4M21 Software Engineering and Design
Human-Computer Interaction

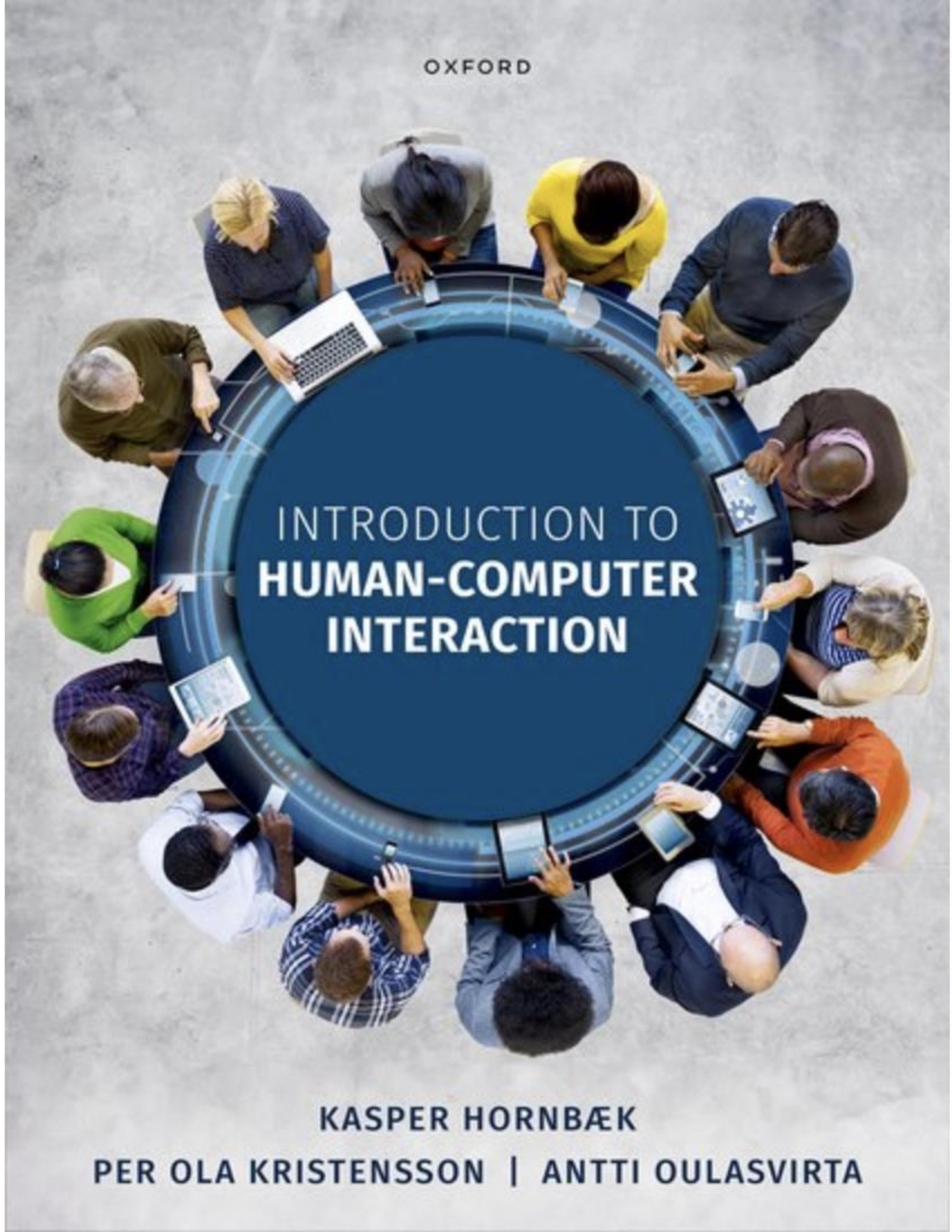
Lecture 5/8

Professor Per Ola Kristensson

Department of Engineering

University of Cambridge

https://global.oup.com/academic/product/introduction-to-human-computer-interaction-9780192864543?cc=gb&lang=en&


OXFORD
INTRODUCTION TO HUMAN-COMPUTER INTERACTION
KASPER HORNBÆK
PER OLA KRISTENSSON | ANTTI OULASVIRTA

User Interfaces

Definitions and elements of a user interface

- **Devices**
  - Input devices: the user providing input to the system
  - Output devices: the system providing output to the user
- **Interaction techniques**
  - Interaction techniques translate inputs from input devices, and possibly context sensors, into changes in the rest of the user interface
- **Representations**
  - Representations determine how data, events, objects, and actions appear to the user
  - The design and appearance of icons and menus are examples of representations
- **Organization**
  - The broader set of principles that organize the data environment in the user interface

Design objectives

- **Usability**
  - Qualities of the user interface that allow users to achieve their goals effectively, efficiently and enjoyably
- **Accessibility**
  - Equivalent levels of usability across user groups
- **Efficiency**
  - The speed-accuracy trade-off: *performance*
- **Learnability**
  - Easy to learn
  - Time to become *proficient*
  - How to allow *optimal performance*
- **Discoverability and explorability**
- **Consistency**
  - Internal consistency: similar features and aspects of the interface are used in similar ways
  - External consistency: compliance with standards, guidelines, and general expectations

Example: guidelines (Nielsen's heuristics)

- **Visibility of system status:** the current state of the system should be visible to the user
- **Match between system and the real world:** the user interface should follow the language and any relevant conventions that users are already aware of
- **User control and freedom:** users should be encouraged to explore different ways to achieve their goals in the user interface
- **Consistency and standards:** external consistency (standards) and internal consistency (similar labels for similar features)
- **Error prevention:** the user interface should be designed to prevent errors, for example, by displaying a warning and requiring user confirmation before a non-reversible action is triggered, such as deleting a file
- **Recognition rather than recall:** It is more difficult for users to recall from memory how to trigger an action than it is to recognize a mechanism for triggering the action shown on the display
- **Flexibility and efficiency of use:** since users inevitably vary in their proficiency of a user interface, it is often effective to provide interface features that tailor to different users
- **Aesthetic and minimalist design:** ensure that the user interface focuses on content and information essential for allowing users to achieve their primary goals
- **Help users recognize, diagnose, and recover from errors:** provide error messages that are understandable by users and offer a clear solution path to rectify the problem
- **Help and documentation**: if documentation is required, ensure that it is focused on aiding users in their tasks and is easy to search

J. Nielsen. Usability engineering. Morgan Kaufmann, 1994.

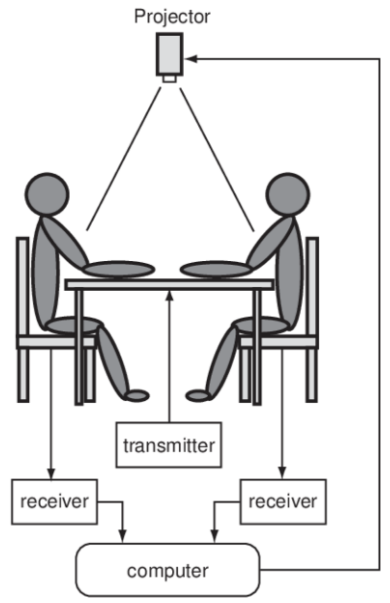
Input

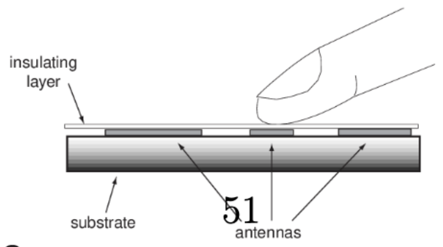
Direct control

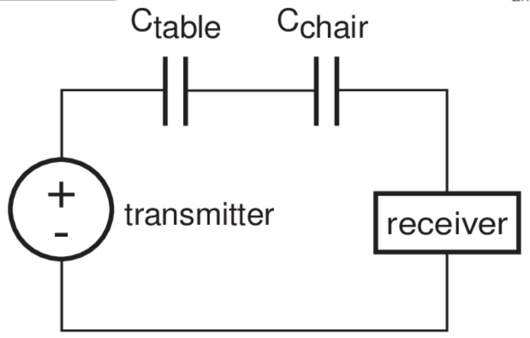
- **Direct control** pointing devices allow the user to select absolute coordinates on displays
- Light pen
  - Light sensitive rod which the user holds in their dominant hand
  - Designed to be used together with cathode-ray tube (CRT) displays
  - A CRT generates a pixel image using a raster scan row by row, column by column
  - The light sensitive component in the light pen can be used in combination with timing information of the raster scan to detect the location of the tip of the light pen
- Resistive touchscreen
  - Usually designed with two flexible sheets separated by an insulator, such as an air gap
  - When two sheets are pressed together using, for example, a tip of a pen or a finger, the two sheets make contact
  - By ensuring the sheets have horizontal and vertical traces it is possible to determine the touch location
  - Portable and inexpensive to manufacture
- Capacitive touchscreen
  - Rely on capacitive coupling between an implement, typically the user’s fingertip, and the display to sense the absolute position the user touches on the display
  - Modern capacitive touchscreens allow light touches and multi-touches
  - Response time is better than resistive touchscreens but precision is worse

Example: touch for several people at the same time?

- The *DiamondTouch* uses a projector that places an image on a table between the users
- The table has been prepared with a number of antennas below an insulating layer
- The antennas works as switches so that it is possible to detect when a user touches within the area of the antenna, but not where
- In the prototype there were four antennas per square centimeter.
- The identification of users were accomplished through instrumenting users' chairs
- When the user touches the table, a circuit is completed from the transmitter, through the antenna on the table, and through the user and the chair, and back to the transmitter
- Since this circuit is unique to a user, the touchpoint allows the identification of the user



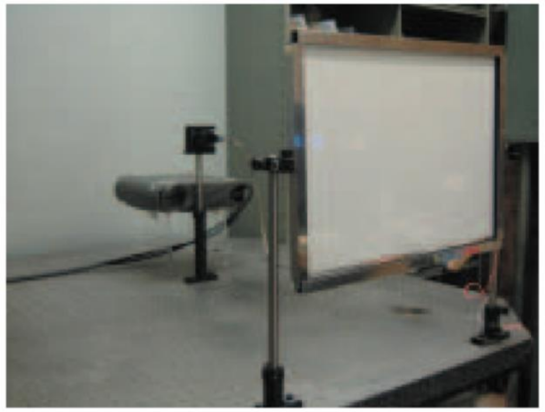




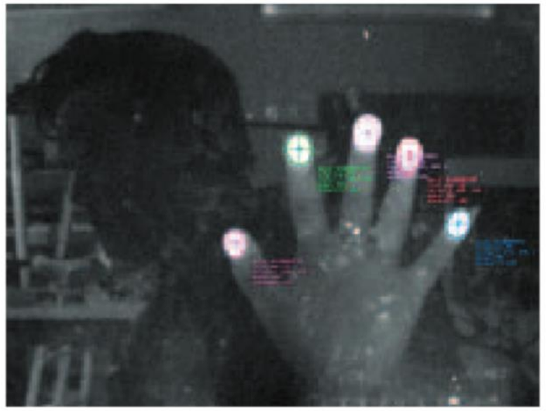
Coupling capacitance is a series circuit with coupling capacitance: $C_{coupling} \approx C_{table}$ *(the user's finger touching the table) since* $C_{table} \ll C_{chair}$.

Example: frustrated total internal reflection

- *Refraction* is when light changes direction, or bends, when it passes from one medium to another
- *Total internal reflection* happens when light passes through a medium with a lower index of refraction and the angle of incident of this refraction is greater than a critical angle
- With the right choice of material this interface at the medium can be designed to frustrate this total internal reflection, resulting in light escaping through this material instead
- A multi-touch FTIR surface uses an acrylic pane which is edge-lit by infrared light-emitting diodes
- Due to the total internal reflection the infrared light is trapped within the sheet
- A video camera is mounted behind the sheet, capturing the resulting image
- Now touch points can easily be detected using basic computer vision



An example of a large surface multi-touch sensor setup



The result of a user touching the surface and thereby frustrating the light trapped inside the pane, causing it to scatter and be visible to the camera

Indirect control

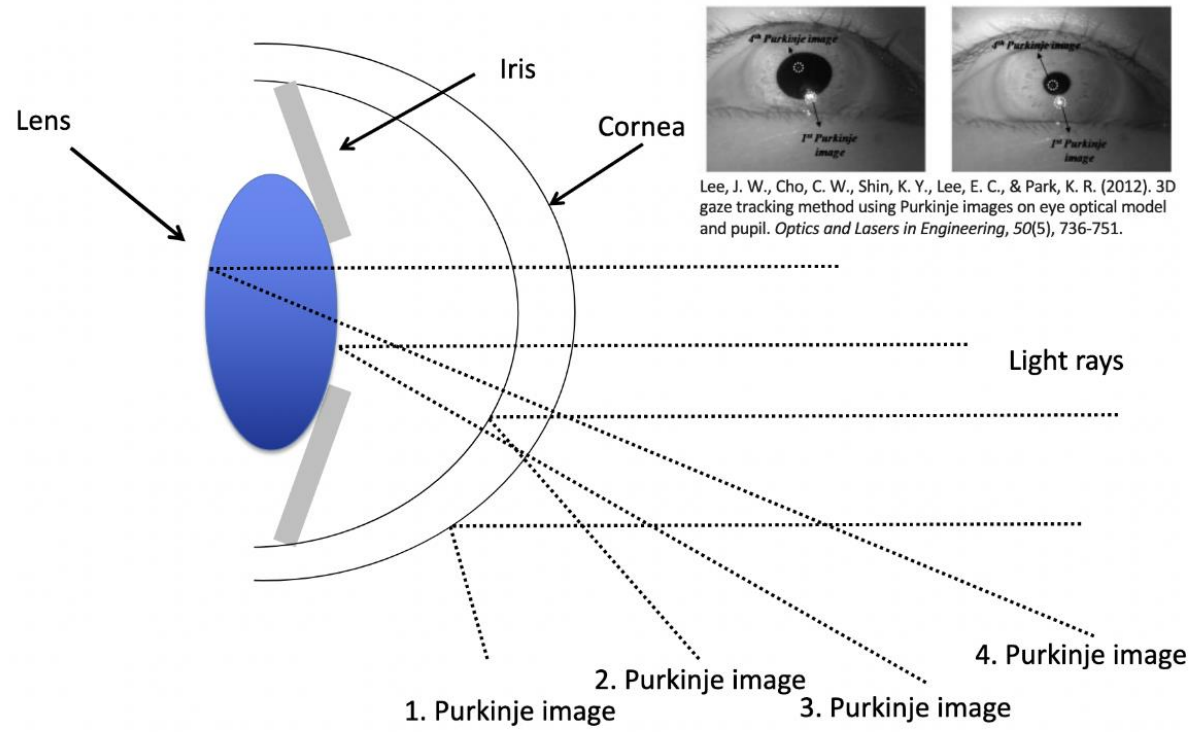
- **Indirect control** pointing devices enable the user to control a cursor on a display in response to manipulation of the pointing device
- Thus, movement of the cursor on the display is relative to the manipulation of the pointing device
- The **C:D ratio** controls the ratio of the controller (such as the mouse) and the display (such as an LCD screen)
  - Manipulation of the C:D ratio enables mouse acceleration and other interaction techniques
- Examples of indirect control include computer mice and touchpads

Uncertain control

- The previous direct and indirect control input devices allow computer systems to understand the users’ intention with a high degree of certainty
- **Uncertain control** is a class of input devices that must be designed to be robust in the presence of significant uncertainty around the user’s intention, such as high noise
- Examples of uncertain control input devices include physical switches, accelerometers and gyroscopes, brain-computer interfaces and electromyography

Eye tracking

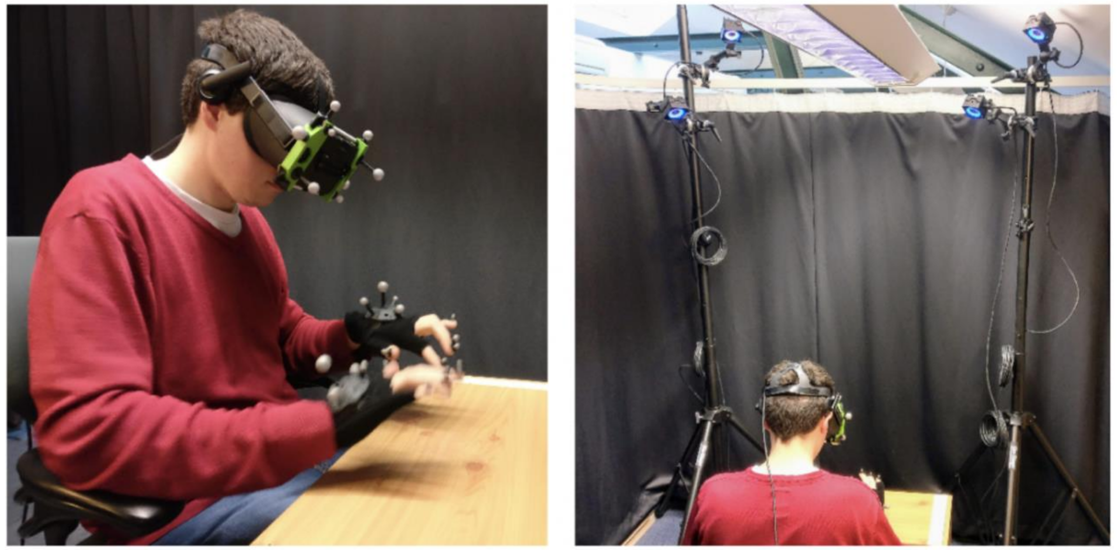
- An **eye tracking** input device estimates the user's gaze on a display by tracking the user's
  rotation of the eye
- Several methods are available, and they all require calibration
- One method is to project infrared light towards the user's eye
- The anatomical parts of the eye that reflect the light efficiently generate so-called Purkinje images
- These images can then be used to infer the user's gaze
- High-accuracy commercial eye-tracking systems typically implement some variant of this or a related scheme



Lee, J. W., Cho, C. W., Shin, K. Y., Lee, E. C., & Park, K. R. (2012). 3D gaze tracking method using Purkinje images on eye optical model and pupil. *Optics and Lasers in Engineering*, *50*(5), 736-751.

Hand- and finger-tracking

- Hand- and finger-tracking is used to estimate the location of the user’s hands and fingers in 3D space
- Accurate hand- and finger-tracking can be based on optical markers
    - Such markers are fitted to the hands and fingers, or other body parts that need to be tracked
    - These markers are then tracked by high-resolution cameras and software is then able to estimate the 3D locations of the markers in space
    - Optical marker-based systems are common in HCI research when there is a need to acquire a relatively high accuracy and low latency view of users’ movement
- An alternative way of performing hand- and finger-tracking is to use a depth camera mounted on a headset, and then use computer vision methods to infer a hand skeleton
    - This approach is common in state-of-the-art virtual and augmented reality headsets.
    - The skeleton serves as a low-dimensional representation of the user’s finger, and hand movement can then be used more easily to, for example, design a gesture control system

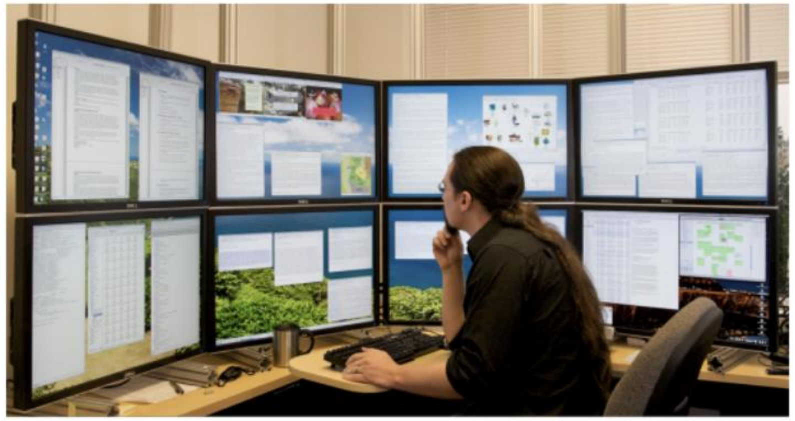


An example of using optical markers to measure users’ hand movements in typing tasks in mid-air

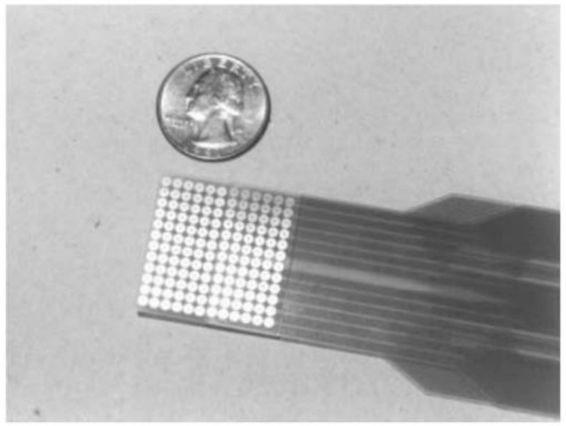
Displays

Displays

- The choice of output device and an understanding of the capabilities of a particular output device are important
- We are using the term *display* to cover visual displays, haptic devices, as well as other devices that can render information in a modality that humans can perceive
- For example, an interface pneumatically driven compression around the arm to deliver notifications to people will be considered a display

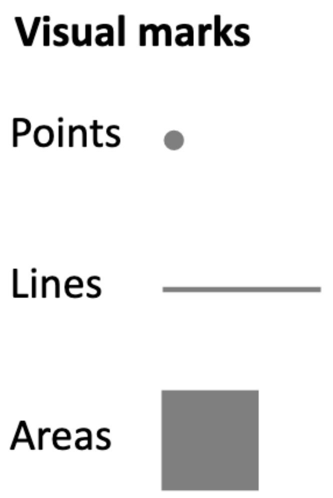

A multi-display setup for sensemaking

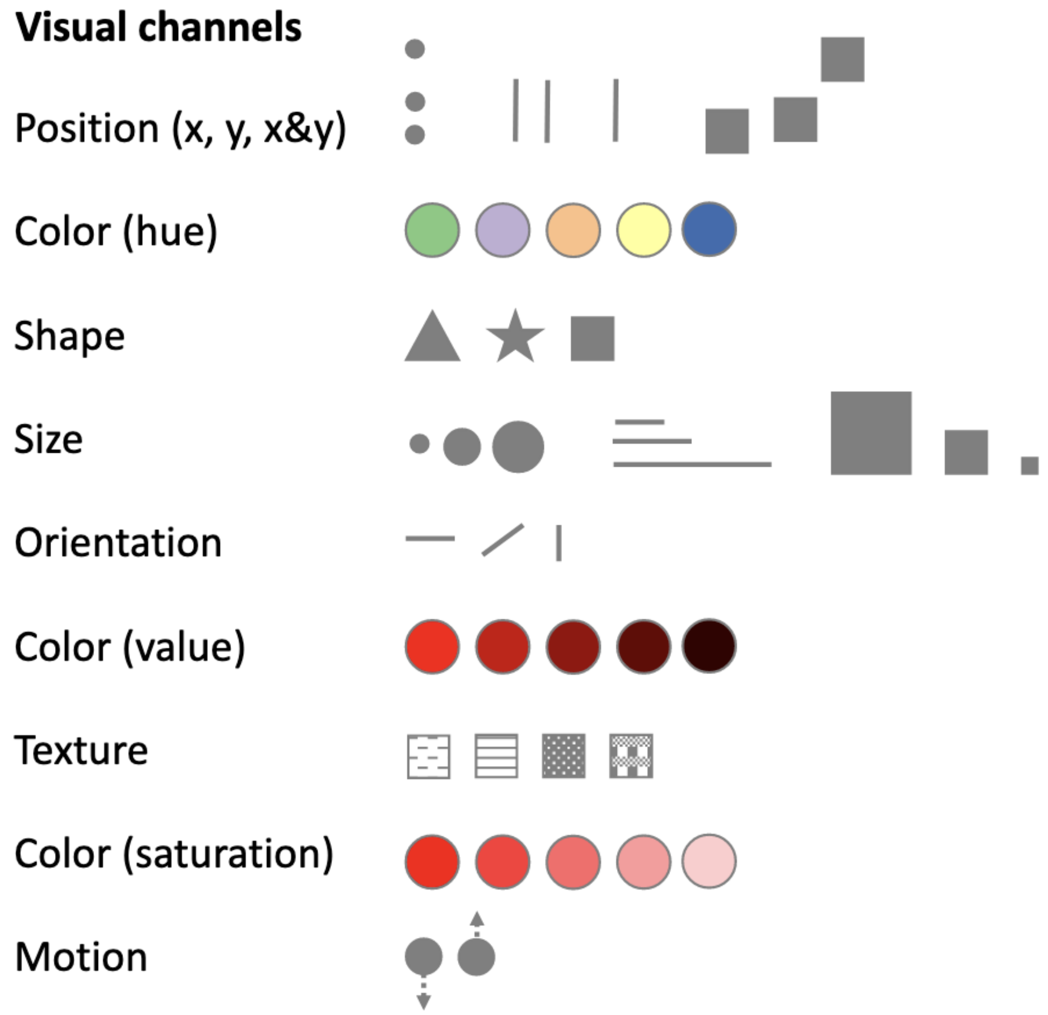
A multimodal display


An electrotactile display to be used on the tongue

Visual encoding

- A **visual encoding** is a way to show a nominal, ordered, or quantitative variable
- Visual encodings are typically separated into visual marks and visual channels
- **Visual marks** are geometric primitives such as points, areas, and lines
- **Visual channels** include the position of a mark, its color, and its texture





Haptics

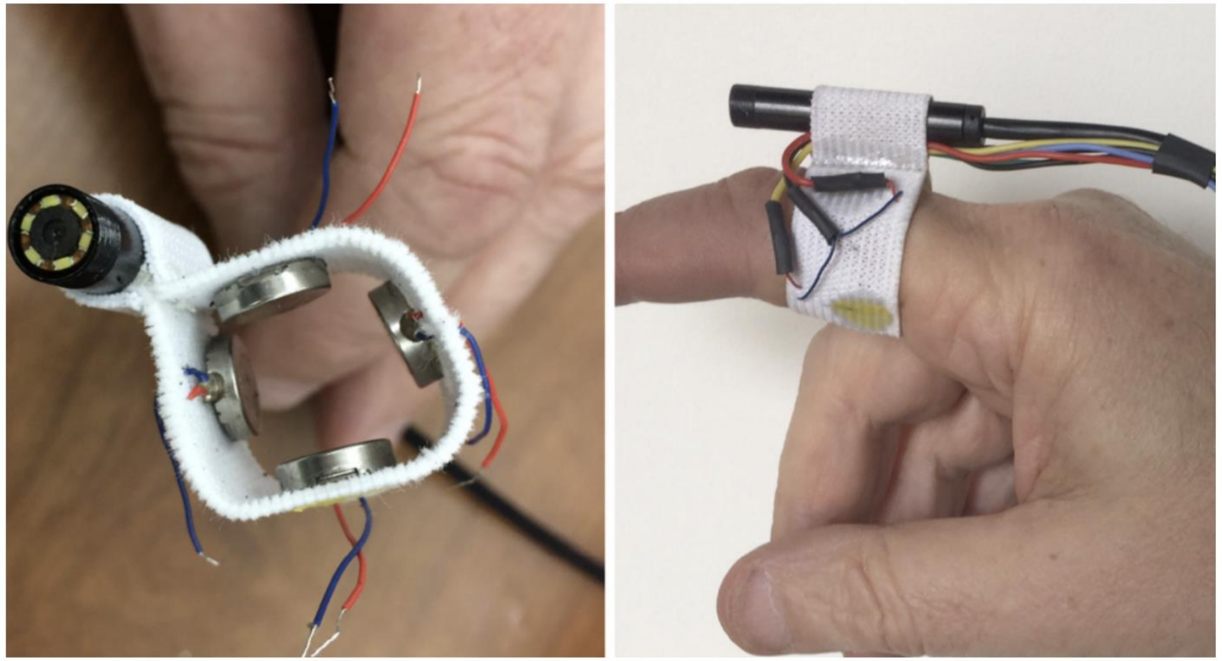
- **Haptic** technologies enable users to perceive touch, or, in general, forces on their body
  - A traditional application of haptics is telerobtics, where the user directly controls of a remote robot
  - Haptics are then used to improve users’ ability to perceive reactions of the robot in response to control commands
- Vibration
  - A common approach to actuate a sense of touch is the use of vibration
  - Vibration can be generated straight-forwardly using a vibrator
  - An electric motor transfers torque (rotational force) using a drive shaft
  - If an unbalanced mass is connected to the drive shaft, the system will produce vibrations
- Force feedback
  - A force feedback device is a system that generates a perceived force by actuation

Mechanical exoskeleton gloves

- One example of force feedback is a mechanical exoskeleton glove
  - When the user wears the glove the system tracks the user's fingers
  - If the fingers are supposed to interact with a virtual object, then the system uses force feedback to simulate the forces the user would have experienced had the user touched a physical object
  - Mechanical exoskeleton gloves have applications in telerobotics, teleoperation and virtual reality
- Mechanical exoskeleton gloves can be grounded
  - Grounding means that the system is connected to a grounding point, such as a desk or a floor
  - This allows the system to simulate the sensation of weight, for example, a user lifting a heavy virtual bag in virtual reality
- Another extension is to augment the gloves with additional feedback, such as vibration feedback or thermal feedback to simulate tactile sensation or touching cold or hot surfaces

Example: haptic guidance for blind people

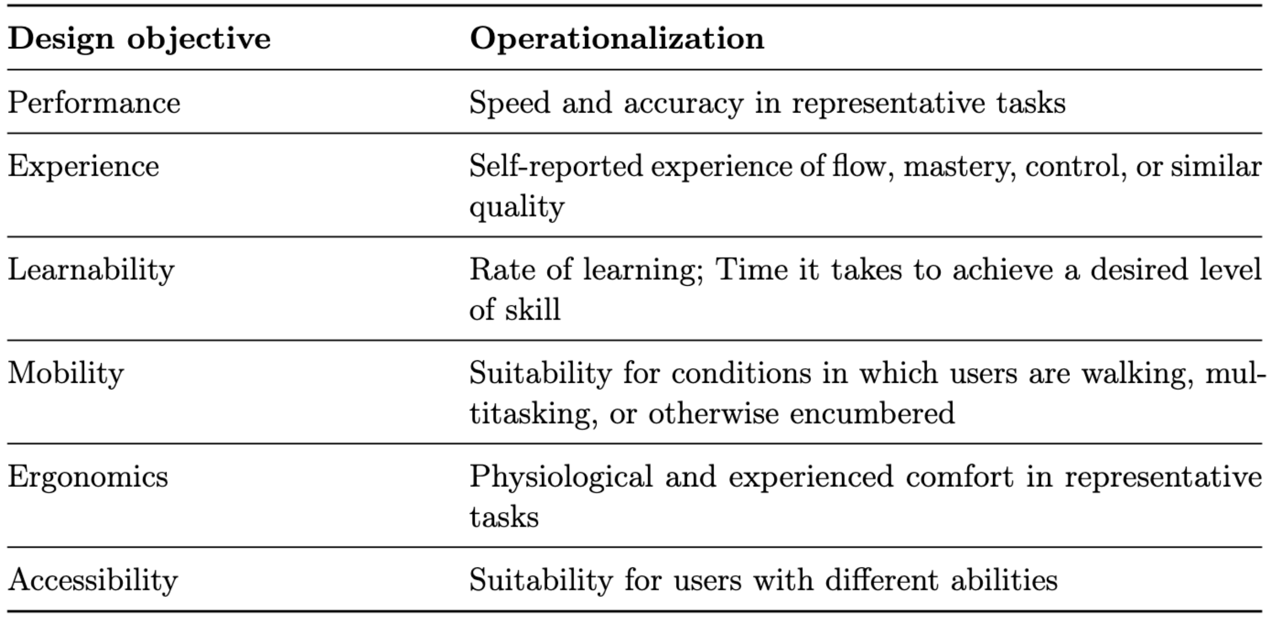
- *FingerSight* is a haptic guidance system that helps people with vision impairments to locate and reach to objects in peripersonal space
- **Peripersonal space** refers to the physical space surrounding our bodies within which we can reach to and grasp objects
- The system consists of a finger-worn ring that houses a camera and a haptic display
- The display has four evenly spaced tactors that can stimulate the index finger
- The camera tracks objects to track the distance to them
- Distance is mapped to the features of the haptic stimulation, with the purpose of indicating how far away an object of interest is
- As the user moves the hand in the air closer or further from an object, the tactors modulate their actuation frequency accordingly



Interaction Techniques

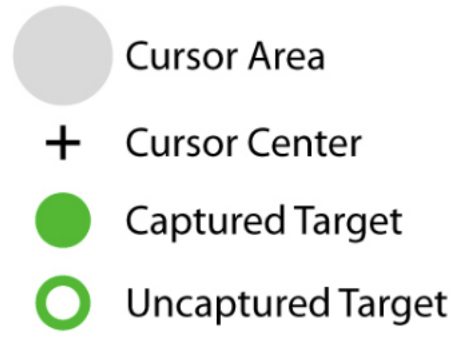
Interaction techniques

- An **interaction technique** is a computation that couples input and output, to support elementary interactive tasks
- Interaction techniques make frequent tasks possible, such as:
    - Moving the active area or cursor in the user interface using pointing techniques
    - Selecting or manipulating an object, including menu techniques
    - Entering numbers and text using text entry methods
    - Changing which part of an information space the user sees, often called camera control or navigation techniques



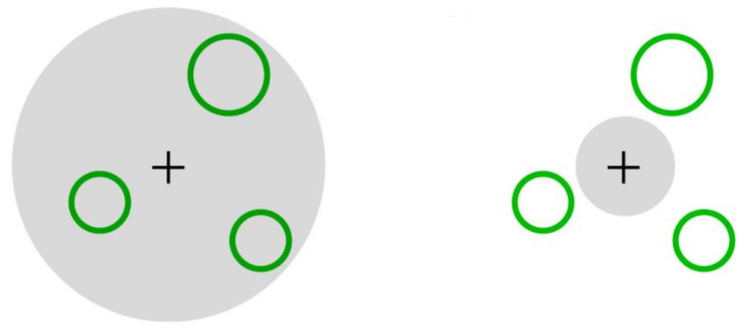
Design objective	**Operationalization**
Performance	Speed and accuracy in representative tasks
Experience	Self-reported experience of flow, mastery, control, or similar quality
Learnability	Rate of learning; Time it takes to achieve a desired level of skill
Mobility	Suitability for conditions in which users are walking, multitasking, or otherwise encumbered
Ergonomics	Physiological and experienced comfort in representative tasks
Accessibility	Suitability for users with different abilities

Example: the end of misclicks?

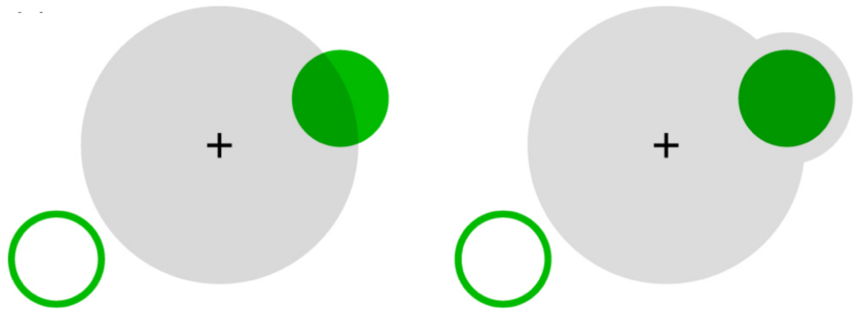


- The *bubble cursor* is an interaction technique to select targets with an indirect pointing device

1. The system computes a Voronoi diagram for the targets on the display
   - Given the targets, a Voronoi diagram partitions the display such that every pixel on the display is mapped to exactly one target
   - This defines selection regions for the cursor
2. When the cursor is moving, the system actively changes the selection region to the region the target the cursor is mapped to at any given time
3. Clicks are assigned to the target within the same Voronoi cell as the cursor



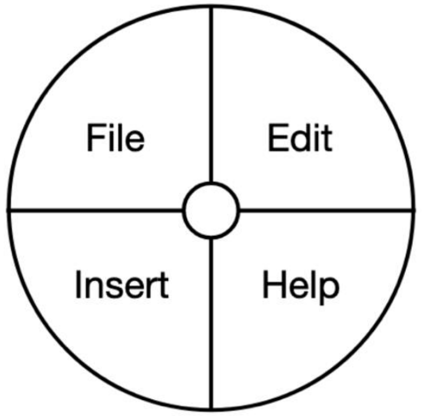
An *area cursor* frequently captures multiple targets, or are inefficient by being too small



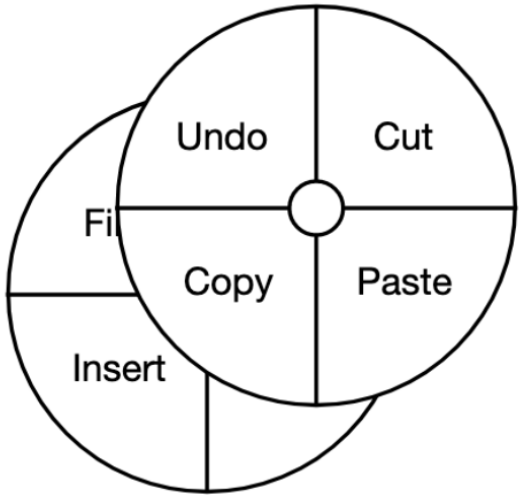
A *bubble cursor* is dynamically adjusted such that only one target is captured at a time

Pie menus

- While hierarchical linear pull-down menus dominate in most modern graphical user interfaces, they are not only means of representing a menu structure
- A **pie menu** presents menu as a circle subdivided into circular sectors associated with commands or further menu structures
- The user typically triggers the pie menu using a button press, such as pressing down the left mouse button
  - When the pie menu is invoked, it appears in the center of the pie menu
  - The user can now perform a selection by sliding the cursor towards a circular sector
- Pie menus can be hierarchical because a circular sector can be associated with either a command or a menu structure
  - If the selected circular sector is associated with a menu structure, then a selection will invoke a second pie menu revealing this menu structure



First level pie menu



Second level pie menu

Marking menus

- A **marking menu** is a refinement of a pie menu that supports a continuous transition from novice to expert behavior
- A marking menu is an interaction technique that couples an incremental angular 2D single-stroke gesture recognizer with a pie menu
- The user triggers the marking menu in a similar manner as a pie menu
    - However, unlike a pie menu, the marking menu does not reveal a pie menu immediately
    - Instead, there is a brief timeout period before the pie menu appears
    - After the timeout, the marking menu behaves like a pie menu
    - However, if the user performs a 2D gesture rapidly before the timeout, the pie menu will not be shown
- The central idea in the marking menu is to ensure that the trajectory for invoking the command, either navigating to the menu item or performing a 2D gesture, is **identical**
- This means that during use, the user learns the motor pattern for the command by motor memory consolidation
- Importantly, this process is both continuous and unconscious
- This allows marking menus to provide users with a seamless novice to expert transition

Marking menus: working principle

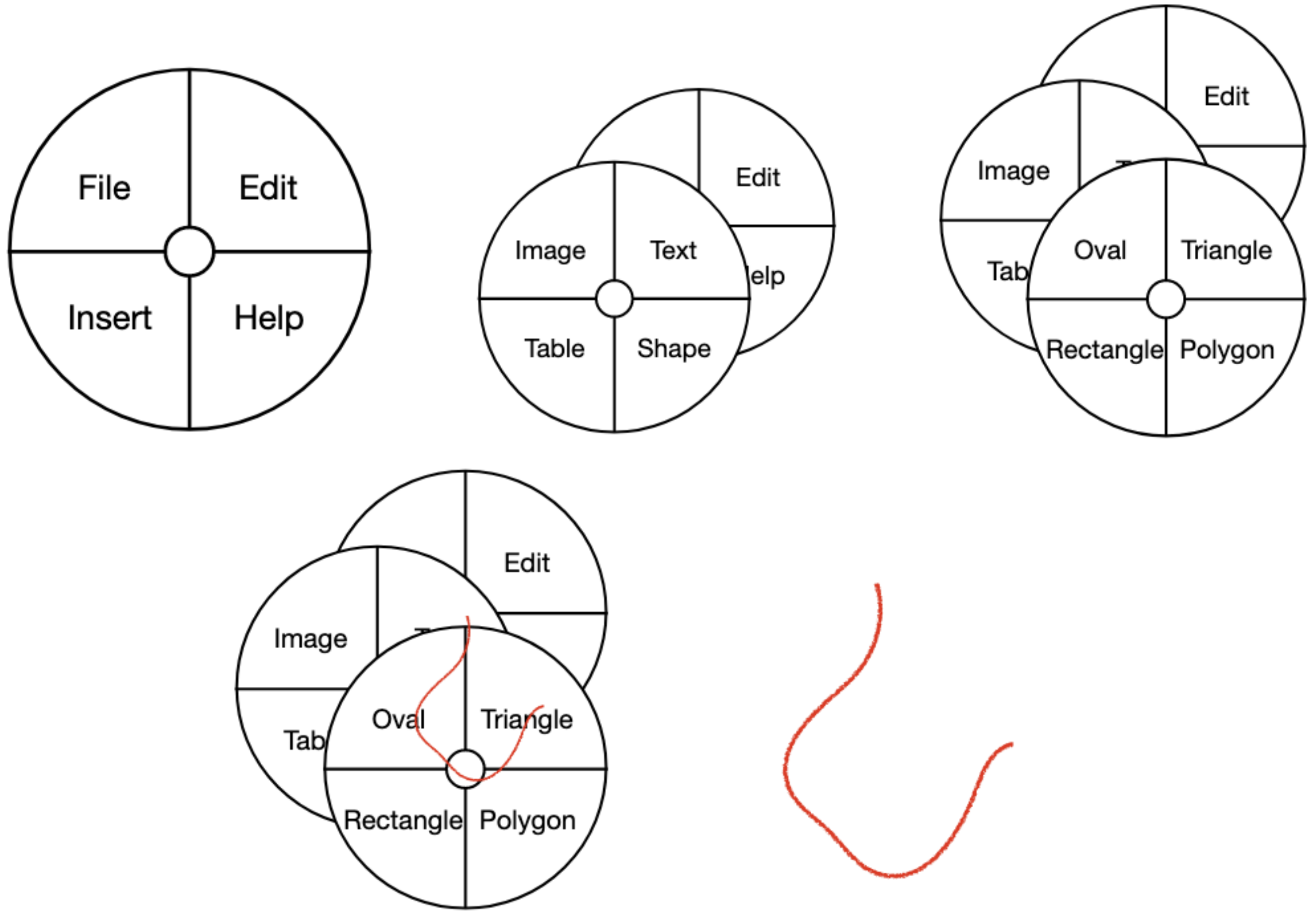
a) The marking menu is triggered in a similar vein to a pie menu
   - The user drags the cursor towards the *Insert* menu

b) This action triggers a second-level marking menu
   - The user drags the cursor towards the *Shape* menu

c) This triggers a third-level marking menu
   - The user triggers the *Triangle* menu item

d) The entire selection motion for the user triggering (1) the marking menu; (2) the second-level *Insert* menu; (3) the third-level *Shape* menu; and (4) the selection of the *Triangle* menu item is illustrated as a red trace

e) This trace forms a 2D gesture which can be directly articulated by the user without invoking the marking menu



Text entry

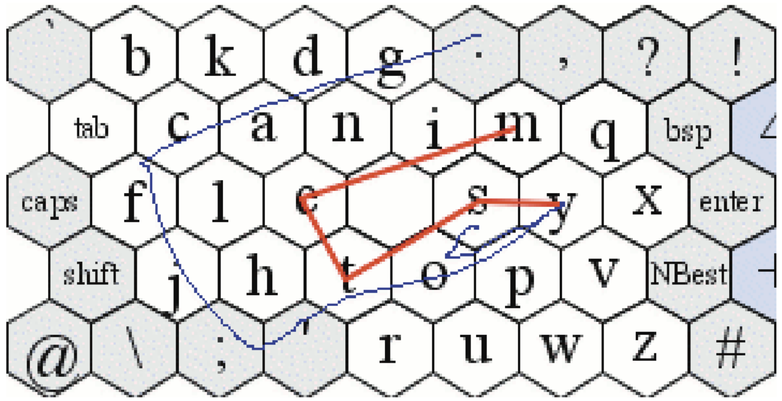
- Entering text and data is one of the most common task carried out with computers, methods that allow such activity are called **text entry methods**
- **Intelligent text entry methods** infer or predict users' intended text
- Examples of such methods include:
  - Speech and handwriting recognition
  - Word prediction and auto-complete on touchscreen keyboards
  - Sentence retrieval and sentence generation methods, primarily for nonspeaking individuals with motor disabilities

Findings that support the need for intelligent text entry on mobile devices

- Young adults type at around 36 wpm (words per minute) and have an average of 2.3 % uncorrected errors
  - This level is far off from what is achievable with physical keyboards, which is around 52 wpm in a comparable sample
- Typing with one finger is slower than using two fingers
  - On average, two-thumb typing is superior to other methods of using fingers
- Errors are costly to correct in comparison to physical keyboards
  - Users differ in how typo-averse they are
  - Some users slow down to avoid having to correct errors
- Typing is not just about motor control
  - Users also need to control how they deploy visual attention between the text display (what is typed) and the virtual keyboard
- Typing while walking or doing other activities slows down typing performance and causes errors
- Many aging users and users with cognitive, motor, or vision deficiencies have serious difficulties in typing with standard keyboards

Example: gesture keyboard

- A **gesture keyboard**, also known as 'gesture typing', 'swyping', etc. is a text entry method based on representing words as traces on a keyboard
  - The word "the" will be the trace from the *T-H-E* keys in sequence
- The user can now articulate a gesture for the desired word that can be recognized by a gesture recognizer
- Allows a smooth transition from novice use (**closed-loop** tracing letter-by-letter, movement time predicted by the Law of Crossing) to expert use (**open-loop** direct recall from motor memory)
- Heavily commercialized, available on all major mobile phone platforms by default, including emerging VR headsets
- Invented by Kristensson and Zhai in 2001-2002; key system paper published in 2004
- First public release in 2004 through IBM Research
- First commercial release by ShapeWriter, Inc. on iPhone and Android in early 2008 and purchased by Nuance Communications in 2010, cross-licensing to all OEMs



A user has written the word "system". The gesture (in blue) is robust to changes in scale and translation in relation to the ideal shorthand gesture for the word (traced in red). This enables expert users to quickly articulate gestures for known words by direct recall from motor memory.

Commands and Navigation

Naming commands

- For commands to be effective they require appropriate naming of actions and objects
- Why is naming things in user interfaces challenging?
    - The **vocabulary problem**: people tend to not come up with the same names for actions and objects (on average, every fifth command would overlap between two people)
    - **Polysemy**: the same word may mean different things
    - The role of **context** when interpreting the meaning of commands
- Techniques for identifying names for commands
    - Pick the names that users employ in real life for objects and actions
        - Such names may be found during user research
        - A straightforward idea, captured in heuristics such as “speak the users’ language”
    - Carry out an elicitation study
        - Elicit from users how they will execute a particular command and study the agreement across users on those commands to decide on a distinct set of commands that users will produce spontaneously with the highest likelihood.

Command-line interfaces

- A command-line interface can be decomposed into a prompt, a command, and *n* parameters:

```
prompt command parameter¹ parameter² . . . parameterⁿ
```

- **Prompt:** a system-generated indication that the system is awaiting a command from the user
  - Prompts can be minimal, providing only some indication (such as a beep) that the system is awaiting a response, or they can be elaborate and provide additional information to the user, such as some aspect of system states
- **Command:** the instruction provided by the user to the system
- **Parameter list:** a set of command-specific additional instructions provided by the user, which will affect the execution of the command

Design issues for command-line interfaces

- **Availability:** how does the user know which commands are available?
- **Naming:** how does the user know the names of the commands the system understands?
- **Learning:** how does the user learn the availability of the commands and their parameters?
- **Recall:** how does the system support the user in recalling commands and their parameters?
  Syntax: how does a user know which commands expect parameters and the form of these parameters?
- **Transparency:** how does the system convey to the user the way in which it interprets commands?
  - The user needs to both understand what is achievable within the command-line interface and be able to diagnose why a particular command input does not result in an intended response

Navigation

- **Hypertext:** text that allow the user to retrieve further information when interacted with
  - Hypertexts are connected using hyperlinks, which are particular elements that trigger further retrieval
  - Hyperlinks consisting of text are usually denoted in some unique way to provide sufficient perceived affordance so that the user can realize a hyperlink is present
- **Menu user interfaces:** a set of selectable options
  - Items are made available via recognition not recall, which is effortful
  - There is no need to memorize command names
  - However, recall can be used to further boost performance
  - *Keyboard shortcuts* allow users to drastically improve performance for oft-used commands
    - This mode, although it requires some effort to learn, is optional

Notational systems

- Systems with novel graphical elements that are composed according to novel visual grammars (rules)
- Can be simple and complex systems
- Examples:
    - Visual programming languages
    - Central heating controls
    - New pervasive and ubiquitous systems
    - AI-infused user interfaces

Types of notation use activity

- **Incrementation:** adding cards to a card file, formulas to a spreadsheet or statements to a program
- **Transcription:** copying book details to an index card; converting a formula into spreadsheet or code terms
- **Modification:** changing the index terms in a library catalogue; changing layout of a spreadsheet; modifying spreadsheet or program for a different problem
- **Exploratory design:** sketching; design of typography, software, etc; other cases where the final outcome cannot be envisaged and has to be ‘discovered’
- **Searching:** hunting for a known target, such as where a function is called
- **Exploratory understanding:** discovering a structure or an algorithm, or discovering the basis of classification

Source: Blackwell, A.F. and Green, T.R.G. 2003. Notational systems–the cognitive dimensions of notations framework. In Carroll, J.M. (Ed.) *HCI Models, Theories, and Frameworks: Toward a Multidisciplinary Science.* San Francisco: Morgan Kaufmann, pp. 103–134.

Cognitive dimensions of notation

- **Viscosity:** resistance to change
- **Visibility:** ability to view components easily
- **Premature commitment:** constraints on the order of doing things
- **Hidden dependencies:** important links between entities are not visible
- **Role-expressiveness:** the purpose of an entity is readily inferred
- **Error-proneness:** the notation invites mistakes and the system gives little protection
- **Abstraction:** types and availability of abstraction mechanisms
- **Secondary notation:** extra information in means other than formal syntax
- **Closeness of mapping:** closeness of representation to domain
- **Consistency:** similar semantics are expressed in similar syntactic forms
- **Diffuseness:** verbosity of language
- **Hard mental operations:** high demand on cognitive resources
- **Provisionality:** degree of commitment to actions or marks
- **Progressive evaluation:** work-to-date can be checked at any time

Source: Blackwell, A.F. and Green, T.R.G. 2003. Notational systems–the cognitive dimensions of notations framework. In Carroll, J.M. (Ed.) *HCI Models, Theories, and Frameworks: Toward a Multidisciplinary Science.* San Francisco: Morgan Kaufmann, pp. 103–134.

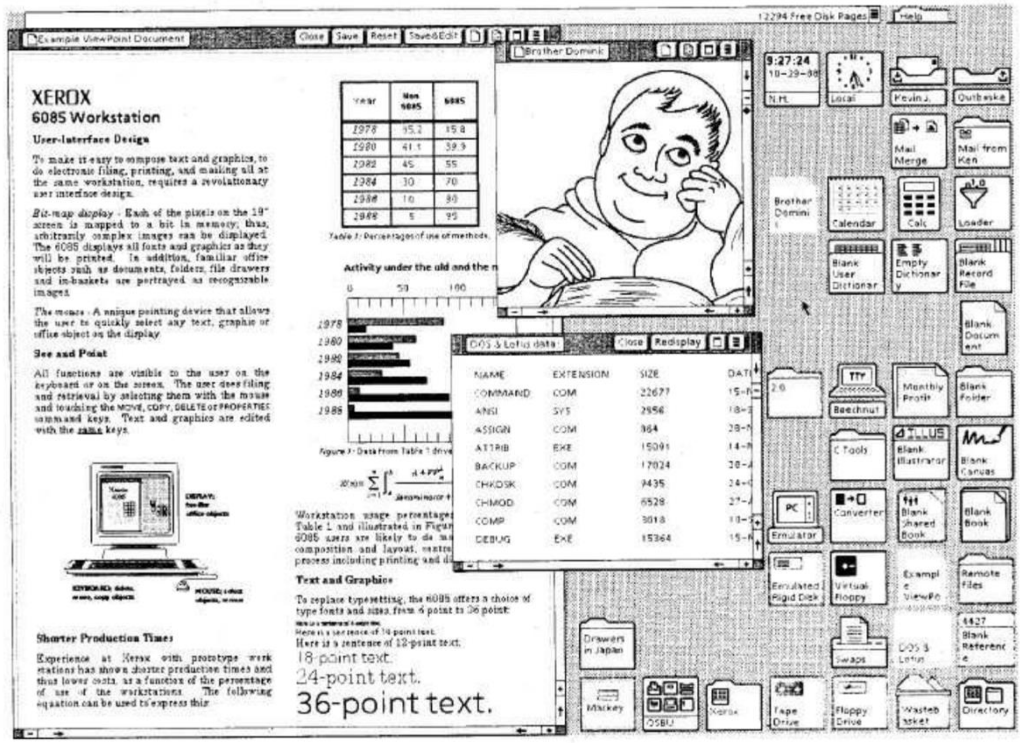
Example

- Consider a smart ring for controlling a heating system:
  - The ring visualizes temperature by color change (heatmap scale)
  - When the user wishes to adjust the temperature the user touches the ring and strokes it counter-clockwise to dial down the temperature and opposite to dial it up
  - The ring color momentarily changes in response to user adjustment and remains in this state for five seconds at which point the ring color reverts to its normal behavior
- This system is **viscous**, has high **consistency** and has low **demand on mental operations**; nevertheless, but it can be argued it has several analytic challenges:
  - The system has poor **visibility** as the temperature adjustment method is not apparent (poor perceived affordance)
  - The **mapping is both close and not close** to the task. On the one hand, the ring motion models a dial on a heating controller although the degree of **role expressiveness** in the design is up for debate. On the other hand, the fact the user is adjusting the temperature **in the future** is not apparent
  - There are **hidden dependencies** as the fact the user is manipulating the target temperature in a control system is not revealed

Graphical User Interfaces

Graphical user interfaces

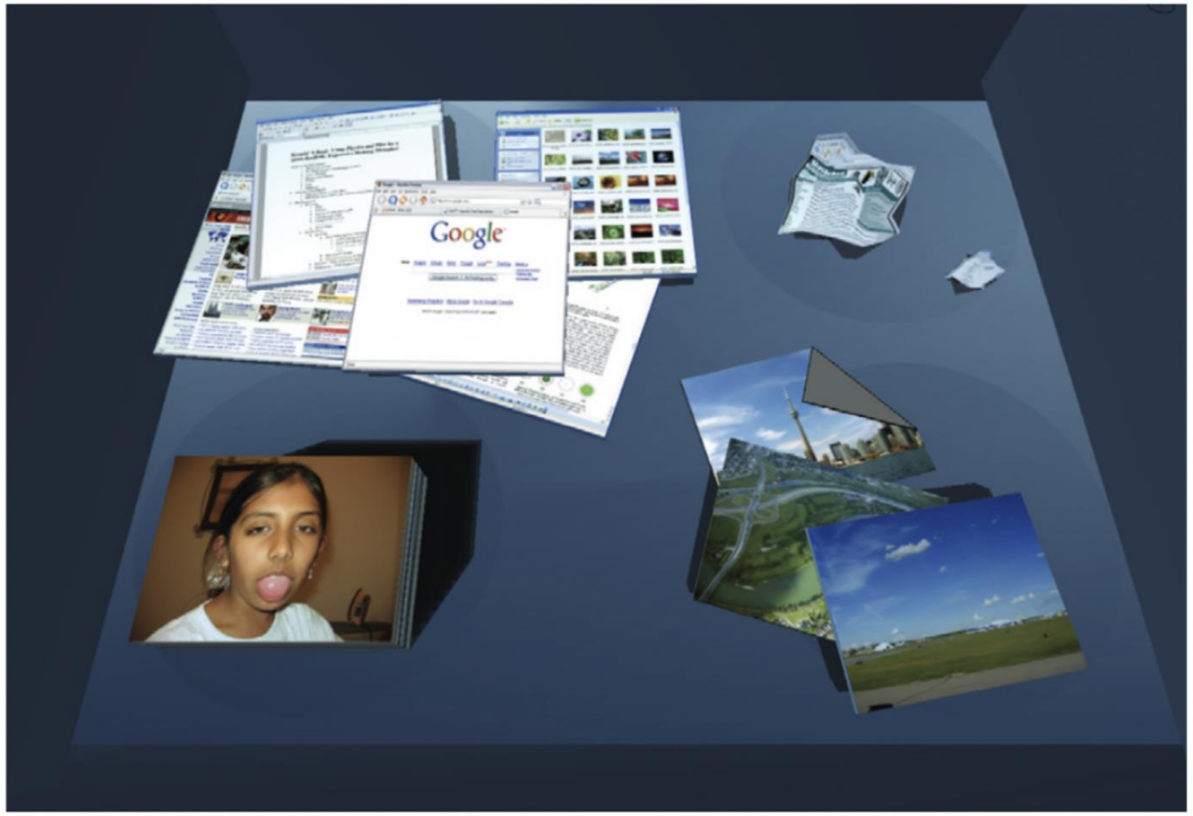
- **Graphical user interfaces** (GUIs) allow users to efficiently carry out many everyday computing tasks with ease that we now take for granted
  - Copying and pasting information
  - Starting applications by selecting them
  - Working in multiple windows at the same time
  - Using word processing and spreadsheets with easy access to hundreds of functions
- GUIs pioneered the idea that we can refer to objects by their location, rather than by their name
- The benefits of metaphors in graphical user interfaces have been much researched; examples of such metaphors are the desktop (for organizing files) and the bin (for deleting files)
- Some studies have shown that GUIs are more useful than command-line interfaces



Xerox Star's WIMP interface: Windows, Icons, Menus, and Pointing devices

Example: a graphical user interface with physics

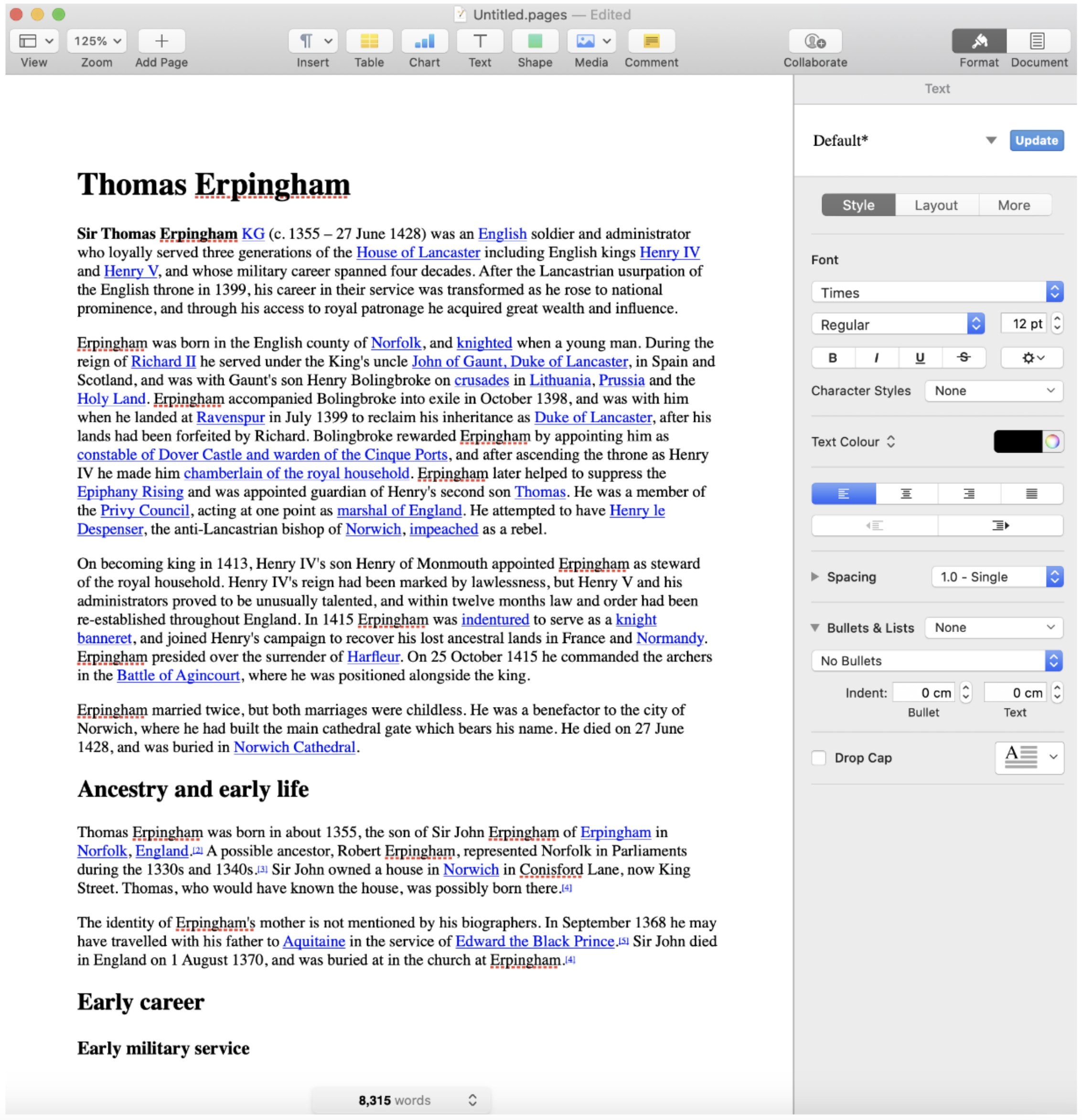
- A graphical user interface does not have to look and feel like a traditional desktop GUI
  - For example, a GUI can be driven by lightweight physics and relax some of the rather rigid organizational principles in traditional GUIs
- *BumpTop* enables users to organize files as piles of documents that can be swiftly moved about by dragging and flicking them with a pen
- As can be seen, windows, photos and files are organized informally in piles
- The movement of objects is physics-based and employs rigid body dynamics, friction and collision detection which means when objects collide, they experience semi-realistic displacement effects



A. Agarawala and R. Balakrishnan. Keepin'it real: pushing the desktop metaphor with physics, piles and the pen. In Proceedings of the Conference on Human Factors in Computing Systems, pages 1283–1292, 2006.

Visibility (1/2)

- **WYSIWYG**—What You See Is What You Get—word processors, which show the effects of choices of font families, sizes and styles directly in the document
- **Visibility of commands**, such as revealing many relevant text formatting options and tools in the toolbar to the right of the text document.



Default*

Font: Times, Regular, 12 pt

Character Styles: None

Text Colour

Spacing: 1.0 - Single

Bullets & Lists: None

No Bullets

Indent: 0 cm Bullet, 0 cm Text

Drop Cap

8,315 words

Thomas Erpingham

Sir Thomas Erpingham KG (c. 1355 – 27 June 1428) was an English soldier and administrator who loyally served three generations of the House of Lancaster including English kings Henry IV and Henry V, and whose military career spanned four decades. After the Lancastrian usurpation of the English throne in 1399, his career in their service was transformed as he rose to national prominence, and through his access to royal patronage he acquired great wealth and influence.

Erpingham was born in the English county of Norfolk, and knighted when a young man. During the reign of Richard II he served under the King's uncle John of Gaunt, Duke of Lancaster, in Spain and Scotland, and was with Gaunt's son Henry Bolingbroke on crusades in Lithuania, Prussia and the Holy Land. Erpingham accompanied Bolingbroke into exile in October 1398, and was with him when he landed at Ravenspur in July 1399 to reclaim his inheritance as Duke of Lancaster, after his lands had been forfeited by Richard. Bolingbroke rewarded Erpingham by appointing him as constable of Dover Castle and warden of the Cinque Ports, and after ascending the throne as Henry IV he made him chamberlain of the royal household. Erpingham later helped to suppress the Epiphany Rising and was appointed guardian of Henry's second son Thomas. He was a member of the Privy Council, acting at one point as marshal of England. He attempted to have Henry le Despenser, the anti-Lancastrian bishop of Norwich, impeached as a rebel.

On becoming king in 1413, Henry IV's son Henry of Monmouth appointed Erpingham as steward of the royal household. Henry IV's reign had been marked by lawlessness, but Henry V and his administrators proved to be unusually talented, and within twelve months law and order had been re-established throughout England. In 1415 Erpingham was indentured to serve as a knight banneret, and joined Henry's campaign to recover his lost ancestral lands in France and Normandy. Erpingham presided over the surrender of Harfleur. On 25 October 1415 he commanded the archers in the Battle of Agincourt, where he was positioned alongside the king.

Erpingham married twice, but both marriages were childless. He was a benefactor to the city of Norwich, where he had built the main cathedral gate which bears his name. He died on 27 June 1428, and was buried in Norwich Cathedral.

Ancestry and early life

Thomas Erpingham was born in about 1355, the son of Sir John Erpingham of Erpingham in Norfolk, England.[2] A possible ancestor, Robert Erpingham, represented Norfolk in Parliaments during the 1330s and 1340s.[3] Sir John owned a house in Norwich in Conisford Lane, now King Street. Thomas, who would have known the house, was possibly born there.[4]

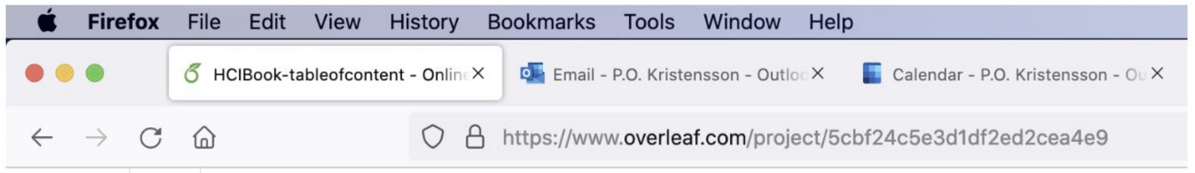
The identity of Erpingham's mother is not mentioned by his biographers. In September 1368 he may have travelled with his father to Aquitaine in the service of Edward the Black Prince.[5] Sir John died in England on 1 August 1370, and was buried at in the church at Erpingham.[4]

Early career

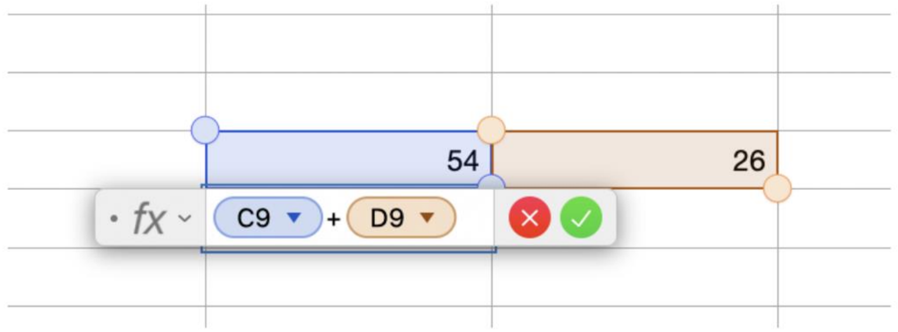
Early military service

Visibility (2/2)

- **Visibility of status** means ensuring that the graphical user interface shows the status of entities relevant to the user's goals
- **Visibility of dependencies**
  - Spreadsheets allow users to link cells together by formulas which refer to the values in other cells
  - Spreadsheets provide a mechanism for visualizing such dependencies
  - When the user clicks a cell with a formula that refers to other cells, the dependent cells are color coded to match the color coding of their role in the original formula


Firefox
File
Edit
View
History
Bookmarks
Tools
Window
Help
HCIBook-tableofcontent - Onlin
Email - P.O. Kristensson - Outlo
Calendar - P.O. Kristensson - Ou
https://www.overleaf.com/project/5cbf24c5e3d1df2ed2cea4e9

An example of visibility of status. The shield icon indicates whether there are any tracker detected on the current website. The padlock indicates whether the currently visited website uses a valid certificate; in other words, that the connection is secure. If the padlock is crossed over with a red line the website does not use a valid certificate.



An example of visibility of dependencies in a spreadsheet. Color coding is used to indicate how the cells relate to the formula.

Consistency

- To support users in learning a user interface, consistency is an important design objective
- Consistency means that representations are similar across the user interface
  - If the print command is represented with an icon showing a particular printer in one part of the interface, the same icon should be used in another
  - Similarly for interaction techniques: if one has to right-click an object to invoke a particular command, one should use the same right-click for other commands
- Is striving for consistency always a good thing? Consider the following:
  1. Consistency must be defined
  2. There is a need to identify good—as opposed to poor—consistency, which means there must be a method of identifying the most suitable consistency among several competing consistencies
  3. There must be a method of identifying when other principles are more important than consistency
- Striving for consistency is often not a useful principle
  - Instead, interface objects are better placed according to the need of the user's tasks
  - This changes the focus from properties of the user interface to task analysis and an understanding of the work context

Minimizing errors

- Errors are outcomes in a graphical user interface that are inaccurate or incorrect
- It is important to realize that errors to some extent are unavoidable and even intrinsic in human-computer interaction
    - For example, noise in human neuromuscular system will make it exceedingly difficult for a user to move a mouse cursor to precise pixel location on a high-resolution display
    - In practice, users are also bound to make mistakes due to lack of attention, training, or a misunderstanding of the user interface
- **Preempting errors:** GUIs should have a clear, simple structure, and a user interface that presents graphical elements in a consistent manner that can help preempt errors to begin with
- **Reversing outcomes:** GUIs should allow users to reverse the results if their actions are later found to be undesirable
- **Explaining outcomes and confirming actions:** an established GUI practice is to seek confirmation when the user desires to issue commands that are irreversible or may have unforeseen consequences

The principle of direct manipulation

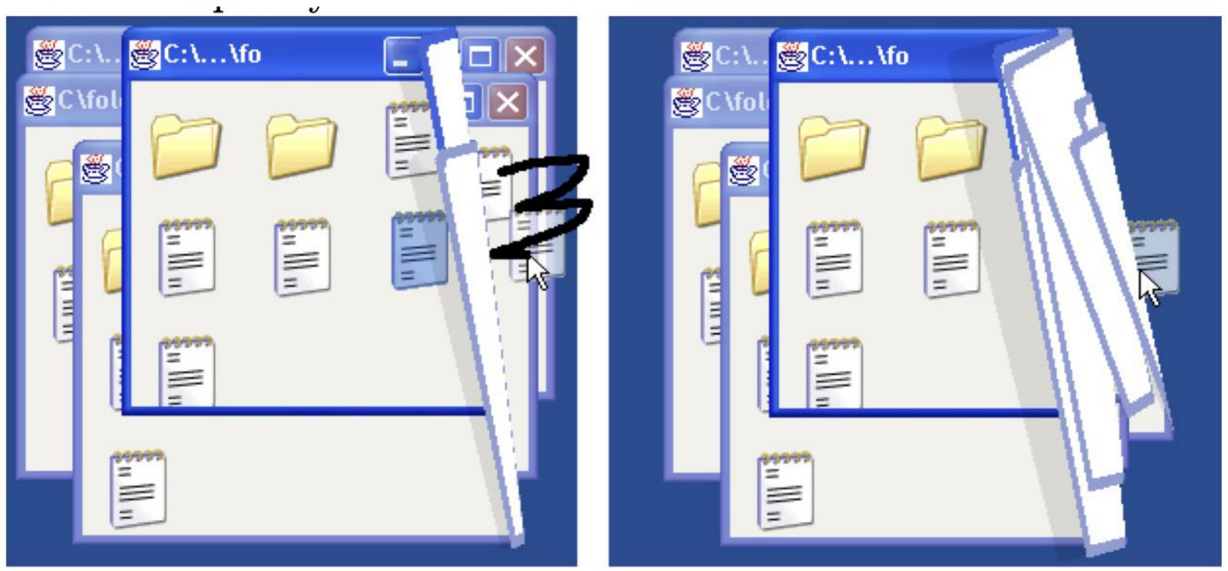
- A **direct manipulation** interface should have the following properties:
  - Visibility of the objects and actions of interest
  - Rapid, reversible, incremental actions
  - Replacement of typed commands by a pointing action on the object of interest
- These properties were later refined to the following:
  - Continuous representations of the objects and actions of interest with meaningful visual metaphors
  - Physical actions or presses of labeled buttons, instead of complex syntax
  - Rapid, reversible, incremental actions whose effects are on the objects of interest are visible immediately
- An example of an interaction using direct manipulation is deleting a file on the desktop by dragging the icon of the file to the bin

Benefits and limits of direct manipulation

- What makes interaction *direct?* **Directness** is *distance* and *engagement*
- *Distance* is the mental effort required to translate goals into actions and then evaluate their effects (the gulfs of execution and evaluation)
- *Engagement* is the locus of control within the system
  - Users should feel like being the agents within the system
  - I contrast, in command language interfaces, the user interact with a hidden intermediary that relays the commands to a third party for execution
- Direct manipulation interfaces offer a **model world**, such as the familiar desktop metaphor
- Direct manipulation supports visual recognition (identifying objects of interest, such as icons, buttons, etc.) rather than recall (remembering the names of commands)
- However, direct manipulation relies on well-designed icons, metaphors, and organizations of the user interface
- Some actions, such as referring to prior actions, and repetitive actions, are more difficult to carry out in a direct manipulation interface than a command line interface

Paper example: folding windows

- *Windows* allow expanding documents and applications into a controllable space on a display
- However, as an interactive widget, present day windows are rectangles that can be moved, resized, and closed
- What if the desktop metaphor was extended to windows?
- Consider an interaction technique for windows that allows folding them like pieces of paper
- When multiple windows are on top of each other, the user can quickly fold their corners to see what is underneath them
- The technique uses a crossing-based interface to achieve this



Reality-Based Interaction

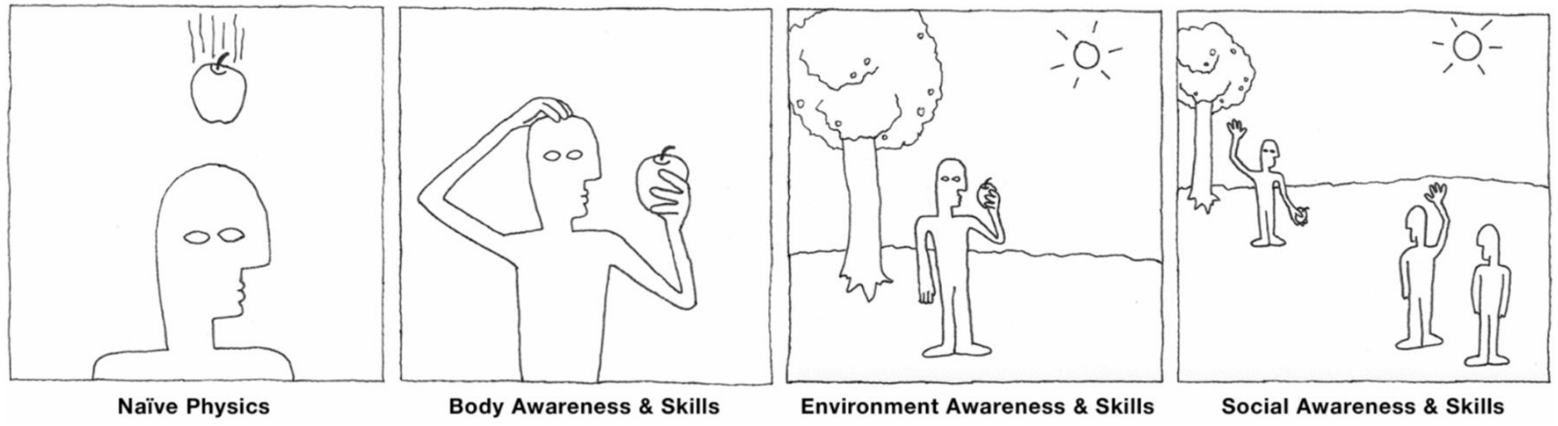
Reality-based interaction (1/2)

- **Reality-based interaction** is a framework that provides four aims for building interactive technology that better exploits and supports our capabilities

1. **Naive physics:** "people have common sense knowledge about the physical world"
   - What have been called tangible user interfaces move the control of computers into the physical world
2. **Body-awareness and skills:** "people have an awareness of their own physical bodies and possess skills for controlling and coordinating their bodies"
   - Interaction in mixed reality is widely based on the idea that you move your limbs as you would do outside mixed reality
3. **Environment awareness and skills:** "people have a sense of their surroundings and possess skills for negotiating, manipulating, and navigating within their environment"
   - The use of touch in user interfaces, rather than pointing with an input device such as the mouse, draw on environment skills
4. **Social awareness and skills:** "people are generally aware of others in their environment
   and have skills for interacting with them"
   - An example of attention to such awareness and skills is the interest in video communication systems of getting the gaze direction of a remote person to appear correct

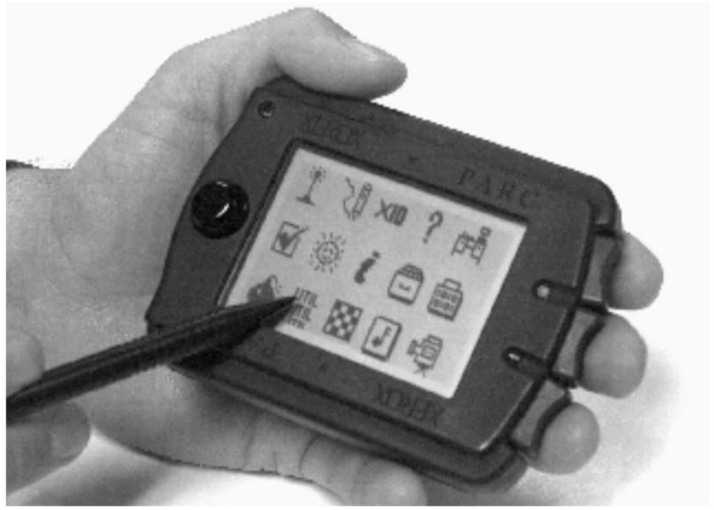
Reality-based interaction (2/2)

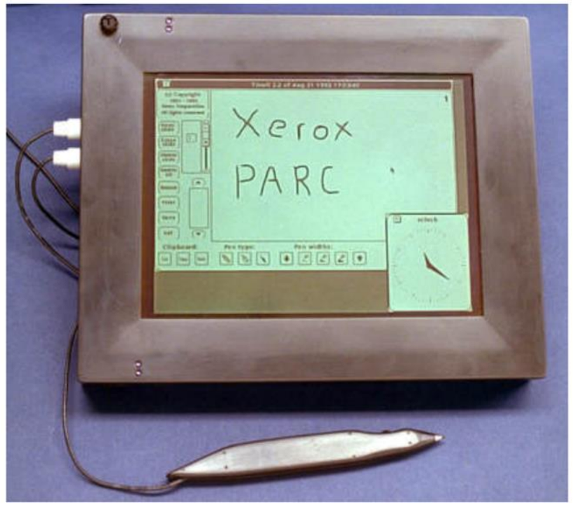
- In practice, reality-based interaction boils down to research on interfaces that in various ways break key assumptions about desktop-based computing, such as:
- **Mobile interaction** that breaks the assumption that interaction means that a user is primarily stationary
- **Multimodal interaction** that breaks the assumption that interaction with computer systems primarily involves input using one modality and output using another, such as a visual display
- **Ubiquitous computing** that breaks the assumption that users must command computers to get things done
- **Mixed reality and tangible interaction** that break the assumption that virtual reality and physical reality are separate

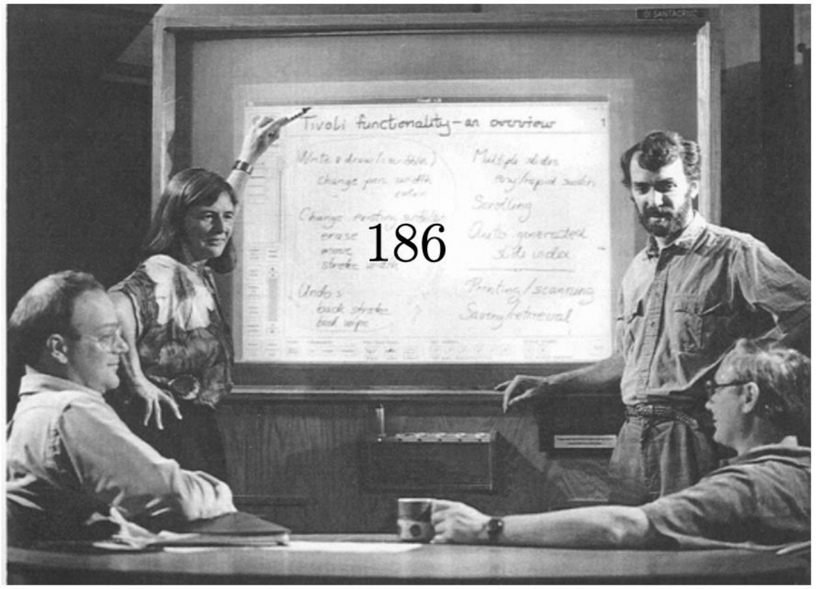


Example: An early vision of ubiquitous computing

- Marc Weiser presented a prominent vision of going beyond desktop computing called **ubiquitous computing**
- Computing should come in different sizes, each suitable for a particular task
- Weiser noted that in any given room, there are hundreds of differently sized writing and display surfaces
- Weiser focused on three scales: (1) tabs; pads; and boards and stated:
  - "Look around you: at the inch scale include wall notes, titles on book spines, labels on controls, thermostats and clocks, as well as small pieces of paper. Depending upon the room you may see more than a hundred tabs, ten or twenty pads, and one or two boards."







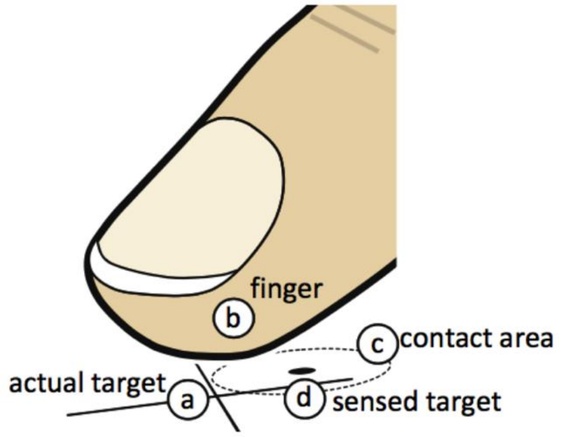
M. Weiser. The computer of the 21st century. *Scientific American* **265**(3):94–104, 1991.

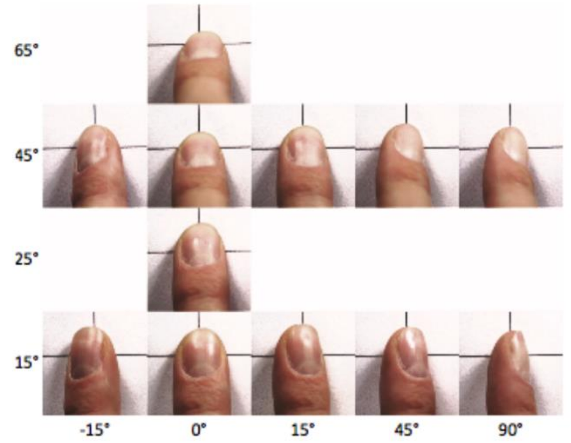
Characteristics of mobile user interfaces

- A small size limits input
  - Typical solutions: touchscreens, speech, and physical button
  - Also: radar-based sensing, computer vision, accelerometers and gyros
- A small size limits output
  - Output devices are not limited to touchscreen and audio
  - Mobile computers can use mobile projectors and they can connect to head- mounted displays and glasses, on-body haptic actuators, and even link to other near by pervasive displays
- A small size affects the visual hierarchy
  - Design for mobile interfaces uses saliency and containment to communicate the most important areas of the display
- A small size affects the viewport
  - Since the viewport is smaller, it requires interaction techniques to support elementary interactions, such as scrolling or changing pages
- A small size enables portability
  - Mobile interaction often takes place while users are in transition: walking, commuting, or just multitasking

Small screen interaction

- Limited viewport size of the display
- Fingers occlude the display
  - *Touch offset* warps the selection point from the inferred touch point to make it easier to select a small target
  - In text entry, a caret marks the location to which characters will be inserted: *caret offset* refers to the offset to the touch location in pixels
  - *Magnifying glass* is a mode in which the region of the display under the finger is zoomed in in a warped area of the display
- Imprecise touch selection
  - If you press a touchscreen with your fingertip, no matter how gently you try to do that, the area that contacts the surface is much larger than a single pixel





Interactive objects presented on mobile displays are small, how do users select them? A study found that users use the perceived features of the finger, especially its outline and the nail, to touch a target, here marked with crosshair. Users align perceived features in a hypothesized top-down perspective. This leads to systematic biases that can be corrected in software-side with a proper model.

Mobility

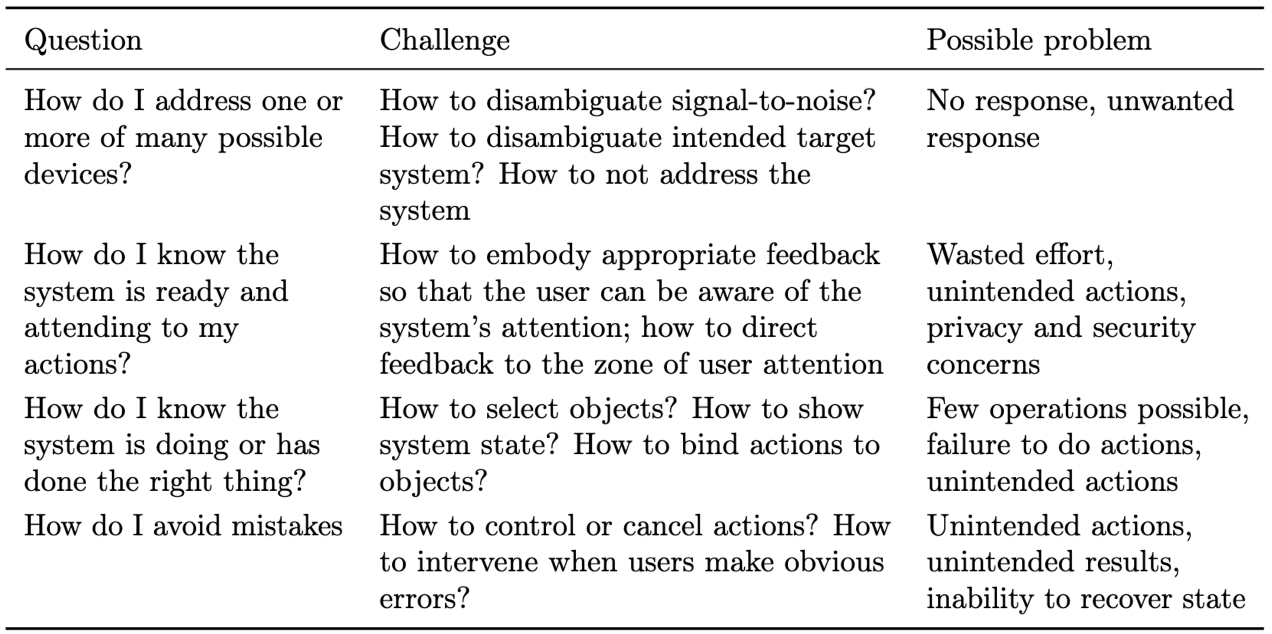
- **Mobile context:** factors that are relevant in a situation that may affect interaction with the computer
- **Locomotion:** walking or moving by other means
  - Walking requires visual attention which now needs to be shared between the mobile device and the environment
- **Navigation:** finding one’s way in an environment
  - This requires not only the ability to localize oneself but also the ability to choose where to go next
  - While moving in an environment, users’ capability for interaction is degraded
- **Environmental noise:** physical perturbations, such as tremble in a metro, as well as visual noise, such as blinking lights, and auditory noise
  - Such factors can degrade a user’s sensori-motor performance
- **Multitasking:** users are strategically allocating limited cognitive resources among tasks
  - We share visual attention to events in our environments while using mobiles
  - We plan and reason about routes when walking, which may distract with planning the mobile device requires
- **Social context:** when mobile, we are constantly positioning ourselves physically and socially among other people
  - Consider the way we queue, take lanes, or sit: there is an invisible ’order’ that we observe, which also affects the way we use our devices

Ubiquitous computing (ubicomp)

- Embedded input and output
    - Input from the environment:, such as keycards, location of devices and people
    - Output from the environment, such as *calm* environments that display output in an unobtrusive manner, and interfaces that react to users' proximity to the devices
- Context awareness
    - Context-aware systems use context to provide relevant information or to supply relevant services for the user's task
    - Context can be geographical location, a current activity, or emotional states of the user
- Ways of using context
    1. Offering different information or services depending on context
    2. Services may run automatically or proactively based on users' context
    3. Context may be tagged so that it can be acted upon by the user

Natural and implicit interaction

- A main point of departure for many ubicomp systems is that interaction should be natural in its context, for example, using gestures, movement, and speech
- A related characteristic of ubicomp is implicit interaction
- Much interaction in ubicomp is command-free, using sensed input
- The concerns around most interaction techniques for ubicomp, then revolve around whether users feel as the agents behind interactions and whether it is clear to them that they are indeed interaction



Question	Challenge	Possible problem
How do I address one or more of many possible devices?	How to disambiguate signal-to-noise? How to disambiguate intended target system? How to not address the system	No response, unwanted response
How do I know the system is ready and attending to my actions?	How to embody appropriate feedback so that the user can be aware of the system's attention; how to direct feedback to the zone of user attention	Wasted effort, unintended actions, privacy and security concerns
How do I know the system is doing or has done the right thing?	How to select objects? How to show system state? How to bind actions to objects?	Few operations possible, failure to do actions, unintended actions
How do I avoid mistakes	How to control or cancel actions? How to intervene when users make obvious errors?	Unintended actions, unintended results, inability to recover state

Sample questions about implicit interaction in ubicomp systems

V. Bellotti, M. Back, W. K. Edwards, R. E. Grinter, A. Henderson, and C. Lopes. Making sense of sensing systems: five questions for designers and researchers. In Proceedings of the Conference on Human Factors in Computing Systems, pages 415–422, 2002.

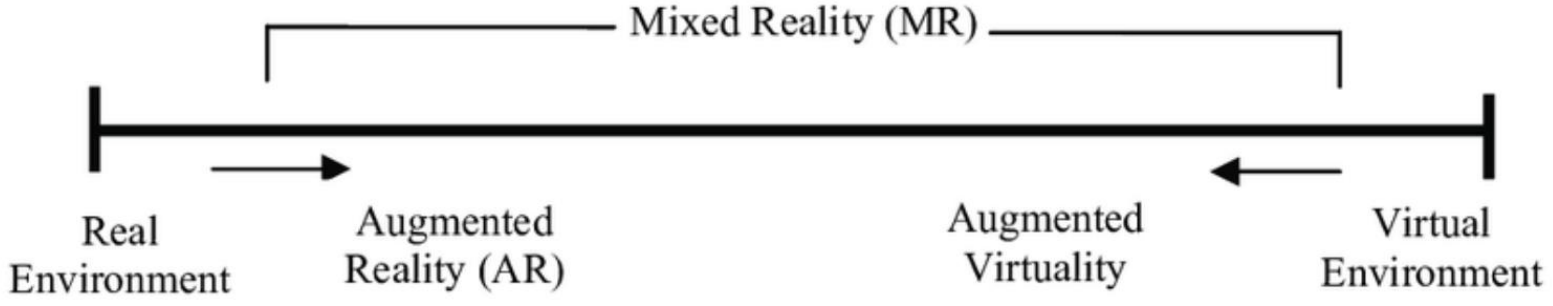
Example: the challenge of sensing complex states

- One goal of ubicomp has been sensing the internal states of people
- The research area of *affective computing* has focused on sensing human affect and adapting interactive systems based on the sensed information
- The researchers explored how to decode physiological signals from a user into a prediction of an affective state for that user
    - They collected about a month of data from a person who tried to experience particular emotions
    - The person was asked to go through eight emotions in turn, including anger, hate, grief, and joy
- Four measures were collected
    1. Facial muscle activity
    2. Blood pressure
    3. Skin conductance, which is increased by sweat on parts of the hands
    4. Respiration rate
- Based on these measures, the authors developed a variety of models for accurately predicting emotions, solving variation among days of the features that indicated the relevant emotions
- This study was an early demonstration that some variation in measures can be dealt with

R. W. Picard, E. Vyzas, and J. Healey. Toward machine emotional intelligence: analysis of affective physiological state. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **23**(10):1175–1191, 2001.

Mixed reality

- **Mixed reality** (MR) refers to user interface technology that mixes virtual and real content to create a new, augmented experience that is interactive and appears authentic to the user
- The two defining aspects of MR are:
    1. A programmatic association between virtual content and the real world
    2. An interactive display of that association
- The *mixed reality continuum* defines the range of options available between the real world and a virtual reality

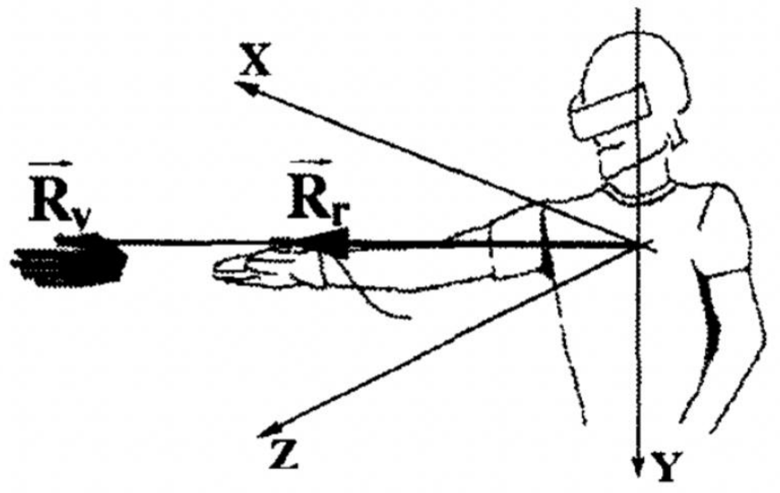


Virtual reality

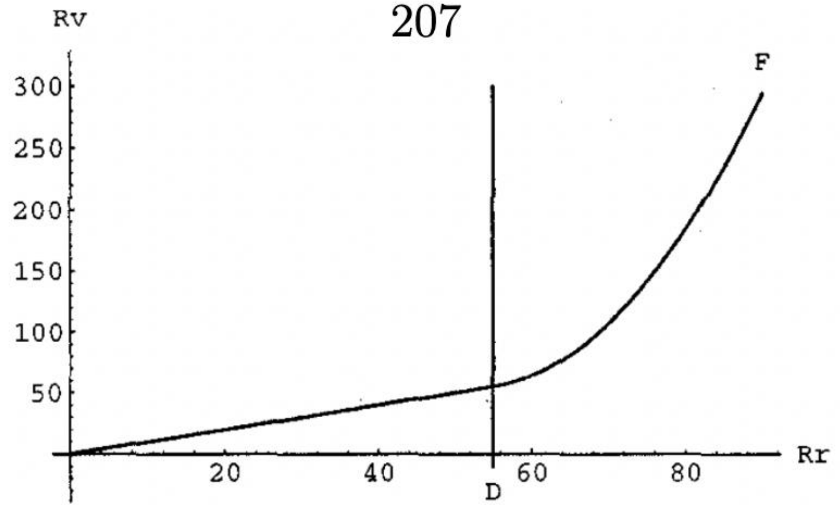
- Virtual reality (VR) refers to an interactive environment that responds to user's actions, for example, via locomotion of camera control
- A *reality* is more than just virtual content
  - It is usually understood as a space that binds together content
- A virtual reality system is defined by seven features:
  1. Blocking out sensory impressions generated by the real world (such as VR headsets)
  2. Continuously updated computer graphics creating a sense of immersion
  3. Tracking of user position and orientation
  4. Software for modeling the relationship between the virtual and the real
  5. Synthesized sound and possibly haptic sensations
  6. Input devices that enable interactions with virtual objects
  7. Interaction techniques that substitute real interactions (such as, hand movement)

Example: interacting with objects at a distance (1/2)

- A problem in virtual reality is that objects can be further away than the user can reach
- The metaphor underpinning the *Go-Go* technique is that the user's arm grows nonlinearly in VR when the user reaches out to touch, grasp, or otherwise interact with a virtual object that is out of the user's reach
- When the user interacts within a set threshold the virtual hand and the user's real hand are mapped to the same place
- When the user extends their arm passed a threshold the virtual arm of the user, and thus the position of the user's virtual hand, starts to grow nonlinearly, allowing the user to reach virtual objects that are further away than the user can normally reach



207

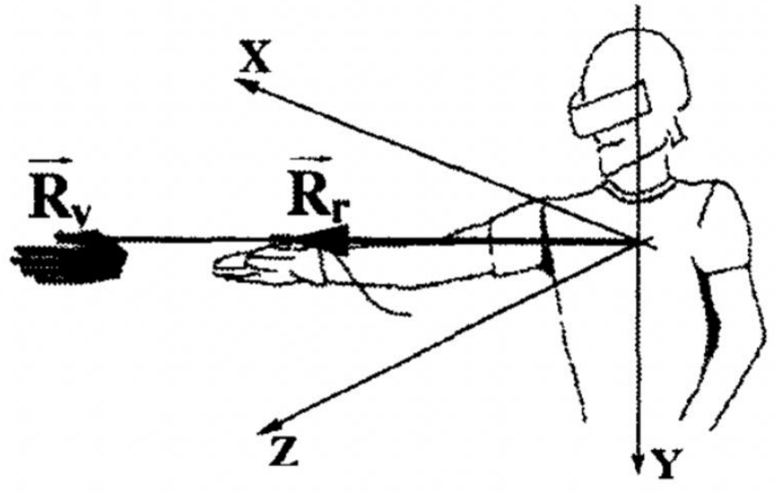


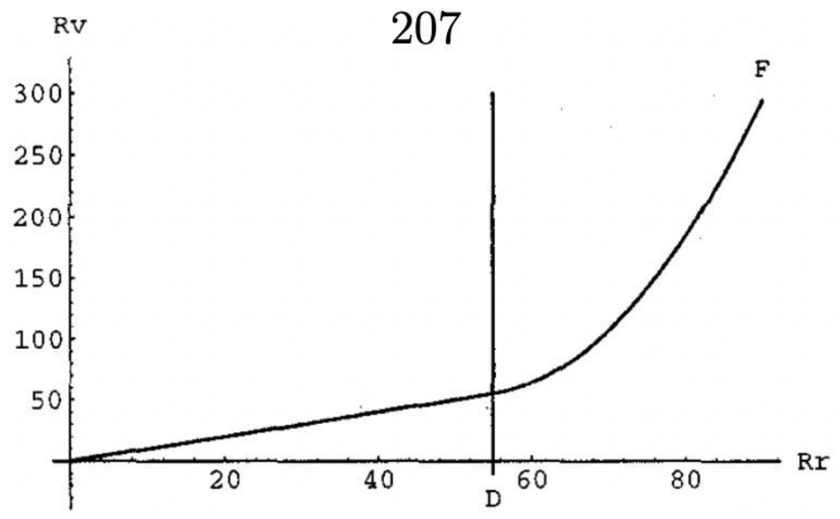
Example: interacting with objects at a distance (2/2)

- The user's real hand position, mapped by the vector $\mathbf{R}_\mathrm{r}$ (with length R_r) in the diagram below is mapped to a virtual hand position parameterized by $\mathbf{R}_\mathrm{v}$ (with length R_v)
- When the user moves their hand, the virtual hand is mapped as follows:

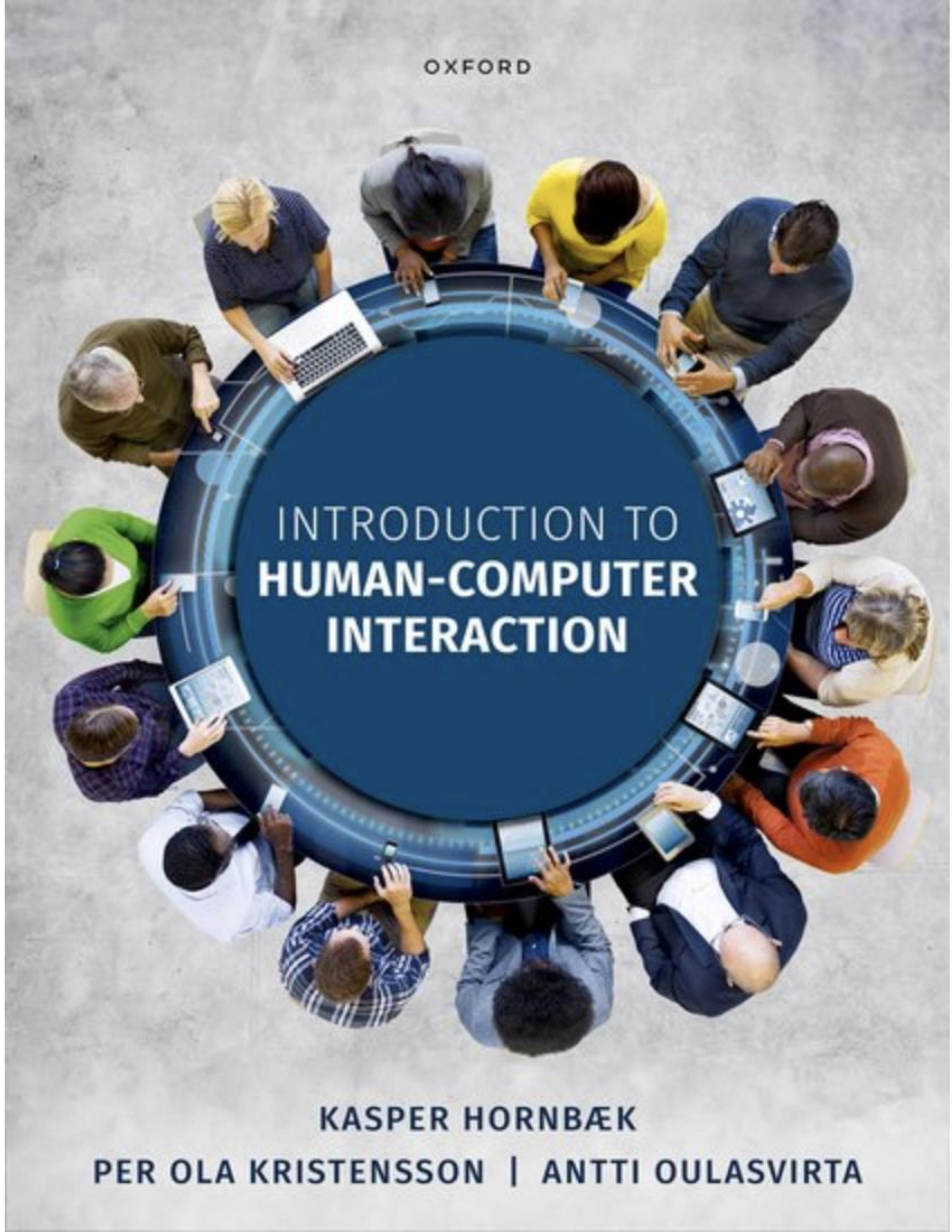
$$R_v = F(R_r) = \begin{cases} R_r & \text{if } R_r < D \\ R_r + k(R_r - D)^2 & \text{otherwise} \end{cases}$$

- k is a coefficient between zero and unity controlling the nonlinear rate of expansion of the virtual arm
- D controls the length the user's real arm has to be expanded to for the virtual arm to start expanding nonlinearly
- In the original paper D is set to $\frac{2}{3}$ of the user's real arm length
- Note that when R_r is below D, the mapping between R_r and R_v is linear but when the R_r exceeds D, the mapping smoothly transitions from a linear to a nonlinear mapping





- Open access (PDF at link)
- Further reading:
  - Part V: User Interfaces
  - Chapters 23–29



Appendix

Worked example of applying the cognitive dimensions of notation

- Apply cognitive dimensions of notation to analyse how two programming environments would support programming an implementation for a particular task.
  - The first programming environment is Java in Eclipse.
  - The second programming environment is the spreadsheet in Microsoft Excel (or Open Office). Note that we mean pure spreadsheets here, not macros or built-in scripting.
- The programming task to consider is the implementation of mathematical expressions where the arguments may be scalar or vectors and the functions used are a subset of all usual mathematical operators such as addition, subtraction etc. plus the following: exponential, square root, absolute value, and standard deviation.
- Use cognitive dimensions of notation to analyse how these two different programming environments would support the programmer for this task.

Viscosity

- *Resistance to change*
- "A viscous system needs many user actions to accomplish one goal. Changing all headings to upper-case may need one action per heading. (Environments containing suitable abstractions can reduce viscosity.) We distinguish repetition viscosity, many actions of the same type, from knock-on viscosity, where further actions are required to restore consistency." (Blackwell and Green 2003)
- **Eclipse/Java:**
  - Is it easy to make changes to the code?
- **Excel/Spreadsheet:**
  - How easy is it to change values / formulas vs. refactoring expressions?

Visibility

- *Ability to view components easily*
- “Systems that bury information in encapsulations reduce visibility. Since examples are important for problem-solving, such systems are to be deprecated for exploratory activities; likewise, if consistency of transcription is to be maintained, high visibility may be needed.” (Blackwell and Green 2003)
- **Eclipse/Java:**
  - How easy is to see the result of edits?
  - How easy is it to view program structures?
- **Excel/Spreadsheet:**
  - How easy is it to see the results of edits?
  - How easy is it to view expressions?

Premature commitment

- *Constraints on the order of doing things*
- “Self-explanatory. Examples: being forced to declare identifiers too soon; choosing a search path down a decision tree; having to select your cutlery before you choose your food.” (Blackwell and Green 2003)
- **Eclipse/Java:**
  - How easy is it to fundamentally change the program structures?
- **Excel/Spreadsheet:**
  - How easy is it to fundamentally change expressions and calculations once initial cell layouts have been chosen?

Hidden dependencies

- *Important links between entities are not visible*
- “If one entity cites another entity, which in turn cites a third, changing the value of the third entity may have unexpected repercussions. Examples: cells of spreadsheets; style definitions in Word; complex class hierarchies; HTML links. There are sometimes actions that cause dependencies to get frozen – e.g. soft figure numbering can be frozen when changing platforms; these interactions with changes over time are still problematic in the framework.” (Blackwell and Green 2003)
- **Eclipse/Java:**
  - How easy is it to see hidden dependencies and how much do they affect program design?
- **Excel/Spreadsheet:**
  - How easy is it to see hidden dependencies and how much do they affect expressions and cell layouts?

Role-expressiveness

- *The purpose of an entity is readily inferred*
- "Role-expressive notations make it easy to discover why the author has built the structure in a particular way; in other notations each entity looks much the same and discovering their relationships is difficult. Assessing role-expressiveness requires a reasonable conjecture about cognitive representations." (Blackwell and Green 2003)
- **Eclipse/Java:**
  - How easy is it to see the type of a) a variable, b) text in the editor?
- **Excel/Spreadsheet:**
  - How easy is it to understand the type of cell and what it performs?

Error-proneness

- *The notation invites mistakes and the system gives little protection*
- “Enough is known about the cognitive psychology of slips and errors to predict that certain notations will invite them. Prevention (e.g. check digits, declarations of identifiers, etc) can redeem the problem.” (Blackwell and Green 2003)
- **Eclipse/Java:**
  - How easy is it to foresee programming errors in the Eclipse IDE? What about ensuring accurate method invocations in Java?
- **Excel/Spreadsheet:**
  - How easy is to foresee errors in expressions? What about validating data for a cell?

Abstraction

- *Types and availability of abstraction mechanisms*
- “Abstractions (redefinitions) change the underlying notation. Macros, data structures, global find-and-replace commands, quick-dial telephone codes, and word-processor styles are all abstractions. Some are persistent, some are transient. Abstractions, if the user is allowed to modify them, always require an abstraction manager – a redefinition sub-device. It will sometimes have its own notation and environment (e.g. the Word style sheet manager) but not always (for example, a class hierarchy can be built in a conventional text editor).” (Blackwell and Green 2003)
- **Eclipse/Java:**
  - How are abstractions supported in Java? How are they further supported in Eclipse?
- **Excel/Spreadsheet:**
  - How are abstractions handled in spreadsheets?

Secondary notation

- *Extra information in means other than formal syntax*
- "Users often need to record things that have not been anticipated by the notation designer. Rather than anticipating every possible user requirement, many systems support secondary notations that can be used however the user likes. One example is comments in a programming language, another is the use of colours or format choices to indicate information additional to the content of text." (Blackwell and Green 2003)
- **Eclipse/Java:**
  - How are secondary notations supported in Java and Eclipse?
- **Excel/Spreadsheet:**
  - How are secondary notations supported in spreadsheets?

Closeness of mapping

- *Closeness of representation to domain*
- "How closely related is the notation to the result it is describing?" (Blackwell and Green 2003)
- **Eclipse/Java:**
  - How close is the mapping of Java in Eclipse to the functions we wish to implement? What notational features does Java not include that could make the functions we wish to implement appear closer to their original domain?
- **Excel/Spreadsheet:**
  - How close is the mapping of spreadsheet expressions to the functions we wish to implement?

Consistency

- *Similar semantics are expressed in similar syntactic forms*
- "Users often infer the structure of information artefacts from patterns in notation. If similar information is obscured by presenting it in different ways, usability is compromised." (Blackwell and Green 2003)
- **Eclipse/Java:**
  - How is consistency enforced or supported in Java? Does Eclipse provide further support?
- **Excel/Spreadsheet:**
  - How is consistency enforced in spreadsheet expressions?

Diffuseness

- *Verbosity of language*
- “Some notations can be annoyingly long-winded, or occupy too much valuable “real-estate” within a display area. Big icons and long words reduce the available working area.” (Blackwell and Green 2003)
- **Eclipse/Java**:
  - How verbose is Java? Does Eclipse affect the verboseness of Java?
- **Excel/Spreadsheet:**
  - How verbose are spreadsheet expressions?

Hard mental operations

- *High demand on cognitive resources*
- “A notation can make things complex or difficult to work out in your head, by making inordinate demands on working memory, or requiring deeply nested goal structures.” (Blackwell and Green 2003)
- **Eclipse/Java:**
  - How much does the user need to understand to use Java? How much extra effort is required to learn Eclipse? How does the user know which methods and APIs to use?
- **Excel/Spreadsheet:**
  - How does the user know which spreadsheet expressions to use?

Provisionality

- *Degree of commitment to actions or marks*
- “Even if there are hard constraints on the order of doing things (premature commitment), it can be useful to make provisional actions such as recording potential design options, sketching, or playing “what-if” games. Not all notational systems allow users to fool around or make sketchy markings.” (Blackwell and Green 2003)
- **Eclipse/Java:**
  - How much code needs to be written to try out an idea? How easy is to try out an idea in general?
- **Excel/Spreadsheet:**
  - How easy is to try out an idea in general?

Progressive evaluation

- *Work-to-date can be checked at any time*
- "Evaluation is an important part of a design process, and notational systems can facilitate evaluation by allowing users to stop in the middle to check work so far, find out how much progress has been made, or check what stage in the work they are up to. A major advantage of interpreted programming environments such as BASIC is that users can try out partially completed versions of the product program, perhaps leaving type information or declarations incomplete." (Blackwell and Green 2003)
- **Eclipse/Java:**
  - How does Java support progressive evaluation? What about if Java code is written in Eclipse?
- **Excel/Spreadsheet:**
  - How is progressive evaluation supported in spreadsheets?