CSCU9N5

Multimedia

&Human Computer Interaction (HCI)

CSCU9N5: Multimedia and HCI

Lecturers:

- Prof. Bruce Graham : 4B120, bruce.graham@stir.ac.uk
- Prof. Carron Shankland: 4B122, carron.shankland@stir.ac.uk (Coordinator)

Schedule:

- 2/3 lectures + 1 practical per week
- 4 tutorials over semester
- See Canvas module page for details and signups

Assessment:

- Assignment (50%): multimedia design and development
- Exam (50%): closed-book

Overview of CSCU9N5

Aim:

- Give you a guided tour of Multimedia & HCI
- Provide a foundation for understanding interface design
  - UX: User Experience
  - Mainly GUIs, but also Audio
  - Physical interaction
- Technical aspects of vision and sound on computers
  - Data representation and compression
  - File formats
- Design and implementation of multimedia applications
  - Using HTML+CSS+Javascript
  - See HTML Resources page on Canvas for tutorials

Note: you have seen some of this before, in the context of web design (CSCU9A2). CSCU9N5 goes broader. Content spans a large range of topics, so the course has more breadth than depth

Multimedia with HTML+CSS+JS

- Assumes a basic working knowledge of HTML+CSS+JS
  - Introductory material for self-study available via the HTML Resources page under Learning Content on Canvas
- Six practical sessions will build your skills
  - Multimedia applications
  - Animation
  - Design
  - Testing
- Assignment has major emphasis on the design, rather than the technical implementation
  - Quality of your HTML+CSS+JSS is not assessed

Resources for CSCU9N5

Recommended reading (copies in the library):

- Rogers, Sharp, Preece, Interaction Design. Wiley, 4th edition 2015, or 3rd edition, 2011
- Don't Make Me Think!: A Common Sense Approach to Web Usability, Steve Krug, New Riders; 3rd edition (2014)
- D Cunliffe and G Elliott, *Multimedia Computing*, Lexden, 2005

Others:

- Norman, The Design of Everyday Things. MIT Press, 2002
- Shneiderman et al, Designing the User Interface. Addison Wesley, 5th edition, 2009
- N Chapman & J Chapman. *Digital Multimedia*, John Wiley & Sons, 3rd Edition, 2009

Online resources available from the Canvas module site

Expectations

We expect you to

- Turn up to all classes
- Be prepared for all classes (read lecture notes, attempt practical work, do the tutorial questions)
- Ask questions and participate in class discussions
- Make the most of our time together
- Work outside contact time (*3, roughly) to read around the subject, broaden your knowledge

You can expect us to

- Provide materials in advance
- Engage with you on the topics of the module
- Present clear marking criteria for the assignment
- Provide feedback on your learning, helping you to assess where and how you might improve

Questions?

Any questions about module organisation or content?

Student representative?

Note that attendance will be monitored at practicals and tutorials, to assess student engagement

UX Design

(Part 1)

UX Design

If you write software, you affect how people use your software:

- UX (user experience) is of concern to you - you are a designer.

Guidance for designers falls into three general categories:

- High-level : Theories & models
- Middle-level : Principles
- Low-level : Specific practical guidelines

This design guidance is applicable to wide-ranging situations, not just software/interfaces.

A key component is putting the USER at the centre of the design.

High-level Design

GOMS Model - Goals / Operators / Methods / Selection

- (Card, Moran, Newell - early 1980s)

Four-level Approach

- (Foley and van Dam - late 1970s)

What does this mean in practical terms for designers?

Shneiderman, Section 2.2

GOMS

Goals, Operators, Methods, Selection

Goals:	what state do you want to achieve?
Operators:	what specific individual actions can be carried out?
Methods:	How can those actions be composed?
Selection:	How do we know what entity is being operated upon?

Four-level Approach

1) The *conceptual level* is the user's mental model

2) The *semantic level* describes the meanings conveyed by the user's commands and the output display

3) The *syntactic level* describes how the units that convey semantics are built up into complete commands

4) The *lexical level* deals with the precise way the user specifies the syntax

   Top-down Approach: The designer moves from the higher levels to the lower levels, recording the connections between each level.

What is a Mental Model?

A key goal of good UX design is helping the user to build a **consistent** and useful mental model.

Class exercise

Preece, Chapter 6

Two Main Types of Mental Models

Structural Models:

- How something is structured / built
- Understanding of the inner workings of the object

- Examples:
  - Bicycle brakes
  - Repairers of appliances have structural mental models

Two Main Types of Mental Models

Functional Models:

- How something functions
- Knowing how to interact with the object

- Examples:
  - TV / DVD
  - Calculators
  - Computer? Car? Mobile Phone? MP3 player? Washing machine? Hoover? Dishwasher? …more and more of them

Uses of Mental Models

Most users get by fine with functional models however when a new situation or a problem occurs, a structural model is more helpful for suggesting an appropriate course of action

- The model will only help IF the user's mental model is a true reflection of what is really the case

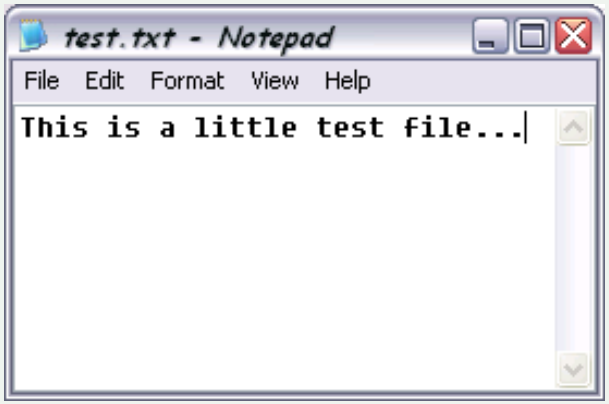
When we design, we must

- try to help the user to develop a productive mental model by designing an interface that reflects that model
- refrain from performing actions that violate the model

Mental Models in Software

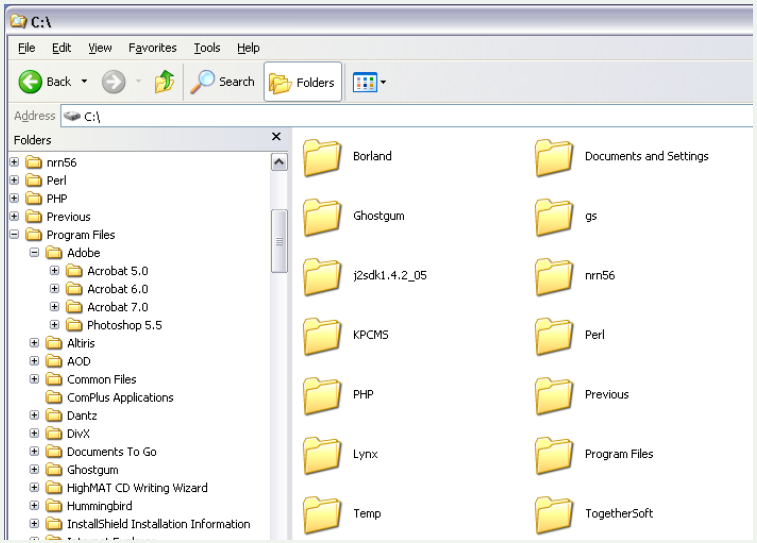
A simple example: Text Editing

- Part of a user's mental model is that text displayed is what is actually in the file



Files stored in a hierarchy of folders

- Helps us to organize our files.



Middle-level Principles

Different sets of principles tend to address roughly the same issues.

We look at Shneiderman's guidelines:

- Principle 1 : Recognise Diversity
- Principle 2 : Follow the Eight Golden Rules
- Principle 3 : Prevent Errors

Shneiderman et al, Section 2.3

Principle 1: Recognise Diversity

Aka "Know thy user" (which is a classic engineering principle).

"Recognise Diversity" does not necessarily mean "cater for all possible users"

- "Know thy user" suggests that one should (where possible) tailor the interface to the needs of expected users.

We consider three useful ways to think about this:

- User profiles
- Task profiles
- Personas

Recognise Diversity - User Profiles

User characteristics:

- Abilities and skills (on the computer), i.e.
  - Novice users
  - Knowledgeable intermittent users
  - Expert frequent users

- Age, gender, physical abilities, education, cultural background, preferences, nationality, training, motivation, goals, personality…

- Environment: physical, social, organisational, technical

Recognise Diversity - Task Profiles

What is the user are trying to do?

Analysis should consider needs of different types of users and common tasks performed.

Recognise Diversity - Personas

(Cooper 1999, Ch 10 Rogers, Sharp and Preece)

Construct some imaginary people who combine your user characteristics and task profiles.

Make the picture rich – make this imaginary person as realistic as possible.

We are all biased

Unconscious Bias

https://royalsociety.org/topics-policy/publications/2015/unconscious-bias/

Implicit Association Test

https://implicit.harvard.edu/

Principle 2
Follow the Eight Golden Rules

1. Strive for consistency
2. Enable frequent users to use shortcuts
3. Offer informative feedback
4. Design dialogues to yield closure
5. Offer simple error handling
6. Permit easy reversal of actions
7. Support internal locus of control
8. Reduce short-term memory load

Shneiderman, Section 2.6

Eight Golden Rules
Consistency

It is very confusing if similar actions are performed differently in different pieces of software.

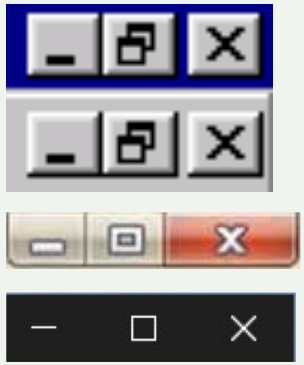
- (this also increases memory load)

Bad Example:

- Website colours of visited links and unvisited links

Good Example:

- Windows control buttons
  - (XP, 7 & 10)



Eight Golden Rules
Shortcuts

Experienced users greatly appreciate having quick ways to initiate actions.

- Examples include reduction of movement from mouse to keyboard and vice versa.

- Bad Example:
  - Notepad did not used to provide a CTRL-S shortcut to save a file. You thought you had saved your file but the application ignored you...

- Good Example:
  - CTRL-X, CTRL-C, CTRL-V in Microsoft applications for cut, copy and paste (also satisfies consistency rule as well!)

Eight Golden Rules
Feedback

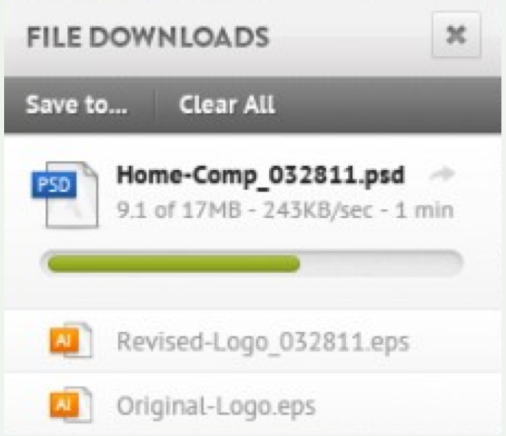
The user should be able to see some result for every action performed.

- Bad Example:
  - Text editors that allow you to type whilst an auto-save is in progress, but you can't see anything on the screen for several seconds.
- Good Example:
  - Current colour selection in a painting program offers automatic feedback if the user changes the colour.

Eight Golden Rules
Closure

Design so that the user can see the beginning, middle and end of (non-instantaneous) actions.

- Example - A progress bar



Eight Golden Rules
Error Handling

If users make an error, the system should (as far as possible) inform the user and offer opportunities for correction.

Good Examples:

- If the user chooses an existing filename when saving a file, a dialog box pops up to see whether the old file should be replaced or not.
- Trying to leave a program without having saved work.
- An error in form-filling should not mean having to fill in the whole form again.

Eight Golden Rules
Easy Reversal of Actions

Every user has performed actions they later wished they hadn't. An UNDO button can be extraordinarily useful!

- Note that undo buttons may reverse just the latest action, or may go backwards through a whole sequence.

Example:

- The Back button in web browsers is a form of undo button; it is highly useful for navigational purposes and is the second-most commonly used feature in a browser.

Eight Golden Rules
Feeling in Control

New users may well not feel in control as they are getting used to a system, but the experienced user strongly wants to feel in control.

Users should be initiators of actions, not merely surprised (and annoyed) if software performs actions without the user's knowledge or permission.

Bad examples from Microsoft Word:

- Hyperlinks automatically underlined and font-selected
- Automatic capitalization and spelling correction
- Automatic extension of text selection

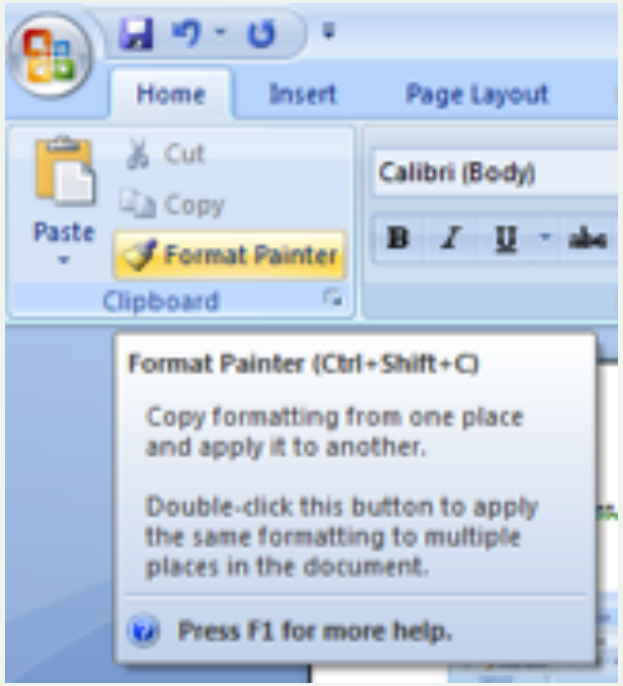
Golden Rule 8 - Short-term Memory Load

Systems should not force the user to remember more than a small amount of information (or force the user to have to look the information up each time)

Humans can only retain a small amount of information in short-term memory.

Examples:

- Helpful pictures on buttons are good
- Tool tips



Principle 3
Prevent Errors

This is different from the 5th Golden Rule – it's about prevention (better than cure!)

- Errors can be prevented by good design.

It is important to understand the types of mistakes that users make.

- Good example: the design of USB memory sticks
  - What happens if the user tries to insert the disk the wrong way?
- Bad Example
  - Putting DVDs into CD drives, putting CDs in upside-down…?
  - The USB-C port – consistent plug, different outcomes...

Principle 3
Prevent Errors

In particular, we can identify three useful techniques for software:

- Correct matching pairs
- Complete sequences
- Command correction

Prevent Errors
Correct Matching Pairs

Examples:

Making some text `<B>bold</B>` will make too much text bold if the `</B>` is omitted or mistyped

Program development environments often provide `{}` match checking, and colouring which shows up keywords, etc.

Phrases used for bibliographic searches
E.g. `computers and (hci or multimedia)`

Prevent Errors
Complete Sequences

Assistance can be provided for the user to complete a sequence of actions to perform a task

Examples:

- Changing all the footers on a set of slides. Rather than having to go through every single one, PowerPoint allows all to be changed at once.

- Wizard that takes the user through the steps for installing new software

Prevent Errors
Command Correction

Aim: trying to prevent users entering incorrect commands (or, more generally, making incorrect choices)

Examples:

- Automatic completion of text entries
- Graying-out of menu options or buttons
- Displaying filenames correctly: