DI 4 — Software Architecture

A Report of the Esprit Project 8579 Miami – WP 3 –

March, 1996

Written by
K. Hartung[5], S. Münch[6], L. Schomaker[3],
T. Guiard-Marigny[2], B. Le Goff[2],
R. MacLaverty[4], J. Nijtmans[3], A. Camurri[1] I. Defée[4], C. Benoît[2]
([1]DIST, [2]ICP, [3]NICI, [4]RIIT, [5]RUB, [6]UKA)

Introduction

Deliverable 4 in MIAMI is concerned with the *"Completed Software Architecture"*. As such, it was envisaged to describe this software in a technical way, much as has been done in Deliverable 1. However, what we have done to report on this software is to show a number of applications in which the integration of modalities becomes apparent.

In chapter 1, an experimental interface is presented which deals with the challenge of information user interfaces in the consumer electronics environment. The demands with respect to usability are much higher than in computer user interfaces since the end user cannot be expected to be technically interested. The proposed interface offers a number of dynamic and multimodal properties in a new fashion. In this setup, the paradigmatic problem of 'choosing one in 500 videos' is approached by combining *breadth first and a limited number of hierarchical levels* with a low number of items on screen to choose from at a given moment in time. These apparently contradictory requirements are achieved by three slowly rotating cylinders with facets or tiles containing information about the items to choose. The result is an interface which can be described as a Jackpot metaphore.

In chapter 2, a major breakthrough in multimodal integration (and cooperation in the MIAMI consortium) is presented. In this chapter, we present the moving and talking face: an artificial face is presented, which produces speech movements and a speech audio signal. The audio is spatialized such that the auditory location of the face matches the visual location on the screen. Furthermore, the face can be moved, using the meta device driver for single-point continuous control.

In chapter 3, the problem of robot navigation through remote control is addressed. In order to give a teleoperator improved information about the effects of his actions in the environment of the robot, force feedback is combined with acoustical feedback. These experiments are directed towards the design of the analogical demonstrator at the end of the project.

Chapter 4 presents an application in which we combine graphical information produced by the pen with an audio stream. A prototypical application is the mathematics teacher, working out a series of derivations. In this process, formulae are being written, deleted and

substituted, during which process the actions are explained by voice. Such an integration of modalities introduces new challenges at the technical level (synchronisation), but also at level of the user interface design.

In chapter 5, the area of interaction support is addressed. Modern user interfaces are designed to be context free and open, allowing all actions and options to be activated at any moment in time (the mode-less interface). However, in practice, users perform typical sequences of actions. Knowledge about the actual usage patterns, e.g., in statistical form, can be used by the system to speed up and facilitate the interaction process.

Chapter 6 presents an application to make and present slide shows on a system with an LCD projection device. It uses the pen and the ink data type in order to allow for quick annotation and lively animations with ink. It is based on the Tcl/TK environment standard of the MIAMI project.

Contents

Chapter 1

An interface for video-on-demand: the Jackpot

The Jackpot machine is a realization of a conceptual metaphore for the interface in interactive multimedia services aimed at consumers. Internet WWW is a simple example of interactive multimedia which will be followed in the future by a very wide range of services. The essential requirement by the consumer user accessing the services via a TV monitor located in the living room is that keyboard input is not convenient and text output on the screen is reduced to minimum. This is essential difference with the current WWW service on the Internet.

The Jackpot machine realizes the interface using a metaphore well-known to the standard user which allows to present on screen a wide range of services for selection in a dynamic way. The user is presented with a gambling machine which shows different services on its rotating wheels, see Figure 1.1. Each face of the wheels can represent a selection option and the rotation increases the number of options available. The user can control the operation of the machine and she/he can select specific service at will when it appears on screen. After the selection of a service, the jackpot machine can present specific offerings, for example movies. Numerous titles and images from the movies can be presented then on the faces of rotating wheels. One can imagine numerous ways of interaction with the jackpot, including a random selection of offerings.

This type of of dynamic interface is potentially very attractive for the user. It can also be seen as a part of a bigger metaphore in which multimedia entertainment services (movies, games, etc.) are represented by a 'casino' metaphore and the 'jackpot' machines located there correspond to the different types those services. In turn the 'casino' can be a part of an information city metaphore with buildings representing different services available like banking, travel agencies, musea, theaters, meeting places (cafes), shops, etc.

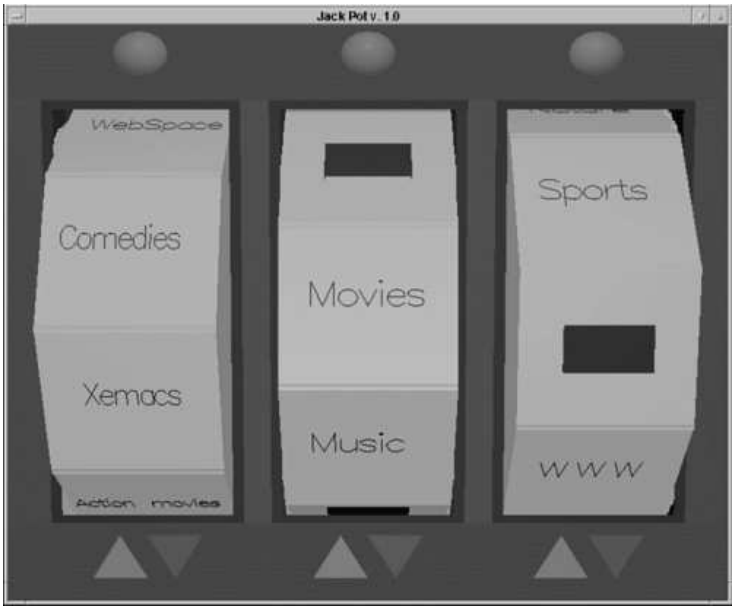

Figure 1.1: An example screen from the JACKPOT DEMO

In the context of MIAMI, the jackpot interface illustrates several problems involved in the design, operation and integration of multimedia information. The interface requires well-orchestrated integration of sound with 3-D graphics used for building the jackpot. Sound integration in the form of background music and synchronized noises accompanying operation of the machine is absolutely necessary for operation.

The graphics interface in the jackpot machine was implemented using OpenGL (RIIT). Sound is generated via the Audio Application Server (RUB) and the Meta Device Driver (UKA). Sound and graphics are synchronized using the audio server. The software is portable but the visual realism is very much depending on the power of graphics accelerator because user interaction demands real time computations. Realistic implementation of the complete information city metaphore would require machine with highest performance. This limits the complexity of the graphics used in the interface. It should be noted however that in the near future such performance will be available in the desktops and consumer set-top boxes. It is possible that with announced development both in the fields of available software and hardware that the jackpot demo could be expanded to incorporate additional features such as a casino environment.

Chapter 2

The Moving Talking Face

The moving talking face demonstration is a program built on three different MIAMI modules: the Talking face (ICP-FACE-ANIM), the META DEVICE DRIVER (UKA) and the Audio Application Server for 3D audio rendering (RUB). This application requires one input device (a mouse) and two output devices (a computer screen and headphones). This demonstration aims to illustrate how a multimodal object, in this case a talking face, can be used in a city navigation environment, for instance.

The ICP-FACE-ANIM object is written with OpenInventor. It is drawn with the help of a polygonal mesh and rendered with the Gouraud shading technique previously described. The whole structure of the face is stored in the Inventor's file format where all the different parts of the face are described. This file contains all the ordered vertices and normals, as well as the reflection coefficients of all parts. This file is the basis of a C++ object which contains all the methods to built up and animate the face.

As said above, the most important points here are

i) how the face movements can be synchronized with the speech signal, and

ii) how the position of the display and of the acoustics can be spatially rendered in a coherent way

The MDD is used in this example to move, from a 2D device such as the mouse, the face and the sound source location in two of the six degrees of freedom. The two degrees of freedom can be easily selected and modified through a graphic interface. According to the mouse position, the MDD sends the desired 3D location through a PVM socket and to both processes (ICP-FACE-ANIM and AAS). Then, the appropriate transformations are applied to the face. The corresponding HRTF coefficients are finally recalculated and applied to the speech signal.

As far as synchrony between face movements and speech is concerned, we use the synchronization module of the Audio Application Server. The previously recorded speech signal and facial parameters are stored in files. Each time the ICP-FACE-ANIM is ready to be drawn, it gets the number of the video frame to be displayed from the audio process (based on the number of audio samples, thus on the time elapsed since the beginning of the sentence). This process keeps on synchronizing face movements and speech without acoustic degradation, whatever the graphic computer used. The higher the graphic computer performance, the higher the video frame sampling. This solution preserves a continuity of the audio output.

Figure 2.1 represents the global interaction between the different modules.

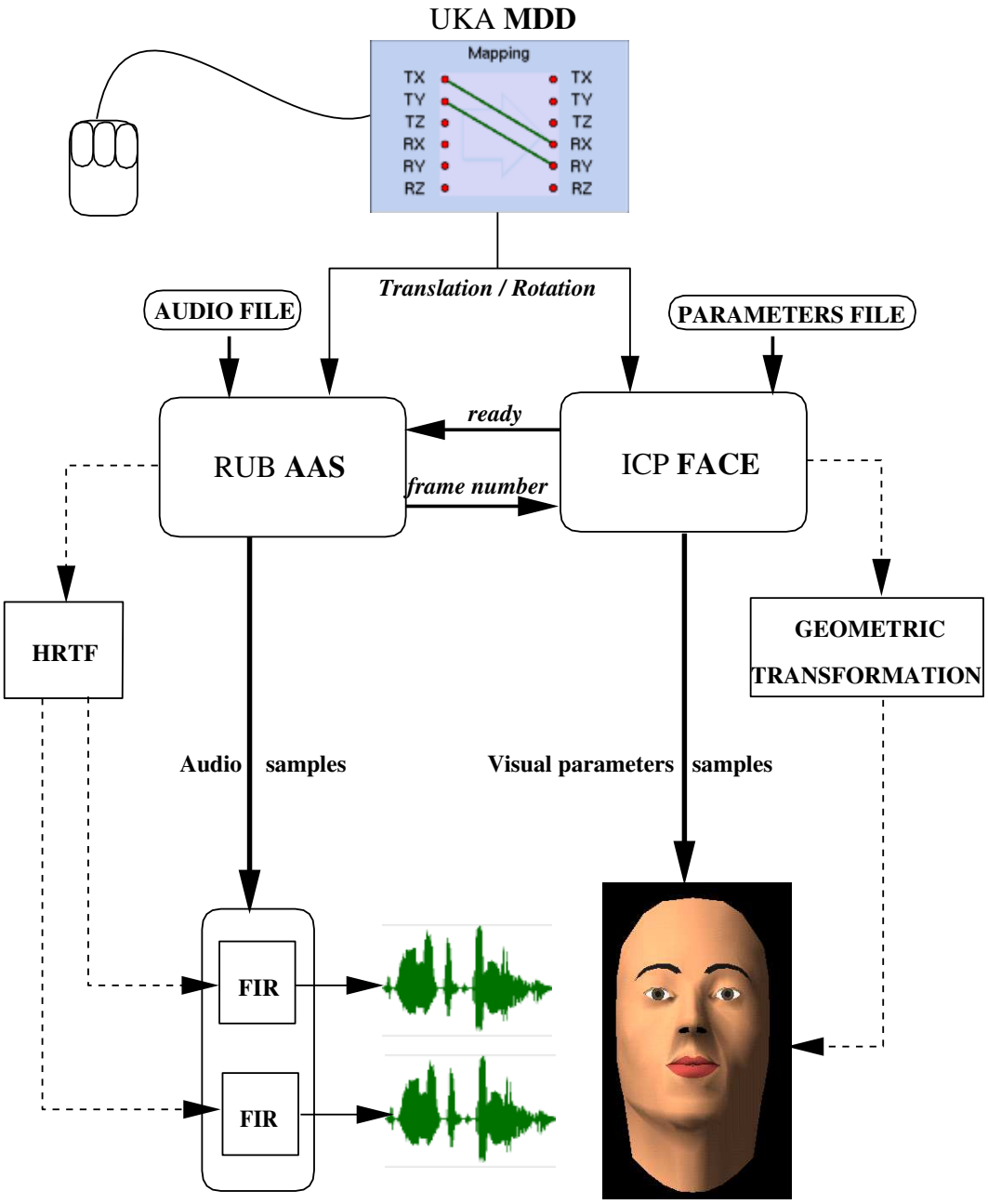

Figure 2.1: Architecture of the moving talking face demonstration

Chapter 3

Robot Navigation

Two applications have been designed and implemented in order to demonstrate the usage of multiple modalities in the robotics domain. In the first one, we have focused on the generation of force feedback in order to support an operator during interactive control of a mobile robot. The second one is an example for the integration of several modalities. Both robotic applications will be combined and extended in the analogical demonstrator which will be presented at the end of the project (see [1], section 7.2).

3.1 Interactive Control with Force Feedback

In mobile robotics, two main goals are addressed: First, it is desirable to realize a mobile robot which as much autonomy as possible. Therefore, they are often called 'autonomous robots' or 'autonomous mobile systems'. Unfortunately, this goal is only achievable with enormous effort regarding man-power and money. And even the best autonomous system will sometimes get into situations where it needs the help of a human operator.

Consequently, the second goal is to design control systems for mobile robots which support the human operator as much as possible. In our view, this means to give the operator the best feedback the system can generate. In the analogical demonstrator which will be presented at the end of Miami (see [1], section 7.2), we will include visual, auditive, and haptic feedback. The last one is what we have started with, so in the remaining part of this section we will describe our *basic control system with force feedback*. An overview of the principle system architecture is shown in figure 3.1. In each cycle of the control loop, input data is received from the user via the Meta Device Driver (MDD), the robot's world model data is updated, and finally an output is generated by the Force Feedback Generator (FFG) and send to the device via the MDD again.

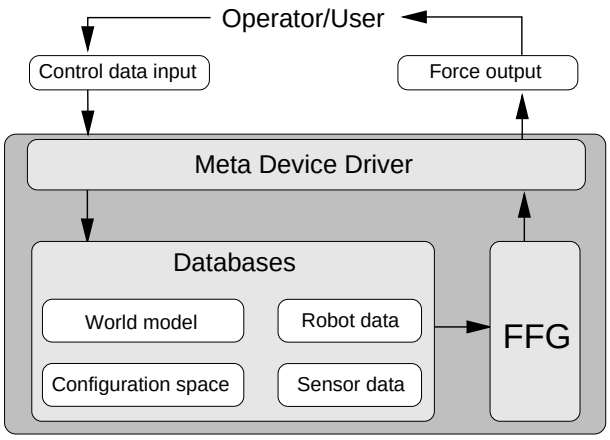

Figure 3.1: The principle structure of the robot control station with force feedback.

Control input and force output

The control system has been implemented in C with a Tcl/Tk user interface[1]. Its main input device is the FORCEJOYSTICK which has been developed during the first year of MIAMI, see figure 3.2. In *feedback mode*, the FORCEJOYSTICK does not only send control values to the application but also generates continuous force feedback controlled by two servo motors and a microcontroller, see right picture of figure 3.2. Although this device is not very sophisticated and could be improved in several ways, it is sufficient to demonstrate the principle appropriateness of our approach.

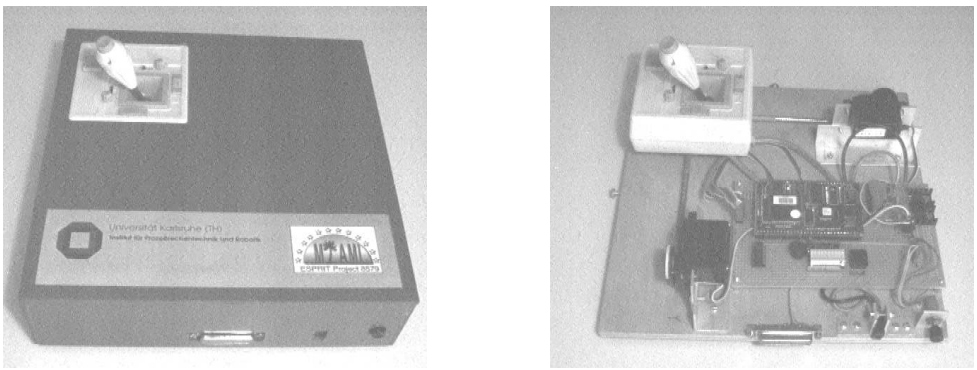

Figure 3.2: The FORCEJOYSTICK is an input device which can also generate force output. (Left picture: ready to use; right picture: a look inside)

[1]It is planned to realize a new version with 3D graphics in OpenInventor until the end of MIAMI.

Generation of force feedback

The FORCE FEEDBACK GENERATOR (FFG) has been developed in order to realize force feedback in a robot control system. Several methods including global and local decisions, taking into account the current position, direction, and velocity of the robot etc. have been combined to generate a situation dependent force feedback which supports the operator best. The basic structure of the FFG is shown in figure 3.3.

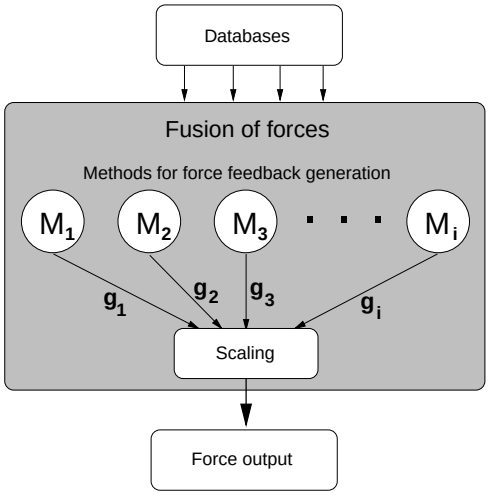

Figure 3.3: The FORCE FEEDBACK GENERATOR calculates a resulting force feedback based on internal model data, sensor data, input data, etc.

Different methods are based on different input data in order to realize independent modules. This approach has two major advantages: First, the modules can be implemented, improved, and exchanged easily. Second, if in some situations some input data is missing or noisy (e. g. sensor data from the ultrasonic sensor system), the system can still generate a force feedback although only some of the methods will contribute to the resulting output vector. An additional benefit is the chance to set a weight for each method. Thus, the influence of the different methods can be tested and an operator can set the weights according to her needs.

The next figures show the user interface of the current version. The first one (3.4) is a screendump of the complete interface. The region which is marked with a dashed line is shown in more detail in figure 3.5. The pictures show a floor with walls and obstacles, the path of the robot, the calculated boundaries, the resulting force vector, the current direction of movement and so on. The different lines are colored in the original version.

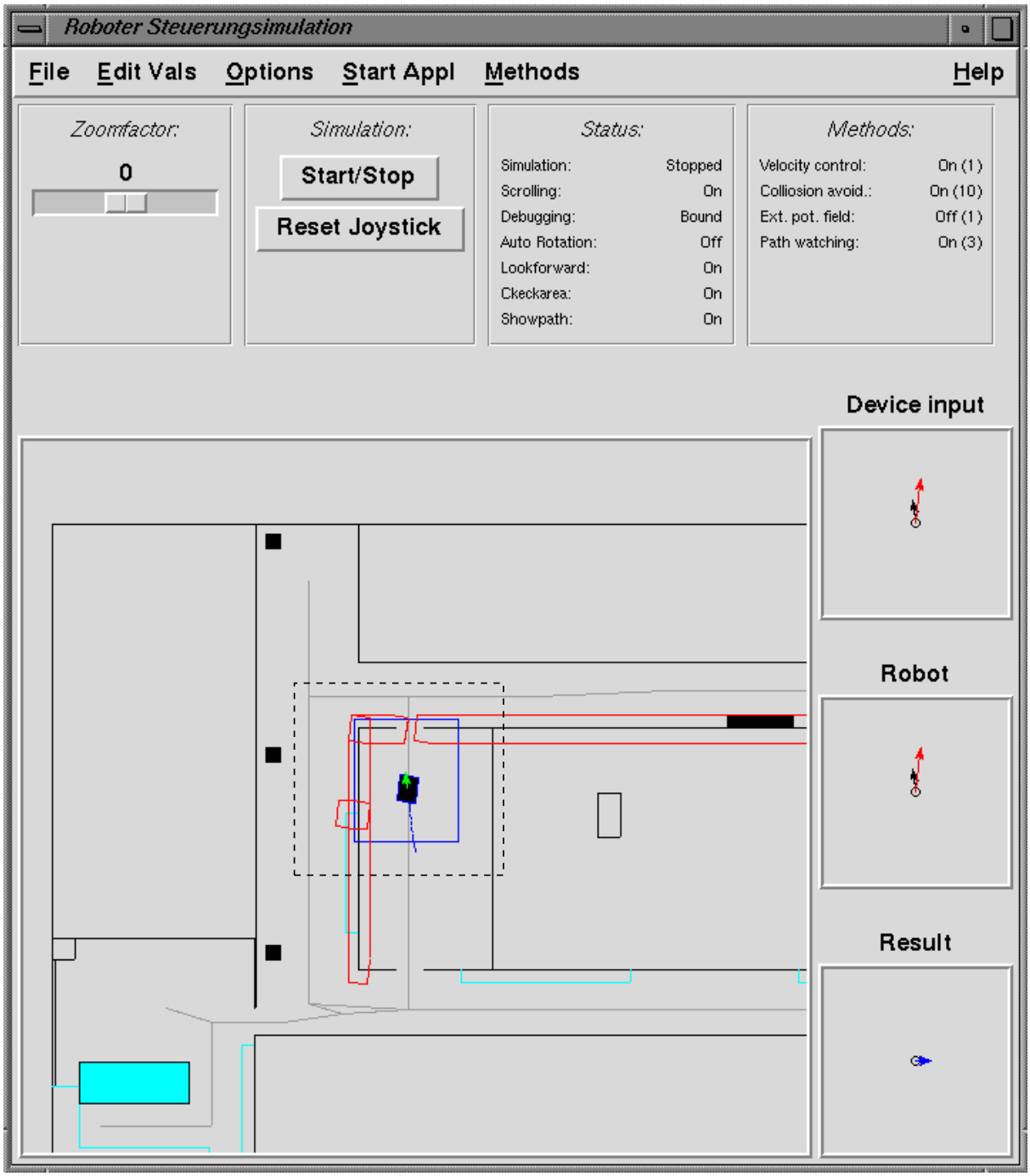

Figure 3.4: A screendump showing the user interface of the robot control system

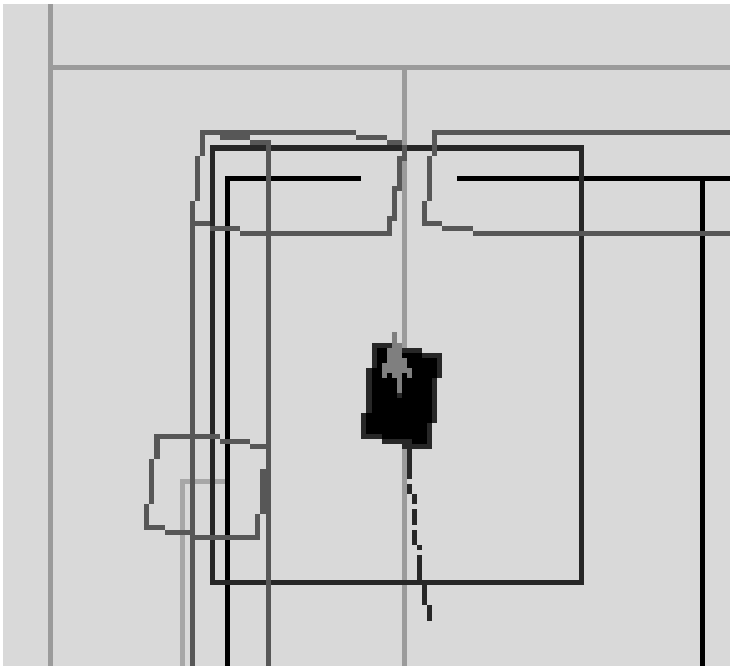

Figure 3.5: The region marked with a dashed line in figure 3.4 is shown in more detail here

3.2 Multimodal Robot Control

The second application is an example for the integration of several modalities. The service robot (see figure 3.6), a combination of a mobile platform and a manipulator, can be interactively controlled by an operator. Both objects are controlled independently by different input devices, and a third device is used to change the camera position if necessary. All devices are connected to the application with the help of the META DEVICE DRIVER (see [1], section 6.1), which means that an operator can exchange the devices and select an appropriate one for the control of each object without changing the devices' interfaces.

For the output, a 3D OpenInventor model of the service robot and a hotel environment is displayed on the monitor. In addition, the AUDIO APPLICATION SERVER (AAS, see [1], section 6.3) is used to provide acoustical feedback and the speaking face (see chapter 2) is connected to the application via PVM. With the help of the face, the system can 'communicate' with the operator by giving introductions or help, so it becomes easier to fulfill a specific task.

Modelling the robot

The service robot is a virtual compound object which has been virtually 'assembled' by combining two OpenInventor models of real objects: a PUMA 260b manipulator has been mounted on top of the mobile platform MORTIMER designed for a service robot in a hotel environment, see figure 3.6.

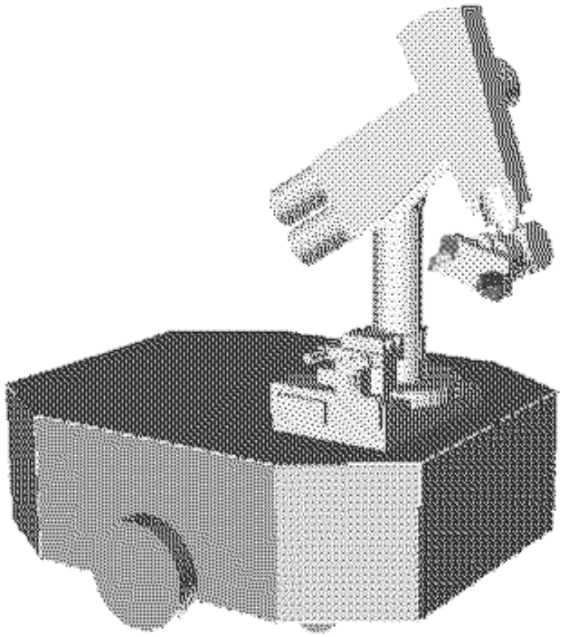

Figure 3.6: The UKA service robot

For the interactive control of 'active' objects, OpenInventor provides socalled *engines*. With these engines, translational and rotational axes can be defined as well as methods how to react to specific kinds of inputs. In the service robot model, we followed different concepts for the platform and the manipulator.

Platform For the *platform*, we have defined two engines: one translational engine for the platform's movement on the floor (two degrees of freedom) and one rotational engine for turning around its z-axis (the third degree of freedom). With these two engines, the way the real platform can be controlled has been imitated with respect to an external point of view,i. e. it is no emulation. The real platform is controlled with two motors which are connected to two wheels, one at each side.

Manipulator In contrast to this imitation, the behavior of the *manipulator* has been emulated by six rotational engines, each of which controls one of the robot's joints.

Therefore, this models copies—more or less—exactly the way the real manipulator is controlled. In addition, a two-finger gripper has been mounted to the manipulator's wrist in order to grasp and manipulate objects.

Multiple modalities

As said in the introduction to this chapter, the multimodal robot control application is an anticipation of the analogical demonstrator to be shown at the end of MIAMI. In the following, we will describe some of our concepts although not everything presented here will be realized until the 2nd review meeting in March 1996.

Multiple devices By using the MDD as a general interface to several input devices, an operator can select the appropriate device for a control task according to her needs. In the simulation, three active components might be controlled:

- The *mobile platform* can be moved in 2D on the floor and has an additional degree of freedom for turning around its z-axis. Here, the FORCEJOYSTICK will be the primary input device because it can also generate force feedback (see below).
- With 6 degrees of freedom (not including the gripper), the *manipulator* is significantly more complex than the platform. Efficient control is provided by input devices with at least 6 different values. Therefore, either the EXOSKELETON, the SPACEMOUSE, or the SPACEMASTER might be used for moving the manipulator.
- Finally, the *camera's viewpoint* can be interactively set by the operator. Because this is usually not a time-critical task, the selection of the best device is not as crucial as for the platform and the manipulator. A good choice would be the pseudo device TCLSCALES which provides 6 degrees of freedom.

Force feedback The FFG described in section 3.1 will be integrated in the final version of this application. When this is done, the platform can be controlled with the FORCEJOYSTICK in force feedback mode which allows to 'feel' it when the platform is approaching a wall or others obstacles.

Continuous acoustical feedback The artificially generated sound of a moving robot might be used with calculated reflections in order to give the operator a more natural feeling. This kind of feedback might also help to estimate the direction and distance to other objects when it is generated as spatialized sound, a feature which is provided by the AUDIO APPLICATION SERVER (AAS).

Discrete acoustical feedback Earcons are one way of isolated sound output. Discrete acoustical feedback can be useful to generate alarm signals, to indicate a person approaching the robot, or to give a kind of support in fulfilling a specific task.

Human-like communication With the *animated face* (see chapter 2), the robot's interface can be equipped with human-like communication capabilities. This is especially interesting when the platform is not operated by an expert but is used in a public area like an exhibition or a museum. Here, the animated face can give instructions, help the operator, or it might be used as a guide telling about things in the environment. In addition, the recognition of *natural speech* would be an interesting 'input device' especially for the last case, i. e. when the mobile platform would operate in a public area.

Chapter 4

Audioscript

Audioscript is a practical application which was devised specifically to investigate the integration aspects and cognitive enhancement of information transfer which might be possible to achieve by applying multimedia. At the same time the example is a testbed for technical aspects and solutions which are dealt using the tools developed in MIAMI.

The generic scenario of audioscript application is a lecture on a highly theoretical topic, in which many complicated formulas are written on the board by the lecturer who provides at the same time a suitable narrative description. Such lectures are typical in mathematics, physics and engineering and to be useful they require very good presentation skills as well as concentration from the audience. One can see that the sense of lecturing in this case lies in the fact that direct, dynamic audio-visual stimulation of the senses of participants takes place and this enhances the information transfer from the lecturer. Important in this case is perfect audio-visual synchronization (or orchestration) as well as dynamism in the information flow: formulas written on the board and speech stream. The total effect of this can not be substituted by reading a book or listening to the speech alone.

One could try to recreate this effect by recording and playing a video of the lecture. This process is has high bandwidth, processing and storage costs. Multimedia offers another possiblity which is just the recreation of the speech and the exact flow of the formulas on the screen. That is, participants will see the flow on the whiteboard on the screen and listen to the lecture. If both media streams are perfectly synchronized and made to feel as natural as possible (which requires high quality voice and well-looking script) one can argue that the information transfer will be enhanced because dynamic flow of information will be preserved in the condition of full concentration of the participant using personal device.

Audioscript represents a practical testbed implementation of this idea. In the implemen-

tation it was required to provide high-quality script production and an on-screen reading facility. This was done using a graphics tablet and viewer implemented in Tcl/Tk, see figure 4.1. The script can be made in the conditions resembling normal writing by pens used for presentations. At the same time audioscript provides audio recording and playback facility. The complete system for both modalities has a number of user functions to make its use convenient.

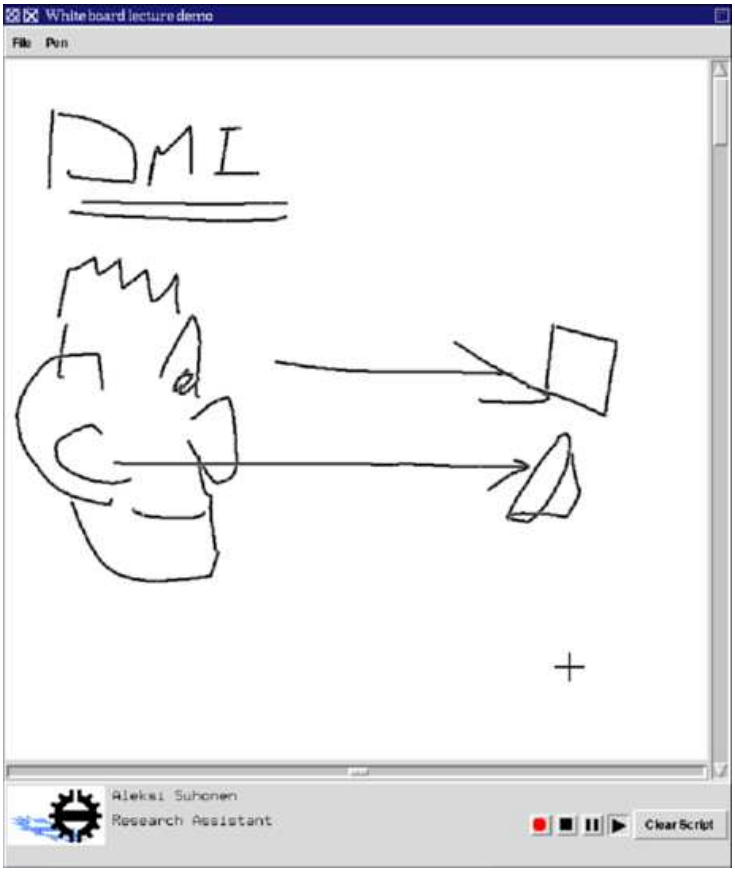

Figure 4.1: An example screen from the AUDIOSCRIPT DEMO

Perfect synchronization of scripting and speech required detailed investigation and has been realized by introducing time stamps into script samples in the recording phase. During the playback, packets of speech samples with known duration are controlling the writing of script samples on the screen.

The complete system is implemented using MIAMI modules, the audio server (RUB) handles sound I/O and the with meta device driver (UKA) controlling the input. These modules are integrated into the application using TkPvm (NICI). Furthermore, the GESTE pen gesture recognizer (NICI) can be added to expand its functionality in operation.

Chapter 5

Supporting Interactions

Based on the tools described in [1, ch. 1], a multi-agent system for observing the user's actions at the user interface level has been realized. Although it has been primarily designed in order to provide 'intelligent' *haptic* feedback, it can be combined with any other output modality due to its modular and flexible architecture [2].

5.1 Idea and Concept

The principle idea of this approach is to predict the next user action regarding the usage of widgets in the graphical user interface, i. e. to predict which interaction object (or widget) will be used next. Therefore, a user and application specific model based on empirical data is created. Basically, the user's interactions are 'observed' and analyzed by a multi-agent system and a statistic reflecting the user's behavior is generated. When a sufficient amount of data has been collected and analyzed, it constitutes the basis for the prediction of future actions. In order to increase the performance of the prediction component and to model the user's behavior as closely as possible, two different approaches—trajectory-based and dialog-based prediction—have been combined.

5.2 Prediction of User Actions

The focus of this work has been set to analyzing the user in order to provide haptic feedback when needed. The overall motto can be described as follows:

The main task is to **predict** *the next user action in order to launch the haptic feedback* **selectively** *and to* **adapt** *this capability over time.*

Nevertheless, the methods developed and implemented for this task can be used for basic user modeling as well as any other kind of feedback, may it be haptic, visual, or acoustical.

The basic problem of predicting the user's actions can be described as follows: Given a set of widgets (potential targets) and the user's observable behavior[1], find the widget which is most likely to be used next. This is a classical classification problem where a given input sample—the user's interactions—has to be classified in one of n classes—the potential targets.

At the user interface (UI), at least two different methods exist to predict the next action based on the analysis of the user's current actions: the first approach analyzes the features of the cursor's trajectory (data-driven, see figure 5.1), whereas the second one is a dialog-based approach in which the semantics of the UI and the dependencies of widgets are reconstructed (model-driven). To yield best results, both approaches can be combined easily.

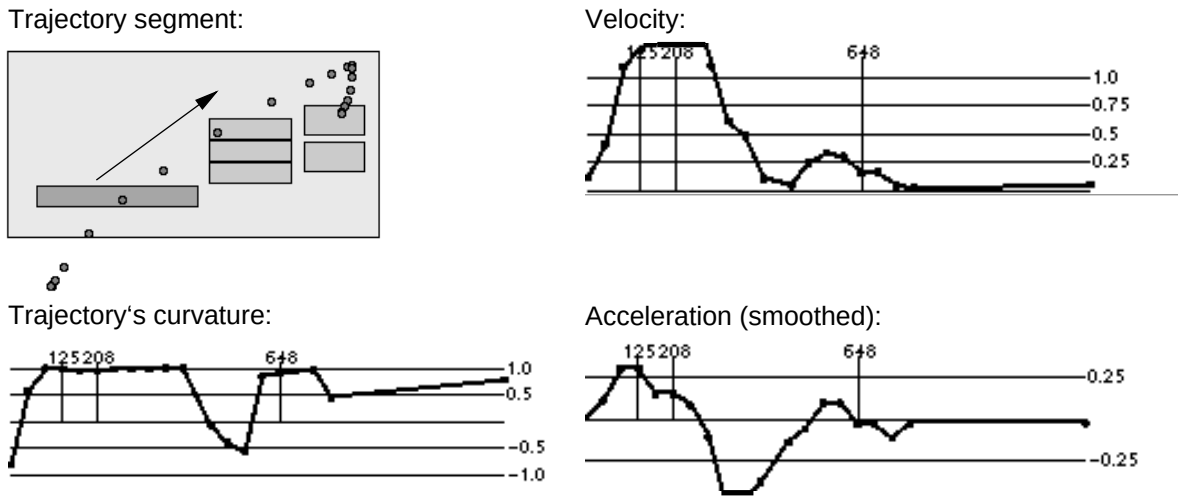

Figure 5.1: Example of a positioning operation

[1]Based on the events provided by the Xlib.

Trajectory-based prediction

The implemented procedure is based on *stochastical classification*. The principle idea is to observe the user's interactions for some time, to collect the features extracted from the trajectories of these interactions, and to set up a statistic of the user's 'behavior'. After some time, the statistic will become more and more powerful and can be used to predict the next action based on previous ones (a-priori-probability). Therefore, the following formula has to be evaluated:

$$P_j(pos|\vec{m}_i) = \frac{P_j(\vec{m}_i, pos)}{P(\vec{m}_i)} = \frac{h_j(\vec{m}_i, pos)}{h_j(\vec{m}_i, pos) + h_j(\vec{m}_i, \neg pos)}$$

Here, $\vec{m}_i$ is a feature vector, h_j a frequency, and *pos* an indicator whether the widget will be the target of the next positioning operation or not. In order to set up the statistic containing the usage frequencies for all widgets, the feature vectors have to be segmented and classified in discrete steps. For the implementation, the following features with the given number of steps have been used:

Feature d	**# steps** k_d
Velocity	5
Smoothed acceleration	3
Trajectory curvature	2
Stopping distance	3
Overall distance	2
$\|M\| = \prod_{i=1}^{d} k_i$	180

In other words, the feature space covers 180 distinct cells. Nevertheless, the prediction usually starts to work after a small number of positioning operations because some widgets are much more frequently used than other ones. By saving the collected data (i. e. the statistic), it can be used in the future for the prediction and it will be refined and adapt with every new user interaction. Thus, a user *and* application specific model of the usage of different widgets will be created.

By using these statistics in combination with *adaptable thresholds* θ_j, a decision whether to launch some feedback or not becomes possible:

$P_j(pos|\vec{m}_i) >= \theta_j \Rightarrow$ User *will* use ω_j, feedback *will* be launched.
$P_j(pos|\vec{m}_i) < \theta_j \Rightarrow$ User *will not* use ω_j, feedback *will not* be launched.

Dialog-based prediction

The second method to predict the next user action has been adapted from *speech recognition*. Instead of using the well-known formula of Bayes, socalled *trigram probabilities* approximate the a-priori-probability: $P(W) \approx \prod_j P(\omega_j|\omega_{j-2}, \omega_{j-1})$.

The idea is to model the semantics of the man-machine-dialog. Usually, some widgets will be used right after a specific other widget has been used. The trigram probabilities are one way to model these interdependencies.

The widgets which already have been used until a certain time are $\omega_1, \omega_2, \ldots, \omega_{j-1}$. Thus, $P(\omega_j|\omega_{j-2}, \omega_{j-1})$ determines the probability that widget ω_j will be the next target to be used if ω_{j-2} and ω_{j-1} were the last two widgets used. This probability, which is completely independent of the probability calculated in the trajectory-based approach, can be modelled as follows:

$$P(\omega_j|\omega_{j-2}, \omega_{j-1}) = \frac{h(\omega_{j-2}, \omega_{j-1}, \omega_j)}{h(\omega_{j-2}, \omega_{j-1})}$$

Here, $h(\omega_{j-2}, \omega_{j-1}, \omega_j)$ denotes the frequency of the usage of the positioning sequence $(\omega_{j-2}, \omega_{j-1}, \omega_j)$.

By introducing a weight λ, the overall probability to use a specific widget ω_j can be calculated from a combination of both, the trajectory-based and the dialog-based prediction:

$$P_{tot} = \lambda P_j(pos|\vec{m}_i) + (1 - \lambda)P(\omega_j|\omega_{j-2}, \omega_{j-1})$$

5.3 The Multi-Agent Architecture

The most flexible implementation of a software system can be achieved by realizing the single parts as independent modules which are able to exchange data. When the modules are completely independent, realized as separate processes and fulfilling some criteria (see [3]), they are said to be *agents*, and the architecture is called a *multi-agent system*, see figure 5.2. By using PVM[2] as the underlying software package for interprocess communication, such a system is relatively easy to realize and might be extended with other components with minimal effort.

The system is composed of four main components: The control tool FORCEAGENT is the heart of the system. It manages the statistical model, analyzes the user's behavior,

[2]PVM (Parallel Virtual Machine) is a public domain software which has been described in more detail in [1, ch. 1].

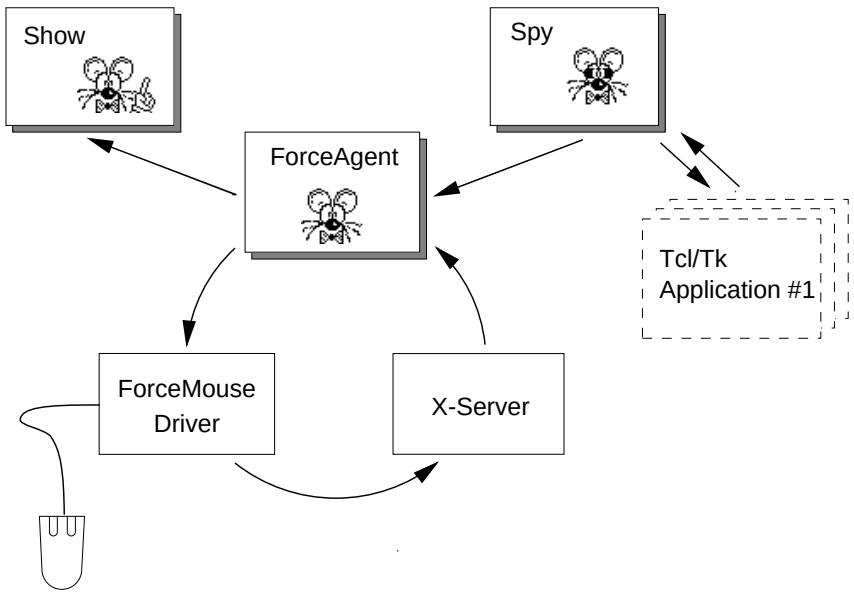

Figure 5.2: The multi-agent architecture

predicts the next widget, and launches the feedback by sending a command to the second component, the FORCEMOUSE driver. It is a low-level module which provides several commands for different feedback modes.

In order to increase the system's performance and to reduce the FORCEAGENT's load, the observation tool SPY has been developed. This module creates a simplified model of the GUI which contains all data needed by the FORCEAGENT. Finally, the visualizer tool SHOW has been implemented. It can be used to visualize the internal models and to record and play back movement trajectories, but it is mainly intended to be used during system development and debugging.

5.4 Specific Advantages

1. The system is a plug-and-play solution which is fully compatible with any Tcl/Tk application (versions 7.4/4.0 or later) running under X. In order to use the haptic modality, the multi-agent system runs in the background, identifies running Tcl/Tk applications, and modifies them without affecting the source code.

2. The implementation is very flexible. Although designed to support haptic feedback, it might be used for other modalities as well.

Chapter 6

Sheet Viewer

The SheetViewer is an application which allows the user to create a presentation or slide show. Running on a PC or notebook computer, it presents its graphics on a large screen via an LCD projector device. A number of commercial applications exists, but none of them includes the use of live and animated ink as a data type. The SheetViewer application is written for the MIAMI project for a number of reasons:

1. It shows the use of the pen as a pointing device and a graphical data (=ink) entry device,
2. It is an application which uses the event-driven environment as designed for MIAMI,
3. It is based on the modular approach, so new item types or input devices can always be added,
4. It can be used in Miami presentations on workshops and meetings.

Ink is being used in two ways: (1) the lecturer may add annotations to the sheets, live. This can be done in reaction to questions from the audience, for instance. (2) In the second mode of ink usage, the user may start an animation of pre-recorded ink. The advantage of the latter option is that a lecture has more cognitive resources available to keep in contact with the audience than if some live modifications have to be performed, and the graphical quality is guaranteed. However, the addition of annotations in real time also make a presentation more lively, as in the Audioscript application (Chapter 4).

6.1 Two modes of interaction

The SheetViewer can be used in two modes: "Edit" and "View". In the "Edit" mode, the user can edit the page in all kind of ways:

- create items (use the third mouse button for a menu)
- drag or resize items
- delete items (drag outside sheet area)
- record ink
- type text via the keyboard
- transform ink into another item or data type

In the "View" mode, the editing is restricted, to prevent the lecturer from destroying essential information. But the user can still create new ink on the page, for live annotation. Clicking on an item doesn't select it for editing, but may result in another user-specified action (e.g. a link to another page, or the start of an animation)

The following items are supported now, but it is easy to add any number of new items:

title Title of sheet

header Header of sheet (in the upper right corner)

footer Footer of sheet (in the lower right corner)

arg Text aligned below the title. Can also contain a link to other sheets

partners a list of logos (in the lower left corner)

photo Any picture (must be in GIF, PPM, or XBM format)

rect Rectangle

oval Oval

line Line

arrow Arrow

txt Text (freely movable in page)

ink Ink. Any sequence of coordinates.

lips An example how to add your own items. It contains a model of the lip-parameters designed by ICP.

Based on a limited number of actions that a lecturer may want to use, the bottom of the SheetViewer screen (Figure 6.1) contains the following buttons:

Previous - go to previous sheet

Back - go to top sheet

Save - save content of sheet to file

Postscript - create postscript output from content of sheet

Clear ink - clear all ink on page

Next - go to next page

6.2 Modular approach

Each item type is fully described in a separate file. For example, the "oval" item is described in the file "oval.tcl". This file is supposed to contain the following functions.

```
foo create foo
save_foo save foo to file
play_foo describes what happens if foo is clicked on. (optional)
```

where "foo" is the name of the item. The best way to create your own item types is to copy an existing item into another file and make the necessary adaptations. All you have to do to use this new item is create a new index file (tclIndex). Saved sheets also have the ".tcl" extension, because the commands in it are just Tcl commands. Loading a sheet is in fact nothing more than loading the file and execute the commands in it. For postscript output the extension ".ps" is used instead of ".tcl". The extensions used are:

```
".ink" ink-data
".ibk" backup of ink-data
".tcl"  sheet or item description
".tbk"  backup of sheet
".ps" postscript output of sheets
```

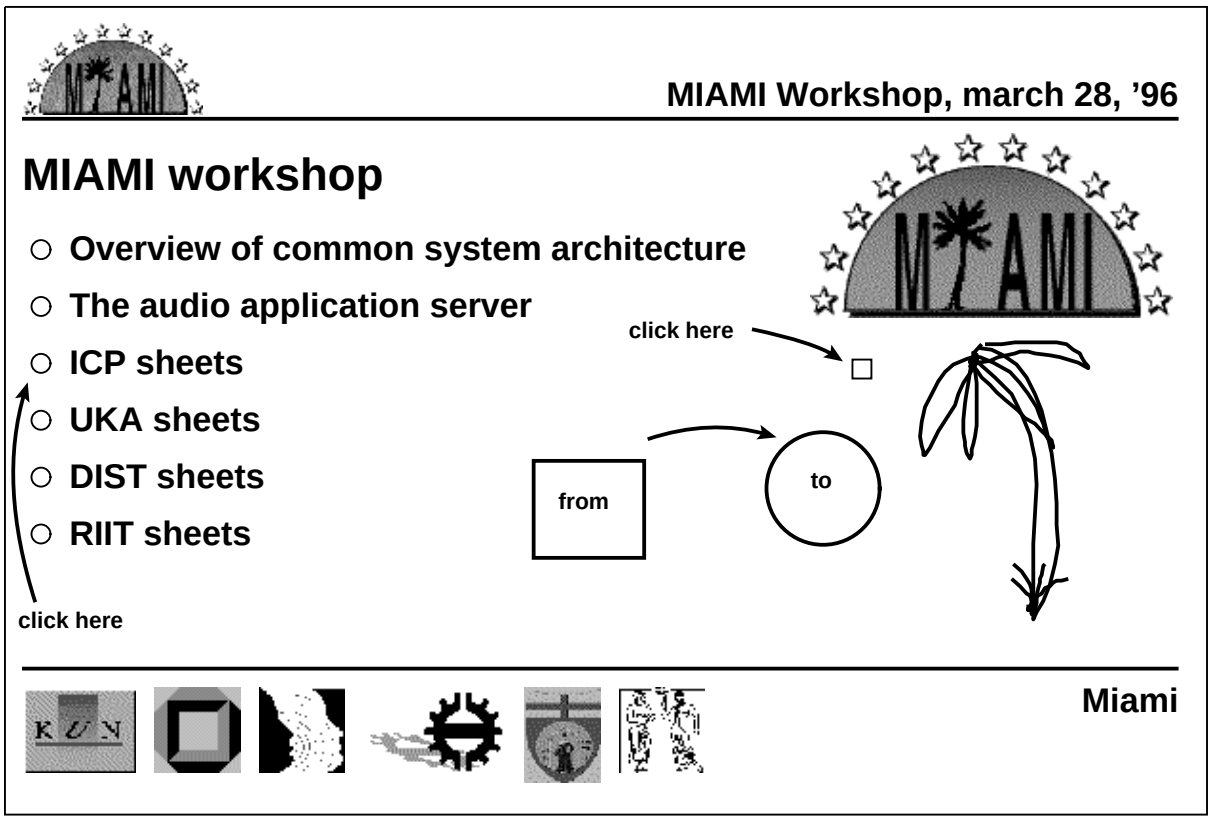

Figure 6.1: A screen dump of the SheetViewer program

6.3 Portability

The SheetViewer needs Tk4.0 or higher. It is not dependent on any of the MIAMI extension (as long as you don't use other input devices). Because Tk4.1 is available now for Windows and Mac too, it runs on these machines too. And of course it will run on any UNIX station.

Bibliography

[1] K. Hartung et al. DI 3 - Development of a System Architecture for the Acquisition, Integration, and Representation of Multimodal Information. Technical report, ESPRIT Project 8579 — MIAMI, 1996.

[2] S. Münch and M. Stangenberg. Intelligent control for haptic displays. In *Proc. of the Eurographics '96 (to appear)*, Poitiers, France, Aug. 1996. INRIA.

[3] M. J. Wooldridge and N. R. Jennings, editors. *Intelligent Agents — Theories, Architectures, and Languages*, volume 890 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, Jan. 1995.