MLMI 10
Designing Intelligent Interactive Systems

Cognitive Ergonomics and Other User Interface Issues (Embodiment Design)

Per Ola Kristensson

Contents

- Ergonomics
- Heuristics
- KLM-GOMS
- Interaction techniques and devices
- Fitts' law (human performance laws)
- Designing intelligent interactive systems
- Eye-tracking design

Ergonomics

Physical Ergonomics

- Anthropometric data is result of the physical measurements of many individual humans
- Provides statistical data about the distributions of physical attributes in the population:
  - Size, strength, mobility, flexibility, endurance, vision, hearing, handedness, clothing bulk, task performance, etc.
- Critical when designing for humans (industrial design, clothing, architecture, etc.)
- Enables a product to be adapted for the physical needs of the population
- Anthropometric data needs to be updated as people change (due to for instance changing lifestyles) and due to new demands on people placed by new systems

Physical Ergonomics: Design Strategies

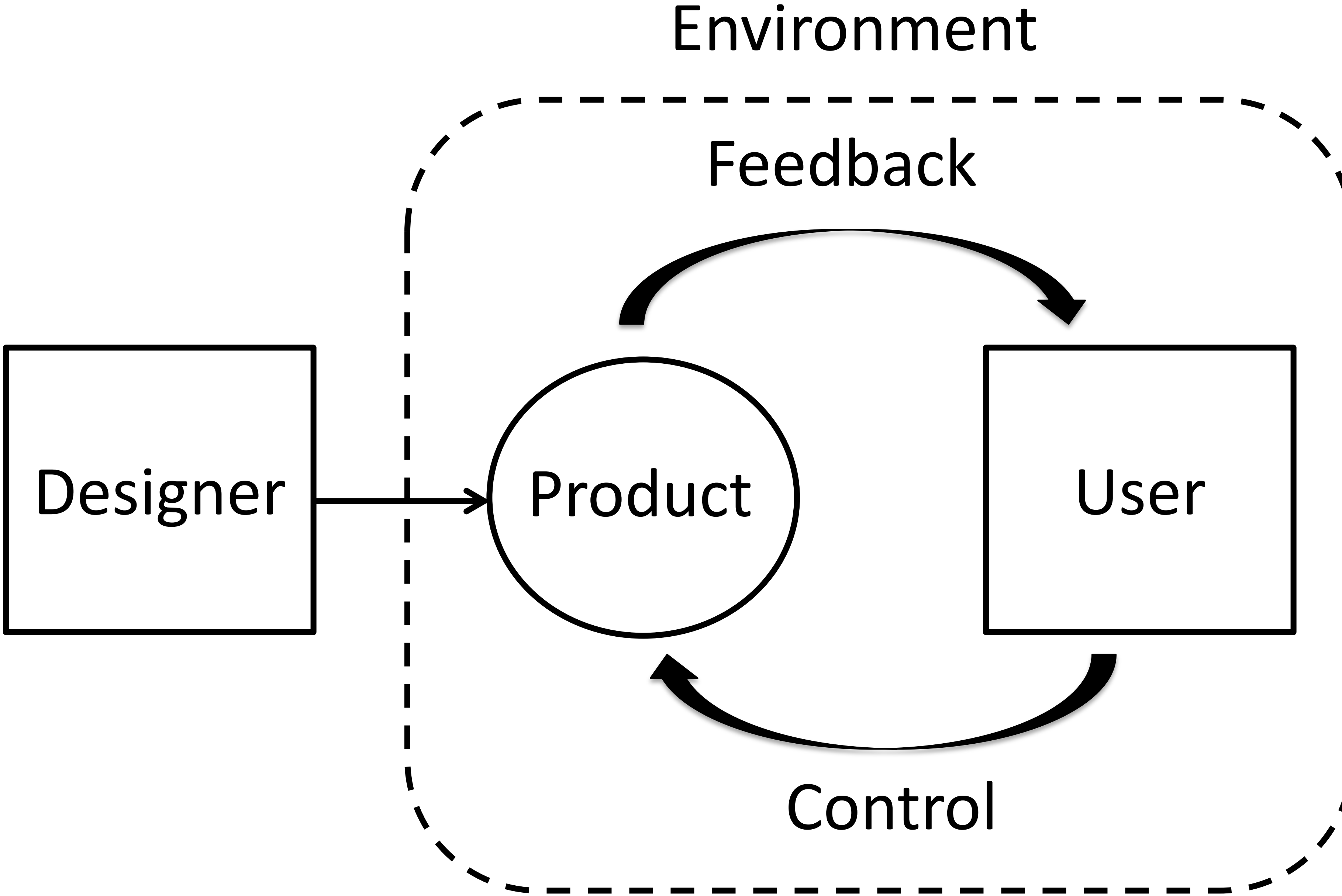
- Design for the average user
  - There is no individual who is an average user
  - Risk: the product might suit no one
- Design for the extreme user
  - Assume other users can adapt
- Design for adjustability of the product
  - Continuous or discrete adjustments?
  - Which user groups are included?
  - Which user groups are excluded?
- Design for variety in a range of products
  - Design a range of products
  - Allow the user to select their size

Physical Ergonomics: Design Process

1. Identify the critical dimensions
2. Specify the user population (the intended audience)
3. Determine the design principles for each feature
4. Select the range of sizes to accommodate
5. Find suitable data
6. Add allowances
7. Test with real people
8. (Design a range of products)
9. (Allow users to select their size)

Cognitive Ergonomics

- Even if the user can physically use a wearable device, they need to know what it is, how it works and how they should use it
- If designers apply some basic principles of cognitive ergonomics, this can assist users in developing the required knowledge
- Need to acknowledge an interaction loop between the user and the product (see next slide)


Environment
Feedback
Designer
Product
User
Control

Mental Models

- A **mental model** is a user's idea (or model) of how a product works
- Mental models are influenced by:
- Affordances
  - With respect to product design the affordances of objects allow certain actions or operations by the user
- Constraints
  - Constraints (or perceived obstacles and barriers) place limits on what actions can (or will) be performed
- Mappings
  - The relationships between a user's actions and the corresponding behaviour of the system
  - Strong mappings reduce the need for learning
- Conventions
  - The established norms for how things are
- Feedback
  - Feedback confirms (or otherwise) that the users' action had the desired results
  - Consider: multiple senses, time delay effects
- Metaphors
  - Metaphors suggest that the designed system is analogous to some other system (e.g. 'computer desktop')
  - The other system must already be understood
  - It must be analogous in ways that are important

Cognitive Ergonomics: Basic Principles

- Mental models contribute to systems that are:

1. Learnable
2. Efficient
3. Memorable
4. Error tolerant and error preventative
5. Satisfying

Cognitive Ergonomics: Basic Processes

1. Identify users
2. Find out what they need / want / expect
3. Conduct user task analysis
4. Identify similar systems
5. Test

User Task Analysis

- Decomposition, grouping and ordering of tasks
- Include physical and cognitive elements
- Identify opportunities to:
  - Eliminate tasks
  - Simplify tasks
  - Increase error tolerance
  - Decrease order dependency
  - Improve feedback
- And so on…

Inclusive Design

- Capability loss
  - Sensory: vision, hearing, touch, smell, taste
  - Cognitive: perception, attention, memory, knowledge, experience, problem solving
  - Motor: static and functional anthropometrics, biomechanics
- Contextual disability
  - Able-bodied loss of capabilities due to the environment, such as temperature, glare, noise, vibration, stress, fatigue, etc.
- Inclusive design is a wide range of design strategies that are aimed to enable designs for a heterogeneous user base with disabilities
- See separate resources on Moodle

Heuristics

Heuristics, guidelines, principles, etc.

- Sometimes called heuristic evaluation
- There are many heuristics, guidelines and principles
- Some are theoretically grounded (often called heuristics)
- Some are practical (for example, vendor-guidelines)
- **Two** prominent sets of heuristics are Nielsen (1994) and Shneiderman and Plaisant (2010)

Two prominent sets of heuristics

Nielsen's (1994) list of ten heuristics in his book *Usability Engineering*:
http://www.useit.com/papers/heuristic/heuristic_list.html

1. Visibility of system status
2. Match between system and the real world
3. User control and freedom
4. Consistency and standards
5. Error prevention
6. Recognition rather than recall
7. Flexibility and efficiency of use
8. Aesthetic and minimalist design
9. Help users recognize, diagnose, and recover from errors
10. Help and documentation

Shnedierman and Plaisant's (2010) list of eight golden rules in *Designing the User Interface* (pp. 88-89)

1. Strive for consistency
2. Cater to universal usability
3. Offer informative feedback
4. Design dialogues to yield closure
5. Prevent errors
6. Permit easy reversal of actions
7. Support internal locus of control
8. Reduce short-term memory load

Comparing Nielsen (1994) against Shneiderman and Plaisant (2010)

Two prominent sets of heuristics

Nielsen's (1994) list of ten heuristics in his book *Usability Engineering*:
http://www.useit.com/papers/heuristic/heuristic_list.html

1. **Visibility of system status**
2. Match between system and the real world
3. User control and freedom
4. Consistency and standards
5. Error prevention
6. Recognition rather than recall
7. Flexibility and efficiency of use
8. Aesthetic and minimalist design
9. Help users recognize, diagnose, and recover from errors
10. Help and documentation

Shnedierman and Plaisant's (2010) list of eight golden rules in *Designing the User Interface* (pp. 88-89)

1. Strive for consistency
2. Cater to universal usability
3. **Offer informative feedback**
4. **Design dialogues to yield closure**
5. Prevent errors
6. Permit easy reversal of actions
7. Support internal locus of control
8. Reduce short-term memory load

Two prominent sets of heuristics

Nielsen's (1994) list of ten heuristics in his book *Usability Engineering*:
http://www.useit.com/papers/heuristic/heuristic_list.html

1. Visibility of system status
2. ~~**Match between system and the real world**~~
3. User control and freedom
4. Consistency and standards
5. Error prevention
6. Recognition rather than recall
7. Flexibility and efficiency of use
8. Aesthetic and minimalist design
9. Help users recognize, diagnose, and recover from errors
10. Help and documentation

Shnedierman and Plaisant's (2010) list of eight golden rules in *Designing the User Interface* (pp. 88-89)

1. Strive for consistency
2. Cater to universal usability
3. Offer informative feedback
4. Design dialogues to yield closure
5. Prevent errors
6. Permit easy reversal of actions
7. Support internal locus of control
8. Reduce short-term memory load

Two prominent sets of heuristics

Nielsen's (1994) list of ten heuristics in his book *Usability Engineering*:
http://www.useit.com/papers/heuristic/heuristic_list.html

1. Visibility of system status
2. Match between system and the real world
3. **User control and freedom**
4. Consistency and standards
5. Error prevention
6. Recognition rather than recall
7. Flexibility and efficiency of use
8. Aesthetic and minimalist design
9. Help users recognize, diagnose, and recover from errors
10. Help and documentation

Shnedierman and Plaisant's (2010) list of eight golden rules in *Designing the User Interface* (pp. 88-89)

1. Strive for consistency
2. Cater to universal usability
3. Offer informative feedback
4. Design dialogues to yield closure
5. Prevent errors
6. **Permit easy reversal of actions**
7. **Support internal locus of control**
8. Reduce short-term memory load

Two prominent sets of heuristics

Nielsen’s (1994) list of ten heuristics in his book *Usability Engineering*:
http://www.useit.com/papers/heuristic/heuristic_list.html

1. Visibility of system status
2. Match between system and the real world
3. User control and freedom
4. **Consistency and standards**
5. Error prevention
6. Recognition rather than recall
7. Flexibility and efficiency of use
8. Aesthetic and minimalist design
9. Help users recognize, diagnose, and recover from errors
10. Help and documentation

Shnedierman and Plaisant’s (2010) list of eight golden rules in *Designing the User Interface* (pp. 88-89)

1. **Strive for consistency**
2. Cater to universal usability
3. Offer informative feedback
4. Design dialogues to yield closure
5. Prevent errors
6. Permit easy reversal of actions
7. Support internal locus of control
8. Reduce short-term memory load

Two prominent sets of heuristics

Nielsen's (1994) list of ten heuristics in his book *Usability Engineering*:
http://www.useit.com/papers/heuristic/heuristic_list.html

1. Visibility of system status
2. Match between system and the real world
3. User control and freedom
4. Consistency and standards
5. **Error prevention**
6. Recognition rather than recall
7. Flexibility and efficiency of use
8. Aesthetic and minimalist design
9. Help users recognize, diagnose, and recover from errors
10. Help and documentation

Shnedierman and Plaisant's (2010) list of eight golden rules in *Designing the User Interface* (pp. 88-89)

1. Strive for consistency
2. Cater to universal usability
3. Offer informative feedback
4. Design dialogues to yield closure
5. **Prevent errors**
6. Permit easy reversal of actions
7. Support internal locus of control
8. Reduce short-term memory load

Two prominent sets of heuristics

Nielsen's (1994) list of ten heuristics in his book *Usability Engineering*:
http://www.useit.com/papers/heuristic/heuristic_list.html

1. Visibility of system status
2. Match between system and the real world
3. User control and freedom
4. Consistency and standards
5. Error prevention
6. **Recognition rather than recall**
7. Flexibility and efficiency of use
8. Aesthetic and minimalist design
9. Help users recognize, diagnose, and recover from errors
10. Help and documentation

Shnedierman and Plaisant's (2010) list of eight golden rules in *Designing the User Interface* (pp. 88-89)

1. Strive for consistency
2. Cater to universal usability
3. Offer informative feedback
4. Design dialogues to yield closure
5. Prevent errors
6. Permit easy reversal of actions
7. Support internal locus of control
8. **Reduce short-term memory load**

Two prominent sets of heuristics

Nielsen's (1994) list of ten heuristics in his book *Usability Engineering*:
http://www.useit.com/papers/heuristic/heuristic_list.html

1. Visibility of system status
2. Match between system and the real world
3. User control and freedom
4. Consistency and standards
5. Error prevention
6. Recognition rather than recall
7. ~~**Flexibility and efficiency of use**~~
8. Aesthetic and minimalist design
9. Help users recognize, diagnose, and recover from errors
10. Help and documentation

Shnedierman and Plaisant's (2010) list of eight golden rules in *Designing the User Interface* (pp. 88-89)

1. Strive for consistency
2. Cater to universal usability
3. Offer informative feedback
4. Design dialogues to yield closure
5. Prevent errors
6. Permit easy reversal of actions
7. Support internal locus of control
8. Reduce short-term memory load

Two prominent sets of heuristics

Nielsen's (1994) list of ten heuristics in his book *Usability Engineering*:
http://www.useit.com/papers/heuristic/heuristic_list.html

1. Visibility of system status
2. Match between system and the real world
3. User control and freedom
4. Consistency and standards
5. Error prevention
6. Recognition rather than recall
7. Flexibility and efficiency of use
8. ~~**Aesthetic and minimalist design**~~
9. Help users recognize, diagnose, and recover from errors
10. Help and documentation

Shnedierman and Plaisant's (2010) list of eight golden rules in *Designing the User Interface* (pp. 88-89)

1. Strive for consistency
2. Cater to universal usability
3. Offer informative feedback
4. Design dialogues to yield closure
5. Prevent errors
6. Permit easy reversal of actions
7. Support internal locus of control
8. Reduce short-term memory load

Two prominent sets of heuristics

Nielsen's (1994) list of ten heuristics in his book *Usability Engineering*:
http://www.useit.com/papers/heuristic/heuristic_list.html

1. Visibility of system status
2. Match between system and the real world
3. User control and freedom
4. Consistency and standards
5. Error prevention
6. Recognition rather than recall
7. Flexibility and efficiency of use
8. Aesthetic and minimalist design
9. ~~**Help users recognize, diagnose, and recover from errors**~~
10. Help and documentation

Shnedierman and Plaisant's (2010) list of eight golden rules in *Designing the User Interface* (pp. 88-89)

1. Strive for consistency
2. Cater to universal usability
3. Offer informative feedback
4. Design dialogues to yield closure
5. Prevent errors
6. Permit easy reversal of actions
7. Support internal locus of control
8. Reduce short-term memory load

Two prominent sets of heuristics

Nielsen's (1994) list of ten heuristics in his book *Usability Engineering*:
http://www.useit.com/papers/heuristic/heuristic_list.html

1. Visibility of system status
2. Match between system and the real world
3. User control and freedom
4. Consistency and standards
5. Error prevention
6. Recognition rather than recall
7. Flexibility and efficiency of use
8. Aesthetic and minimalist design
9. Help users recognize, diagnose, and recover from errors
10. ~~**Help and documentation**~~

Shnedierman and Plaisant's (2010) list of eight golden rules in *Designing the User Interface* (pp. 88-89)

1. Strive for consistency
2. Cater to universal usability
3. Offer informative feedback
4. Design dialogues to yield closure
5. Prevent errors
6. Permit easy reversal of actions
7. Support internal locus of control
8. Reduce short-term memory load

Comparing Shneiderman and Plaisant (2010) against Nielsen (1994)

Two prominent sets of heuristics

Nielsen's (1994) list of ten heuristics in his book *Usability Engineering*:
http://www.useit.com/papers/heuristic/heuristic_list.html

1. Visibility of system status
2. Match between system and the real world
3. User control and freedom
4. **Consistency and standards**
5. Error prevention
6. Recognition rather than recall
7. Flexibility and efficiency of use
8. Aesthetic and minimalist design
9. Help users recognize, diagnose, and recover from errors
10. Help and documentation

Shnedierman and Plaisant's (2010) list of eight golden rules in *Designing the User Interface* (pp. 88-89)

1. **Strive for consistency**
2. Cater to universal usability
3. Offer informative feedback
4. Design dialogues to yield closure
5. Prevent errors
6. Permit easy reversal of actions
7. Support internal locus of control
8. Reduce short-term memory load

Two prominent sets of heuristics

Nielsen's (1994) list of ten heuristics in his book *Usability Engineering*:
http://www.useit.com/papers/heuristic/heuristic_list.html

1. Visibility of system status
2. Match between system and the real world
3. User control and freedom
4. Consistency and standards
5. Error prevention
6. Recognition rather than recall
7. Flexibility and efficiency of use
8. Aesthetic and minimalist design
9. Help users recognize, diagnose, and recover from errors
10. Help and documentation

Shnedierman and Plaisant's (2010) list of eight golden rules in *Designing the User Interface* (pp. 88-89)

1. Strive for consistency
2. ~~**Cater to universal usability**~~
3. Offer informative feedback
4. Design dialogues to yield closure
5. Prevent errors
6. Permit easy reversal of actions
7. Support internal locus of control
8. Reduce short-term memory load

Two prominent sets of heuristics

Nielsen's (1994) list of ten heuristics in his book *Usability Engineering*:
http://www.useit.com/papers/heuristic/heuristic_list.html

1. **Visibility of system status**
2. Match between system and the real world
3. User control and freedom
4. Consistency and standards
5. Error prevention
6. Recognition rather than recall
7. Flexibility and efficiency of use
8. Aesthetic and minimalist design
9. Help users recognize, diagnose, and recover from errors
10. Help and documentation

Shnedierman and Plaisant's (2010) list of eight golden rules in *Designing the User Interface* (pp. 88-89)

1. Strive for consistency
2. Cater to universal usability
3. **Offer informative feedback**
4. Design dialogues to yield closure
5. Prevent errors
6. Permit easy reversal of actions
7. Support internal locus of control
8. Reduce short-term memory load

Two prominent sets of heuristics

Nielsen's (1994) list of ten heuristics in his book *Usability Engineering*:
http://www.useit.com/papers/heuristic/heuristic_list.html

1. **Visibility of system status**
2. Match between system and the real world
3. User control and freedom
4. Consistency and standards
5. Error prevention
6. Recognition rather than recall
7. Flexibility and efficiency of use
8. Aesthetic and minimalist design
9. Help users recognize, diagnose, and recover from errors
10. Help and documentation

Shnedierman and Plaisant's (2010) list of eight golden rules in *Designing the User Interface* (pp. 88-89)

1. Strive for consistency
2. Cater to universal usability
3. Offer informative feedback
4. **Design dialogues to yield closure**
5. Prevent errors
6. Permit easy reversal of actions
7. Support internal locus of control
8. Reduce short-term memory load

Two prominent sets of heuristics

Nielsen's (1994) list of ten heuristics in his book *Usability Engineering*:
http://www.useit.com/papers/heuristic/heuristic_list.html

1. Visibility of system status
2. Match between system and the real world
3. User control and freedom
4. Consistency and standards
5. **Error prevention**
6. Recognition rather than recall
7. Flexibility and efficiency of use
8. Aesthetic and minimalist design
9. Help users recognize, diagnose, and recover from errors
10. Help and documentation

Shnedierman and Plaisant's (2010) list of eight golden rules in *Designing the User Interface* (pp. 88-89)

1. Strive for consistency
2. Cater to universal usability
3. Offer informative feedback
4. Design dialogues to yield closure
5. **Prevent errors**
6. Permit easy reversal of actions
7. Support internal locus of control
8. Reduce short-term memory load

Two prominent sets of heuristics

Nielsen's (1994) list of ten heuristics in his book *Usability Engineering*:
http://www.useit.com/papers/heuristic/heuristic_list.html

1. Visibility of system status
2. Match between system and the real world
3. **User control and freedom**
4. Consistency and standards
5. Error prevention
6. Recognition rather than recall
7. Flexibility and efficiency of use
8. Aesthetic and minimalist design
9. Help users recognize, diagnose, and recover from errors
10. Help and documentation

Shnedierman and Plaisant's (2010) list of eight golden rules in *Designing the User Interface* (pp. 88-89)

1. Strive for consistency
2. Cater to universal usability
3. Offer informative feedback
4. Design dialogues to yield closure
5. Prevent errors
6. **Permit easy reversal of actions**
7. Support internal locus of control
8. Reduce short-term memory load

Two prominent sets of heuristics

Nielsen's (1994) list of ten heuristics in his book *Usability Engineering*:
http://www.useit.com/papers/heuristic/heuristic_list.html

1. Visibility of system status
2. Match between system and the real world
3. **User control and freedom**
4. Consistency and standards
5. Error prevention
6. Recognition rather than recall
7. Flexibility and efficiency of use
8. Aesthetic and minimalist design
9. Help users recognize, diagnose, and recover from errors
10. Help and documentation

Shnedierman and Plaisant's (2010) list of eight golden rules in *Designing the User Interface* (pp. 88-89)

1. Strive for consistency
2. Cater to universal usability
3. Offer informative feedback
4. Design dialogues to yield closure
5. Prevent errors
6. Permit easy reversal of actions
7. **Support internal locus of control**
8. Reduce short-term memory load

Two prominent sets of heuristics

Nielsen’s (1994) list of ten heuristics in his book *Usability Engineering*:
http://www.useit.com/papers/heuristic/heuristic_list.html

1. Visibility of system status
2. Match between system and the real world
3. User control and freedom
4. Consistency and standards
5. Error prevention
6. **Recognition rather than recall**
7. Flexibility and efficiency of use
8. Aesthetic and minimalist design
9. Help users recognize, diagnose, and recover from errors
10. Help and documentation

Shnedierman and Plaisant’s (2010) list of eight golden rules in *Designing the User Interface* (pp. 88-89)

1. Strive for consistency
2. Cater to universal usability
3. Offer informative feedback
4. Design dialogues to yield closure
5. Prevent errors
6. Permit easy reversal of actions
7. Support internal locus of control
8. **Reduce short-term memory load**

Two prominent sets of heuristics

Nielsen's (1994) list of ten heuristics in his book *Usability Engineering*:
http://www.useit.com/papers/heuristic/heuristic_list.html

1. Visibility of system status
2. Match between system and the real world
3. User control and freedom
4. Consistency and standards
5. Error prevention
6. **Recognition rather than recall**
7. Flexibility and efficiency of use
8. Aesthetic and minimalist design
9. Help users recognize, diagnose, and recover from errors
10. Help and documentation

Shnedierman and Plaisant's (2010) list of eight golden rules in *Designing the User Interface* (pp. 88-89)

1. Strive for consistency
2. Cater to universal usability
3. Offer informative feedback
4. Design dialogues to yield closure
5. Prevent errors
6. Permit easy reversal of actions
7. Support internal locus of control
8. **Reduce short-term memory load**

Bold entries are unique for its specific set of heuristics

Nielsen's (1994) list of ten heuristics in his book *Usability Engineering*:
http://www.useit.com/papers/heuristic/heuristic_list.html

1. Visibility of system status
2. **Match between system and the real world**
3. User control and freedom
4. Consistency and standards
5. Error prevention
6. Recognition rather than recall
7. Flexibility and efficiency of use
8. **Aesthetic and minimalist design**
9. **Help users recognize, diagnose, and recover from errors**
10. **Help and documentation**

Shnedierman and Plaisant's (2010) list of eight golden rules in *Designing the User Interface* (pp. 88-89)

1. Strive for consistency
2. **Cater to universal usability**
3. Offer informative feedback
4. Design dialogues to yield closure
5. Prevent errors
6. Permit easy reversal of actions
7. Support internal locus of control
8. Reduce short-term memory load

Other, vendor-specific, guidelines

- Mac OS X Human Interface Guidelines
  - http://developer.apple.com/library/mac/#documentation/UserExperience/Conceptual/AppleHIGuidelines/Intro/Intro.html
- Microsoft Official User Interface Guidelines
  - There is a lot of these, an instructive example can be found here:
  - http://msdn.microsoft.com/en-us/library/Aa511327
- As you will notice, vendor-specific guidelines are more prescriptive and obviously vendor-centred

Usability Inspection

Usability inspection

- Expert evaluation methods for usability analysis
- Cheap and cheerful methods
- Should be contrasted against **user studies**:
  - In a user study **users** are asked to perform tasks in an interface by virtue of representing end-users
  - In a usability inspection **experts** are asked to inspect an interface based on their expertise

- Two primary methods of carrying out usability inspection:
  1. Heuristic evaluation
  2. Cognitive walk-through (not discussed here)

Heuristic evaluation

- Informal method of usability analysis
- Created by Jakob Nielsen in the late 1980s
- Key paper:
  - Nielsen, J. and Molich, R. 1990. Heuristic evaluation of user interfaces. In *Proc. ACM Conference on Human Factors in Computing Systems (CHI 1990)*. ACM Press: 249-256.
  - http://dx.doi.org/10.1145/97243.97281

Heuristic evaluation, continued

- Experts are asked to evaluate a user interface based on a set of heuristics (such as the set of heuristics we discussed earlier in this lecture)
- It is a holistic evaluation procedure (it does not focus on specific tasks, rather it focuses on the entire system)
- Research has shown that a single expert is poor in finding all issues (Nielsen and Molich 1990)
- However, aggregated evaluations have better coverage
- A **heuristic evaluation is easy to carry out on your own and is a way to demonstrate a minimal level of usability in your wearable device**

KLM-GOMS

Analysing a GUI: GOMS

- Classic modelling method in human-computer interaction
- GOMS is a predictive method that can be used to evaluate a user interface design before any users have actually used the system

GOMS

- **Goals**
  - Aims of the user
- **Operators**
  - Actions that can be done in the interface, such as clicking, dragging and typing
- **Methods**
  - Sequences of sub-goals and operators that can be used to achieve a particular goal
- **Selection rules**
  - The rules by which a user chooses a particular method (from a set of methods) to achieve a goal
- A GOMS task analysis breaks down tasks into a hierarchy of goals (unit tasks) and sub-tasks using the four GOMS elements: goals, operators, methods and selection rules

Example from Cairns and Cox (2008)

- Method for goal: **delete object**
  1. Accomplish goal: move object to trash
  2. Return with goal accomplished
- Method for goal: **move object [destination]**
  1. Accomplish goal: drag object [destination]
  2. Return with goal accomplished
- Method for goal: **move object [destination]**
  1. Accomplish goal: send object [destination]
  2. Return with goal accomplished
- Method for goal: **drag object [destination]**
  1. Locate icon for object on screen
  2. Move cursor to object icon location
  3. Hold left mouse button down
  4. Locate destination icon on screen
  5. Move cursor to destination icon
  6. Verify that destination icon indicates activation
  7. Release mouse button
  8. Return with goal accomplished
- Method for goal: **send object [trash]**
  1. Locate icon for object on screen
  2. Move cursor to object icon location
  3. Hold right mouse button down
  4. Locate 'Move to Trash' item on pop-up menu
  5. Move mouse cursor to 'Move to Trash' item on pop-up menu
  6. Release mouse button
  7. Return with goal accomplished

Example from Cairns and Cox (2008), continued

- The previous example specified two ways to place objects in the trash: by dragging them to the trash can using the mouse, or by sending the object there using a pop-up menu
- Depending on circumstances one method may be preferred over the other, this can be formalised, for example using the decision rules below:

IF	the goal is to delete an object
AND	the trash can is visible
THEN	use the *drag object* method

IF	the goal is to delete an object
AND	the trash can is not visible
THEN	use the *send object* method

Model implementations of GOMS

- Several different model implementations of GOMS exist
- CMN-GOMS
  - Card Moran Newell (CMN)
  - Original version of GOMS
- KLM-GOMS
  - Keystroke-Level Model (KLM)
  - Simplification of CMN-GOMS
  - (We will talk more about this particular GOMS variant shortly)
- CPM-GOMS
  - Cognitive Perceptual Motor (CPM)
  - Does not assume a serial information-processing process (which KLM-GOMS does)
  - Models multi-tasking

KLM-GOMS: Keystroke-level model

- A subset of GOMS that only includes operators and methods
- KLM predicts task completion times
- All operators have a specific execution time
- Task completion times are calculated by summing up the execution times for the different operators that need to be used to perform the task
- **KLM-GOMS** is the name of this particular GOMS model

KLM-GOMS standard operators (standard time estimates in parentheses)

K	Press a key on the keyboard (0.28 s)
T(n)	Type a sequence of *n* characters on the keyboard ($n \times$ K s)
P	Point the mouse to a target on the display (1.1 s)
B	Press or release the mouse button (0.1 s)
BB	Click mouse button (0.2 s)
H	Move hands between mouse and keyboard (or vice versa) (0.4 s)
M	Mental act of routine thinking or perception (1.2 s)
W(t)	Wait time for system response (*t* s)

Example from Cairns and Cox (2008)

- Deleting a file using the previously defined methods
- Operator sequence: **drag object [trash]**
  - Point to file icon (**P**)
  - Press and hold mouse button (**B**)
  - Drag file icon to ‘Trash can’ icon (**P**)
  - Release mouse button (**B**)
  - Point to original window (**P**)
- Operator sequence: send **object [trash]**
  - Point to file icon (**P**)
  - Press and hold mouse button (**B**)
  - Point to ‘Move to Trash’ item on pop-up menu (**P**)
  - Release mouse button (**B**)
  - Point to original window (**P**)
- Both involves 3**P** + 2**B** = 3 × 1.1 + 2 × 0.1 = 3.5 s

Limitations of KLM-GOMS

- Assumes error-free expert behaviour
  - Few interfaces are only used by users who interact using error-free expert behaviour
  - Ignores learning curve effects
- Assumes reliable fixed time estimates are available for all operators
  - We know, from for example Fitts' law, that not all tasks, such as pointing tasks, take an equal amount of time

Checklist when carrying out a KLM-GOMS analysis

1. Define the higher level activity (for example, “delete file”) precisely, so that it can be defined as a sequence of clearly defined tasks (for example, “select icon, drag icon to trashcan, release mouse button)
2. If possible, create “subroutines” for repeat tasks. This is possible because time predictions are constant.
   – For example, the “delete file via trashcan” operation we previously defined can be defined as a subroutine DeleteFileViaTrashcan that takes 3.5 seconds.
3. Carefully review your analysis, do you include all low-level steps (carried out via KLM-GOMS operators) that are required for the task?
   – It is easy to forget steps in a KLM-GOMS analysis

Interaction Techniques and Devices

Direct manipulation

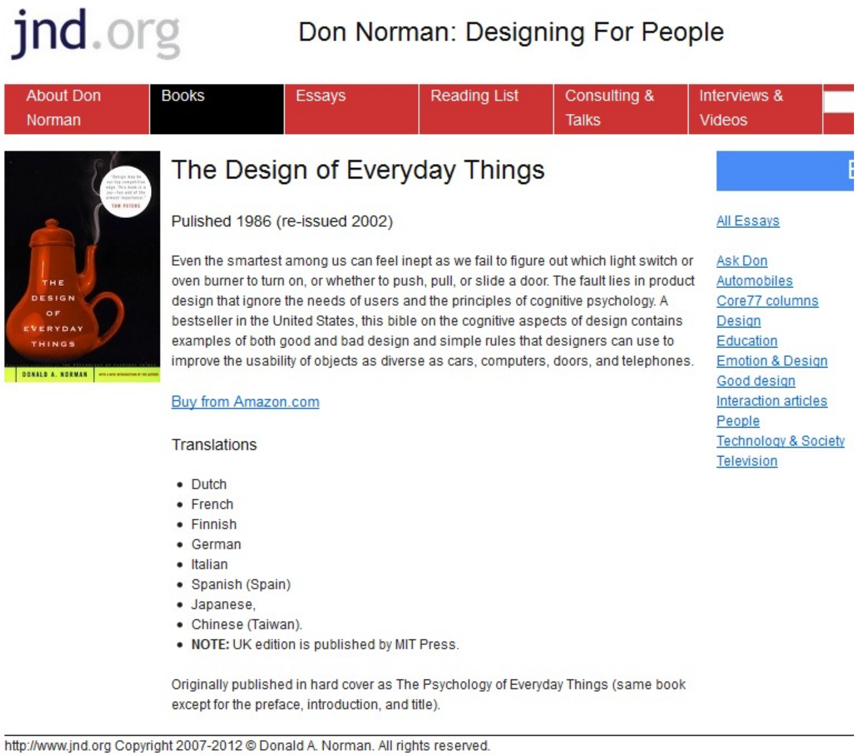
- Original definition:
  - Visibility of the objects and actions of interest
  - Rapid, reversible, incremental actions
  - Replacement of typed commands by a pointing action on the object of interest
  - (Well-known example: dragging a file to the trashcan)
- Three principles of direct manipulation according to Shneiderman and Plaisant (2010):
  - Continuous representations of the objects and actions of interest with meaningful visual metaphors
  - Physical actions or presses of labelled buttons, instead of complex syntax
  - Rapid, reversible, incremental actions whose effects are on the objects of interest are visible immediately

Advantages of direct manipulation according to Shneiderman and Plaisant (2010)

- Novices can learn basic functionality quickly, usually through a demonstration by a more experienced user
- Experts can work rapidly to carry out a wide range of tasks, even defining new functions and features
- Knowledgeable intermittent users can retain operational concepts
- Error messages are rarely needed
- Users can immediately see whether their actions are furthering their goals, and, if the actions are counterproductive, they can simply change the direction of their activity
- Users experience less anxiety because the interface is comprehensible and because actions can be reversed easily
- Users gain a sense of confidence and mastery because they are the initiators of action, they feel in control, and they can predict the interface’s responses

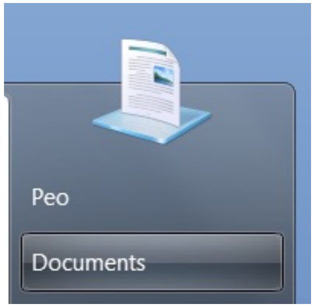
Perceived affordance

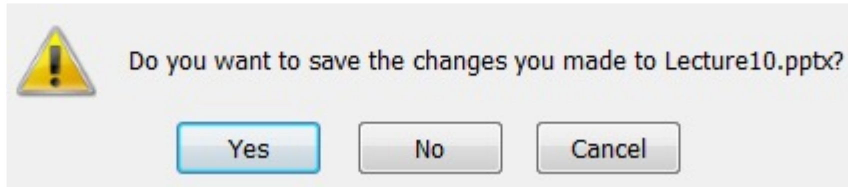
- "Perceived affordance" is a concept introduced by Don Norman in his world-famous book "The Design of Everyday Things"
- "In similar vein, because I can click anytime I want, it is wrong to argue whether a graphical object on the screen "affords clicking." It does. The real question is about the perceived affordance: Does the user perceive that clicking on that location is a meaningful, useful action to perform?"
  - http://www.jnd.org/dn.mss/affordances_and_design.html

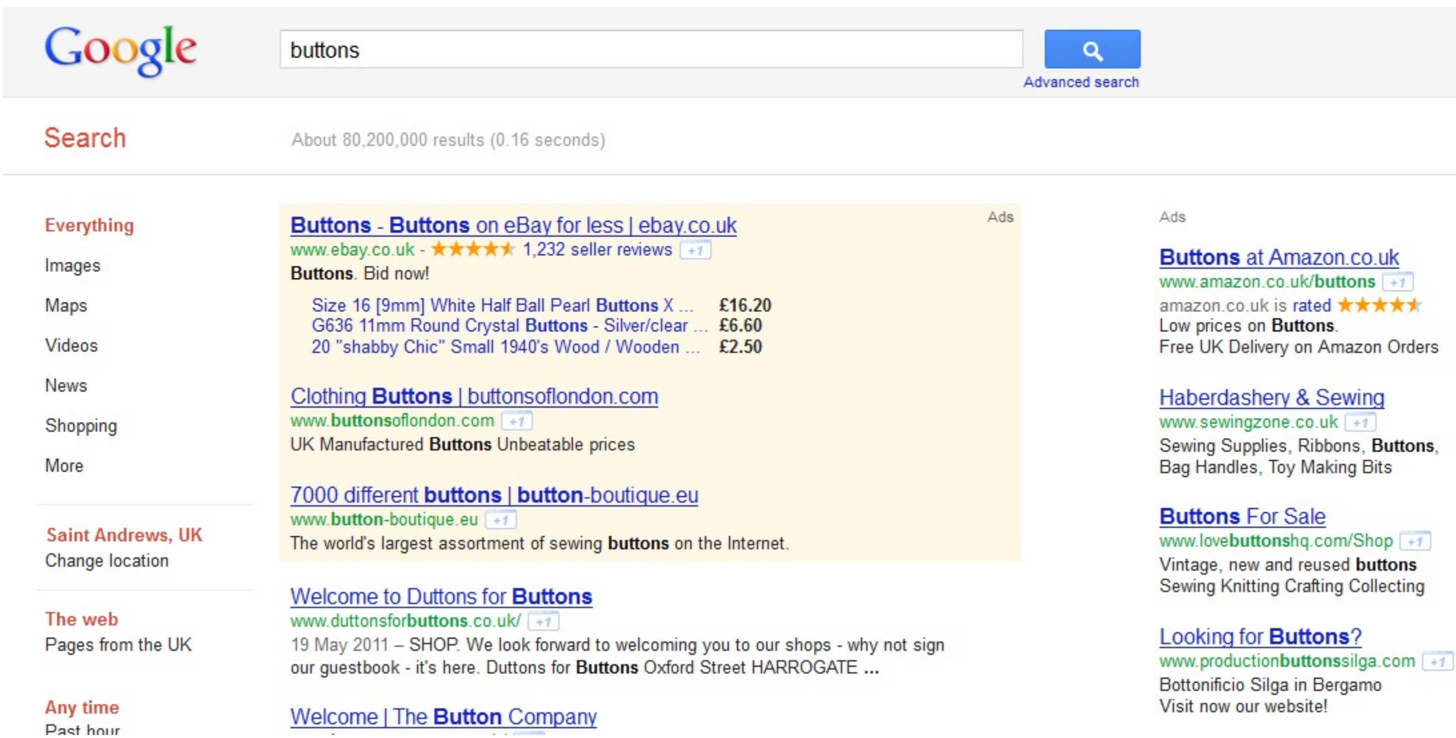


Screenshot from http://www.jnd.org/books/the-design-of-everyday-things.html

Perceived affordance, examples

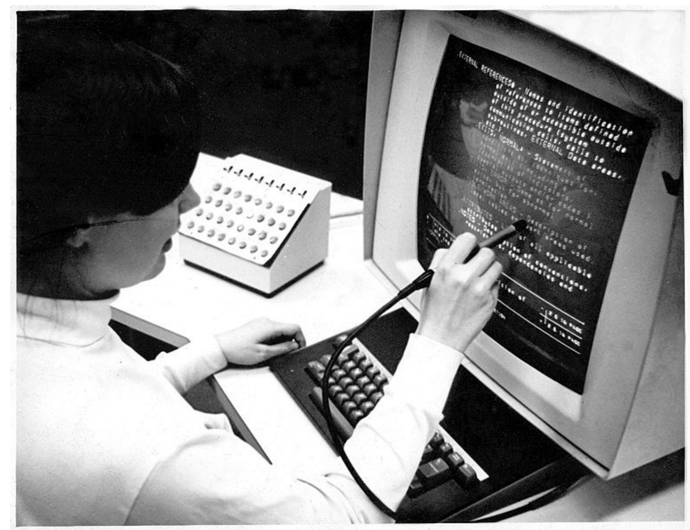






Pointing

- Mouse
- Pen
- Finger
- Laser pointer
- 3D full-body motion sensor, such as the Kinect
- Mobile device



http://commons.wikimedia.org/wiki/File:HypertextEditingSystemConsoleBrownUniv1969.jpg

Using pointing for interaction tasks

- Select
  - Choosing from a set of items
- Position
  - Choosing a point
- Orient
  - Choosing a direction
- Path
  - Rapid series of positioning and orientation operations
- Quantify
  - Specification of a numeric value
- Gesture
  - Indicating an action by performing a gestures, such as left-to-right swipe
- Text
  - Entering, moving, and editing text in a two-dimensional space

Pointing—direct vs. indirect control

- Examples of **direct control**: lightpens and touchscreens
  - A user interface problem is the pen or finger obscuring the screen
  - Early touchscreens had problems with sensor errors
  - One solution was using the **lift-off technique** rather than the land-on strategy
    - Target selection is determined as the point when the finger leaves the surface, rather than as the point when the finger first touches the surface
- Examples of **indirect control**: mouse, trackball, joystick, trackpoint, touchpad
  - Require more cognitive resources and hand/eye coordination than direct control
  - The mouse has been the by far most common pointing technique
    - Uses forearm for long movements and small finger movements for precise movements
    - A problem is **clutching**—having to lift up and reposition the mouse
    - The **C:D ratio** is the ratio between the **control** and **display**
    - A high C:D ratio means the user has to move the control far to move the pointer a certain distance in the display, and vice versa

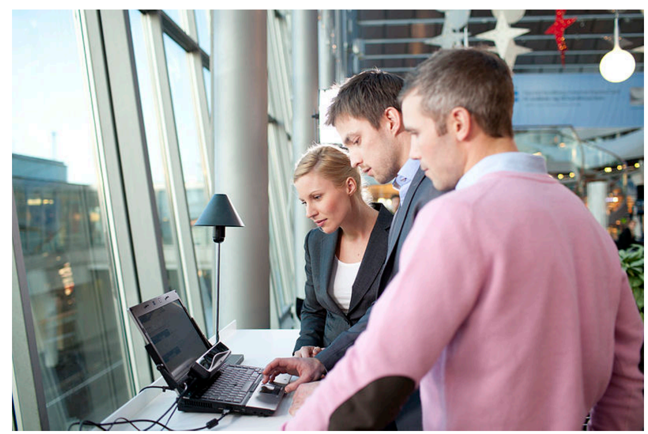
Pointing devices

- Direct control devices
  - Lightpen
  - Touchscreen
  - Stylus
- Indirect control devices
  - Mouse
  - Trackball
  - Joystick
  - Trackpoint
  - Touchpad
  - Graphics tablet
- Non-standard devices and strategies
  - Multitouch tablets and displays
  - Bimanual (two-handed) input
  - Eye-trackers
  - Sensors
  - 3D trackers
  - Haptic feedback
  - Foot controls
  - Tangible user interfaces
  - Digital paper

Beyond pointing...

- The vision of ubiquitous computing—computer interfaces become invisible and pervasive
- Fluent interaction with wall-sized displays
- Easy-to-use eye-tracking technologies
- Using your own body as an input and output device
- Interaction with multiple displays
- Fusing multiple sources of sensor information (accelerometers, computer vision, speech, gestures, etc.)

http://commons.wikimedia.org/wiki/File:Wikitude.jpg



http://upload.wikimedia.org/wikipedia/commons/1/15/Tobii_X1_Light_Eye_Tracker_Laptop.jpg

Typical research cycle when researching interaction techniques and tools

- Identify a problem
  - For example, navigatinging large menu structures on mobile devices is impractical
- Identify relevant **design principles**
  - These can be found by reading about how other researchers have previously discovered
  - They can also be discovered in a process of iterative development and testing
  - For example, one design principle can be to “Support mobile interaction in bursts”, meaning that when a user is walking around it has been demonstrated that the user attends to the mobile display in 5-seconds burst. This design principle can be implemented in a design by for example never assuming that users are constantly looking at the mobile display (avoid animations for example).
- Propose a solution based on a set of design principles, other (perhaps context-dependent) observations, and series of designs, sketches, pilot studies, etc.
- Evaluate said solution, ideally in actual use contexts while keeping internal validity high
- Extract **design implications** from the results of the evaluations, what design implications can other researchers use as design principles in future designs?
- Use these design implications as part of new design principles for new ideas on how to best enable users to interact with computers
- And the cycle repeats…

Some issues when creating new interaction techniques

- Learning curve
  - How long time does it take for users to reach the **crossover point**—the point in time when a new interaction technique performs better than what users are already familiar with?
- Walk-up use
  - Can the interaction technique be used right away with minimal setup?
- Robustness and error handling
  - Is the interaction technique robust with respect to noise? How are errors handled? Can the user easily recover from errors?
- Intrusiveness
  - Does the interaction technique require the user to wear markers or attach cumbersome apparatus?
- Social acceptability
  - Can a user use a particular interaction technique in society?
- Context of use
  - How does the use context affect users' interaction with the technique?

Evaluating interaction techniques

- Typically interaction techniques are evaluated in controlled experiments
- The techniques themselves are levels of an independent variable
- The primary dependent variables are typically time and error
- Good choices of baselines and experimental tasks are crucial
- There is a trade-off between internal and external validity
  - Reliable experimental outcomes versus being able to generalise the results to actual use contexts in the real-world
- Good evaluations are tricky to execute
  - The key is to ask the right questions—does the user study answer whether the proposed interaction technique actually is more useful than techniques we are already aware of?

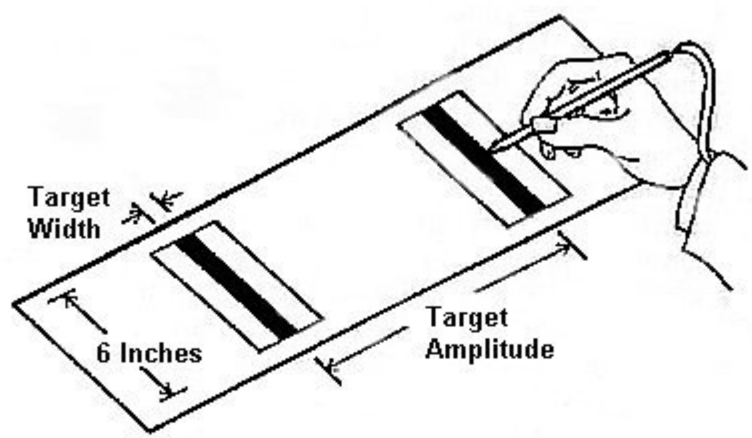
Fitts' Law
(Human Performance Laws)

Fitts’ law

- Fitts’ law is a mathematical model that predicts that the average movement time required to hit a target along a one-dimensional path is proportional to the distance of the target and the width of the target

Fitts’ reciprocal task paradigm

- Fitts’ law is **one dimensional**
- What is meant by the **width** of a target is indicated in the figure below
- Sometimes the **distance** is referred to as **amplitude**



Fitts, P.M. 1954. The information capacity of the human motor system in controlling the amplitude of movement. *Journal of Experimental Psychology* **47**(6): 381–391.

Fitts' law

- Probably the most well-known and well-established model of human performance
- Models **average movement time** as a function of **index of difficulty**:

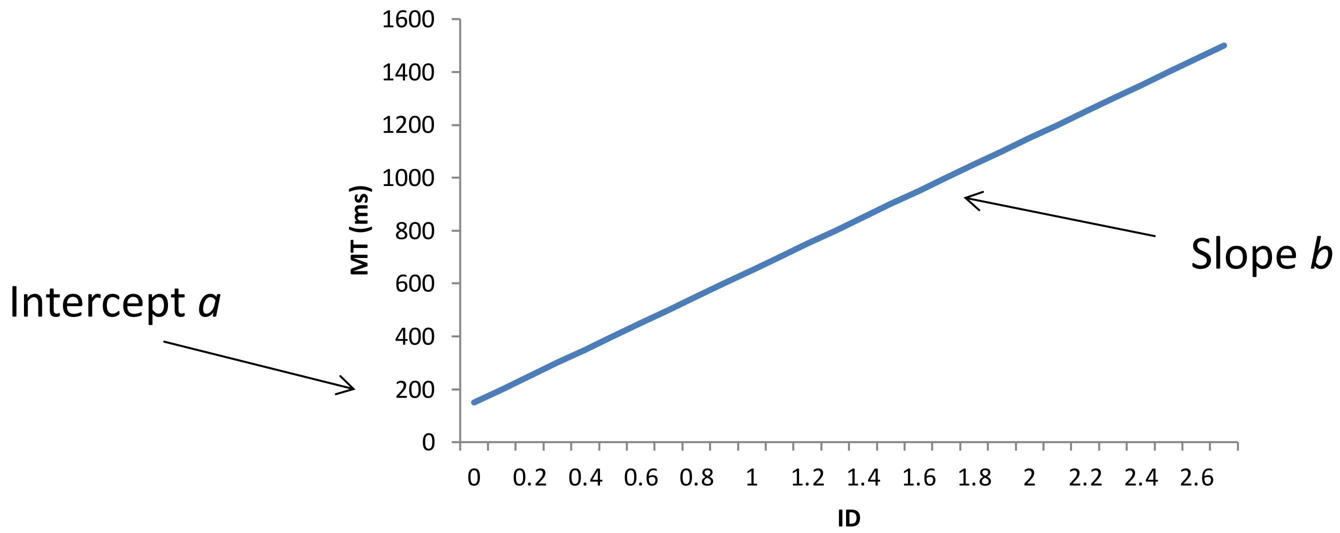
$$MT = a + bID$$

where ***a*** and ***b*** are **regression coefficients** and index of difficulty is:

$$ID = \log_2\left(\frac{D+W}{W}\right)$$

where ***D*** is the distance to the target and ***W*** is the width of the target

Fitts' Law is a linear regression model



$$MT = a + bID = a + b\log_2\left(\frac{D+W}{W}\right) = a + b\log_2\left(1 + \frac{D}{W}\right)$$

Index of difficulty

- Index of difficulty is a measure in bits of how difficult it is to hit a target of a certain diameter (*W*) from a certain distance (*D*)

$$ID = \log_2\left(\frac{D+W}{W}\right)$$

Throughput

- Fitts' law enables us to measure users' **information capacity** via a measure called **throughput**
- Throughput (*TP*) is the number of **bits** of information a user can communicate per second (bits/s) **independently** of a specific target
- One definition of throughput is:

$$TP = \frac{ID_{avg}}{MT_{avg}}$$

- The downside is that this definition is dependent on an arbitrary average ID. An alternative definition of throughput is:

$$TP = \frac{1}{b}$$

- The downside with the latter formulation is that it ignores the effect of *a*.

Fitts' Law implies a speed-accuracy trade-off

- *D*/*W* relationship in Fitts' law
- Two ways to minimise average movement time *MT:*
  - Minimise *D*
  - Maximise *W*

$$MT = a + bID$$

$$ID = \log_2\left(\frac{D+W}{W}\right) = \log_2\left(1+\frac{D}{W}\right)$$

$$MT \propto \frac{D}{W}$$

Fitts' law: assumptions

- Fitts' law assumes closed-loop interaction
  - In other words, it models a visually guided action
- Fitts' law only models a particular biologically plausible interval
  - The distance between two buttons cannot be three meters...
- The regression coefficients *a* and *b* vary depending on the task
- Different pointing devices can be compared by comparing their throughput

Examples of other models of performance

- Steering law
  - Models the time it takes to steer through a tunnel
  - http://dx.doi.org/10.1145/258549.258760
- Law of crossing
  - Models the time it takes to cross a succession of targets
  - http://dx.doi.org/10.1145/503376.503390
- Single-stroke gesture models
  - Models the time it takes to articulate a single-stroke gesture on a touch-screen or resistive surface
  - http://dx.doi.org/10.1145/1240624.1240850

Limitations of performance models

- Currently only low-level actions are modelled reliably
  - This limitation is evident even when trying to model single-stroke gestures; a lot of assumptions are made on the type of gestures that can be articulated in order to make predictions
- As a consequence of this limitation, the usefulness of such models when engineering HCI interfaces and systems is limited

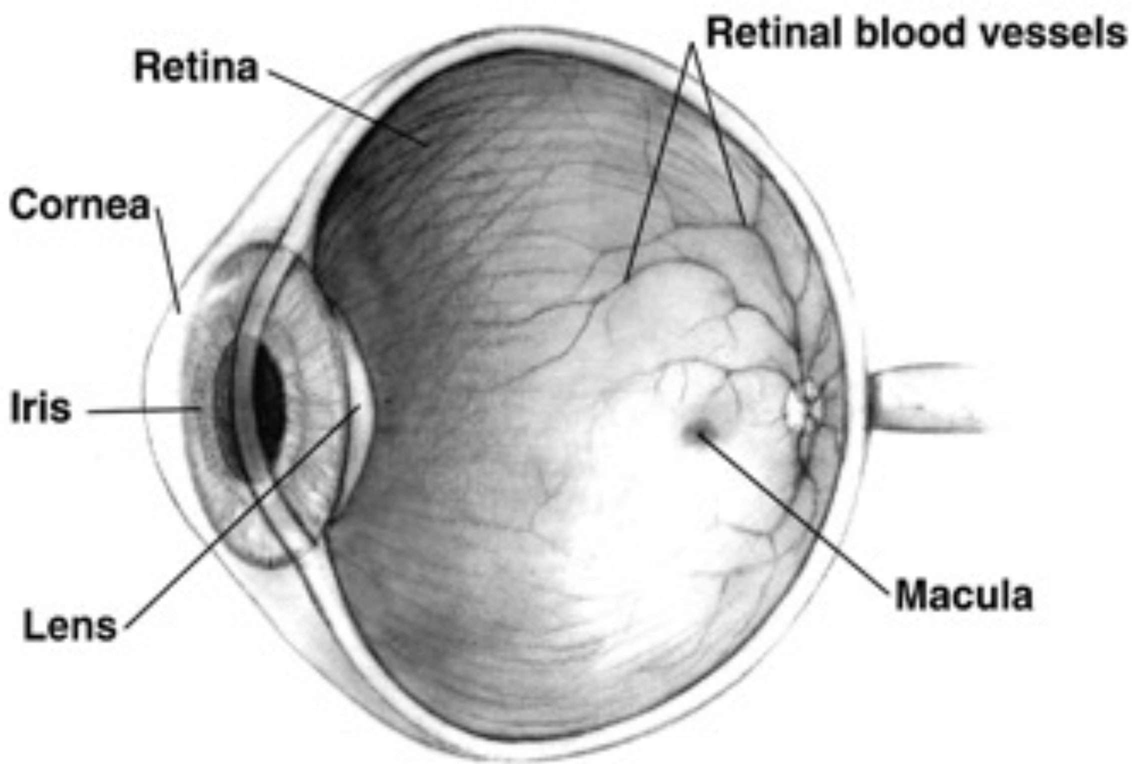
Designing Intelligent Interactive Systems

Design issues for intelligent interactive systems

- **Error correction**
  - How do we ensure users can correct (inevitable) recognition errors as quickly and as accurately as possible?
- **Supporting error detection**
  - How do we help users detecting errors made by the system?
- **Transparency**
  - How do we explain to users why the system interpreted their input in a particular way?
- **Agency**
  - How do we ensure users feel in control when the computer system infers their intentions via sensors?

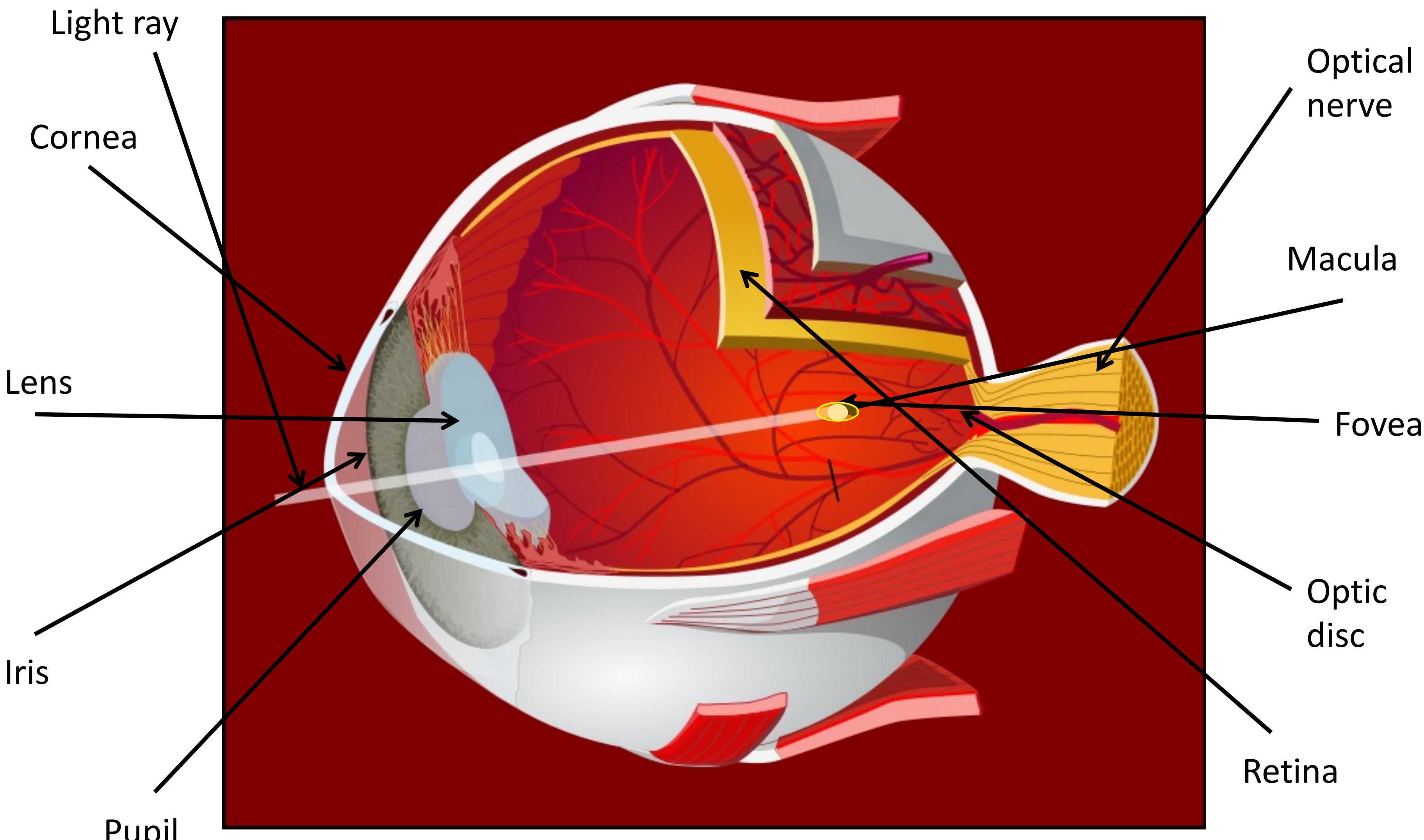
Eye-Tracking Design

The anatomy of the human eye



http://commons.wikimedia.org/wiki/File:Human_eye_cross-sectional_view_grayscale.png

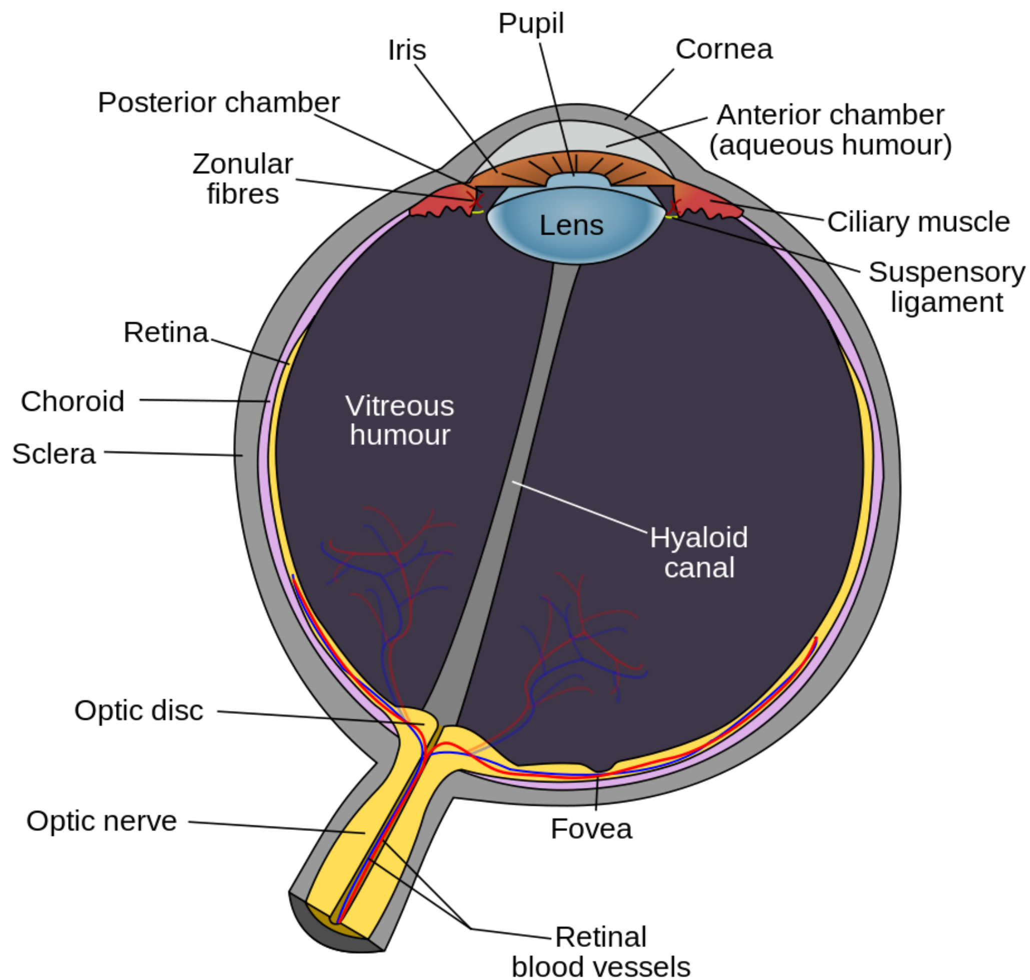
The anatomy of the human eye



http://commons.wikimedia.org/wiki/File:Eye-diagram_bg.svg

Retina, fovea and macula

- The retina converts light into nerve signals
- The macula is an oval yellow spot near the centre of the retina
- The macula is responsible for central vision
- A substantial part of the brain's visual capacity is dedicated to processing visual input from the macula
- The fovea is located near the centre of the macula
- The fovea has a higher concentration of photoreceptors with high acuity (clearness)

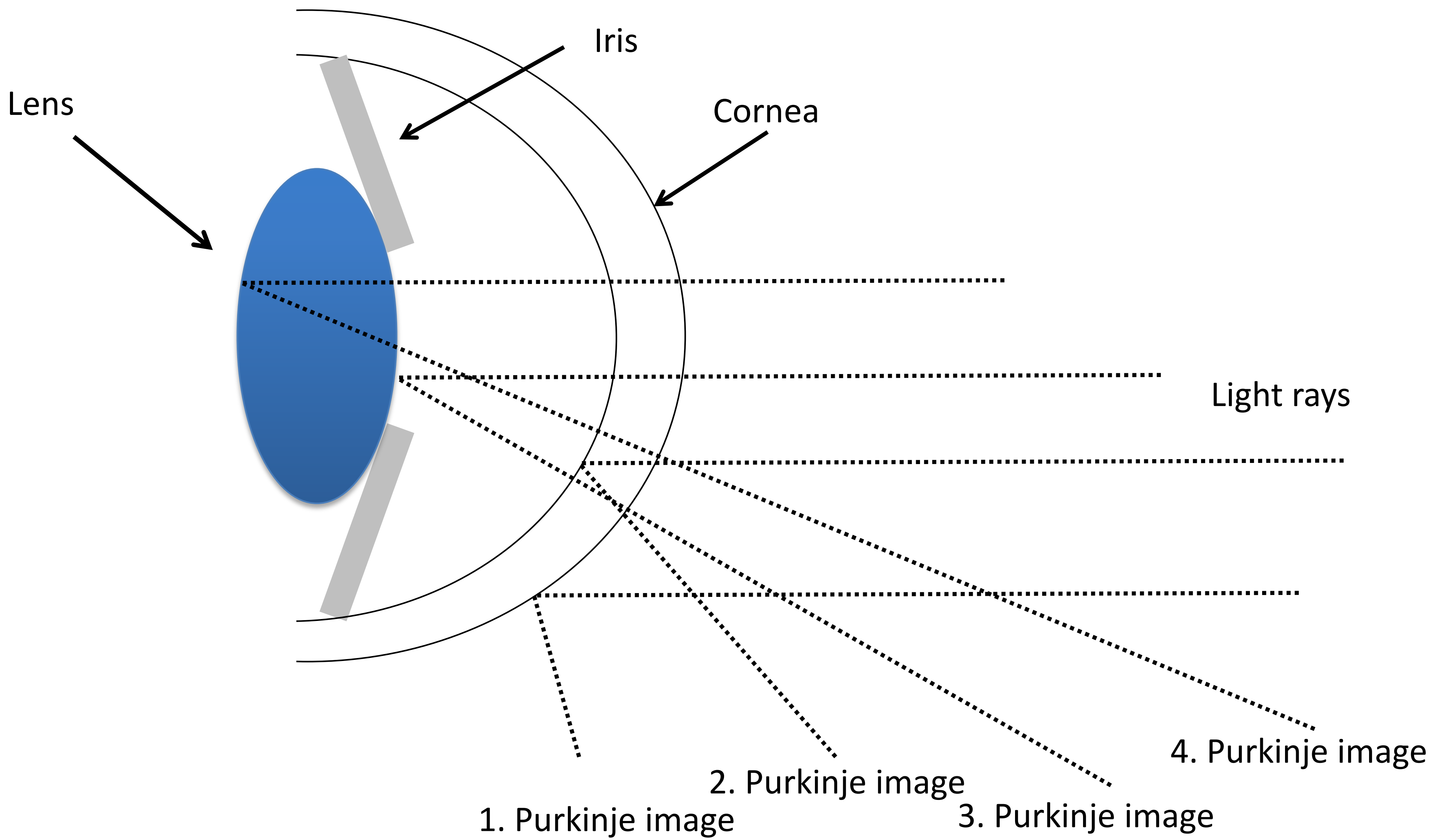


http://commons.wikimedia.org/wiki/File:Schematic_diagram_of_the_human_eye_en.svg

Eye-tracking

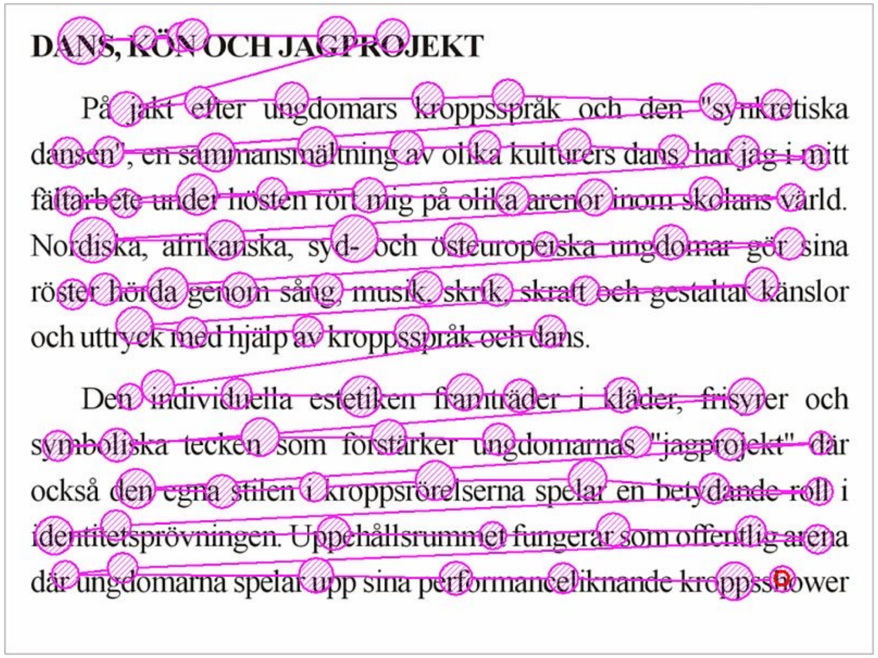
- Eye-trackers infer where the user is looking at the screen by tracking rotations of the eye, typically using one out of four methods:
  - Projecting infrared light towards the participants' eyes. The anatomical parts of the eye that reflects the light efficiently generate so-called Purkinje images. These images can then be used to infer the user's gaze. High-accuracy commercial eye-tracking systems typically implement some variant of the above or a related scheme. As with all other eye-tracking techniques, this method requires calibration.
  - Optically detecting users' gaze via a regular camera. Gaze position is inferred via computer vision methods. These methods are very sensitive to users' head movement and as a consequence are not very practical today. Requires recalibration whenever the user is significantly moving their head.
  - Electrooculography (EOG). Electrodes are placed above/below or left/right of the eye. The electrodes measure the resting potential of the retina. If the eye is moved away from the centre position towards an electrode then the electrode will sense the change. This information can then be used to infer relative eye position changes. A problem with EOG is drift. A downside is the required set-up before use.
  - Contact lenses with some form of sensor, such as a magnetic field sensor. As the eye rotates the contact lens rotates with it. Enables highly accurate eye movement measurements. The downside is that the technique is invasive and requires set-up.

Eye-tracking using Purkinje images



Saccades and fixations

- A **saccade** is a ballistic jump
- Saccades are pre-planned and take 150–250 ms to plan and execute
- A saccade ends with a **fixation**
- A fixation the period of time after a saccade when the eye is relatively stationary (200–300 ms)



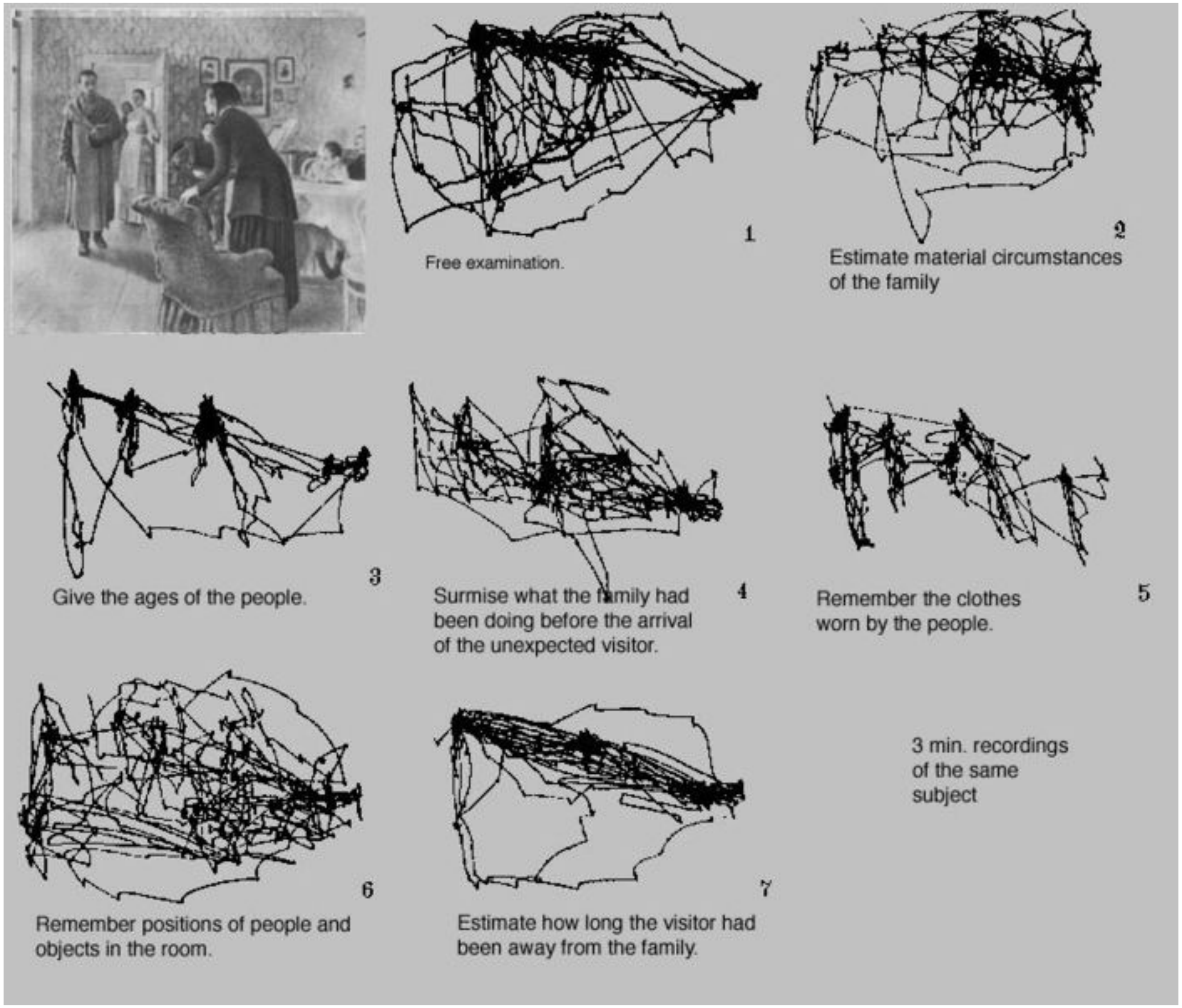
http://commons.wikimedia.org/wiki/File:Reading_Fixations_Saccades.jpg

Scanpaths

- A sequence of saccade-fixations forms a **scanpath**
- Such scanpaths can be recorded using eye-tacking technology that records users' gaze positions at a certain rate (typically 30-60 Hz)
- These scanpaths are then aggregated into fixations if they meet certain time and location thresholds
- The assumption is that if a user has fixated on an area it will influence the user's further actions or the details that the user is registering in memory
  - However, for this to be likely to be true, the user needs to be given a suitable task

Eye-mind hypothesis

- The **eye-mind hypothesis:**
  - *What people look at is what they are thinking about*
- In other words, visual attention is a proxy for mental attention
- Yarbus, A. L. (1967). *Eye Movements and Vision*. New York: Plenum.



http://commons.wikimedia.org/wiki/File:Yarbus_The_Visitor.jpg

Qualitative analysis using eye-trackers

- Manually inspecting scanpaths and replaying gaze movies to determine:
  - Areas users tend to look at
  - Interesting patterns of behaviour that can be studied in more detail in later sessions
- Qualitative eye-tracking analysis is common in commercial settings
  - For example, when evaluating the effectiveness of advertisements or website layouts

Quantitative analysis using eye-trackers

- Eye-tracking can produce vast quantities of objective data which can be treated as a dependent variable in a controlled experiment
- One method is self-reporting: correlating objective data with a ‘think-aloud’ protocol instructing participants to tell the experimenter what they are currently trying to achieve and how they are trying to achieve it
- Various computational methods can also be used to aggregate data and infer where participants focussed their visual attention

Areas of research suitable for eye-tracking

- **Search**
  - Where do people search for the correct navigation link?
- **Comprehension**
  - What do people look at and for how long in order to make sense of the stimulus?
- **Making decisions and solving problems**
  - Do different groups of users show different behaviour when solving a problem?
- **Reading and scanning**
  - What parts of a webpage were read versus scanned?
- **Efficiency of task completion**
  - Do experts have a visual strategy that can be taught to novices?
- **Visual elements capturing attention**
  - Do people notice banner ads when visiting websites?

Examples of applications of eye-tracking interfaces

- Automatically dismissing pop-ups
  - A user can look at a pop-up and the system can automatically dismiss it when it detects that the user is looking at it
- Gaze-contingent displays
  - **Gaze-contingent displays** change or adapt the content on the display depending on what the user is looking at. A gaze-contingent display is typically used in psychology experiments to adapt or change the stimuli depending on the participant's gaze. One application of a gaze-contingent display could for example be to provide more visual structure and information in an information visualisation interface where the user is actually looking.
- Attentive user interfaces (AUIs)
  - Eye-trackers can be used to identify if users are focussing visual attention on a particular display, or a part of a particular display. This information can be used to adapt the interface. Such interfaces are called **attentive user interfaces (AUIs)**.

Leveraging eye movement in an attentive user interface

- Some guidelines for using eye input for control has been provided by Zhai (2003):
  - The hands and eyes tend to work in combination
  - Eye gaze provides context of interaction
  - Use of eye-gaze for control should therefore be implicit (indirect)
  - The hands of the user tend to act within the context of where the user is focussing visual attention
  - Therefore, the user of the user's hands for control input should be explicit (direct)
- These guidelines follow from the insight that the eyes are not always under voluntary control
- Therefore, loading the visual communication channel with motor-control tasks is conflicting with the eyes' natural behaviour in searching and acquiring visual information about the world

Zhai, S. 2003. What's in the eyes for attentive input. *Communications of the ACM* **46**(3): 34-39.

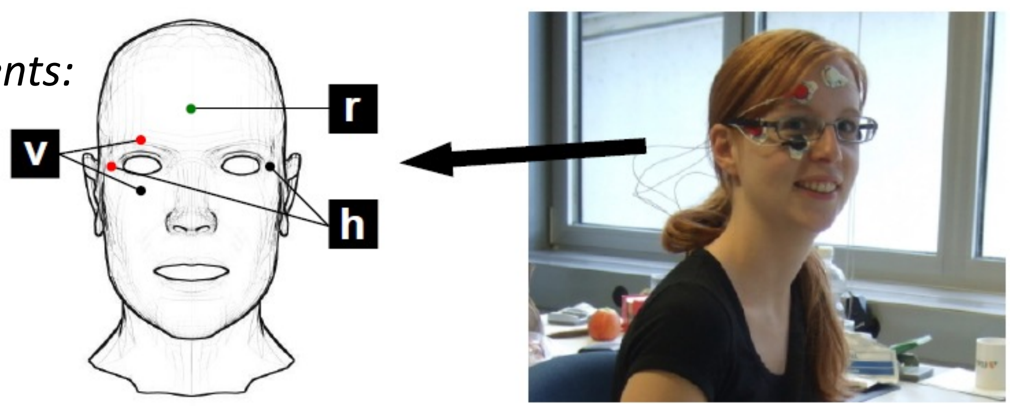
Issues when designing eye-tracking interfaces

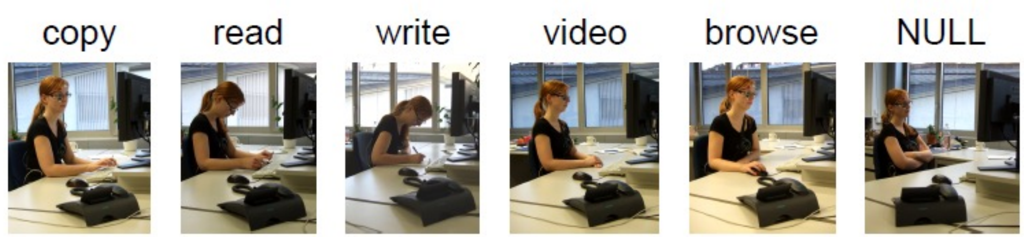
- **The “Midas touch” problem** for gaze pointing
  - The issue of how to reliably separate looking around from activating interface elements
  - Can be worked around using a number of techniques. A common method is to use a **dwell-timeout**: users have to gaze at an interface element for a fixed amount of time to trigger an action.
- Cursor tracking: balancing smoothness and lag
  - The exact gaze position is unstable and the raw eye-tracking signal tends to oscillate between several estimations
  - To avoid a jittery cursor the signal is **smoothed** (by for example averaging the last six gaze positions)
  - However, smoothing results in **lag**: slower cursor movement that necessary due to the fact that new cursor updates get averaged with old ones
  - A good work-around to balance smoothness and lag is to implement an **adaptive low-pass filter**

Using eye-tracking for activity recognition

- **Activity recognition** is the field that attempts to infer users activities via sensor data
- Eye-based activity recognition (EAR) can be used to detect activities that would be difficult or impossible to detect using alternative modalities
- For example, mobile eye movement can be recorded using an electrooculography (EOG) system
- EOG has been used to classify six classes of activity in an office environment: copying a text, reading a printed paper, taking hand-written notes, watching a video, browsing the web, and no specific activity

Electrode placements:
h: horizontal
v: vertical
v: reference





Bulling, A., Ward, J.A., Gellersen, H. and Tröster, G. 2011. Eye movement analysis for activity recognition using electrooculography. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **33**(4): 741-753.