
Design Guide for Developers of Educational Software

Russell Beale and Mike Sharples

Contents

Why have guidelines?	3
Basic principles	4
Educational Aspects	5
Teaching and Learning Approach	6
Teaching Strategy	7
Teaching Components	9
Learning styles and preferences	10
Information architecture	11
Screen design	12
Navigation	15
Multimedia	17
Motivation	18
External Representations	18
Text	20
Colour	20
Images	24
Sound	25
Video	25
Devices	26
Accessibility	27
Conclusions	28
References	29

Introduction

Our aim is to improve the quality of educational software by providing some general principles and guidelines for ease of use. Although the Guide is intended mainly for developers of educational software and websites, it should also help users (teachers and students) and adopters to evaluate educational software and to give feedback to the software developers.

Ease of use can be split into three aspects: *usability* (can people use the software effectively and efficiently to perform a task?), *usefulness* (does it improve teaching and learning?) and *desirability* (do people enjoy using it?). This guide is primarily about usability. Designing and assessing the usefulness of educational systems is a complex topic, beyond the scope of a short report. Desirability depends on usability (the user will not enjoy a system they can't use) but it also involves issues such as motivation, fashion, marketing, and peer pressure. Good software is all of these things, and whilst the guidelines presented here focus on usability, they assist the wider issues as well.

Why have guidelines?

Guidelines provide a set of principles, rules and advice on producing a system that fits current good practice. They try and answer the main questions that are encountered by designers in such a way that the results are valuable. They allow us to access expert opinion without necessarily being experts ourselves. Guidelines are very useful, but to extract the maximum benefit from them, their use and limitations should be properly understood.

Guidelines are like cooking recipes. A recipe has prepared ingredients that are combined in a particular order and cooked for a set time at a specified temperature. It should produce a dish that is consistently good and, as the details have been worked out by the experts, it can be recreated by people who have little skill in the kitchen. However, recipes are subject to change. In the long term, tastes change and ingredients come and go out of fashion, whilst on a daily basis good cooks cope with missing ingredients and cater for guests with particular tastes or dietary needs. Indeed, the ability to adapt recipes and cooking styles to produce appetising food from available ingredients is what marks out a capable cook from an average one.

So it is with usability guidelines. They provide a basic starting point and a collection of received wisdom from experts, who have done the research and experimentation and produced a series of basic principles that generally serve them well. Some of these principles are very detailed and specific (e.g. font size and colour), akin to the ingredients, whilst others are more about overall issues (such as screen layout), akin to the cooking instructions. As with recipes, it is important to use the guidelines selectively and modify them to take account of the topic, learner, and context in which the software will be used.

The two general principles that underpin most software guidelines are *know the user* and *know the system*. The first category is concerned with fundamental principles of cognitive reasoning, visual abilities, and physical and psychological effects. The second category is necessarily more specific than the first: interactive systems are always used by people, but the delivery mechanism can vary hugely. Guidelines for this category try to maximise the benefits of the platform (web page, Windows program, palmtop device) whilst minimising its problems.

There are many reasons for adapting, or ignoring, guidelines, but there should always be good reasons for deviating from them. These usually fall into the same categories as above – namely, systems may be designed for a particular set of users who have a known profile that can dictate a different approach, or a particular delivery platform is specified that does not conform to the assumptions given in the general guidelines.

Basic principles

Guidelines can cover high-level issues and be very general, or can focus on specifics. In this report we have tried to move from a set of general principles through to detailed issues. At a high level, the basic principles of usability are rules of thumb, or heuristics. A good set of usability heuristics can be found at http://www.useit.com/papers/heuristic/heuristic_list.html, devised by Jakob Nielsen. A simplified version is presented below. Whilst all these principles are important, the order of the list is not entirely random.

- *Feedback*: inform the user about what is going on using appropriate feedback in a timely manner.
- *Everyday language*: use simple language, avoid technical terms, follow real-world conventions to make things appear logical.
- *Undo*: people make mistakes, so it should be easy to recover to a sensible point.
- *Consistency*: doing similar things in similar places should have similar effects. Also, support the conventions of the specific types of computer and operating systems such as Windows or MacOS.
- *Recognition not recall*: make next steps and critical information visible and memorable. Allow people to recognise what they should do next, not remember what it is.
- *Simple design*: keep things crisp and simple, to minimise the information presented to the user. Make the design aesthetically pleasing to the target audience.
- *Expert use*: provide accelerators (keyboard short cuts and advanced techniques) that allow experts to work faster.
- *Error recovery*: try and design the system to prevent errors occurring, and when they do provide clear messages and suggest appropriate solutions.
- *Documentation*: it is best to design a system that requires no documentation, but complex features or very different systems may need it. It should be well organised, (searchable and well structured), focussed on the task of the user, simple to follow with concrete steps, and concise. It should ideally be available on the system so it is accessible when needed.

The main ideas of 'know the user' and 'know the system' are clear. Systems designed for specific user groups or for specialised educational outcomes, or to be delivered on known platforms, will take these basic principles and refine them in a particular manner, tightening them in places and ignoring other parts. Because of this, it is not appropriate to present one set of definitive guidelines that people can follow to produce excellent software.

Educational Aspects

Users of educational software should focus on their learning, not on how to operate the computer. For this to happen, the software needs to be designed from clear educational principles and considerations. The basic steps in designing educational software are as follows:

Define the educational aims and objectives. What area of the curriculum and topics will be covered? What is the purpose and level of the learning (e.g., to introduce a topic, to explore a topic in more depth, to revise for exams)?

Identify the learning needs. What topics are hard to learn? What misunderstandings need to be addressed? Do the learners need to acquire knowledge, or learn skills?

Decide which needs could be addressed by computer. Is the topic difficult to visualise (e.g., needing a microscope)? Does it relate concrete activities to abstract concepts (e.g. understanding a food web)? Does it involve constructing and linking diverse material (e.g. exploring the causes of the Civil War)? Does it require access to many sources of information (e.g. exploring a population census)? Does it require controlled interactions between dynamic objects (e.g. dynamic physics experiments)? Could it be helped by communication or collaboration between distant learners (e.g. learning a foreign language or culture)? All these could be suited to computer-mediated learning.

Determine the general teaching and learning approach. See below.

Determine the teaching strategy. See below.

Choose the teaching components. See below.

Design and test the software. The larger the project, the more important it is to take a structured approach to design, testing and implementation. For guides to structured software design, see [1, 2]. Whatever the scale, it is essential to produce a clear set of requirements, to consult users (teachers and learners) during the design, and to *verify* (is it free of bugs, robust and usable?) and *validate* (does it meet the requirements?) the software.

Evaluate the entire system. An evaluation can be *formative* (to identify problems and suggest improvements) or *summative* (to assess the quality of the system or to compare it against competing software or teaching methods). The evaluation should assess the complete learning system, including documentation and the contribution of a teacher and other learners, for *usability*, *usefulness*, and *desirability*.

Deploy and maintain the system. This can range from a teacher introducing the class to a set of web resources that s/he has collected, to a company carrying out a full-scale marketing and technical support operation. Whatever the scale, it is important to ensure that the software works correctly on all the computers, that the students have been trained in how to operate it, that they understand how it relates to their learning, and that there are procedures in place for the teacher and student to report problems and queries. Lastly, the system needs to be kept up to date, with technical developments (such as new computers and web browsers), with resource changes (such as broken web links), and with developments to the curriculum.

Teaching and Learning Approach

There are some general approaches to classroom teaching and learning that are reflected in the design of educational software. At their most basic, they can be described in terms of oppositions (see figure 1).

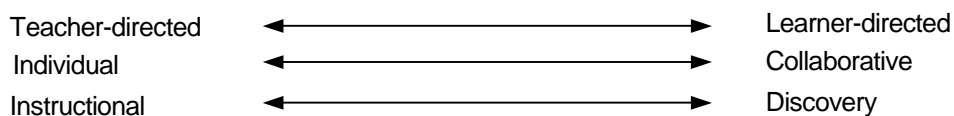


Figure 1. Dimensions of teaching and learning

Teacher directed: the software is designed to be used in a classroom under the control of a teacher.

Learner directed: the software is designed to be used without direct teacher supervision.

Individual: the software is intended for a single learner working alone.

Collaborative: the software is designed to support pairs or groups of learners.

Instructional: the software provides direct teaching of a piece of knowledge of skill.

Discovery: the software is designed for learners to explore concepts, discover knowledge or practise skills.

The dimensions are separate (a software package or website could, for example support teacher-directed, collaborative, discovery-type learning). Each component of the software package will lie somewhere between the two ends of the scale, and the design should make clear how the learning should be carried out. For example, a learner-directed, collaborative, discovery system should provide clear instructions on what other resources and equipment the learners might need, what they should prepare before using the program, how they should collaborate (e.g. taking turns at the keyboard), and what general kinds of discoveries they might make. The more teacher-directed, individual, and instructional the system, the more these instructions can be embedded in the teaching.

Teaching Strategy

A *teaching strategy* (or in the U.S. *instructional strategy*) is a planned set of actions to achieve some educational goal. For educational software, the strategy may either form part of the computer program (for example to present an ordered sequence of teaching and assessment) or it may come from a teacher incorporating computer or web resources into a lesson, or some combination of these. (For a comprehensive glossary of instructional strategies see <http://glossary.plasmalink.com/glossary.html> . For an overview of teaching strategies and instructional design, see [3], available at <http://www.ifi.uib.no/staff/barbara/papers/Euroaied96.html> .)

One way to understand the possible strategies is to view teaching and learning as a set of actions and conversations. Figure 2 shows a (greatly simplified) diagram of learning as conversation. (To take this further, see [4, 5]. These are guides to the design of multimedia in higher education, but the conversational model applies equally to other types of teaching and age groups).

Learning starts with a person engaged in some activity in the world (Box 1), such as following a lesson, carrying out an experiment, or solving a problem. As the learner performs the activity he or she tries out new actions, reflects on their consequences and makes decisions about what to do next (Box 2). The learner is actively *constructing* an understanding of the activities. There is continual interaction and adjustment between the person's thoughts and actions. That is the minimum requirement for anyone to learn: they must be able to converse with themselves, by reflecting on their actions and adapting their actions to their thoughts.

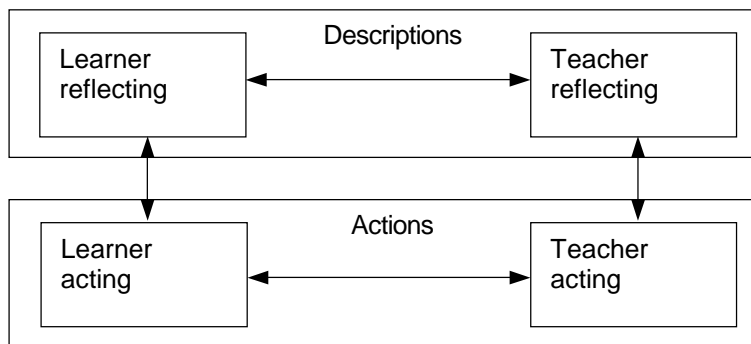


Figure 2. A model of learning as conversation

A more effective form of learning is when two or more people can converse with each other, by interrogating and sharing their descriptions of the world. Assume that two people are working together on some project and also sharing a pool of descriptions (such as notes or results of an experiment). Figure 2 shows a learner and a teacher but it could be two learners. Both people are interacting with the world and conversing at the level of actions, through phrases such as “look over here”, “what’s this?”,

“do that”. They are also conversing at the level of descriptions, exchanging and questioning descriptions of their knowledge through utterances such as “this result doesn’t match the others” or “why did you do that”? Learning is a continual conversation, with the external world and its artefacts, with oneself, and with other learners and teachers.

What is the place of ICT within this conversational space? The computer can be programmed to act as a teacher, as in traditional computer-aided instruction (see figure 3), providing instruction, setting problems and assessing responses. This may be useful for teaching and testing well-defined knowledge or skills.

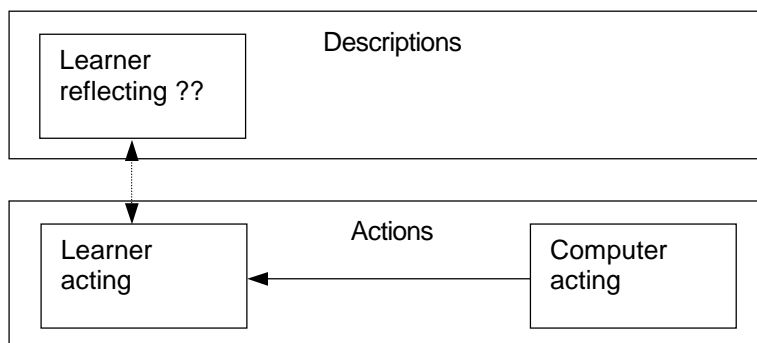


Figure 3. Computer-based training

In this case, the teaching and learning strategy is part of the software. The traditional computer-based learning strategy is *teach and test* (see Table 1). The system *introduces* a topic and sets the learning goals. It then provides a short piece of *teaching*. Next, it *tests* the learner’s knowledge of the taught item. If the learner responds correctly then it moves to the next teaching item. If the learner is incorrect then it provides some *remedial teaching*. This is the starting point for increasingly complex teaching strategies, such as *branching remediation*, *socratic dialogues*, *generative* and *adaptive tutoring*.

<i>Introduce</i>	French words for animals
<i>Teach</i>	The cat : Le chat
<i>Test</i>	What is the French for “the cat”
<i>Remediate</i>	No, it is “le chat”

Table 1. Teach and test strategy

The problem is that this only covers part of the conversational space. The computer can only hold a limited dialogue at the level of actions: “look here”; “what’s this?”; “do that”, but is not able to reflect on its own activities or its own knowledge. Because it cannot engage in a conversation at the level of

descriptions, it has no way of exploring learners' misunderstandings nor helping them to reach a shared understanding. So, this type of learning strategy needs to be carefully managed, with a human teacher adding the missing conversations.

An alternative is for the ICT to provide tools and resources to support conversational learning (see Figure 4). These can include *simulations* or dynamic *models* to be explored, or *resources* such as databases and web pages for analysis or discussion. Here, the emphasis is on the computer assisting rather than replacing the human teacher. Its problem is that it requires the teacher to manage the learning, muster the resources, and provide the teaching strategy. An ideal teaching system will enable all the conversations and interactions shown in Figure 4, through a combination of computer (providing resources, tools and teaching) and human teacher.

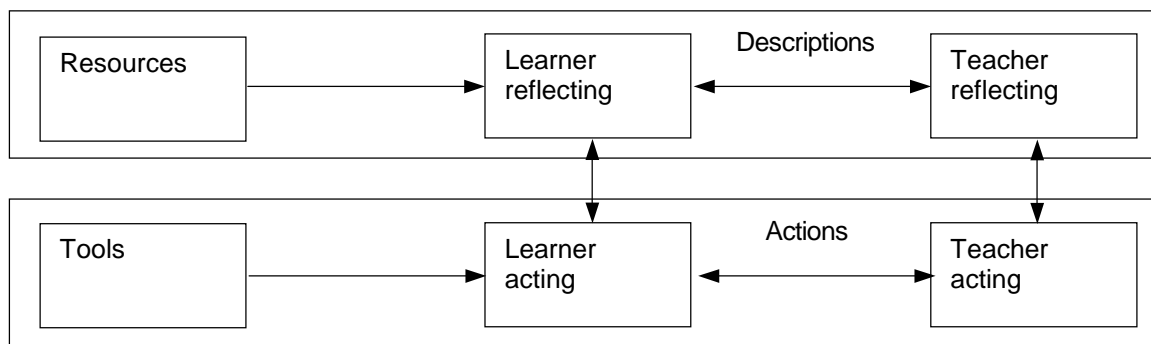


Figure 4. An ideal teaching system

Teaching Components

The teaching components are the basic building blocks of educational software. By fitting them together in an appropriate collection or sequence a software designer can produce a package to achieve the educational goals through the most appropriate teaching strategies. Below is some, but by no means all, of the most important components.

Advance organiser: A means of preparing the learner, for example a list of topics to be covered, a map of the resources, or a worked example.

Learning resource: A web page or other piece of online information designed for learning.

Self-assessment question: A question for the learner to test their understanding, often with a link to a model answer.

Computer-marked assignment: A quiz or test in a form, such as “multiple choice” or “fill in the blanks”, that can be marked by the computer program.

Simulation: A dynamic representation of some complex system, such as an electrical circuit.

Model: As above. The difference between a simulation and a model is that in a simulation the learner can only create structures and alter settings or parameters (for example, adding components to a circuit and adjusting the voltage or resistance). In a model, the learner can change the underlying system that drives the simulation (for example, by altering the Ohm's Law equation so that the circuit behaves in an "unreal" way).

Concept map: A visual map of ideas or topics and their relationships. It can be presented as a fixed map (for example, for use as an advance organiser), or can be constructed by learners as a means of exploring their knowledge.

Interactive tutorial: A computer-generated dialogue with the learner. Creating a convincing interactive tutorials where the computer can respond to a learner's queries and misconceptions is one goal of research into Intelligent Tutoring Systems.

Bulletin board: (or discussion area) An online facility for learners to post, and respond to, messages, and group them into "threads" of themed discussion.

Chat area: An online facility for learners to communicate at a distance, by typing and responding to messages.

Learning styles and preferences

Each of these components can be implemented through a combination of media, including text, images, animations, sounds, music and video. The challenge for an educational software designer is to chose the components and media that best support each student's *learning style*. Gardner's theory of multiple intelligences suggests that each person possesses different forms of intelligence to varying degrees (see <http://tip.psychology.org/gardner.html>). He proposes seven primary forms: linguistic, musical, logical-mathematical, spatial, body-kinesthetic, intrapersonal (e.g., insight, metacognition) and interpersonal (e.g., social skills).

According to Gardner, learning and teaching should build on the particular intelligences of each person. For example, a person with strong interpersonal intelligence will learn best through collaboration and social interaction. A person with musical ability might respond to teaching accompanied by music or structured around musical harmony. A further implication of the theory is that assessment of abilities should measure all forms of intelligence, not just linguistic and logical-mathematical.

Learners also have preferences as to how their teaching material is organised. Some prefer to learn in a *holist* way, first getting an overview of the topic and then exploring how it is made up from individual pieces of knowledge. Others prefer to fit together bits of knowledge or skill, step by step in a *serialist* manner, until they gain a complete understanding (For some notes on learning styles, see <http://staff.psy.gla.ac.uk/~steve/lstyles.html>).

In theory, it may be possible for a human teacher or a computer program to discover a learner's styles and preferences, but in practice any teaching system must compromise, either by mixing different media and teaching strategies, or by providing a flexible package of teaching components and allowing

the learner to choose their own method of learning. For this to be successful, it is important to provide clear topic maps and aids to navigating through the material, to ensure that the learner does not get confused by a jumble of multimedia resources and teaching aids.

Information architecture

As discussed above, information should be presented in a clear and logical manner. For practically any non-trivial system, this will mean that it has to be clustered and structured so that manageable chunks of related information can be presented together, with a coherent transition from one chunk to the next.

There are many ways of clustering information. For example, a web site for a pet shop could choose to cluster in terms of product (food, bedding, toys, health) or in terms of pet (cat, dog, fish, hamster). Both schemes are logical and comprehensible, and it should be easy to find what you want. It makes sense to continue the theme through the different levels: within 'food' could come dried food, tins, packets, and fresh; if the other route were chosen, you may expect to find tropical, marine and freshwater fish subcategories. The choice of which approach may depend on the number of categories that you require, user preferences, graphic design considerations and so on. Typical clusters should partition things up into not more than seven to nine categories.



Figure 5. Platform conventions provide strong guidelines for menu content. This is a typical system. Notice too how items have been sub-clustered into smaller, related chunks within the menu.

The principles of information architecture apply to the design of menus as well. Related items should be clustered under an appropriate menu heading. For Macintosh or PC systems, there are strong conventions as to what goes into many of the menus and their titles, see figure 5.

Screen design

Screen design is a critical area for interactive systems, as it is here that users form first impressions as to the 'look' and 'style' of the system. The screen design is affected by aesthetics and fashion as well as function, and forms a critical component in the overall desirability of a system, as well as its usability and effectiveness.

Screen design has to accomplish two tasks: it has to present information to the user, and also assist interaction with the system. It therefore encompasses issues of graphic design as well as user psychology.

Many of the components used in interactive systems have their own guidelines, often produced by the platform vendors (e.g. the Apple Macintosh Human Interface Guidelines available at <http://developer.apple.com/techpubs/mac/HIGuidelines/HIGuidelines-2.html> or the Microsoft Windows User Experience [6]). These encourage consistency and familiarity across applications developed for that platform, always helpful. The different elements of the system design should be put together coherently, so that the dialogue between the system and the user is consistent. Mixing dialogue styles (e.g. menu systems with natural language with entries typed at a command line) is rarely satisfactory.

We can identify a few key issues for screen design; not as a definitive list, but as examples of the issues to consider. More information can be found in texts such as [7].

Screen size: Computer monitors come in a variety of different sizes, both in physical dimensions, and in resolution. Resolution refers to the number of separate picture elements (pixels) that make up the image – basically, the more, the better the detail – whilst the screen size refers to the physical dimensions of the display area of the monitor. Size is generally less critical from a design viewpoint than resolution, though too high a resolution on a small screen makes small details illegible. Resolutions range from the smallest at 800x600 up to a typical maximum of 1200x1600. When designing software, screen estate is a precious, scarce resource, and you are likely to want to use as much as possible. But if the delivery platform is not known, you may be forced to design within an 800x600 area, to ensure that all people will be able to see the information. There are many web sites that are designed to this criterion. On larger resolution screens they are typically surrounded with a white border, focussing attention onto the system in the centre. However, most computers have resolutions of at least 1024x768, and if you know your users and your systems, you may well be able to make a decision to design to this size instead, trading off occasional users outside of the target audience who cannot use the system against improved interactivity for the key audience.

System conventions: Menus are generally found across the top of the screen, with icons representing shortcuts below those. Supplementary information is placed on the screen periphery, with key information taking prime position in the middle. Users tend to scan screens in the same manner in which they read (usually top left to bottom right), so key information should come near the top.

Screens and scrolling: Designers often have to choose how to break the system down into its component screens, in itself a difficult decision, and this is compounded by choosing whether to make more information available on one screen than can be displayed at once. If this is done, users have to scroll to see the additional information. It is generally wise to take the scrolling route when presenting closely related information that it makes little sense to split up, and when rapid movement between different parts may be required, though information that is over three or four screens long is often better split up. However, scrolling should be avoided when it moves critical information off the bottom of the screen, or when the information presented is not worth the effort of scrolling.

Most modern PCs allow multiple 'windows' to be available, allowing users to flick between different parts of a system at once, or overlay information; if you allow users this flexibility, remember not to rely on specific information in certain screens being noticed by the user, as they may have closed that window earlier.

Popup windows containing advice or information are best avoided as they suddenly force themselves to the front and grab the users attention, diverting focus from the task at hand. Unless absolutely necessary, a dialogue box (that pops up to let the user make a choice) should not disable the program forcing the user to deal with it immediately. A good use of a dialogue box would be to confirm the deletion of a file, when you want the user to stop and think for a moment.

Graphic design: Design elements should not obscure information or usability. Good graphic design is hard to do well, requiring specialist skills, and often good designs simplify complex information and concepts, presenting them succinctly and clearly. Design is often used to build or emphasise a coherent brand or user experience. This is not a problem, and can often be beneficial, as long as the design does not obscure the message it is trying to present.

Menus and icons: Menus, discussed earlier, provide access to functionality. Icons provide graphic representations of shortcuts to functionality or new parts of a system. They differ in both their representation (text versus graphic), and because icons offer one-touch access to functions that often require the user to navigate through more than one menu. Icons are therefore most often used by experts to accelerate their working and should be *recognisable*, *memorable*, and *clear*. Icons bring functionality to the front of the interface, and so present more options to the user. Figure 6 shows icons from a common word processing package, and the icon bar from a professional product that is designed for expert use.

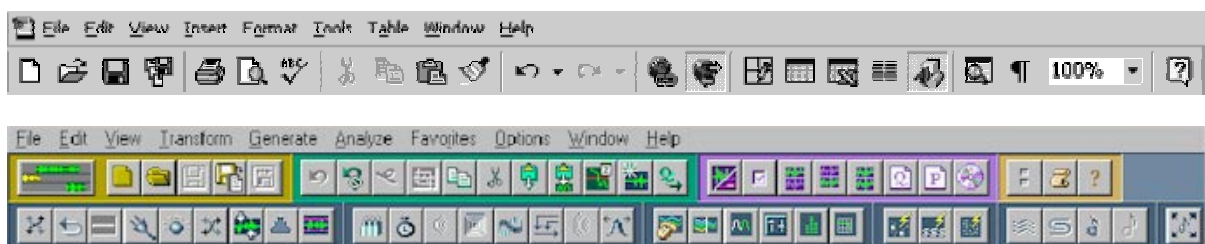


Figure 6. Menu titles and icons from two software programs: one a common but powerful word processor, the other a semi-professional audio editing and mixing system.

Figure 7 shows an example of a basic online learning resource: a screen that teaches about primary and secondary colours. The screen has been designed to aid readability and comprehension. Below are some of the screen design guidelines that the example has followed.

- Design the layout for an 800 by 600 pixel screen, the standard on older computers;
- The centre of the screen is the focus of vision, so it should hold the main teaching text;
- Divide the screen into blocks (see figure 8) that hold the main elements;
- Use colours rather than lines to separate the main elements, with muted pastel colours so that they don't intrude on the teaching material. (Software for children often uses bright colours for borders and buttons, but it is generally better to have eye-catching colour in the teaching material rather than the fixed page elements);
- Provide page elements that help the learner to know where they are in the material (the title and page number) and how far they have to go (the highlighted contents item and "page 2 of 10");
- Allow the learner to move to the next and previous page, and to any main topic;
- Design and label the buttons to show the direction and page number;
- Put the elements in positions that are familiar from websites: logo at top left, contents bar at the left, title and subtitle at the top;
- Make the typesize for the teaching material at least 14 point, the minimum readable size;
- Use a readable font that is available on most computers, such as Arial;
- Restrict the teaching text to 15 lines, with no more than 7 words per line;
- Make the text is ragged right rather than justified (lined up to the right). Ragged right is more readable as it does not add uneven spaces;
- Confine the teaching on each page to conveying a single concept, or pair of related concepts (in this case, primary and secondary colours);
- Where appropriate, provide graphics or animations to illustrate key concepts by the learner clicking on the highlighted text (e.g. "primary colours" and "secondary colours");
- Provide each illustration with a caption that relates to the teaching text.

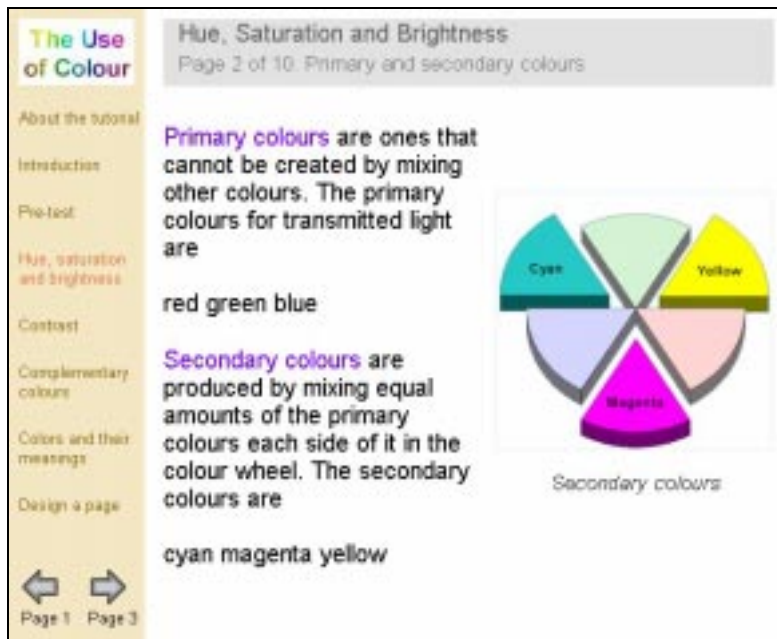


Figure 7. A simple learning resource.



Figure 8. Elements of a learning resource screen.

This is only the most basic example of an online learning resource, but many of the same principles apply to more sophisticated learning material such as games or simulations. The key guidelines for page design are: keep it simple, consistent, uncluttered, and focused on the teaching.

Navigation

Navigation allows people to move about the system, and should be both map and compass - showing you where you are, what's out there, as well as where to go and how to get there.

Information architecture has a great effect on the navigation of a system. The clustering of information produces a tree structure, with each branch leading deeper into the system and closer to an individual item or concept. As users move about this structure, whether learning things or browsing a shopping site, they should be able to move in a suitable manner (which does not necessarily mean anywhere).

Desirable characteristics in any navigation system are the ability to go back to the previous point, the ability to reach the 'home' screen simply, some indication of where in the structure they are, and guidance on where to go next. This is typically achieved through a navigation section, placed either across the top of the screen or down the left side, with buttons or hypertext links to take users where they want to go.

One useful and common navigational element is a menu of choices, often presented as tabbed pages across the top of the screen or down the side. Figure 9a shows a web system with tabbed pages, each of which is a separate area of the system that the user is taken to. Figure 9b shows a vertical navigation bar in a different system. Either can be used, but their use should be consistent within the application and across the platform.



Figure 9a. Tabbed pages across the top allow access to very different areas of the system, whilst a link to key home pages is down the left side.



Figure 9b. A navigational bar down the left side of the page

Web-based systems often identify each of the branches that have been taken, as in figure 10. With each of these clues being a hypertext link, it is easy to move back up the hierarchy, easy to see where you are, and makes the classification scheme more apparent, which in turns aids people in working out where to find what they are looking for.



Figure 10: Typical navigational elements on a web page. The search facility allows deep access quickly, whilst category text tags ([Home](#) > [Recreation](#) ...) show where you currently are and indicate the structure of the information.

Another useful navigational tool is the ability to jump straight to a certain point. In educational systems, this is often returning to where you left the teaching (for instance to look at help or a glossary). On the web, it provides direct access and is usually implemented through a site-specific search button. On reaching the specific page, however, it ought still to be clear where in the system you are, using the navigational hints discussed above.

Note that menus and icons are also navigational items, allowing users to move to different parts of the system – the same principles still apply, of making it clear where they are and allowing them to return easily.

Certain systems, usually those that take people through a prescribed learning route, take a more rigid approach to navigation, producing limited options for continuing, repeating or exiting the system only at certain stages. This is accomplished through dialogue boxes. Whilst this style of system is easy to build and monitor, it inhibits learning strategies and is becoming less common. Its use should be carefully thought through if it is suggested as appropriate.

Multimedia

Much has been written about the value of multimedia (graphics, sound, animation, video) in education. This section offers some general guidelines that are based on sound theory or experiment.

There are two good educational justifications for multimedia software: it can increase *motivation* and it can assist learning through use of appropriate *external representations*. The difficulty is that the two do not necessarily go hand in hand. Adding multimedia elements may not help learning and providing educationally valid external representations may not motivate learners. Thus, it is wise to treat the two aspects separately, then see how they can best be combined.

Motivation

Malone produced a seminal study of what makes software fun to learn [8]. He concluded that there are three key aspects of motivation: *challenge*, *fantasy* and *curiosity*.

Challenge involves setting appropriate goals. Simple teaching programs should provide an obvious and compelling goal; a more complex system should be structured so that users will be able to generate goals of appropriate difficulty (as with the simulation game *SimCity* where users can build cities, manage them, and overcome problems such as natural disasters). The most motivating goals are often practical or fantasy ones (such as reaching the moon in a rocket), rather than simply achieving a skill (such as solving arithmetic problems). The users must be able to tell whether they are getting closer to a goal. Challenge can also come from:

- varying the level of difficulty;
- providing multiple levels or stages of goal;
- hiding information which the user has to find;
- randomness.

Fantasy has the appeal of satisfying an emotional need. This can depend greatly on age, gender, culture, and individual preference, so should be used with great caution. For example, an effect as simple as adding fairground music to a “darts” arithmetic game in one of Malone’s studies was generally like by the girls but not by the boys.

Curiosity can be provoked through *mystery* (for example, leaving clues to solve a puzzle), *inconsistency* (making the learner confront preconceptions, such as “yes, most plants need light to grow, but are there some that don’t?”), and *parsimony* (giving a number of examples of a general rule and then helping the learner to discover the rule).

External Representations

External representations (ERs) are observable, and often manipulable, structures, such as graphs, tables, diagrams or sketches, that aid problem solving or learning. They can be created by the learners (as when a child adds “working” to a subtraction problem) or provided for them. What is important for learning is not the ERs themselves, but how learners interpret and interact with them [9].

ERs can assist learning by:

- *reducing mental effort*, by taking over some aspects of the problem or computation as with spreadsheets;
- *enabling learners to see relationships*, such as clusters, trends, and missing values;
- *constraining interpretations*, by drawing attention to relevant parts of a problem or ruling out some incorrect interpretations.

To see how ERs can help to indicate relationships, consider a simple problem of seating six people for dinner round a circular table. The relationships can be stated in words, as follows:

Freda sits directly to the right of Ewan.

Alice sits opposite Colin.

Colin sits between Betty and Ewan.

Dan sits directly to the left of Betty.

Who sits either side of Alice?

It is almost impossible to solve that problem without drawing a diagram, but once the diagram has been drawn (a circle, with six equally spaced placings) then the relationships can easily be added to it and the question answered (we leave it for you to work out).¹ Furthermore, the diagram can then be manipulated to answer further questions such as “If Alice and Betty swap places, who is now opposite Alice?” The diagram appropriately constrains the relationships by providing the circular table with six equally spaced settings. This is a simple example, but the same properties apply to more complex and productive ERs, for example Venn diagrams in set theory.

A great value of computer-based ERs compared with ones on a paper or whiteboard is that the computer can produce representations that are *animated* and *linked*. Animations can show causal relations (for example how eclipses are caused as the earth moves round the sun) and how events change over time (such as the path of a bouncing ball). Linked ERs make a dynamic connection between a model or abstract representation and some more direct event or perception. For example, the ECOi project designed software that linked a food web with a simulated pond, so that learners could add the relationships between items on the web and see how that affected the pond (e.g. the tadpole eats the weeds), or reverse the links and watch “impossible” events (the weeds eat the tadpole) <http://www.cogs.susx.ac.uk/users/yvonner/ecoihome/IMMI.html> .

Animation and linked representations need not be restricted to dynamic events. Figure 11 shows a very simple animation, part of a tutorial on colour, that links bars for selecting combinations of red, green and blue to a display that shows the resulting colour.

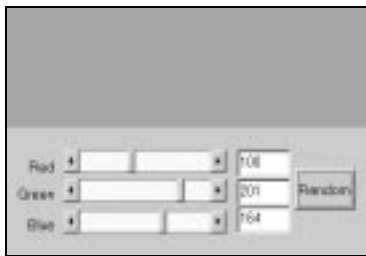


Figure 11. An animated colour selector, at <http://www.eee.bham.ac.uk/sharplem/colourtutor/RGBexp/RGBframeset.htm>

¹ The answer is that Dan sits to the right of Alice, and Freda to the left.

Providing learners with the opportunity to create and alter their own ERs can bring benefits by allowing learners to save and examine ideas, re-order, translate and re-represent information, and keep track of their progress through a problem [10]. An example is the use of computer-based concept maps (see <http://cmap.coginst.uwf.edu/info/>) to organise topics and ideas. It is important that the representations should be *meaningful* to the learners. Just showing a graph or providing a concept mapping tool is not enough; the learner needs to understand the meaning of the objects and their relations, and how they can usefully be created, explored and altered.

The next sections cover some basic design guidelines for elements of multimedia.

Text

Text is the bedrock of education. It can inform, instruct, persuade, provoke and enlighten. Unfortunately, the computer screen is not a good place to read it. Although studies have shown little or no difference in *speed* of reading from the screen compared with the page, the screen is worse for *interpreting* and *using* the text. It is not easy to make notes on the screen, or to keep track of where you are [11] (see <http://www.unifr.ch/perso/corti/docs/epc-1997-101.pdf> for a summary).

When text is the main medium for teaching, then either it needs to be presented in a form that can be printed out (e.g., a user guide provided as a .pdf file), or short and interactive (e.g., a chat area or discussion group), or it must be carefully designed and structured to aid readability. Basic guidelines include:

- use a sans serif font, such as Arial;
- make the teaching text at least 14 point size;
- make the text ragged right justified;
- separate paragraphs by spaces;
- use bullet points to indicate lists;
- use bold or italics to emphasise important concepts;
- avoid shadow or outline characters (they just look amateurish).

For more on the design of instructional text see [12].

Colour

Colour is an important element in the design of educational software. By comparison with modern multimedia packages, computer-based learning programs of the 1980's look dull and uninspiring, like an early 20th century textbook. But colour should be used wisely and sparingly, to enhance the overall

design and aid understanding, rather than just to brighten up the screen. The over-riding guideline is “make colour meaningful”.

Colour has two kinds of meaning: *aesthetic* and *cultural*. Aesthetic meaning is when colours appear to clash or complement each other, or when they seem to recede or stand out from the screen. Cultural meaning is where colour relates to activities, attributes or objects in the world, such as “red for danger”.

Colours on a printed page are caused by light from another source being reflected by inks in the page (reflected colour). Colours on a computer screen are created by light being emitted by the screen to the eye (transmitted colour). All reflected and transmitted colours can be composed from three “primary colours”. For reflected light these are red, yellow and blue. For transmitted light, the primary colours are red, green and blue. A television-type computer display is produced by beams of electrons hitting tiny dots on the screen that give out light in the three primary colours. All the colours you see are composed of combinations of these colours. For example, adding equal mixes of each pair of primary colours gives the “secondary” colours (red+blue=magenta; red+green=yellow; blue+green=cyan). Mixing these colours gives further colours, and so on. Combining pure red, green and blue transmitted light gives white.

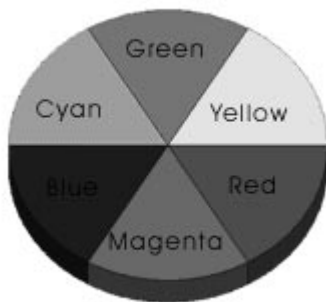


Figure 12. Primary and secondary transmitted colours

There are various methods to describe the full range of colours. One way is to indicate the amount of red, green and blue. A better method for purposes of design is to characterise a colour by the three dimensions of *hue*, *saturation*, and *brightness*. The different colours are known as *hues*. To get the full range of colours, you can also alter the *brightness* of each colour. You can also add different amounts of white or black, to get levels of *saturation*. Figure 13 shows a *colour wheel* that depicts the range of hue (around the wheel) and saturation (towards or away from the centre). For any point on the wheel, the colour can be made brighter or darker.

When designing a screen, smaller objects need to be more saturated to be visible. Avoid using fully saturated colours for large blocks of text, as it can be uncomfortable to read. A headline in a saturated colour will draw attention. Less saturated colours tend to look warm and subtle, while more saturated ones look vivid or garish.

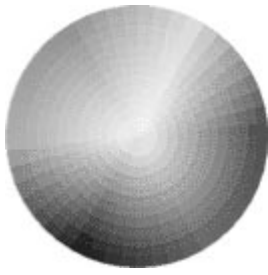


Figure 13. A colour wheel showing hue (around the wheel) and saturation (towards or away from the centre).

Our visual perception is such that colours at opposite points on the colour wheel appear to clash (Figure 14a), and colours close together on the colour wheel tend to be difficult to tell apart (Figure 14b). The colours that go best together tend to be around 30 degrees apart on the colour wheel (Figure 14c).



Figure 14. Comparison of colours.

Colour can be used to good effect to distinguish between alternate lines of text, for example in a table, or for titles and section headings. You should avoid using colour to emphasise a word or phrase within a block of text, such as this, because readers may think it is a web link. Similarly, you should avoid using colours that are too close to the standard link colours of blue and violet.

Colours have complicated psychological effects on the viewer. Many colours evoke meaning. Some meanings are conventional and obvious, such as red for danger. Some colours have meanings only in a specific culture, or their meaning changes across cultures. In many Western cultures death is associated with the colour black, but in China white is the colour of death (see <http://library.thinkquest.org/50065/psych/meaning.html> for the meaning of colour in different cultures).

Table 2 shows the meaning of some colours, and ranges of colour, in different cultures, with examples of websites that make use of the colour for its association.

Other general design guidelines for use of colour include:

- Design the interface using shape, patterns, position of objects and size. Use colour for additional information.

- When using colour for information, use colours that are equally separated around the colour wheel, e.g. green, yellow, orange, red, violet and blue
- Using red and blue together creates a depth effect (red appears to stand out, blue to recede). Avoid using saturated red and blue together, unless the depth effect is intended.







	Western Europe & USA	China	Japan	Middle East	
Red	Romance, sex, fire, heat, danger	Joy, festive occasions	Anger, danger	Danger, evil	 www.pnc.com.au/~kima
Green	Environment, countryside, conservation, safe, go	Youth, growth	Future, youth, energy	Fertility, strength	 www.greenpeace.org
Yellow	Sunshine, holidays, caution, slowness, cowardice	Honour, royalty	Grace, nobility, childish, gaiety	Happiness, prosperity	 www.jamaicatravel.com
Blue	Water, sky, coldness, masculinity, calm, authority	Strength, power	Villainy		 www.nationaltrust.org.uk
Black	Death, evil, elegance, mysticism	Evil	Evil	Mystery, evil	 www.blairwitch.com/
White	Purity, virtue, cleanliness	Mourning, humility	Death, mourning	Purity, mourning	 www.weddingchannel.com/

Table 2. The meanings of colours in different cultures (adapted from <http://library.thinkquest.org/50065/psych/meaning.html>)

Images

One way in which computer-based learning has a clear advantage over print is in displaying and interacting with colour images. Full colour printing is expensive and although the quality of a printed image can be very high, the computer can display millions of colours by transmitted light so that the colours appear vivid and shining. But presenting images on the computer, particularly if they are delivered over the web, is not straightforward. This is due to a number of factors including: the low screen resolution compared with print; differences in the range of colours (some colours that can be printed with inks cannot be displayed on the screen); the time it takes to send large colour images over the internet; and the fact that older computers and web browsers only display a limited range of colours. The main considerations when preparing pictures for the screen are: *adjustment*, *compression*, and *rendering*.

The first stage in producing a screen image is to *adjust* the image so that it looks natural on the screen. This may involve altering the colour balance if it was taken in artificial light, or boosting the mid-range of brightness to show more detail. Image manipulation software such as Paintshop provides a range of tools to carry out these adjustments.

A large uncompressed high colour image can take 2 minutes or more to load from the web over a standard modem. To reduce this time and the memory space needed to store the picture on the computer disk, images need to be *compressed* by computer algorithms that remove information which is least important to the human eye. The two main compression algorithms are JPEG and GIF. In brief, JPEG is best used for photographic images and GIF for graphics.

Older Web browsers are restricted to 216 colours out of the full range, so to ensure that a colour picture or graphic will *render* correctly on all computers, it need to be converted to these 216 colours. Paintshop and similar software provides tools to convert images to browser safe colours.

A great advantage of JPEG is that you can choose the degree of compression. The more compression, the smaller the image but the lower the resolution. JPEG also stores a picture as a full colour (24 bit) image, unlike GIF, and then uses a technique called "dithering" to display it on a reduced colour web browser. Dithering uses tiny dots of web safe colours (like a pointillist painting) to give the impression of a continuous colour picture (see Figure 15).

JPEG is not suited to showing graphics, because it is a "lossy" compression method that introduces ripple effects round the edges of blocks of colour. It is better to choose browser safe colours for a graphic and then save it as a GIF image.

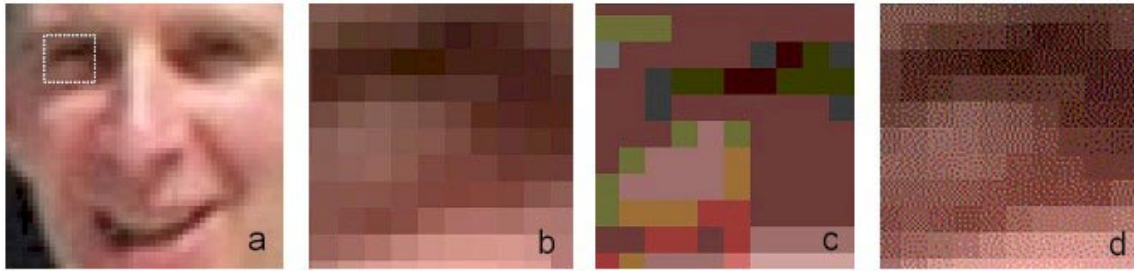


Figure 15. a: Region from original photo, b: expanded eye area in full colours; c: in browser safe colours; d: dithered in browser safe colours

Sound

There are two main reasons for using sound in educational software: to provide information that cannot be conveyed using another medium, and to give a sense of mood, person, event or place to a multimedia event [13]. Teaching music by playing recordings of performances, or a foreign language by presenting recorded interviews are examples in which sound is the only effective way to communicate the information. Adding background music to a computer game or simulation is an example of sound setting the mood. Emitting a low-key “click” sound when a button is pressed can give a sense of reality and confirmation to the event.

The ECOi project worked with children as design informants [14] and a suggestion the children offered was to add sound to basic multimedia events and button presses. In the ECOi pond simulation, whenever one fish ate another it was accompanied by an appropriate “chomping” sound. A learner could alter the graphical food web that was linked to the simulation, so that energy was transferred in the wrong direction. The sounds of plankton eating fish or small fish eating larger ones caused much hilarity and made the learning more memorable.

Good quality sound is known to enhance a user’s experience of multimedia [13], for example in a study of video games where users claimed that the graphics were better when the sound was improved. But avoid using repetitious or annoying sounds, such as announcing “well done” in the same way for each correct answer.

Video

Video holds great promise for online learning, with uses that include foreign language teaching, personal and social education, and videoconferencing. To achieve its full effect, the video needs to be large (ideally full screen), of at least video recorder (VHS) quality, and at 25 frames per second (fps). Many older computers cannot play video at this quality, nor can it be sent over a modem connection to the internet. Even if the computer can cope, it eats up disk space: one minute of high resolution video takes up around 60Mb.

Any alternative requires some compromise in quality: either the picture size is tiny or fuzzy, or the video is jerky. Studies have shown [15] that communicating by poor quality video may offer little or no benefit to performing collaborative tasks. It can also be demotivating. If you tell a learner that they will be taking

part in a videoconference, they will expect to appear on a TV show, not to peer at a tiny jerky image on a computer screen.

Producing the good quality video requires some technical expertise. The main factors are:

- *camera quality*: Use a high resolution digital camera if possible.
- *shooting the video*: Use a tripod, keep the camera still, and take close-ups rather than long shots.
- *converting from camera to computer*: Use direct digital input (known as “firewire”).
- *compressing the video and the audio*: If the learning requires high quality video, for example to show social interactions, then use MPEG-2 compression, 640x480 size, 25 fps, with 44KHz, 16 bit sound. If its purpose is to introduce the speaker or give a sense of an event (e.g. a newsreel to introduce history teaching), and the video is stored on a CDROM, then MPEG-1 compression, 352x288 size, 44KHz, 16bit sound should be adequate. For web video, then MPEG-1, 160x120, 10fps, is standard, but try to keep at least 22KHz, 8bit sound, and don't use the video for providing the core teaching.
- *playing it on different computers*: You need to make sure that your video will play on different computers. The three most popular video players for personal computers are QuickTime, Real Player and Windows Media Player. Although each has its own proprietary format, all three can play back MPEG files.

Other methods can provide some of the benefits of video, without the problems. For remote collaboration, good quality audio is most important. You could show still images of the speakers to introduce them, and then use a shared drawing package or jointly operated word processor to carry out the work. For teaching, a well presented animation or simulation can in some cases be more valuable than video because it hides the detail and focuses on the concept being taught.

Devices

The hardware on which the system is to be delivered will influence the design and usability of the system. Input devices are mice, keyboards, touchscreens, pen tablets, and so on. Output devices are screens, printers, projectors and suchlike. Most systems are developed for conventional computers: mouse, keyboard, screen and perhaps a printer. Some systems require specific devices: for example, a public information point should use a touchscreen. Each of these devices has different characteristics, and these will affect the guidelines and system designs. There is not space to list the characteristics of each device and an associated set of guidelines. Instead, we have provided a brief summary of the main devices in Table 3. The key feature is to consider how people will interact with the system and define the devices accordingly.

Device	Pros	Cons
PC	general purpose	expensive, not portable
laptop	general purpose	expensive, limited battery life
palmtop	portable, multipurpose	small screen, lack of processing power
mouse	pointing, clicking	drawing, painting, needs space
trackball	pointing, clicking, space efficient	drawing, painting, less familiar
tablet and pen	freehand drawing, point & click	unfamiliar, pen gets lost
touchscreen	compact and robust, simple	no fine detail, limited text entry
keyboard	text, number entry	bulky
screen	general purpose	size, power requirements
projector	large display	cost, fragility
printer	permanent record	lack of interactivity

Table 3. Typical devices and their pros and cons

Accessibility

With accessibility we return to the start of these guidelines, and the need to understand the learners and their needs. Designing for learners with special needs is one aspect of suiting the software to the user. While it is not possible to anticipate every learning need and difficulty, you can ensure that the material is accessible to people with the most common disabilities. The Web Accessibility Initiative was set up by the World Wide Web Consortium (the body that oversees the development of the Web) to promote access for people with disabilities. It provides a useful “Quick Tips” guide to improving accessibility at <http://www.w3.org/WAI/References/QuickTips/>. Here are some basic guidelines based on the quick tips. Most of them are general good design.

- *Colour*. Make sure that your material can be read in shades of grey, to assist people with colour blindness.
- *Images*. Use the **alt** tag on web images or graphics, to describe the function of the image (e.g. “next page button”).
- *Text*. Use relative font sizes rather than fixed ones (e.g. when using style sheets don’t set **font size** to a fixed point or pixel size, but to a percentage of the default font size).

- *Multimedia*. Provide captions and transcripts of audio presentations, and descriptions of video.
- *Hypertext links*. Use text that makes sense when read out of context. For example, avoid "click here."
- *Page organization*. Use headings, lists, and consistent structure.
- *Graphs & charts*. Summarize them, or use the **longdesc** attribute.
- *Scripts, applets, and plug-ins*. Provide alternative content in case these active features are inaccessible or unsupported.
- *Frames*. Use the **noframes** element and meaningful titles.
- *Tables*. Make line-by-line reading sensible. Summarize the content.
- *Check your work*. Validate. Use tools, checklist, and guidelines available at <http://www.w3.org/TR/WCAG>

Conclusions

The guidelines given here are the start, not the end, of good design of educational software. Here are some useful sources of further information that we drew on when preparing this guide:

R. Phillips, *The Developer's Handbook to Interactive Multimedia: a Practical Guide for Educational Applications*. London: Kogan Page, 1997.

A practical handbook for developers of educational multimedia. Somewhat dated, and aimed at higher education, but provides a mix of educational considerations and design guidelines.

P. J. Lynch and S. Horton, *Web Style Guide: Basic Design Principles for Creating Web Sites*. New Haven and London: Yale University Press, 1999.

A well-informed guide to good design of websites. It is available online at <http://info.med.yale.edu/caim/manual/>

J. Nielsen, *Designing Web Usability*. New Riders Publishing, 2000.

A valuable, though opinionated, guide to designing usable websites.

A. Dix, J. Finlay, G. Abowd, and R. Beale, *Human-Computer Interaction*, 2nd ed: Prentice Hall, 1998.

A standard textbook on human-computer interaction. Covers interface design and much more besides.

References

- [1] I. Sommerville, *Software Engineering*, 6th ed: Addison Wesley, 2000.
- [2] R. Pressman and D. I. (Editor), *Software Engineering: a Practitioner's Approach*, 5th ed: Schaum, 2000.
- [3] B. Wasson, "Instructional Planning and Contemporary Theories of Learning: Is this a Self-Contradiction?," in *Proceedings of the European Conference on Artificial Intelligence in Education*, P. Brna, A. Paiva, and J. Self, Eds. Lisbon: Colibri, 1996, pp. 23-30.
- [4] R. Phillips, *The Developer's Handbook to Interactive Multimedia: a Practical Guide for Educational Applications*. London: Kogan Page, 1997.
- [5] D. Laurillard, *Rethinking University Teaching: a Framework for the Effective Use of Educational Technology*. Routledge, 1993.
- [6] Microsoft Corporation, *Microsoft Windows User Experience*: Microsoft Press, 1999.
- [7] D. J. Mayhew, *Principles and Guidelines in Software User Interface Design*: Prentice Hall, 1992.
- [8] T. W. Malone, "What makes computer games fun?," in *Byte*, 1981, pp. 258-277.
- [9] M. Scaife and Y. Rogers, "External cognition: how do graphical representations work?," *International Journal of Human-Computer Studies*, vol. 45, pp. 185-213, 1996.
- [10] R. Cox, "Representation construction, externalised cognition and individual differences," *Learning and Instruction*, vol. 9, pp. 343-363, 1999.
- [11] K. O'Hara and A. Sellen, "A comparison of reading paper and on-line documents," in *Proceedings of CHI'97 Conference*. Atlanta, GA: ACM Press, 1997, pp. 335-342.
- [12] J. Hartley, *Designing Instructional Text*, 3rd ed: Kogan Page, 1994.
- [13] J. Nielsen, *Designing Web Usability*. Indianapolis, Indiana: New Riders, 2000.
- [14] M. Scaife, Y. Rogers, F. Aldrich, and M. Davies, "Designing for or Designing With? Informant Design for Interactive Learning Environments.," in *Proceedings of CHI 97, Human Factors in Computing Systems.*, New York: Addison Wesley, 1997, pp. 343-350.
- [15] J. J. Cadiz, A. Balachandran, E. Sanocki, A. Gupta, J. Grudin, and G. Jancke, "Distance learning through distributed collaborative video viewing," in *Proceedings of the ACM 2000 Conference on Computer supported cooperative work*: ACM, 2000, pp. 135-144.