercise

	Linear			Rotar			
	X	Y	Z	rX	rY	rZ	
P							R
dP							dR
F							T
dF							dT
	1 Inf	1 Inf	1 Inf	1 Inf	1 Inf	1 Inf	

8

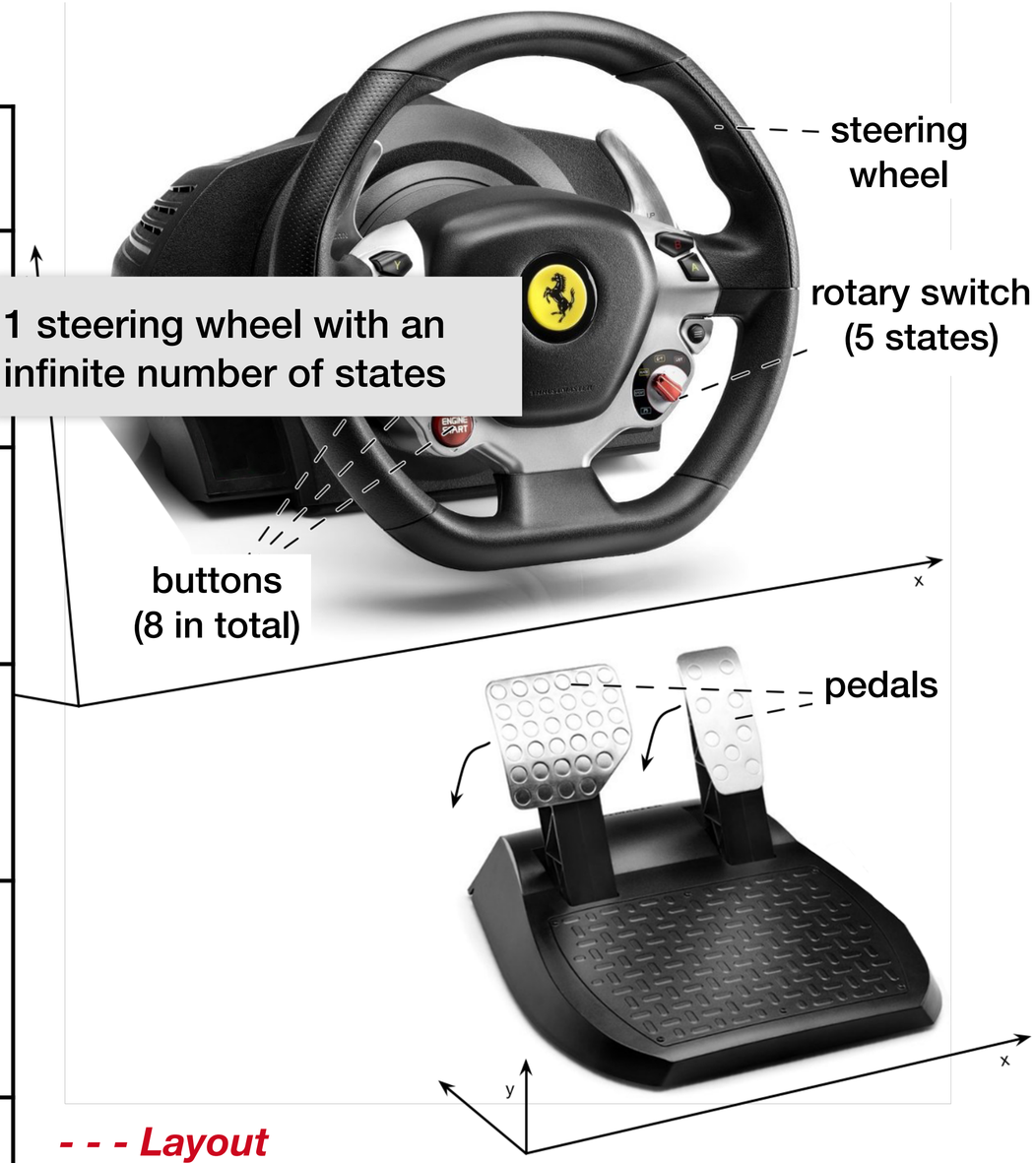
2

1 switch with 5 states

8 buttons with 2 states each

2 pedals with an infinite number of states each

1 steering wheel with an infinite number of states



- - - Layout

Is This Space Complete?

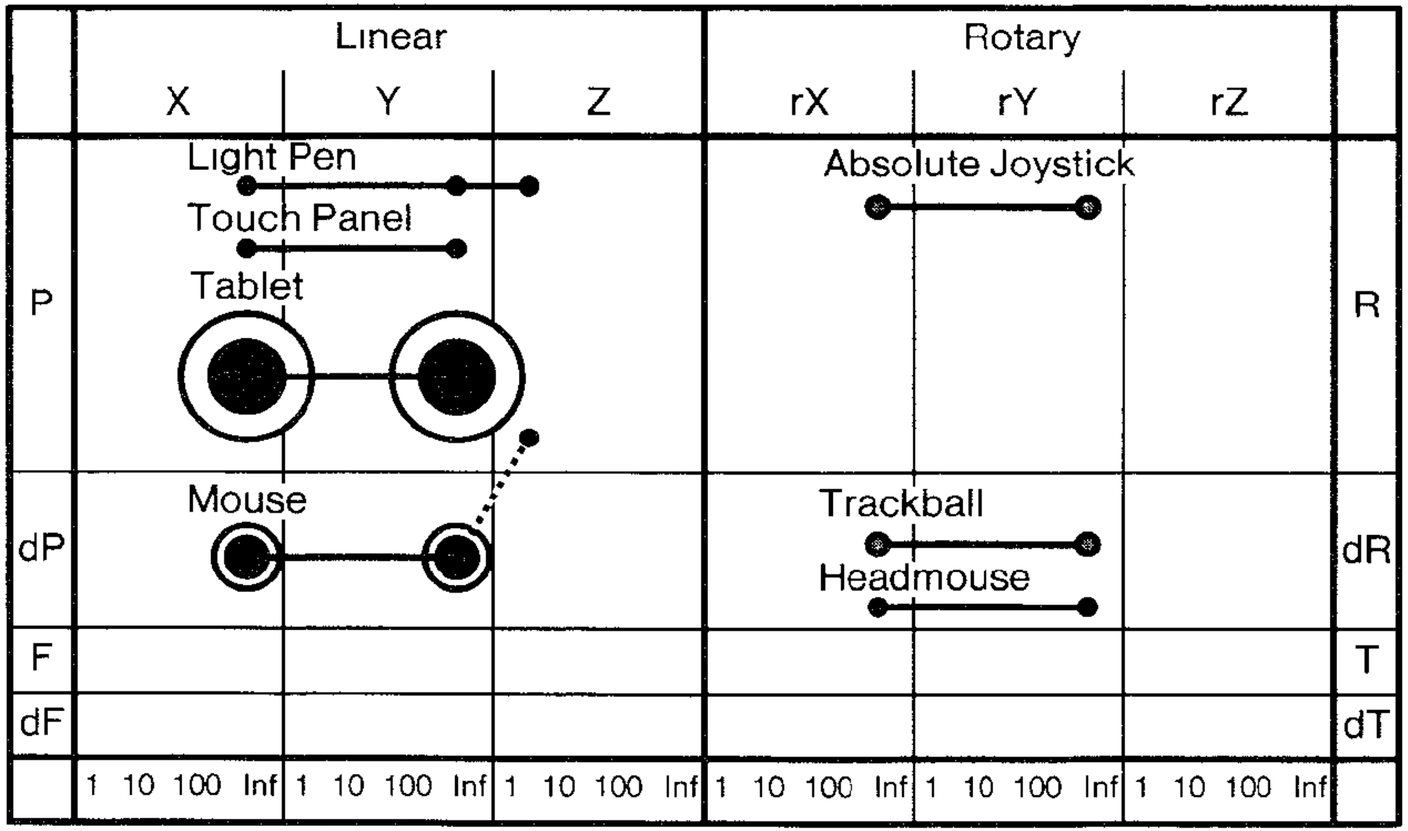
Testing Points

- **Expressiveness** describes how precisely the meaning is conveyed

- For input devices, expressiveness suffers if $|In| \neq |Out|$
  - $|In| < |Out|$: Cannot specify all legal values
  - $|In| > |Out|$: Can specify illegal values

Testing Points

- **Effectiveness** describes how well the intention can be communicated



“Will I Get a Seat in This Class?”

CHAPTER 3

Window System Architecture

Window Systems: Basic 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

Window Systems: Requirements

- **Independent** of hardware and operating system
- No noticeable **delays** (few ms) for basic operations,
  e.g. moving window, redrawing cursor
- **Customizable** look&feel for user preferences
- Input & Output in **parallel**
- **Multimedia** support: Graphics, audio, …
- Support for various **input devices** and modalities

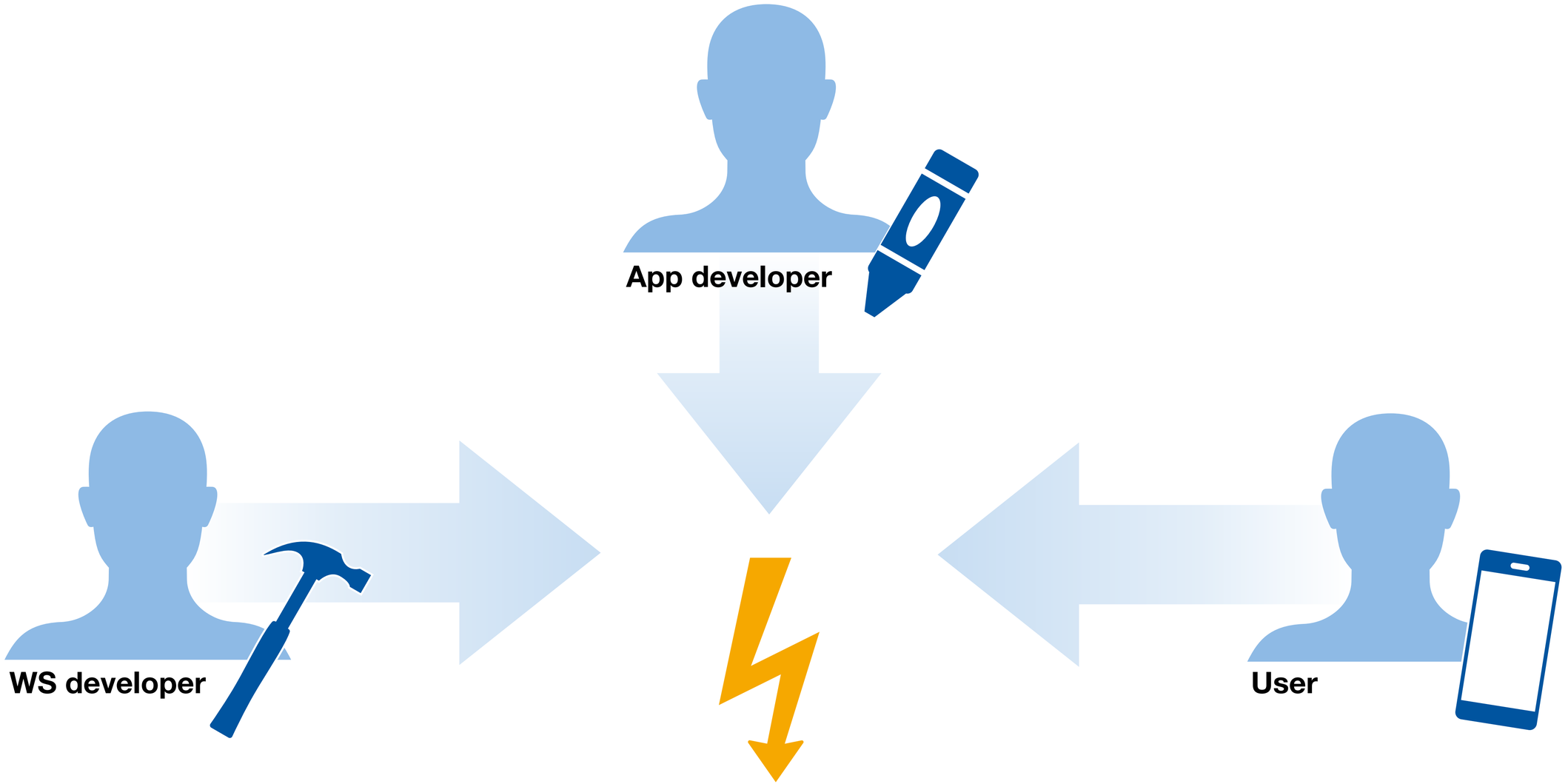
Window Systems: Evaluation Criteria

- **Availability**
  Platforms supported
- **Productivity**
  For application development
- **Parallelism**
  External and internal
- **Performance**
  Usage of resources and latency
- **Graphics model**
  RasterOp vs. vector
- **Appearance**
  Look & Feel, exchangeable?

Window Systems: Evaluation Criteria

- **Extensibility**
  In source code or at runtime
- **Adaptability**
  Localization and customization at runtime
- **Resource sharing**
  E.g., fonts
- **Distribution**
  Over network
- **API**
  Structure and comfort
- **Independence**
  Of application and interaction logic inside programs written for the WS
- **Inter-Application Communication**
  Copy & Paste, Drag & Drop

Window Systems: Conflict



Window System Architecture