Table of Contents

[User Experience (UX) 2](#_Toc500934564)

[Guidelines 3](#_Toc500934565)

[Navigating the interface 3](#_Toc500934566)

[Organizing the display 3](#_Toc500934567)

[Facilitating data entry 3](#_Toc500934568)

[Principles (Schneiderman’s guidelines) 4](#_Toc500934569)

[Recognise diversity (Know Thy User) 4](#_Toc500934570)

[The Eight Golden Rules of interface design 5](#_Toc500934571)

[Prevent errors 6](#_Toc500934572)

[Ensuring human control while increasing automation 6](#_Toc500934573)

[Theories 7](#_Toc500934574)

[GOMS Model (Goals, Operators, Methods, Selection) 7](#_Toc500934575)

[Four-level approach 7](#_Toc500934576)

[Interfaces 8](#_Toc500934577)

[Metaphors 8](#_Toc500934578)

[Affordances 8](#_Toc500934579)

[Data display 9](#_Toc500934580)

[Visual effects 9](#_Toc500934581)

[Audio 9](#_Toc500934582)

[Web design 9](#_Toc500934583)

[Tools and techniques 10](#_Toc500934584)

[Page design 10](#_Toc500934585)

[Testing 11](#_Toc500934586)

[White box 11](#_Toc500934587)

[Black box 11](#_Toc500934588)

[Techniques 11](#_Toc500934589)

[Interaction styles 13](#_Toc500934590)

[Type vs Style 13](#_Toc500934591)

# User Experience (UX)

*“The overall experience of a person using a product such as a website or computer application, especially in terms of how easy or pleasing it is to use.” Oxford Dictionary 2017*

The experience of a software user is affected by how the software is designed. In order to provide the best possible experience to the user, theories, models, principles, and guideless should be followed.

Both user-interface designer and researchers have accumulated experience, and empirical evidence and theories which can be organized into:

* Low level: **Specific practical guidelines** (good/bad practices)
* Middle level: **Principles** to analyse and compare design alternatives
* High level: **Theories and models** that describe objects and actions with consistent terminology so that explanations can be made to support communication and teaching.

This guidance is applicable to many situations, not just software interfaces.

The user should be put at the centre of the design.

## Guidelines

A guidelines document helps by developing a shared language and then **promoting consistency** among multiple designers in terminology usage, appearance, and action sequences. It records best practices derived from practical experience or empirical studies with **appropriate examples**.

### Navigating the interface

Since navigation can sometimes be difficult for many users, providing clear rules is helpful. Some of the guidelines that could be followed to create user-friendly websites are:

* **Standardize task sequences**: allow users to perform tasks in the same sequence and manner across similar conditions.
* **Ensure that embedded links are descriptive**: link text should accurately describe the link’s destination.
* **Use thumbnail images to preview larger images**: viewing full-size images is not critical, first provide a thumbnail.
* **Distinguishable**: make it easier for users to see and hear content, including separating foreground from background.
* **Predictable**: make web pages appear and operate in predictable ways.

### Organizing the display

The correct organization of the data displayed is also an important part that should not be neglected. Smith and Mosier (1986) offer five high-level goals as part of their guidelines for data display:

1. **Consistency of data display**: during the design phase, the terminology, colours, formats, etc. should be standardized and controlled by use of a dictionary of these items.
2. **Efficient information assimilation by the user**: the format should be familiar to the user and should be related to the task performed.
3. **Minimal memory load on the user**: users should not be required to remember information from one screen to another screen.
4. **Compatibility of data display with data entry**: the format of the information displayed should be linked clearly to the format of the data entry.
5. **Flexibility for user control of data display**: users should be able to get the information from the display in the most convenient way for them, e.g. they should be allowed to sort by column

### Facilitating data entry

…

## Principles (Schneiderman’s guidelines)

### Recognise diversity (Know Thy User)

A good designer must recognize the type of users that will interact with the interface to be created, and design it accordingly. The interface should be tailored to the needs of the expected users. In order to do this, a few steps can be taken:

#### Determine users’ skills level

* Novice or first-time users
* Knowledgeable intermittent user
* Expert frequent user

When multiple user classes must be accommodated in one system, the basic strategy is create a multi-layer (level-structured or spiral) interface.

#### Identify the tasks

After drawing the user profile, the developers must identify the tasks to be carried out. High-level actions can be decomposed into multiple middle-level task actions, which can be further refined into atomic actions that user execute with a single command.

* Frequent actions might be performed by pressing special keys;
* Less frequent actions might be performed by pressing a single letter plus another key (e.g., CTRL + z, ALT + F4, etc.);
* Infrequent actions or complex actions might require going through a menu.

#### Choose an interaction style

Novice or intermittent users:

* **Direct manipulation**: create a visual representation of the action (metaphors).
* **Menu selection**: the user reads a list of items and selects the appropriate action.

Intermittent or frequent users:

* **Form fill-in**: the user enters values into a form, but they must know the permissible values.
* **Command languages**: these provide a strong feeling of being control. The user learns the syntax and can perform complex actions without having to interact with menus or prompts.
* **Natural language**: often used with other interaction styles. This may sometimes be inefficient because of the need for clarification of the commands. E.g. Amazon Alexa, Google Home, etc.

Constructing imaginary users (personas) that will interact with the software interface can be helpful to avoid losing the sight of the objective. Personas, or personae, should have as many details as possible to avoid making them *elastic*, meaning that their objective has been bent to match the design objective/choices.

### The Eight Golden Rules of interface design

1. Strive for **consistency**

Perform similar actions in the same way, and keep the interface consistent.

1. Design for **different users**

Provide shortcuts for experienced users.

1. Offer informative **feedback**

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

1. Design dialogs to yield **closure**

Design actions so that the user understands when the process has started and finished.

1. Prevent **errors**

The system should prevent users from making errors, for instance by limiting certain actions. If the user makes an error, inform that it happened and how they can fix it.

1. Permit easy **reversal of actions**

Users should always have the possibility to use an undo button.

1. Support internal **locus of control**

Users should be the initiatiators of actions: avoid unnecessary automatic actions.

1. Reduce short-term **memory load**

The software should not force the user to remember something from one screen to another.

**CDFCERLM**

### Prevent errors

This is an expansion of the fifth rule.

One way of reducing the user’s productivity due to errors is to provide specific error messages: better error messages improve the success rates in repairing the errors, lower future error rates, and increase subjective satisfaction (Schneiderman, 1982).

*Correct actions*: errors can be also reduced by preventing users from performing actions that could have bad consequences on their work, e.g. cars cannot be put in reverse while traveling forward.

*Complete sequences*: sometimes actions require several steps before being completed, but some users may forget to complete every step. Designers could help the user by creating a single action that will automatically perform a series of steps to achieve the wanted result.

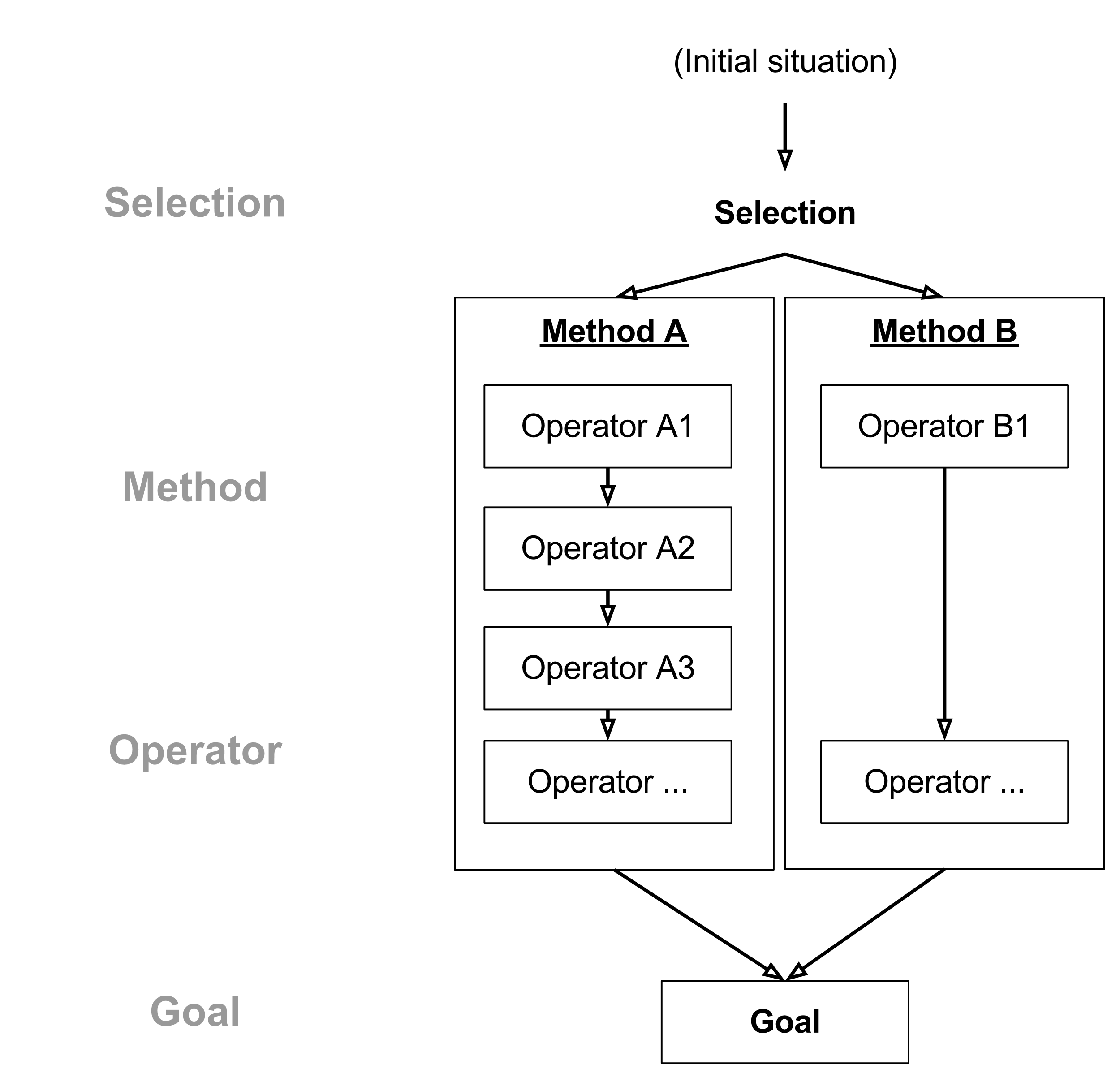
### Ensuring human control while increasing automation

…

## Theories

### GOMS Model (Goals, Operators, Methods, Selection)

*“A GOMS model is composed of methods that are used to achieve specific goals. These methods are then composed of operators at the lowest level. The operators are specific steps that a user performs and are assigned a specific execution time. If a goal can be achieved by more than one method, then selection rules are used to determine the proper Method.”*

**

### Four-level approach

The four-level approach theory consists in separating concepts according to levels. These levels are conceptual, semantic, syntactic, and lexical model (Foley et al., 1995):

1. The **conceptual** level is the user’s “*mental model”* of the interactive system.
2. The **semantic** level describes the meanings conveyed by the user’s input and by the computer’s output display, e.g. deleting an object in a drawing program could by clicking the undo button or by invoking a delete-object action.
3. The **syntactic** level describes how the semantics are assembled into complete commands, e.g. deleting a file by dragging it onto the bin, followed by a confirmation click in the dialog box.
4. The **lexical** level deals with the mechanisms users specify the syntax, e.g. a function key to undo an edit.

#### Mental models

A mental model is what the user believes about the system at hand. Metal models are based on **beliefs**, not on facts. Users base their predictions on the system using their own mental model. One of the biggest dilemmas of HCI is the common **gap between designers’ and users’ mental models** (Nielsen Norman Group, 2010). This gap can be explained by the two different mental models that designers and users have:

* **Structural models** are used by someone who knows the inner workings of an object.
* **Functional models** are used by someone who how to interact with an object.

Functional models work most of the times but, when a new situation or a problem occur, a structural model is more helpful

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

# Interfaces

## Metaphors

Historically, interfaces were only designed to provide the user with the **functions** they needed, but now interfaces have also a **fashion** aspect that is often considered by UI/UX designers. Editors used to be text only and required the user to type a command to access functions, but not every action can be performed by clicking a button with the mouse, or the finger (**WYSIWYG** editors).

To help the user understand what a button will do, **metaphors** are more and more used. The main point of metaphors is that the user is already familiar with the object represented, e.g. waste bin. They are often an advantage because the learning time is shorter when used appropriately, but they can be also confusing if the metaphor does not fit the user case.

## Affordances

Affordances should help the user understand how a button will affect the current view, e.g. an arrow will move the page up or down, a play button will play the selected sound track, etc.

## Data display

As already said in the Organizing the display section, the data displayed should always respect the following characteristics:

* **Consistency**
* **Efficient assimilation** of information
* **Minimal memory load**
* **Compatibility** with data entry
* **Flexibility** of the display area

## Visual effects

Visual effects should be used carefully to avoid being distractive or even annoying. They should be used to catch the user’s attention to certain areas of the screen. For instance, text size could be increased or the colour changed when items of a menu are hovered with a mouse, thus allowing the user to understand which element they are going to click.

## Audio

Hearing is one of the most important senses to humans, so sound can be useful to inform the user about something that is happening in the software. When used appropriately, sounds allow the user to quickly understand if an error happened or if a process was successful.

# Web design

When designing a website, it will be necessary to think about what type of users will visit it. Some of the aspects to be considered are:

* Users
  + Age
  + Experience
* Website
  + Purpose
  + Navigation/Structure
  + Content
  + Reference pages

## Tools and techniques

* **Brainstorming**

Ideas are gathered within a group of people and ratings are given to each idea.

* **Wireframes and Storyboarding**

The website layout is sketched to understand how the content should be organized.

* **Nav Maps**

Nav maps help understand how the user will navigate the website and if all sections of the website are reachable.

* **Task analysis**

It helps us understand how the user will a certain task if it can be improved.

* **Prototyping**

Prototypes are useful for the client to understand how the websites will look and to correct errors in early stages. They can also be useful for the developer to test different technologies.

* **Testing**

See testing…

* **Delivery methods**

DVD, Internet, Kiosk, etc.

## Page design

There are a few elements to be considered when designing a user-friendly website:

* **Navigation**

The user should always know where they are. This could be achieved by using a navigation bar and breadcrumbs.

* **Elements colour**
* **Special needs**

Users with special needs should not be forgotten: the website should be usable by anyone.

* **Size**

The user should be able to display the websites pages without using a specific resolution, so pages should adapt to the device width.

* **Layout**

The layout should be simple and consistent across the website.

## Testing

### White box

The software is tested to ensure that is functioning properly, i.e. checking that all the necessary elements are present, the code works as expected, etc.

### Black box

The software is tested to verify that it complies with the client’s specifications and that all the tasks can be performed with any problem.

## Techniques

Since testing is not always perfect, it is possible to get feedback using other methods that are not explained here. Such methods could include e-mails, mail lists, surveys, etc.

#### Heuristic evaluation

#### Usability testing

Usability tests are performed to ensure that all the tasks can be performed by any user. As explained in the section about Mental models, designers and users usually approach software in very different ways, therefore it is necessary if the users are able to interact correctly with the application interface. Another reason for carrying out tests is that that it is easier (and cheaper!) to correct an error before the software has been released.

The testing phases are often the following:

1. **Determine the** **goal** of the test(s).
2. **Design the test**.
3. **Find unbiased testers** (80% of problems can be found with 5 users - Nielsen and Landauer)
4. **Set up the test**
   1. Prepare the equipment
   2. Provide a script for the test supervisors
5. **Execute the test**
6. **Analyse the results**

#### Think aloud

Users should be asked to explain what they are thinking while they go through the tasks.

#### Usability labs

Usability labs provide professional equipment for testing a software.

#### Online testing

Online testing can be used to test many aspects of an application at once, such as internationalization, specialized tests. They can also obviate to the scarcity of local testers.

#### Online surveys

Surveys could be used to find out what users like or don’t like about an application, what they think it’s missing or could be improved, but the downside is that they only collect opinions, not facts. Sometimes users may not be able to express why or where they are having difficulties.

# Interaction styles

When talking about interface interaction, both the different **manipulation styles** and the **interface components** have to be considered carefully.

WIMP (Windows, Icons, Menus, Pointer).

## Type vs Style

There are different **types of interaction** based on the aim of the actions that the user has to perform:

* Instructing
* Conversing
* Manipulating
* Exploring

The interaction styles are the mechanisms used to interact with the software. The following are examples of **interaction styles** that can be adopted:

* **Typing**
* **Speech**
* **Gesture**
* **Touch**
* **Menu**

## Direct vs indirect manipulation

**Direct manipulation** is achieved by having objects and action continuously visible, so that the interface is an extension of the real world. It is preferred for novices because it’s easier to learn, while experienced users can perform actions quickly. Since it’s more comprehensible, direct manipulation also offers a feel of control to users. The disadvantages are the difficulty of representing actions through objects - which is not always possible, that some actions cannot be performed directly - they need several steps to be achieved, and that it could lead to cluttered interfaces when too many options are available.

**Indirect manipulation** through command languages present advantages such as being very powerful and specific, and allowing experienced users to perform actions very quickly. The obvious disadvantages are the memory required to remember the commands and the steep learning curve for novice users.

## Interface components

### Menus

Throughout the years we have experience many types of interfaces, from the early command line interfaces (CLI) to speech controller interfaces (Alexa, Siri, etc.).

Another way of interacting with the software is through the use of menus. Many types of menus can be used, but they should be chosen appropriately: **hierarchical menus** could be used when a lot of nested options are available, while a **fisheye menu** could be useful when many non-related actions are available.

Types of menu:

* Scrolling
* Iconic: ribbon (Word), jump list (Windows 8/10), etc.
* Hierarchical
* Fisheye

### Dialogs

Dialogs are important components when it comes to actions confirmation. They need to have a meaningful title, consistent layout, standard buttons, they must be easy to dismiss and show the actions available clearly.

# Human factors

In order to design an interface, a software, or a product, it is necessary to understand the capabilities and limitations of the users we are creating them for. Some aspects to consider are:

### Physiology

Different users may have different needs: a child will have smaller hands than a construction worker, therefore interfaces and commands should have an appropriate for the target user.

#### Reaction times

People react differently to audio, visual, or pain stimuli, and each one of these has a different reaction time, so they should be used appropriately according to the reaction we expect.

#### Movement

The size of the target (button, menu, etc.) should be also considered in relation of the function they have: if a button will be used frequently or needs to be used quickly, it would be better to make it bigger, so that users won’t have problems clicking it.

Magic areas of the screen (corners) can be used for important buttons because they are very easily reachable.

#### Disabilities

#### Cognition

#### Attention

#### Memory

#### Perception

# Physical interaction

## I/O Devices

Different input/output devices can be used to interact with an interface.

**Input devices**:

* Keyboard
* Mouse
* Touchscreen
* Trackball
* Trackpad
* Joystick
* Scanner
* Digital pen
* Graphic tablet
* Etc.

Input devices must be designed so that it is comfortable to hold them.

Left-handed and right-handed devices are designed differently.

Precision depends on the device used.

**Output devices**:

* Monitor
* Speakers
* Printer

## Ergonomics

These devices must be comfortable for the task they are used for.

## Printing and colours

Since colours cannot always be represented how they appear in the real world, some techniques to improve them can be used:

### Quantization

Quantization is used to transform **analogical images to digital images**. A large number of inputs is converted into a small set of outputs. E.g. sinusoidal wave converted to a square wave. This technique introduces errors in the output media, e.g.

In order to reduce the errors introduced by quantization, dithering and halftoning can be used:

### Dithering

This technique is used to create the illusion of colour depth in images with a limited colour palette. When two similar **colours are in close proximity**, the human eye perceives them as a mixture of colour.

This can be thought as **pointillist paintings**.

### Halftoning

Halftoning is used to **simulate a continuous colour tone** through the use of dots, varying either in size or distance, so that images have parts with different intensity.

When scanning halftoned printed material, the output has often lower quality because of Moiré patterns. This can be solved by using a higher resolution.

# Why HCI?

Good design choices improve the user productivity, often result in less training, and can sometimes avoid physical danger for the user. The software should be tested often to avoid costs after it’s been shipped.

…

# Multimedia

What the fuck is it?

# Colour

Colour is the perception we have of electromagnetic radiation (ER) through our eyes. Objects absorb and reflect ER in different ways, thus determining the colour of the object.

To represent colour on computers and digital devices, different colour models can be used:

* **RGB** (Red, Green, Blue)
* **CMY** (Cyan, Magenta, Yellow)
* **CMYK** (Cyan, Magenta, Yellow, blacK)
* **HSV** (Hue, Saturation, Value)
* **CIE** (Commission Iternationale d’Eclaringe) primaries

RGB is an *additive system*, meaning that colours are added together to form a new one. Its values are measured from 0 to 255 (1 byte).

CMY is a *subtractive system* because it measures the amount of RGB colour missing from white. E.g. cyan is the amount of red missing from white. That’s why **RGB and CMY are complementary** models.

Even if the same exact model is used on two devices, the result is usually different because of different factors. That is why we need a device-independent colour model. **CIE XYZ** colour space was created in 1931 to address this problem.

The range of colours displayed by a monitor is called **gamut**.

# Graphics

## Physical vs Logical representations

A distinction has to be made between the **physical representation** of graphical data that is how it actually *appear on devices*, and the **virtual representation** that is the representation of the data on a file or in a program. To convert (display) a virtual representation of colours into a device, a process called **rendering** is used.

## Pixels

**Pixels** are the smallest logical unit of display on screen. They are arranged logically on a 2D grid, but they are not arranged in a perfect grid on the actual devices, otherwise we would see visual artifacts.

**RGB** used 4 bytes of data to present a single pixel.

**Transparency** is achieved by combining different colours and carefully considering their alpha channel.

## Bitmap

* Bitmap is a logical representation of pixels in a 2D array.
* They are also known as *raster* images.
* The colour of the image is given pixel-by-pixel.
* They have a fixed resolution.

#### Scaling bitmaps

A bitmap image can be scaled up or down. This process is known as **resampling**. When an image has to be scaled down, we talk about **down-sampling**, while the we have **up-sampling** when the image is scaled up.

The resampled image will have a lower quality because we will be removing pixels from it when down-sampling, while we will add new pixels when up-sampling it. There are three common algorithms used for resampling an image through **pixel interpolation**:

* **Nearest neighbour**: the pixel colour is the same of the nearest pixel.
* **Bilinear interpolation**: the pixel colour is obtained from averaging the surrounding pixels and weighted by their level of overlap.
* **Bicubic interpolation**: similar to bilinear, but more complex.

## Vector-based formats

Another way of representing images is through vectors. They contain descriptions of objects rather than representing pixels. Since they *don’t have a fixed resolution* like bitmaps, it is possible to resize a vector-based image to any size wanted.

They are useful when the objects to be represented are composed of polygons or 3D objects, but they are not appropriate for storing photographs because of the difference of representation depending on the device used and the longer rendering time.

## Graphical issues

When digitizing an image, the system has to convert an image from the physical representation to the logical representation of the software used, i.e. 2D grid. **Anti-aliasing** can be used to solve this problem, thus resulting in images with smoother edges.

The same problems are present when representing lines or text.

## Fonts

Fonts are a way of representing text with different styles. The main types of fonts are:

* **Bitmap**: first type of fonts, not clearly resizable.
* **Vector**: solve the issue of resizing, but anti-aliasing cannot smooth them properly.
* **TrueType/OpenType**: make use of vectors, but also use hints to properly display straight lines.

To display fonts properly, different techniques are used:

* **Anti-aliasing**
* **Hinting**
* **Kerning**: text appearance is improved by changing the spacing of adjacent text

## Bitmap images manipulation

The techniques used to manipulate bitmap images are:

* **Pixel selection**
  + Rectangle
  + Ellipse
  + Photoshop’s magic wand
  + Lasso
* **Pixel point processing**
  + Used to redefined a single pixel independently from its neighbours
  + Uses transformation functions
* **Pixel group processing**
  + Used to redefine pixel colours based on the neighbours
  + Anti-aliasing
  + Blurring
  + Sharpening
  + Morphing
  + Etc

# Data compression

Compression is used to reduce a file size, so that it’s easier to transfer it over the network or to store it. There are two types of compression algorithms:

* **Lossy**
  + The original file cannot be retrieved from the compressed file.
  + Acceptable for images and audio where some parts can be discarded.
* **Lossless**
  + The compressed file can be restored exactly as it was before being compressed.

## Image compression algorithms

* **Pixel packing**: more pixels are stored in the same byte.
* **Run Length Encoding**: the algorithm tries to find repeating sequences of pixels. Doesn’t generally work for photos because they rarely have the exact repeating pixels.

### Dictionary methods

Data is replaced by the index it has in a standard or image-dependant dictionary.

* **Huffman & CCITT compression**
  + Makes use a dictionary of patterns.
  + Developer for fax machines.
  + Common recurring patterns have low indices (less space used).
  + The dictionary is not part of the file.
* **Lempel-Ziv-Welch (LZW)**
  + Also uses a dictionary, but this is created during the encoding process.

Both algorithms are useful for images containing repeating patterns because they are lossless and fast, but they are not good enough for photographic images due to the range of colours and features they contain.

### JPEG

* It was designed for photographic images.
* **Lossy** algorithm.
* Exploits the inability of the human eye to see a full range of colours.
* It has a 2:1 compression ratio using the best-known loss-less algorithm.
* Ratios of 10:1 to 20:1 are good for medium quality images.
* A ratio of 100:1 can be used for poor quality images.
* Every time a JPEG is modified, more data is lost.

How does it work?

1. The image is transformed to a model that has a **brightness component** (HSV).
2. (Optional) Down-sample the colour map by 2 (?) – LOSSY STEP.
3. Divide the image (both γ and colour map) into macroblocks (8x8 pixels).
4. For each block, perform a DCT (Discrete Cosine Transform) on the data (how rapidly the colours changes across the block).
5. Use quantization on the 64 values: the most used frequencies are represented using a range from 0 to 255, while higher frequencies use less colours – BIG LOSSY STEP.
6. The values are stored using RLE and Huffman encoding.

It is not efficient when used with images containing text on a solid background colour because of the high spatial frequency. When JPEG is used, artifacts can be seen across the text and the background.

## Video compression

Videos or movies compression is achieved in a similar way, but also the temporal aspect has to be considered…

# Graphics File formats

Different graphics formats exist because of the different ways of storing and retrieving data they adopt. Some factors to consider when choosing a format are:

* Compression
* Colour model
* Animated or static?
* Etc

## Lossless formats

* BMP
* TIFF

The following use a dictionary compression algorithm to reduce the storage space:

* GIF
* PNG
* SVG

## Lossy

* JPG

## Interlacing

Interlacing is a method of encoding images such that the person that receives the image (e.g. visitor of a website) sees a degraded version of the image (loading layer over layer).

# Animation

Animation is produced by a rapid moving sequence of still images. This was achieved first by using a rotating wheel containing a small number of images at a fixed distance, but now we are more used to watching moving at cinemas or on Netflix. To produce a decent animation, at least 10 fps are needed, but much more are needed to produce a smooth animation (from 48 to 60).

Often, to create animations, only the key frames are created by the animators while the software creates the intermediate images. This process is called **tweening**.

# Sound

Sound can be used as an output (media) or as an input (text-to-speech, recordings, etc.). Some of the common problems with sound recognition and processing are:

* Language, dialects
* One or more people speaking
* Isolated words or continuous speech
* Sound can be used to control interfaces (Alexa, Siri, Google Home, etc.).
* Adds an element of reality when used in games.
* The direction of sound can also be emulated.

## Physical characteristics

* Sound is a pressure wave travelling at 331 m/s
* Frequency that we can hear is between 20 and 20000 Hz

The main characteristics are:

* **Amplitude**: vertical distance of the wave
* **Frequency**: number of waves per second (Hz)

Sound can be:

* Converted to an analogue electrical signal (**transduced**)
* Converted to digital signal (**digitised**)

From a psychological perspective, sound has the following characteristics:

* **Loudness**: how loud it is, also depends on the frequency of the sound (apparent loudness)
* **Pitch**: its tone, usually more frequencies together
* **Timbre**: the nature of the sound

## Digitising sound

* Sound digitised using an analogue to digital converted (**ADC**)
* Sound converted back to analogue using a digital to analogue converted (**DAC**)
* Both of them can introduce alterations

Analogue to digital has two parameters:

* **Sampling rate**
  + Describe how frequently the signal is converted.
  + Measured in samples/second.
  + It must be at least twice the highest frequency of interest (Nyquist sampling theorem) otherwise aliasing occurs.
* **Sample size**
  + Refers to the characteristics of the sample value taken at each sample time (e.g. amplitude)
  + Samples have fixed length (e.g. 8, 16, or 32 bit). The range of the sample will be -128 to +127 for 8 bit.

The signal is digitised through quantization: the sinusoidal wave is transformed into a square wave. Aliasing occurs if the signal is sampled too slowly.

Sampling may be:

* **Linear**
* **Logarithmic**: provides more resolution at lower frequencies

## Sample size

A major problem for the sampled sound is the space it requires to be stored. Data length is proportional to rate \* sample size, so 1 second of sound sampled at 44100 Hz 16 bit uses 88200 bytes/second for just one channel.

## Power and loudness

Dynamic range is the range of amplitudes of sound we are recording (loudest to most quiet). It is measured in **decibels** as 10 \* log10 (P of the loudest sound / P of the quietest sound), where P is frequency \* sample size.

## Compression

Sound is difficult to compress using lossless methods due to unpredictable nature of sound waves. A simple lossless consists in recording the length of the periods of silence (this is *almost lossless*).

**Differential Pulse Code Modulation (DPCM)** computes a predicted value for a sample based on preceding samples

* stores difference between predicted and actual sample
* difference will be small if prediction is good

**Adaptive Differential Pulse Code Modulation (ADPCM)**

Similar to DPCM, but uses an adaptive sample size to make a more efficient of bits, taking account of the change of signal.

**Linear Predictive Coding**

Linear Predictive Coding uses a mathematical model of the vocal tract. Instead of transmitting the original sound, it is converted into a model of the vocal tract, and it is sent over, achieving great compression, but very low voice fidelity.

**Perceptually Based Compression**

This wide area of compression tries to reduce the size, by deleting signals our ear is not capable of perceiving. For instance:

* Too quiet
* Obscured by another sound: masking
* Outside the hearing range

**MP3**, **AAC**, and **MPEG-4** make use of these techniques to achieve good lossy compression, with very little perceivable difference.

**Known Sound Formats**

* **WAV** ─ Windows Waveform. Can be both lossy and lossless using (A)DPCM.
* **SND** ─ NeXT/SUN file format.
* **MPEG** ─ Both in MP3 and the new AAC variant.
* **FLAC** ─ Free Lossless Audio Codec, part of the OGG family.

**Midi**

Musical Instruments Digital Interface is a set of commands used to reproduce notes with different instruments digitally. It is left to the target machine to reproduce the sound. There is no guarantee on support or final quality due to the MIDI implementation often being hardware-implemented. Additionally, it has no support for voice. Like SVG for music. Just way less useful.