Die approbierte Originalversion dieser Dissertation ist an der Hauptbibliothek der Technischen Universität Wien aufgestellt (http://www.ub.tuwien.ac.at).

The approved original version of this thesis is available at the main library of the Vienna University of Technology (http://www.ub.tuwien.ac.at/englweb/).

TECHNISCHE UNIVERSITÄT WIEN

VIENNA UNIVERSITY OF TECHNOLOGY

DISSERTATION

Designing multimodal interaction for configurable distributed systems

ausgeführt
zum Zwecke der Erlangung des akademischen Grades eines
Doktors der technischen Wissenschaften

unter der Leitung von
Univ.-Prof. Dr. Ina Wagner
Institut für Gestaltungs- und Wirkungsforschung
Arbeitsbereich CSCW

eingereicht
an der Technischen Universität Wien
Fakultät für Informatik

von
Dipl.-Ing. Thomas Psik
Piaristengasse 2-4
1080 Wien
Matr.-Nr. 9125243

Wien, im Oktober 2004

Dipl.-Ing. Thomas Psik

Designing multimodal interaction for configurable distributed systems

Thomas Psik – Dissertation

Reviewers:
Ina Wagner
Dieter Schmalstieg

Abstract

Designing multimodal interaction for configurable distributed systems is a challenge that originates from the early work of M. Weiser in 1991 described in the article "The Computer for the 21st Century". Each of the aspects – *multimodal interaction* and *configurable distribution* – is being actively researched by different research communities. A number of projects also attempt to combine these aspects, with the goal of creating a flexible and usable system.

Multimodal interfaces described in this thesis are: graphical user interfaces (GUI), tangible user interfaces (TUI), gesture-based interaction, and speech recognition. Real-time response and feedback are important issues for the interaction with a system. Configurable and tailorable applications allow users to adapt the system to their needs, hence increasing performance and improving the ability to cope with different types of users (novice, beginner, intermediate and expert users), or different hardware. Distribution of applications allows gains in processing speed as well as support of multiple users, interacting with each other in real-time and across time and space boundaries.

Interaction design patterns have been developed that ensure a proper design of the interfaces, especially important when developing applications that support ubiquitous interfaces. Rules that enforce these patterns, can be used by developers to verify the usability of the interfaces created.

This thesis is based on the experience of working on two different systems: the ATELIER project and the STUDIERSTUBE framework. Several applications of both systems are discussed from a technical and interaction design view. Based on these applications and on the experience gained during the work on the two systems some *design patterns* are described.

These design patterns are often not dealt with by developers, but are important for creating a usable system that supports users working with *multimodal, configurable, and distributed* applications.

Kurzfassung

Das Design von multimodaler Interaktion für konfigurierbare verteilte Systeme ist eine Herausforderung, die auf das frühe Werk von M. Weiser im Jahr 1991 beschrieben im Artikel "The Computer for the 21st Century" zurückgeht. Jeder der Aspekte – multimodale Interaktion und konfigurierbare Verteilung – wird von aktiv von verschiedenen Gruppen erforscht. Eine Reihe von Projekten versucht diese Aspekte zu vereinen, mit dem Ziel ein flexibles und brauchbares System zu erstellen.

Multimodale Schnittstellen die in dieser Arbeit beschrieben werden sind: Grafische Benutzerschnittstellen, "begreifbare" Schnittstellen (Tangible Interfaces), Interaktion, die auf Gesten basiert, und Spracherkennung. Eine unmittelbare Reaktion und aussreichende Rückmeldungen vom System sind wichtige Voraussetzungen für interaktive Systeme. Konfigurierbare Applikationen erlauben es dem/der Benutzer/in das System an die persönlichen Vorlieben anzupassen, dadurch können verschiedenen Typen von Benutzern (Anfänger bis Experte) optimal unterstützt werden, ebenso ist die Verwendung von unterschiedlichen Hardware-Konfigurationen möglich. Die Verteilung von Applikationen kann die Geschwindigkeit der Programme erhöhen, erlaubt aber auch den Benutzer/inne/n in Echtzeit bzw. über Raum- und Zeit- grenzen hinweg zu kommunizieren.

Beim Entwickeln von Programmen, die allgegenwärtige Schnittstellen (ubiquitous interfaces) unterstützen, müssen grundlegende Interaktions-Design Regeln beachtet werden. Diese Regeln können verwendet werden, um die Benutzbarkeit der erstellten Schnittstellen zu überprüfen.

In dieser Arbeit werden die durchgeführten Arbeiten an zwei unterschiedlichen Systemen (ATELIERund STUDIERSTUBE) beschrieben. Mehrere Applikationen der beiden Systeme werden hinsichtlich ihrer technischen und interaktionsspezifischen Umsetzung vorgestellt. Basierend auf den Erfahrungen, die bei der Erstellung dieser Applikationen und dem Arbeiten mit den Systemen gewonnen wurden, werden Design-Muster vorgestellt.

Die in dem letzen Kapitel beschriebenen, teilweise in den beiden Projekten umgesetzten Funktionalitäten, sind für die erfolgreiche Umsetzung eines *multimodalen, interaktiven, konfigurierbaren, und verteilten* Systems von Bedeutung.

Acknowledgements

During the work for this thesis I worked together with great group of people (*great* in the sense of *a large number of people* and also *personally great*).

I want to thank my family for their support and for *not* keeping me busy during the work on this thesis.

Special thanks go to *Prof. Wagner* and *Prof. Schmalstieg* for giving me a (paid) workspace to develop my ideas and letting me be part of their project teams, the ATELIER team and the STUDIERSTUBE team. Also for the valuable discussions and giving me the chance to *see things differently.*

Prof. Wagner helped me to understand that there also other important things – besides frame-rates and sophisticated technical solutions, *Prof. Schmalstieg* showed me what the scope of *scientific* work is.

From all the people I had the great pleasure to work with, I want to thank – first of all – my dear colleague *Krešimir Matković*, who helped me in so many ways: discussions on software-design, interaction-design, for sharing economic and scientific work, writing papers and for having fun while working.

The people I worked with through the years, each contributed to the way I work, mentioning them in historical order seems to be appropriate:

From Imagination (though not part of the ATELIER team) I want to thank *Reinhard Sainitzer*, for showing me how to write software that is designed properly, can be debugged, and then actually works ☺. Also from Imagination: *Thomas Gatschnegg*, because he is hardly ever mentioned, though he did a marvellous job in the ATELIER project, and for having interesting discussions about *scientific* versus *economic* work practices.

From the ATELIER project team I want to mention – Giulio Iacucci for discussing with me my suggestions and then creating working prototypes. *Antti Juustila*, *Pekka Pehkonen*, *Toni Raisanen* all from University of Oulu, Finnland for their prompt help on various software problems with the Ker-

nel. *Peter Warren* from II Malmö, Sweden for keeping up the spirit in the ATELIER project and his never ending efforts to keep the developments technically synchronised, and *Marco Loregian* from University of Milano Bicocca, Italy for never giving up on the ontology support and for our fruitful cooperative scientific work.

Andreas Rumpfhuber for organising all ATELIER related stuff at the Academy of Fine Arts, Vienna (fighting with administrators, ...), making pictures, and his strife for thoroughly documentation.

From the STUDIERSTUBE team I want to thank Gerhard Reitmayr, for being a *living* STUDIERSTUBE *and* OpenTracker *manual* ☺, his enormous knowledge about software design and the will to share this knowledge, and for the LaTeX 2_ε template from his thesis. *Florian Ledermann* for letting me use parts of his Master thesis, but also for the discussions about configurability. *István Barakonyi* and *Hannes Kaufmann* for discussing their projects with me and letting me collaborate. *Daniel Wagner* for helping me with my personal handheld and the discussions about computer games and movies ☺. *Joseph Newman* for proof-reading our papers, *Tamer Fahmy*, *Thomas Pintaric* for their general help on various occasions.

Thanks go to our student *Valèrie Maquil* for the work on the STUDIERSTUBE PUC framework and to *Jeffrey Nichols* – the developer and maintainer of the PUC framework at the Mellon University, USA for granting us access to the PUC source.

Thank you all.

For my grandfather Prof.Dr. Franz Weiler

for being my never-ending inspiration.

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation and Problem Statement

The aim of this thesis is to provide a framework for discussing *multimodal interaction* for *configurable distributed* systems. Based on the systems described and on our own experience in creating such systems insights are provided that should help and encourage researcher aiming at creating or modifying such systems.

- *Interactive* means that the system should directly and immediately react on a user input. Design issues are response time, feedback and handling errors. Interactive also incorporates the meaning of users and the system being "active" and that a *communication* between both is actually taking place.

- *Multimodal* in this thesis has the meaning of different input and output channels. Users can interact with a computer system using different means: through keyboards, pointing devices, speech and gestures. The computer system on the other hand can respond to the interaction in many ways: sounds, text-to-speech, images, videos, haptic feedback - like vibration or wind -, even olfactory, and gustatory outputs are possible outputs in an augmented environment. Multimodal output – especial the later ones – are uncommon in current setups, yet research is conducted in finding out what "augmenting an environment" really means and how to address those senses.

- *Configurable* means that the users have control how the systems interprets the interaction they are performing. In general the users should be *in control*, meaning that they can control: which input and output

components are being used, how much feedback is displayed by the system, connecting functionalities and changing the interaction patterns, which ultimately results in designing their own system. This possibility (for the users to design their own way of working) blurs the distinction between *user* and *system designer* and demands for a new type of system design.

- *Distributed* computer systems in this thesis, means that more than one computer is used simultaneously, that input and output are distributed, and that any number of users (one to many) can make use of the distributed input and outputs. Distribution also has the meaning of spatial distribution, meaning that actions and reactions are spread over the space, allowing the users to create their own working places.

1.2 Contribution

Pierre Weller et al. stated in 1993 in the article "Back to the real world" [151]:

> Computer-augmented environments raise many issues, both technical and social. They may require a complex, distributed infrastructure, precise alignment between the real and electronic worlds, novel input and output devices, ...

In the scope of this thesis principles will be described and discussed that should be applied when creating multimodal distributed systems. These principles were developed during the work on two different frameworks ATELIER and STUDIERSTUBE. Both projects aim at creating a system as described in [151]. ATELIER and STUDIERSTUBE represent two quite different strategies for implementing such a *computer–augmented environment.*

Additional to the discussion of those two systems, literature that has been published in the field of multimodal interaction, human computer interaction (HCI), computer supported collaborative work (CSCW), interaction design, tangible interaction (tangible user interfaces - TUI), distributed systems, augmented reality (AR), and software design is presented to set a frame for the discussion.

To be an expert in all of these fields is nearly impossible, but all of them are needed if a good and usable system has to be created.

1.3 The Systems

In the last three years I had the opportunity to work on two projects that focus on creating such a system. Each of these projects has its own goals, while the focus of the ATELIER project is creating a flexible ubiquitous environment used by design students, the STUDIERSTUBE project is devoted to building a framework and application programming interface (API) for various applications in the field of augmented reality, to be used by software developers. A short description of each project is now presented followed by a list of applications that will be discussed in this thesis.

1.3.1 The Atelier framework

The ATELIER project (architecture and technology for inspirational learning) aims to make a contribution to our understanding of inspirational forms of learning and to building augmented environments. The project has studied design education practice, developed prototypes to enhance such education, introduced prototypes to different real use settings (design and architecture classes), and partly in collaboration with the students reflected upon the interventions to learn both about how to improve architecture (of the learning) space, technology and the learning situation.

The ATELIER experiences are related to the general field of ubiquitous computing and especially to ideas of embodied interaction as a new stance for understanding of both, social and physical, interaction with artefacts and space. The concept 'configuration of mixed objects' forms an interesting challenges for the design of inspirational learning environments beyond the physical–digital divide.

1.3.2 The Studierstube framework

The STUDIERSTUBE framework was created in 1996 at the Vienna university of technology [123]. This framework is used to create multimodal, multi-user augmented reality (AR) applications. The focus of AR lies in overlaying the *Real* and the *Virtual*. A vast number of applications have been created using this framework and have also been published in the past. The evolution of this system is still going on and I am thankful to be part of that ongoing process.

> The goal of the development of STUDIERSTUBE is a software framework to support the technical requirements of augmented reality applications. It is a set of extension nodes to the Open

> Inventor [131] rendering library and an additional layer of objects that provide advanced runtime functions. It includes support for interaction based on 3D tracking events, rendering and output modes for all available virtual and augmented reality output devices, tools for developing distributed applications, and user management functions to support multiple users in a single setup. (*from [115]*).

1.4 Applications analysed in this thesis

Several applications have been developed for both of these systems. An overview of the discussed applications is presented in the table 1.1. I participated in the design and implementation of most of them, therefore this thesis includes personal experience gained during the realisation of those applications.

The *AR Puppet Framework* [6] is the work of István Barakonyi and will be part of his PhD thesis. The *APRIL framework* was created by Florian Ledermann and was described in detail in his master thesis [76]. *Construct3D* was created by Hannes Kaufmann and is the topic of his PhD thesis [69].

The work on these applications led to the concepts discussed in this thesis. Actually implementing those applications provided us with insights and understanding of the issues that are being raised during the development. We believe that proving the concepts through implementing and testing is the best method of exploring these issues and gaining new insights.

The development of prototypes helps both software designers and users to discuss about the solutions. Through the feedback of the users progress is ensured, sometimes leading to a total redesign of the concepts, and sometimes the users also redesign their work practices due to solutions presented. This design process is formally called "user-centred design".

The work presented here contains material previously published in:

- T. Psik, K. Matković, R. Sainitzer, P. Petta, and Z. Szalavári. The Invisible Person: Advanced interaction using an embedded interface. In *Proceedings of the workshop on Virtual environments 2003*, pages 29–37. ACM Press, 2003.

- K. Matković, T. Psik, and I. Wagner. The Sensitive Sample. In *Proceedings of the workshop Beyond Wand and Glove 2004, VR2004*, pages 21–24, 2004.

Application	Project	Issues addressed
Configurator	ATELIER	Interaction Design, Physical Handles, Multimodal Interaction, Distribution, Configurability, I/O Abstraction
Sensitive Samples	ATELIER	Tangible Interaction, Mixed objects, Physical Handles, Interaction Design, I/O Abstraction
Texture Painter	ATELIER	Physical Handles, Distribution, Mixed objects
Tangible Image Query	ATELIER	Tangible Interaction, Feedback, Distribution
Ontology Integration	ATELIER	Physical Handles, Feedback, Distribution
HMDB2HTML	ATELIER	Software Design, Multimodal Interaction
Invisible Person	TMW	Multimodal Interaction, Distribution, Software Design
APRIL Framework	STUDIERSTUBE	Distribution, Configurability
AR Puppet Framework	STUDIERSTUBE	Mixed objects, Multimodal Interaction, Distribution, Software Design
PUC Framework	STUDIERSTUBE	Software design, Multimodal Interaction, I/O Abstraction, Distribution
Construct3D Redesign	STUDIERSTUBE	Interaction Design, Multimodal Interaction, I/O Abstraction

Table 1.1: Overview of the discussed applications.

- K. Matković, T. Psik, I. Wagner, and W. Purgathofer. Tangible Image Query. In *Proceedings of Smart Graphics: 4th International Symposium, SG 2004, Banff, Canada, May 23-25, 2004. Proceedings*, pages 31–42. Springer-Verlag Heidelberg, 2004.

- I. Barakonyi, T. Psik, and D. Schmalstieg. Agents that talk and hit back: Animated agents in augmented reality. In *Proc. ISMAR 2004*, page *to be published*, Washington, USA, October 2004. IEEE.

- T. Binder, G. De Michelis, M. Gervautz, G. Iacucci, K. Matković, T. Psik, and I. Wagner Supporting Configurability in a Tangibly Augmented Environment for Design Students, In *Special Issue on Tangible Interfaces in Perspective*, Pers and Ubiq Comp Journal, Springer Verlag, *forthcoming.*

- G. Iacucci, K. Matković, T. Psik, and I. Wagner. Configurability in and integration with the environment: Diverse Physical Interfaces for Architecture Design. In *Online Proceedings of Physical Interaction (PI03)*, Workshop on Real World User Interfaces, 2004.

Chapter 2

Setting the Frame: Systems

Both projects aim at providing *multimodal interaction* for a *configurable distributed* system and have proved that they are suitable for creating useable applications that incorporate all these qualities.

ATELIER and STUDIERSTUBE. The work on these two systems has been demanding as both systems have their own way of providing solutions to a problem. Before starting to work with a system one has to understand the basic concepts and but also studying the details, to be able to use the full power of a framework. Both systems are fundamentally different in concept and even in the programming language.

STUDIERSTUBE is implemented (mainly) in C++ [29], while ATELIER is implemented using Java[132]. Both are designed "object oriented" providing the programmer a large number of objects, that can be used to implement individual solutions.

Both systems make use of XML[148], the eXtensible Markup Language, which is the emerging standard primarily aimed at web-based applications and software systems. XML is a markup definition language that allows to define hierarchical markup languages. There are a number of software libraries and tools available for creating, parsing and validating XML structures.

STUDIERSTUBE and ATELIER are based on fundamentally different design paradigms for interactive systems: While STUDIERSTUBE, with its typical real-time tracking and rendering task for delivering a high quality AR experience can be characterized as being predominantly a stream processing framework, ATELIER is the classic case of an event processing framework, which uses individual events for coordinating the behaviour rather than frequently recurring information items such as tracker position updates.

2.1 The Atelier framework

The IST Project ATELIER (Architecture and Technology for Inspirational Learning Environments, IST-2001-33064 [136]), develops a set of architectures and technologies in support of inspirational learning in two areas - architecture and interaction design. ATELIER is an EU-funded project that is part of the Disappearing Computer Initiative.

> The mission of the initiative is to see how information technology can be diffused into everyday objects and settings, and to see how this can lead to new ways of supporting and enhancing people's lives that go above and beyond what is possible with the computer today. (*from [137]*).

Being part of the "Disappearing Computer Initiative" [137], which encouraged user interfaces that will hide computer interfaces in a manner that "computers disappear", our approach was to implement a number of different tangible interfaces, where, because of the physical aspect of the interaction, the actual processing is naturally happening in the background. Single computer interfaces were also created, the most successful implementations were distributed setups.

It involves partners from Austria, Finland, Italy and Sweden. The project started in 2001 and ended after 30 months in May 2004. The great approval from the EU IST project reviewers encouraged us to continue the work on the project and on the framework that was created.

> The last few years has seen increasing interest in tangible user interfaces. ATELIER is among the global leaders (and perhaps the leader) in developing this kind of work systematically for a useful and usable application. (from the final review of the IST ATELIER project [18])

The aim of the ATELIER project is to build a digitally enhanced environment supporting creative and inspirational learning. The functionalities, described in this thesis, were developed, because of the different work-practices at the two sites. One being an interaction design school in Malmö, Sweden the other an architecture school in Vienna, Austria.

Students in both locations share the need for computational support working on their projects. While the students in Malmö are focusing on *designing* how to interact with a system, the students in Vienna *use* the system to communicate their ideas and concepts (among themselves and to their supervisors). The group of users, we developed the system for, needs both: *means to create a system* and *means to work with a system.*

Nevertheless both groups study design and how to express and model the results of their creativity. Exploring the results and concepts, redefining the goals and finding new solutions. They share the need for content creation, management and presentation (*digital*: images, videos, sounds and *physical*: mock ups, models, prototypes).

We observed how students configured and reconfigured their workspace as well as the relationships between design representations. This motivated a design approach, which focuses on configurability as an important feature of tangible computing environments.

The presence of inspirational resources - images, music, metaphors, atmospheres, film, samples of materials and everyday objects – stimulates the learning process in inspirational learning environments. Inspiration emerges through the particular qualities of objects, people, ambience, and place.

As a result of the initial field trials the project identified particular atmospheric, material and spatial qualities that should be created and supported. These qualities were: the transient and ephemeral, materiality and the diversity of materials and representations, creative density, re-programming and the "different view", experience of dimensionality and scale, forging connections and multiple travels, configuring, tempo and rhythm. The project has contributed to architecture and technology for inspirational learning by development of artefacts captured in the tension between generic design patterns and concrete working demonstrators.

The major demonstrators that were developed are:

- The *Mixed Object Table* with the texture painter for painting computer generated visual overlays as texture on physical models using a physical-digital brush,
- The *Tangible Image Query* and *Ontology* for physical and informal search in a database,
- The *Interactive Stage* combining element of a theatrical space with technological augmentation to manipulate media and events,
- The *E-Diary* for recording and navigating field visits combining positional information with picture material,

The components of these demonstrators were integrated via a shared platform, an independent infrastructure and a hypermedia database.

Weiser's *Ubiquous Computing* vision was that the computers becoming invisible among the objects of everyday life. The same goal has been postulated in the Disappearing Computer Initiative of the European IST project

[137]. In ATELIER – as one project of this call – we tried to create a framework to create such interfaces.

Ubiquitous Computing and *Pervasive Computing* propose an environment that can be controlled by the users in different ways. This includes HTML pages for remote access to content, Pocket or Handheld PCs to control the system and also "tangible interfaces", thus a *multimodal interaction* system.

2.1.1 Infrastructure

The infrastructure software platform in ATELIER environment is a generic platform that allows different kinds of technologies to be added to the environment, thus extending and also changing the functionality of the environment. The platform is open in a sense that different kinds of technologies, display systems and mobile devices (that may in themselves contain advanced functionalities) can be brought into the space, as long as the technologies conform to specified interfaces of the Infrastructure. ATELIER itself has been build with Java, but components or systems built with other technologies can be integrated into the system.

The ATELIER infrastructure acts as a mediator between the ATELIER components that provide the separate functionalities that the components enclose. A component can be a simple piece of technology or a large system by itself. The infrastructure itself contains functionality that is needed across components, and it also provides context for requirements that are not necessarily functional (such as the need for location independence).

2.1.2 Hypermedia Database (HMDB)

The HMDB is an essential element in the whole ATELIER technical environment. It acts as a central database for multimedia content. All images, video files, sound files and any other types of digital documents can be stored in the HMDB. It has an application programming interface to allow Component builders to add, search, browse and modify multimedia data. There are also tools developed in the project for browsing the database by the user. The HMDB is also used for storing other types than multimedia data. Because of the complexity and size of the HMDB, it is implemented as a separate server software.

2.2 The Studierstube framework

A number of target hardware platforms and various input and output components are supported by STUDIERSTUBE. Following interaction patterns have been implemented or integrated and used in various applications: Direct Manipulation of 3D content, multimodal – multi-user interaction, physical handles to digital media, presentation creation and management, GUI widgets, distributed output, and teleconferencing.

The STUDIERSTUBE framework has been redesigned a number of times in the past and a new design is just at the beginning. The constant work of a large group of people that contribute to the functionality of STUDIERSTUBE, has created a complex and versatile system that allows to explore the usage of AR in many application areas. Another strength of the STUDIERSTUBE framework is that a number of other frameworks have been integrated or a way to access the functionality has been implemented (DWARF, PUC, SAPI).

Several course at the university present the STUDIERSTUBE framework and computer science students can attend lab's, where they make use of the STUDIERSTUBE framework to implement *augmented reality* applications. A reduced version of STUDIERSTUBE was developed to present them an easier access to the functionality and reduce the learning effort. At the same time the full scope of STUDIERSTUBE is used to teach software design patterns and explore new concepts both for software and interaction design.

STUDIERSTUBE is a framework based on *Open Inventor* (at the time on *Coin3D* from *Systems in Motion* [134]), which uses OpenGL to render three dimensional scenes. The purpose of STUDIERSTUBE is to create a framework for implementing interactive collaborative Virtual Reality (VR) and Augmented Reality (AR) applications. To achieve this goal, an interaction framework has been implemented. To be able to process the input from many devices and to add a level of abstraction between different devices *OpenTracker* [117] was designed and implemented.

The Open Inventor (OIV) [131] rendering library is the basic software layer upon which STUDIERSTUBE is build. It is a framework of C++ classes that implement a scene graph based rendering library using OpenGL. To be able to create multi-user setups a method of sharing nodes in distributed scene graph were implemented. The Distributed Inventor (DIV [52, 124, 51, 125] module is necessary to create multi computer – multi user applications.

> Collaboration between different users requires distribution of the application's state among different setups. To simplify development of such distributed applications, we have implemented Dis-

> tributed Open Inventor, a powerful object-oriented distributed shared scene graph. Changes to a distributed part of the scene graph are transparently communicated to other instances of the application without exposing this process to the application programmer. Consequently, the development of multi-user applications is not much harder than developing for a single user.

Collaborative applications benefit from the augmentation of the real world, because they allow to naturally interact with a partner while working together on a task. By sharing the physical working space or being connected through a teleconferencing environment the partners can use natural real–world collaboration strategies like pointing, gesturing and explaining issues they are working on.

A large number of areas in computer science are combined in the development of augmented reality systems. Sensor technology is required to measure the state of the real world such as the position and view direction of a user, position and orientation of objects or interaction devices to augment them. Advanced computer graphics are necessary to render convincing images of virtual information. Finally, collaborative applications are implemented using distributed systems to support several users. Therefore, a comprehensive AR system (like STUDIERSTUBE) needs to address all of these aspects and build upon a scalable design that combines the individual areas.

2.2.1 OpenTracker(OIV)

OpenTracker is both a separate project, as well as a part of STUDIERSTUBE. The purpose of *OpenTracker* is to implement an abstraction layer between tracking hardware and tracking data. In general *OpenTracker* is a data flow management system. The generic concept of a data flow graph is used to describe the processing of the data. Various nodes have been implemented that allow to define data sources, data sinks, data filters, and data calculations.

OpenTracker is configured by supplying a XML file that allows the developers to define the processing graph. Network protocols have been implemented that allow to receive and send tracking data over the network. Calculation nodes allow to insert offsets, perform matrix operations and combine static and dynamic data, filter nodes can be defined that smooth tracking errors or decide between different tracking sources based on e.g. the quality of the tracking data.

The data that can be processed by the *OpenTracker* framework has static structure, but a new version of *OpenTracker* will allow to define an arbitrary data structure.

2.2.2 Open Inventor

Since the early developments of computer graphics *scene graphs* have been used to describe a graphical scene. Nodes represent

- objects – that can be rendered
- transformations – to move, scale and transform the objects
- materials – that will be used by the subsequent objects
- separators – that limit the properties like transformations, materials, ..., to a part of the scene graph
- switches – that allow to select between different scene graphs

Nodes can incorporate sub-nodes and fields, where the fields represent values that belong to the nodes like e.g. the centre and radius of a sphere or the transparency value of a material. In OIV these fields can be connected (through field-connections), that will ensure that when ever a field in the source object is changed, the connected field will also change it's value. A software designer can use sensors to detect changes in a field (or node) and react on the change.

Based this framework STUDIERSTUBE was implemented, using nodes and fields to introduce tracking data (by the way of *OpenTracker*) into the scene graph and reacting on changes using sensors.

OIV also contains a concept for scripting (semi-programming interface) that provides means to describe a scene graph and field connections in a text file. When implementing new nodes the definition for the scripting interface is automatically created and therefore does not pose additional implementation effort. APRIL for example is mainly based on the scripting interface, adding only a few general purpose nodes [76].

For a more detailed discussion of these concepts, see the Inventor Toolmaker [152].

2.2.3 Application areas

Following application areas (among others) have be investigated and demonstrators to evaluate the use of AR in these fields have been created.

Tourist Guide. The Outdoor Collaborative Augmented Reality (OCAR) framework based on STUDIERSTUBE was created to support individuals or a group of tourist by supplying them with navigation tools and information about sites in a city [118].

Machine Maintenance. With the AR Puppet framework based on STUDIERSTUBE a virtual character support for machine setup and maintenance was implemented. The virtual character provides guidance and can interact with the user to build or repair a physical device [6].

Construct3D. This application focuses on a learning environment for geometric education. A group of students and a teacher are supported through AR to solve various geometric construction exercises [71].

Video Conference. By adding tangible interaction and realtime 3D visualisation techniques to a standard video conference (video and audio transmission) a powerful remote collaboration tool for medic was created [5].

Virtual Showcase. This is a system targeted at presentations in museums to augment real artefacts and supply additional information. It also includes a tool for creating interactive presentations (APRIL) [77].

To process the input of the users the STUDIERSTUBE framework makes use of the *OpenTracker* framework. With this framework any tracking technology can be used to enable the users to interact in 2D and 3D. Devices supported by *OpenTracker* are: tangible tracked objects (using ARtoolkit), tracking hardware like ARTtracker, acceleration sensors, ... but also speech input and the integration of simple buttons and switches.

Multimodal output of STUDIERSTUBE supports 2D and 3D screen output, sound and speech output, and through the AR Puppet framework also physical device control (e.g. LEGO robot integration).

2.3 Comparing Atelier and Studierstube

STUDIERSTUBE and ATELIER are based on fundamentally different design paradigms for interactive systems. STUDIERSTUBE, is a typical real-time tracking and rendering system, that delivers a high quality AR experience and can be characterized as being mainly a stream processing framework.

In ATELIER event processing is the base for the framework, which uses individual events for coordinating the behaviour of the different components.

This difference is actually very fortunate, because it is complementary. Combining both project will result in a versatile framework that incorporates functionality for *ubiquitous computing* and *augmented reality*, two research fields that have much in common, and at the same time have quite different requirements on the used software framework.

AR needs fast update rates of sensors, displays and complex calculations to present the "illusion" of merging real objects with virtual ones. This goal can only be reached by extensive usage of computer devices and massive computing power. Though the final goal is to hide the computers and just leave the view of the augmented reality. With the development of more powerful, small, and light weighted computers, smaller and higher quality see-through displays, and improvements in the sensoric devices this goal will become reality.

Ubiquitous computing focuses on embedding interaction devices in the environment, also making use of already available output devices (like stereos, television, watches, ...). The development of ubiquitous computing environments struggle more with interaction design problems than with missing (or too big) hardware.

A problem that is shared by both research fields is getting hold of the technology. While consumer products become smaller and more powerful, the technical knowledge and effort to achieve these enhancements, become more and more inaccessible for (interaction) researchers. The special hardware needed for AR systems is targeted at a very small group of researchers, which makes them expensive, and furthermore technical development for this application field is making only marginal progress. Big consumer electronic companies sell HMD's, but these are more focused on providing means to watch TV or VR setups for entertainment (non-see-through), than the for AR needed see-trough displays [89].

Due to the long history of STUDIERSTUBE (nearly 8 years) and the large number of people contributing to the framework, STUDIERSTUBE covers a wide field of use, but also due to the history of the project some structures are inflexible, an issue that is tackled constantly and leads to re-designing of the framework (in two to three year intervals).

In ATELIER we were "fortunate" to only have 30 months time to develop the system. Leading from the first independent prototypes, to a system that incorporates nearly all ideas we wanted to realise. Also – due to the short time – the number of people that contributed to the development was limited, which eases the consistency, as the different work practices of all the STUDIERSTUBE contributors sometimes leads to software design problems.

ATELIER and STUDIERSTUBE can hardly be compared, but each project has unique advantages:

- STUDIERSTUBE is a huge project with an enormous variety of applications, software designs, being open to the public, well documented and tested as API for educational use, and is maintained by competent group of people.

- ATELIER has the advantage of a consistent global storage facility (see *HMDB* 3.2, *p.* 24), the consistent integration of tangible interaction patterns, and a small set of applications. Furthermore ATELIER was developed in conjunction with the users in a participatory design approach, defining a clear and small set of use scenarios.

2.4 Survey of Interaction Design Rules

Designing and implementing interactive systems requires understanding the users of the system [47]. Interaction between the user and the system is actually just a method for communicating with the designer of the system, by the means of using a computer system. The designer defines what and how the work can be done. A lot of principles and rules have been developed and described in the vast field of research that is concentrating on interaction design. Note that interaction design is not only limited to computer systems but also to the real world and objects of daily use [97]. There are also rules and methods to evaluate a particular interaction design, some concentrate on graphical user interfaces (GUI) [9] and some are more abstract and can be applied to any interface.

Many authors have published rule collections that can be applied to verify that an interaction pattern meets certain requirements. Some of these rules have to be adopted to fit a special application area, some of them are so generic that they will always fit. Rules and guidelines provide a good framework for thinking and reflecting about a system, choosing the "right" guidelines however also reflects already the focus of a system.

Some of the rules that we have used and compared our work with are now presented. Please note that a lot more guidelines are available and that each year new guidelines are published.

Interaction frames. Bellotti describes in her article [10] that Goffman [43] developed a notion of frames, that are social constructs (like 'performance' or 'games'), that allow humans to make sense of otherwise incoherent human actions. Explicitly setting a *HCI frame* – for both the system and the user – will ensure that both sides understand the actions.

Communication theory. In the beginning of computer systems the interaction was centred at the capabilities of the machines. With the development of more powerful computer systems the interaction can be centred at the human again. Human to human communication (HHI) can be very complex

and is based on languages that people share. With language every kind of communication is meant like sounds, facial or body gestures. These languages are normally used in conjunction. For human–computer interaction similar languages have to be developed.

As in human to human communication both communication partners have to have an understanding and learn how to interact [41, 42, 44]. When people have to use a language that is not their mother tongue, the partners reduce the complexity until a stable communication is achieved. Often people ask questions to verify they understood each other correctly. All these communication patterns can also be found in human computer interaction.

As humans are more adaptive, one may assume that they are willing to accept restrictions in the complexity of the communication, as long as they are able to verify that they have been understood. In this thesis a system is proposed that makes use of the ability of humans to learn. The goal is to help the users to learn the human computer interface language by enabling them to verify the system understood and giving them insights into the functionality of the system.

A strong statement against automatic adaptation of computer programs was presented by B. Sheiderman [126]. The author proved that spontaneous adaptation of applications is disrupting the work flow. Actually users do not expect that an application changes or adapts the way of communicating with the user. Like all machines a computer is seen as a static object, which should have a static interface that does not change.

2.4.1 Sensible, sensable and desirable

Benford et al. in [12] describes *a framework for designing physical interfaces.* This framework especially concentrates on the physicality of the interface (keyboard and mice also have a physical representation) and on the means of sensing the users input (keyboard – buttons, gesture detection – vision).

- *Sensible* are movements that users normally would perform. These movements are shaped by the physical appearance of the physical interaction objects [97, 98].
- *Sensable* are movements that can be measured by the device that is being used. This category depends on the technical realisation of the sensors used in the device. The accuracy of the sensors also influence what can be sensed by the system.
- *Desirable* are those movements that are wanted by an application. Independent from the used sensors and what is sensible the application

designers envision a particular use of the device.

Benford states that only the design space where all three categories overlap are in general inspected by the designers of a system or device. Yet the design field where only some, one, or none category is covered by the device incorporates opportunities for design. This framework is especial useful when evaluating a device or system that will be used by untrained staff – as trained persons tend to respect the scope of the usage. In a public context however nearly all fields outside the *sensible, sensable and desirable* design space will be explored by the users.

2.4.2 Properties of Instruments

Beaudouin-Lafon discussed in [9] that user interfaces can be evaluated using the terms: degree of indirection, degree of integration, degree of compatibility. Although the original publication focuses on widgets, it can also be adopted for tangible user interfaces.

- *Degree of indirection.* The degree of indirection is a measure of the spatial and temporal offsets generated by an interface. The *spatial offset* is the distance between the input part of the interface and the object it operates on. The *temporal offset* is the time difference between the physical action on the interface and the response of the object.
- *Degree of compatibility.* The degree of compatibility measures the similarity between the physical actions of the users on the interface and the response of the object.
- *Degree of integration.* The degree of integration measures the ratio between the degrees of freedom (DOF) provided by the logical part of the instrument and the DOFs captured by the input device. The term degree of integration was introduced in integral tasks [62].

These terms have a very general character and are especial helpful, when comparing different interfaces for the same or similar interaction design. Beaudouin-Lafon concentrates on GUI interfaces and demonstrates his framework on different widgets, where he proves that the new widgets are "better" – when applying his measure criteria.

2.4.3 Affordances, Feedback, Constraints, Mappings

[15] Norman in his book *The Design of Everydays Things* [97] concentrates on design of devices. Based on his concepts Bowman and Hodges [15] created *User Interface Constraints for Immersive Virtual Environment Applications.* They proposed that the guidelines of Norman [97], *p.* 48 should be applied to *interaction objects* in virtual environments (VE).

- *Affordances.* The object must inform the user of the way it can be used, by visual or other clues (like handles) the user should immediately understand the interaction pattern that should be used with the object.
- *Feedback.* The object must supply feedback when it is used, displaying the change in the state of the object (e.g. like highlighting, changing visual appearance ...).
- *Constraints.* This refers to the limitations on the use of the object that supports the user to use it in a proper way (e.g. like a slider that can only be moved along one axis).
- *Good Mappings.* The conceptual model, on which an object is based, should be easy to understand. Setting the frame (see [10], *p.* 416) or situation, in which an object is being used, plays and important role in building a conceptual model for the users. If the user does not understand the specific usage model, then the interaction with the object will not lead to the expected results.

Norman stated those requirements should be satisfied by physical devices. In a ubiquitous environment, although the computational background should be hidden, these properties ensure that the users will understand that they are actually using a *device.* So the way the device has to be used must be visible, while hiding the complex system in the background.

2.4.4 Eight Golden Rules of Interface Design

Shneiderman [126], *p.* 67–78 proposes 3 principles: *Recognise the Diversity* referring to the diversity of users (novice, beginner, intermediate, expert) and the different work practices of the users (command language, visual focused). *Use the Eight Golden Rules of Interaction Design* and *Prevent Errors* the later meaning that meaningful messages should be provided (not *"SYNTAX ERROR"* for every error etc.) and designs that in itself prohibit users to make errors (by sketching a very rigid approach), or at least help them to avoid errors.

The Eight Golden Rules of Interface Design [126], *p.* 74–75

1. *Strife for consistency.* The actions, the interfaces and the terminology should be kept consistent, while allowing a limited number of exceptions for specific situations (e.g. asking for confirmation when a delete command is issued)

2. *Enable frequent users to use shortcuts.* To allow a beginning user to advance to be an expert, shortcuts and macro recording functionality should be provided.

3. *Offer informative feedback. "For each user action, there should be a system feedback.".* This rule actually enforces the notion of "interaction" as the action–reaction pattern is stressed.

4. *Design dialogs to yield closure.* With "dialogs" general user – system dialogs are meant. If an action requires a number of steps (e.g. saving and specifying a filename) these should be designed to have a *beginning, middle, and end* to

   > "give operators the satisfaction of accomplishment, a sense of relief, ... and an indication that the way is clear to prepare for the next group of actions".

5. *Offer error prevention and simple error handling.* The system should prevent errors, by only allowing valid entry of data (e.g. only digits for numerical fields) *"If users make an error, the system should detect the error and offer simple, constructive, and specific instructions for recovery."*

6. *Permit easy reversal of actions.* This feature relieves anxiety and therefore helps the users to explore the possibilities of the system without the fear of making a irreversible error. This rule enforces the *try and error* principle. A discussion about this specific rule is presented in the section (see *Undo Qualities* 6.3, *p.* 110).

7. *Support internal locus of control.* Gaines [40] described this with his rule *avoid acausality* and his statement that users should be the *initiators* of actions rather than respond to actions. Users should always have the feeling that they are *in control* and are able to produce the action they desire.

8. *Reduce short-term memory load.* Provide easy access to help, and explanations of how tasks can be accomplished, provide clear and consolidated interfaces that limit the information processing needed. Interfaces should be easily perceivable and provide clear groupings.

These explicit rules create a framework to discuss about different solutions and also help to think about usability when it comes to interface design. The rules – as they are described in [126] – focus on traditional (if the last 10-15 years already provide something like a tradition) user interfaces, hence the single user workstation with display, keyboard, and mouse.

When new interface devices should be developed, questions that seem to be solved (at least in GUIs) pop up again.

2.4.5 Five Questions for Designers and Researchers

Bellotti et al. in [10] provides in the paper "Making Sense of Sensing Systems" the *Five Questions for Designers and Researchers.* She compares the solutions in different *genres*, meaning *a set of design conventions anticipating particular usage contexts with their own conventions*, by asking these questions and also providing solutions from the GUI genre (and simpler genres like cell–phones, microwaves, ...).

Five Questions for Designers and Researchers [10]

1. *When I address a system, how does it know I am addressing it?*
2. *When I ask a system to do something how do I know it is attending?*
3. *When I issue a command (such as save, execute or delete), how does the system know what it relates to?*
4. *How do I know the system understands my command and is correctly executing my indented action?*
5. *How do I recover from mistakes?*

These five questions raise following issues, which are based on Norman's "seven stages of execution" [97], *p.* 45–48, but are more focused on *communication* than on *cognition.* Those basic issues expose challenges that are solved in GUI interaction, but have to be solved in a ubiquitous environment:

- *Address.* How to filter the signal from the input stream (signal–to–noise), how can the user avoid to address the system?

- *Attention.* What feedback is presented to display that the system is ready for input, and how is this feedback directed to the zone of user attention?
- *Action.* How to identify and select objects, how to avoid unwanted selections, how to perform complex operations (multiple objects, save, selective save)?
- *Alignment.* How to make the system state perceivable, persistent or queryable? How to provide timely feedback and how is this feedback presented?
- *Accident.* How to interrupt a system action, how to undo an action, how to intervene when a user made an obvious error?

These issues where addressed by us in both systems, and are described in the next chapters. In STUDIERSTUBE – through the direct manipulation method and the use of PIP and GUI like widgets – the solution to these issues is easier than in ATELIER, where the ubiquitous character of the interfaces raise these problems.

2.4.6 Error handling

Stressing the issues *error handling* and *error recovery* in 2.4.4 and 2.4.5 is based on the strong statement of Norman [97], *p.* 105–140 that people make (a lot of) mistakes, where some of the may be hard to detect (even for themselves).

Some computer systems lack detection of errors, handling of errors, or providing means of error correction. As Norman describes it *"When someone makes an error, there usually is a good reason for it."* [97], *p.* 131.

Also an error should not be replied on "harshly" by the system, but with help how to proceed and correct what went wrong. This is only true for errors that can be detected by the system – or more correctly: have been foreseen by the designer of the system. Sometime like issuing a "print" command, was just a slip of the user. Undoing or interrupting an ongoing action is also a sort of "error handling".

Chapter 3

Setting the Frame: Applications

Each application will be shortly introduced, describing the setting, how the applications are being used and sketching the concepts behind each application. In the following chapters the related aspects of *distribution and configurability* and *multimodal interaction* are being discussed referring to these applications. Therefore this chapter is important to get an understanding for the applications.

First the basic elements of the ATELIER will be described, then the applications will be presented.

3.1 Infrastructure

The infrastructure software platform in ATELIER project is a generic platform that allows different kinds of technologies to be integrated to the environment. This allows to extend and also change the functionality of the environment. The platform supports to bring different kinds of technologies, display systems and mobile devices into the space. Each of these components most conform to the specified interfaces of the Infrastructure. Although the Infrastructure itself is implemented in Java, components and systems built with other technologies can be integrated into the system.

The ATELIER infrastructure acts as a mediator between the ATELIER components, a component can be simple or a large system by itself. For example, an infrared remote controller device, such as a TV remote controller, with associated component software can be used as a component in the system to control any other component or system in the environment. Components themselves can be combined into applications, larger wholes of functional entities.

The infrastructure contains functionality that is needed across compo-

nents, and it also provides context for requirements that are not necessarily functional (such as the need for location independence). The infrastructure itself is based on *Microkernel software architecture* pattern, and can be expanded by providing new internal or external services. The services are then available to all components, that are connected to the ATELIER environment. Examples of ATELIER external services are the *Hypermedia Database* service – for storing hyperlinked multimedia information – and the *Email Entrance* service – for entering new media into the hypermedia database service by sending e-mail attachments from any kind of interned enabled device.

The main advantage of the infrastructure is flexibility and configurability; it is possible, for example, to replace a positioning (tagging) technology, display or a mobile device with another kind, without losing the interoperability of the ATELIER environment. This is feasible as long as the new technology is able to communicate with the ATELIER environment using Internet technologies and XML messages. If the technology per se does not have this ability, it is possible to write adapters to enable the connectivity. This architecture thus allows us to build more than just one implementation usable in a specific context, but an environment that is reconfigurable and also extensible in future experiments utilizing different technologies.

Because of the requirements for flexibility and extensibility, the specifications and design of the ATELIER Infrastructure software has been based on the principles of expandability and abstract interfaces. The use of messaging enables in this goal, as system elements communicate by sending XML messages, that are routed by infrastructure Kernel. This mechanism ensures that the elements are efficiently isolated from each other.

3.2 Hypermedia Database (HMDB)

The HMDB acts as a central database for multimedia content like images, video files, sound files. Any other type of digital documents can also be stored in the HMDB. A application programming interface allows Component builders to add, search, browse, and modify the data stored in the database. Components for browsing the database by the users were also implemented. Because of the complexity and size of the HMDB, it is implemented as a separate server software.

There is also the possibility of creating hierarchies and add meta-information to each element in the database. The meta-information is a list of key and value strings that can be used in different ways. There are also methods for searching the database for specific elements (using the meta-information) and retrieving whole parts of the hierarchy. Two generic interfaces have been

developed to edit and manage the content of the database. The *Path Creator* is a part of the E-Diary setup and a HTML-Access, which enables the users to retrieve information and modify the database from any computer equipped with an internet browser.

3.3 Configurator

The Configurator is actually not an application, but some sort of meta control mechanism for the ATELIER project. As described in the next section 3.3.1 and 3.3.2, the input and output of components of the ATELIER platform provide abstracted representations of the input and output possibilities. The Configurator allows the users to combine these components into a meaningful system. Here the Configurator, as it was implemented in the ATELIER project is described, a more advanced version will be presented in the chapter *Configurability* (4.9, *p.* 73) that incorporates the experience we have gained in studying the users and their problems with the Configurator.

As stated above the main task of the Configurator is to supply the students with the means to configure the ATELIER environment (see *Configurability* 4.9, *p.* 73). Based on the experience we gained from the first experiments at the Academy of Fine Arts, we developed a system that allows the students to connect different functionalities of the ATELIER project and also presents them a consistent interface to the ATELIER environment.

3.3.1 Input devices

In the ATELIER project we incorporated support for a number of input devices. Due to the commitment of the *Disappearing Computer Initiative* we focused on tangible interaction devices. Nevertheless also other input devices are supported.

RFID tags are supported to allow the students to issue commands and also to access content from the HMDB (see 3.2).

Barcodes can be read with a barcode reader and are also used to issue commands and to access content.

Various Sensors are uses to detect gestures and – based on the gesture – perform an action. The *Sensitive Sample* is a development of the ATELIER team, while the *SoapBox* is developed by VTT Electronics [142, 30, 147].

IR Remote Control (infrared remote control) is a well known interface to a number of devices. Using a simple IR receiver connected to a PC the buttons of the remote control are used as input component for the ATELIER environment.

HTML access grants access to the HMDB by the means of a web interface. Any workstation that is connected to the internet and has a web browser can be used to add, view and manipulate data of the HMDB.

Tracking of various markers (ARTtoolkit and retro reflective markers) is being used in the ATELIER project. The tracking of the retro reflective markers was realised using the DynaSight Sensor from Origin Instruments [102] and GPS to record the position of e.g. photos.

A complex "input device" was developed by our Finish partners: the E-Diary [55]. A short description will now be presented.

The E-Diary. The typical activities of a student includes visits to some location at the beginning of a project. She takes a lot of digital pictures, captures outdoor ambient sound, records video clips and writes some notes on the site. Once the student gets back in the ATELIER space the data gathered has to be brought into the system, and the student can start to explore the data, combine them with the items already present in the system, and work on the concepts. Various ATELIER components support the student at each different stage of the work.

Figure 3.1: The E-Diary in action

The simplest way to collect data is to take a digital camera, go to the site of interest and take photos. The students usually have to write notes for each photo taken, stating at least where and when the photo was taken. In order to make this cumbersome task of matching pictures and notes easier, the E-Diary application has been developed. The E-Diary application runs on a personal digital assistant that is equipped with a GPS card and a compass, using the *SoapBox* (see figure 3.1). The position and orientation of the user during the whole trip is recorded on the PDA using the attached sensors. Additionally the students can record sounds or take notes that will also be stored on the PDA. Once the student gets back in the ATELIER environment, the GPS data, time and orientation will be automatically assigned to each photo or note taken, by matching timestamps of the recorded data and the photos taken.

3.3.2 Presentation devices in ATELIER

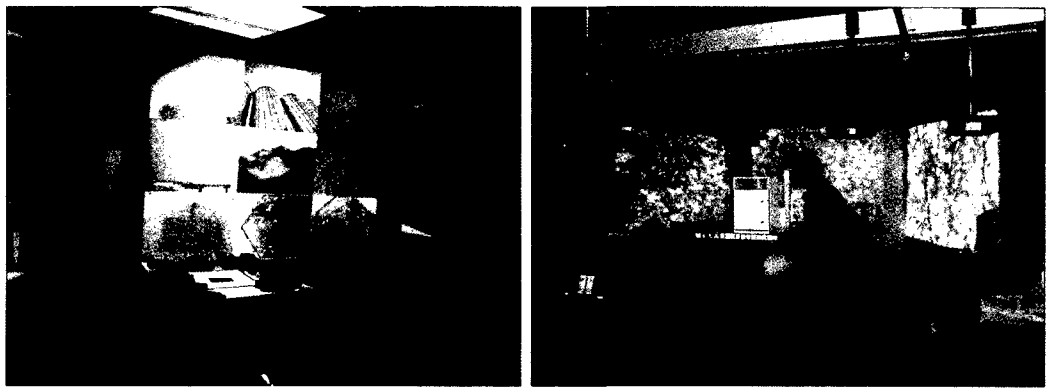

Figure 3.2: The three large displays

We developed a series of display components that can be used to present digital material that are stored in the HMDB.

In the ATELIER room we provide several large screens to project on (i.e. a three-screens wall), where the display planes can be adjusted to form a closed space or a flat wide screen or anything between those extremes (see figure 3.2). To reach a higher degree of immersion in our environment also sliding projection screens that can be shifted along the room and be placed with different angles were introduced. A rail grid was installed on the ceiling of the room at the Academy of Fine Arts, these screens are hanging from the grid and can easily be moved back and forth on the grid. Creating an immersive environment is important both for learning and presenting.

Another relevant factor is the easiness to configure such an environment and the possibility to have different, flexible arrangements not only of the global projection surface (multiple screens) but also of the inner parts of each screen.

We provide the users with physical handles, that are paper sheets, or posters, to control the inner layout of single projecting panels (this facility will be discussed later in more detail). For example each of the three screens in figure 3.2 may be used as a single canvas or split into up to a number of smaller regions for presenting search results (described in *DisplayManager* 3.3.5, *p.* 32).

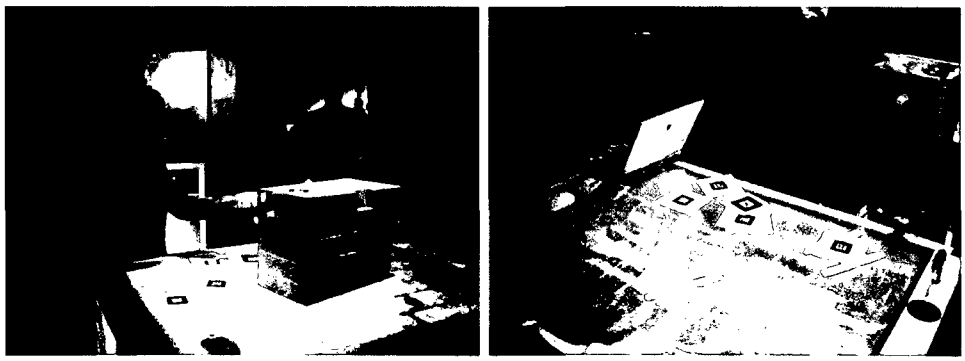

Figure 3.3: The table

Additionally there is a table on which the designer can place their artefacts. On the surface an image can be projected from underneath, changing the visual appearance of the table surface (see Figure 3.3). The Texture Painter (see figure 3.16 and the following section) is used to create compositions of several images and textures to augment artefacts. Having large and various displays allows exploring all possibilities given by workspace and material, moreover it allows great creativity and freedom for presentations.

We also included a fan that could be activated by the students and a fog generator. The fog generator was only used once, but enhanced the atmosphere of the presentation (it was about a football game, where the audience sometimes – though it is forbidden – throws fireworks on the playground), see figure 3.4 .

A simple application that could play audio files was included, we experimented with positioned sound (using 4 loudspeakers), but we soon found out that the students had to concentrated on the visual content, and – although all of them found that sound is an important quality of architecture – just a few of the actually had the time to investigate and make use of sound medias.

Additionally printers were used as output devices, paper is still the material with the most appealing affordances in a learning environment. Apart

Figure 3.4: The wind and fog output

from producing sketches and images printouts, printers were used to print pages that include both: an image and its barcode, which is the physical handle to access the picture (see figure 3.7). Moreover printers themselves can be accessed using physical handles, removing thus the necessity of accessing a personal computer for printing.

3.3.3 Handling media content

We already stated that the main users of the ATELIER environment are Information Technology (IT) novices and this factor deeply influenced our design efforts. We also aimed to build a disappearing computer environment so it is undesirable to populate the room with numerous computer work places that can only be used with keyboards and mice. Our students do not know how the HMDB works, and actually they do not have to. They have to concentrate on their primary goal, learning of architecture and design. In order to facilitate content manipulation, we provide a physical handle (e.g. a barcode) for each added image. The students continually collect content for their project.

Adding content to the HMDB. Once the data is gathered the next step is to bring it into the system so that it can be accessed and manipulated.

Data can be added to the database in several ways. In case a digital camera is used without the E-Diary application, then the fastest method is to connect the digital camera to a computer, which runs the Entrance component: an application that adds all pictures in the camera memory card automatically to a new collection. Such a collection is then available to all the members of the workgroup. Otherwise, if the E-Diary application is used, then a component called *Path Creator* is the most suitable way of adding the data to the system.

The Path Creator downloads the data both from the camera and from the PDA. Additional information is then assigned to each photo (using the time-stamp of the photo and that of the GPS and orientation data) and stored in the system. Furthermore, the Path Creator makes it possible to use a map of the visited area. The path travelled during the site visit is depicted on the map (see Figure 3.5) and locations where photos were taken are marked by red dots. Using a HTML access it is possible, to retrieve the map again and click on the red dots to browse all the pictures taken on that particular spot.

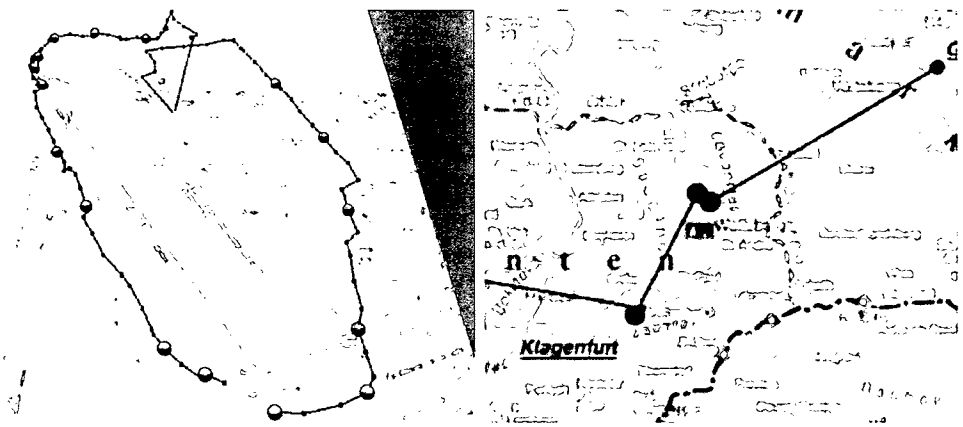

Figure 3.5: The Path Creator

Remote access to the system, to be used for example in case one of the members is far from the ATELIER room and there is the urgent need to show something to other workgroup members, is also available. By sending images to a particular email address, pictures are automatically added to the HMDB and meta-information is extracted from the email message body.

We also implemented a HTML upload component that can be used with any (JAVA enabled) web browser to add content to the HMDB (see figure 3.6).

After uploading images to the HMDB in any of before mentioned ways thumbnail pages are printed out (see Figure 3.7). For each "upload session" the users get pages with thumbnails of each picture they added, with a bar-code printed underneath each of them. These barcodes may be used as a starting point for all actions in the ATELIER environment.

Physical Handles to digital media. Barcodes are easy to produce and inherit all affordances of paper they are printed on (i.e. lightness, flexibility,...). Nevertheless barcodes have some downsides as well (such as their fragility and visual intrusiveness). As a complementary way for physically

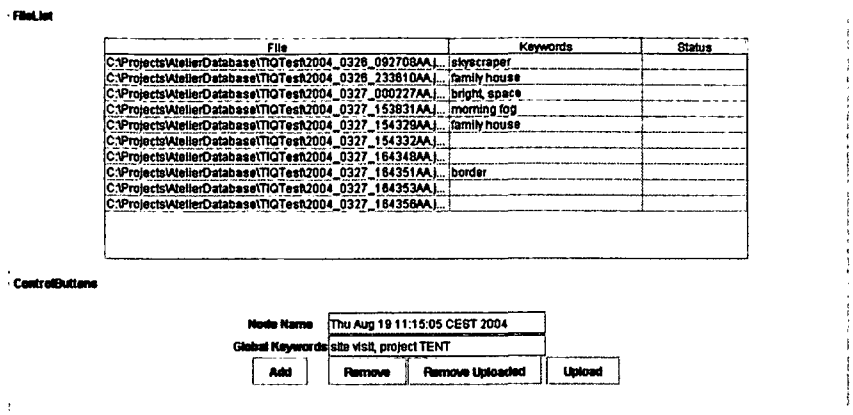

Figure 3.6: The HTML Upload applet

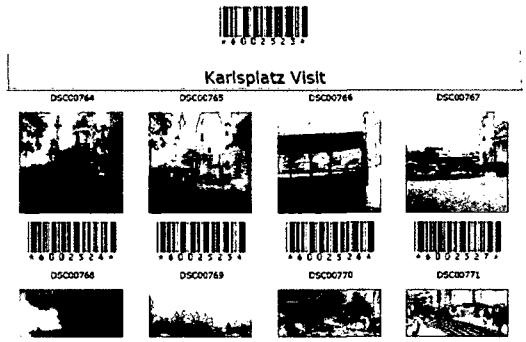

Figure 3.7: The thumbnail page

handling data and control devices, RFID technology is also widely used in the ATELIER project. Barcodes showed to be more suitable to be applied for augmenting large scale, heavyweight objects or fixed ones, such as three dimensional models or wall mounted enriched maps. RFID technology is used to create three dimensional controllers that are more robust. Moreover, thanks to the possibility of placing RFID tags inside other objects, it is possible to explore physical dimensions of objects and to investigate the properties of object compositions. Additionally RFID tags are used to analyse gestures and actions performed in the learning environment. Using either barcodes or RFID tags we are able to identify both objects in the HMDB and actions performed in the environment.

Once images are stored in the system HMDB, and physical handles (e.g. barcodes) for each image are available to students, we explored suitable ways of exploring, accessing and displaying contents. In the ATELIER environment there are many devices that are enabled to present multimedia content and

several ways of browsing the database. We aimed at building up a configurable space in the sense that those devices could easily be put together according to user needs.

3.3.4 HMDBLookup

This component of the ATELIER system is used to convert inputs into references to the HMDB, serving as an abstraction layer for the different input components that were included in the project.

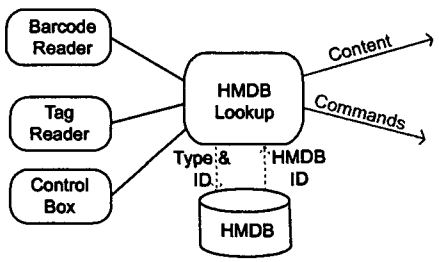

Figure 3.8: HMDB Lookup Schema

As the name states it simply performs an lookup to the HMDB searching for content that is associated to the input. Based on the type of the input and the ID (barcode, tag ID, action performed with cube, ...) an element from the HMDB is retrieved. Based on the meta-information of this element a command or content message is send. The content can be any digital content, a configuration setting, or a saved session. In case that it is a saved session, the content will be sent to the component that stored the information in the HMDB.

Due to the introduction of this component, the method for accessing the mentioned content is not bound to a specific input device. This component forwards an abstract information to the DisplayManager, that will (based on the state of the system) react accordingly.

3.3.5 DisplayManager

The DisplayManager serves as an abstraction layer for output devices. It has a state that defines the "active" display and also controls the layout of the (layout-able) displays. When a content should be displayed the identification

ID of that content is send to the DisplayManager. Based on the configuration, the ID (or a component specific content ID) will be forwarded to the specific output component. Please note that this includes not only images and movies, but also sound and haptic output.

This component manages displaying of results of the *Ontology Search* (3.7, *p.* 39) and *Tangible Image Query* (3.6, *p.* 37), and the cross–search feature, described in *Combining the Search Methods* (6.4.4, *p.* 114).

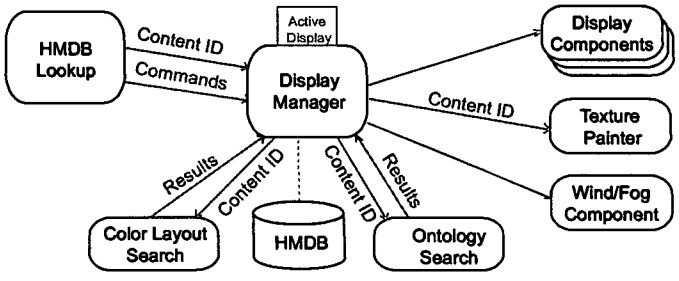

Figure 3.9: DisplayManager Schema

It also serves as a FeedbackManager that can display feedback on any display. Depending on the context the users are able to define the display that should be used for the feedback – eventually use sound as a feedback method.

Due to the combination of the HMDBLookup and the DisplayManager the system is highly flexible and allows easy integration of new input and output components.

3.4 Web (HTML) Interface

The HTML access was implemented as a set of Java-Servlets, which create the HTML documents that represent the interface to the HMDB. It can be used to add and manage content in the HMDB, to change keywords or to extend the ontology by adding new keywords. It was mainly used as a tool to print out the physical handles (barcode thumbnails), that were then used in the ATELIER environment.

The main advantage of the HTML access is that it is independent of the JAVA runtime environment (and some libraries like the Java Media Framework - JMF). The only software needed to use the HTML access is an internet browser and the right URL. This is the main difference to the described

Path Creator that must be installed on each machine, where this component should be used.

This component will be discussed in more detail in the section *HTML as a User Interface* (5.1, *p.* 83).

3.5 Sensitive Samples

The *Sensitive Sample* concept combines a hardware solution and some prototypes that are being presented. The main goal of *Sensitive Sample* is to provide means for the students to include their physical models in their presentations, making them interactive. The German word "*begreifbar*" means both "touchable" and "understandable", which expresses exactly what we aimed for, helping the students to better explain the physical aspects of their concept.

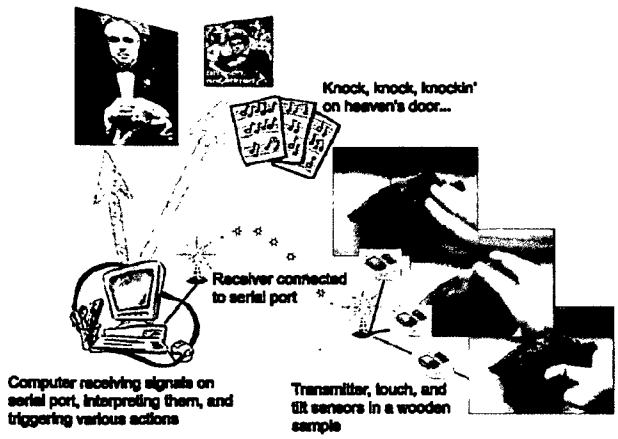

Figure 3.10: A typical configuration of the system using sensitive samples.

The main idea of the *Sensitive Sample* is to make common artefacts, found in architecture studios or the office, become part of a computer interface. In order to fulfil this requirement, the electronic device has to to be small enough to fit inside objects of various sizes. The electronic device should also support a wide range of sensors and simultaneous use of more than one device. The communication with the computer system has to be wireless, and in our case the device had to be inexpensive even in small production runs. Once the device is installed the artefacts can be used as input devices. Figure 3.10 illustrates a typical configuration.

A *Sensitive Sample* is any object enriched with various sensors, that can sense the environment. Sensors are completely hidden inside the object, and the actual sensitive artefacts do not appear to be different from their non sensitive counterparts as shown in figure 3.11. We have experimented with various samples ranging from a simple paper cube to architecture models. The use of Sensitive Samples instead of traditional user interfaces was a new experience for the architecture students. We have developed an electronic device which supports various sensors. Tilt switches and touch sensors were mostly used. The idea of making a small device, which can be incorporated into real objects in order to make them a part of interface is not new. The SoapBox [142], SmartIts [53, 128], and Navigational Blocks [22] are some well known examples. The Sensitive Sample device is a low-cost alternative, which supports arbitrary sensors and multiple samples simultaneously.

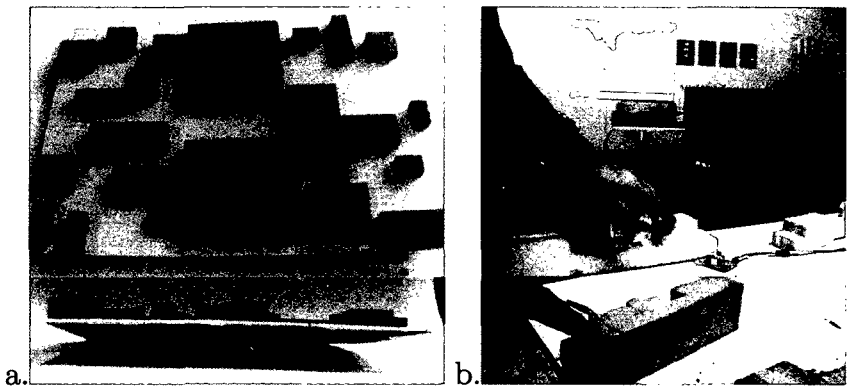

Figure 3.11: a) Concept model with touch–sensors mounted underneath, b) a model made of cement, with touch sensors embedded in the cement

Programmable objects were also deployed by Resnick et. al. [119] in student and school classes to allow them to explore more complex concepts like e.g. context sensitive processing. The difference to the *Sensitive Sample* is that the *Toys to Think With* are deployed in casings (meaning finished objects), that can not be changed by the users. The actual target of the *Toys to Think With* is enhancing the building blocks in kindergarten, by letting the students explore different combinations of software – like children explore the different configurations of building blocks. On the other hand the *Sensitive Sample* should be hidden inside an object – that has to be provided by the students – and enhance it, they are not programmable, but configurable (see (see *Configurability* 4.9, *p.* 73)).

Three example applications, were some details of which will be described later are:

- the ControlCube – a device to control the ATELIER environment and select states
- the NavigationBox – a device to navigate in the HMDB
- the MaterialKammer – a concept for experiencing qualities of different materials

ControlCube. The ControlCube is a simple cube, which can detect which side of the cube is facing up. This device can be used to switch between different configurations and modes in the ATELIER environment. Figure 3.12 shows an example of a ControlCube created by the students.

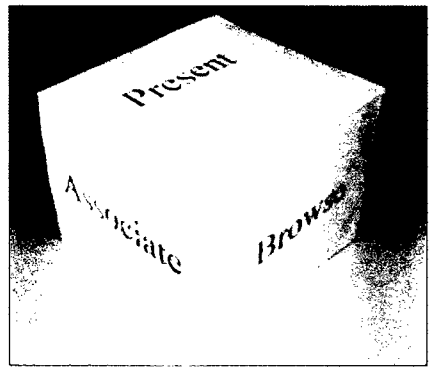

Figure 3.12: The ControlCube made by the students.

NavigationBox. As an alternative method for navigation control, the NavigationBox was developed. Sometimes students wanted to browse through their images, sounds, and videos and then select a certain image. For such a purpose we have build a box with two touch sensors on one side. If the side with the sensors is facing up, the user can browse through the content by touching the places which correspond to previous and next. Once the user has found the content she wanted, the NavigationBox is turned up-side down. With this gesture the media is send to an other applications in the system (see *Configurator* 4.9.2, *p.* 79).

MaterialKammer. Since materials play a crucial role in architecture, it is very important for the students to explore and to get familiar with various materials and their qualities. MaterialKammer is a room filled with material samples, which is envisioned to be an inspiring and learning environment.

The material samples are of various sizes and shapes. A concrete block has to be big and heavy, it's not something the user can take in her hands to play with it. On the other hand a brick can be easily picked up and explored using different manners. All samples are sensitive, and can sense various user actions. Based on these actions the whole environment will be changed, using all the output possibilities available, to create an atmosphere, that provides information and inspiration for the students (see *MaterialKammer concept* 5.4.2, *p.* 93).

3.6 Tangible Image Query

In ATELIER our students had to handle a large number of images. Conventional ways of data retrieval became just insufficient for large amounts of visual material. Popular thumbnail views are useless, if we have thousands or tens of thousands of images.

Content based image retrieval, which has been a subject of extensive research in the last decade, tries to offer a solution for retrieving images from large databases. The original and still often used idea is the query by example method. This means that the user supplies an image, and the system tries to find similar images. In this case the central problem is the definition of similarity. As humans themselves can not always agree on what is similar and what is not (and also what is *more* similar), the results of image retrieval are often unexpected and sometimes disappointing. Figure 3.13 shows an example where such a system was used to search for images similar to the bird image. If the user understands that the system tries to find images with similar colour layout, and not content (bird in this case), results are much more satisfactory. On the other hand if the user expects birds she might be really disappointed.

The next step in image retrieval was not to search only for overall similarity, but rather to find images containing a specific pattern. A company logo is a good example. Imagine a company searching for images containing their logo. The logo can be anywhere in the image, it can be taken under various lighting conditions, it can be distorted due to the perspective projection and so on. Clearly this is not a trivial task. Furthermore, if one tries to find all images containing a bird, for example, the whole search becomes practically impossible – when using a query by example approach.

There are numerous systems capable of various kinds of image queries available. IBM's QBIC System [104] was one of the first systems, and it can be tested online at [59, 114]. The VIR Image engine [46] from Virage, Inc. and the Photobook Project [103] developed in the MIT Media Lab are

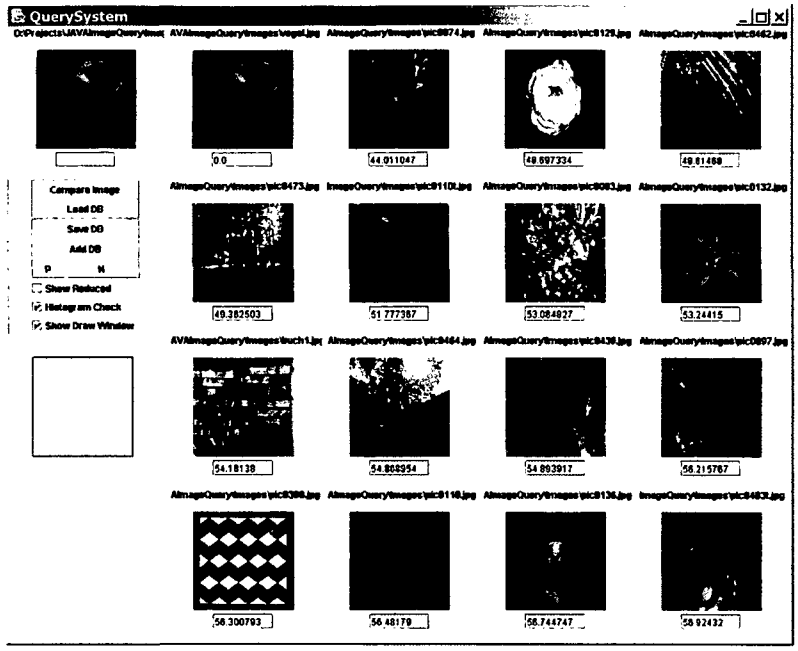

Figure 3.13: Query by Example can be disappointing if the user does not understand the underlying algorithm. Here the system searches for a similar colour-layout, and not for birds. The number underneath the pictures states the calculated difference to the image provided.

two also well known examples. The work of Jacobs at al. [63] is especially well known in the computer graphics community. All of these as well as [33, 73, 145], represent the query by example approach. There are systems like Blobworld [11, 24] or ICONA [14, 34] which represent another group of systems, they go beyond simple query by example, and try to find similar shapes, textures, and other features.

Some systems offer a possibility for the user to sketch the query image. A user might remember how the image looked like (but cannot remember the image's name), so the user sketches the image, and the system finds matching images from the database. Another possible scenario of use comes from the designers' and architects' perspective. In the concept design phase of a project it is common practice to browse through image collections in order to be inspired, to see some unexpected connection between images. Visual image query can be used for such a browsing. The drawback of the method described above (see Figure 3.13) suddenly becomes an advantage. Asking for a parrot, and getting a flower can be either: frustrating or inspiring, depending on the user and the context.

This project is based on such a system, and a new kind of user interface

Figure 3.14: The Tangible Image Query

for sketching images is introduced. Instead of using a mouse to draw, users are provided with small cubes of various sizes and colours, and they try to sketch an image using the colour cubes. Cubes are placed on a semitransparent glass surface. Besides the cubes, users may use any coloured objects. This kind of "sketching" using currently available artefacts is particularly common among designers and architects. We implemented the method, built a prototype and tested it with users (see figure 3.14). We will compare the results with conventional sketching using a mouse (see *Image Query Mouse Interface* 5.2.1, *p.* 86).

3.7 Ontology Integration

To lead the learning process and facilitate conceptualization on one hand and indexing material on the other, an ontology was provided to be used by the students. An ontology can be roughly defined as a hierarchically organized set of keywords that can be applied for giving descriptions of items in the HMDB. The hierarchy captures different kinds of relation among terms (e.g. the *part-of* or the *is-a* relations). We built several facilities around the ontology [21] and the way the ontology is dynamically built and maintained is described in [78]. The most important thing to be noticed is the way the ontology is intended to be used in our environment. While students work is in progress, teachers aim to show them unconventional ways for seeing the materials they are working with, in order to open their minds to new ideas and conceptualizations.

It is therefore not useful to provide some kind of exhaustive ontology in advance and to force students to stick upon it. They usually start from a very short list of keywords that are rather uncommon and through them they give non-trivial descriptions of collected items. Ontology is thus enriched and hier-

Figure 3.15: Ontology used by Sophie at a presentation. She used printed pictures and her own keywords, sorted into a drawer.
(used with permission.)

archy is modified during students projects according to their understandings and agreements. In order to make this refinement process easier to handle it is necessary to provide also an easy to understand interface to maintain the ontology. We initially created an Ontology Interface Component that is a personal computer GUI devoted to these tasks (i.e. adding new keywords, re-arranging the hierarchy, modifying image descriptions...). We will describe later in more details some other ways, we experimented with, for applying keyword-based descriptions to multimedia contents without the need of such an interface (see *Multiple Ways of manipulating keywords in Atelier* 6.4.4, *p.* 115).

3.8 Texture Painter

The Texture Painter is an example of a non-trivial device for showing pictures on multiple projection planes. It basically consists of a projector, a camera and a brush. A software component tracks the movements of the brush in front of the camera and this information is used to virtually paint a physical model in front of the projector using digital textures, which can be any digital picture or even video clips.

While painting, a texture palette, with ten different textures, is displayed and users choose one of them by placing the brush upon the selected slot in the palette. Functionalities for altering textures (rotating, scaling and moving) are also available to extend graphical possibilities (see figure 3.16).

The Texture Painter can be used by the *DisplayManager* (3.3.5, *p.* 32) as an output device, in this way it can be directly controlled using physical objects: for example textures can be switched by using the described handles to digital media (see figure 3.17).

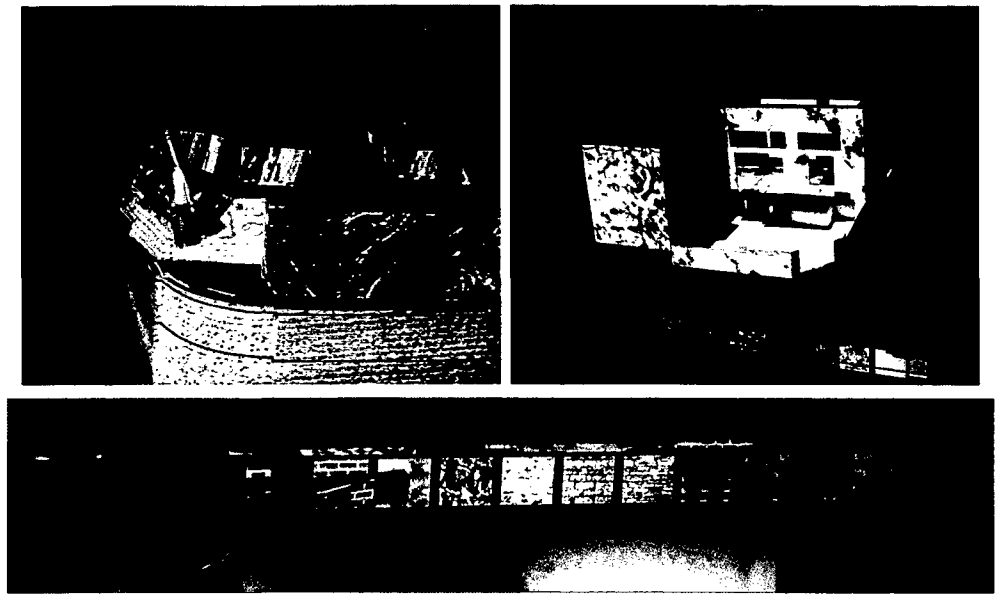

Figure 3.16: Left: the Texture Painter in action, Right: double Texture Painter setup. Bottom: the texture palette

When an image is sent to one of the Texture Painters the currently active texture is replaced by the new image. In this way the part of the artefact, that was painted with that texture, changes its look, creating a new vision of the artefact. Using the Texture Painter mixed objects, meaning objects that inherit properties and affordances both from their physical and digital components, are created. Properties of physical models are deeply investigated and architecture students are moreover allowed to experiment and to test different possibilities before realizing final (coloured, textured) models of objects they are working on. Actually in some experiments students requested two Texture Painters to project both on the vertical and horizontal

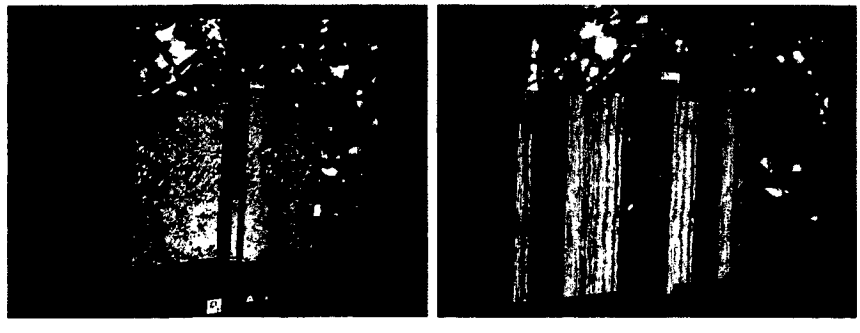

Figure 3.17: Exchanging the material of a model.

faces of their artefacts.

3.9 Invisible Person

Based on body movement and posture an advanced user interface was developed enabling even playing games in an immersive virtual environment. The users presence in the environment, movements, and body postures are the available tools for interaction. Furthermore, a publicly accessible installation in the Vienna Museum of Technology implementing such an advanced environment was created. In this installation computers are completely hidden, and it is one of the most popular exhibits in the museum, which has been accessed by more than 200,000 visitors since September 1999. The new "Invisible Person" interactive installation, which is on exhibit in the Technisches Museum Wien (TMW , Vienna Museum of Technology) is an interactive, immersive virtual environment installation. The system is placed in a publicly accessible place, where a lot of users access the system. Therefore high demands on stability and robustness are required than those required for a lab prototype. The installation consists of a stage, with a sizeable (2.5 x 3.8 m) display on one side. Figure 3.18 shows the actual installation in the TMW.

Figure 3.18: Pictures of the installation displaying the working scheme of the virtual mirror. Displaying the IP and the users

A camera is placed above the display, and the video of the stage is shown in real time on the screen, creating some sort of mirror effect. The mirror image does not only show the mirrored stage, but includes "The Invisible Person" living in the mirror (it is "invisible" only in the real world). "The Invisible Person" (IP) tries to interact with the users in real time. The only interaction tools available for both are their movements and full body postures. No facial gestures, hand gestures or use of fingers are traced. This makes it simpler for the user and also lessens the computational demand on

the system. Actually, this setup has been in the museum since September 1999, when the original IP installation was set up. Petta et al. [105] described the original installation. It was up and running from September 1999 to June 2001. The original installation was based on the *ALIVE* system [27] from the MIT Media Labs. The ALIVE uses a virtual mirror metaphor and an additional virtual character, which looks like a dog and interacts with user. The users can play with that virtual dog by throwing virtual sticks that are brought back, and additional "doggy" behaviour was implemented.

For the original installation in the TMW a child-like virtual character was designed. An artificial intelligence (AI) module controlled the behaviour of the virtual character forming IP. The IP installation allows real-time interaction, the visitor's movements are displayed immediately and interaction with the IP is direct and instant. This real-time requirement puts hard constraints on the realization of the system and even more on the design of the AI that has to react within just a moment.

An important feature of all characters living in virtual environments is their "intelligence". There is a lot of research done on modelling emotional and intelligent virtual characters populating virtual environments. Petta et al. [105], ALIVE system [27], and Tu and Terzopoulos [141], describe the emotional model of virtual characters.

Body movements were used to create an advanced interaction schema, which makes it possible for users to step beyond simple interaction, and even play games with the IP. In order to give a brief insight into the complexity of such an interactive real-time immersive virtual environment installation the actual application will be described. Especial the *distributed* (4.5.2, *p.* 65) and *multimodal* (5.4.3, *p.* 94) aspects of this installation will be presented.

The system was developed by Imagination [58], the Computer Graphics institute from Vienna's University of Technology [25] and the AI-Group [4].

3.10 APRIL Framework

The APRIL framework for STUDIERSTUBE was created by Florian Ledermann in the scope of the IST Project VirtualShowcase.

> To support users in creating content-rich applications for AR systems, we wanted to create a language especially designed for the needs of this task.
>
> Authoring interactive, dynamic presentations is a process of varying complexity that may include several professionals working

> with different tools, but may also be performed by a single individual with more limited resources. The authoring process should therefore be scalable, offering all the possibilities to model simple presentations or prototypes quickly, and offering a structured workflow to teams working on larger presentations, incorporating various tools for modelling and content creation. APRIL supports such a workflow by separating presentations into different parts (story, components, media objects, hardware description, behaviours and user interaction are the main aspects). [76]

A language similar to HTML [149], and UML [100] was created to allow users (with users the story authors are meant) to describe an AR presentation. This language is based on XML [148] and has a rich set of elements for media presentation, dynamic interaction, and scripting functionalities. The strength of this framework is the abstraction of input and output, and the flexibility to adapt the created presentations to different environments.

Due to this abstraction any target system (VirtualShowcase, HMD, Screen, ...) can be used to experience the presentation/story. The flexibility is based in the design, because new elements of the language can be defined by the users. Ledermann also sketches different roles during the creation of a presentation:

> The APRIL authoring process helps a single author to structure the work, and teams of specialists working together to coordinate their efforts. For the authoring process, it is possible to define various roles of people contributing to the presentation. These roles may be embodied by distinct professionals (or teams of professionals), or by fewer or even a single person authoring a simple presentation. The following roles of people that contribute to a VS presentation have been analyzed:
>
> - *Domain Expert.* The domain expert is the individual or group with the necessary knowledge about the presentation's subject. For history presentations, this might be an archaeologist or historian, for scientific presentations an expert on the given subject.
> - *Story Author.* The story author is the person who comes up with the ideas of how the subject should be presented in an interactive way, and defines the storyboard for the presentation. In our model, the story author is also the communicator between the domain experts and the content creation people.

- *Content Creator.* Content creators design and deliver multimedia content for the presentation, following the storyboard as a specification document. Content creators deliver images, graphics, video, sound and 3D-models to be used in the presentation.
- *Component Implementer.* For sophisticated presentations, static media content has to be turned into components that can expose behaviour and react to user input. Additionally, custom components may be needed to realise complex user interaction or behaviours.
- *Story Integrator.* Finally, the story integrator puts together the components and media elements according to the storyboard, and specifies interaction techniques offered to the user. This is a similar, integrative position as the story author, and might well be performed by the same person. But while the story author acts a priori to the content creation to specify the details of the presentation, the story integrator takes the results of the content creation phase and puts them together. (*from [76]*).

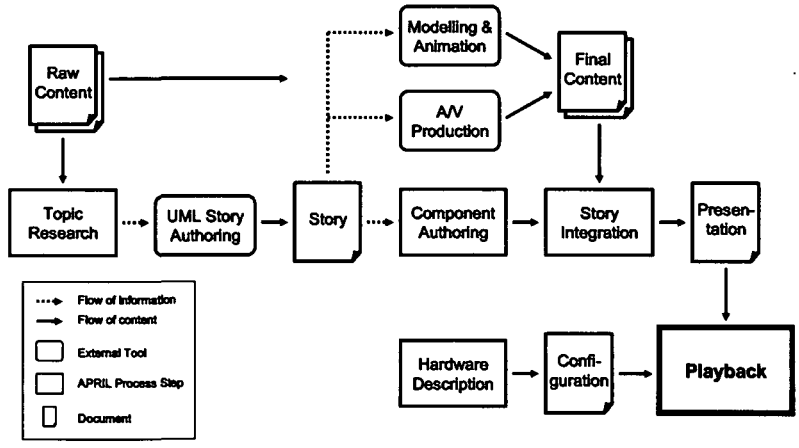

Figure 3.19: The APRIL workflow for creating a presentation.
(used with permission.)

Using the definition of roles given above, we get a sequence of steps to create an APRIL presentation, using the various tools that have been integrated. A graphical representation of this workflow is given in 3.19. The first

step in the APRIL workflow is research of the subject of the presentation. Raw material (text, images, video, sound, models) is collected, and the idea for the presentation is developed, possibly in sessions with domain experts and museum staff. This brainstorming phase results in the story document, the UML model of the flow of the presentation.

This structure does not necessarily mean that each of them must be a separate person, it is merely a distinction of different work-stages and also defines a clear interface between those stages. Using APRIL these stages can be separated, with an exactly defined interface that allows each of the stages to be worked on without having to think about the other stages. This is especially helpful for designers that do not have to knowledge to implement an AR system, while on the other hand help the programmers to implement general solutions that can be reused easily.

3.11 AR Puppet Framework

Based on ideas from *Invisible Person* and previous work on virtual characters and agents Barakonyi developed the *AR Puppet Framework* for STUDIERSTUBE. This framework defines the separation between different layers of control in a character enriched application. Together with the flexible APRIL language complex interaction patters between the users (meaning the presentation consumers) and a set of characters in the presentation are possible.

> In digital storytelling it is common to use a hierarchical structure similar to that used in a theatre [129] since these terms, which often represent complex system components, are familiar even to non-technical people. Although the comparison is not novel, we found that tasks to control AR agents can be divided into discrete groups which closely match the layers of a puppet theatre's multilevel structure. We therefore borrowed the stage metaphor for AR spaces, story metaphor for applications, puppet metaphor for AR agents and puppeteer, choreographer and director metaphors for various control logics. Interaction is performed by the storyteller, who also assures that the story proceeds in the desired direction. These components build up our hierarchical animation framework (see figure 3.20), which we call AR Puppet. (*from [6]*).

The Puppet as the bottommost component stands for one representation of an agent. It can be physical or virtual, for agents that have multiple representation, one puppet for each has to be implemented.

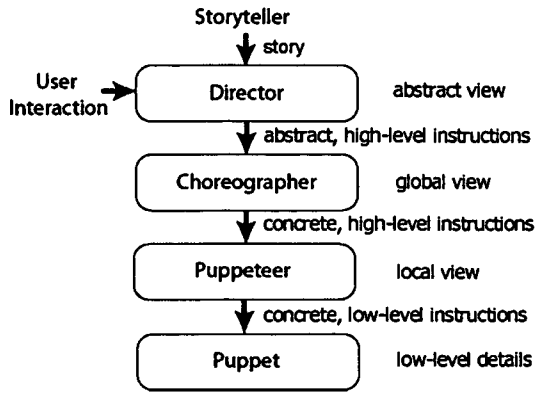

Figure 3.20: Overview of the AR Puppet framework.
(used with permission.)

The Puppeteer is the component that groups puppets together and controls a selected set of agent representations at the same time. It knows exactly "which strings to pull", that is how to implement higher-level instructions for each puppet to obtain the desired effect.

The Choreographer has a general overview of all puppeteers and their attributes. This level does not deal with character-specific details, but focuses on higher-level concepts like motion planning (to avoid collisions), and synchronisation between the puppeteers.

The Director drives the "story" (the application) forward based on choreographer events and feedback, user interaction and scripted behaviour.

Through the introduction of the AR Puppet framework to STUDIERSTUBE a consistent concept has been found to combine real and virtual representations of active components. This also includes devices like lights, printers or robots, which can now be represented and controlled in a uniform manner. With this framework it is possible not only to change the virtual augmentation of the reality, but reality itself (through the physical active components).

3.12 PUC Framework

One of the most used interaction concept in STUDIERSTUBE is the use of the personal interaction panel (PIP) and pen introduced by Szalaravri et.al. [135]. The pen is a tracked artefact with at least one button, PIP is a tracked plane that allows the user to perform two handed interaction with the system

- one hand holds the PIP and the other the pen as shown in figure 3.21. As described in [64] two handed input generally improves the accuracy of aim and hit actions. The PIP is used to display widgets similar to GUI widgets to enable the user to interactively change system parameter. Based on the metaphor of the PIP some other interface tools where also created like a virtual camera (to take pictures of the 3D scene with the PIP, using it as a window), a magic lens metaphor and also a Drag and Drop collector (selecting objects on the PIP and putting them into the scene).

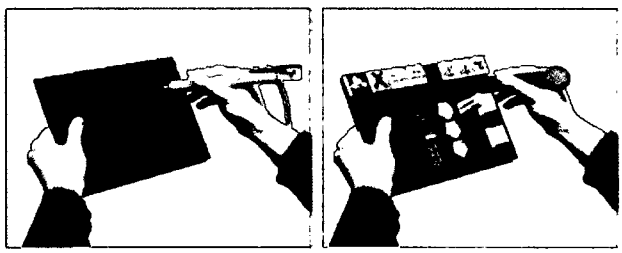

Figure 3.21: Pip and Pen

To further extend the input possibilities of STUDIERSTUBE, a bridge between the Personal universal controller (PUC) framework [96] and STUDIERSTUBE was implemented. With this extension the separation between input and functionality is possible. The PUC framework was actually designed to provide GUI on mobile devices for real devices that do not have a GUI.

Nichols et al. describe in [95, 96] a system developed as an approach for improving the interfaces to complex appliances by introducing an intermediary graphical or speech interface. This system automatically generates a graphic or speech interface with which the user can interact and control any appliance.

The PUC architecture consists of appliance adaptors, a specification language, a communication protocol and interface generators. The appliances allow connection to the PUC by means of the appliance adaptor, which represents a translation layer to its built-in protocol. The communication between PUC devices and appliances is enabled by a two-way communication protocol and a specification language that allows each appliance to describe its functions to an interface generator. The specification language constitutes the separation of the appliance to the type of interfaces it uses. The interface generator builds then the interface for the device that is going to control it, such as a graphical interface on a handheld or a pocket PC or a speech interface on a mobile phone. A diagram of this architecture is shown in figure 3.23.

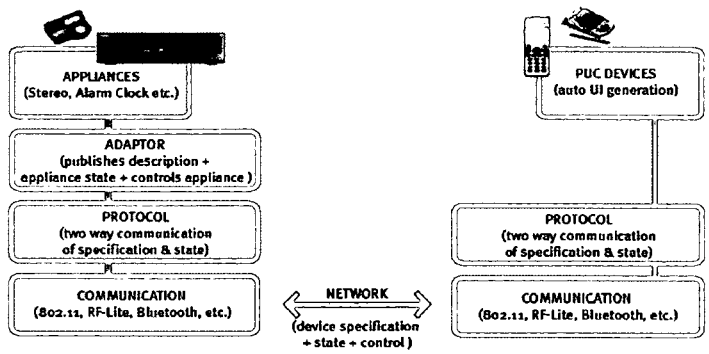

Figure 3.22: A diagrammatic overview of the PUC. A graphical or speech interface is introduced and communicates with the appliance by means of the specification language.

Figure 3.23: An architectural diagram of the PUC system showing one connection.

The PUC framework explicitly allows multi device scenarios using any of the possible interaction methods - using the buttons of the device, using a handheld PC or a smart-phone - each device can be used independently and even at the same time. No conflict handling is implemented, so the inputs are handled in the order they arrive at the device. If someone mutes the device and another user using the PocketPC changes the volume, it just depends on which command is received last, so either the music will be louder or muted (with a louder volume).

Using a handheld PC or a desktop program the states in the STUDIERSTUBE can be controlled. For the PUC framework each STUDIERSTUBE application is just another appliance providing the necessary interfaces. So a GUI is generated for all the states that are defined by the application. These states can also be changed by the application, which allows displaying data on the handheld device. As STUDIERSTUBE is a framework for rendering 3D content normally most of the data will be visualised by the application using the graphic tools available, but some times is useful to display data also on the handheld.

```
DEF CIRCLE_COMMAND SoPucCommand {
    labels ["circle", "add circle"]
    activeIf SoPucActiveIfNode {
        explanation ["You have to select at least 2 points",
                     "to be able to add a circle."]
        activeIf SoPucActiveIfClause {
            state = USE NUMBER_POINTS.value
            op GREATER_THAN value 1 }
    } #end activeIf
    explanation ["Adds a Circle, based on 2 points",
                 "that define center and a point on the circle"]
} # end circle command
```

Figure 3.24: An example how a command is defined using the PUC framework in STUDIERSTUBE(taken from the *Construct3D* definition file). The *activeIf* structure defines that this command is only available if at least 2 points are selected. The *explanation* defines the tooltip text that will be shown.

Using the PUC framework to describe an interface, allows to rapidly develop and change the interface for an application. It also allows to represent internal application states in the interface by defining conditions, under which a command or value may be changed. Also values can be displayed in a "read-only" manner, so that the value can be perceived by the user, but not changed (see figure 3.24).

In STUDIERSTUBE a similar auto GUI generation algorithm has been implemented [84]. By specifying the states and their interdependency an automatic layout of widgets is created. These widgets are then being displayed on the PIP and can be used to control the application. This way an abstraction layer between input and functionality is introduced to the STUDIERSTUBE framework, allowing the application builders to concentrate on the functionality of the program.

3.13 Construct3D

The *Construct3D* project started in the September 2000.

> In order to solve three dimensional mathematical but especially geometrical problems, spatial abilities are an important prerequisite [16, 35, 45]. Many students have difficulties solving tasks that require spatial visualization skills and spatial thinking. To get passing grades they use strategies such as learning construc-

tion steps by heart without fully understanding spatial problems and their solutions in 3D space...

Construct3D [67, 68, 70, 72] is a three-dimensional dynamic geometry construction tool that can be used in high school and university education. It is based on the STUDIERSTUBE AR system and uses augmented reality to provide a natural setting for face-to-face collaboration of teachers and students. The main advantage of using AR for constructing geometric objects is that students actually see three dimensional objects which they until now had to calculate and construct with traditional (mostly pen and paper) methods.

Mid- to long-term plans are to integrate *Construct3D* and the concept of teaching geometry with Augmented Reality in high school and higher education by collaborating with Austrian schools and external partners such as the Institute of Geometry at Vienna University of Technology. (*from [69]*).

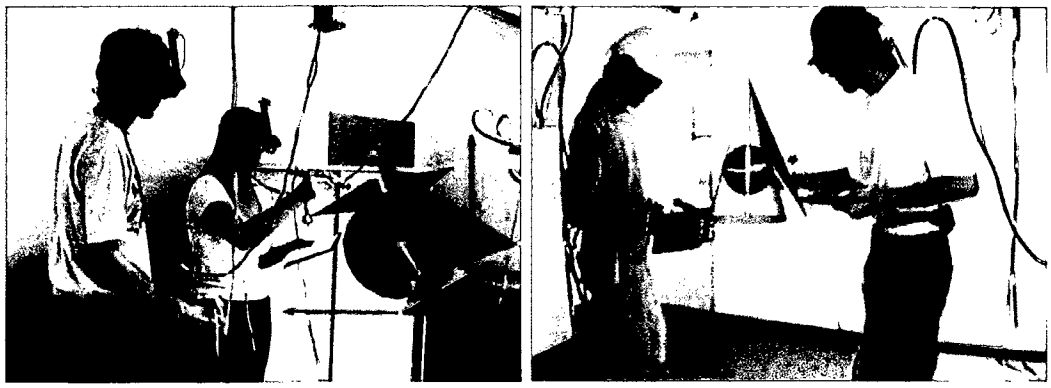

Figure 3.25: Students work with *Construct3D*.
(used with permission.)

The project was evaluated twice (and will be evaluated again in the near future). In the second evaluation (documented thoroughly in the PhD thesis of Kaufmann [69], *p.* 111-126) interaction issues could be clearly observed from the questionnaire.

As a standardized usability questionnaire we used the ISONORM questionnaire by Prümper [112] and adapted it to our needs. The ISONORM questionnaire was derived from the software ergonomic standard DIN EN ISO 9241 Part 10 (German Industry

Standard). This questionnaire is designed to test the usability quality of software following the ISO 9241 part 10 principles. It represents an operationalism of the seven dialog principles in the ISO-standard: suitability for the task, self-descriptiveness, controllability, conformity with user expectations, error tolerance, suitability for individualization as well as suitability for learning. (*from [69]*).

The second field trials displayed clearly that some points of the application need improvement. With a rating between -3 (very bad) to +3 (very good) the overall score was 1,44 (see figure 3.26). For some issues, where rating was not satisfying (lower than 0,5), can be tackled with the latest developments in STUDIERSTUBE. In the table 3.1 parts of the questionnaire are presented. Beginning with the some of most positive results, followed by the issues that are dealt with in the redesign.

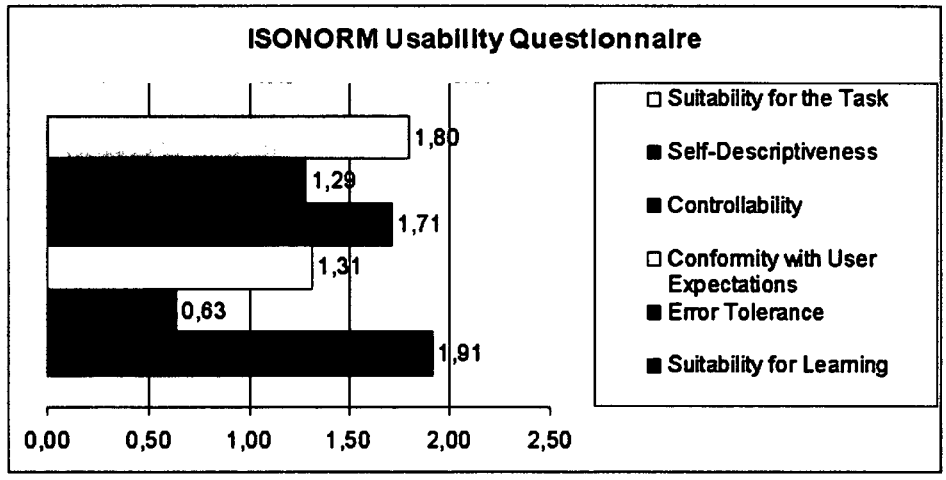

Figure 3.26: Overall results from the questionnaire.
(*from [69]*). *(used with permission.)*

Based on the improvements of the widgets capabilities in STUDIERSTUBE and the integrated PUC framework, we (the author and Kaufmann) decided to deal with these problems and add new features that were requested during the field trials.

Though it must be noted that students did not feel the need for improvements in the field of interaction design (as most users are not aware that the design is causing them having problems, with a system [97], *p.* 1–33). Suggestions for improvements were mostly concerning the physical configuration of the HMD and software speed and stability.

Question	Mean Value	Standard Deviation
is complex/easy to use	2,36	0,50
offers no/all functions to solve tasks efficiently	2,14	1,10
uses incomprehensible/comprehensible terms in menus	2,29	0,47
allows no/an easy change between menus	2,50	0,52
quires a lot/very little time to learn	2,29	1,07
informs sufficiently/does not inform about valid and invalid input	0,29	1,64
informs insufficiently/sufficiently what it currently does	0,36	1,82
informs too late/immediately about wrong inputs	0,00	2,12
does not/does give concrete tips for error correction	-0,57	1,83
is difficult/easy to learn without somebody's help or a manual	0,57	1,34

Table 3.1: Some results of the ISONORM usability questionnaire.
(from [69], p. 122–124). (used with permission.)

The improvements will be described in the section Construct3D *improvements* (5.6, *p.* 100).

Chapter 4

Distributed and Configurable Aspects

The meaning of *distributed computer systems* in this thesis is that more than one computer and device is used simultaneously. But distributed has also the meaning of spatial distributed, that the space is divided into different workplaces (see *Configuring the Space* 4.9, *p.* 74). Distribution also allows to create multi-user applications that allow real time collaboration over space boundaries.

Early work on distributed systems was published by Morris et.al. [90] who described a system that has been develop for universities. Most concerns (about 20 years ago) are about networking and global repositories. The principles that he proposed are still valid, although computing power, network capabilities and personal and local storage space has increased.

Selected related work from the field of distributed systems and distributed aspects of the applications is presented in this chapter.

4.1 Distribution of Input and Output

The exchange of data between different machines is performed using computer networks. Due to the rapid development of wired and wireless networks the transport of data from one computer to another is no longer a major problem. Distribution has some advantages while at the other hand it increases the complexity of a system. Synchronisation is just one of the many issues that arise from a distributed setup, also the system design is heavily influenced by the decision to create a distributed system. The software designers have to create simple interfaces between the distributed parts of the system to ease the development of the network protocols. Further

more decisions have to be made, which functionalities will be available and in which part of the system those functionalities will be implemented (see *Display Manager* 6.7.1, *p.* 122).

"Distributed" means that input, output and processing are distributed. While distributed input is quite common - special computers to handle and calculate tracking - distributed output is a complicated research field. Part of the research is focusing on "intelligent" systems that try to automate the selection of an appropriate output device, based on heuristics and learning from users interactions, sometimes implementing peripheral displays [80, 20, 50, 92].

When distributed input is available the binding between action and reaction is also a issue. When a user interacts with a input device in a distributed system the meaning of the user can be unknown or even confusing (for the system). Some systems try to solve this by applying heuristics or predefined behaviour [23], if the heuristic is based on the location of the user this is known as "location based service". Other systems allow the users to create this binding using various patterns (see *Configurability* 4.9, *p.* 73).

4.1.1 Input and Output abstraction

Before the data is send over the network, the representation of the data is abstracted, which means that a general description for that data has to be created. In this way differences that originate in the different hardware and software used, can be made transparent for the application developers.

Input abstraction When creating a system that incorporates multiple input devices, the input interaction must be abstracted using different layers. Input abstraction on a low level is already present in most frameworks [117].

The *PUC Framework* (3.12, *p.* 47) uses a abstract description of the interface that consists of states and commands. The different input possibilities are then implemented in the clients that allow changing the states or issuing commands. Each of the states has a name that should describe the state's meaning, but the real meaning of the state or command is grounded in the reaction of the application. The *PUC Framework* lacks of a concept of distributed interaction as it was created to control appliances that are normally integrated into one device. Multiple devices can be controlled, but they normally do not influence each other, also it is not possible to create one global interface for a number of (distributed) devices.

We propose that "input abstraction" should be extended to "interaction abstraction" meaning that high level interaction events are created and processed by the system.

While *input abstraction* is quite common in most implementations, *interaction abstraction* is rather seldom. To get the most out of the abstraction a three layered abstraction is suggested. The first layer hides the actual hardware from the system – generating hardware independent low level events. Examples for these events are orientation of an object, state of a button and so on (see *OpenTracker* 2.2.1, *p.* 12).

These events are then processed by the second layer to generate high level events that do not depend on the application context. Such events are `Button 'Save' pressed`, or `Object X, side 3 facing up (changed from side 2)`.

The third layer takes the context of the current situation into account and creates messages like `save data of Device 'Screen 3'`, or `save data of all active Screens`, or `switch to configuration mode`, and `display Image 254 on Screen 2`. This final message then triggers a functionality that performs an action based on the content, or to an output device that displays the content.

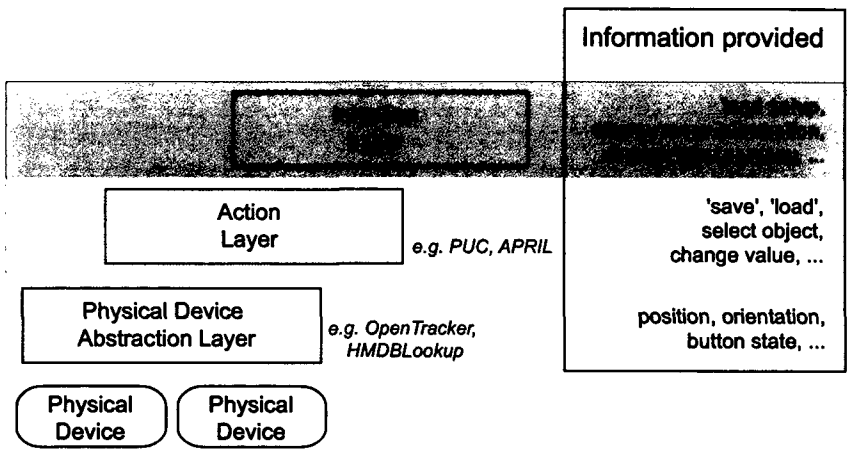

Figure 4.1: The three layers of input abstraction, in the topmost layer the "interaction abstraction" is realised.

In the *APRIL* language this kind of "high-level" information is partly described, still a presentation author would specified that a object has to be "selected" (where the "how" to select that object is left to the actually target platform) to get additional information. The "high level" information, that the user wants to have additional information displayed (where the users is communicating that by "selecting" the object) is the type of information we propose to use in a framework.

OpenTracker: The approach of STUDIERSTUBE is to use the *OpenTracker* framework to merge different input sources into one graph that includes all

the input devices. The merging of more than one input source is also possible and transparent for the application developer. Up to now this has to be done at the start of the application, but already the development of a more flexible framework has started.

Atelier: In ATELIER the different input devices are abstracted as *Components* that communicate their state over the network. This means that they can all run on one machine or each component on a separate computer (or something between that) . This approach is very flexible but rises some issues that have to be solved, like *conflicting inputs* (6.2, *p.* 108) and the problem of uniquely identifying similar input devices. Also the abstraction of the data being distributed has to be defined. This ranges from very basic information like "key pressed" too complex information like "the string 'Hello' was entered". In ATELIER we tended to make all abstraction levels available by sending the basic information to another component that would process the input and again produce a higher level information. As long as network bandwidth is not an issue, this is the most flexible solution, as all layers of information are available for the other components.

Output abstraction The abstraction of output goes beyond visual device abstraction, but should also be extended to internal output of the functionalities. In the ATELIER project this was quite simple as most of the outputs where connected to the hypermedia database.

Output abstraction has a lot of different meanings. Like display abstraction - providing images and videos for a variety of display technologies ranging from small displays in cell phones, handheld computers, desktop screens, head mounted displays, up to full scale projections or even tiled displays. But also cross modal displaying is a form of output abstraction, what if only an audio channel is available? For HTML pages special descriptions can be provided to automatically convert the visual representation of a page into spoken text. Visualising sound and performing other cross modal conversion is often used by artist as a source of inspiration.

Output abstraction is a way of representing content, independent of the use context and the available output devices. For example if a presentation is prepared and the presentation room is not available for all presenters to prepare their presentation in the room itself. So some sort of abstraction is needed.

This abstraction of output has been fully described in the *APRIL Framework* (3.10, *p.* 43) of STUDIERSTUBE, and is used to be able to reuse a presentation in various environments, even a pure text-based representation could be realised.

In the ATELIER project a different concept of output abstraction was realised. As ATELIER is merely a collection of components the input and output of these components were implemented in a general way to allow easy coupling of the components to create component assemblies.

For example if a user selected a specific image this information could be received by a lot of different "output" devices. Some of these devices would really display the image on screen or on paper (print the image), others would use the image as input to for example search for similar images in the database (see *Combining the Search Methods* 6.4.4, *p.* 114).

4.2 Distributed Input

As already envisioned in the article by Weiser [150] more than one computer system is used to control the environment. Therefore the input and output can be distributed – meaning being processed at different machines.

The implementations of STUDIERSTUBE and *OpenTracker* are typical representations of a server client concept. *OpenTracker* is the input server component of the system while STUDIERSTUBE with distribution is the output component of the system. When creating a distributed STUDIERSTUBE application a "master" (server) is defined, which updates the clients. The clients can update their scenegraph and perform calculations, but are practically just used to render the scene graph that is kept up to date by the master.

The input component (*OpenTracker*) sends the input data to all clients, which can locally update the part of scene graph that is depending on the input devices. This is only true for the rendered parts of the application, important states of the application are distributed from the master to the clients. A good example is the slider widget: The clients are allowed to render the slider widgets based on the interaction of the user's pen with the slider itself, but once the slider drag operation is completed (the button is released) the master sends the final value of the slider to all clients so that glides in the tracking updates are synchronised again. This ensures that no conflicts arise, as the master will always send the final results of an interaction to all clients.

In the ATELIER framework the shared space concept was realised. As each of the components has the same importance in the system it is impossible to specify what is a server component and a client component in this system. This makes the communication between the components easy. The kernel component can be seen as a sort of server, but actually it is just a mean to establish a communication between components. The kernel is just

performing routing and filtering of data that is exchanged between different components.

4.3 Distributed Output

Distribution of output can be a very challenging task, where the technical issues are relative simple. The main issue that has to be tackled are design issues. Where does the output happen, how is it presented to the user, how does the user know where the output has gone? This question is related to the *alignment* that is described in section 2.4.5, *p.* 21.

Due to the multimodal capabilities of modern computers other output possibilities aroused. Graphic display (mono or stereo), sound and vibration are now quite common. In most of the systems that are centred around one user sitting in front of the computer screen, distributed output is not really a big issue. But in environments where multiple displays are used by one or more users distributed output has to be designed.

A number of projects concentrate on peripheral displays that take advantage of people's ability to utilize peripheral information with little effort [20, 60].

> Moreover, constraining the interaction to the desktop is a poor match for common human behaviours such as using large amounts of physical space to simultaneously organize, monitor, and manage multiple activities [93] ...
>
> There have also been attempts to leveraging our 3D abilities within a standard display by replacing the 2D desktop with a 3D world containing 2D documents (e.g., [19]) .... (*from [80]*).

Especial the *Rooms* [50] and *Flatland* [92] try to make use of peripheral displays in office settings, which support the ability of people to perform multiple simultaneous activities [88].

In early network printing systems the print task could be re-routed to another printer, when the main printer queue was full. This situation led to the amusing and annoying office game of finding the printout. This issue was of course soon solved by presenting the user a feedback that stated at which printer the pages can be retrieved [99].

In many situations the decision of *where* to present the output can be solved based on the actions performed by the users of the system. For example if a user actives a widget on a single computer display with the mouse, the obvious place to present the outcome of this action is the same display (or

the sound system of the machine the user is working on). When using more than one display – as it is quite common for graphic designer – , changing the adjoined display is also suitable, because it can be expected that the user watches both screens, probably expecting a change on the other display.

In [9] Beaudouin-Lafon proposes that the *Degree of indirection* measures how *direct* an interaction is. Definitely activating something on one screen and displaying the results on a different screen incorporates a *spatial offset.*

In a distributed environment this *spatial offset* will always occur. Actually the *direct manipulation* paradigm described by Shneiderman [127] is only satisfied by augmented reality applications with direct manipulation and augmentation of the results.

In some situations the expectation of the user can not be determined, especial if a certain command is used for the first time by that user. To take the above example of the designer working with two screens for the first time, it may not be clear that activating an element on one screen can change the feedback on the other screen. The design of distributed output has to be carefully though over and tested with different users to ensure that it is understandable.

4.4 Distribution in the Systems

The both systems discussed in this thesis STUDIERSTUBE and ATELIER have different strategies in distributing interaction, processing and content. First other systems with distribution will be presented, then we will go into detail of the two systems and the applications.

4.4.1 Systems with Distribution

In AR systems distribution is quite common. Due to the amount of calculation and therefore computing power a distributed system is needed. In ubiquitous systems distribution is more a feature than a need, in general CORBA, JINI or JavaWebStart technologies are used to implement the distribution. Research about distribution in a ubiquitous environment is mostly published under the term "intelligent environment" [48, 49]. A extensive survey of systems providing distribution is presented by Endres et.al. in [31], where most of the *smart spaces* or *instrumented rooms* systems are described as being in concept phase or just starting. Also the vast field of frameworks for *distributed mobile systems* is described in the work of Endres et.al.. In total 29 frameworks are described in this publication (which will be published in 2005).

A number of projects provide distribution for AR systems, some of them will now shortly be described.

The *DWARF* project [8] aims for a design concept that differs greatly from traditional AR software designs (like e.g. STUDIERSTUBE). The basic units of the DWARF framework are distributed services. A *service* is a piece of software running on a stationary or mobile computer that provides a certain piece of functionality such as optical tracking. Services can be connected to use the functionality of other services establishing a data flow network to achieve a more complex function.

The *Coterie* system [79] supports scene graph based graphics programming and abstractions for tracking devices. Support for distribution was implemented, but required some implementation effort, nevertheless scene graphs could be easily shared. *Avango* is a framework for developing virtual and augmented reality application that also supports transparent distribution [139, 140]. Distributed applications can be developed that implicitly share data between different instances. The *Tinmith* system is a full featured software architecture for mobile augmented reality applications [109]. Originating from an older architecture that was built from a network of communicating agent processes [107, 110], it is now supporting hierarchical scene graph based modelling and generic data flows between objects [108]. It also features a persistent storage to the file system.

To model what a service can offer to other services and what it needs from other services, DWARF uses the concept of *needs* and *abilities*. A match of one service's need to another service's ability leads to a connection between the services; this is set up by the distributed service managers [81].

4.4.2 Distribution in Atelier

The design of *DWARF* is quite similar to the ATELIER framework as all components of ATELIER have to register the messages that they provide or handle. With this meta information the kernel is able to apply filters to the messages that are routed through the kernel. An additional feature was introduced that allows to send a message to a specific component. Each component has a unique name, that identifies the component. When sending a message the receiver of the message can be specified. This ensures that the message will only be received by the component and also limits the network traffic. Messages without a specified receiver will be sent to all components that are interested (as mentioned they have to register those messages). To provide a flexible way of filtering those messages 2 mechanisms where implemented:

1. *Hierarchical structure of the message category.* The message category can be specified by the sender of a message. A preliminary structure has been defined by the ATELIER group to ensure that the structure of the category hierarchy is meaningful.

2. *Hierarchical structure of the message type.* The message type is also defined in a hierarchical structure. This specifier can again be used to transport any additional information for the receivers.

Because there are actually two independent information for each message the processing of the messages is very flexible.

```
<registration name="Configurator" type="component">
  <metainfo name="provides" type="registration" >
    <provides name="category" value="Root/event" />
    <provides name="type" value="configuration/text" />
  </metainfo>
  <metainfo name="handles" type="registration" >
    <handles name="category" value="Root/event" />
    <handles name="type" value="input" />
  </metainfo>
</registration>
```

Figure 4.2: A configuration structure from the ATELIER Project.

If a component registers to handle all "Root/event" messages practical all interaction related messages, which are send to all components, will be received by this component. The HMDBLookup Component (see 3.3.4) on the other hand is only interested in "input" type messages. Therefore the "output" message events will not be send to the HMDBLookup Component.

The actual presentation devices we have used in the ATELIER project were already described in the section *Presentation devices in* ATELIER (3.3.2, *p.* 27).

4.4.3 Distribution in Studierstube

In STUDIERSTUBE Open Inventor scene graphs are used to store and share information between the clients in the system. The abstract definition of a "node" that contains fields with values and other nodes is the way information is structured. This way of specifying an "information database" originates in the idea of scenes that store graphical information to generate a rendered image. Using this specification any information - also application states - can be represented.

The communication in STUDIERSTUBE follows some simple principles: if a field or node is changed, this change is distributed to all other scene-graphs where this information is represented [52, 124, 51].

In the ATELIER framework there is a global message "Element X (of the HMDB) has been updated", which can be used by any component to act accordingly. This has the advantage that the state change is also stored in the HMDB and is persistent even if some components are not active at the time of the change.

4.5 Distributed Processing

We will now go into detail of some of the applications that have been developed, describing how the distribution of data and processing was implemented. Two extremes will be discussed

- The *Sensitive Sample* project, where the low-level of data exchange between micro-controllers and Desktop PC will be presented. Furthermore we will describe how the sensor data is processed by the ATELIER components.
- The *Invisible Person* project, where a high-level approach of distribution of data and responsibilities will be described.

4.5.1 Sensitive Sample

After specifying the requirements for the *Sensitive Sample* (3.5, *p.* 34) and with a relatively small budget a micro-controller solution was chosen. The Atmel [3] micro-controllers were used. The printed circuit board is designed so that it includes only the necessary electronics needed for the controller to function. There are connectors to all I/O ports of the micro-controller and a reset button. All additional sensors or signal LED's are realised on separate modules and can be connected to the main board.

The micro-controller can be easily programmed, and can communicate with a PC via the serial port. After successfully finishing the first board, various sensors were added. Tilt switches and touch sensors were the mostly often used kinds of sensors. Tilt switches are a very simple piece of hardware that connects two wires depending on the inclination of the switch. This sensors can be used to estimate, which side of the object is facing up, or to detect shaking of the object. Touch sensors provided by Quantum Research [138] have also been investigated and deployed in the samples. These sensors have to be mounted beneath an insulator and can sense if the object is being

touched at a certain place. They can be built in various sizes, cover various areas of an object, and are not visible from the outside, which makes them popular among the students, because the look of their model is not changed. Incorporating such sensors in the architecture model, and triggering various multimedia files during a presentation, just by touching the model, was a great fun for students and teachers.

Figure 3.11 (*p.* 35) shows a model and touch sensors mounted underneath the model. Once the main board and the sensors are installed the communication between the sample and the PC has to be solved. A wireless communication was required, and support for more than one sensor simultaneously. A communication protocol on a time sharing principle was developed. There is a special module attached to a PC, which is called server. Each sensitive sample has an unique ID, and acts as a client in the system. All clients and the server are equipped with radio transceivers. This is a device, which can transmit and receives radio signals, though there is a small time delay when switching from receiving to transmitting. Figure 4.3 shows the main board with the transceiver connected. The following protocol was defined:

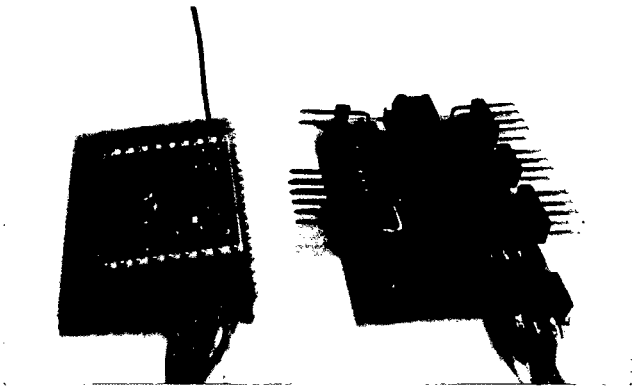

Figure 4.3: The main board with micro–controller and transceiver board connected.

- the server is in transmit mode and all clients are in receive mode
- the server sends the message "send status"
- the server switches to receive mode for a certain time
- the clients analyze the message, prepare the status, and then send an answer with a time-offset depending on their ID, and switch back to receive mode
- the server collects all the stati and then sends them to the PC
- the server switches to transmit mode

This list is performed 20 times a second, this allows us to support up to 10 clients simultaneously. The advantage of such protocol is that only one frequency can be used, and since every client has its own time slot depending on the ID, a collision can never occur. The server also knows which clients are active, and if some message did not reach the server, they will be sent again in the next cycle.

MaterialKammer For the MaterialKammer setup a distributed processing of the sensor data was developed. When users are interacting with the *Sensitive Sample*, the signals are sent to the main computer. Actions are being interpreted there, and high-level messages such as: "shaking wood hard", or "knocking on glass" are being sent to a multimedia content selector. The multimedia content selector selects a media (in broader sense, where light and wind are media as well) depending on randomness level (surprise factor) and the users current and previous actions.

4.5.2 Distribution of Data in the Invisible Person

The *AR Puppet* framework (see 3.11, *p.* 46) features the distribution of responsibilities, as each layer has its own responsibilities and just relies on the layer directly beneath it. In the Invisible Person the different modules are not designed as a hierarchy, but as a network of inter-operable modules, that more or less depend on each other. Still each module has a exactly defined scope of responsibilities.

Based on the quality of the *vision system* (5.4.3, *p.* 96), which means better information about the people on stage a new interaction pattern was implemented. The system is divided into four modules. The vision-module analyzes two live camera video streams and generates information blocks, which are used by the AI to make decisions about new interactions. The AI-module decides what animations the virtual character should display. The game-module holds all information and algorithms about the games, detaching the game-knowledge from the emotional and interaction algorithms of the AI. The render-module displays the animations and the additional graphics needed for the games, additionally the rendering of the occlusion is performed by this module. These modules communicate over a 100 MBit network connection. The TCP/IP protocol was chosen to establish reliable communication. Different internal protocols were designed to allow compact and lean information-flow.

The vision-module delivers information about the user's position and current posture to the AI and the game module. The render-module is supplied with occlusion-masks and depth information. The AI-module sends control

information to the game-module and advises the render- module to display the appropriate animations and emotion-textures. Together with the virtual character from the render-module the AI forms a symbiosis of body and mind: "The Invisible Person". The game-module provides information about the game status for the AI. The graphic elements of the games are controlled by communicating with the render-module. As the games and their specific rules must be separated from the implementation of an artificial intelligence, the game algorithms were implemented in a separate module. To keep communication straight and simple, a generic representation for the game information was developed. The render-module sends position and posture of the virtual character to both the AI- module and the game-module. The game-module and the AI-module need this information to merge the user input - from the vision-module - and the character input - from the render-module - to get an overall view of the interaction. We will now describe the distributed processing approach and then in the section *The human body as an interface* 5.4.3, *p.* 94 discuss in more detail the different modules.

Game-module Because the game-module was realised as an independent module, careful consideration of the interfaces was required. The inputs for the game-module are position and posture data of the users from the vision system, the position and action of the virtual character from the render-module and control messages from the AI. As the information had to be distributed via a network protocol, the interfaces had to be general and compact. The AI has control over the game-module via the control- interface. This interface allows the AI to start and stop a game. Additionally, several parameters of the game can be set, such as the difficulty level and which player has the first move. The outputs from the game-modules are game-status information including overall game score, a game history and end causes. This information is processed by the AI to update the emotional model. Possible moves for both the users and the IP are distributed by specifying regions (2D boxes) and a rating for these regions. A timeout was added to enable time-critical games (such as volleyball). If a player has to wait, no moves are available. As both user and IP moves can be set at the same time, real-time games that allow both parties to play can be realised. Avoid regions for the IP are supplied to give the AI a hint about the game-board, and when to avoid it. This is necessary to let the user have a look at the game-board and choose his game-pad without being disturbed by the IP. The game module does not interfere with the AI system itself, that is, the game-module does not make any decisions. It simply supplies the AI with information so that the AI can decide what to do. For the render-module

a different interface was needed to ensure the correct visualization of the game-pads and other graphical gimmicks.

Adding Games. We wanted the different modules to be independent, to be able to add more games later, without needing to change the AI system, so the interface between AI, game-module and render-module had to be general and versatile. Although the AI decides when and what *type* of game to start, selection between the different games available is done by the game-module. Thus more games can be added without changing the AI system. Our abstract interface allows the creation of very different games. As the information contains only regions, ratings and timeouts, games without game-pads are also possible. As a matter of fact photo-shooting – the IP can make pictures of the users, if it likes – was also implemented as game (see figure 4.4).

Figure 4.4: Photo-game of the Invisible Person application.

The rating information for the user region is supplied to the IP to enable it to give a feedback while it waits for the user to move. Depending on its internal state it can encourage the users with appropriate gestures to make good or bad moves.

4.6 Distributed Context

If the processing of data is distributed, so can the "state" of the environment be distributed. Especial if a number of components are used simultaneously, the current context of the system can be distributed among a number of components. How we handled this situation in the ATELIER project is discussed

now. In STUDIERSTUBE due to the master-client concept, all information will be present at the master and therefore not distributed. Although a different approach can be realised in STUDIERSTUBE until now such a application with a context being distributed among the clients was not implemented.

4.6.1 Saving and loading in distributed systems

Saving the work that has been achieved using an application is something natural for single computer applications. It is merely a matter of saving the current state of the work, to be able to continue at a later time.

ATELIER. In distributed systems like ATELIER this very important feature can get quite complex. A lot of different network components have to store their current state at a specific point in time. To load this state all these components must access their individual savings and restore their state. Some components are actually state-less, which means that they do not preserve a state or are just responsive elements in the assembly. Those components do not have to store their information, but sometimes it is important that they were active in the setting, so that information should be saved. Some "Meta-Component" has to take care about the information that a specific component is actually part of the setting. The ideal case is that the whole distributed state, all the running services and even the physical state of the environment is stored and recreated.

In the ATELIER project a "Store State" and "Restore State" method was implemented in each component, which carried a state. By triggering a save action and supplying a physical handle, the state of all components can be saved by the students (see *Discussion – DisplayManager* 6.7.1, *p.* 122). It is also possible to just store the state of selected components, saving just parts of the setup. This is actually just a short cut, it speeds up the process of saving and restoring, as not all components have to store and restore their state. Using this feature different states could be mixed in an easy way, as only the respective part of the environment that was stored earlier will change when restoring that setup later. A better method would be to selectively restore states leaving the rest of the environment unchanged (see *Improving the Save command* 6.7.2, *p.* 124).

Capturing the physical state. For the physical parts of an environment the current state can only be captured, if actuators are installed that move the physical objects in the environment. Either the users are not allowed to move the objects directly, or the moveable objects are somehow registered. If

the objects are tracked (their position and orientation is known), they then must be able to move to the position in space, where they have been, when the state was saved.

In the ATELIER project this was impossible. For example the projectors were moved all the time. To recreate a specific physical setup, the users had to re-adjust the physical props in the environment. This was especially complicated for the positions of the projectors that were used in conjunction with the Texture Painter. In one session different groups of students actually used different physical setups to augment their artefacts.

Recreating these setups "on the fly" during their presentation actually did not work out perfect. But because of the "sketchy" character and inspirational focus of their presentation there was no need for prefect setups and therefore it did not pose a problem.

For physical interaction tools this is a specific problem that will be discussed in section *Tangible Interaction* (5.2, *p.* 84).

In STUDIERSTUBE the physical objects are tracked, which means that their position is always taken into account and does not carry a "state". That way their position is not important to recreate a setting that has been saved.

The tracked physical devices in STUDIERSTUBE are props (like Pip and Pen) or displays like HMD's, and the tiled display (*StubeRena* project). These props have to be calibrated before they can be used. To ease this process a number of algorithms have been implemented to calculate the needed data. This calibration information is then stored in files that are used by the applications. The process is rather static and is only partly related to the problem of capturing or recreating the physical state of an environment [38, 39, 121].

4.7 Managing resources in a distributed system

In a distributed system there is always the issue of handling the different resources. In STUDIERSTUBE resources like geometry or images are usually distributed from the master to the clients by sending them over the network. Another approach is to mirror all needed resources by soft- or hardware solutions and keep them synchronised [17]. A different approach was implemented in the ATELIER project. A global repository was installed, which stored and provided all resources for the applications. Each resource in the HMDB has a unique id with which it can be retrieved.

4.7.1 HyperMedia DataBase (HMDB)

Besides the possibility to store digital content in the HMDB, there is also the possibility of creating hierarchies and add meta-information to each element in the database. The meta-information is stored as a list of key and value strings that can be used in different ways. Methods for searching the database for specific elements (using the meta-information) and retrieving whole parts of the hierarchy were implemented.

Two generic interfaces have been developed to edit and manage the content of the database. The Path Creator (part of the E-Diary setup) and a HTML-Access, which enabled the users to retrieve information and modify the database from any computer equipped with an internet browser.

The Path Creator was written in JAVA presenting the whole structure of the database to the user. Generic edit methods for adding and removing elements in the database are available, as well as a method for editing the meta-information for each element.

Its original purpose was to add information that had been recorded with the E-Diary. The information of the E-Dairy had to be completed with user information and additional maps, to be able to correctly present the information to the users.

The HTML access was implemented as a set of Java-Servlets that create the HTML documents, which represent the interface to the HMDB. The main advantage of the HTML access is that it is independent from the JAVA runtime environment (and some libraries like the Java Media Framework JMF). The only software needed to use the HTML access is a web browser and the right URL (unique resource location). The HTML access also allows the users to print out the physical handles (see *Physical Handles to digital media* 3.3.3, *p.* 30).

4.7.2 Messages versus HMDB

Because all components can store and retrieve information from this shared storage component, it can also be used to establish a communication between the components. This allows the component to process information that is persistent (and not a message), which can be retrieved by the same or a different component.

An example is the storage of keywords used by the *ontology* (3.7, *p.* 39) in the HMDB. The keywords are used and manipulated by the ontology component, while they are at the same time used by the display component to provide the users with feedback of what has been changed and which keywords a media is currently attached to.

When a student activates a "toggle keyword" barcode, the HMDBLookup component processes this barcode, and sends a message to a component that updates the keyword information in the HMDB. This activates another message that specifies that an entry in the database has been changed, which results in updating the ontology database and the display, that is currently showing the image.

The cause for us to choose this method is that even if the ontology service is not running, the change of the keywords will be stored in the HMDB. The ontology database can be resynchronised with the HMDB at any time, ensuring that changes in the keywords will also be reflected in the ontology database.

4.8 Distributed User Identification

Identifying a user in a normal desktop setup is easy: there is just one user. In a network system this is solved by the login mechanism, that ensures that only allowed users can perform actions. In a high security environment just entering a password is sometimes not enough, the users have to perform various steps to identify themselves. The ATM machines for example only allow access to the account, if the user has a valid card and knows the code that belongs to that card. Other login mechanisms include face, voice or even fingerprint recognition.

Once the user is identified in the system a number of tasks can be fulfilled:

- *Restrict access.* The rights to open, read, add, delete or change content is based on the user identification. This ensures that users can only perform actions that they are allowed to do, or destroy (unintentionally) the work of another user.

- *Add ownership to content.* Whenever content (files) are added or changed, the user information can be stored, at the same time the access rights can be set for that content.

- *Record user actions.* Every action a user performs can be stored together with the information, which user performed that action (see *Undo – Qualities* 6.3, *p.* 110).

- *Personalised Settings.* After the user has logged into the system the personal settings can be loaded (and stored) so that the user has the possibility to configure the system in a persistent way.

- *Receive messages.* Messages (from the system or other users) can be addressed directly to a specific user. Email's are a common way of informing users of something. But there are also some more direct messaging systems like the "MSN Instant Messager","Yahoo Messenger", and "Jabber" just to name of few.

In ATELIER we did not implement user identification. We started to explore the issue of content ownership by including a meta-information "author" for content in the HMDB. We also thought of user management including login and registering users, but due to the openness of the system and the aim for a quick and easy access to the system the user management idea was dropped again. Tangible interfaces commonly share this problem of not being able to determine the user of the interface, because it can be used by any person, who has physical access to the device.

Some systems evaluate especially this issue, by e.g. the use of RFID tags. A system was tested on a conference, that identified the current speaker or people gathering in a meeting place. The users had to be registered in the system before or during the conference (using a web interface). They were provided with a special conference label that also incorporated a RFID tag. Special places (like the place for the speaker) and some meeting places were equipped with wide field RFID readers to identify the person (or group of persons) that is present in a specific space. The users were detected at these places through their tags and additional information was displayed related to these users [28].

Another approach is the use of the "Bat" system that was explored in the AT&T laboratories [23]. This system is a wide field tracking system that is able to track objects (using the bat's) in a whole building. The personal bats, the researches took with them, allowed the system (and the users of the system) to locate each individual in the building. This functionality was e.g. used to route calls to a phone in the same room where the user was last located [94].

Using a system, where the users can be identified without performing an action, raises e.g. the issue of privacy.

A user of a tangible interface could also identified using this technology. This must also include groups of people, for example when a group of users searches for images using the Tangible Image Query. Another issue is the precision of the used technology. If two different interaction devices are near to each other, the user of a device can probably not identified correctly.

In STUDIERSTUBE the users also do not have to login, but the user of a distributed application can be distinguished, due to the setup they are using. It is therefore possible for the users to use different colours and also different

views on the 3D scene. The users can not be identified to be a specific person, but as being a different user in the distributed collaborative setup.

4.8.1 Distributed Undo

In a distributed, multi user environment different kinds of undo commands can be implemented:

- *global undo*: the last command issued by any of the users is undone.
- *local or selective undo*: the last command issued by the user that triggers the undo function is undone.

For the second scheme to work, the user commands must always be assignable to a specific user, and the actions of the users must be independent. This is necessary to satisfy the issue addressed in the work of Prakash [111] who says "In any undo scheme, it is important that undo behaves according to users expectations.". In a collaborative working environment the users should be able to build upon the work that other users have created, which generally means that something like a selective undo is not desirable (also see [69]).

Furthermore the "undo command" can only influence states of the system (and not "real" objects). This argument is discussed in detail in *Undo - Qualities* (6.3, *p.* 110).

4.9 Configurability

Based on the experience with the two systems we found that configurability is an important quality of a system.

Configurable means that the users have control - and *have to* control - how the system interprets the interaction they are performing. To enable the users a quick start with the system a basic configuration is necessary, but the users should always be able to change this predefined configuration to their needs and abilities. Configurability also has something to do with personalised systems. Every human has different needs and abilities, the computer system should be flexible enough to be adapted to this needs and abilities.

Systems can be configurable in many ways, configured by the developers, by users, on-the-fly or before start-up.

Human to human communication shows that after two persons have created a basis for their communication, the language gets compressed meaning

that less redundant information needs to be communicated to achieve an understanding. This process can also be seen in human computer interaction, a good example is the checkbox in dialogs "Do not show this message again". With this option users can tell the system that they are aware of a particular situation and that they will take care about it, without being remembered by the system.

This is in fact a configuration of the feedback "density". Still in most applications a mechanism is missing that allows the users to (easily) turn this feedback back on again. For example if a program has not been used for a long time, the user may have forgotten about this particular situation. It can happen that she does not understand why something is not working. Asking the "WHY question" could be helpful. It would also be easy to implement this functionality by just turning the feedback on for this event.

Another approach that we will implement in *Construct3D*are the "disabled" tooltips, explaining "why" a certain command is not available and what has to be done to be able to issue this command (see *Displaying the state* 5.6.1, *p.* 101).

Configuring the Space. The meaning of *distributed computer systems* in this thesis is that more than one computer and device is used simultaneously. But distributed has also the meaning of spatial distributed, that the space is divided into different workplaces. This is an issue we learned from the work with the architecture students. They take the available space and transform it to their current work practice. A space can be used to discuss, work on a project, or present a concept. These different work-practices can not strictly be separated, and the borders of these different stages in a project are blurred. Therefore the space has to be adapted dynamically to the current needs of the users [13].

This is probably the biggest difference between the ATELIER and STUDIERSTUBE applications. The ATELIER interfaces support this dynamic change in the environment, while most STUDIERSTUBE applications are depending on the technology available in a space, where it is hard to bring all the different devices (tracking technologies, HMD, large projection screen, ...) into one space.

The available space at the different project sites plays an important role. While in ATELIER we had luck to be able to deploy all the technology in one large room at the Academy of Fine Arts in Vienna, the STUDIERSTUBE project is developed at the Institute for Software Technology and Interactive Systems, where the rooms are smaller and the available space is limited. Also the mobile AR system developed [65, 116, 122] needs a special

environment where fiducial markers are exactly positioned.

Configuring a system. In the ACCORD [120, 1] project the research was focused on the home environment. Bringing technology piecewise to the environment and supplying the users with means to connect different devices by abstracting these configurations as jigsaw pieces. Several interfaces were presented, some of them are tangible (paper jigsaw pieces), some of them are displayed on embedded devices, others are screen based (either HTML or special GUI applications).

The focus of the ACCORD project was to explore ways to allow home users to perform certain configuration tasks. Multimodality is not described in detail, although it is available for the users. The problem of contra dictionary settings are also not discussed (see 6.2).

Tailoring a system. In [130] Stiemerling stated following questions

- How can the designer capture diversified and future requirements and how can he distil the necessary range of flexibility from these requirements?
- How can this range of flexibility be implemented technically, leading to the question of software architecture?
- How can the technical flexibility offered by the architecture be made accessible for end users through the user interface?

Tailorability is a key issue in modern software [86, 83]. Tailoring enables the users of a system to adapt functions or behaviours of an application to their specific needs. Writing a special application for each user is impossible, so the applications must be adaptable. To tailor a system still needs work and time for the users they try to minimize the overall work, so they only use the tailoring if they think it will pay off for them [130]. Several incidences are reported where user stated that although they knew they could use tailoring (and how), they did not take the time, because it is not efficient to tailor a function that is only used once and a while.

Tailoring is connected to the actual work that has to be done, but is not really part of it, e.g. if someone wants to write a document, specifying the layout and style of the document will not replace the work on the document itself. However professions like designers actually work on these aspects and not on the document, for them the mentioned "tailoring" is the actual work.

The difference between configuring and tailoring is that the later is changing a behaviour of an application to better fit the users work practice, while

the first actually is part of the work. Still both are a kind of meta interaction with the system. In the work described by Rodden et.al [1, 120] this is actually the motive for a system, while in ATELIER for example it is just a way of working with the system. When supplying tailoring functionality one has to keep in mind that this should actually allow each user of the system to save and restore the tailored settings. This implies also some sort of user management, where the system keeps track of different users and their specific tailored settings (see *User Management* 4.8, *p.* 71).

4.9.1 Studierstube

On-the-fly configuration is not so common in the STUDIERSTUBE applications. The configurable aspects of the STUDIERSTUBE framework are grounded in the abstraction and flexible configuration of the input devices (through the use of *OpenTracker*) and the scripting interface of Open Inventor. This interface allows to configure and change applications without recompiling the source code. Up to now the configuration of *OpenTracker* can only be changed before the start of an application, but a new design, that is currently worked upon, will soon allow to perform on-the-fly changes in the *OpenTracker* framework.

APRIL The (see *APRIL framework* 3.10, *p.* 43) allows to define presentations without actually knowing the hardware setup. The description for the configuration of the hardware is then merged with this abstract description to create a hardware specific application.

Configuring the different hardware solutions used is the responsibility of the *Component Implementer* that will have to take care of different implementations (based on the environment) of the same interaction possibilities.

> The story document acts as a specification for content creation (using the raw material) and component authoring. Components can be re-used from a set of default components or earlier presentations, and for sophisticated interactive presentations new, customized components will be developed.
>
> Integrating the components, interaction tools and content items is the goal of the final phase, story integration. The result is a complete presentation specified in the APRIL language. Independent from the story authoring, for each hardware setup there is a configuration file, describing the arrangement of displays, interaction hardware, speakers, and other aspects of the available hardware.

During story development and testing, it is not necessary to run the presentation on the actual hardware setup; the same presentation can be run in an "emulation mode" on the developers PC. Media content, components and hardware description files are re-usable parts of an APRIL presentation, and are only loosely coupled by the story to contribute to a specific presentation. The same media, components or hardware can be used to tell other stories about the same or completely different subjects.

With this language several requirements are fulfilled:

- Support at least a reasonable subset of input and output hardware and their manifold combination possibilities.
- Support the portability of presentations by separating the presentation's content from the system and hardware specific definitions. Also support desktop based developer setups for creating and debugging presentations.
- Support standards for geometry and media data, and support upcoming industry standards for animation of 3-dimensional content.
- Support the re-use of content by providing a modular structure of presentations. Allow the creation of content archives and the sharing of content between multiple users and setups.
- Provide a well-defined set of interaction techniques and support various input devices to implement them.
- Provide authors with tools to structure their presentations in a non-linear way for the user to explore them interactively. (*from [76]*).

4.9.2 Atelier

Configurability is one the goals of the ATELIER project. Therefore a number of examples of configurability can be found in the ATELIER environment. We experimented with flexible projection planes, that can be configured to achieve different effects – from an open space to an enclosed room. This flexibility poses some problems, that were not addressed in the ATELIER project.

These "problems" are mainly connected to the re-creation of a setup, once the configuration of the room has changed. But as the students preferred flexibility over precision, "recreating" a setup was also seen as "creative" act, where new effects can be achieved (see figure 4.5).

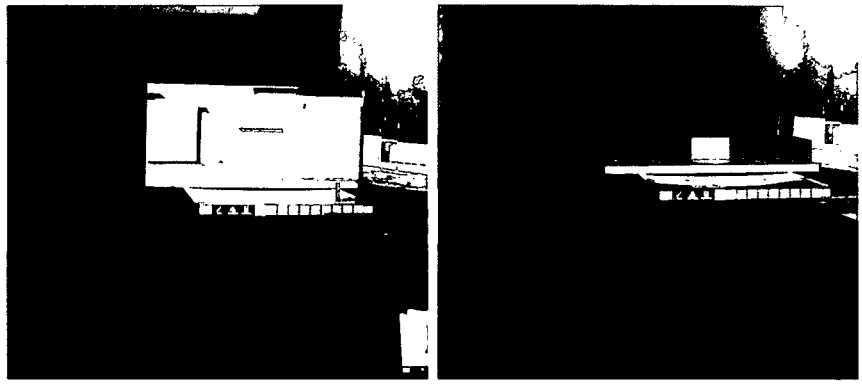

Figure 4.5: Recreating a physical setup during a presentation.

Configuring the environment As part of the students work is actual physical, they explored ways to configure the working space in the Academy of Fine Arts. For a presentation they created various projection planes (see figure 4.6), which ultimately lead to the development of the three display walls and the grid described earlier.

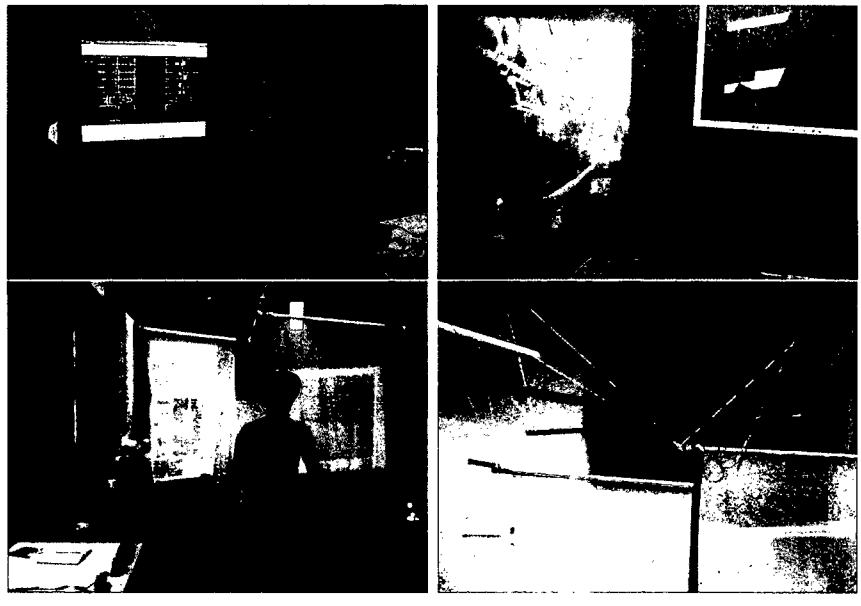

Figure 4.6: Different projection planes in space.

Another way of configuring the physical world was demonstrated by the students, who made use of the "physical representation" of the *physical handles* (6.7.3, *p.* 125), see figure 4.7. They cut the barcode pages into parts, rearranging them by gluing them on their own posters or their models, plac-

ing the physical handles in the physical world. One student created his own presentation list – just like a slideshow – by placing the barcodes neatly in a column, while others distributed the handles in the space an activated them while walking around.

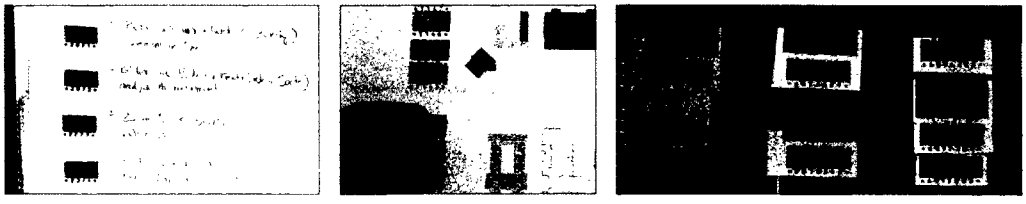

Figure 4.7: Different configurations of barcodes.

Configurator Apart from physically configure the environment, we experimented with different components that provide means for the students to configure the ATELIER computer environment. One way of doing that is for the students to use the three display walls. They "configured" them by placing images or video's on those walls. As the context, in which a model is presented, plays an important role in "seeing it different", this was one way of experimenting with the models (see figure 4.8).

Figure 4.8: Configuring the context of a model.

These configurations were saved and recalled during a presentation, also the students took pictures of their models (including the configured environment) and used these pictures during their presentations to present special aspects of their concepts (see figure 4.9).

Figure 4.9: A student captures a special viewpoint for the presentation.

Predefined configurations of the inner layout of the three display walls could be activated by the students, to adapt the layout to their current work.

Configuring the Feedback. The feedback is also configurable. Changing the display, on which the feedback will be presented allowed the students to reroute the feedback to a display that is near to their current working place. To really make use of the multimodal output possibilities it should be possible to reroute the feedback also to a sound device, that would output the feedback in an audible way.

We did not investigate the cross-modal conversion using text-to-speech in ATELIER, but with the *Construct3D improvements* (5.6, *p.* 100), we will investigate this possibility in the near future.

Chapter 5

Multimodal and Interactive Aspects

Multimodal in this thesis means that different input and output channels are used to interact with a computer system. Human computer interaction (HCI) is no longer limited to typing or moving a pointing device like mice or tracked artefacts. Speech, gestures and the human body itself can be used to establish a communication between the users and the computer system.

By implementing sensors on and in objects those object become an interface. The computer system on the other hand can respond to the interaction in many ways: sounds, text-to-speech, images and videos (on many different displays), haptic feedback - like vibration or wind - even smell [154] or taste [61] are possible reactions of a computer system. Multimodal interaction is not only focusing on multimodal input but also on multimodal output. This chapter also includes *interaction*, which can only happened when stimulating the senses of users (using a multimodal output).

In 1993 an experiment was conducted by Van Gogh TV [146] at a television program in Europe in the scope of the *Prix Ars Electronica*. Multiple users could get in contact via a television show and interact with each other. It was a live show where the television screen was used as a shared resource for the users participating and also for the audience that watched and observed the users actions.

Additionally users could log in to a text chat system. The sentences they typed were displayed in a separate area of the screen. In the middle of the screen some applications could be operated, by using the dial buttons as an input device. The applications were quite simple like a drawing application. Up to three users were able to move a cursor with the dial buttons (2,4,6,8) with the other buttons the colour or the brush type could be changed. Especially interesting were several of the cross modal communications for example

when users of the text chat area were asking about the audio participants location or interests. Phone call users gave hints for the users testing the applications, like which button they should press to get a different colour and so on.

With the beginning of multimodal internet these methods became available for internet communities. The "Yahoo Messenger" [153] allows a group of people to write text, use an audio channel and a live video feed to interact and communicate with each other (a number of other multimodal communication platforms have been developed also). These functionalities are now combined under the buzzword "teleconferencing". Other projects focus on the tele-presence of remote users within a setting.

5.1 Graphical User Interfaces – GUI's

In STUDIERSTUBE the Pip and Pen interface allows to make use of the known widget based interface. For most people (who are *not* computer novices) the widgets on the Pip are instantly understood. The users can interact with them in a known way, but with the advantage that the Pip can be moved freely in space, therefore they can show and hide the interface just by placing the Pip within the viewing frustrum, or when using a HMD based setup by just looking at the Pip.

Also when using handhelds for mobile and personal services a GUI will be presented to the users. The *PUC* framework makes use of this known interface to allow access to various functions of appliances [91]. The most important feature of the PUC framework is that ,based on an abstract description, a GUI is created. For various devices a client was already implemented: handheld's, smart phones, desktop PC, which can be utilized for interaction with the appliances.

In STUDIERSTUBE we integrated the PUC framework, which now allows to create STUDIERSTUBE applications that can be controlled using a handheld. Barakonyi used this feature in STUDIERSTUBE to control the look of the agents, and also explained how this interface is embedded into his *AR Puppet* framework [6] (see figure 5.1).

The automatic GUI generation algorithm was implemented and can now be used to

1. create similar looking interfaces for STUDIERSTUBE and handhelds
2. rapidly prototype applications, because the interface is automatically created

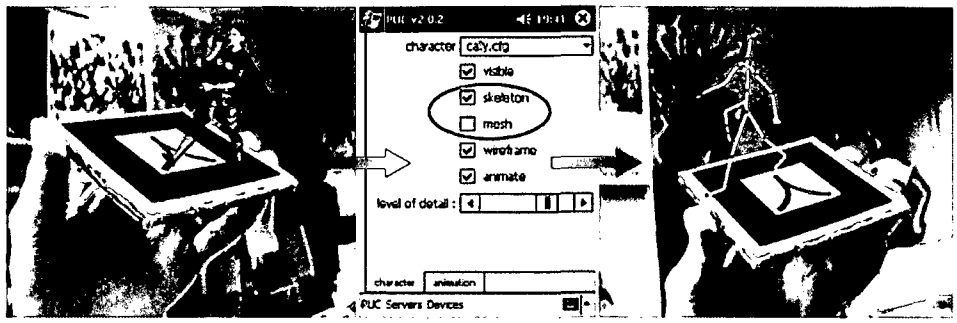

Figure 5.1: Animated character bound to a tangible optical marker and controlled by a PUC generated interface on a PocketPC (monitor + PDA screenshots). *(used with permission.)*

This approach was chosen for the Construct3D *redesign* (5.6, *p.* 100), described in later in this chapter.

HTML as a User Interface In the ATELIER project we implemented a HTML access to the HMDB. In general a HTML interface has many advantages, it can be accessed from any device with a web browser, while at the same time providing a (for most users) known interface. We have used HTML for realizing several functionalities. First the students could upload new content to the HMDB (see *Upload Applet* 3.3.3, *p.* 29), make use of the ontology and the image query search (using the mouse interface described in *Image Query Mouse Interface* (5.2.1, *p.* 86) to find content. By using a printer they could create *physical handles* (5.3, *p.* 89) to the material they found or just have uploaded (see figure 5.2).

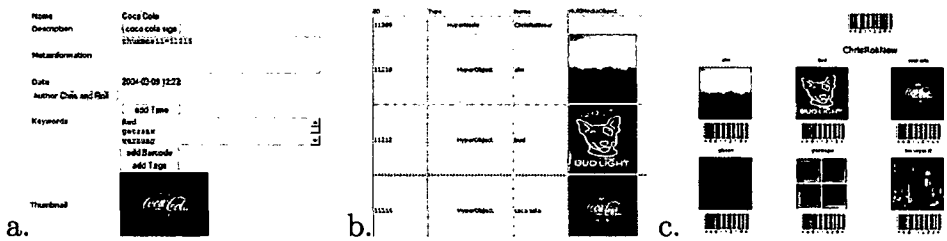

Figure 5.2: a. HTML interface to change an element, b.,c. different views on upload group

We integrated a editing page, where the students were able to add information to the data created, like adding new keywords, or specifying place and author of the content. The way the content was displayed can be changed by the students, to e.g. fit more (but smaller) content on one page, or to change the format to fit a landscape page.

5.2 Tangible Interaction

Our experience shows that non-expert users, who are sometimes afraid of technology, find tangible interfaces less complicated and more user friendly. Descriptions of experiments with the students conducted during the ATELIER project are described in [56].

5.2.1 Tangible Image Query

The Tangible Image Query is an example of tangible interaction in the ATELIER project. We will describe now the special adaptations that we had to implement to change the interface from the conventional GUI sketching tool to the tangible interface (see *Tangible Image Query* 3.6, *p.* 37).

Although the underlying system is arbitrary, and the method can be combined with any query by example method, we needed one method for our implementation. The system is based on the visual image query by Matkovic et al. [85].

Just like most of the image query methods, the method uses descriptors calculated for each image, these descriptors are created in the database during a pre-processing phase. When the user performs a query, a descriptor is created for the query image and compared to the stored descriptors. Various query systems differ in the way how descriptors (sometimes called signatures) are created. The Visual Image Query (the system he have used) calculates descriptors using 1000 quasi-randomly distributed rectangles of various sizes in the image. The rectangles partly overlap. The sizes of the rectangle are chosen according to the contrast sensitivity function of the human eye. Figure 5.3 illustrates the distribution at the first 100, 250, 500, and 1000 rectangles. For each rectangle the average colour is computed, and all 1000 Luv colour triples are stored in the signature. The signature contains only colour information for each rectangle, and the system can not distinguish if, e.g., an orange spot in the middle is a flower or a fish. The only information known is that there is an orange spot in the middle. The exact shape of the spot is also not known. It is sampled using the rectangles, and can never be precisely reconstructed. The comparison of two descriptors is done in the Luv colour space, i.e. for all 1000 triples the luv-difference is computed and a weighted sum, according to the contrast sensitivity function, of these differences is taken as distance function of the two images.

This method was selected since it is particularly convenient for the comparison of user sketches. The sketch is not precise, and actually, only the colour layout matters. However, in order to make it suitable for the new interface, and in order to compare it with conventional input, the original

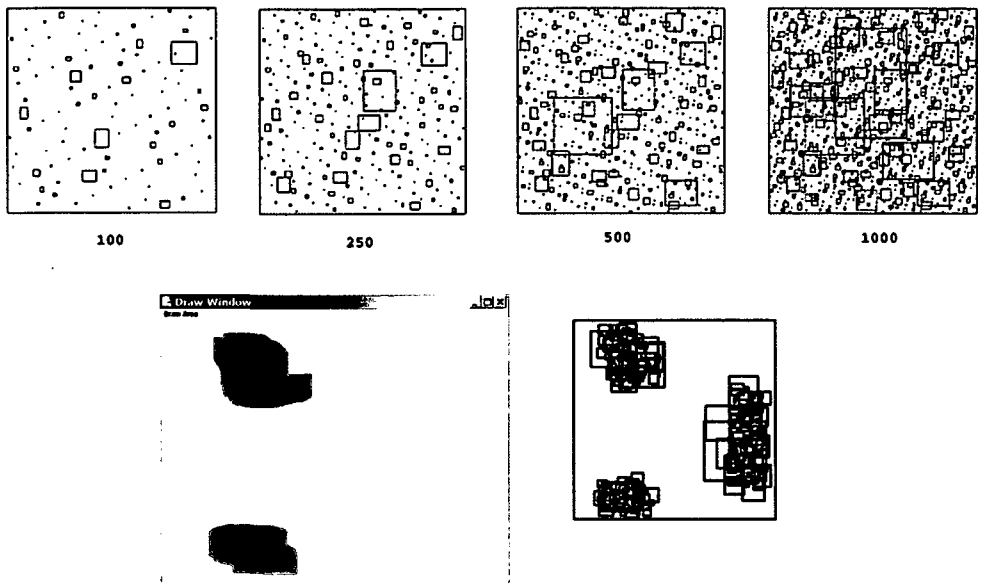

Figure 5.3: Top: Rectangle distribution for the first 100, 250, 500 and 1000 rectangles in the algorithm we have used.
Bottom: The user draws three separate areas, and only the corresponding rectangles are used for this query.

algorithm had to be changed slightly.

Changes to the original algorithm In the original algorithm the descriptor consists of 1000 Luv triples. Comparing two descriptors means computing the Luv difference for 1000 triples. In order to speed up the process, the algorithm slightly was modified slightly. 1000 rectangles are placed in the image, and the average colour of each rectangle is computed. This average colour is then mapped to a colour set consisting of 64 colours. The descriptor consists of 1000 indexes (in the 64 colour set) instead of 1000 Luv triples. The difference between two descriptors can be computed faster using a matrix of predefined differences between all 64 available colours.

In the original algorithm either the whole image or one selected area was compared. This approach had to be changed to allow multiple areas. Only the parts of the image, where the user sketched something, will be used in the comparison. In this way the user does not need to provide a background colour, but only the significant places she remembers. Furthermore, the query starts automatically when the user does not change the sketch for a second, and the results are displayed immediately. Figure 5.3 illustrates an example of a simple sketch and the subset of rectangles used in this case. Of course, support for the new interface had to be added as well.

Image Query Mouse Interface Tests with the original system using conventional mouse input, showed that there are two groups of users. The first group of users, forming a majority, are the users who claim that they cannot draw (or paint, or sketch). It was not easy to encourage them to try the system. They were just saying "I can not draw". Although we explained that they do not need to draw an exact reproduction, but just a red spot here, and a blue spot there ...just a colour layout sketch, it was still not easy to get sketches from them.

The second group of users were users who can draw. The problem with them was that they were not satisfied with the sketch, they wanted to have it "perfect".

Some systems are offering tools for drawing circles, rectangles, and other primitives. If such a system is used for colour layout search results are even more disappointing. Imagine a user drawing a yellow circle, and the system responding with flower images, or even yellow triangles. Of course, the system was not recognizing the shape, but only the colour. The use of 3D primitives is misleading for the users in most cases.

It was clear that conventional sketching is a good solution only for a very limited number of users. Another kind of interface is needed, an interface that is very suitable for sketching, but which is not suitable for drawing. In this way, users who cannot draw will not be disappointed with their drawing results. It is impossible to draw with that interface anyhow, and for the same reason the users who can draw will not try to draw perfectly.

New sketching interface The whole setup consists of a table with a semi-transparent white glass plate. There is a set of colour cubes, and the users can arrange them on the table in order to make a sketch. A simple web camera is mounted under the plate, which is used to retrieve the colour layout sketch. This sketch image is then used as a query image. Figure 5.4 shows a part of the setup with the table used for sketching. It was common practice during our experiments that users "draw" together. They stood around the table, and instead of the others instructing one user what to do (which was common with the mouse), the group could work together. The collaboration is another important quality of the cubes interface. Furthermore, not only the cubes can be used to sketch. As soon as a bowl of fruits was placed next to the table, some users used oranges and apples as sketch tools.

Vision based colour sketch The Crayon project [32] provides a good overview of the current state of vision based interaction. In the project the researchers use a camera for hand tracking and explored the field of colour

Figure 5.4: Students experimenting with the new interface.

calibration and machine learning. Our approach is related to their work in the respect that we also extract colour information from a live video stream.

Various problems that are related to colour vision had to be faced. First tests showed that for certain colours (especially cyan and grey) that were desirable, no stable calibration was possible. This is because web cams provide compressed video information and use optical sensors that are optimized to capture images of faces. The main usage of this kind of camera are video meetings, so the red part of the visual spectrum is covered quite well, but blue and contrast are not of high concern.

Hardware setup To reduce the problems that come with computer vision, like changing ambient light and dynamic in-camera adaptation, a setup where these external interferences are reduced was created. The camera was mounted underneath a semi transparent surface, on which the coloured cubes were placed. Also a light source was installed underneath this surface to ensure proper lighting conditions. The setup was surrounded by a non transparent casing leaving only the surface visible to the users and exposed to the ambient light in the room. It was possible to achieve good results with a static calibration of the colour detector with this setup. The output of the query was then displayed in the ATELIER space on the display wall.

The need for such a special hardware setup might be considered to be a drawback of the system. Not everyone has the possibility to allow extra space in the office for such an installation. In such a case a simplified system consisting of a web cam pointing down on the desktop (the real desktop),

and a set of colourful cubes, game stones, pieces of paper, or similar things can be used to interact with the system. Furthermore, the use of flatbed scanners for this purpose was briefly exploited with advantages like better colour and contrast but also drawbacks like increased response times and reduced interactivity.

The steps of the colour sketch retrieval Two approaches were implemented to create our setups. In the first implementation the colour segmentation was applied at the vision part of the system. First an image is grabbed from the web cam, then a colour segmentation is performed and finally a colour indexed image is sent to the image search engine. The colour segmentation was implemented using the HSV (hue, saturation, value) colour space. For each colour (white, yellow, orange, red, green, blue, magenta, black) ranges for the HSV values are specified. Using a simple filter, regions in the grabbed image that have colour values within these ranges are copied to a colour index image. The colour index range was defined from 1 to 8. Zero is being used to indicate that none of the colours was detected. This indexed colour image is then sent over the network to the search algorithm.

In the second implementation a background subtraction approach was used to filter out the parts of the video stream that have been changed or added by the users. This approach sends a full colour image (with reduced size) and an alpha channel (specifying the regions that are not background) to the search engine.

Both image segmentation approaches have their advantages. The colour segmentation provides better results in respect of removing the background and not used areas. Because the background subtraction algorithm dynamically updates the reference image it is more stable to ambient light changes. Also the background subtraction allows the use of more than 8 colours, because the colours are not mapped to one of the indexed colours of the cubes. At the same time the network traffic increases as more data has to be sent to the search engine.

Independent from the segmentation method a"change" parameter is extracted from the live stream, measuring how much the image has changed between two updates. A high value indicates that the users are currently changing the sketch or just moving the hands within the observed area (for example to point out certain regions and compare them with the results). During this period of vivid interaction no update is sent to the search algorithm, not even the colour segmentation or background subtraction is evaluated. Such intermediate results would confuse the users and also distract their concentration from the task of creating or changing a sketch. When

the "change" parameter drops below a certain value the image segmentation is activated. If the difference between the resulting sketch and the previous query to the search algorithm is above a certain value (indicating that the vivid change in the video stream was not just moving the hand but also moving some objects), the new sketch is sent to the search algorithm. This makes it possible to achieve fast update rates, as no unnecessary video frames and queries are evaluated.

Selection of colours for the tangible interface After our first tests with the colour segmentation we had to realise that not all desirable colours would be sensable with the setup we have chosen [12]. In the previous implementation of the colour layout search, about 50 different colours were available for the users. For the web cam and colour cubes based interface we had to reduce the colours to the basic primary colour set.

White and black as representatives for the grey spectrum and red, green, blue as the basic colours. As yellow, orange and magenta are also remembered colours we provided them too. Cyan as a mixture of green and blue seemed to us to be also a very important colour. But because of the colour dynamic of web cams this particular colour is hard to be extracted from a web cam image. In sake of stability of the colour detector we did not provide this colour. Our user tests have shown that cyan was not requested by the users when they had to sketch the images.

When using the background subtraction approach we noticed that with a standard web cam it is not possible to capture the full colour spectrum, which means that the cyan colour cubes were more often mapped to grey than to the cyan colours. The results and tests we conducted with users are described in *Tangible Image Query* (6.6, *p.* 119).

5.3 Physical Handles to Digital Content

The mediaBlocks project [144] presented the use of physical markers as handles to digital media and a number of appliances was presented. The project was influenced by the metaDESK/Tangible Geospace prototype [60, 143] that introduced the *phicon concept.* Tangible Geospace was developed in part to explore physical instantiation of the GUI metaphor, that concentrated on tangible control of a augmented space. making use of tangible interaction to navigate trough information space was also described in the Navigational Blocks paper [22]. Using physical objects that represent data queries, the Navigational Blocks interface allows people to explore the relationship between topics in a database and create simple and complex queries - but no

updates of the database.

The whiteboard-based mediaBlock functionality draws upon an earlier whiteboard TUI called the transBOARD [60]. The trans-BOARD used paper cards called hypercards as physical carriers for live and recorded whiteboard sessions. The hypercard interaction was based upon barcode wanding. They concentrated on managing the tangible interaction in creating and manipulating the connection between physical handle and digital content.

The ubiquitous computing vision of Weiser [150] speaks to moving computation and networking off the desktop and into many devices within the physical environment. Dynamic association between digital properties and physical handles through the *tray device* in described in [37]. The often cited Bishops Marble Answering Machine [26] demonstrated the use of passive marbles as "containers" for voice messages. Later work by Bishop prototyped an early object-GUI gateway and demonstrated physical objects associated with diverse digital content.

The LEGO structures described in Molenbachs LegoWall prototype (discussed in [36]) are used to contain information about ocean-going ships. These objects were combined with *display* and *control* objects that could display shipping schedules and send this data to hardcopy printers, etc.

The AlgoBlock system uses the manipulation of physical blocks to create computer programs [133].

5.3.1 Barcodes and RFID Tags

Barcodes have a quite a long tradition in the economy due to their qualities in respect of cheap production and available devices. In supermarkets the barcode of a product is a physical handle to the price, but also to manipulate the warehouse database. Using barcodes to access digital content has recently been described in [74, 75].

It the ATELIER project we included barcodes and RFID tags to represent content of the HMBD. The barcodes have the obvious qualities of being cheap (a normal printer is enough) and easy to be produced. The RFID tags on the other hand are more suitable for an environment like a working place, as they are robust against various liquids (e.g. coffee, energy drinks, etc.) and dirt (like glue, left over material from creating models, etc.), see figure 5.5. As described in Sensitive Sample (5.4.1, *p.* 92) the models itself can be used to access multimedia content.

In STUDIERSTUBE the idea of physical handles is investigated, but because a global storage is missing, this feature is only usable in short presentations and not a main research area. The fiducial markers of ARToolkit are

Figure 5.5: RFID tags and barcodes in use.

mostly used to supply "location based" information rather than direct access to digital content [65].

Often the applications in STUDIERSTUBE combine the interface with the actual handle to the digital content like e.g. described in [5], where the fiducial markers are used in a teleconferencing application, that allows the users to position the data in the live-video stream. Additionally by choosing between different markers they can decide what model to show.

5.3.2 Persistency as a Quality of Tangible Interfaces

An important quality of tangible interfaces is that they can be used in a persistent way. A current discussion in the community is about: which qualities a tangible interface must have to be a "real" tangible interface. In this thesis we are not following the distinction between "graspable interface" (meaning that the interface has a physical representation that can be manipulated) and "tangible interfaces" satisfying all the qualities. The description of those qualities is discussed in detail in the PhD thesis of Eva Hornecker [54].

Persistency means that the interfaces keep their states even if the computer system is turned off. Furthermore the interfaces can also display their state, therefore it is perceivable by the users without the need for an augmentation (this is only true for "real" tangible interfaces). Examples are:

- RFID tagged objects on a tag reader.
- The *ControlCube* (3.5, *p.* 36)
- The Tangible Image Query – as it can be used to perceive the current sketch.

This quality of tangible interfaces – to display the current state – and therefore providing feedback for the users, makes this kind of interface so useful.

5.4 Gesture-based Interaction

Using gestures to interact with a computer system is a ambitious task. Gestures of humans depend (among other factors) on their personality, physical abilities to perform certain gestures, and a lot on the culture they have. Often cited in the related work is the example of the culture where moving the head up and down (nodding) means "no" and moving it side to side means "yes". Still the research in this field is making progress, while keeping the cultural differences in mind.

We will now describe the two applications where the interaction is based on gestures:

- The *Sensitive Sample* and the *MaterialKammer* concept.
- The *Invisible Person* project where the human body is utilized as an input device.

5.4.1 Sensitive Sample

The *Sensitive Sample* implies a violation of the *affordances rule* (2.4.3, *p.* 19) as the object, which is enhanced, is not displaying the embedded functionality. This stresses the inspiring character of the application area, where surprise is used as an element of a presentation. Another approach – following the rule of *affordances* – was to use barcodes (see 5.3.1).

To enable the physical objects to detect various actions, tilt and touch sensors are installed inside. Tilt switches are needed to detect user actions like shaking hard, shaking gentle and turning upside down. Touch sensors are used to recognize user actions like touching the object, stroking the object (more sensors are used here and responses from them can be interpreted as stroking in various directions and speeds) and knocking on the object. Figure 5.6 illustrates the actions that can be recognized by the *Sensitive Sample.*

The ControlCube was realised using the board with the micro-controller described in *Sensitive Sample* (4.5.1, *p.* 63) and 3 tilt switches. The three switches are connected orthogonally, and placed inside the cube so that axis of the switches do not correspond to the cube axis. When a particular cube-side is facing up, the three tilt switches will have a unique state. With this simple setup it is possible to detect the side of the cube that is facing up,

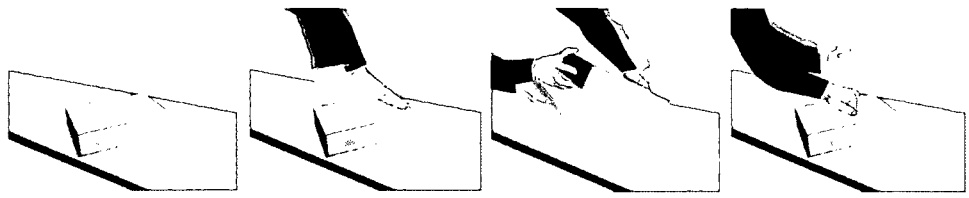

Figure 5.6: Some of the actions that can be sensed by the Sensitive Sample.

but also whether the cube is currently turned (the tilt switches will change their state).

5.4.2 MaterialKammer concept

Based on the *Sensitive Sample* idea we developed a concept for supporting the students in investigating different materials for their models and concepts (see *MaterialKammer distribution* 4.5.1, *p.* 65). This concept is a result of our field studies at the Academy of Fine Arts, that showed that students carefully investigated materials, by touching them and get a feeling for the qualities of the material.

Scenario of Use.

> The ATELIER room is equipped with projectors, sound devices, fans, lighting equipment, etc. The users enter the room and start exploring the materials. Depending on the way how the users interact with the samples, an atmosphere in the room is created. If someone plays with the wooden samples, and does it in a quite gentle and smooth way, an atmosphere of a wood in spring could be created. Another user shaking the wood samples more erratic can trigger a saw-mill like atmosphere. If the users play with different kinds of samples, like wood, bricks, glass,... completely different moods will be created.

In order to make the whole room inspirational, and not predictable, a lot of parameters influence the choice of the presented materials. Videos, sounds, still images, but also wind coming from the fans and controlled lighting are used as output devices. Such a setup represents an unconventional way to explore materials. The connection between materials, actions and presented materials is not always clear to the users, the surprising element that emerges plays an important role in the inspirational process (see figure 5.7).

Another use of the samples would be to use them for texture selection. For example a wooden board can be used for wood texture selection. When a

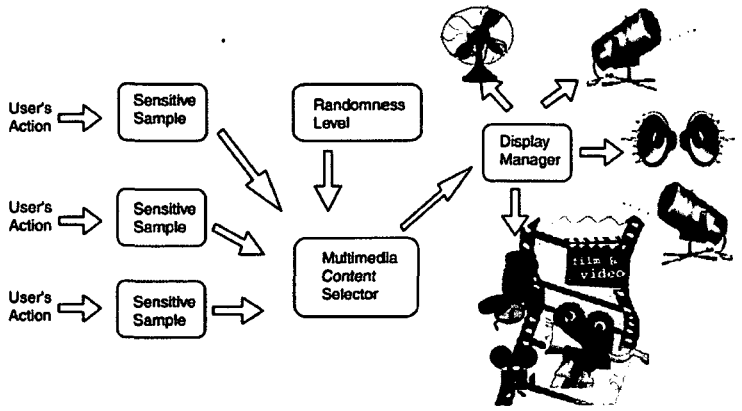

Figure 5.7: The concept of MaterialKammer interaction.

user takes the board a wooden texture is displayed. Now simple by touching the board on the right or left side the textures can be browsed. When the desired texture is found, the user knocks in the middle, and the texture is being exported to another application. In a scenario like this a stone would be used to select stone textures, a brick for brick textures, etc.

The multimodality of this concept is not only grounded in the gesture based interaction, but also in the multiple ways the senses of the students are stimulated – they can see, feel and smell the material they are interacting with. As the real materials are used there is no need for generating these impressions using a computer system, which would anyway be impossible with the technology available.

5.4.3 The human body as an interface

Using the human body for interaction with a system has many advantages. The interaction seems to be natural, because everyone is used to their own body. There is no need for devices that can be broken, malfunctioning or removed by someone. The users body is always available and can be controlled by the users - it is probably one of the best interfaces for human-computer interaction. The main drawback is that it is quite hard to detect gestures and body poses in a normal environment. A lot of experience and common culture is needed to understand the body language of other persons. Many researchers try to analyse body language and the meaning that lies within the movements of the human body. Stereo vision and visual clues to determine the distance, interpretation of the body language, knowledge how people react in special situations and so on are still beyond the capabilities of computers and the designers of applications. Still research progress on

all these problems is being achieved. Even if all those mentioned issues are solved, there is no common culture between humans and computers that could be a basis for a common body language. In the "Invisible Person" project we tried to follow the principle of body language.

Adding some new forms of advanced interaction was our main goal. The system in the TMW was realised to allow direct communication between an artificial intelligent (AI) and humans without the need of custom input devices. The enhancement of the interaction should use only the available interface. As the original system proved to be a well-working environment supplying human-computer interface [105], an enhancement of the system by adding more advanced interaction was desired. We decided to use the well-known interaction frame of game playing (see *Interaction frames* 2.4, *p.* 16), so that the situation and the meaning of the users' actions are easily understandable. We defined that the minimum interaction tools, players must have for our games, are "selection" and "mark". Thus all players must be able to select a region on the playground. After that selection is done, a "marking" action referred to by us as "click" must be possible. Both interaction tools must be very reliable to encourage the users to play and not frustrate them with a non-working interface. We introduced a new graphical element to the system serving as an interface for the game-relevant communication between the IP and the users.

Game-pads. Bowman and Hodges [15] proposed that the guidelines of Norman [97] should be applied to interaction objects in virtual environments (VE) (see *Affordances, ...* 2.4.3, *p.* 19). The game-pads we have used in the installation satisfy all four criteria which are: affordances, feedback, constraints and good mappings. To satisfy the affordances criteria, the object must be able to inform the user of the way it can be used. Usage of the game-pads is demonstrated by the IP, when it makes the first move. The IP uses the same posture that is registered by the system for a user "click". Repeating the gesture, if the user is not marking a field within a certain time and giving additional hints for usage. Feedback is realised by highlighting the game-pads when a player moves over or stand on them (see figure 5.8).

They change both colour and shape as they are highlighted or marked, displaying a clear difference in their state. Constraints refer to the limitations on the use of the objects that help the user to use them in a proper way. As the game-pads do not give any feedback to players who are not allowed to make a selection, this supports the users in understanding that they have to wait until the IP has made his mark. Good mapping requires that the conceptual model, on which an object is based, is easily understood in the

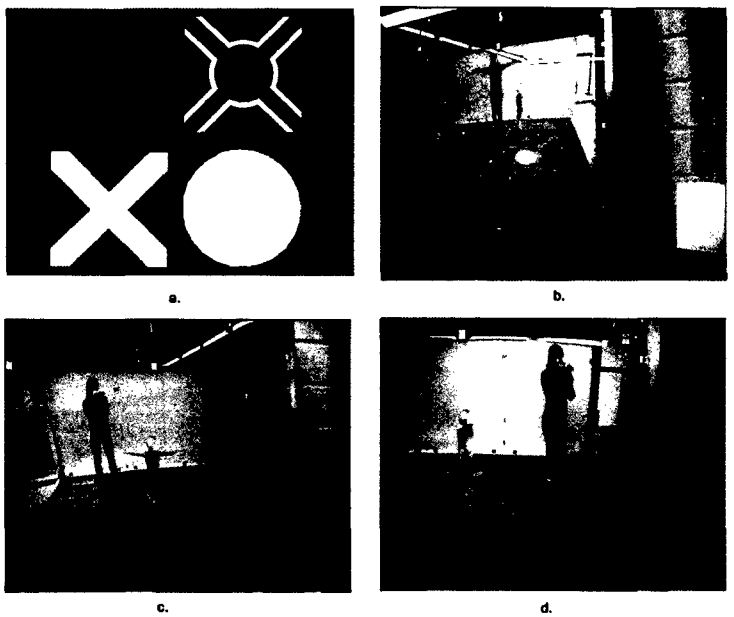

Figure 5.8: a. Game pads shoving default pad, selected pad, clicked pads; b. User has selected a pad and just makes a click; c. IP makes a click; d. The user selects next pad

specific usage model. The game-pads are based on an easily understandable metaphor of some floor element that is sensitive to people standing on or moving over it, as they exist e.g. in underground stations by the escalators. For the enhanced system in the TMW we came up with two different types of games. The first type deals with games like "TicTacToe" or "Senso" that allow only one player to be active at a time. The second type is more complex, by allowing all participating parties to select pads at the same time. We found a generic representation of the information that is needed for the AI to play with the users.

Vision The main task of the implemented vision system is to find the silhouettes of users seen from the front, to estimate the users' position and posture on the stage. The silhouettes from the front camera are needed to implement efficient occlusion in rendering. The position on the stage is used to add depth information (needed for occlusion) to the occlusion masks and to supply the AI with the users' position. This information is also used by the AI to avoid collisions with the users. The posture recognition is necessary for user interaction with the system. As stated above, selection and click are necessary to implement user interaction within our games.

We implemented an interface where the user position will be used for

selection, and a predefined posture that is used for the click. The posture chosen is "two arms spread apart", since this posture showed to be the most reliable one. The vision system runs on a dual Pentium PC running the Linux operating system with two video grabber cards and two cameras. The first camera is the same one used for the output video, and it is placed in front of the stage, just above the display screen. The second camera is positioned above the stage, pointing downwards, and this camera is used by the vision system only. As stated above, the whole system is intended to run in a publicly accessible museum hall, which has a frosted glass roof. Due to the frosted glass roof the lighting conditions change practically all the time. These changing lighting conditions, the demand to design a stable and robust system for operation in a publicly accessible place and the demand to design a real-time system were the three most challenging requests towards the vision module.

Basic Vision. The basic vision is implemented using the OpenCV library from Intel [101]. The basic task is to determine which pixels belong to the foreground (user) and which to the background (stage). The basic principle used is background extraction, which is a common technique in people-tracking systems. We updated the background image using a running average of the input images. The current image is then compared to the estimated background image. The pixels where the difference is greater than a certain threshold are considered to belong to the foreground. The threshold is estimated per pixel using the standard deviation of its brightness distribution over the last 200 frames. Once the pixels are classified, the background image is updated using different update rates for background (faster update) and foreground (slower update) pixels. The foreground updating is necessary due to constantly changing lighting conditions. The foreground pixels contribute to the background as well, in order to make the system more robust in case of error. If a sudden change of lighting is interpreted as a foreground object it will become part of the background, and will not remain foreground "forever". On the other hand, if a user stands still for a long period of time, he or she will become part of the background, and once she moves there will be two foreground objects. The basic vision described above is a standard technique. We added some new features to the vision system in order to fulfil our requirements.

Advanced Vision. Due to constant lighting changes and large range of possible lighting levels, the camera iris system must adjust itself automatically. If a sudden lighting change occurs; for example a cloud is passing by

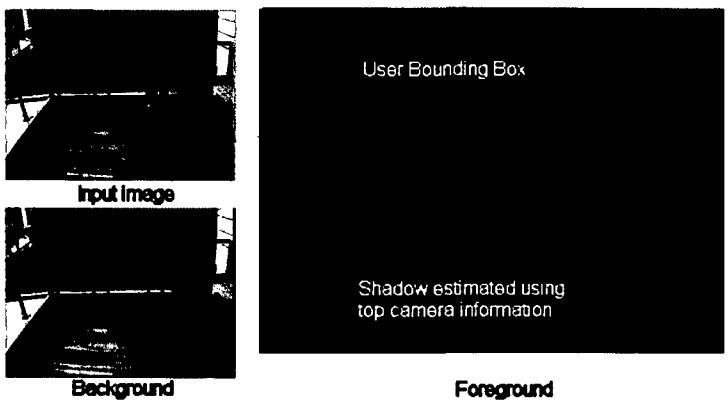

Figure 5.9: Vision module extracts the foreground pixels from the input image, and estimates the shadow using the top camera information.

and covers the sun, the pixel intensity in the background pixels can change so much that system assumes the whole background to be foreground. In order to overcome this problem, a kind of software iris balance is implemented. The light is measured on five spots, which are unlikely to be covered by the users, and the current image is made darker or brighter depending on the readings in these measuring areas.

The vision system runs parallel for each camera, using the algorithm described above. When the mask images from the front camera and from the top camera are extracted, both mask images are used to compute user masks as seen from front camera and user positions on the stage. The original front camera mask very often contains a shadow. The shadow can be cut out using the information from the top camera. The user position is known, and the floor level in the front mask can be computed using the user position information. Figure 5.9 illustrates the result of estimating the background, the foreground and shadow in the foreground image. Furthermore, the top camera information is used to divide the front blob in cases where two users are standing one beside the other from the front camera's point of view, but they are actually standing apart (behind and beside each other). Once the mask images are merged, a new final mask image is produced containing masks with depth information. These masks are used for occlusion and posture recognition.

Posture Recognition. Besides the depth information for the masks, a posture is assigned to each mask as well. There is a predefined set of postures from which one (or "none" if the posture is not recognized) is chosen and added to the mask. The bounding rectangle of the mask is divided into 25

sub-rectangles, and a fill ratio is computed for each sub- rectangle. The fill ratio is the ratio of foreground pixels and total pixel count in a sub-rectangle. These fill ratios are compared with the fill ratios of predefined posture masks, and the posture with the smallest mean square error is considered to be the right one. If the mean square error of the closest match exceeds the maximum allowed threshold, the posture is considered to be unrecognized. The system recognizes postures such as left arm stretched out, right arm stretched out, both arms stretched out, left leg stretched out, etc.

5.5 Sound and Speech Interaction

Sound as an output facility is a quite complex media. When a sound is played through loudspeakers, all people in the room are influenced by it. They do not have to look a specific direction as the audio sense of humans works ubiquitous, which can be useful or disturbing [97], *p.* 102-104.

The use of headsets allows to limit the influence of the sounds to just one individual. For a presentation situation loudspeakers are a better choice.

In an application design it is important to keep in mind that sounds can be more annoying than visual output, as the visual sense can be better controlled by and individual - by looking in a different direction or closing the eyes. Sound can only be blocked by moving the hands to the ears.

A interesting project that builds upon earlier work in the field of speech and gesture-based systems is presented in [106]. They present a virtual environment where users can use pointing gestures combined with speech input to select and move virtual objects. Their work is based on Avango [139, 140] and incorporates a language interpretation engine that is able to resolve ambiguous references from both – gesture based and speech – interaction inputs.

In the two projects we also investigated the use of sound and speech to interact with the system. In the STUDIERSTUBE project, due to the integration of Microsoft's Speech API (SAPI) [87], also speech output is available for the software developers. In combination with the *AR Puppet* framework even lip-synchronised computer generated facial agents are possible [7]. The SAPI framework can also be used to perform a quite limited speech recognition. Kaufmann tested the speech recognition (that was implemented in the *OpenTracker* framework) with the *Construct3D* application. He mentions in his PhD thesis, that this interface is more suitable for a single user environment as background noise, especial other users talking to each other, disturbs the speech recognition. The quality of the microphone used and the sound-card are also important parts of a speech recognition setup.

In ATELIER we shortly (actually just a few hours) investigated the use of

speech recognition to access the HMDB. Although it seemed to work quite well in the test setting, it would certainly fail in the normal work environment – where various groups of students discussed different issues in the ATELIER room, while working on their projects.

Although we implemented also sound output components most architecture students were so concentrated on their models and the visuals that they did not explore the use of sound. This may also be grounded in the fact that the second field trails (with the more elaborated prototypes) were conducted with first year students, while our initial field trails were done in collaboration with master students, who actually used sound scapes in their presentations.

5.6 Construct3D improvements

As stated in Construct3D (3.13, *p.* 50) the feedback and self-explanation of the application is limited. To improve this issues for the next evaluation we discussed how to proceed. Additional features requested by the teacher's included more control and some privileged access to the system (more based in the social setting than on practical needs for teaching).

Working on the system we decided to also include some changes to the internal functionality and improve the structure to create a solid ground for future enhancements. As described in [69], tests combining the application with the scripting functionalities of APRIL were performed in the past. The interface that is needed to be able to integrate *Construct3D* with APRIL is also needed for the PUC framework. The PUC framework – developed after the last evaluation – provides a number of features that can be utilized.

Device independent modelling of the interface that allows also to incorporate dependencies between the states.

Automatic GUI generation that can be used in the AR environment as well as on a PDA or a desktop PC.

Additional text for states that can also be rendered on the different devices and can also be used for speech synthesis.

Scripting Interface that allows rapid prototyping (in conjunction with the Automatic GUI generation), by adapting only a text-file without the need for compiling the code.

5.6.1 Displaying the state

Many commands in *Construct3D* depend on the internal state of the application (like how many points are selected). The fist step is to visualise the availability of commands. Until now the application will not display which commands can be issued. A user can highlight (by moving the pen inside button geometry) and activate (by pressing the button on the pen) any widget on the pip. The application just does not react on the invalid commands (a debug message on a screen – not visible for the students – noted that the command is currently not available). Therefore it is not possible for the students to perceive the state of the application (violation of the *constraints* rule, see *Constraints* 2.4.3, *p.* 19).

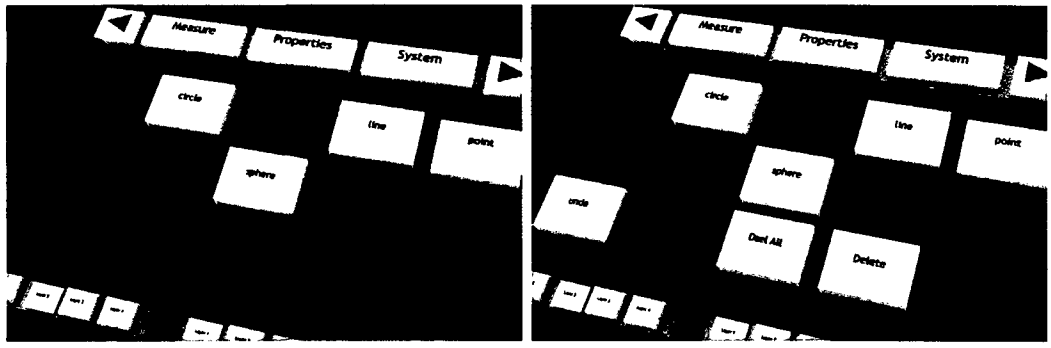

Figure 5.10: Displaying the possible commands by disabling inactive commands

By disabling the unavailable commands (the disabled state is rendered transparent – in analogy to the greyed widgets in common GUI's), the students can now see, which commands are currently available (see figure 5.10). The disabled widgets also do not react (with highlighting) on the pen, which stresses the difference between enabled and disabled state.

5.6.2 Tooltips and Speech–output

By exploiting the functionalities of the PUC framework additional information can be displayed. We implemented tooltips (well known from conventional GUI's) to display additional information for the widgets. To go beyond the well known tooltips, we also display different tooltip text depending on the state of the widgets. With the help of these tooltips the students can understand what they have to do, to make a specific command available.

The tooltips can be tailored by the students (and teachers) to increase or reduce the time they have to highlight a widget before the tooltip will be displayed (like in conventional desktop applications). Setting the timeout to a

low value will display the tooltip nearly immediately after entering the widget with the pen, setting it to a high value will practically disable the tooltip (no student holds the pen within one widget for more than 10 seconds.)

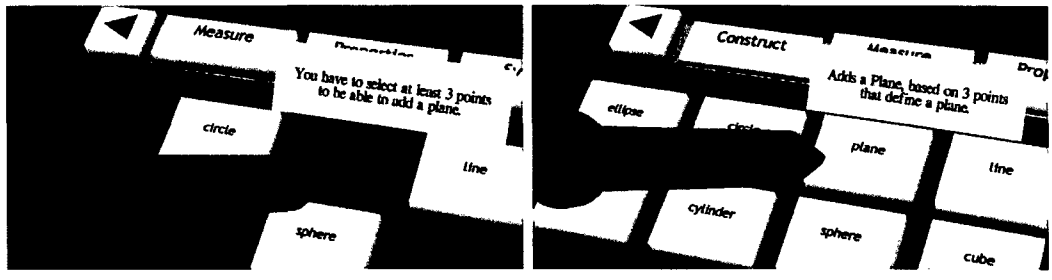

Figure 5.11: Displaying different tooltip text, depending on the state of the widget. Left: disabled tooltip text, Right: enabled tooltip text

The students are, beside the HMD, equipped with a headset that allows them to communicate with remote students and hear pre-recorded instructions for an exercise. We will make use of the head sets to output a speech synthesis of the tooltip text, because the tooltip text can be hard to read wearing a HMD.

The speech output can also be tailored by the students to enable, after some time working with the system, to turn the speech feedback off, which can get annoying when using it for a longer period.

5.6.3 Speech input

The PUC framework can also be used to include speech recognition (based on Microsofts Speech API (SAPI)) for the defined interface. We are yet not sure, if the already stated problems ([69], *p.* 60) of background noise (especial in a multi-user setup) and the detection rate of the speech recognition are yet sufficient. As this field of research is hardly worked upon in the computer interface community, we may see a workable solution in the next years. By using the PUC framework to describe the possible commands the recognition is reduced to just a number of recognizable words, which could eventually work out.

Chapter 6

Results and Discussion

In this chapter we present the results that have been achieved so far using the both systems. A discussion about the issues that we discovered during the implementation and tests of the prototypes is also presented. The chapter *Important Functionalities of a system* (7, *p.* 130) will address these issues by explicitly formulating them as issues that should be tackled by the designers of a framework.

First the general results and findings are presented, then the results of some applications are discussed.

6.1 Qualities of Input Devices

Often user interfaces are discussed and evaluated on the basis of usability, understandability, stableness, error prone .... But sometimes although all these criteria are met, an interface can be wrong. Each interaction method has different qualities, and to exploit these qualities is often overlooked. This is especially true if a range of interfaces can be used to perform a certain task. Sometimes a user interface, even when it is less suitable for a task, is never the less more appreciated by the users, depending on their abilities to use a device and personal settings and configuration [97].

A rather strange example is sending short text messages with the cell phone. Although the cell phone is designed to be used for audio conversation, some people prefer sending a text message (with a most inconvenient interface for editing texts) than calling the other person.

In the ATELIER and STUDIERSTUBE project we often faced the opportunity to choose between different solutions. Based on the discussions with both teams following *qualities of input devices* were collected.

Stableness. Some of the tangible interfaces that are published are merely in a beta state, meaning that they are not ready to be used in a normal working scenario. Too much "manual" work is needed to get them running or to keep them running. The actual amount of work that has to be invested to develop an interface from a lab setting to really work in a real-life scenario is sometimes underestimated.

Some properties can be summed up with the word "stableness":

- *Physical stableness* – is the device constructed so that it does not break under normal use scenarios, there are good test procedures available for industrial device testing. The first *Sensitive Sample* were too fragile. Also in STUDIERSTUBE for the development of the wireless pen we needed some iterations to produce a pen that satisfied this criteria (see figure 6.1).

- *Data stableness* - does the software driving the device produce stable data? This can be a problem with vision based devices, where the data extracted from the live-video stream can be unstable. Short breaks in the data, indicating that the object can not be detected, have to be taken into account. Sudden changes in the position or orientation data also can be caused by unstable tracking.

- *Environment stableness* - is the device still functioning even if the environment is changed? Examples are that devices are influenced by metal (magnetic tracking), light (vision), or conflicts with other devices using the same radio frequency. The *Texture Painter* and *Tangible Image Query* are good examples for devices were these issues are investigated.

- *Error prone* - if some device only works 80 percent of the time users try to use it, it is probably the wrong choice. This issue was investigated in the *Sensitive Sample* project, in detail in the development of the NavigationBox.

Often the authors argue that there are still improvements to be done, before that device can really be used, but before these improvement are not realised, it is hard to tell whether it is possible at all to overcome those problems.

Shared environmental requirements. Some devices require specific environmental settings that sound plausible for a specific setup using this device. But if more than one specific device should be used in a setting, allowing the user to choose between different devices, the requirements can be contradictionary.

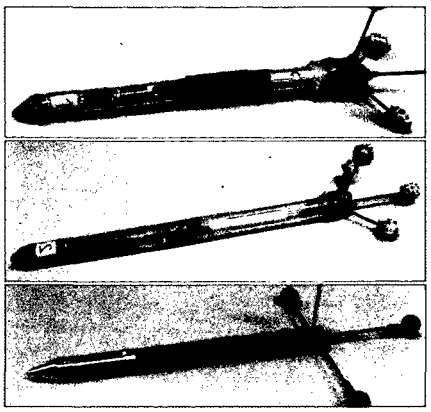

Figure 6.1: Physical evolution of the wireless pen, ending up with a pen that is robust (professionally designed and manufactured).

The devices that we wanted to combine in the ATELIER project, this was mainly the light situation in a setup where the *Texture Painter* (3.8, *p.* 40), the Display Table *Display Table* (3.3.2, *p.* 28), and ARToolkit markers were combined.

To achieve a good image quality of the table surface the incoming light had to be minimal, for the web cam to register the ARToolkit Markers the light had to be turned on. Finding a good compromise between those extremes needed some tweaking. In theory these issues can be easily solved by for example using infrared markers and an infrared camera and therefore separate the requirements for the different devices in the light spectrum, actually diminishing the problem of a shared global variable (incoming light).

Setup and calibration requirements. Some devices like e.g. The Tangible Image Query needs some preparation work, it does not work "out of the box". This is probably true for all vision based devices that are not commercial like [2] or especially created for changing lighting situations [113]. Also important is the time of the calibration – does the device have to be calibrated every time the users want to use the device, or is the calibration only needed when a setup is built up in a new place or environment. Also a "user based" calibration is sometimes needed: e.g. for speech recognition, a user in general has to finish a training of the speech recognition system that can take up to 20 minutes [57].

Some devices do not need any calibration e.g. keyboards, mice or other environment and user independent devices. Although the mouse can be tailored to better fit the needs of the users (like switching the mouse buttons for a left–handed user). So if a left–handed person comes to a different work-

place, sometimes some setup must be performed, but this is related to the preferences of the user and not to requirements of the device.

Tailoring possibilities. Changing the mouse button order (for left-handed users) or even the direction of the mouse pointer relative to the movement of the mouse can be an issue (some users prefer also an inverse vertical axis). These are tailoring activities that help the users to use the device in a manner that suites their needs and abilities best (see *Tailoring* 4.9, *p.* 75).

Time delay. Also an important quality of an input device is the time delay introduced by the device itself – excluding the time to process the input and display a result. For example the *Tangible Image Query* needs some time to process the web cam images and also uses timeout methods to realise that an input is stable. Also in the *Invisible Person* this time delay can be irritating for the users, who normally do exactly the wrong thing namely changing or retrying the input instead of waiting. With some devices – special those based on timeouts – it will never be possible to reduce this time delay (see *Time delay in Invisible Person* 6.5, *p.* 117).

Persistency. A very important quality of an interface device is the level of persistency they provide. Most of the input devices actually do not provide any persistency. This feature is then implemented in the software or the application. By using save and load functions a kind of persistency is created that allows the users to work incrementally. When working with "real life" tools persistency is kept by the physical basis of the tools.

A "real life" example: On a desk a paper is not removed after the user of this desk has left the room (or closes a window). This also is an advantage of tangible user interfaces. As they have physical representations those physical props provide - by the means of being physical - persistency (see *Tangible Interfaces* 5.3.2, *p.* 91).

Most interfaces like for example a mouse or keyboard do not provide persistency, instead they provide "at the moment" information.

Information provided. *Sensitive Sample*, the *SoapBox* or tracked objects provide a stream of information about their current state: position, orientation, temperature or light that is detected by a sensor. In STUDIERSTUBE this stream of information is normally used to directly control the state of virtual objects or virtual cameras. From these information stream events can be extracted like "the object has been moved" or "the object has been shaken".

Some devices provide information like:"at the moment the 'h' button is pressed" and some do not even provide this information, but just something like " '1234' was read with the barcode reader". So the current state of the device can not be detected, just that something happened a short time period before.

Visibility. Some input technologies are more visible than others. While barcodes, and fiducial markers (e.g. ARToolkit markers [66]) have to be visible and clearly indicate that they can be used for interaction with the system, RFID tags and *Sensitive Sample* can be hidden and do not – at least at first sight – seem to be part of an interface. Although the *affordances rule* (2.4.3, *p.* 19) states that an interface should clearly indicate how to use an interface – which includes that it can be perceived as being an interface – this information can be included in the shape of the object, or in the environment itself. The *MaterialKammer* is a good example of a concept, where users can expect that *everything* in the environment will be usable as an interface.

Through the rapid development in the radio technology, *tethered* devices are replaced with *wireless* ones (keyboards, mice, microphones, ...). Through applying the rules of Norman [97] these devices are still recognized as input devices.

Theory and Praxis and about budgets. When choosing a suitable device for a interaction design, it is also important what currently is possible and doable.

Also budget arguments are taken into account, because budget and person power are restricted. In reality not everything that is possible is realistic or even realisable, under the restrictions of budget and available working power. The discussion in research whether everything possible should be developed or not (like e.g. the nuclear-bomb), is now switched to the discussion to "is it worth the money that must be put into the development". Although this seems to be a "business like" approach, research is facing a more and more a business like situation in terms of resources available.

6.1.1 Matching qualities and requirements

As described by Ishii and Ullmer in [60] a set of interaction methods have to be found for the emerging field of tangible interaction tools. A concept similar to widgets in GUIs, have to be found for tangible interaction tools.

In this thesis we propose to analyse the qualities described above to find a proper matching between requirements and interaction tools.

For example: *Persistency* suits well with the selection widgets like list-boxes or combo-boxes. The current selection is visible as well as the alternatives that can be chosen from. While *trigger* devices like e.g. barcodes are better suited for commands (that trigger some reaction).

In the ATELIER project the RFID tags and barcodes where used in both interaction situations: choose from a number of items and issuing a command. Because we applied the *affordances* guidelines, the prototypes were usable. Still it would be better to actually use the different input devices so that their qualities support the interaction.

Conclusion. Each user interface has special qualities. In a good system design this qualities should be used to support the interaction with the system. Giving the users the choice to configure their setup still leaves them the choice to use different input devices, but the predefined setup should actually be grounded on the qualities of the input devices to already providing a good start-point for the users.

6.2 Dealing with conflicting inputs

When dealing with simultaneously inputs, that can influence the same state, situations can arise that can not be solved in a straight forward manner, especial in interaction where a state is selected. An example of the ATELIER project can be used to explain this with an example:

> A user selects a special configuration of the displays by using the ControlCube by turning the side of the cube up, that activates a configuration. Then a different – or the same – user reads in a barcode for changing to a different configuration.

In this situation a conflict arises, on the one hand a *persistent* interface is used to select from a number of options, on the other hand a *trigger* interface is used to issue a command that is in conflict with the *persistent* interface.

There are actually a number practical solutions to this specific problem:

1. Inform the user of the problem and request for additional information how to proceed.

   +) it is transparent for the user that there is a problem. The user gains insights about the system and is able to identify the problem,

probably avoiding it the next time. Giving the user the ability to solve it in a way she feels it is understandable for her, storing this information in the system as a personal setting (see *Configurability* 4.9, *p.* 73). This procedure can enhance the interaction language, creating a configurable interaction language, which means that the user *configures* how to handle this conflict.

-) interaction is interrupted - the focus on the task is lost and the user has to focus on solving a problem not related to the work that should be accomplished.

2. If the configuration is selected with a persistent interface it overrides all other input devices.

   +) the interaction continues without a break

   -) Reaction of the system on the action is not expected – *Is the barcode reader not working?* Even if the system informs the user about the problem, and what has happened - it is a source for misunderstanding the interaction pattern.

3. The last command is taken, deactivating and ignoring the persistent interface.

   +) the interaction is continues without a break

   -) the quality of the physical interface - it represents the state of the system in the physical world - is lost. Even if the system informs the user about the problem, and what has happened - it is a source for creating inconsistent system states. The physical interface is no longer representing the state of the system, taking away the one of the most useful qualities of this input device. Also the virtual aspect of the system is enhanced, as physical object have less influence.

4. Do not allow simultaneously input for selecting states.

   +) no conflict arises

   -) The question arises: How is a user able to specify which input should be activated? To find an answer for this question, new "meta" tools have to be introduced. These "meta" tools tend to make interaction more complex then it needs to be. This meta interaction is not even a tailoring activity as it just changes the "active" state of a device.

Probably the best solution could be to combine point 4 and 1. Giving the users means of (simply) specifying that they want to use a specific device and telling them, if two conflicting inputs happened, what went wrong and how they are able to resolve the problem.

Conclusion. The example makes clear that some input situations can arise that create conflicts. Actually none of the provided solutions is really perfect. Still providing the user with multiple means of interacting in specific situation creates a system that will more likely be accepted by the users. The system is flexible to be used in different working situations and also can be adapted to different work practices.

6.3 Undo - Qualities

Nearly all "state of art" applications provide an undo function. This is one of the requirements presented by Sheiderman (see *Eight Golden Rules of Interface Design* 2.4.4, *p.* 19).

> There is no undo in real life.

This statement reminds us, that in real life we do not have the possibility to undo something, this also true for tangible and physical interaction. There is no way for the system to "undo" an action on a tangible user interface. Tough the action that was invoked based on this action can be undone (except for some special commands like "exit application"), the change to the physical object can not made undone – at least by the system. For some interactions like navigating to the next picture in a series a "back" command can be issued by the user - this is actually not undoing the command (next), that was triggered before. Changing a state e.g. by turning the ControlCube, can not be undone by the system, but the user must turn the cube back to the side that was facing up before. The system could help the user to recreate the state that was active before, by providing some information of how to accomplish the inverse command e.g. `Please turn the cube so that the 'browse' side is facing up.`

The Tangible Image Query has no undo function, and even worse it is practical impossible to redo a colour sketch. Once the sketch is changed - the coloured objects are moved - there is no way of reproducing the previous colour sketch exactly. Most interfaces that share this problem try to minimize the indeterministic part, that originates from the interface. By using thresholds and discrete transformations, which filter out small changes, providing a interface that can be controlled by the user.

But this indeterministic feature is also sometimes used to increase the inspiring character of an interface. From the view of a developer this quality is causing a lot of problems: when an error occurs, it is impossible to recreate the situation to track the problem down.

For the Tangible Image Query we found a practical solution:

- the results of the query can be stored in the database, eventually printing a thumbnail page, to retrieve the results or browse them in more detail
- the users can turn the Tangible Image Query on and off. If turned off, no new sketches are detected - for example originating in a change in the lighting situation, or by a different user changing the sketch - the current results are not changed.

However this locking is not displayed in any matter. This means that we violated the rule that a feedback should be given (violation of *alignment* see 2.4.5, *p.* 21), which states that the user should be able to perceive the system state – namely that the Tangible Image Query interface is turned off and changing the sketch will not create new results. A possible solution would be to turn off the background light of the Tangible Image Query, which would be a perceivable and understandable signal that the device is off.

Conclusion. We did not manage to implement the undo functionality in the ATELIER project, and in STUDIERSTUBE this feature is only supported in the *Construct3D* application. An alternative to an *undo-function* would be the "save early, save often" approach. This approach will be discussed in the section *Improving the Save Command* (6.7.2, *p.* 124).

6.4 Input and Output Abstraction

In both projects input and output abstraction were implemented. Special issue we discovered during the development of these abstractions are now summarised.

6.4.1 Input Abstraction

In the *AR Puppet framework* (3.11, *p.* 46) a abstraction layering was implemented, but is limited to the actions of the agents that are managed by the framework. The *APRIL Framework* (3.10, *p.* 43) has – in the story editing language – fully realised this concept. But the binding between the "meaning" and the physical interaction of the user has to be realised by a *Component Implementer*, depending on the presentation environment selected. The *PUC framework* (3.12, *p.* 47) provides an abstract description of interaction possibilities. The clients then use this information to create an interface, that can be used by the users. Additional information about the

dependencies and possible commands combined with explanations can also be provided.

The *Sensitive Sample* from the ATELIER project are objects that are equipped with sensors to detect interaction with the objects. Examples for actions that can be detected are touching, shaking or turning (see *Sensitive Sample* 5.4.1, *p.* 92).

The tilt sensor can only specify whether they are on or off (1 bit). By combining more than one sensor an estimate of the rotation of the object, meaning which side of the object is on top can be calculated. This realises the second abstraction layer, the system can provide messages like `object X, side 2 is facing up`. This state could also be register with other kinds of sensors or tracking of the object. When adding time information and context sensitive processing of the data events like `the object X has been turned from side 2 to side 3` can be produced. Also the event of shaking can be detected – for example when the sensors produce a lot of changes in a small time period. Now information like `Object X has been shaken` is available for processing. This high level information is no longer coupled with the technology behind the interaction.

Still the third layer of abstraction is needed to make the system really flexible. What does it "mean" when the user is shaking the *Object X* in this particular interaction context? An example could be that the user wants to select a random image from the database or to get more detail about the contents of the object (to "shake information out" of the object). The most useful information about the interaction is actually *what the user wants* and not *what is being done*!

In the ATELIER project parts of this abstraction were realised. For example to select a specific image from the database many different ways are possible. By reading in the barcode associated with the image, by placing a tagged object (picture cards) on a tag reader, by activating a result of an image search. The message that was produced by the system was always `image X selected`. The source of this selection is not important, but that this particular image was actually selected by the user.

Conclusion. Messages like `select random image` or `display information about wood` is what the input system should be producing. The *intention of the user* can also be created by using a GUI by selecting from a list-box or pressing a pushbutton, or by placing a specific tag on a tag reader.

Still the intention of the user is the same, just the way she communicates this intention is different. A system design should therefore incorporate the proposed input abstraction layers, for this is a foundation for a *multimodal*

and *configurable* system.

6.4.2 Output Abstraction

The *APRIL Framework* (3.10, *p.* 43) allows to specify, independent of the setting and the interaction devices, outputs and behaviours, without explicitly having to know the target presentation platform. The executable presentation is then created by specifying the available inputs, outputs and the general setting of the presentation and the actual presentation. By combining this information, a presentation specific for the available tools is created just before the presentation is started [76].

The output abstraction in the ATELIER framework was also implemented in the inter-component communication.

6.4.3 Connection between Selection and Output components

In the ATELIER environment we created a very flexible way of working with images. Different methods for accessing and selecting images from the HMDB were implemented. Once the user has found the image she wants to work with, several output components could be made use of to output the image. The most interesting feature is that the different search methods could also be used as output devices, meaning that based on the image that was send to these "output" devices, new results would be displayed, creating a useful and inspiring method of browsing through the HMDB. This browsing is an alternative to the hierarchical or sequential browsing, which was also implemented.

Output Components We developed a series of output components that can be used by the users to display (or output) images that are in the HMDB. We provided several display screens to project on, a three screen wall, where the display planes could be adjusted to form a closed space or a flat wide screen and anything between those extremes. Additionally there is a table on which the designer can place their artefacts, on which an image can be projected from underneath, changing the visual appearance of the table. The Texture Painter was used to create compositions of several images and textures to augment artefacts. Actually two *Texture Painters* were used by the students to project on the vertical and horizontal faces of their artefacts. When an image is sent to one of the *Texture Painters* the currently active texture is replaced with the new image. Therefore the part of the artefact,

that was painted with this texture, changes its look, creating a new vision of the artefact. A printer can also be used to print out an image – including a barcode to get a physical handle to the picture.

Tangible Image Query, Ontology Search and Thumbnail pages We implemented different methods to access images from the HMDB. These methods can be used in conjunction and are connectable. The starting point for most of the users was the thumbnail pages. These pages where printed after new pictures had been added to the database. So for each "upload" session the users would get pages with thumbnails of each picture they added, underneath each a barcode was printed (see *Barcodes and RFID Tags* 5.3.1, *p.* 90).

These pages are the physical handles to the digital material. The barcodes are then be used to send the images to any output component, for example to display the image on one of the screens or to the Texture Painter (described in *Display Manager* 6.7.1, *p.* 122).

The Tangible Image Query can be used to find and browse through images in the database based on their colour layout. By creating a colour layout with coloured object the users could create a query to the image search engine. The results of this query were displayed on the display wall. The same procedure can be used in conjunction with the ontology search. By reading in an ontology query barcode, the users can browse through all images in the database that are annotated with that specific barcode.

Barcodes to browse through the results (showing the next or previous 9 images) were provided. For this the source of the images is no longer relevant. By selecting a output - using barcodes - and then selecting one of the result images - again with a barcode - this selected image would be send to the output device. Using those two different approaches - colour sketch and keywords - it is easy for the users to find visual material to work with.

6.4.4 Combining the Search Methods

The most interesting with the system is actually to combine these two approaches. An image found with the image query can be sent to the ontology search, resulting in images that have the same keywords, but also to sending images that were found through the ontology search to the image query, which will display image that have a similar colour distribution. With this method the users are able to get a different understanding for the visual material - starting with images of fruits ending up with a picture of a sailing boat. This encouraged the students to think different and find new meanings for their artefacts, and new visions of what their project is actually about.

Nevertheless these methods can be used for the standard usage of search methods to actually find something one is looking for. By sending the results to the printer physical handles for the images are produced.

Multiple Ways of manipulating keywords in Atelier In the setup we created there are multiple ways of annotating images with keywords. During the upload procedure keywords can be assigned to the list of pictures or to individual pictures. Both the Path Creator and the Upload applet can be used to add images to the HMDB. While the Path Creator allows the user to specify any meta-information, with the Upload Applet only keywords may be added. If keywords where added that are currently not in the ontology a new ontology barcode set must be printed.

Once the images are in the database they can be accessed using any method described above. When using the HTML interface, the users are able to change any keywords. Even adding new keywords is possible with the HTML interface (see *HTML as a User Interface* 5.1, *p.* 83). This interface is especial useful if the ATELIER room was used by another person, so that the annotation of the images was possible without even being in the ATELIER environment.

Once an image is displayed on one of the screens, keywords from the ontology can be added or removed by reading in the ontology barcodes. We also experimented with speech recognition, but due to the rather poor recognition rate of the system we used, this feature was removed again. Nevertheless it is worth mentioning, that this interface was actually available, as it increases the interaction possibilities and demonstrates the multimodal interaction in the ATELIER project.

The use of the ontology barcodes was also a good solution as most of the interaction with the display wall was based on barcodes. A feedback of the current action is displayed beneath the image that shows the current set of keywords that are specified for that particular picture. As this feedback line is hidden after a period (approximately 3 seconds) reading in a new keyword will show the user that something has changed - making clear, what the result of her action has been.

Conclusions. The output abstraction on a inter-component level proved to be a flexible concept. New "output" components were added easily, expanding the possibilities to make use of content stored in the HMDB. In the ATELIER project we only investigated the use of images, but this approach can be expanded to other output medias (algorithms to search for "similar" sounds were developed already). By using the different attributes – content

as a color layout, and the keywords attached to that content – as a query to a search engine can be extended to other attributes (like author, project connected too, ...), to allow the users to investigate all aspects of a specific content.

To be able to implement such an approach a uniform representation for the content is needed, which can be achieved by using a *global repository* 7.3, *p.* 132.

6.4.5 Applications versus Assemblies

The terms *application* and *assemblies* are used to specify different approaches in providing a collection of functionalities that the users can make use of to work with the system. With *application* special purpose setups are meant that allow the users to work on specific tasks. *Assemblies* on the other hand are generic tools that allow users to perform generic tasks. These two concepts are interwoven, meaning that the implementers of a application use functionalities to create applications for a special purpose. A collection of functionalities is often referred to as "framework" or "software library".

Component Assemblies In the ATELIER project applications are called *Component Assemblies* already stressing that they are a mere combination of components, which represent a functionality. By coupling the functionality and presenting the user a predefined configuration of functionalities a sort of "application" is provided.

When allowing the users to dynamically create connections between different functionalities a new system for new tasks can be created by the users. There are many examples for dynamic configurable systems, most of them provide GUI's for the users to create the configurations (see figure 6.2).

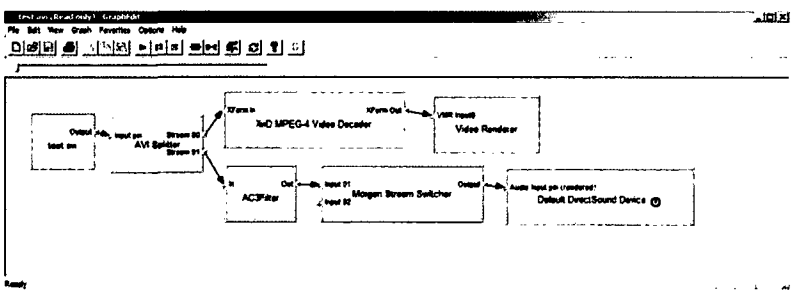

Figure 6.2: A screenshot of the DX-GraphEdit tool (included in Microsoft's DirectX-SDK), that allows user to perform configurations with a GUI.

Conclusions. Developing a system that can be configured by the users (without the need of special trained personnel) to created their own workflow, needs abstract and flexible interfaces and a well designed system. The goal of this thesis is to enhance this idea and make it a general concept when a *distributed multimodal* system is created.

6.5 Discussion – Invisible Person

Game-board and Game-pads. The Game-pads were designed to have enough information to function independently of the rest of the system. No supervision is necessary, the game pads will highlight or even change their state if any player interacts with them. The game-board has knowledge about the connection between the pads and it also distributes information from the game to the pads (active player, initialize). Most of the game-boards realised used the Game-pads.

Performance We have implemented a real-time system, to enable real time interaction, a system had to be developed that recognizes user actions and replies to these actions in a short period of time. As most of the described user actions can only be detected at the end of the interaction (like "double click" can only be detected at the end of the second click) some delay in the response is system immanent. The vision-module updates the information about the people on the stage with a mean rate of 10 information blocks per second. The render-module is able to produce about 15-18 fps (depending on the complexity of the occlusion masks). As soon as a person (including the IP) moves over a game-pad it changes its colour and shape (which means, depending on the frame rate of the render-module and the detection rate of the vision-module, about 0,1 seconds). The reaction to a click action is determined by the detection of the user action (normally about 0,6 seconds after rising both arms while standing on a game-pad).

Conclusions. We have implemented and installed an advanced user interface in an immersive virtual environment accessible for a large public. Some extra work had to be done to develop the lab prototype to be functional in a public environment. By utilizing well-known metaphors and analogies the user interface was understood by almost all users. We have shown that it is possible to implement an advanced interaction scheme in an immersive virtual environment, which is well perceived by everyday users, satisfying all four interface design requirements described by Bowman and Hodges [15] and

realizing a stable and reliable setup to simulate a "mouse input" without requiring the user to handle any input device. The idea of using this advanced interface as an input for games originated from the environment in which the system was installed. The "playful" interaction between technology and humans is a widely used way to transport knowledge in a museum. For modern museums the "interaction" between exhibited objects and visitors is a way of getting people to be really involved with the themes. Especially school classes populated the system with great excitement during their visit to the museum. The public accepted the installation and tried to communicate with the IP with anticipation. Most of the users' cognitive power is absorbed by the IP and its effort to communicate with the users. The actual interface to the game is not the main intellectual task for the users. Most of them try to understand what the IP has to tell them. By giving the users an understandable metaphor of "someone" living in the mirror, they do not concentrate on learning the usage of the interface. So the simplest learning scheme "imitation" is performed by nearly all visitors. The hardest task for the users is to recognize that they are playing a game. Once users made that cognitive step, the interfaces were well understood. The most difficult part was developing a vision system which would function in a non-controllable light environment.

6.6 Discussion – Tangible Image Query

Comparing the Interfaces Comparing mouse and tangible interface of the Tangible Image Query

As described in *Properties of Instruments* (2.4.2, *p.* 18) a user interface can be evaluated applying the measures: degree of indirection, degree of integration, degree of compatibility. Although the original publication focuses on widgets, it can also be adopted for tangible user interfaces.

Referring to the Tangible Image Query setup, the object that the interface operates on can be interpreted in two ways. The users manipulate the colour layout sketch, but they do that because they want to change the results of the colour layout query. We will now compare the mouse interface with the tangible interface by applying those measurements, to evaluate the benefits and drawbacks.

The degree of indirection is a measure of the spatial and temporal offsets generated by an interface. The spatial offset is the distance between the input part of the interface and the object it operates on. The temporal offset is the time difference between the physical action on the interface and the response of the object. The temporal offset is quite the same for both interfaces, as the

sketching of the colour layout is performed in real time with both interfaces, without any time delay. And after a specific time without manipulation both interfaces send the created sketch to the search algorithm. The spatial offset is slightly better with the mouse interface as the drawing area and the display of the results are on the same screen and the tangible interface needs two separate areas, one to sketch the colour layout and one to present the results.

The degree of compatibility measures the similarity between the physical actions of the users on the interface and the response of the object. The tangible user interface provides a higher degree of compatibility as the users directly manipulate the colour layout sketch with the coloured cubes. The interface is a very direct approach without abstract mapping between input and effect on the query. With the mouse interface the users have to draw by selecting a colour from the palette and then move the mouse to create a coloured area in the sketching window.

The degree of integration measures the ratio between the degrees of freedom (DOF) provided by the logical part of the instrument and the DOF captured by the input device. The degree of freedom can be evaluated in two dimensions: the colour dimension and the layout (2D) dimension. The mouse interface provides only a 2D interface. Therefore an indirect colour selection method has to be incorporated. The tangible interface in our current setup allows direct access to all three dimensions (colour and 2D), but as one of our test users stated, the cubes can also be stacked to create a three dimensional structure. So the tangible interface has four dimensions that can be operated on. These do not match with the needed three dimensions, but can be resolved if coloured objects are used that cannot be stacked, like half-spheres instead of cubes.

Test with users We have tested the system with a large number of users. The application was presented at various workshops, including a "Beginner's day" at the university, a workshop that was part of the review of the ATELIER project and also at the final workshop in conjunction with an open day for the CHI2004. Some of the workshops were publicly accessible, therefore different types of users tested the system, where the users were chosen to represent different drawing and computer usage skills. The general feedback from the testers was very positive. The tangible interface is very attractive and easy to use.

In addition he have interviewed selected special users, that work with pictures in their profession and some students with a technical background. Some of them first used the tangible interface and then the mouse interface, for the other group the ordering was reversed.

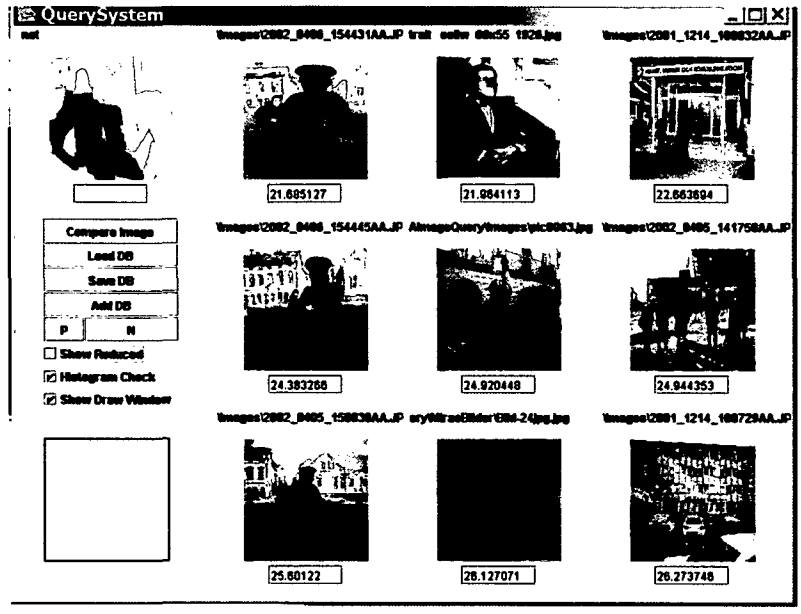

Figure 6.3: Sketch done by a painter looking for the portrait of the sitting person (second image).

Most of the users tried to "draw" the picture with the mouse, and the tangible interface helped them to understand that a sketch is better for the search than a "redraw" of the image they searched for. The results that were presented by the search algorithm often did not fit their expectations when drawing with the mouse (and trying to draw structures). Some did not like the shape of the cubes and suggested other shapes, while a number of users reported that the coloured cubes reminded them of their childhood, which they described as a more naive and playful time in their life.

It is interesting that the users sometimes could remember the shapes but not the colour, but this can be a subjective characteristic, or it can be correlated with some professions like architects, for example. Some users from the technical background had deep concerns whether the colour-based query for images makes sense at all, while most users with a creative background found this approach interesting.

The general response was very good, and most of the users liked the tangible interface better than conventional one. As we had some test users with visual arts background, we noted that they were very pleased with the surprising component of the tool. E.g., a user searched for a sun-set that was instantly within the top 15, but mixed with images of red flowers and a firework. These results were far from disappointing, and the flowers and firework images fitted well in the users expectations.

Many available examples prove that the colour layout search is an interesting approach to realise an image query. Our work presents a new tangible user interface that allows creating colour layout sketches in an easy and straight forward manner. Rather than improving the query algorithm itself, we tried to find a new interface, which suits the existing algorithms better. The algorithm needs a certain level of abstraction, which is often hard to achieve using a traditional mouse-interface. The new colour cube interface makes it impossible to draw precisely, and therefore helps the users achieve the needed level of abstraction.

Still many of the problems of the underlying methods persist. Tangible user interfaces enrich the possibility of collaboration and multi user input, with all the problems that come with it. For example there is no method of helping the users with synchronization, as all users that use the interface, actually shared this interface physically. They have to sort out conflicts between them without (computational) help, for example: someone adding his ideas to the sketch without asking.

The method of colour layout image retrieval also has its flaws. Most of the users cannot clearly identify the distinction between shape and colour layout. A good example is the search for a sunset. A red shape placed in the middle of the image is a good approach, but images where the sun is not close to the centre will not be found, even if it is a picture of a sunset, and images of a red flower in the centre of the image will be found instead.

Conclusions. We observed that the use of a tangible user interface helps the users to create colour layouts rather than shapes. More over the interface can be used in a more vivid way. It allows direct access to the sketch rather than the indirect method of using a mouse.

The colour cubes interface fits very well with the underlying visual image query, and helps the users to cope with the limitations of the query algorithm. In this way the usability of the whole system is significantly enhanced.

We had access to a repository with over 14.000 images, with which we evaluated the scalability of the algorithm. There was no noticeable time differences when searching a database with just 500 images compared to the search in the full database. For the 14.000 images the database for the signatures had a size of about 28 megabytes.

6.7 Configuring

The *Configurator* (3.3, *p.* 25) from the ATELIER project allows the students to perform limited configuration actions, but demonstrated that this approach

is appreciated by the users. A development in ATELIER did eventually go beyond the simple configuring provided by the *Configurator*. This component allowed the users to specify in detail, which inputs and outputs should be connected. For the architecture students this application was to complicated, this approach was more appreciated by the interaction design students in Malmö.

6.7.1 Discussion – DisplayManager

The redesign of the DisplayManager described in 3.3.5, *p.* 32 includes some issues we learned from the users and their feedback.

We will now shortly describe how the ATELIER system handles input. As described in the section *HMDBLookup* (3.3.4, *p.* 32) this component introduces an abstraction layer. There are three methods of accessing content from the HMDB:

1. use a barcode from the thumbnail page,
2. use a tag associated with a specific content,
3. select a result from any search method (selecting the barcode of one of the results).

By using any of these methods the HMDB-ID of that specific content will be send to the DisplayManger.

One design issue of the DisplayManager is that it keeps a state of an "active" component. For example if a student wants to display a content on the left display, she has to active this display be reading the barcode associated with that display, and then the barcode for the image or video. All subsequent actions to access content in the HMDB will change the content on this "active" display. Note that the TexturePainter, the *Image Query Search*, and the *Ontology Search* are also possible "displays".

The weakness of this design is twofold:

1. there is no feedback (except for the fact that the content will be changed) which component is "active".
2. we implemented also other commands that refer to this "active" display – namely the "save" command and also the "toggle keyword" commands.

The Save command. Our first approach when a user issued a "save" command was a three step procedure:

1. "Activate" the component that should be saved – including "all screens".
2. Read "save" barcode
3. Read an unused barcode – these barcodes were given to the students on a separate sheet with their name on top (see figure 6.4).

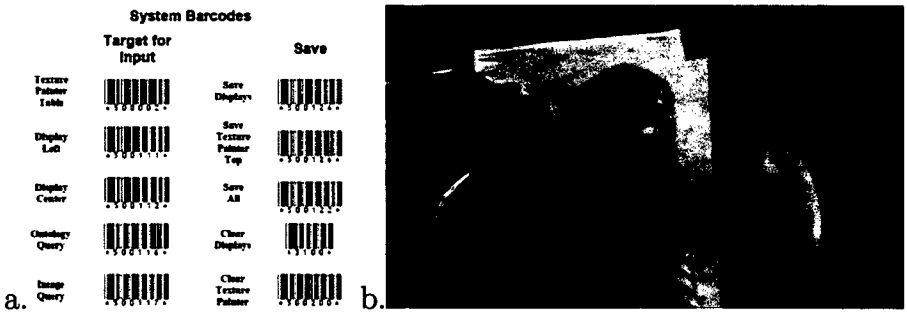

Figure 6.4: a. DisplayManager command barcodes b. Andreas using his personal barcodes

This procedure violated *two* rules for a good design. First the feedback was missing, second the users had to keep in mind in which state the system currently is (2 or 3). Our next step was to remove the second step by providing barcodes that would combine phase 1 and 2. So barcodes like "Save Display Wall", "Save TexturePainter", and "Save All" were presented to the students. We then implemented a feedback mechanism (default position was the middle display of the wall) to ask the users for an unused barcode, which will be associated with the current setting.

6.7.2 Improving the DisplayManager

As already stated a system designer should make use of the "qualities" of an input device. The barcodes are an unsuitable interface for selecting, as in the DisplayManager, the active display. A better choice would be to use RFID tags, that could be left on the reader to give a visible feedback of the active display. Objects representing the different output devices could be designed so that a strong feedback is presented to the users. Another approach could be to make use of the *Sensitive Sample* (e.g. the ControlCube) to select the active display. This would also ensure that the currently active display could be perceived by the users.

Improving the Save command To get rid of the still two step "saving" procedure, we envision a different approach that will be implemented in the future. At the start of a working session the users will have to activate a "start work for group XX" barcode that will be given to them as a sort of "login" barcode.

If they then issue a "save" command the current state of all components will be stored in the HMDB in a special list. At the end of a work-session – or any time in between – the users could issue a "print saved configurations", which will print out a list of *physical handles*. A visual representation for each component will be created, supplying a barcode for the whole setting, as well as for each part. This would enable the users to perform a "selective load" as described in *Saving and loading in distributed systems* (4.6.1, *p.* 68).

If the users are able to create these "saving points" so that their current state of work is stored in the system, they themselves would also define "undo steps". Restoring them just by reading in the barcode underneath the configuration.

This approach would also create a documentation of the work that had been performed. Following this scheme, if every action is stored, the "undo" command could be easily be implemented (see *Undo – Qualities* 6.3, *p.* 110). This is actually the approach that was implemented in *Construct3D*, where a "save" command saves a list of issued commands.

Browsing the multipart elements

"Multipart elements" means a collection of any content that could be images, videos, sounds or 3D content like different models. In ATELIER this is any part of the hierarchy in the HMDB, that contains any number of children. Also a result vector of the search methods described (Image Query and Ontology) are multipart elements in the ATELIER environment.

Up to now the only browsing method was to select "next" and "previous" commands with barcodes. Additional navigation methods could be implemented like "auto next" simulating a slideshow that displays the next image after some time like "auto next 3 sec." and "auto next 5 sec.". Also a "random next" could be implemented that selects randomly content from the active collection. To select these different browsing methods the Control-Cube using the *Sensitive Sample* would be a good choice, due to its qualities. An alternative would be to use tagged objects that represent these different methods.

Once a "stop" input is triggered by the user the automatic replacing of the content should stop. The "previous" command should work as expected displaying the last images. To continue the browsing, the "next" command

could be used. Supplying barcodes or speech input for these commands is the right choice, because their qualities fit the requirement of fast and not persistent input.

6.7.3 Configuring physical handles

The students made use of the "physical representation" of the physical handles in the ATELIER environment. They placed the physical handles in the physical world by sticking them on their models or their posters. One student created his own presentation list – just like a slideshow – by placing the barcodes neatly in a column, while others distributed the handles in the space an activated them while walking around (see figure 6.7).

This feature of *physical handles* to be *configurable* allowed the students to support their personal working style.

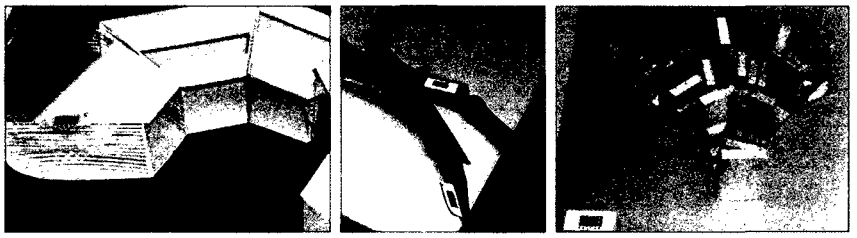

Figure 6.5: Using barcodes to augment models.

Conclusions. Although physical handles to digital media have a number of advantages: they can easily be configured, they can be used to add comments by writing on them, ... There are also some drawbacks: when the physical handles exceed a certain number, searching for a specific element can get tedious, they also tend to get lost – especial in a work space like the Academy of Fine Arts (see figure 6.6). With the methods described to search for content and to produce new handles, these drawbacks are partly solved. Additional information has to be provided like when the content was added to the HMDB, to identify "outdated" handles and also the problem of versioning of content has to be tackled. In the ATELIER project we always created new barcodes for changed versions of content, so soon the students had a number of sheets full with "old" and "new" handles and sometimes they used the wrong handles.

Still our experiments showed that the reuse of content was especially easy, one group of students did not bring any material, they just scanned the physical handles from the previous groups and used suitable content for their

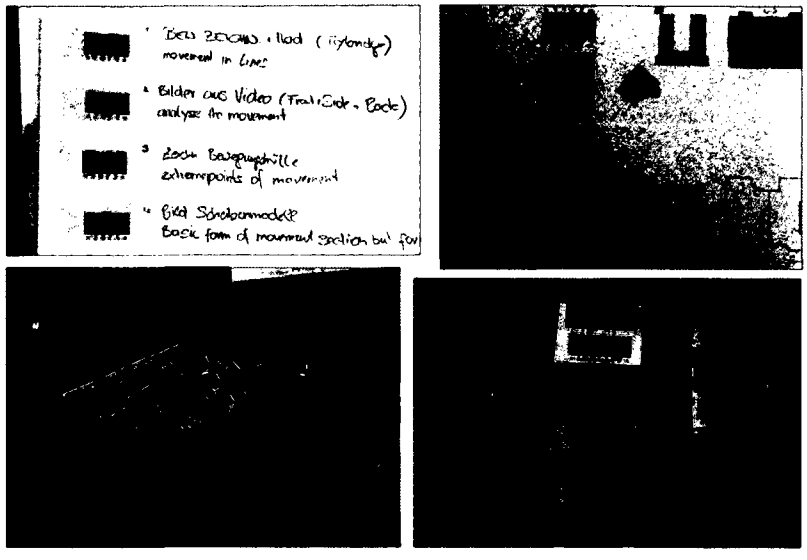

Figure 6.6: Rearranging barcodes for a presentation and spreading out the material.

presentation. This example proves that the physical handles allowed others to easily re-use content.

6.7.4 Configuring the context

Using the physical handles and the *Texture Painter* the students were able to quickly create new settings for their models, exploring different contexts and scales of their models within the environment.

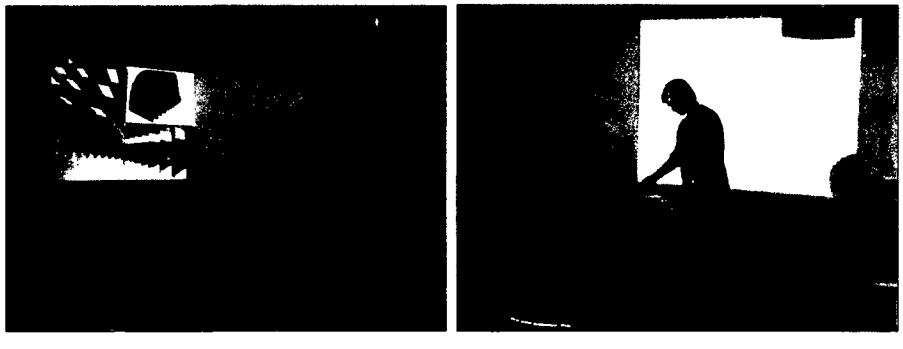

Figure 6.7: Using the barcodes during the presentation.

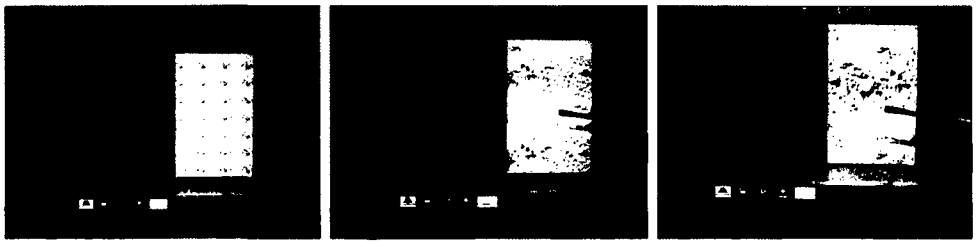

Figure 6.8: Experimenting with scale.

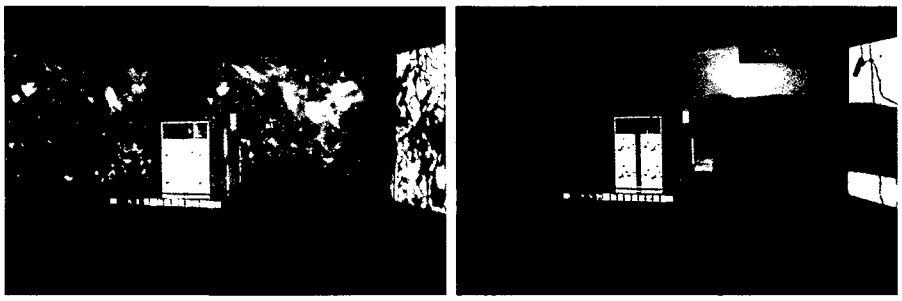

Figure 6.9: Experimenting with different contexts.

6.8 Distribution

Our partner in Oulu, Finland created a common communication platform (the ATELIER-kernel) over which most of the communication between the components was realised. In theory all components should be able to interact with each other, but in praxis this was not always true. Using XML messages to exchange information between the components should ensure that all components are able to "understand" each other.

XML is a very generic concept, which only specifies the way hierarchical content has to be formatted, therefore XML does not specify how a specific meaning should be encoded. This lead to a situation where the internal language between the different implementation sites started to differ and each group in the project created its own dialect of the "ATELIER language".

On occasional workshops and joined demo sites – Jamborees in Gothenburg, Ivrea and the workshop in Vienna – those differences could be resolved to reach a common language again.

From this common language was a profit for all setups, as functionality was accessible for each component independent of the group that had implemented it. A good example is the tight integration of the ontology search into the system. The ontology could be used by all of the already implemented components, without having to change or adapt the output components.

In STUDIERSTUBE the PUC framework provides an easy to use way of creating distributed interaction, providing similar looking GUI's on various devices. The general approach of STUDIERSTUBE of using a distributed scene-graph supports developers to create distributed applications. In the *Construct3D* setup several users can work cooperatively on a geometry task making use of these distribution possibilities of STUDIERSTUBE.

Conclusions. The distribution implementation of the framework is an important basis to create distributed applications. The different approaches of ATELIER and STUDIERSTUBE, both have their advantages and drawbacks, heavily influencing the development of the applications. In a recent discussion with G. Reitmayr, he stated that the design of the middle-layer of a distribution framework is already have influenced by the goals of the overall system.

6.9 Discussion – Sensitive Sample

The design of the samples is modular, so that practically any kind of sensor can be attached to the main board. We have only experimented with two kind of sensors: tilt sensors and touch sensors. Even with such a reduced set of sensors, the ControlCube and the complex MaterialKammer application can be implemented. The ControlCube has been extensively used by the students to start various videos during the presentations, and MaterialKammer will be realised in the near future. The first mock-ups that were tested by the students, using a very simple media content selector and just one material sample, proved that the fundamental concept is highly appreciated.

Now, after the technical issues are solved, the biggest challenge is to understand the actions of the users. What does it mean when someone strokes a sample, or when someone knocks on it. Many social and cultural issues will certainly influence the meaning, and all of these should be taken into account.

For the setup to provide the above described functionality, two parts are crucial. First enough media must be collected and indexed to specify to which material the media is connected to. For example a picture of a saw-mill is connected to wood and steel. And second the multimedia content selector must be implemented so that the resulting medias are connected to the users actions in a meaningful way. The media displayed should not be totally random (which is quite easy to implement), but be related to the actions of the users. But on the other hand there should be a surprising element that

encourage the users inspiration and reflection on the media that is being displayed.

Conclusions. By interacting with the real material, feeling, smelling and seeing it, and getting additional inspiration from the medias presented by the system, the students are encouraged to reflect about the meaning of a specific material. This helps them to understand how the usage of this material influences their project. The *Sensitive Sample* represent an alternative and inexpensive way of creating real objects that can sense the environment.

6.10 Conclusions

Our experience in both projects prove that the concepts of distributed multimodal interaction are useful. The users have more confidence in a system they themselves can configure, where providing physical handles to digital material eases the access to a system, our experiments showed that the interaction with the ATELIER system was immediately understood even by inexperienced computer users.

Some of the issues that we found useful are summarised in the next chapter.

Chapter 7

Design Patterns for a multimodal, configurable, and distributed system

The software design for a system is crucial for the success of the applications based on the framework. Missing features can sometimes be easily integrated and sometimes they pose invincible problems. A software design will never be "perfect" as new issues will surely arise during the development of applications, that can not be foreseen by the designers of the basis framework. Nevertheless some features can be easily designed, if the software-designers are aware of them, and when they are available, they will be used by the application developers. Features available in the basis framework are more likely included in the applications than solutions that are migrated from application to application.

New features will first be implemented in an application, and then, after they have proved to be useful, be integrated into the framework. This way of "moving" features from special applications to a general feature available in the framework was observed by the author in a number of occasions. Although this work practice obviously works in practice, it is suboptimal as some "old" applications will not support the new but useful features.

To "get it right" the first time it is better to design a system that incorporates (nearly) all of the following patterns. The applications described in this thesis will be referred to as examples of how these features may be implemented. These issues are mainly needed at the API layer of a framework. Easy access to API features will ensure that the application developers include this features in their applications. For some of the following items the API can be designed in a way that no additional implementation effort is needed for the application developers – similar to the scripting feature of

Open Inventor (2.2.2, *p.* 13).

All frameworks should, of course, be

- well designed
- usable and extendable
- stable (execute without crashes), contain (almost) no software-bugs
- support debugging and logging
- well documented
- provide simple, and well documented example applications.

Some frameworks also need to be designed for performance, but *improving performance* should always be the last step in a development, as the developers sometimes have to sacrifice some of the mentioned qualities to achieve a higher performance.

7.1 Keep Functionality simple and modular

A complex system should be created through connecting and combining simple functionalities. The principle to break a problem down into small problems that can be solved and then combining this simple solutions to solve the complex one is known as *divide and conquer* strategy. Many software designers have used this pattern. It is especially important in a distributed system as functionality should be accessible from everywhere in the system and therefore it must have simple interfaces.

This pattern is often used in application programming interfaces, while accessing these different functionalities by the users of a framework is often only possible by using specialised applications that expose those functionalities. We propose that these functionalities should be available for the users, in a way that they can connect them based on their current needs.

Although each component in the system is simple and has a simple interface a complex system can be created. It is also easier for the users as they are able to start working with the system with just a few functionalities and by learning how to combine them, and integrating new components in the setup they have already been using, their workflow will become more complex in time. This allows the users a graceful evolution from a novice to an expert.

Also the functionality of the system can be enriched without increasing the complexity for the user interaction. If the same design principles are applied to all the components the user is actually able to abstract from the known interaction patterns and extrapolate the new interaction patterns.

7.2 Abstraction of Input and Output

Abstraction of input devices is now a standard approach in system designs. Yet most of the systems only provide a single abstraction layer. As stated in *Input abstraction layers* (6.4.1, *p.* 111) the most important about a user interacting with the system is the intention of the user. *What should be done? What does the user wish to communicate?* So the top abstraction layer should communicate the *intention* of the user and not the *action* performed.

In ATELIER we used predefined messages that would be used to distribute this information. In the *APRIL Framework* (6.4.1, *p.* 111) this information is transported by defining events (that can be configured using the describe method), the *AR Puppet framework* (6.4.1, *p.* 111) – in the top layer of the director – has a similar representation. The PUC framework has a implicit representation of that information, by specifying the "name" of the state or command – but a basis set must be defined and consistently used in a system to really make use of that (like "save", "load", "exit", ...).

7.2.1 Provide concurrent multimodal access

As discussed in *Dealing with conflicting inputs* (6.2, *p.* 108), if concurrent input is available, a conflict can arise. Despite this issue we would suggest to provide means for the users (and the application developers) to use different input channels. This will ensure that a suitable input method is available in nearly every working situation. Surely some interaction devices still need development (like the interesting idea of speech recognition), but we believe that there is *no* "perfect" input device that will replace all the others. It will be rather a mix of interaction methods that will ensure a usable environment.

7.3 Global repository

In ATELIER all components profited from the availability of a global content repository. The file-based exchange of content – as it is current practice in most networks – is more a technical than a usable solution. PDM systems (Product Data Management) try to provide an abstracted view on the content that is available. With the possibility to cross-link content and apply

different search methods (see *Combining the Search Methods* 6.4.4, *p.* 114) to access the content the re-use of digital material is ensured.

7.3.1 Physical Handles to digital media

People have their own way of organising important data ([82]), most users (including the author) prefer – for some situations – physical representations of the content. It can be stapled, sorted, laid out on the desk and stapled again. On the other hand people got quite used to organise their telephone numbers on their cell phones.

Providing a system wide way of creating and managing physical handles to digital media, seems for us a valuable feature for a system. Examples are the barcodes used in ATELIER (see *Physical Handles* 3.3.3, *p.* 30), and also the ARTtoolkit markers that are being used in STUDIERSTUBE in various applications to represent this feature.

Physical handles also provide being physically *configurable*, which is an important feature for the users (see *Configuring physical handles* 6.7.3, *p.* 125).

7.3.2 Global save and restore

A framework should provide means for the applications to save and restore their states in the global repository. By providing a system wide method of saving and restoring, all the applications will include this feature and a uniform interaction pattern can be realised to perform this action. This will ensure that a "global" save command is available that will include all the applications in a distributed system.

User based tailoring information can also be stored in the repository, to activate them according to the current working scenario.

Selectively load An interesting issue raised during the development of the "save" feature in the ATELIER project. What happens if a user wants to restore only a part of a "global" save result? How to introduce what components should load the state and which components should ignore the restore command. A possible solution is to provide means to "lock" a component, so that it keeps the current state, independent of the actions the user performs. If such a feature is implemented. The important issue of providing feedback of that "locked" state has to be realised.

Another approach is described in (see *Improving the Save command* 6.7.2, *p.* 124).

7.4 Configuration possibilities

To give the user the feeling of *being in control* (2.4.4, *p.* 19), the possibility to configure a system is important. The students felt much more comfortable with the ATELIER environment once they were able to configure it by themselves, without the need for a technician. A system should support dynamic configuring, which is a complex task, that allows the users to change the system at run-time. The development of the Microsoft Windows ®OS needed nearly 7 years to provide this feature, before that the operating system had to be restarted after various configuration changes.

The *Configurator* (3.3, *p.* 25) from the ATELIER project allows the students to perform limited configurations. The concept of *APRIL Framework* (3.10, *p.* 43) explicitly states that creating a configuration should be placed in the hands of experts.

To find a balance between the complexity of the interface and the possible configurations is obviously the hardest task.

7.5 Keep the Golden Rules in Mind

The rules presented in this thesis provide guidance, when discussing about features and interaction patterns. Most experienced system designers have a "feeling" for what is right. Using these rules they can argue why one solution is "better" than another one, and they also help to support the users, by pointing out why users have problems with an interface. The rules provided a helpful starting point for the redesign of *Construct3D* 5.6, *p.* 100.

7.6 Support Undo

Although already mentioned in the *Golden Rules*, supporting undo is such an important feature that it is granted its own section. Norman describes in [97] in over 30 pages the causes for humans to err. Although there "is no undo in the real world", people have developed means to cope with their errors. A system should always provide means to undo a command or provide a reverse command (e.g. "next" – "previous").

Actually I think that, because of the "undo" possibilities, typewriters were replaced by text processing systems. It is easy to undo a wrong typed character so that even untrained typists can write large documents (like this one).

While most proper designed interfaces incorporate this feature, still interfaces (and applications) are designed without an undoing feature. Also

the possibility to interrupt an ongoing action of the system is a feature that sometimes is missing.

From own experience in implementing applications, I know, that exactly these important features are hard to implement. Therefore the framework should take care about this issue, to ensure that all applications include the undo feature. In a ubiquitous environment probably only a "global" undo will be realisable as discussed in *Distributed Undo* (4.8.1, *p.* 73).

It will be worth the effort, as users only will explore all the features, if they can be sure that they do not risk their achieved results being lost through an unthought action.

7.7 Provide Feedback

Also mentioned in the *Golden Rules*, providing *enough* feedback for the users is sometimes overlooked. Furthermore methods must be integrated into the system that allow the application developers to easily provide feedback. Which modality is being used to provide the feedback and how the users will be able to tailor the feedback should also be handled by the framework. The *DisplayManager* (3.3.5, *p.* 32) in the ATELIER project included some basic feedback managing capabilities.

A good approach is the progress meter used in conventional desktop interfaces. This concept has to be developed further to allow displaying progress in a ubiquitous environment.

7.8 User Identification

A lot of usable features are based on the user identification. Starting from determining the level of complexity (based on the experience of a user) to restricting or granting access to features. User identification in a ubiquitous environment is a very complicated issue – until now the Bat System [23] is one of the few projects, where this was the main goal.

Other issues are whether the users *want to be identified.* Data security and privacy are some of the social issues that have to be thought about. The tailoring of a system to different groups of users can also be preformed by configuring the system and the store that configuration (re-activating it using a tangible interface) in the global repository.

7.9 Record evaluation data

The framework should provide means to record the interaction performed by the users. Which functions were used, how often did the users issue the undo command (and which action was undone), how often did they access the help system and for what features did they require help? A lot can be learned and then improved on basis of this data. Sheiderman described in [127] that these actions were recorded for a system his group had to evaluate, and what they could learn from the data.

Although a logging service was sketched in the ATELIER project (that would simply log all messages that were routed through the kernel) it was never realised. In STUDIERSTUBE such is feature is missing completely, therefore some applications (like *Construct3D*) implemented this logging in the scope of their setup.

If this feature is grounded in the framework, which is the basis for the applications, all the applications and their interfaces could be more easily be evaluated.

7.10 Answering the "Why" Question

Distributed output can be very confusing for users. What happens where, how are the results of their interaction with the system is displayed. Especially when a *configuration* (4.9, *p.* 73) of the system is active that was not created by the users itself. The question "why did this happen" is often asked.

Although a number of rules try to diminish this problem through proper design, this situation may arise. Though the interaction design is (nearly) flawless, sometimes the users just do not know what happened, sometimes they even achieved something they did not try to achieve. Then a feedback from the system "why" and "how" the last action changed the environment can be very helpful.

So a computer system should be capable of answering this question. Example: a users activates a barcode, reacting on this action the system displays an image on one of the screens. If the user now asks the "why" question, some explanation for what happened should be available for the user.

An Example:

```
The barcode selected corresponds to a media entry in
the database, the active screen is display (name) -
  therefore this image is displayed on the screen
                        (name).
```

Additional help should also be provided like:

```
If you want the image to be displayed on a different
screen, please select one from the list by activating
a barcode or by speaking 'active display' and the name
of the display. Available displays are 'projection,
screen, ...'.
```

Most help systems focus on helping the user to perform an action. This normally includes "HowTo's" that explain how a specific task can be accomplished with the system. A more advanced help system will also be able to answer questions *after* the users has performed an action, explaining what happened and why.

This will lead to a deeper understanding of the system and enhance the abilities of the user to use the system [126]. This form of communication can also be found in human to human communication, when a person ask an other why she responded like she did. This form of *requested feedback* can be a helpful tool in the learning process of the users getting acquainted with a system. It also stresses the communicative aspect of a system as it can be asked for feedback at any time, by issuing a command that does not change the state of the system, it will only change the user's understanding of the system.

7.11 Conclusions

This thesis contributes to future developments of *multimodal interaction for configurable distributed* systems. By reflecting on the examples and applications presented in this thesis some insights were gained.

For a person with a technical background as a system designer and application developer the *Rules of Interaction Design* (2.4, *p.* 16) are most valueable. Other issues that were raised during the work on the two systems, led to the *Design Patterns for a multimodal, configurable, and distributed system* presented in this chapter. Some of them may be included in the ATELIER or STUDIERSTUBE frameworks in the near future.

The author hopes that he can contribute to that future developments and realise some of the described features. These features are an important basis to be able to create interesting and useful *multimodal, configurable, and distributed* applications.

Bibliography

[1] K. P. Åkesson, A. Bullock, C. Greenhalgh, B. Koleva, and T. Rodden. A toolkit for user re-configuration of ubiquitous domestic environments (demo). In *Companion to Proceedings of the 15th Annual ACM Symposium on User Interface Software and Technology.* ACM Press, 2004.

[2] A.R.T. - Advanced Realtime Tracking GmbH. Arttrack. http://www.ar-tracking.de/, July 2004.

[3] Atmel Corporation ©2004. Atmel website. http://www.atmel.com/, 10th September 2004.

[4] Austrian Research Institute for Artificial Intelligence (ÖFAI). http://www.ai.univie.ac.at/oefai/oefai.html, 2004.

[5] I. Barakonyi, T. Fahmy, and D. Schmalstieg. Remote collaboration using augmented reality videoconferencing. In *Proc. Graphics Interface 2004*, pages 89–96, May 17–19 2004.

[6] I. Barakonyi, T. Psik, and D. Schmalstieg. Agents that talk and hit back: Animated agents in augmented reality. In *Proc. ISMAR 2004*, page to be published, Washington, USA, October 2004. IEEE.

[7] I. Barakonyi and D. Schmalstieg. Ar puppet: Animated agents in augmented reality. In *First Central European International Multimedia and Virtual Reality Conference*, pages 35–42, May 6–8 2004.

[8] M. Bauer, B. Bruegge, G. Klinker, A. MacWilliams, T. Reichner, S. Riss, C. Sandor, and M. Wagner. Design of a component-based augmented reality framework. In *Proc. ISAR 2001*, pages 45–54, New York, New York, USA, October 29–30 2001. IEEE and ACM.

[9] M. Beaudouin-Lafon. Instrumental interaction: An interaction model for designing post-wimp user interfaces. In *Proceedings of the ACM*

CHI 2000 Conference on Human Factors in Computing Systems, pages 446–453. Association for Computer Machinery, 2000.

[10] V. Bellotti, M. Back, W. K. Edwards, R. E. Grinter, A. Henderson, and C. Lopes. Making sense of sensing systems: Five questions for designers and researchers. In *Proceedings CHI 2002, CHI Letters*, volume 1 of *1*, pages 415–422, Minneapolis, USA, 20-25 April 2002. ACM.

[11] S. Belongie, C. Carson, H. Greenspan, and J. Malik. Color- and texture-based image segmentation using EM and its application to content-based image retrieval. In *Proceedings of the Sixth International Conference on Computer Vision*, 1998.

[12] S. Benford, H. Schnadelbach, B. Koleva, B. Gaver, A. Schmidt, A. Boucher, A. Steed, R. Anastasi, C. Greenhalgh, T. Rodden, and H. Gellersen. Sensible, sensable and desirable: a framework for designing physical interfaces. Technical Report 03-003, Equator, 2003.

[13] T. Binder, G. D. Michelis, M. Gervautz, G. Iacucci, K. Matković, T. Psik, and I. Wagner. Supporting configurability in a tangibly augmented environment for design students. *Special Issue on Tangible Interfaces in Perspective, Pers and Ubiq Comp*, page *forthcoming*, 2004.

[14] N. Boujemaa, J. Fauqueur, M. Ferecatu, F. Fleuret, V. Gouet, B. L. Saux, and H. Sahbi. Ikona for interactive specific and generic image retrieval. In *Proceedings of International workshop on Multimedia Content-Based Indexing and Retrieval (MMCBIR'2001), Rocquencourt, France*, 2001.

[15] D. A. Bowman and L. F. Hodges. User interface constraints for immersive virtual environment applications. Technical Report 95–26, Graphics, Visualization, and Usability Center, 1995.

[16] D. Brown and G. Wheatley. Relationship between spatial knowledge. In C. Maher, G. Goldin, and R. Davis, editors, *Proceedings of the 11th Annual Meeting, North American Chapter of the International Group for the Psychology of Mathematics Education*, pages 143–148. Rutgers University, Brunswick, NJ, 1989.

[17] C. Browne. Sql databases. http://cbbrowne.com/info/rdbmssql.html.

[18] K. Buckner and M. Diaz. Review Report No: 3, ATELIER-Project IST-2001-33064 , 16th June 2004.

[19] M. Büscher, P. Mogensen, D. Shapiro, and I. Wagner. The Manufaktur: Supporting work practice in (landscape) architecture. In *Proceedings of the The Sixth European Conference on Computer Supported Cooperative Work (ECSCW 99)*, pages 21–40, Copenhagen, Denmark, 1999.

[20] W. Buxton. Integrating the periphery and context: A new model of telematics. In *In Proceedings of Graphics Interface '95*, pages 239–245, 1995.

[21] S. Calegari and M. Loregian. Ontologies help finding inspiration: a practical approach in multimedia information management. In *Proc. of PAKM2004*, 2004, to appear.

[22] K. Camarata, E. Y.-L. Do, B. R. Johnson, and M. D. Gross. Navigational blocks: navigating information space with tangible media. In *Proceedings of the 7th international conference on Intelligent user interfaces*, pages 31–38. ACM Press, 2002.

[23] A. L. Cambridge. The bat ultrasonic location system. http://www.uk.research.att.com/bat/.

[24] C. Carson, M. Thomas, S. Belongie, J. M. Hellerstein, and J. Malik. Blobworld: A system for region-based image indexing and retrieval. In *Third International Conference on Visual Information Systems*. Springer, 1999.

[25] Computer Grafik Insitut. The institute of computer graphics and algorithms, computer graphics group. http://www.cg.tuwien.ac.at/home/, July 2004.

[26] G. Crampton Smith. *The Hand That Rocks the Cradle.* I.D. magazine, May/June 1995.

[27] T. Darrell, P. Maes, B. Blumberg, and A. Pentland. A novel environment for situated vision and behavior, 1994.

[28] A. K. Dey, D. Salber, and G. D. Abowd. A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. *Human-Computer Interaction Journal*, 16(2-4):97–166, 2001.

[29] Dick Botting, California State University at San Bernardino. http://www.csci.csusb.edu/dick/c++std/, 10th September 2004.

[30] V. electronics. http://www.vtt.fi/ele/indexe.htm, 2004.

[31] C. Endres, A. Butz, and A. MacWilliams. A survey of software infrastructures and frameworks for ubiquitous computing. Jan-Mar 2005.

[32] J. A. Fails and D. R. Olsen. A design tool for camera-based interaction. In *Proceedings of the ACM CHI 2003 Conference on Human Factors in Computing Systems*, pages 449–456. Association for Computer Machinery, 2003.

[33] C. Faloutsos, R. Barber, M. Flickner, J. Hafner, W. Niblack, D. Petkovic, and W. Equitz. Efficient and effective querying by image content. *Journal of Intelligent Information Systems*, 3(3/4):231–262, 1994.

[34] J. Fauqueur and N. Boujemaa. Logical query composition from local visual feature thesaurus. In *Proceedings of Third International Workshop on Content-Based Multimedia Indexing (CBMI'03)*, 2003.

[35] E. Fennema and J. Sherman. Sex-related differences in mathematics achievement, spatial visualization, and affective factors. *Americal Educational Research Journal*, (14):51–71, 1977.

[36] Fitzmaurice. *Graspable User Interfaces.* PhD thesis, University of Toronto, 1996.

[37] G. W. Fitzmaurice, H. Ishii, and W. Buxton. Bricks: Laying the foundations for graspable user interfaces. In *CHI*, pages 442–449, 1995.

[38] A. Fuhrmann, D. Schmalstieg, and W. Purgathofer. Fast calibration for augmented reality. In *Proceedings of ACM Virtual Reality Software & Technology '99 (VRST'99)*, pages 166–167, December 20-22 1999.

[39] A. Fuhrmann, D. Schmalstieg, and W. Purgathofer. Practical calibration procedures for augmented reality. In *Proceedings of the 6th EUROGRAPHICS Workshop on Virtual Environments (EGVE 2000)*, pages 3–12, June 1-2 2000.

[40] B. R. Gaines. The technology of interaction: Dialogue programming rules. *International Journal of Man-Machine Studies*, 14:133–150, 1981.

[41] E. Goffman. *The Presentation of Self in Everyday Life.* Anchor, Doubleday, 1959.

[42] E. Goffman. *Behavior in Public Places.* Free Press, Macmillan, 1963.

[43] E. Goffman. *Frame Analysis: An essay on the organization of experience.* Northeastern University Press, 1974.

[44] E. Goffman. *Forms of Talk.* Oxford: Basil Blackwell, 1981.

[45] R. Guay and E. McDaniel. The relationship between mathematics achievement and spatial abilities among elementary school children. *Journal for Research in Mathematics Education*, (8):211–215, 1977.

[46] A. Gupta. The virage image search engine: an open framework for image management. In *Storage and Retrieval for Image and Video Databases IV, SPIE proceedings series*, volume 2670, pages 76–87, 1996.

[47] Hansen. User engineering principles for interactive systems. In *Proceedings Fall Joint Computer Conference*, pages 523–532. AFIPS Press, 1971.

[48] T. Heider and T. Kirste. Supporting goal-based interaction with dynamic intelligent environments. In *In Proceedings of the 15th Eureopean Conference on Artificial Intelligence, ECAI2002*, pages 596–600. IOS Press, July 2002.

[49] M. Hellenschmidt and T. Kirste. Sodapop: A software are infrastructure supporting self-organization in intelligent environments. In *In Proceedings of the 2nd IEEE International Conference on Industrial Informatics, INDIN04*, 2004.

[50] J. Henderson and S. Card. Rooms: The use of multiple virtual workspaces to reduce space contention in window-based graphical user interfaces. In *ACM Transactions on Graphics,*, volume 5, pages 211–241, July 1986.

[51] G. Hesina. *Distributed Collaborative Augmented Reality.* PhD thesis, Vienna University of Technology, May 2001.

[52] G. Hesina, D. Schmalstieg, and W. Purgathofer. Distributed open inventor: A practical approach to distributed 3D graphics. In *Proc. ACM VRST'99*, pages 74–81, London, UK, December 1999.

[53] L. E. Holmquist, F. Mattern, B. Schiele, P. Alahuhta, M. Beigl, and H.-W. Gellersen. Smart-Its Friends: A Technique for Users to Easily Establish Connections between Smart Artefacts. In *Proceedings of the*

3rd international conference on Ubiquitous Computing, pages 116–122. Springer-Verlag, 2001.

[54] E. Hornecker. *Tangible User Interfaces als kooperationsuntersttzendes Medium.* PhD thesis, published electronically at Elektronische Bibliothek, Staats und Universitätsbibliothek Bremen, July 2004.

[55] G. Iacucci, A. Juustila, K. Kuutti, P. Pehkonen, and A. Ylisaukko-oja. Connecting remote visits and design environment: User needs and prototypes for architecture design. In *In the Proceeding of Mobile HCI 03, Fifth International Symposium on Human Computer Interaction with Mobile Devices and Services*, pages 45–60. Lecture Notes in Computer Science, Springer Verlag, 8-11 September 2003.

[56] G. Iacucci and I. Wagner. Supporting collaboration ubiquitously: an augmented lerning environment for architecture students. In *Proceedings of the Eight European Conference on Computer Supported Cooperative Work (ECSCW) 2003*, pages 139–159, 2003.

[57] IBM. Viavoice®: You talk, it types®. http://www-306.ibm.com/software/voice/viavoice/, July 2004.

[58] Imagination Computer Services GesmbH. The invisible person. http://www.imagination.at/, July 2004.

[59] Image Retrieval Service (IRS) of the EVlib, http://visinfo.zib.de/irs.

[60] H. Ishii and B. Ullmer. Tangible bits: towards seamless interfaces between people, bits and atoms. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 234–241. ACM Press, 1997.

[61] H. Iwata, H. Yano, T. Uemura, and T. Moriya. Food simulator: A haptic interface for biting. In *Proceedings of the IEEE Virtual Reality 2004 (VR2004)*, pages 51–58, March 27-31, 2004.

[62] I. Jacob and J. Oliver. Evaluation of techniques for specifying 3d rotations with a 2d input device. In *Proceedings of HCI'95 Conference, People and Computers X*, pages 63–76, 1995.

[63] C. E. Jacobs, A. Finkelstein, and D. H. Salesin. Fast multiresolution image querying. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 277–286. ACM Press, 1995.

[64] P. Kabbash, W. Buxton, and A. Sellen. Two-handed input in a compound task. In *Proceedings of CHI 94 Conference on Human Factors in Computing Systems*, pages 417–423, 1994.

[65] M. Kalkusch, T. Lidy, M. Knapp, G. Reitmayr, H. Kaufmann, and D. Schmalstieg. Structured visual markers for indoor pathfinding. In *Proceedings of the IEEE First International Workshop on ARToolKit*, 2002.

[66] H. Kato, M. Billinghurst, I. Poupyrev, K. Imamoto, and K. Tachibana. Virtual object manipulation on a table-top ar environment. In *Proceedings of ISAR 2000*, 2000.

[67] H. Kaufmann. Dynamische geometrie in virtual reality. *Informationsblätter der Geometrie (IBDG)*, 21(1):33–37, 2002.

[68] H. Kaufmann. Collaborative augmented reality in education. Position paper for keynote speech at Imagina 2003 conference TR-188-2-2003-01, Technical University of Vienna, Feb. 3rd, 2003.

[69] H. Kaufmann. *Geometry Education with Augmented Reality.* PhD thesis, Vienna University of Technology, 2004.

[70] H. Kaufmann and D. Schmalstieg. Mathematics and geometry education with collaborative augmented reality. *SIGGRAPH 2002 Educators Program. In SIGGRAPH 2002 Conference Abstracts and Applications*, pages 37–41, 2002.

[71] H. Kaufmann, D. Schmalstieg, and M. Wagner. Construct3d: A virtual reality application for mathematics and geometry education. *Journal of Education and Information Technologies*, 5(4):263–276, 2000.

[72] H. Kaufmann, D. Schmalstieg, and M. Wagner. Construct3d: a virtual reality application for mathematics and geometry education. *Education and Information Technologies*, 5(4):263–276, 2000. TY - JOUR.

[73] P. M. Kelly and M. Cannon. Query by image example: The candid approach, los alamos national laboratory white paper, 1995.

[74] T. Kindberg. Implementing physical hyperlinks using ubiquitous identifier resolution. In *Proceedings of the eleventh international conference on World Wide Web*, pages 191 – 199. ACM Press, 2002.

[75] S. R. Klemmer, J. Graham, G. J. Wolff, and J. A. Landay. Books with voices: paper transcripts as a physical interface to oral histories. In *Proceedings of the conference on Human factors in computing systems*, pages 89 – 96. ACM Press New York, NY, USA, 2003.

[76] F. Ledermann. An authoring framework for augmented reality presentations. Master's thesis, Viennas University of Technology, 2004.

[77] F. Ledermann and D. Schmalstieg. Presenting past and present of an archaeological site in the virtual showcase. In *Proc. of the 4th International Symposium on Virtual Reality, Archeology, and Intelligent Cultural Heritage (VAST 2003)*, pages 119–126, Brighton, UK, Nov. 2003.

[78] M. Loregian and M. Telaro. Dynamic ontologies and cooperative learning. In *Supplements to Proceedings of COOP 2004, Hyères Les Palmiers, France, May 11-14, 2004. (http://klee.cootech.disco.unimib.it/Atelier/loregian-telaro.pdf)*, 2004.

[79] B. MacIntyre and S. Feiner. Language-level support for exploratory programming of distributed virtual environments. In *Proc ACM UIST'96*, pages 83–94, Seattle, WA, USA, Nov. 6–8 1996. ACM.

[80] B. MacIntyre, E. D. Mynatt, S. Voida, K. M. Hansen, J. Tullio, and G. M. Corso. Support for multitasking and background awareness using interactive peripheral displays. In *Proceedings of UIST'01*, pages 41–50, 2001.

[81] A. MacWilliams, C. Sandor, M. Wagner, M. Bauer, G. Klinker, and B. Bruegge. Herding sheep: Live system development for distributed augmented reality. In *Proc. ISMAR 2003*, pages 123–132, Tokyo, Japan, October 7–10 2003. IEEE.

[82] T. W. Malone. How Do People Organize Their Desks? Implications for the Design of Office Information Systems. *ACM Transactions on Office Information Systems*, 1(1):99–112, Jan. 1983.

[83] T. W. Malone, C. Fry, and K. Y. Lai. Experiments with oval: A radically tailorable tool for cooperative work. In *CSCW 92. Sharing Perspectives, Procceedings of the Conference on Computer-Supported Cooperative Work*, pages 289–297. ACM -Press, May 17–19 1992.

[84] V. Maquil. Automatic generation of graphical user interfaces in studierstube.

https://www.ims.tuwien.ac.at/publication_detail.php?ims_id=140, Juli 2004.

[85] K. Matković, L. Neumann, J. Siglaer, M. Kompast, and W. Purgathofer. Visual image query. In *Proceedings of the 2nd international symposium on Smart graphics*, pages 116–123. ACM Press, 2002.

[86] A. McLean, K. Carter, L. Lövstrand, and T. Moran. User-tailorable systems: Pressing the issue with buttons. In *Proceedings of the Conference on Computer Human Interaction (CHI 90)*, pages 175–182. ACM-Press, New York, April 1-5 1990.

[87] Microsoft Corporation. Microsofts speech api (sapi). http://www.microsoft.com/products/msagent/support/dev/speech.asp, September 2004.

[88] Y. Miyata and D. A. Norman. Psychological issues in support of multiple activites. In *User Centered Design*, pages 265–284, Lawrence Erlbaum, NJ, 1986.

[89] i.-T. U. P. D. Monitor. I-glasses PC HR. http://www.i-glassesstore.com/iglasses-pc-hr.html, October 2004.

[90] J. H. Morris, M. Satyanarayanan, M. H. Conner, J. H. Howard, D. S. Rosenthal, and D. F. Smith. Andrew: a distributed personal computing environment. In *Communications of the ACM archive, Volume 29, Issue 3 (March 1986)*, pages 184 – 201. The MIT Press scientific computation series, 1986.

[91] B. A. Myers, J. Nichols, J. O. Wobbrock, K. Litwack, M. Higgins, J. Hughes, T. K. Harris, R. Rosenfeld, and M. Pignol. Handheld devices for control. In *Human-Computer Interaction Consortium (HCIC'2003)*, Winter Park, CO, USA, Feb 5–9, 2003.

[92] E. Mynatt, T. Igarashi, W. Edwards, and A. LaMarca. Flatland: New dimensions in office whiteboards. In *In Proceedings of CHI'99*, pages 346–353, 1999.

[93] E. D. Mynatt. Writing on the wall. In *Proceedings of INTERACT'99*, pages 196–204, 1999.

[94] J. Newman, D. Ingram, and A. Hopper. Augmented reality in a wide area sentient environment. In *Proceedings of the IEEE and ACM International Symposium on Augmented Reality (ISAR'01)*. IEEE Computer Society, 2001.

[95] J. Nichols and B. A. Myers. Studying the use of handhelds to control smart appliances. In *Proceedings of the 23rd International Conference on Distributed Computing Systems Workshops (ICDCS '03)*, pages 274–279, May 19–22 2003.

[96] J. Nichols, B. A. Myers, M. Higgins, J. Hughes, T. K. Harris, R. Rosenfeld, and M. Pignol. Generating remote control interfaces for complex appliances. In *15th annual ACM symposium on User interface software and technology*, pages 161–170. ACM Press, 2002.

[97] D. Norman. *The Designs of Everyday Things.* Basic Books, 1988.

[98] D. A. Norman. Affordances, conventions and design. *Interactions*, 6:38–42, May-June 1999.

[99] Novell. Netware6: Printing meets the internet. http://www.novell.com/products/netware6/nw6_w_printing.html, July 2004.

[100] OMG Object Management Group. Unified modeling language (uml). http://www.uml.org/, 20th September 2004.

[101] Opencv–Intel Open Source Compter Vision Library, http://www.intel.com/research/mrl/research/opencv/.

[102] Origin Instruments. DynaSight Sensor. http://www.orin.com/3dtrack/dyst.htm, 2004.

[103] A. Pentland, R. Picard, and S. Sclaroff. Photobook: Content-based manipulation of image databases. In *SPIE Storage and Retrieval for Image and Video Databases II, number 2185, Feb. 1994, San Jose, CA.*, 1994.

[104] D. Petkovic, W. Niblack, M. Flickner, D. Steele, D. Lee, J. Yin, J. Hafner, F. Tung, H. Treat, R. Dow, M. Gee, M. Vo, P. Vo, B. Holt, J. Hethorn, K. Weiss, P. Elliott, and C. Bird. Recent applications of ibm's query by image content (qbic). In *Proceedings of the 1996 ACM symposium on Applied Computing*, pages 2–6. ACM Press, 1996.

[105] P. Petta, A. Staller, R. Trappl, S. Mantler, Z. Szalavári, T. Psik, and M. Gervautz. Towards Engaging Fullbody Interaction. In *Proceedings of the 8th International Conference on Human-Computer Interaction (HCI International '99)*, 1999.

[106] T. Pfeiffer and M. E. Latoschik. Resolving object references in multimodal dialogues for immersive virtual environments. In *Proceedings of the IEEE Virtual Reality 2004 (VR2004)*, pages 35–42, March 27-31, 2004.

[107] W. Piekarski, B. Gunther, and B. H. Thomas. Integrating virtual and augmented realities in an outdoor application. In *Proc. IWAR'99*, pages 45–54, San Francisco, CA, USA, October 21–22 1999. IEEE CS.

[108] W. Piekarski and B. H. Thomas. Tinmith-ev5 - an architecture for supporting mobile augmented reality environments. In *Proc. ISAR 2001*, pages 177–178, New York, New York, USA, October 29–30 2001. IEEE and ACM.

[109] W. Piekarski and B. H. Thomas. An object-oriented software architecture for 3D mixed reality applications. In *Proc. ISMAR 2003*, pages 247–256, Tokyo, Japan, October 7–10 2003. IEEE.

[110] W. Piekarski, B. H. Thomas, D. Hepworth, B. Gunther, and V. Demczuk. An architecture for outdoor wearable computers to support augmented reality and multimedia applications. In *Proc. Third International Conference on Knowledge-Based Intelligent Information Engineering Systems*, Adelaide, Australia, August 1999. IEEE.

[111] A. Prakash and M. J. Knister. A framework for undoing actions in collaborative systems. In *ACM Transactions on Computer-Human Interaction (TOCHI)*, volume 1, pages 295–330, 1994.

[112] J. Prümper and M. Anft. Die evaluation von software auf grundlage des entwurfs zur internationalen ergonomie-norm iso 9241 teil 10 als beitrag zur partizipativen systemgestaltung - ein fallbeispiel. In K. H. Rödiger, editor, *Software-Ergonomie '93: Von der Benutzungsoberfläche zur Arbeitsgestaltung*, pages 145–156. Teubner, Stuttgart, 1993.

[113] T. Psik, K. Matković;, R. Sainitzer, P. Petta, and Z. Szalavári. The Invisible Person: advanced interaction using an embedded interface. In *Proceedings of the workshop on Virtual environments 2003*, pages 29–37. ACM Press, 2003.

[114] The State Hermitage Museum, St. Petersburg, Russia, QBIC Color and Layout Search. http://www.hermitagemuseum.org/fcgi-bin/db2www/qbicSearch.mac/qbic?selLang=English.

[115] G. Reitmayr. *On Software Design for Augmented Reality.* PhD thesis, Vienna University of Technology, 2004.

[116] G. Reitmayr and D. Schmalstieg. Mobile collaborative augmented reality. In *Proc. ISAR 2001*, pages 114–123, New York, New York, USA, October 29–30 2001. IEEE.

[117] G. Reitmayr and D. Schmalstieg. OpenTracker – an open software architecture for reconfigurable tracking based on XML. In *Proc. IEEE Virtual Reality 2001*, pages 285–286, Yokohama, Japan, March 13–17 2001.

[118] G. Reitmayr and D. Schmalstieg. Collaborative augmented reality for outdoor navigation and information browsing. *Geowissenschaftliche Mitteilungen (Proc. 2nd Symposium on Location Based Services and TeleCartography)*, pages 53–62, 2003.

[119] M. Resnick, F. Martin, R. Berg, R. Borovoy, V. Colella, K. Kramer, and B. Silverman. Digital manipulatives: New toys to think with. In *CHI*, pages 281–287, 1998.

[120] T. Rodden, A. Crabtree, T. Hemmings, B. Koleva, J. Humble, K.-P. Åkesson, and P. Hansson. Configuring the ubiquitous home. In *Proceedings of the 6th International Conference on Designing Cooperative Systems.* IOS Press, 2004.

[121] D. Schmalstieg and G. Eibner. Hybrid user interfaces using seamless tiled displays. In *Proc. of 8th Immersive Projection Technology Workshop (IPT 2004)*, May 13-14 2004.

[122] D. Schmalstieg, A. Fuhrmann, G. Hesina, Z. Szalavári, L. M. Encarnao, M. Gervautz, and W. Purgathofer. The Studierstube augmented reality project. *PRESENCE - Teleoperators and Virtual Environments*, 11(1), 2002.

[123] D. Schmalstieg, A. Fuhrmann, Z. Szalavári, and M. Gervautz. Studierstube – an environment for collaboration in augmented reality. In *Proc. of Collaborative Virtual Environments Workshop '96*, Nottingham, UK, September 19–20 1996.

[124] D. Schmalstieg and G. Hesina. Distributed applications for collaborative augmented reality. In *Proc. IEEE VR 2002*, pages 59–66, Orlando, Florida, USA, March 24–28 2002. IEEE.

[125] D. Schmalstieg, G. Reitmayr, and G. Hesina. Distributed applications for collaborative three-dimensional workspaces. *Presence*, 12(1):53–68, February 2003.

[126] B. Sheiderman. *Desiging the User Interface.* Addison Wesley Longman, 3rd edition, 1998.

[127] B. Shneiderman. Direct manipulation: A step beyond programming languages. *IEEE Computer*, 16:57–63, 1983.

[128] http://www.smartits.org/.

[129] U. Spierling, D. Grasbon, N. Braun, and I. Iurgel. Setting the scene: playing digital director in interactive storytelling and creation. In *Computers & Graphics*, number 1 in 26, pages 31–44, 2002.

[130] O. Stiemerling, H. Kahler, and V. Wulf. How to make software softer - designing tailorable applications. In *Proceedings of the DIS (Designing Interactive Systems) '97*, pages 365–376. ACM Press, 1997.

[131] P. Strauss and R. Carey. An object oriented 3D graphics toolkit. In *Proc, ACM SIGGRAPH'92.* ACM, 1992.

[132] Sun Microsystems, Inc. . http://java.sun.com/, 10th September 2004.

[133] H. Suzuki and H. Kato. Algoblock: a tangible programming language, a tool for collaborative learning. In *Proceedings of 4th European Logo Conference*, pages 297–303, Aug. 1993.

[134] Systems in Motion. Coin 3d library. http://www.coin3d.org/, April 5th 2004.

[135] Z. Szalavári and M. Gervautz. The Personal Interaction Panel — A two-handed interface for augmented reality. *Computer Graphics Forum*, 6(13):335–346, 1997.

[136] The ATELIER Consortium. The atelier website. http://atelier.k3.mah.se/, 10th September 2004.

[137] The Disappearing Computer Initiative ©2002 -2003. The disappearing computer initiative. http://www.disappearing-computer.net/, 10th September 2004.

[138] The Quantum Research Group. Touch control and sensor ics. http://www.qprox.com/.

[139] H. Tramberend. Avocado: A distributed virtual reality framework. In *Proc. IEEE VR 1999*, pages 14 - 21, Houston, Texas, USA, March 13 - 17 1999. IEEE.

[140] H. Tramberend. Avango: A distributed virtual reality framework. In *Proc. of Afrigraph'01*, 2001.

[141] X. Tu and D. Terzopoulos. Artificial fishes: Physics, locomotion, perception, behavior. *Computer Graphics*, 28(Annual Conference Series):43–50, 1994.

[142] E. Tuulari and A. Ylisaukko-oja. Soapbox: A platform for ubiquitous computing research and applications. In *Proceedings of Pervasive Computing 2002*, pages 125–138, 2002.

[143] B. Ullmer and H. Ishii. The metadesk: Models and prototypes for tangible user interfaces. In *ACM Symposium on User Interface Software and Technology*, pages 223–232, 1997.

[144] B. Ullmer, H. Ishii, and D. Glas. mediaBlocks: Physical containers, transports, and controls for online media. *Computer Graphics*, 32(Annual Conference Series):379–386, 1998.

[145] A. Vailaya, Y. Zhong, and A. Jain. A hierarchical system for efficient image retrieval,. In *Procedeengs of International Conference on Pattern Recognition (August 1996).*, 1996.

[146] Van Gogh TV. Piazza virtuale. http://www.aec.at/de/archives/prix_archive/prixJuryStatement.asp?iProjectID=2585, 1993.

[147] VTT Electronics. Kaitoväylä 1, P.O. Box 1100, 90571 Oulu, Finland, 2004.

[148] W3C World Wide Web Consortium ®. Extensible markup language (xml). http://www.w3c.org/XML/, 20th September 2004.

[149] W3C World Wide Web Consortium ®. Hypertext markup language (html). http://www.w3c.org/MarkUp/, 20th September 2004.

[150] M. Weiser. The computer for the 21st century. In *Scientific American*, volume 265 of *3*, pages 94–104, 1991.

[151] P. Wellner, W. Mackay, and R. Gold. Back to the real world. In *Communications of the ACM, Vol. 36, Number 7*. ACM Press, 1993.

[152] J. Wernecke. *The Inventor Toolmaker: Extending Open Inventor.* Addison-Wesley, 2nd edition, April 1994.

[153] Yahoo! Inc. Yahoo messenger. http://messenger.yahoo.com/, July 2004.

[154] Y. Yanagida, S. Kawato, H. Noma, A. Tomono, and N. Testunani. Projection-based olfactory display with nose tracking. In *Proceedings of the IEEE Virtual Reality 2004 (VR2004)*, pages 43–49, March 27-31, 2004.

Curriculum Vitae

Thomas Psik

Piaristengasse 2-4
A-1080 Vienna
Austria
tpsik@pop.tuwien.ac.at

1972	Born on the 6^{th} of September in Vienna, Austria
1978 – 1983	Primary School (Volksschule) at the Volksschule Zeltgasse, Vienna, Austria
1983 – 1991	Secondary School at the Bundesgymnasium Albertgasse, Vienna, Austria
June 1991	Graduation (Matura) from the Bundesrealgymnasium Albertgasse
1991 – 1998	Studies in computer science at the Vienna University of Technology
1. Dez. 1998	Graduation "Diplom-Ingenieur der Informatik" from the Vienna University of Technology Thesis "Nichtlineare Raumverkrümmung beim Raytracing"
1999 – 2002	Employee of Imagination Computer Services GmbH
2002 – 2004	Research Assistant in the scope of the EU-IST project ATELIER at the "Institut für Gestaltungs- und Wirkungsforschung, Arbeitsbereich CSCW", Vienna University of Technology (Univ.Prof. Wagner)
2003 – 2004	Research Assistant in the scope of the START project at the "Institut für Software Technologie and Interaktive Systeme", VR Group, Vienna University of Technology (Ao.Prof. Schmalstieg)
October 2004	Dissertation "Designing multimodal interaction for configurable distributed systems", advisor: Ina Wagner