Chapter 14

User Interfaces

In computer science, we understand the *user interface* as the interactive input and output of a computer as it is perceived and operated on by users. *Multimedia user interfaces* are computer interfaces that communicate with users using multiple media, sometimes using multiple modes such as written text together with spoken language [May93a].

Multimedia would be without any value if applications did not use the various media at the user interface for input and output. Media determine not only *how* human-computer interaction occurs, but also *how well.* For example, to boot the first computers, the user had to enter a range of addresses through register switches and commands at a mainframe console. Punch cards were used for input and paper was the main form of output. *Text* was the only medium for interaction with terminals. Later, applications were controlled through text menus, which simplified user input. Nevertheless, the user had to adjust to the computer.

Graphical user interfaces – using the mouse as the main input device – have greatly simplified human-machine interaction. A large number of graphical commands are hidden to the users through the use of a *Window System* (e.g., *Presentation Manager*^{TM}, *GEM, NeWS, MS-Windows* or the *X Window System*^{TM}). There are also other software programs which achieve similar user interfaces as an X Window System. The computer has been adapted – at least in part – to the user.

Despite these advances, there are still many well known problems with current user interfaces. One problem is *computer interaction* which is still neither natural nor effective. Speaking is often more suitable for the situation than writing. Changes and commentaries can be made verbally, which is more effective (i.e., faster) than making changes and comments in electronic text. Reading and listening are not alternatives to each other; they complement one another (e.g., audio textbooks).

Another problem is the *specification of object movement.* A specification of movements using graphics or text is often much more difficult and complicated than using a motion video. For example, consider an electronic textbook about tennis. The individual movements and typical errors can be presented much more easily using motion video than graphics images, or even text alone.

The development goes toward more effective human-computer interfaces using new interactive devices, which is an area of research in the field of *virtual reality.* The goal is to provide interactive devices such as *data gloves and body suits* for input, and *holography, head-mounted displays and three-dimensional sound device* for output. These devices help to move objects in a 3D space.

14.1 General Design Issues

The main emphasis in the design of multimedia user interfaces is *multimedia presentation.* There are several issues which must be considered:

1. To determine the appropriate information content to be communicated.

2. To represent the essential characteristics of the information.

3. To represent the communicative intent.

4. To chose the proper media for information presentation.

5. To coordinate different media and assembling techniques within a presentation.

6. To provide interactive exploration of the information presented.

The objective of the multimedia presentation system should be the *appropriateness principle* [Nor91]: "The surface representation used by the artifact should allow the person to work with exactly the information acceptable to the task: neither more nor less."

14.1.1 Architectural Issues

An effective presentation design process should not only involve sequential flow of actions, but also parallel and interactive actions [SF91]. This means that there is a requirement for extensive feedback going on between the components making decisions about media and modalities. Additionally, the design includes a number of higher-level concerns, such as goals and focus of the dialogue, the user's context and current task, and media selection to represent this information in a way that corresponds to these concerns. A conceptual architecture with a knowledge base (lower part of the figure), used by an intelligent multimedia presentation system (upper part of the figure – both parts are separated by the black arrow), is shown in Figure 14.1 [RH93].

14.1.2 Information Characteristics for Presentation

A complete set of information characteristics makes knowledge definition and representation easier because it allows for appropriate mapping between information and presentation techniques. The information characteristics specify:

- *Types*

  Characterization schemes are based on *ordering information.* There are two types of ordered data: (1) *coordinates versus amount*, which signify points in time, space or other domains; or (2) *intervals versus ratio*, which suggests the types of comparisons meaningful among elements of coordinate and amount data types.

- *Relational Structures*

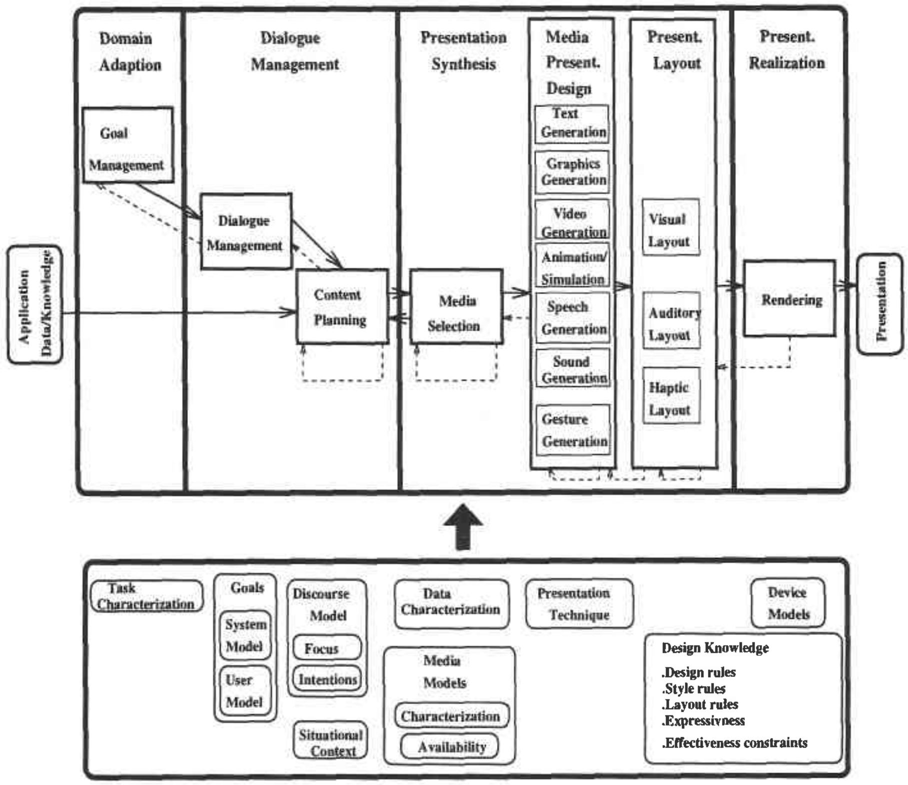


Figure 14.1: *Conceptual architecture of a multimedia presentation system [RH93].*

This group of characteristics refers to the way in which a relation maps among its domain sets (dependency). There are *functional dependencies* and *non-functional dependencies.* An example of a relational structure which expresses functional dependency is a *bar chart.* An example of a relational structure which expresses non-functional dependency is a student entry in a relational database.

- *Multi-domain Relations*

  Relations can be considered across multiple domains, such as: (1) *multiple attributes* of a single object set (e.g., positions, colors, shapes, and/or sizes of a set of objects in a chart); (2) *multiple object sets* (e.g., a cluster of text and graphical symbols on a map); and, (3) *multiple displays.*

- *Large Data Sets*

  Large data sets refer to numerous attributes of collections of heterogeneous objects (e.g., presentations of semantic networks, databases with numerous object types and attributes of technical documents for large systems, etc.).

14.1.3 Presentation Function

Presentation function is a program which displays an object (e.g., *printf* for display of a character). It is important to specify the *presentation function* independent from presentation form, style or the information it conveys. Several approaches consider the presentation function from different points of view. For example, one approach views the presentation function as a set of information-seeking goals [RM91], another approach considers it as a hierarchical representation of media-independent presentation goals derived from a plan-based theory of communication [May93b].

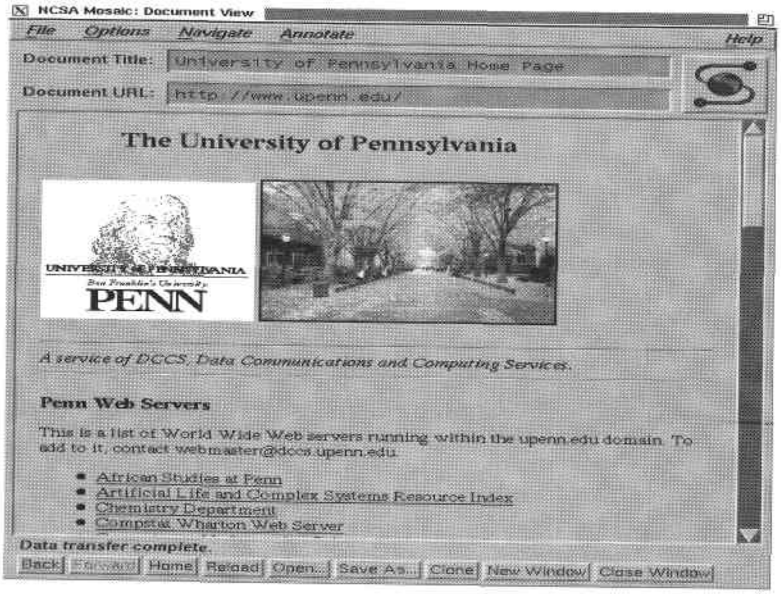
14.1.4 Presentation Design Knowledge

To design a presentation, issues like *content selection*, *media and presentation technique selection* and *presentation coordination* must be considered.

Content selection is the key to convey the information to the user. However, we are not free in the selection of the it because content can be influenced by constraints imposed by the size and complexity of the presentation, the quantity of information, limitations of the display hardware, and the need for presentation completeness and coherence.

Media selection determines partly the information characteristics described earlier. For *selecting presentation techniques*, rules can be used. For example, rules for selection methods, i.e., for supporting a user's ability to locate one of the facts in a presentation, may specify a preference for graphical techniques. Media must be chosen to be "adequate". For example, to present a course on how to play tennis, graphics and video are more suitable than text only. On the other hand, it may not be of great help to receive all electronic mail as audio data only because the receiver has very few opportunities to scan over the content; he/she must listen to most of the received information.

Coordination can be viewed as a process of composition. Coordination needs mechanisms such as : (1) encoding techniques (e.g., among graphical attributes, sentence forms, audio attributes, or between media); (2) presentation objects that represent facts (e.g., coordination of the spatial and temporal arrangement of points in a chart); and, (3) multiple displays (e.g., windows). Coordination of multimedia employs a set of composition operators for merging, aligning and synthesizing different objects to construct displays that convey multiple attributes of one or more data sets. For example, the user interface shown in Figure 14.2 results from the composition of objects with attributes such as color, position, size and medium specification (text, graphics, image) (user interface taken from the NCSA Mosaic tool).



Document Title: University of Pennsylvania Home Page

Document URL: http://www.upenn.edu/

The University of Pennsylvania

UNIVERSITY of PENNSYLVANIA
Ben Franklin's University
PENN

A service of DCCS, Data Communications and Computing Services.

Penn Web Servers

This is a list of World Wide Web servers running within the upenn.edu domain. To add to it, contact webmaster@dccs.upenn.edu

- African Studies at Penn
- Artificial Life and Complex Systems Resource Index
- Chemistry Department
- Compstat Wharton Web Server

Data transfer complete.

Figure 14.2: *User interface of University of Pennsylvania's Mosaic home page illustrating a result of coordination.*

14.1.5 Effective Human-Computer Interaction

One of the most important issues regarding multimedia interfaces is effective human-computer interaction of the interface, i.e., *user-friendliness.* We will discuss this property in detail in Section 14.6.

Here, we will just briefly enumerate the main issues the user interface designer should keep in mind: (1) *context*; (2) *linkage to the world* beyond the presentation display; (3) *evaluation of the interface* with respect to other human-computer interfaces; (4) *interactive capabilities*; and (5), *separability* of the user interface from the application.

14.2 Current Work

Although the topic of generating a user interface with several media is crucial for the success of multimedia systems and applications, there is still a lack of serious attention given to this issue. The current literature tends to be either very abstract or problem-specific.

A primary source of information is the *SIGCHI (ACM Special Interest Group for Computer-Human Interaction)* and the annual *Human Factors Society Conference.* A good overview for generating graphical interfaces is given in [FWC84], discussing matters such as classification of the interactions and classification of the actions performed. A joint work in the area of intelligent multimedia interfaces is presented in [May93a]. Issues discussed include presentation design, the communicative act of multi-sentential text in multimedia presentations and a presentation planner that composes different media together at the user interface.

Some authors worked on partial problems in user interfaces such as color mapping [Mur84], the size and position of windows among each other [Gai85, Gai86] and the use of icons in alternative window systems [Mye84]. Other authors consider input interface problems to provide the ability to interpret typed or spoken natural language utterances together with deictic mouse or data-glove gestures to resolve ambiguous references (e.g., "put that there") [NTDS89]. The most work to integrate video and audio at the input interface is done for hypertext/hypermedia applications (for early work see [Fri87, Hyp88, MDR90, Sch87]). Further, relevant work on user interfaces exists at the MIT Media Lab [Bra87].

For output media, the majority of research concentrates on automated generation of single output media. Progress has been made in *graphical design* (e.g., the design of tables and charts [Mac86], network diagrams [Mar91b, Mar91c], business graphics displays [RMM91], three-dimensional explanatory graphics [Fei85]), *linguistic realization* and *text planning.* Temporal media (e.g., animation and speech) and temporal information associated with events (e.g., points in time, duration, ordering relations) are done in the COMET project, using Allen's temporal logic to construct temporal plans [FM93].

Several projects provide problem-specific solutions to design of multimedia presentation systems, such as:

- *WIP* is a multimodal presentation system which presents and understands combinations of graphics, text and pointing gestures, and generates illustrated instructions for technical devices (e.g., it can generate visual instructions on how to operate an espresso machine) [AFG$^+$93]. It can be part of a system which automatically generates multimodal and illustrated documents.
- *COMET (Columbia Operations and Maintenance Explanation Testbed)* automatically designs integrated textual and three dimensional graphical presentations to explain the operation and maintenance of an army field radio [FM93].
- *TEXTPLAN (Textual EXplanation PLanner)* provides narrated, animated directions over an object-oriented map using a collection of multimedia actions (e.g., speech acts, graphical acts) for media integration and control [May93b].
- *AIMI (An Intelligent Multimedia Interface)* has a goal to develop a portable intelligent multimedia/multimodal interface [BM93]. *Portable* means that much of the system processing is independent of any particular back-end system. With this functionality, AIMI can be connected to a new back-end with minimum effort. AIMI reasons in detail about the meaning of user input, using a number of AI (Artificial Intelligence) approaches. Similarly, when presenting information to the user, the system is able to intelligently choose among the

presentation alternatives available to it and then design the most appropriate presentation. Currently, AIMI includes natural language, maps, mouse gestures, business charts, specialized interactive inspectors, still images and non-speech audio.

14.3 Extension through Video and Audio

Continuous stream audio and video play a significant role in multimedia. The main issue during the presentation of continuous media streams is the continuity in *time*. Hence, time is a new presentation dimension in a user interface. Figure 14.3 shows multimedia at the user interface taking time into account.

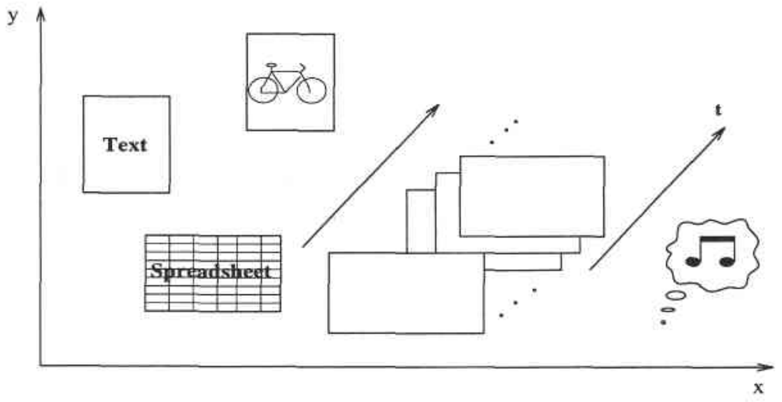


Figure 14.3: *Multimedia at the user interface with the presentation dimension "time".*

An illusion of a *continuity* by the user is created through the presentation of a sequence of static elements In the case of audio, the continuity of data is achieved through reconstruction of an analog signal.

14.4 Video at the User Interface

A continuous sequence of, at least, 15 individual images per second gives a rough perception of a continuous motion picture. At the user interface, video is implemented through a continuous sequence of individual images. Hence, video can be manipulated at this interface similar to manipulation of individual still images. An example of a user interface for manipulating images is the software package *xv*, developed by John Bradley of the GRASP Laboratory at the University of Pennsylvania (shown in Figure 14.4).

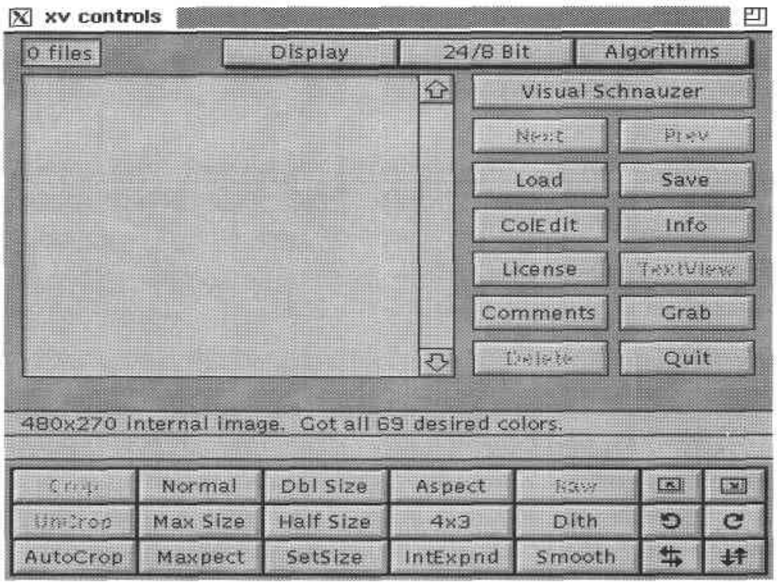


Figure 14.4: *xv user interface.*

When an individual image consisting of pixels (no graphics, consisting of defined objects) can be presented and modified, this should also be possible for video (e.g., to create special effects in a movie). However, the functionalities for video are not as simple to deliver because the high data transfer rate necessary is not guaranteed by most of the hardware in current graphics systems. Also, window systems (software) as the main graphical user interface are a limiting factor. Usually, there can be one to at most four video windows with a quality of 25-30 images per second presented simultaneously because the service of a video window obeys the same rules as any other window.

14.4.1 Hardware for Visualization of Motion Pictures

Special hardware for visualization of motion pictures is available today, mostly through additional video cards. Early examples of such additional hardware are IBM-M-Motion and ActionMedia II (Intel/IBM) cards, and the Parallax, Sun and RasterOps cards. Today, these cards have become an integral part of the multimedia system.

Most motion video components integrated in a window system use the chroma-key methods where an application generates a video window with a certain color. Traditionally, this color is a certain blue (coming from a video technique used in the TV). The window system handles, in general, the video window as a monochrome pixel graphic window, but on the device level, there is a switch which allows for the selection of the display between the standard graphics and motion video. This switch usually brings the standard graphics to the screen. If the hardware switch detects motion video, such a video window presents the video signal taken directly from a camera. Using a communication-capable multimedia system, this camera can be controlled remotely. The video data may be transmitted from the camera into a computer network and then displayed.

14.4.2 Example: Remote Camera Control Application

Remote camera control is used, for example, in surveillance applications. Another example is a microscope, remotly controlled in a telesurgery environment. We discuss below an application in which an engineer remotely controls a CIM-completion process with the help of a remote-control video camera [WSS94].

Application Specification

The camera is connected to a computer which serves as a *camera server* through a standardized analog interface. The camera control occurs, for example, through a serial RS-232-C interface. The camera-server sends commands such as *focus*, *zoom* and *position* to the camera through this serial interface. The actual control of the camera is initiated by the camera-client, which can be located remotely. In addition to the data path for camera control, there is also a video data path, i.e., the video data are digitized, compressed and sent by the camera-server to the camera-client where the engineer is located. The video image taken from the camera is displayed.

User Interface

The simplest decision would have been, in this case, to use the *keyboard.* Fixed control functions could be assigned to individual keys. For example, the keys *left, right, up* and *down* would move the camera in the corresponding directions.

In a window system, individual *buttons* can be programmed to position a camera. In addition to the directions *left, right, up*, and down, other directions, such as *left up* and *left down*, can be specified through other buttons. Pushing the button initiates the positioning process. The particular movement is stopped explicitly with the *stop* button. Another possibility to position a camera is by the pushing and releasing of a button, i.e., continuous movement of the camera follows through several consecutive "push" and "release" button actions.

Instead of using *buttons* in a window system, positioning in different axes can also be done through *scrollbars.* To activate the different graphical elements, a *touch-screen* can be used instead of mouse. A finger or pen can activate the user interface elements. A more natural, simpler, form of camera control could be implemented with a *joystick.* The movement of the joystick is translated into a camera position. Here, additional hardware for support of the joystick is necessary. Another possibility is a *data glove*, and graphical simulation of the camera views and the production line. The movement of the data glove is translated into the camera position in the graphical simulation; if the camera position in the simulated (virtual) environment is correct, it is sent to the camera-server, which moves camera in the real environment.

Note that *graphical simulations* as user interfaces for controlling teleoperation are preferred. The user sees movements in the *virtual environment*. If the movement is correct, the position and possibly other information are transmitted to the real remote site. The input devices to the graphical simulations are mice or data gloves.

Direct Manipulation of the Video Window

In our setup we decided to use a very user-friendly variant known as direct manipulation of the video window [SS94]. There are two possibilities:

1. *Absolute Positioning*

   Imagine a tree in the upper right corner of the video window. The user positions the cursor on this object and *double-clicks* with the mouse. Now, the camera will be positioned so that the tree is the center of the video window, i.e., the camera moves in the direction of the upper right corner. This method of object pointing and activating a movement of camera is called *absolute positioning*. The camera control algorithm must derive the position command from: (1) the relative position of the pointer during the object activation in the video window; and, (2) the specified focal distance.

2. *Relative Positioning*

   Imagine the pointer to the right of the center of the video window. By pushing the mouse button, the camera moves to the right. The relative position of the pointer with respect to the center of the video window determines the direction of the camera movement. When the mouse button is released, the camera movement stops. This kind of direct manipulation in the video window is called *relative positioning*. A camera can move at different speeds. A speed can be specified through the user interface as follows:

   - If the mouse has several buttons, different speeds can be assigned to each button. For example, the left mouse button could responsible for slow, accurate motion (e.g., for calibration of the camera). The right buttons could be for fast movement of the camera.

- Instead of working with several mouse buttons, the distance of the pointer to the window center could determine the speed; the larger the distance, the faster the movement of the camera.

From the viewpoint of user-friendliness, the best method of camera control is direct manipulation of the video window. More specific, the relative positioning with speed specification, using the distance of the pointer to the window center, is the preferred solution.

14.5 Audio at the User Interface

Audio can be implemented at the user interface for application control. Thus, speech analysis is necessary.

Speech analysis is either speaker-dependent or speaker-independent. *Speaker-dependent* solutions allow the input of approximately 25,000 different words with a relatively low error rate. Here, an intensive learning phase to train the speech analysis system for speaker-specific characteristics is necessary prior to the speech analysis phase. A *speaker-independent* system can recognize only a limited set of words and no training phase is needed.

During audio output, the additional presentation dimension of *space* can be introduced using two or more separate channels to give a more natural distribution of sound. The best-known example of this technique is *stereo.* Further developments include the mono-subwoofer channel and quadraphony. In the case of stereo, spatial positions are assigned to audio sources.

For example, during a conference with four participants, a fixed place is assigned to each participant. The motion video of participant L is displayed in the upper left corner of the screen. The corresponding sound of this participant is transmitted only through the left speaker. Participant M is visually and acoustically located in the middle. Participant R is positioned to the right. In this example, the conference system always activates the video window with the loudest-speaking participant. The recognition of the loudest acoustic signal can be measured over a duration of five seconds. Therefore, short, unwanted and loud signals can be compensated for. In the active video window, the video is displayed with 25 frames per second. Motion video is displayed only in one video window at one time because of technical reasons. In the video windows of the other (quieter) participants, the last image of the motion video sequence, when they were active, is displayed.

In the case of *monophony*, all audio sources have the same spatial location. A listener can only properly understand the loudest audio signal. The same effect can be simulated by closing one ear. Stereophony allows listeners with bilateral hearing capabilities to hear lower intensity sounds. It is important to mention that the main advantage of bilateral hearing is not the spatial localization of audio sources, but the extraction of less intensive signals in a loud environment.

In a window system, it is possible for a user to position a window on the screen. In [LPC90], this paradigm was applied to medium audio as follows. The sound application opens different audio sources using location identification (e.g., source to the right or left). To each audio source there is assigned an *audio window* which appears on the screen at the location specified (e.g., right or left). The location of the source can be changed by moving the audio window. The number of audio windows is limited. First experiences have shown that a maximum of two audio windows can be opened at the same time. As an extension of this approach, an audio window can offer the user control over the relative volume and depth.

The concept of the audio window allows for application independent control of audio parameters, including spatial positioning. Most current multimedia applications using *audio* determine the spatial positioning themselves and do not allow the user to change it. An example of such an application is the audio tool for SUN workstations. Figure 14.5 shows the user interface of this audio tool.

14.6 User-friendliness as the Primary Goal

User-friendliness is the main property of a good user interface. As an example, compare a multimedia-integrated telephone service with an ISDN telephone service. Today's telephones consist of a large number of touch keys, each sometimes representing three different functions. It is not an easy task to operate such a telephone.

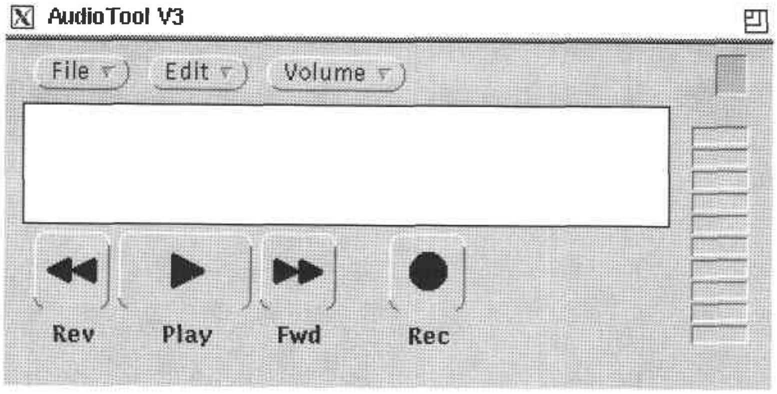


Figure 14.5: *Audio tool user interface.*

Indeed, given sporadic user the user may forget many of these functions. However, with a multimedia-integrated telephone, some of the disadvantages of the current ISDN telephone user interface can be eliminated through use of multimedia data, e.g., through display of some multimedia information on the screen [SG88]. The goal is implementation of a user-friendly human-computer interface.

What this user-friendliness means and *how* this property is achieved are not always clear. The design of a user-friendly graphical interface requires the consideration of many conditions. The addition of audio and video to the user interface does not simplify this process. In this section, a number of generally applicable criteria for multimedia user interfaces is presented.

We will restrict the discussion of user-friendliness to multimedia systems in the office or home. The reason is that different applications have different user-friendliness requirements. For example, in the case of a car phone, speech recognition is one important requirement of user-friendliness. By speaking a person's name, that person is called. Loud noise caused by a car, together with reflected sound waves, put higher requirements on this technique.

14.6.1 Easy to Learn Instructions

Application instructions must be easy-to-learn. The older dial phones required no time to learn. A simple touch phone, with a few additional buttons, requires at most 10 minutes to master. An ISDN telephone, unfortunately, often requires more than 20 minutes. A telephone application should require at most 10 minutes to understand its main functions. This is achieved through the homogeneous interfaces of different applications provided by a window system. A multimedia application must support similar mechanisms which are known to the user from other applications.

14.6.2 Context-sensitive Help Functions

A context-sensitive help function using hypermedia techniques is very helpful, i.e., according to the state of the application, different help-texts are displayed. For example, after selecting the *call re-routing* function, the help function provides a brief explanation of call re-routing.

14.6.3 Easy to Remember Instructions

A user-friendly interface must also have the property that the user easily remembers the application instruction rules. Easily remembered instructions might be supported by the intuitive association to what the user already knows. For example, the user knows the phone book (register). Hence, the user interface of a telephone service, implemented in a multimedia system, can show the participant a list on the screen. The user can simultaneously select and call another user through a double-click of the button. With a simple click in a window system, generally, an element is highlighted. The data of the selected callee can be displayed and, if necessary, changed, or additional data can be appended. The designer of such a user interface must put him/herself in the user's situation. Thus, different user-classes need to be considered. In addition to both sporadic and everyday users, there are also users in office areas and homes.

14.6.4 Effective Instructions

The user interface should enable effective use of the application. This means:

- Logically connected functions should be presented together and similarly. For example, *call re-routing* and *call forwarding* are two such functions of a telephone service; both functions require the input of a phone number.

- Graphical symbols or short video clips are more effective than textual input and output. They trigger faster recognition. For example, the notions trash and short cuts can be replaced by symbols; people can be identified by their pictures, as well as by their names.

- Different media should be able to be exchanged among different applications. For example, the same address book is used for facsimile, telex, teletex and mailing applications. A simple clipboard function is generally insufficient. Hypertext and hypermedia provide good links for such an approach.

- Actions should be activated quickly. For example, in a telephone service the selection of a callee must be very fast. It is possible with the input of the callee's last name or his/her telephone number. This is an alternative to opening the electronic telephone book (list), scrolling the list and then double-clicking.

- A configuration of a user interface should be usable by both professional and sporadic users.

14.6.5 Aesthetics

With respect to aesthetics, the color combination, character sets, resolution and form of the window need to be considered. They determine a user's first and lasting impressions.

It is desirable to develop only one application for different users and languages. This is achieved by separating (either in the window system or application) the text, graphics and actual program.

14.6.6 Effective Implementation Support

To achieve effective implementation of a user-friendly human-computer graphical interface, the user's requirements must be considered. This influences the cost of the implementation.

An effective implementation of a user-friendly interface can be influenced by the following:

- If the user's requirements are missing, *Rapid Prototyping* should be used. This means that the user-interface is developed, changed and tested without filling in the contents of the actual programs. Object-oriented programming tools are good environments for this approach.

- If window systems are used, the user interface becomes hardware-independent and the development effort is shorter. Also maintenance-friendliness increases if program generators are available, or uniformly accessible programming environments are used.

14.6.7 Entry Elements

User interfaces use different ways to specify entries for the user:

- *Entries in a menu*

  In current menus there are visible and non-visible entry elements Entries which are relevant to the particular task are visible. For example, the xv (Figure 14.4) entry *TextView* is not initially displayed because xv is an X-window image/video grabber. However, if the user clicks on *Load*, the current directory files are displayed in the working window. If a text file is then chosen from the directory, *TextView* becomes visible.

- *Entries on a graphical interface*

  - If the interface includes text, the entries can be marked through color and/or a different font (e.g., NCSA Mosaic's blue color and underlined text). Figure 14.2 shows the NCSA Mosaic user interface, displaying a hypertext document, where the entries are underlined. The user clicks the underlined text and a more detailed description appears in the NCSA Mosaic working window.

- If the interface includes images, the entries can be written over the image. An example is an intelligent training system for tutoring technicians to repair computers (Visual Repair [Goo93]). The system presents images of different computer parts, and by clicking on text written over the parts, the help tool accesses information about the parts and the particular operations that can be applied to them.
- If the interface includes images, the functions can be activated through direct positioning of the cursor on the image object. This is done, for example, with city maps or building images. The user clicks on a certain object (street or room) and a new image, audio and/or text describes the object.

14.6.8 Meaningful Location of Functions

Individual functions must be placed together in a meaningful fashion. This occurs through (1) alphabetic ordering or (2) logical grouping.

For example, entries in the telephone book are sorted alphabetically, whereas the *call re-routing* and *call forwarding* functions are logically connected, similarly processed and displayed. In the case of a telephone application, using pull-down menus for certain functions may be critical. If a graphical interface is available, alphabetic order can be used, but the general rule is to provide a balanced interface which is aesthetically satisfying [Gai86].

14.6.9 Presentation

The presentation, i.e., the optical image at the user interface, can have the following variants:

- Full text

- Abbreviated text
- Icons, i.e., graphics
- Micons, i.e., motion video

Each possibility has its advantages and disadvantages, often depending on the individual user. For example, the full text (e.g., *Call Waiting*) can be easily understood but not clearly displayed. An abbreviation (e.g., CW for *Call Waiting*) may be obvious to the frequent user, but not to a sporadic user. Icons must have a uniform semantics inside of the window system (e.g., a trash container is associated with a delete function). In the case of micons, the reduced content of the motion picture must be recognizable.

14.6.10 Dialogue Boxes

Different dialogue boxes should have a similar construction. This requirement applies to the design of: (1) the buttons *OK* and *Abort*; (2) joined windows; and, (3) other applications in the same window system [Ber88]. Semantically similar entry functions can be located in one dialogue box instead of several dialogue boxes.

It is important to decide how many dialogue boxes should be opened at the same time, how the entry should be visible and how a new requirement can be expressed through an additional box. For example, the *Abort* button in a window should never be positioned in the same location as the *Save* button of the underlying window.

14.6.11 Additional Design Criteria

Some additional useful hints for designing a user-friendly interface should be mentioned:

- The form of the *cursor* can change to visualize the current state of the system. For example, a rotating fish instead of a static pointer shows that a task is in progress.

- If *time intensive tasks* are performed, the progress of the task should be presented. For example, during the formatting of a disk, the amount formatted is displayed through a filling bar; during the remote retrieval of a file, the number of transmitted bytes in relation to the whole size of the file is presented. (In NCSA Mosaic, where large image and sound files are transmitted from the servers to the client, this is an useful display.) This display allows the user to evaluate the state of the task and react to it, i.e., let the task continue or cancel it. Thus, the *Abort* function to cancel the activity should always be present during a time-intensive task.

- A *selected entry* should be immediately highlighted as "work in progress" before performance actually starts. This approach ensures that no further input is given to the entry.

14.6.12 Design-specific Criteria

In addition to the above described general criteria for the design of a user interface, the problem specific properties of the actual task need to be considered. These properties are demonstrated in our telephone service example.

The telephone network and telephone-specific end-devices are provided by the telephone companies. They specify the user interface characteristics:

1. The end-device must have the *basic function* of dialing a number. The requirement may be that the dialing is performed using keys and that there is an alphanumeric, single-line display. This requirement provides *compatibility* among different phone devices.

   In a multimedia system, dialing with keys can be programmed, but it is not very meaningful; the main advantages of the different media are unused. To provide compatibility, a key set with corresponding user procedures should be emulated.

2. *Ongoing tasks* should be signaled. For example, if the call re-routing function is activated, this function should be signaled optically on the device. In the case of a telephone (computer) application, its state does not have to be

displayed on the whole screen. A *telephone icon* can be used if no window is opened, but the application is still active.

3. A telephone device must always be *operational.* This requirement influences the corresponding hardware and software.

   If a telephone service is implemented on PCs as a multimedia application, these devices are not always meant to be operational for 24 hours. It also cannot easily become operational when a call arrives. The simplest solution is external storage of the telephone hardware which is operational without the PC being turned on. In the context of communication-capable workstations, a solution that activates the computer through external events (e.g., arriving call) is best. The hard disk can then also serve as the voice storage. When the PC boots, the telephone application must be loaded, i.e., become operational.

4. The *state* of the telephone-device (i.e., telephone application) must be always visible.

   While working with the telephone application, it gets into different *states.* In different states different functions are performed. Some states imply that a function can be selected, some states imply that a function cannot be selected.

   The nonselective functions can be:

   - *Nonexistent*: The function disappears when no activation is possible.
   - *Displayed*: The function is displayed, but marked as deactivated and any interaction is ignored; for example, deactivated menu functions are displayed in gray, the active functions are displayed in black.
   - *Overlapped*: If a function is overlapped with another window which is designed as the *confirmer*, this function cannot be selected. First, after closing the confirmer, other input can be taken.

   It is important to point out that the functions most often used are always visible in the form of a *control panel.* It is necessary to pick carefully which functions will belong in the control panel.

5. When a call request arrives, it must be immediately signaled (e.g., ringing).

For a telephone application on a workstation, the window system provides a preemptive scheduling algorithm. Each process of the window system must be able to interrupt after a certain time span, independent of the task currently being performed. Therefore, arriving calls can be signaled promptly. An intermediate solution can be achieved if the arrival of the call is signaled acoustically by the telephone hardware or at the low system interrupt level. After arrival of the call is signaled, the user can interrupt his/her tasks and activate the telephone application.

Design of a user interface is also influenced by a specific implementation environment. For example, in addition to the primitives of the particular window system, the quality of the graphical terminal with its resolution, size and color-map is important.

14.7 Comments

We have discussed in this chapter several issues regarding multimedia user interfaces. The main emphasis has been on video and audio media because they represent *live information.* At the user interface, these media become important because they help users learn by enabling them to choose how to distribute research responsibilities among applications (e.g., on-line encyclopedias, tutors, simulations), to compose and integrate results and to share learned material with colleagues (e.g., video conferencing). Additionally, computer applications can effectively do less reasoning about selection of a multimedia element (e.g., text, graphics, animation or sound) since alternative media can be selected by the user.

Another important issue, that of user-friendly interfaces, was also discussed. This property is crucial, especially with multimedia. Different media provide many possibilities and opportunities to express functionalities at the human-computer interface, but if not implemented correctly, can also lead to confusion and problems.