Integrated Project on Interaction and Presence in Urban Environments

FP6-2004-IST-4-27571

ipcity.eu

Initial Demonstrators of Interaction Tools

Deliverable D4.1

Doc-Id:	**D 4.1**
Version:	**1.0**
Author(s):	**Jan Ohlenburg,** **Wolfgang Broll,** **Antti Juustila,** **Markus Sareika,** **Valérie Maquil**
Date:	**2007-02-08**
Status:	**Final**
Availability:	**Public**
Distribution:	**Project Partners / EC / Web**

Table of Content

Abstract

The cross-reality interaction tools research workpackage focuses on support of mixed reality user interface creation, development, and execution. In contrast to traditional user interfaces mixed reality user interfaces are typically not limited to one or two particular devices, but rather use a large variety of individual devices. No standard interaction techniques have yet been established in the area of mixed realities – unlike the WIMP metaphor on windows desktop systems. This especially applies to mobile mixed reality environments. Additionally, interaction techniques often involve multiple modalities. Beside this, individual platforms require individual solutions, while they should at least partially be exchangeable for the individual user. Support for user interfaces and interactions should however not be limited to system and implementation aspects, but also include appropriate support for authoring, or else creators of mixed reality content will be overwhelmed by the complexity of the process.

In that sense we have started developing a couple of interaction and authoring tools to create, author and evaluate mixed reality user interfaces. A couple of demonstrators for device independent application development have been created, such as DEVAL, OpenTracker and OpenVideo, as well as a tool for collaborative planning environments using colored tangible user interfaces, the ColorTable. Additionally, a tool for authoring, orchestration and evaluation of public demonstrations by augmenting the map of the event area has been proposed to the showcases and two tools, which focus on interaction on Symbian mobile phones.

This document summarizes the scientific and technical achievements in workpackage 4 during the first year. According to the internal report I4.1 which includes a requirements analysis for WP 4, first demonstrators have been developed which have already been tested within several showcase.

Intended Audience

This document is intended to all partners of the project, the EC, and to the reviewers for the first project's phase.

1 Workpackage Objectives

Objectives Phase I	Development of a set of generic tools for supporting interactions, design of multi-model user interfaces and authoring of interactive Mixed Reality environments. A requirement analysis of the individual showcases will be the basis for this development.
Results Phase I	During the first phase, the following tools have been developed and are available as prototypes for the showcases: • **Interaction Prototyping Tool:** A component-based approach, which allows for modeling interaction techniques and object behavior from a set of basic components. • **AuthOr:** A map augmentation tool for authoring, orchestration and evaluation. • **DEVAL:** The DEvice Abstraction Layer classifies input and output devices in a hierarchy of functional interfaces for device independent application development. • **OpenTracker:** A generic tracking framework implemented with pipes and filters dataflow pattern. For IPCity OpenTracker has been extended by several new modules. • **OpenVideo:** Similar to OpenTracker it uses pipes and filters for providing access to video streams. • **Color Table:** Providing collaborative working scenarios by providing tangible interaction possibilities on an augmented table. • **MMS Entrance:** A smartphone application that extracts contents of multimedia messages and forwards them together with meta information to a PC system. • **Extendable DataMatrix reader:** Reads two dimensional DataMatrix barcodes and passes the data to application specific plugin components for processing.
Evaluation Results Phase I	The tools have been developed in the lab and have been partially tested as tech

	probes within the showcase.
Objectives Phase II	After an initial set of tools has been developed and most of them already have been used within the showcases, these tools have to be redesigned based on the experience and the results of the evaluation from the showcases. Additional tools will be designed and developed according to the needs of the showcases. The redesigned and new tools will be delivered to the showcases to be included within phase two. The focus of our work will be on the following topics and tools due to demands from the showcases: • **Interaction Prototyping/Authoring:** A graphical user interface on top of the language describing the interactions will be developed. This will be a major building block together with the language to support easy creation and evaluation of new interaction mechanisms. • **Authoring and Orchestration Interface:** This tool supports the showcases by augmenting arbitrary maps with 2D information, e.g. text, objects, users. The functionality can be used to author a showcase event as well as orchestrating and monitoring the running event and evaluating an event by playback functionality. • **Color Table:** Based on the feedback of users, we will further develop the interaction with this Tangible AR Setup. Adaptations of the interaction will be based on further feedback during the planned workshops and methods that are developed in WP3 • **Audio/Video Streaming:** Publishing arbitrary audio and video sources to local and remote hosts in an efficient way, while providing a simple interface in order to access a stream. Integrate the streaming into the device abstraction. • **Device-independent user interfaces:** Describe user-interfaces independent of the final execution development and devices available using a mark-up language and/or a MR interaction framework. • **Mobile content tools:** A mobile tool for entering media (images, video, sound) into the MR environments. Also includes

	a PC/server side components for importing the media wirelessly (short range wireless connection). • **Two dimensional, extendable mobile tag reader for smartphones:** a tool for reading two dimensional bar codes for various purposes.

2 State-of-the-Art

The state-of-the-art report covers all research areas related to Cross-Reality Interaction and Authoring Tools. We have identified a number of key research areas, which are already listed in the Technical Annex. In each of the research areas we describe the current available and relevant technologies and approaches. This should help the showcases to get a good overview about what they can use from the shelf and were additional research has to be conducted.

2.1 Device abstraction

Especially the device abstraction concept will be one of the key building blocks on which the different tools and showcases will rely on. By classifying the functionality of devices and map this classification onto a hierarchy, the device concept will be able to define common interfaces for each node within the hierarchy. Due to dividing the functionality of a device into common interfaces, the application is able to abstract from a specific device and concentrate on functionality, which is provided by a range of devices via the common interface. Devices can also be partially or fully virtualized, i.e., implemented as a combination of hardware and software.

The resulting hierarchy will not only contain the common interfaces, but also the specialized interfaces, which provide the unique functionality of a specific device. Therefore it is possible to utilize a device, which is custom-tailored for a special task, while on the other hand being able to exchange device, which provide the same interfaces. E.g. a turning knob would be exchangeable by a mouse wheel. In order to exchange devices with very different functionality, e.g. substituting a speech recognition input by a keyboard input in a noisy environment, design patterns will be defined.

2.1.1 OpenTracker

OpenTracker [28] is a generic framework for accessing and manipulating tracking data. It implements the well-known pipes & filters dataflow pattern and provides an open software architecture based on a highly modular design and an XML configuration syntax. OpenTracker incorporates drivers to most commercial tracking devices. Using OpenTracker, MR developers can configure their tracking setup (including data fusion from multiple sources) with a few lines of XML code. Switching between different tracking setups does not require changes to the application code; instead, editing of a single XML file is sufficient. Since OpenTracker provides a network data-transport mechanism, mobile devices can easily access outside-in tracking hardware via Wireless LAN.

2.1.2 OpenVideo

OpenVideo is a general data integration and processing software with special support for video data. It implements a hardware abstraction layer by interfacing several different device drivers either directly or through the functionality of third party video libraries. OpenVideo is designed to be as extensible and easily configurable as possible. OpenVideo is currently implemented on windows and on linux systems.

OpenVideo's functionality is split into two parts, the 'Core' and the 'Nodes' modules. The Core module is responsible for setting up the runtime environment as well as for processing the various entities in OpenVideo's runtime data structure. This runtime structure is implemented as a directed acyclic graph which consists of nodes and edges.

2.1.3 VRPN

The Virtual-Reality Peripheral Network (VRPN) [14][53] is an open-source project for various input and output devices. It provides a network transparent interface for application development to get access to physical devices and allows the dynamic discovery of them. In

regards to the device abstraction it has only limited functionality for exchanging devices required for MR interactions.

2.1.4 DirectShow / DirectInput

Microsoft provides two device abstraction layer APIs as part of DirectX. DirectShow [12] is an abstraction layer for streaming media such as video and audio. By using DirectShow, the application developer is able to access a a wide variety of different devices to be handled equally, independent of the source stream. This can be an internet stream, a file or direct input from a device. This abstraction is achieved by providing a common interface, which has to be implemented by all devices and components that want to provide an appropriate audio or video stream. In case a component wants to upstream data, it has to implement particular a interface in conjunction with a filter.

The second abstraction layer of DirectX for mouse, keyboard and joystick (including force feedback) devices is the DirectInput API. This abstraction layer allows direct access to these devices without using the Windows messaging mechanisms. Currently most available devices for Windows support this standard and hence it is most convenient for application developer. The API provides functionalities like iterating through the available devices and acquiring the state of a device even if the application is in background mode. Additionally, an application can use action mapping to retrieve and map input data without the need to know which device generated it.

The obvious limitation of both APIs is the missing platform and compiler independency and the limited access to special device functionality, which is not covered by the interfaces. Especially professional video cameras provide much more functionality than offered by DirectShow.

2.2 Device-independent user interfaces

There is a large variety of interaction devices that are employed for MR applications, such as head-mounted displays, PDAs, space mice, tracked placeholder objects, microphones, gestures and so on to name just a few.

Many MR applications are hard-coded for a specific set of input and output devices, but there are efforts to decouple MR applications from the underlying hardware by defining device-independent user interfaces.

Device-independent user interfaces are abstract specifications of user interfaces that can be instantiated for different hardware and software platforms. An overview of existing UIDLs is provided in [15].

2.2.1 UIML

UIML (User Interface Markup Language, [1]) is an XML-compliant meta-language that provides a highly device-independent method to describe user interfaces. Interface-specific properties are separated into a vocabulary that is used within UIML descriptions. [13]) is an extension of UIML towards collaborative user interfaces.

2.2.2 MRIML

MRIML (Mixed Reality Interface Markup Language, [3]) – developed by Fraunhofer FIT – is a UIDL destined to describe user interfaces independent of a specific platform. Despite of its name, MRIML generally is not limited to 3D or MR user interfaces, but also provides support for more traditional WIMP-style user interfaces. Thus MRIML focuses on widget-based interface elements, while providing specific elements and extensions for MR environments. The basic mechanism to use MRIML is to describe a user interface by the appropriate elements provided by the MRIML vocabulary. This vocabulary can either be used in conjunction with UIML or within an independent XML-compliant MRIML description.

2.2.3 Exchangeability of 3D interface components

As it has been noted by Lindt, device abstraction layers are not sufficient to realize device-independent 3D user interfaces [10], since applications typically rely on specific input and output data. For example, exchanging a ray-cast interaction technique for selecting a 3D object with a speech command based interaction technique would not work, since the application would require pose tracking data rather than recognized speech commands. For realizing 3D user interfaces that are independent of particular interaction devices, appropriate abstraction layers for 3D interaction techniques and 3D widgets are required.

2.2.4 PUC

To further extend the input possibilities of Studierstube, a bridge between the Personal universal controller (PUC) framework [35] 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 [35][36] a system developed as an approach for improving the interfaces to complex appliances by introducing an intermediary graphical or speech interface. This system, called personal universal controller (PUC) automatically generates a user interface to serve as a remote control for any application. 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.

The PUC framework explicitly allows multidevice 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. There is actually no conflict handling, 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 setting).

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 visualized by the application using the graphic tools available, but some times is useful to display data also on the handheld. In Studierstube a similar auto GUI generation algorithm has been implemented [37]. Figure 1 shows a scripted PUC node in Studierstube.

```
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 1 - Example of a PUC definition in Stb

Using the PUC framework to describe an interface, allows rapidly developing and changing 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.

These widgets are then being displayed on the PIP and can be used to control the application. That 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.

2.2.5 Atelier

The infrastructure software platform in Atelier framework is a generic platform that allows different kinds of technologies to be integrated to the environment. This allows extending and also changing 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 must 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 platform is discussed more in 0.

Atelier as such does not aim to solve device independence in user interfaces, but enables application architecture to be designed so that services can be isolated from the devices that enables users to interact with the service. Thus the user interface can be a physical device with no “traditional” computing user interface such as the mouse and keyboard, but for example a barcode reader or any physical object instrumented with, say, a RFID tags and readers. The role of Atelier in device independent user interfaces is thus in the architecture design of the application or system.

2.2.6 CAPNET Device independent UI component

CAPNET [39] is a mobile middleware platform that supports the development of ubiquitous applications and services for mobile phones. The architecture includes functionality for interoperability, discoverability, location transparency, adaptability, and context-awareness. The CAPNET architecture includes an application independent GUI component for displaying simple graphical user interface widgets and offers interaction between the user and the application in a distributed way using XML-RPC. The application may reside in another device than the UI component. There are both J2ME [40] and Symbian OS implementations of the GUI component, based on XML based XUL [38]. The CAPNET GUI component can be integrated with the Atelier infrastructure since they both are based on the requirements of distributed application architectures and XML.

2.3 Interaction Prototyping/Authoring

The first attempts to support rapid prototyping for 3D graphics were based on text file formats and scripting languages such as Open Inventor [31], VRML [33] or X3D [34]. New types of objects and behaviors can only be added by implementing them in C++ and compiling them to native code. While scriptable frameworks represent an improvement in the workflow of programmers, who can create application prototypes without the need to compile code, they do not offer the necessary concepts and abstractions for controlling an application's temporal structure and interactive behavior, and provide no built-in support for AR/VR devices.

Platforms like Avango [32] or Studierstube [29] add the necessary classes to such frameworks to support the creation of AR/VR applications. However, from the perspective of an author the power of these frameworks further complicates matters rather than providing the required level of abstraction.

2.3.1 Behaviors

Behaviors (developed by Fraunhofer FIT) [3] represent a component-based approach for realizing interaction prototypes. A text-based description allows the developer to model user interactions or autonomous object behaviors from a set of existing components. These components represent integral elements of user interaction and object behavior. This description is interpreted and executed by an AR or VR environment at runtime. Behavior objects communicate with other system or scene graph components by events only. Thus specific components exist to react to events related to particular scene graph geometry or issued for this Behavior, or to register for arbitrary input. Other components allow performing calculations, time-dependent behaviors, interpolations, etc. In order to apply changes to the scene graph or other system components, a component for issuing appropriate events exist. The control flow between the individual components of a single Behavior object is specified using a signal/slot mechanism. Thus, the components form a dataflow graph. In addition to these signal connections, data connections may be specified to transfer data values between component data fields. These transfers allow for implicit type conversions.

2.3.2 The APRIL Language

APRIL, the Augmented Reality Presentation and Interaction Language, provides elements to describe the hardware setup, including displays and tracking devices, as well as the content of the application and its temporal organization and interactive capabilities. Rather then developing APRIL from scratch, the authoring and playback facilities were built on top of the existing *Studierstube* [29] runtime system. The XML-based language allows expressing all aspects needed to create compelling interactive AR content.

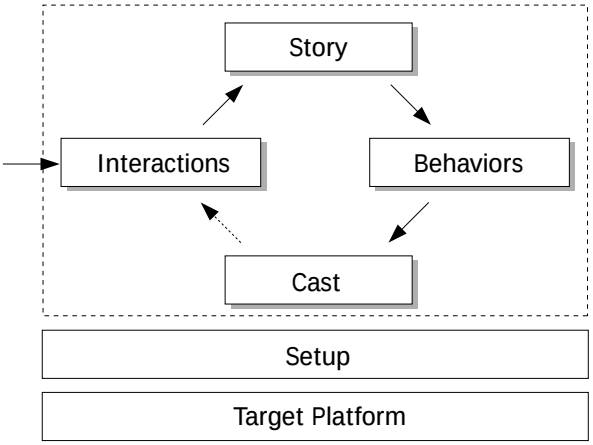

Figure 2 - The main components of APRIL.

The main aspects that contribute to an application are encapsulated in four top-level

elements: *cast, story, behaviors* and *interactions* which are visualized in Figure 2. They can be easily exchanged, allowing customization of the application for different purposes. The *story* is an explicit representation of the temporal structure of the application, composed of individual *scenes*. In each scene, a predefined sequence of *behaviors* is executed by *actors*, which are instances of reusable components which expose certain fields for input and output. The transitions that advance the story from one scene to the next are triggered by user interaction, potentially provided by interaction components.

Applications are modeled with UML state charts, a formalism that has been used successfully by other projects like alVRed [17] and for which professional graphical modeling tools exist. UML state charts can be hierarchical and concurrent, meaning that a state can contain sub-states, and there might be several states active at the same time (see Figure 3).

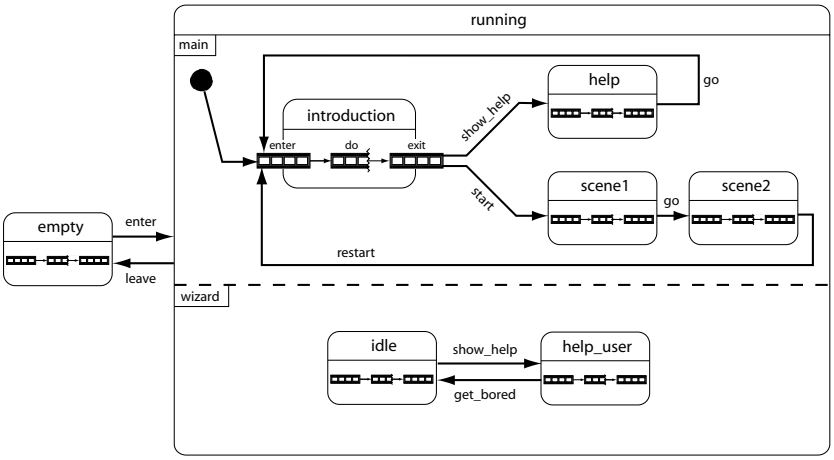

Figure 3 - The storyboard of a simple APRIL application, modeled as a UML state diagram. For the "introduction" scene, the three timelines enter, do and exit are emphasized.

Enumerating all elements and features that APRIL provides is beyond the scope of this report. Interested readers are referred to [9], where detailed information and the APRIL schema specification can be found.

APRIL based Applications:

Magic Book story

We have made significant effort to enable the creation of visually attractive AR applications from simple yet powerful building blocks. These building blocks offer an easy way to facilitate the rendering of various multimedia elements such as 3D models, sound and animation, and support elementary user interaction with these components using standard AR techniques and hardware setups.

The Magic Book metaphor [18] is popular and well-known in the AR community as a basic yet attractive application, in which users browse through a real book to view animated virtual content displayed on top of fiducials printed on the pages and tracked by a camera (see Figure 4). Using built-in APRIL components such as model to display and control visual parameters of a 3D model, sound to render background music or sound effects, canvas to display 2D textures on a 3D image plane or label to display 2D text information in AR scenes allow authors to quickly compose simple sequential stories with rich multimedia content.

User interaction in the Magic Book application relies solely on predefined APRIL methods. The button action method automatically renders a virtual button (the use of a real button is also possible) with a user-defined caption and defines a navigation point to jump to a desired scene in the presentation sequence. The touch method checks when a user-manipulated fiducial serving as a 3D cursor enters or exits a 3D area surrounding another marker printed on a real book page. This area is defined by the hotspot component. To enable the use of fiducial-based pose tracking only the configurable hardware setup description file for optical

tracking needs to be completed with the appropriate marker and camera parameter information. The control element connects the pose of a certain marker to an application object, the visibility attribute of which can be modified by the set method depending on the current story state to show or hide it on the display.

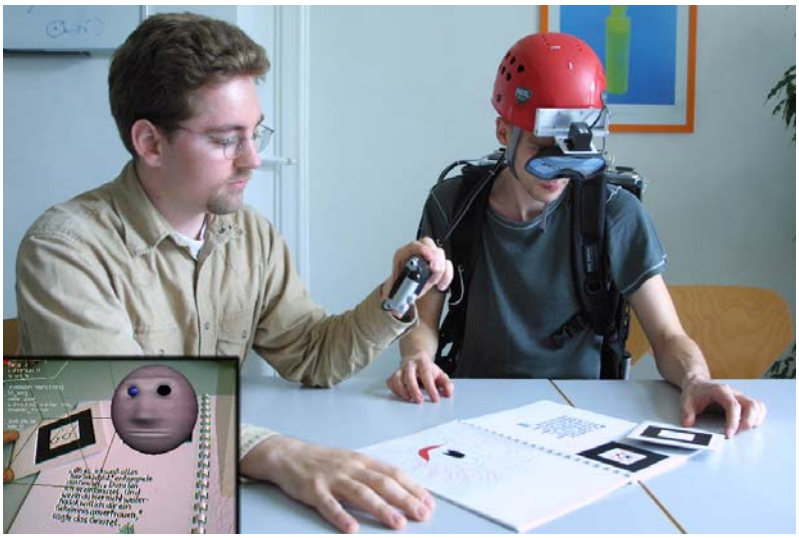

Figure 4 - Two users with different AR platforms using the same application, a "Magic Book" created with the APRIL authoring toolkit.

Virtual Showcase

The Virtual Showcase has been designed to show interactive augmented reality presentations in a museum setting by overlaying virtual information on top of physical exhibition objects. User interfaces for museum visitors impose numerous requirements and constraints onto content authors: they should be intuitive and easy to handle, responsive, attractive and rich in multimedia elements, and should render a user-specific view for multiple visitors. Our demonstration application presents the architectural highlights and the historical background of the Austrian archaeological ruin Heidentor (see Figure 5).

The hardware setup of the Virtual Showcase application is prepared to render individual views for up to four users using built-in CRT monitors. The displayed virtual information is merged with a scale model of the Heidentor by half-silvered mirrors making up the walls of a pyramidical showcase. Shutter glasses, tracked by a magnetic tracking system, ensure a stereo view of virtual augmentations correctly aligned with the physical model and matching the users' viewpoint. Visitors can interact with the scene using a trackball, physical and virtual buttons and a tracked pen serving as a raypicker, and select the desired type of information such as imaginary reconstructions of the ruin, historical images or an explanatory narrative about the real model.

Although such a setup normally requires a complex combination and configuration of hardware devices and software tools, APRIL provides a simple way to compose the setup description file for Virtual Showcase. The screen and display elements allow the configuration of virtual information rendering with adjustable parameters such as resolution, stereo rendering, head tracking, raypicking device and mouse pointer. The definition of stage elements enables the easy integration and registration of the physical model of the Heidentor with its virtual counterpart, and the rendering of an optional head-up display to display viewpoint independent 2D information.

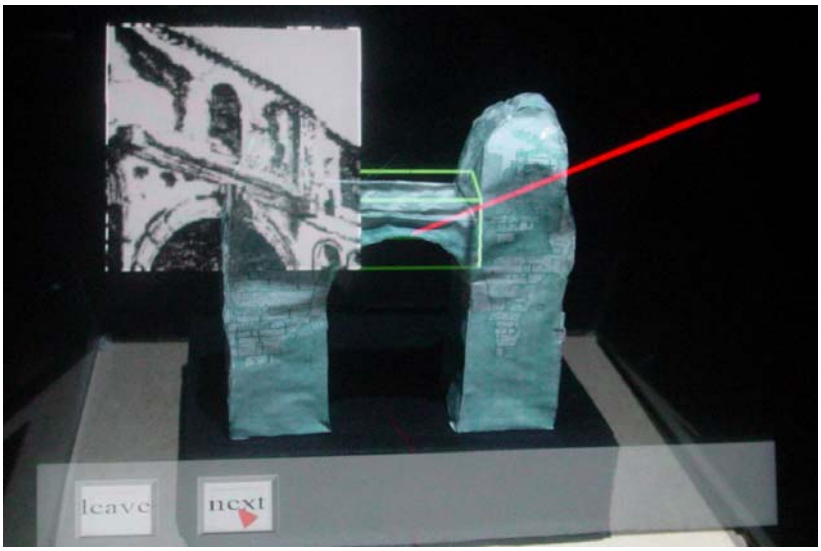

Figure 5 - An archaeological ruin inside the Virtual Showcase. A raypicker is used to select parts of the real object for retrieving further information. A projector is used to cast virtual shadows onto the physical model.

Virtual Tour Guide

The Virtual Tour Guide application embeds a virtual animated character acting as a tour guide into a mobile indoor navigation application called Signpost (Reitmayr & Schmalstieg, 2003). The user wears a mobile AR setup integrated into a backpack and a helmet, and perceives the augmented world through a head-mounted display (HMD). A camera mounted above the HMD tracks fiducials placed onto walls of the building area covered by the application. The markers help locate the user within this area since the system knows their exact position in a precisely measured virtual model of the building that has been registered with its real counterpart.

The virtual tour guide character is placed into the reference frame of the real building. While walking around, the character provides assistance to find selected destinations and provides location-specific explanation about the content of various rooms and people working in them using body gestures (e.g. looking towards, pointing, asking the user to follow, etc.), 2D and 3D visual elements and sound. Since the tour guide is aware of the building geometry, it appears to walk up real stairs and go through real doors and walkways, thus further enhancing integrity with the user's physical environment.

The tour guide character is controlled by the AR Puppet framework that is a hierarchical character animation system enabling the use of embodied animated agents in AR applications. Both AR Puppet and the Signpost navigation application are large systems with a complex network of internal modules responsible for various subtasks, therefore it is difficult and undesirable to modify their internal structure in order to establish communication between them. Relying on the APRIL framework's component model and turning AR Puppet and Signpost into custom APRIL components enables the encapsulation of these frameworks' functionality. These components can be used as black boxes that expose relevant input and output fields for communication with other, external components while hiding internal implementation details. Figure 6 illustrates the fields exposed by Signpost and monitored by AR Puppet to provide the tour guide character with relevant navigation information.

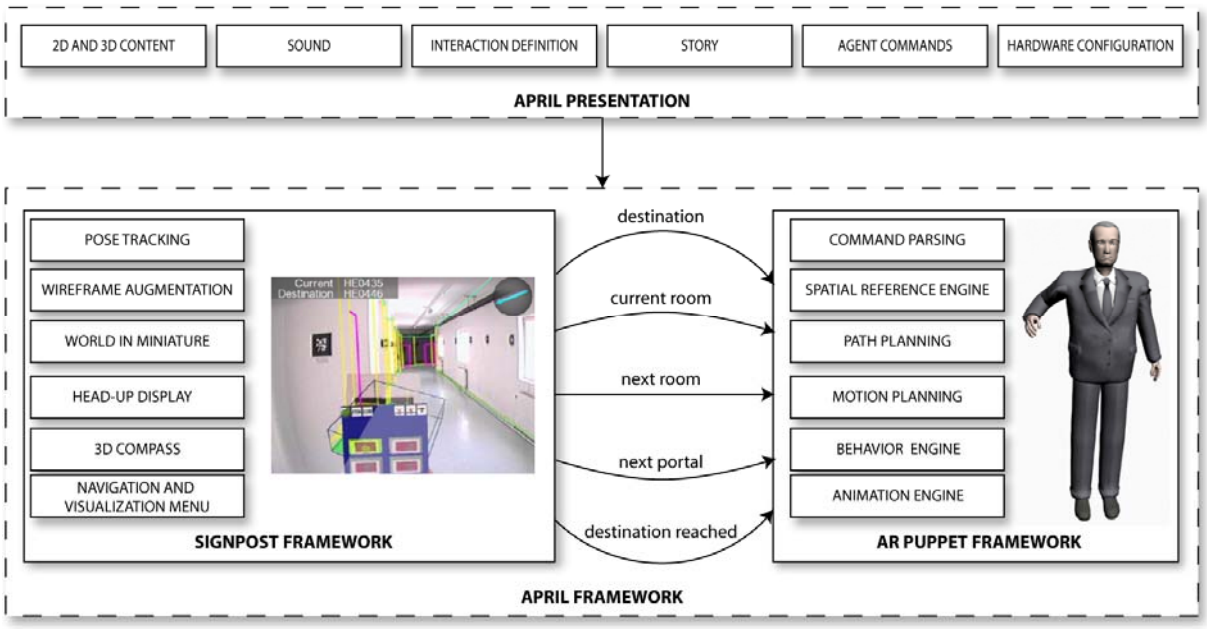

Figure 6 - Communication between the AR Puppet and Signpost systems within the APRIL framework.

The expensive and bulky mobile AR system required by the Signpost application in its original form makes content and application authoring, debugging and testing a difficult task, therefore we needed to develop a desktop simulator system that is able to run the same navigation application with simple keyboard input and screen based output. The hardware abstraction feature of APRIL conveniently hides details such as the type of display or exact tracking setup from authors and components. Only symbolic names are used that allows exchanging the internal implementation of the hardware setup. See Figure 7 for an illustration of the Virtual Tour Guide application running on the desktop simulator and the mobile AR system.

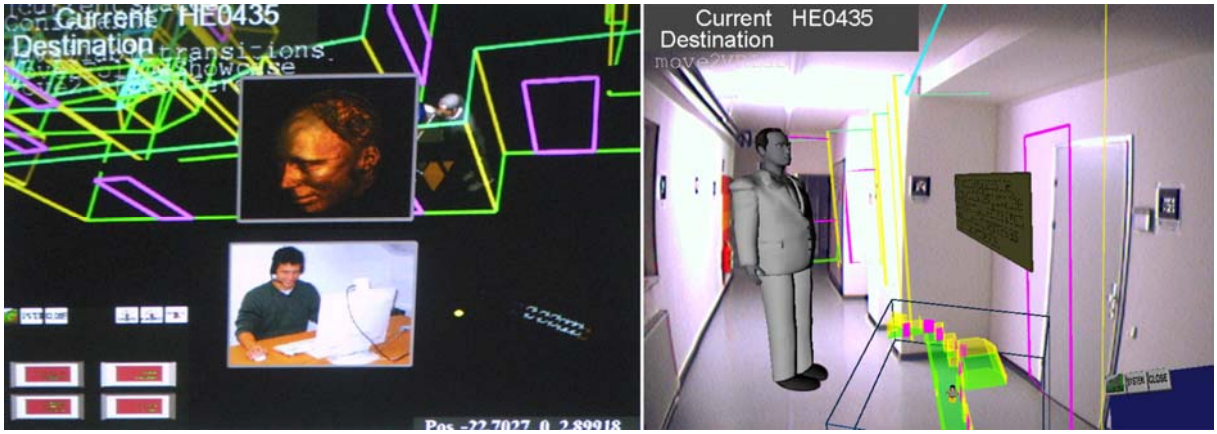

Figure 7 - a) The indoor tour guide application running on the desktop developer setup. b) The indoor tour guide application view captured from the HMD of a user wearing the mobile AR backpack system. In both setups a world-in-miniature view of the building mode.

2.3.3 Alice

Among the tools targeted towards beginners, the Alice system [21] is particularly noteworthy. It was designed as a tool to introduce novice programmers to 3D graphics programming. Alice comes with its own scene editor and an extensive set of scripting commands, but is clearly targeted at an educational setting. For creating “real world” applications, the reusability and modularity of Alice is insufficient. Also, Alice focuses on animation and behavior control of individual objects and does not offer any high-level concepts for application control.

2.3.4 VRSS

The Virtual Reality Slide Show system (VRSS) [22] provides a set of high-level concepts for authoring through a collection of Python macros. VRSS draws inspiration from conventional slide shows, and provides the concepts necessary to the user when creating similar slide shows within a VR environment.

2.3.5 Geist

The Geist project [23] aims at the presentation of historical and cultural information for mobile AR users roaming a city. The Geist engine builds on a detailed analysis of drama theory and interactive storytelling and provides several runtime modules to support applications based on these concepts. Using Prolog, authors can create semiotic functions that drive the story. Virtual characters that are controlled by an expert system demonstrate compelling conversational and emotional behavior. While this approach is very general and powerful, it can only reveal its full potential in fairly complex applications, incorporating dynamic behavior of multiple real and virtual actors, and hence requires a correspondingly high effort in content creation. The authors do not provide details of their application examples, making it difficult to assess the final results.

2.3.6 alVRed

There are few systems that support authoring (as opposed to programming) for general purpose AR/VR. An initial inspiration for the work presented in this paper is alVRed [17], developed at Fraunhofer IMK. The alVRed project is an authoring solution which uses a hierarchical state machine to model the temporal structure of VR applications. In their model, a state represents a scene of the application, while the transitions between states represent changes in the application triggered by user interaction. alVRed provides a runtime engine built on top of the Avango [32] environment for the executing the state machines, as well as a number of editors for supporting various stages of content creation. Particularly interesting is an editor for fine-tuning graphics and animation parameters from within an immersive projection environment. The one area not adequately addressed by alVRed encompasses key AR requirements such as interaction abstraction and multi-user operation.

2.3.7 DART

The Designers Augmented Reality Toolkit (DART) [25] developed at GeorgiaTech is built on commercial software: DART extends Macromedia Director, the premier authoring tool for creating classical screen based multimedia applications. DART allows design students who are already familiar with Director to quickly create compelling AR applications, often using sketches and video based content rather than 3D models as a starting point. Director is an extremely versatile platform used by an extensive community of multimedia developers for a large variety of applications, and these properties are inherited by the DART plug-in. However, DART is ultimately limited by the technical constraints of Director, such as inadequate support for 3D models, stereoscopic rendering, optical see-through displays or multi-user applications.

2.3.8 DWARF

Finally, the Distributed Wearable Augmented Reality Framework (DWARF), developed at Technische Universität München, deserves mentioning, although it is not strictly an authoring solution. DWARF is a strongly component-oriented middleware, composed of communicating objects. In [26], the authors report on interactive development in “jam sessions”. This expression describes incremental prototyping of a running system by multiple programmers working concurrently. A graphical monitor program allows convenient inspection and remote control of the components. DWARF’s dynamic reconfiguration capabilities allow the developers to replace software components and restart parts of an application on the fly without re-starting the whole system. This is exceptional insofar as it pertains to a distributed multi-user system with full hardware abstraction capabilities rather than a single computer

authoring workplace.

2.3.9 AMIRE

The AMIRE framework [4] is component-oriented and designed for creating MR applications and authoring tools. AMIRE components are configurable and communicate via in- and out-slots, comparable to the signal/slot mechanism of the graphics library Qt.

AMIRE allows for the development of various authoring tools adapted to the requirements in a certain domain. This way, authors with different background knowledge and technical skills can be provided with appropriate means to create MR applications. Authoring tools developed with the AMIRE framework are an Authoring Wizard for furniture assembly [16], an Oil Refinery demonstrator and an authoring application for augmenting museum exhibitions [2]. Typically, the result of the authoring process is a XML based application description that is interpreted at runtime. As the same components are used during the authoring process and at runtime, AMIRE allows for desktop based authoring as well as authoring within the MR environment. Typically, the authoring process includes the selection of MR components, the adaptation of the selected components, the specification of the connections between the components and the calibration within the MR environment.

Similar to the Graphics Gems collection that includes useful algorithms for graphics programming MR Gems have been developed within AMIRE that provide solutions to common programming problems in MR applications such as pattern-based object recognition, and path animation among others [4].

2.3.10 MARS

The MARS (Mobile Augmented Reality System) Authoring Tool [6] uses a timeline to arrange virtual objects and other media objects such as audio, video, images and text. The authoring tool runs on a desktop computer. The author selects media objects and arranges them temporally using the timeline and spatially using a 3D model of the real environment. The virtual content is interconnected with hyperlinks. The author may preview the MR application in a VR mode on the desktop. The content description is stored in a XML-file which is interpreted by the MARS AR presentation tool. The general layout and the timeline-oriented authoring approach are comparable to the multimedia authoring software Macromedia Director.

2.3.11 PowerSpace

A straight-forward approach to MR authoring is the authoring system PowerSpace [5] which utilizes the commercial presentation program Microsoft PowerPoint to arrange MR content. PowerSpace focuses on technical documentation and service manuals. The author arranges 2D objects on PowerPoint slides. The slides are stored in an XML-based file format which can be interpreted by the PowerSpace editor. The PowerSpace editor allows the spatial arrangement of the 2D objects and the addition of 3D objects. The order of appearance that has been defined within PowerPoint (slide order and appearance order within a single slide) may be changed within the PowerSpace editor. The results of the authoring process are stored in the same XML file and may be viewed with the PowerSpace viewer.

2.4 Ambient, Ubiquitous and Tangible Interfaces

Quite a number of frameworks for ambient, ubiquitous and tangible interfaces have been developed, but rather than describing the frameworks – which are mostly too specific and not general solutions – design concepts will be described here. These design concepts are general guidelines for designing those interfaces.

As ambient, ubiquitous and tangible interfaces are a combination of hard- and software it is rather difficult to find established “ready-to-use” frameworks. Most research groups do not have the capacity to manufacture hardware in large quantities (and therefore at a reasonable price). On the other hand - software frameworks are tightly connected to the hardware used,

therefore general solutions are hard to find (or very expensive). Up to now the industry did not start to produce “general” ubiquitous and tangible hardware – as it is true e.g. for AR hardware (probably because the military is not interested in tangible computing, but in AR).

Another solution is to use already available frameworks – like Atelier, Morgan or Opentracker to abstract the interfaces for applications – actually most ambient, ubiquitous and tangible interface frameworks utilize frameworks already available.

Some examples that have been published in the past will be described now, to provide an overview of existing concepts.

2.4.1 Tangible Bits

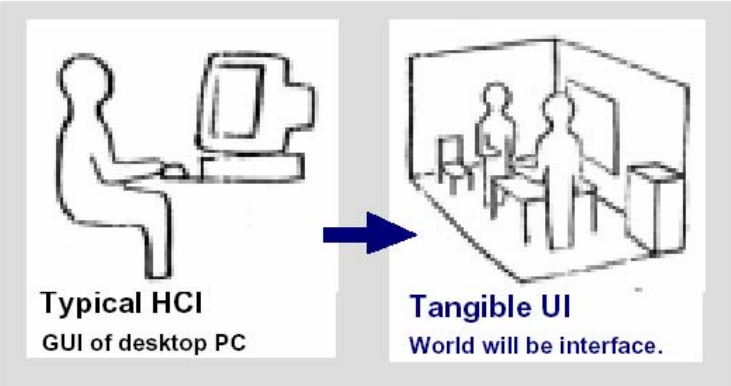

Figure 8 From GUI to TUI [41]

Ishii et al.describe in [41] “Tangible Bits” as their vision of Human Computer Interaction:

> *"Tangible Bits" is an attempt to bridge the gap between cyberspace and the physical environment by making digital information (bits) tangible. We are developing ways to make bits accessible through the physical environment.*
>
> *Key concepts are:*
>
> *1) Interactive Surfaces: Transformation of each surface within architectural space (e.g., walls, desktops, ceilings, doors, windows) into an active interface between the physical and virtual worlds;*
>
> *2) Coupling of Bits and Atoms: Seamless coupling of everyday graspable objects (e.g., cards, books, models) with the digital information that pertains to them; and*
>
> *3) Ambient Media: Use of ambient media such as sound, light, airflow, and water movement for background interfaces with cyberspace at the periphery of human perception.* [41]

As illustrated in Figure 8, the transition of a Graphical User Interface to a Tangible User Interface changes the world itself into an interface.

The mediaBlocks project [42] presented the use of physical markers as handles to digital media and a number of corresponding appliances. The project was influenced by the metaDESK/Tangible Geospace prototype [41][43] 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 [44]. 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 [41]. The trans-BOARD used paper cards called hypercards as physical carriers for live and recorded whiteboard sessions. The hypercard interaction was also 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 [45] speaks to moving computation and networking of 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 [46]. The often cited Bishops Marble AnsweringMachine [47] 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 [48]) 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 [49]. The manipulatives work makes strong progress towards developing objects with rich digital/physical couplings.

2.5 Data and event distribution

One of the key issues for all distributed applications is the way how data and events are distributed between remote processes. Since MR applications usually involve a number of processes running on different hosts, due to the multi-user capability as well as the different components such as trackers which typically do not reside on the user's hosts. The usability of a distributed MR application does strongly depend on the way how data is exchanged between the different processes. Inefficient message passing reduces the acceptance of the application by the users, since even slightly longer latencies might destroy the presence of the users.

Before discussing communication middleware, low level message passing protocols and different network architectures are introduced. Finally, some high-level AR frameworks are described.

2.5.1 Network architectures

Network applications usually use one of the two network layer protocols, TCP and UDP, both sitting on top of the transport layer protocol IP. The difference between these two network layer protocols is the type of connection. TCP/IP is a connection-oriented reliable transfer protocol and UDP/IP is not. TCP/IP ensures that no transferred data is corrupted or lost and all data will arrive in order. In addition, at the beginning of a TCP/IP connection between two hosts a *hand-shake* is performed, where both establish parameters for the following data transfer. UDP/IP does not guarantee anything, neither that received data is not corrupted nor that data will arrive in order or at all. UDP/IP is connection-less, i.e. no handshake between the two hosts is performed [8].

By ensuring reliable data transfer, TCP/IP induces longer latencies and slower data transfer than UDP/IP. In addition, TCP/IP utilizes congestion control, which stops sending packages to the remote host if the connection is congested.

Another transport layer protocol is the multicast protocol, which enables a host to send and receive data to and from a group of hosts without having a direct connection to them. The multicast protocol is unreliable as well, but since it is able to support bulk data transfer to a group of receivers while limiting required bandwidth, a lot of effort is done to implement reliable multicast protocol. No standard has been proposed yet, due to the great challenges

of this problem, also reliable multicast is widely used in frameworks for distributed virtual environments.

Client/Server

Network applications typically have two parts, the client and the server part. The server *serves* the client and usually both parts reside on different computers (hosts). If a distributed virtual environment uses this network architecture, the user's system provides the client which communicates with the server. Without loss of generality, we assume that the server is a stand-alone system not hosted by a user of the distributed virtual environment. A user connects from the client host to the server host to enter the distributed virtual environment.

Usually the server does not forward all messages to every user. To keep network traffic low, only messages are forwarded which are *useful* for the user.

The advantage of this network architecture is the simplicity of the centralized approach. Users need only one connection between their host and the server. The disadvantage of the Client/Server architecture is that they do not scale. While the number of users increases, the server becomes more and more the bottleneck of the system, since the amount of work increases quadratically for the server.

Peer-to-Peer

In a peer-to-peer network architecture each user's host is client and server at the same time. A user's host has to establish connections with all clients of the distributed system. This is done by connecting to all hosts of the system. Each user action is sent directly to all other users. Filtering messages for specific users to lower network traffic also applies, but has to be done by each client.

The advantage of peer-to-peer architectures is that it is decentralized and extremely failure tolerant. If one host crashes or suffers from network congestion, no other host suffers from this defect. The disadvantage is that each host has to establish connections to every host of the community. Each host has to provide server functionalities, like persistency and consistency control.

Multicast group

Multicast networks allow arbitrarily sized groups to communicate on a network via a single connection. Each user message is sent to the multicast group once and is distributed to all users of the group. The sender can specify how far a message is sent by using the time-to-life (ttl) field, which is decremented when it passes through a router. A ttl value of 16 will address all members of the group within the same local site. Distributed virtual environments utilizing this architecture are usually combined with a Client/Server architecture for the initialization phase. The user connects the server to receive the current world's scene information and information about the multicast group addresses used by the environment. After the initialization phase the user joins the multicast group to receive all update messages. Consistency control has to be performed by each host of the group.

Filtering messages for specific users to lower network traffic cannot be supported, since no direct connection to other users is used.

The advantage of multicast architectures is that they are scalable and the network traffic is reduced to a minimum. The disadvantage is that the underlying protocol is unreliable.

2.5.2 Communication middleware

There are many communication middleware, which work above the network protocols and provide more control over the network communication and abstract sockets and data packages. Some of them offer remote procedure calls, like XML RPC and CORBA.

XML RPC

XML RPC is a specification which uses XML and HTTP as the transport protocol to provide remote procedure calls for applications. Since the specification is very simple and only defines a small set of data types, is this very light-weight in comparison to other RPC systems. The specification is platform independent, because XML and HTTP is available on nearly every platform. Implementations for a lot of platforms do exist.

CORBA

CORBA [54] (Common Object Request Broker Architecture) is more than just a RPC system. It is an object-oriented middleware which supports the creation of heterogeneous distributed systems by providing APIs, communication protocols and services. CORBA is platform and programming language independent and applications. Since CORBA provides access to remote objects, it is completely location transparent.

IDL (Interface Description Language) is used to specify the interfaces of distributed objects. These files are translated to skeleton and stub declarations in a particular programming language.

DCOM

DCOM (Distributed Component Object Model) allows COM objects to communicate across network boundaries. Since both COM and DCOM are Windows specific specifications they are not platform independent.

2.5.3 High-level MR frameworks

Studierstube

Studierstube [29] extends OpenInventor [51], a scene-graph rendering library, and uses OpenTracker [28], a modular dataflow middleware for pose tracking. Both, Studierstube and OpenTracker make use of ACE [50] for network communication abstraction. Studierstube allows application development in C++ or, more rapidly, via OpenInventor scripts. Alternatively, application developers can choose to use APRIL [9], a high-level descriptive language for authoring story-driven AR presentations.

The tracking data and events in Studierstube originate either from its scene-graph or from OpenTracker which serves as easily reconfigurable middleware for interfacing a very broad range of hardware devices and as an abstraction layer for other tracking frameworks. Studierstube supports two different mechanisms for data and event distribution. The acyclic graph traversal builds upon Inventor's event system which uses the "visitor pattern" to distribute events to potentially interested nodes in the scene graph. A visitor traverses the scene graph and calls a previously specified method on each node it encounters, which contains its specific event handling behaviour. This mechanism is important in cases when nodes need to know something about their spatial relations to other nodes (typically along a path), e.g. widgets. The drawback of this mechanism is the temporal responsiveness when traversing the scene-graph. This is suboptimal for near real time tracking which is essential for MR. To account for the drawbacks of this mechanism a second method for the distribution of data and events is available. The generic event bus works with a publish and subscribe mechanism which delivers data and events directly to the subscribers and makes the whole system very responsive. Similarities of the two distribution methods are the same source of events, same event data (reuse event representation) and the use of the same meta-data which is the description of information about events or event sources. The two different subscription and transport mechanisms profit from each other creating a win-win situation in terms of a contemporary MR-framework.

Morgan AR/VR Framework

Morgan [11] is a component-based framework that relies on the CORBA middleware for network communication. It currently supports many devices, including mouse and keyboard as well as haptic input devices, object tracking systems and speech recognition libraries. Thus, multi-modal user interfaces can be rapidly developed and evaluated. Additionally,

Morgan provides a distributed render engine with automatic scene graph synchronization capabilities. All components are accessible from remote computers.

A broad range of free or commercially available devices and systems have already been integrated into the framework, including Object tracking systems from Intersense (IS 600, IS 900, IS 1200, InertiaCube 2, InertiaCube 3), ARToolkit [7], IBM ViaVoice, NMEA and Garmin GPS receiver, Stereo HMDs.

All components of the framework can easily be located within the distributed system by a centralized service managing the components. This service is also responsible for remote instantiation of components. The central component of the framework is a render engine that provides high performance rendering capability for the application. Beyond standard functionalities like collision detection, picking and real-time CSG, the render engine has some unique features. The internal scene graph used to store the objects of the scene is designed for efficient rendering, holding only information needed for rendering. All other data of 3D graphics formats, e.g., semantic information, is kept in external scene graphs that provide the mapping of the data onto the internal scene graph. This data can later be mapped back into the source format without loss of semantic information. Initially, the ISO standard VRML'97 is supported by this mechanism.

Distributed multi-user applications are supported by the built-in functionality of the scene graph to automatically synchronize itself with all other scene graphs within the distributed system. An abstraction layer for the frame buffer, e.g., OpenGL or Direct3D, simplifies support for additional frame buffers such as the upcoming Direct3D Mobile.

A viewer realized on top of the framework provides 3D render output for the render engine. Besides keyboard and mouse and headtracking navigation and interaction capabilities, it offers different output modes, e.g., mono, quad buffered stereo and dual head stereo. They can all be used full screen or in a window, making it possible to display the result on a wide variety of displays, e.g., desktop monitors, stereo projectors, stereo headmounted displays and virtual glasses. Non-see-through displays, like VR glasses, can be used for AR environments by augmenting a video background provided by a USB or Firewire camera.

Atelier

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 components, 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 internet 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 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.

The Atelier infrastructure is under active development. Current development initiatives are the additional JXTA protocol module to support peer-to-peer networking in addition to the TCP/IP connected sockets, and a context service to enable social contextual information to be managed centrally for many simultaneous users.

Equip Component Toolkit

The Equip Component Toolkit (ECT) is a software toolkit for ubiquitous computing for rapidly prototyping and realizing ubicomp installations, applications and environment [56]. Additionally, it tries to increase the potential involvement of designers and users. EQUIP is part of the EQUATOR project.

The toolkit supports loosely coupled distributed applications running over multiple hosts by creating, configuring and interconnecting of software components and component that represent physical devices and sensors. Data between components is mainly distributed through the coordination data-space, a tuple-space approach allowing tuple producers and consumers to be decoupled.

2.6 Summary of the State-of-the-art Report

This section gives an overview of all research areas related to Cross-Reality Interaction and Authoring Tools. Since part of these techniques is not relevant for the project itself, we summarize the technologies which are interesting for our goals, and will be further developed according to the needs of the showcases. The following table lists the core technologies which are interesting for the project partners.

Device Abstraction

Technology	Relation to WP	Relevance (1 low, 5 high)
OpenTracker	OpenTracker – developed at TUG – will be used and extended within this project (see 4.5).	5
OpenVideo	OpenVideo – developed at TUG – will be used and extended within this project (see 4.6).	5
VRPN	VRPN is a technology relevant for OpenTracker and DEVAL as related work.	3
DirectShow/DirectInput	Both technologies are used by DEVAL for accessing devices (see 4.4).	5

Device-independent user interfaces

Technology	Relation to WP	Relevance (1 low, 5 high)
UIML	UIML is a standard closely related to MRIML.	4
MRIML	MRIML – developed at FIT – will be applied to the Morgan framework, used and extended within the project.	5
Exchangeability of 3D interface components	The conclusions of this work have a great impact on MRIML and DEVAL.	5

PUC	This work could have an impact on MRIML. Looking at this project will help give insights for MRIML.	2
Atelier	Atelier – developed at UOulu – will be used and extended within the project.	5
CAPNET	CAPNET might become relevant for WP 4, but currently no integration is planned.	2

Interaction Prototyping/Authoring

Technology	Relation to WP	Relevance (1 low, 5 high)
Behaviors	Behaviors – developed at FIT – will be used and extended within the project (see 4.2).	5
APRIL Language	Minor relevance for WP 4, since it is currently not further developed at TUG and the focus lies on interactive storytelling.	1
Alice	Minor relevance for WP 4 due to the insufficient reusability and modularity.	1
VRSS	Minor relevance for WP 4 due to the focus on slide shows.	1
Geist	Geist is intented for interactive storytelling and therefore only of minor relevance for WP 4.	1
alVRed	Although quite interesting, the lack of recent development and support, makes it irrelevant for WP 4.	1
DART	Due to the limitation to use Macromedia Director, DART has no relevance for WP 4.	1
DWARF	Minor relevance from authoring perspective, since we chose to use other application frameworks, Studierstube, Morgan, Atelier.	1
AMIRE	The authoring functionalities are very interesting for WP 4 and provide a good reference for development in this workpackage.	3
MARS	Due to the limitation to use Macromedia Director, DART has no relevance for WP 4.	1
PowerSpace	The authoring functionalities are very interesting for WP 4 and provide a good reference for development in this workpackage, but the focus of this authoring system is too different for IPCity.	2

Ambient, Ubiquitous and Tangible Interfaces

Technology	Relation to WP	Relevance (1 low, 5 high)
Tangible Bits	Tangible User Interfaces have a high relevance for the whole workpackage, especially for the ColorTable (see 4.7)	5

Data and event distribution

Technology	Relation to WP	Relevance (1 low, 5 high)
Studierstube	Studierstube – developed at TUG – will be used and extended within the project and is one of the main technologies.	5
Morgan AR/VR Framework	Morgan – developed at FIT – will be used and extended within the project and is one of the main technologies.	5
Atelier	Atelier – developed at UOulu – will be used and extended within the project and is one of the main technologies.	5
Equip Component Toolkit	Equip is an interesting toolkit with partial comparable functionality, but only with minor relevance for the project.	2

3 Requirement Analysis

Based on a questions catalogue, which has been used to query the different showcases about their requirements on the basic interaction tools, we have conducted a requirement analysis for the first set of tools, which will be developed during the first phase of the IPCity project. The questions catalogue has been put together in cooperation with the Mixed Reality Infrastructure workpackage, and it was very helpful for the analysis since it produced comparable results and helped the showcases to formulate the requirements without exactly knowing their initial scenarios.

The questions catalogue has five sections focusing on different aspect of the application scenarios of the showcases. While some questions focus on general requirements such as operating systems and end-user devices, other questions are directly focusing on the different research issues addressed by the research workpackages WP4 and WP5.

The first requirement for all tools and services developed within this workpackage is already stated in Annex I: “The general success criteria for all research workpackages applies: all tools and services developed have to be required by at least two showcases and must actually be used by at least one of them. Additionally, all tools and services must be flexible enough to be used in other showcases and even in other projects.”

3.1 Initial observations

One of the key initial observations from the requirement analysis is not a very surprising one. Due to different reasons, the showcases will implement their application scenarios on different programming languages and will used different operating system. While the Time Warp showcase will mostly restrict their development using the C/C++ programming language and target at different Windows platforms, such as Windows XP and Windows Mobile 5, the CityTales showcase will also include JAVA application and the portable game consoles Playstation Portable in addition to Symbian smart phones. Similar, Large Scale Events and Urban Renewal are also targeting on a wider range of devices, operating systems and programming languages.

Additionally, all showcases will not be able to rely on a specific hardware setup, due to changing environment constraints. This requires easy configuration mechanisms and tools, which allow for simple exchange of many interaction components, such as input modalities.

Since the showcase developers will use existing frameworks and tools during the first phase, and naturally utilize those where they already have a very good knowledge of, it is very likely that the showcase will rely on different frameworks, network protocols and network architectures. One basic requirement derived from this situation is that the interaction tools need to be used across different frameworks and network protocols. Also, the amount of work required to integrate a tool developed within the research workpackages has to be as minimal as possible to ensure that integration of the tools is ensured across the different showcases.

The main technology aim of the IPCity project is to move high-quality MR a step further from the labs to real settings; following this aim all showcases are targeting mobile users with a wide range of different mobile devices, e.g. smart phones, PDAs and Wearable MR systems, and also operate with nearly any available communication technology, e.g. SMS/MMS, WiFi, Bluetooth and 3G. Consequently, each showcase is faced with users which cannot be connected to application at any time and even suffer from lack of communication for quite a while. In order to ensure the different presence aspects for the users, the interaction tools and in particular the underlying frameworks have to deal with spontaneously connections and disconnections of the users.

Last but not least scalability is an important issue (as for all distributed systems). Each showcase is targeting a large user group and even in the first phase multiple users will simultaneously participate in the application scenarios. Also collaboration among the users is

not an initial issue for the Time Warp and the CityTales showcases, it is definitely planned for proceeding project phases. This requires handling scalability especially within the underlying tools and frameworks, which have to support such application settings, i.e. every interaction tool or component has to be designed and developed to deal with scalability issues.

One problem of this requirement analysis is that the showcases are in a very early stage of the scenario planning, therefore some of the requirements they might have cannot be analyzed at this time.

3.2 Device abstraction

Although the state-of-the-art report for the device abstraction lists a lot of technology, which is already available in this area, the work in this workpackage has to go a lot further than that. This is basically due to the missing unified approach, which does not focus on certain aspects or a special class of devices, such as trackers. The replies from the showcases support our initial expectation that device abstraction will be one of the key research issues for this workpackage. If we want to support the exchangeability and the substitution of different device not necessarily part of the same device class we have to come up with an overall solution including all major devices classes. In addition to the possibility to exchange devices it is also very important that devices can be combined to form a new device, e.g. creating a 6 degrees-of-freedom tracker by combining a GPS tracker with an inertial tracker. It is not surprising that all of the showcases stated an interest for such functionality especially due to the changing hardware and environment setups of the showcase scenarios.

The results of the questions catalogue stated the necessity to exchange different tracking devices and input modalities such gestures and speech commands. While the application scenarios of the showcase will further evolve and will become more and more concrete we expect them to require exchanging also other input and output devices similarly.

The device abstraction is not only about exchangeability and combination of devices, but also about easy access to them. Since the showcases include end-user devices with very distinct capabilities in regards to processing power, memory and bandwidth, the device abstraction must also allow the accessing input data at different frequencies and allow specifying the desired information fine grained.

3.3 Device Independent User Interfaces

Similar to the requirements for the device abstraction and closely related are the requirements which came out for device independent user interfaces. But of course the research in this area has to go beyond device abstraction. The main research questions that have to be address here are how is it possible to define user interfaces in a device independent way and how can we exchange input modalities without changing the user interface description. The existing approaches listed in the state-of-the-art report can only serve as a starting point and the goal for this workpackage should be to take the current approaches further.

The requirements for the user interface description language, which has to be developed in this workpackage, result from the same constraints as for the device abstraction. Each showcase will target at different output devices and will use a large set of interaction mechanisms. That means we have to decouple the description and the implementation of user interfaces from the execution settings such as input modalities and hardware platform.

Because the user interfaces for the individual showcases are not completely defined yet and are likely to change during the time of the project, we have to closely work with the showcase developers together to receive constant feedback on their changing requirements. Apart from the overall requirements such as platform and programming language independency, scalability and the easy integration with other tools like the device abstraction and the interface prototyping and authoring tools, further requirements are difficult to extract at this time, since the progress in the showcase is not far enough.

3.4 Interface Prototyping/Authoring

The current available technologies from the project partners like Behaviors the rapid prototyping of interfaces, the APRIL language for authoring of applications in respect to hardware setup and content. Both will be probably used by the showcase for their development during the first phase of the project to realize the first demonstrators, since all showcases have stated an interest in developing new interfaces and interaction mechanism. In order to support this effort we have to develop tools which help this process and target the research questions how can we better support the easy creation of new interaction mechanisms than with the currently available tools and what are the major building blocks for MR interactions.

3.5 Ambient, Ubiquitous and Tangible Interfaces

Ambient, Ubiquitous and Tangible interfaces were stated as necessary and important for each of the showcases. Due to the unclear development within the showcases at this time, this workpackage will not start the development of any interface or a support tool for ambient, ubiquitous and tangible interfaces. The current state-of-the-art is sufficient for the showcases to start their development. The only work which will be done in this area during the first year will need a clear pull from the showcase, but this cannot be decide before the scenarios of the showcase are planned in more detail.

For the ongoing project phases we will support the showcases by tools which help them to develop ambient, ubiquitous and tangible interfaces and also to realize them on different platforms.

3.6 Data and Event Distribution

The main building block to ensure integration among the different tools and services developed in the workpackages WP 4 and WP 5 is the data and event distribution services. The key requirements are obviously platform and programming language independency as well as a scalable approach in order to be feasible. Since the goal is to develop a service that allows every process on each hardware platform to exchange data and events regardless of a specific framework which is used, this service has to be powerful and light-weight at the same time. It has to be powerful, because it has to support a number of transport protocols starting with TCP/IP and UDP and has to provide message forwarding to work as a bridge between different systems. Additionally, such as service must provide functionality like grouping and filtering of events as well as conversion of data types, e.g. J2ME smartphones do not support all data types like doubles or unsigned integer naturally. In case this service would be used in a client/server architecture, where each clients sends the data to this service, which would forward them to the interested clients, persistence and logging can easily being integrated.

One to other hand there must be a light-weight solution in order to allow all end-user devices to send and receive data and events using this service, without the requirement to implement a full specification. Especially, small memory and small processing power devices like PDAs and smart phones require a minimal implementation that serves their needs but not more. So one of the requirements has to be that unknown data and events can be ignored by each system, also the requirements for event produces, such as GPS trackers connected to a smart phone should be as minimal as possible. While a GPS tracker component on a laptop might send all available information like number of satellites, the dilution of the precision and the course over ground, a GPS tracker component on smart phone only sends information about the longitude and latitude.

This can be achieved by defining different profiles of this service to ensure the seamless communication between the different profile settings.

Other issues, which have to be addressed by this kind of distribution service, are the extensibility, i.e. arbitrary data and new events can be send without changing the service, and it should induce only a very small additional latency.

4 Year 1 Demonstrators

During the first year, different tools have been developed and tested in order to see if these are suitable for the showcases. Based on the initial requirement analysis which is described above, a set of demonstrators have been delivered to the showcases, some of them have already been used for tech probes of the showcases, others have raised a high interest by the showcases to be used during the second phase.

4.1 Overview

In this section the demonstrator that have been developed during the first year are described. Based on the state-of-the-art report and the requirement analysis we decided after consulting the showcases the following list of demonstrators to be the first set of basic tools. Three of the demonstrators are focusing on the research issue of device independent cross-platform access to different input, output and streaming devices – each with a different focus. While OpenTracker focuses mainly on tracking devices and has now been extended to include multimodal input, OpenVideo is targeted at video cameras and provides the data as video streams. DEVAL on the other hand approaches this issue in a more general way and tries to include all devices, input, output and streaming devices, into one hierarchy and make them accessible by abstracting from specialized functionality.

The Interaction Prototyping Tool – initially requested and utilized by the Timewarp showcase – allows authoring and prototyping of interactions for simple and quick evaluation of multi-modal user interfaces.

AuthOr is a first prototype of an authoring, orchestration and evaluation tool, which is suitable for all stages of a showcase event and will be part of at least one showcase, but all showcases have stated an interest in evaluating it for later use.

The ColorTable fosters collaborative interactions in urban design scenarios by providing tangible interaction possibilities and is used in the showcase Urban Renewal. Colored tangible user interfaces on top of the table allow for modelling tasks in a very unintrusive and collaborative way.

Two demonstrators are targeted at smartphone devices, the MMS Media Extractor extracts media from incoming MMS and forwards it via Bluetooth and the Extendable DataMatrix reader scans 2D barcodes and executes network resources as defined by the barcode.

The research issue of data and event distribution has been started by initial conceptual work of the workpackage partners in order to allow inter-framework communication of data and events.

4.2 Interaction Prototyping Tool

4.2.1 Description

This demonstrator represents the first version of the interaction prototyping tool. It is realized using a component-based approach, which allows for modeling interaction techniques and object behaviors from a set of basic components.

In contrast to 2D desktop environments, no standards have yet been established for 3D user interfaces in Mixed Reality environments. Due to the variety of input and output devices and different interaction techniques, the realization of a particular user interface typically is quite difficult and time consuming. On the other hand, the user interfaces created require intensive user tests followed by redefinitions and adaptations or even complete reimplementations. This typically is not feasible if realizing the user interface techniques by programming.

The approach here fundamentally changes this by providing an easy to use mechanism for defining and modifying interaction techniques and object behaviors, adapting them to the needs of the users on the fly.

Developers of interaction techniques typically will model the corresponding interaction prototypes (Behavior objects) using a text-based description (see Figure 9 for an example). Beside a simple text-based description format, an XML-based scheme is currently evaluated (see Figure 10). Interaction and communication between the individual Behavior objects and the Mixed Reality environment is realized by events (only).

```
Behavior
{
        targets [ XSG:X3D::*/Text ]
        attach  [ LOAD_FILE ]

        Sensor
        {
                targets "DEVICE:Keyboard::Key"
                targetConditions [ key != '' ]
                result Key { }               # standard key event

                CONNECT result.Key.character INPUT_EVALUATOR.charInput
                SIGNAL fire EVALUATOR_KEY.execute
        }

        Evaluator INPUT_EVALUATOR
        {
                Char charInput ''
                String text ""

                statements [   text += charInput;
                               text -= 1; ]

                CONNECT text Action.output.text
                SIGNAL finished Action.execute
        }

        Action
        {
                output Event          # user defined event
                {
                        String 1D text "Text"
                }
                recipients "." # local node
        }
}
```

Figure 9 - Interaction prototyping example (simple text file format)

Each Behavior object typically will consist of a number of components. So far a set of approximately 20 basic components has been defined, while six of them are currently supported by the initial demonstrator. The components allow to register for certain events or services, to query scene graph and system status data, to react on user input, to set and modify system and scene objects, and to execute time-dependent behaviors. Execution of individual components and data transfer between them is realized by a signal/slot mechanism.

```
<Behavior targets="[ XSG:X3D::*/Text ]" attach="LOAD_FILE">
        <Sensor targets="DEVICE:Keyboard::Key::" result="Key"
                targetConditions="[ Key.key != '' ]">
                <CONNECT from="result.Key.character"
                        to="INPUT_EVALUATOR.charInput"/>
                <SIGNAL from="fire" to="EVALUATOR_KEY.execute"/>
        </Sensor>
        …
</Behavior>
```

Figure 10 - Interaction prototyping example (XML file format) - fragment

The approach is based on previous work at FIT, where VRML'97 was extended by a similar mechanism (see Section 2.3.1). However, the current demonstrator expands those concepts

to a far more general and flexible approach, allowing us to use this mechanism for various aspects:

- Scene-graph/-object-related interaction and animation
- Application-specific (scene-graph-independent) interaction techniques
- Combination, modification, and simulation of input and output devices

While for the first two areas the Behavior definitions typically are loaded by or as an application, manipulating scene graph objects, the third case allows us combining multiple devices in a flexible way. The Behaviors engine directly maps events from input devices and to output devices to appropriate devices in DEVAL (see Section 4.4). Figure 11 provides a simple example of a user interface for setting two color values for color interpolation of an animated object. Here all user interface elements (input, modification of color values and color bars) and the animation were realized using the above mechanism. The input in this example is published by a *Button* publisher.

Figure 11 - Simple example of a user interface created by the interaction prototyping mechanism

The current demonstrator has been realized within the Morgan AR/VR framework. In the next project phase it will be evaluated within showcases and further extended to support more components as well as component and Behavior object prototyping. Additionally a graphical editor is planned to be added.

4.2.2 Specification

Hardware and OS	PC/laptop, Windows XP
Software	• Morgan Framework • DEVAL (see 4.4)
Core Features	• Modeling interaction techniques and object behaviors • Object-oriented component-based approach • Modification at run-time
Status	Early demonstrator
Intended users	Mixed Reality user interface and application developers
Showcases	WP 8, WP 9
Relevance beyond project	General mechanism applicable to all types of Mixed Reality and Virtual Reality applications

4.2.3 Testing / Evaluation

Due to the state of the demonstrator, no testing or evaluation has been performed.

4.3 AuthOr

4.3.1 Description

This demonstrator describes a tool that will be used be the showcase to augment arbitrary maps with 2D overlays, e.g. lines, images, text. Such overlays are localized by a GPS position and can represent anything like a player position, a path, an event or a virtual item. Since the maps have localization information as well the overlays can be shown on top of the map at their exact location. Currently AuthOr interfaces with Google Earth (see Figure 12) and Google Maps (see Figure 13) in order to download the appropriate tiles on the fly. Other maps, e.g. images or satellite pictures from the NASA, will be supported during the next iteration. Additionally, a hybrid mode is already supported, where an alpha map is created from Goople Maps tiles in order to overlay only the streets onto another map, e.g. Google Earth (see Figure 14).

AuthOr is written as a Qt widget and therefore it will be very easy to integrate it into any authoring, orchestration and/or evaluation application.

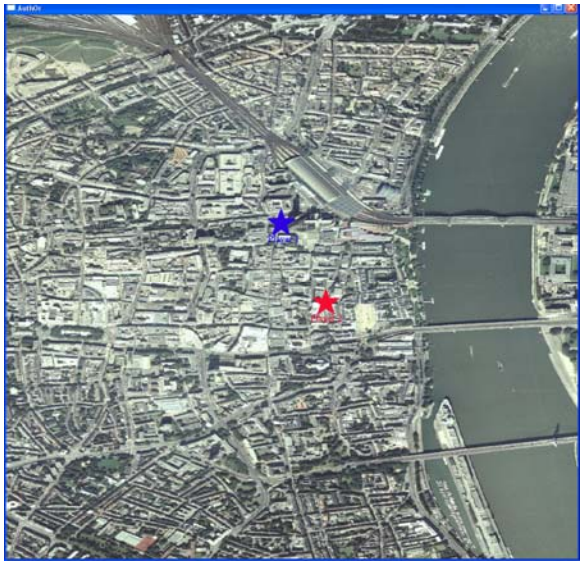

Figure 12 - Augmented satellite map of Cologne city centre

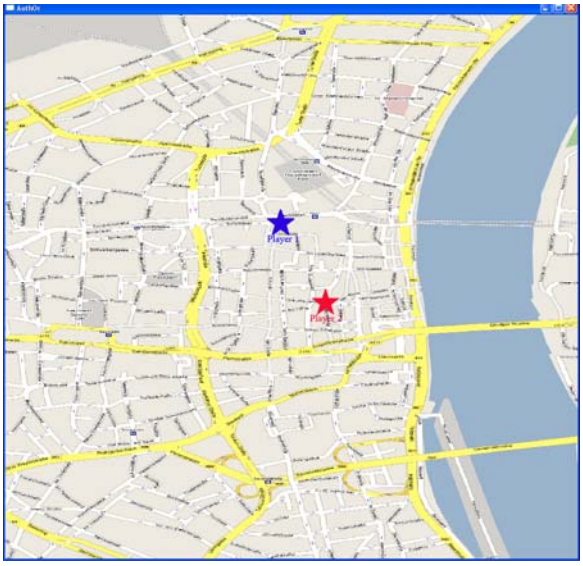

Figure 13 - The same map region as on the left, except using normal maps

The interface of the widget provides methods to set the borders of the widget in GPS positions and therefore calculating the optimal zoom level. Additionally, the center of the map and a zoom level can be specified, from which the borders are calculated.

The user is able to navigate with the mouse:

- Left button click: Set new center of map.
- Left button drag: Drag the starting location to the final location. This results in a new center for the map.
- Mouse wheel: Zoom-in and zoom-out of the map.
- Middle button click: Switch between Google Earth and Google Maps.

The navigation can also be handled through the application by the provided interface methods, e.g. if AuthOr should track a player and always center the map on her position.

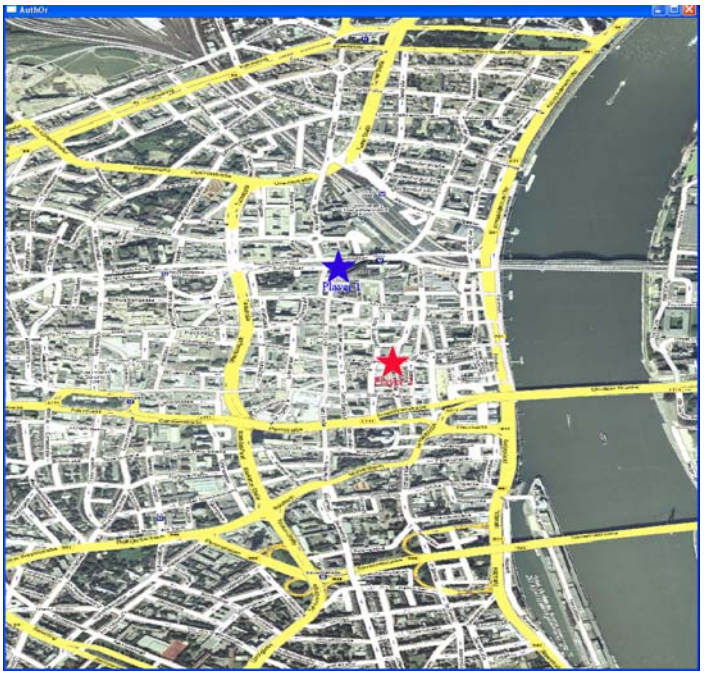

Figure 14 - Same map region as above shown as a hybrid map of Cologne city centre.

4.3.2 Specification

Hardware and OS	Windows XP, Linux
Software	• Qt 4.1.4 • Morgan AR/VR Framework
Core Features	• Access to Google Earth tile • Access to Google Maps tile • Hybrid mode • Mouse-based navigation
Status	Initial prototype which will be further developed.
Intended users	Showcase and tools developer
Showcases	WP 6, WP 7, WP 8, WP 9
Relevance beyond project	The tool is developed as a general tool, that can be used

4.3.3 Testing / Evaluation

AuthOr is still in an early prototype state and therefore only functional tests have been conducted. During the general assembly of the IPCity consortium in Berlin AuthOr's functionality has been demonstrated to the showcases. Afterwards all showcases stated an interest in using the tool for their development during the second year of the project.

4.4 DEVAL

4.4.1 Description

In Mixed Reality applications standard interaction devices are usually not the mouse and the keyboard as it is in desktop PC-based applications, instead a large variety of heterogeneous interaction devices is used (see Figure 15). Device abstraction layers (compare Section 2.1

and 3.2) are a possibility to allow applications to be independent of a particular device. DEVAL (DEVice Abstraction Layer) provides such a unified approach without focusing on a specific device class or specific aspects. Our approach is based on an overall device hierarchy, where each abstract interface exposes certain common aspects of a class of devices. Concrete devices are also represented by an interface of their own, which is derived from a number of abstract interfaces, therefore providing device specific functionality. Thus, we are able to abstract from the actual concrete device on the one hand and on the other hand allow application developer to access all functionality of specialized devices. The device hierarchy not only covers input devices, but also output devices and streaming devices.

Figure 15 - Some sample input and output devices for Mixed Realities applications

A general taxonomy covering all (input/output) devices used in Mixed Realty applications is provided by DEVAL and the classification allows application developers realizing Mixed Reality applications faster and more flexible, providing a significant higher flexibility regarding the devices actually used. The main requirements for our approach are:

- New devices and device types, not already part of the classification, to naturally extend it, without requiring any changes to the original taxonomy.
- Where devices can be sub-divided into logical sub-units or may only be used in part, should be reflected by the device hierarchy.
- Users should be able to replace one device by another of similar functionality or even a set of devices at runtime (i.e. the application developer does not need to be aware of the particular device).
- It should be possible to connect devices to any machine in the system, running on arbitrary operating systems.

Initially, CORBA has been chosen as the general communication mechanism, since it allows specifying the device interfaces in a platform-independent way and also allows for separating the implementation (skeleton) of the device interfaces from the interface itself (stub). Additionally CORBA can be efficiently used for mapping the hierarchy due to multiple inheritance of interfaces, thus a concrete device may be derived from multiple abstract interfaces.

All devices are derived from one device base class, providing a general interface to query and set common device information and properties, including the label, operational status and device features. The second level of the hierarchy is split into the main classes for input, output and streaming devices. Although streaming devices are technically input or output devices or a combination of both, they have specific requirements regarding the distribution of data and will therefore handled separately.

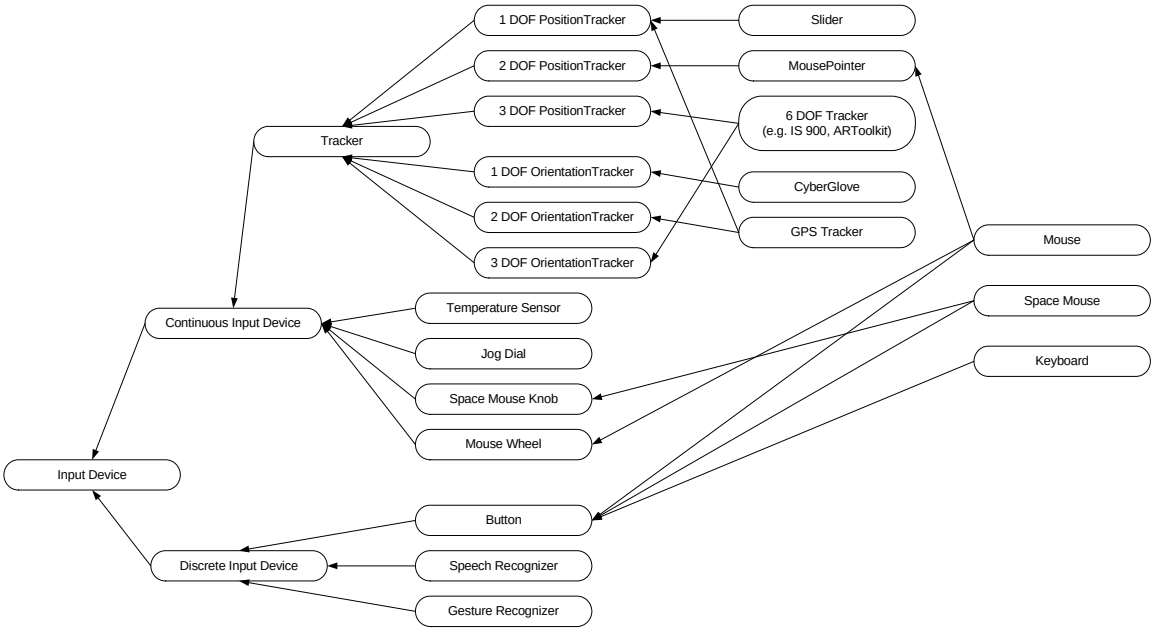

Figure 16 - The hierarchy for input devices. Interfaces for common device classes are inherited by derived device classes.

The most important subgroup of devices is the group of input devices (see Figure 16), which is also reflected by the fact that most existing device concepts focus on this particular subgroup. Input devices cover the whole range of devices used for interaction or for providing sensor information. Derived from *Input Device* are the two groups *Continuous Input Device* and *Discrete Input Device*. While a discrete input devices can only have a state from a finite set of predefined states (e.g. up or down), continuous input devices have a state within a predefined range (e.g. between 0 and 100). Continuous input devices differ also in that sense to discrete input devices, that their each state is not as critical and therefore missing state updates can be accepted. Two examples demonstrate this difference:

- An application wants to react on button clicks. The Button device sends the states *down-up-down-up* as four consecutive updates. In case the application misses the second and third update, it will only notice one button click.
- A virtual object is connected to the state of a Tangible Interface and the position and orientation of the object should reflect the pose of the real object. Even if several updates do not arrive at the application, the application will always be able to resynch if an update occurs.

These scenarios are explicitly exploited by DEVAL. As a general mechanism for receiving updates the *publish-subscribe* pattern is used, i.e. subscribers are able to register at a specific device or an abstract interface for updates. Each state change of a device triggers that all subscribers are notified by distributing the new state. Continuous input devices allow subscribers to receive state changes at a maximal frequency in order to limit the bandwidth. This functionality is provided on a per subscriber base, i.e. each subscriber may choose their own maximal frequency. Of course they may also decide to receive the updates at the maximum frequency. Another mechanism to reduce the number of updates for a single subscriber is that they may choose a subset of the handled objects. An ARToolkitPlus tracker handles a number of distinct markers, each of them identified by a unique id. A subset of valid ids can be specified and thus limiting the state updates to these ids.

As the input device hierarchy clearly shows interface may derive from multiple interfaces, e.g. Tracker6DOF inherits PositionTracker3DOF and OrientationTracker3DOF, and subscribers may subscribe at each interface providing state changes and will not have to deal with unnecessary information that is sent. A device derived from Tracker6DOF will provide its data through all interfaces in their data format, i.e. only 3-DOF orientation data is

send to subscribers from OrientationTracker3DOF. Therefore it is possible to not only exchange devices of the same type, but any device that inherits the specific interface.

Although an OrientationTracker2DOF sends the exact same data type as a PositionTracker3DOF, namely an array of three doubles, they are not exchangeable directly since they do not provide the same information, orientation versus position. Therefore, DEVAL introduces Adapters, which transform, filter or combine data from one or several devices and provide them through another interface, e.g. the head tracking of the Mobile AR System used in WP 8 transforms GPS tracking into a Cartesian coordinate system and combines it with an inertial tracker and acts as a Tracker6DOF. This is achieved by deriving the GPSInertialTracker adapter from Tracker6DOF, which subscribes at runtime at an OrientationTracker3DOF and a GPSTracker.

Most of the efforts have been put into the input devices, since they are the most important for the showcases. Some output devices already exist as early prototypes, e.g. a Text2Speech component. Streaming devices have not yet been completely defined, but a first concept has been put together.

DEVAL is demonstrable through the tech probes of WP 8 and the Behavior demonstrator (see Section 4.2).

4.4.2 Specification

Hardware and OS	Windows XP, Linux
Software	• CORBA • Morgan AR/VR Framework • Device drivers
Core Features	• Realization of concrete devices o Intersense IS-600, IS-900, IS-1200 o Intersense Inertia Cube 2 + 3 o XSens Tracker o ARToolkitPlus o Nmea GPS Tracker o Mouse and Keyboard • Publish-Subscribe pattern o Selection of max. update frequency o Selection of object subset o Subscription to derived interfaces
Status	stable prototype
Intended users	Showcase developers
Showcases	WP 8, WP 9
Relevance beyond project	Device abstraction layers are relevant for all Mixed Reality applications that are dependent from a variety of interaction devices.

4.4.3 Testing / Evaluation

DEVAL and the discrete devices have been tested by WP 8 and the Behavior demonstrator.

4.5 OpenTracker extension

4.5.1 Description

The generic tracking framework OpenTracker implements the well-known pipes & filters dataflow pattern and provides an open software architecture. OpenTracker's functionality is provided by nodes that describe sources, transformations and sinks of tracking data. Nodes are in turn supported by modules that implement any special functions such as device drivers, computations and network code. The Multi-Modal Event Streams of OpenTracker where introduced by Spiczak et al [55]. OpenTracker was further extended by the following new modules which integrate new devices and enable well known as well as new interaction possibilities for applications.

- Camera Control - Zoom and Pan Tilt Unit
- Space Device Module
- GoGo Interaction
- Virtual Key Module
- 3D to 2D Filter
- Sys Mouse Sink

The functionality and typical applications of these specific OpenTracker modules will be described in detail below.

Camera Control – Zoom and Pan Tilt Unit

The OpenTracker PTU-Node provides control for the viewing direction and the zoom level of the mounted camera. The interface for relative input to this node allows the use of various devices which are supported by OpenTracker and deliver relative data. A typical device used for the control is a Wireless Joy Pad as in Figure 17.

Figure 17 - PTU and Wireless Joypad

Space Device Module

Tracking input from an expert device like the Spaceball in Figure 18 can be distributed within the OpenTracker graph through a space device node. The output of this node is relative and therefore needs to be modified in order to be able to manipulate e.g. an absolute position of a virtual object in the scene. That's why this node would typically be used in combination with a GoGo Node which does this kind of data conversion.

Figure 18 - Spaceball - simultaneous 6DOF interaction

GoGo Interaction

This new module converts relative data input to an absolute location on the output side of the node. Furthermore this node provides some convenience functionality when used in combination with the Studierstube framework where a special scenegraph node is used for interactively configuring the behavior of the data conversion. Control in the opposite direction is used for interactively changing the virtual representation of the device. This is like changing the appearance of a mouse pointer.

Virtual Key Module

The Virtual Key is a generic tool which distributes the current state of a key on the keyboard in OpenTracker. This data source can be configured individually for indicating any keys state and can be used in multiple instances.

3D to 2D Filter

This Filter converts a 3D Input of an absolute device position with respect to a virtual screen (ASPD) see Figure 19 which is calibrated by 4 points in space and located by a 6 DOF input. The output of this filter is a normalized 2D point which is typically a screen point.

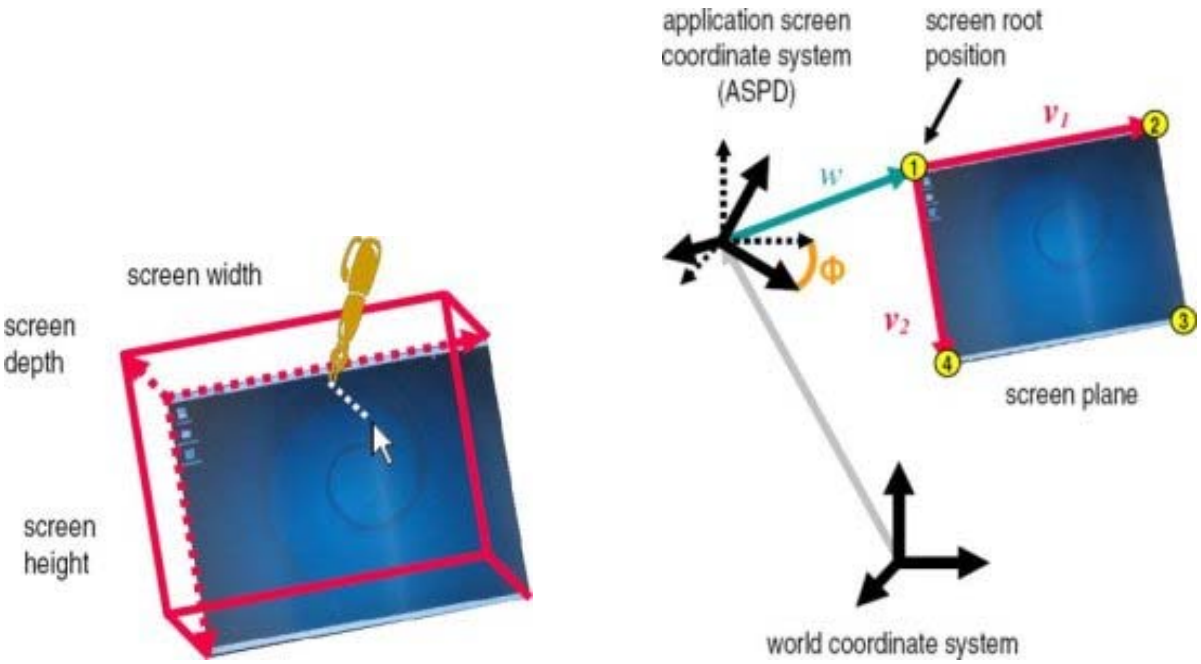

Figure 19 - Virtual screen location

The 3D to 2D filter is intended to be used with the Sys Mouse Sink node but not restricted to this configuration. The typical application is shown in where the projected image is calibrated and a pointing device is used as mouse input. The 3D to 2D Filter supports multiple instances and therefore allows controlling multiple computer systems with just one device.

Sys Mouse Sink

The Sys Mouse Sink has an input for absolute and relative data and therefore supports a very wide range of devices. It forwards the input events to the mouse control of the current computer system. An application scenario is visualized in Figure 20.

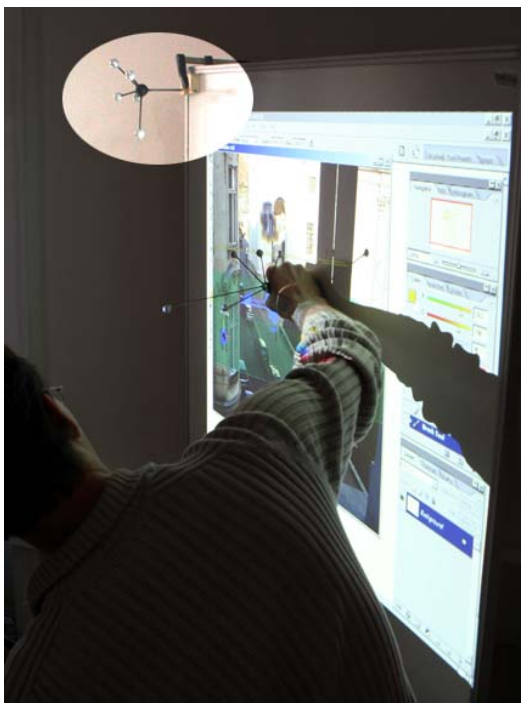

Figure 20 - 3D to 2D OpenTracker node in combination with the Sys Mouse Sink for controlling multiple applications

4.5.2 Specification

Hardware and OS	Windows XP, Linux
Software	• OpenTracker • Studierstube AR/VR Framework
Core Features	• Camera control unit node for PTU • Space Device input node • Virtual Key input node • 3D to 2D Filter node • Generic System Mouse Sink
Status	Stable prototypes which will be further developed.
Intended users	Showcase and tools developers as well as MR-application designers and expert users
Showcases	This prototype is available for all interested showcases and is currently used in WP6
Relevance beyond project	Can be reused in appropriate context due to generic design Contributes to MR research community

4.5.3 Testing / Evaluation

The prototypes were tested with Urban Sketcher on various occasions see WP6:

- Workshop at Sainte Anne, Paris, June 15th-16th, 2006
- Workshop at the Vienna Urban Renewal Office, Vienna, September 25th-26th, 2006
- Demo at General Assembly (reduced version), Berlin, October 18th-20th 2006
- Open Lab Night, Graz, October 9th 2006
- Vienna Workshop, November 16th, 2006

4.6 OpenVideo extension

4.6.1 Description

OpenVideo is designed to be extensible and easily configurable on windows and on Linux systems. The runtime structure of OpenVideo is implemented as a directed acyclic graph which consists of nodes and edges with special support for video data. The integration of OpenVideo into the Studierstube framework is the basis for video augmentation where the scene is rendered on top of the video background. In current demonstrators the visual input is used in the Urban Sketcher. In Figure 21 a white canvas is registered in the scene for sketching.

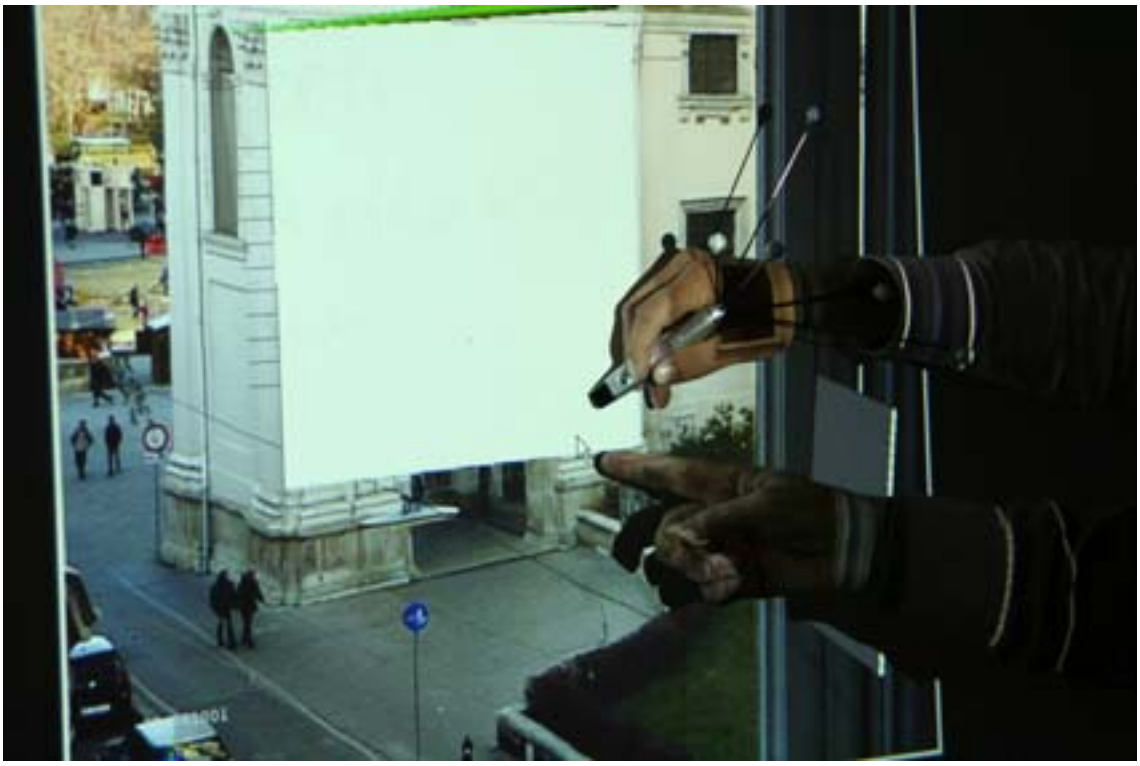

Figure 21 - Video background in Studierstube used in the Urban Sketcher

4.6.2 Specification

Hardware and OS	Windows XP, Linux
Software	• OpenVideo
Core Features	• Video image distribution
Status	Stable/beta prototype
Intended users	Showcase and tools developers as well as MR-application designers and expert users
Showcases	This prototype is available for all interested showcases and is currently used in WP6
Relevance beyond project	Can be reused in appropriate context due to generic design Contributes to MR research community

4.6.3 Testing / Evaluation

The prototype was tested with Urban Sketcher on various occasions see WP6:

- Workshop at Sainte Anne, Paris, June 15th-16th, 2006
- Workshop at the Vienna Urban Renewal Office, Vienna, September 25th-26th, 2006
- Demo at General Assembly (reduced version), Berlin, October 18th-20th 2006
- Open Lab Night, Graz, October 9th 2006
- Vienna Workshop, November 16th , 2006

4.7 Color Table

4.7.1 Description

The main ambition of the Color Table is to support collaborative working scenarios by providing tangible interaction possibilities.

The system consists of a white surface placed on a conventional table and a large amount of colored objects of different shape and size that may be placed and manipulated on the table. The users may thus stand around the table and interact simultaneously from different positions.

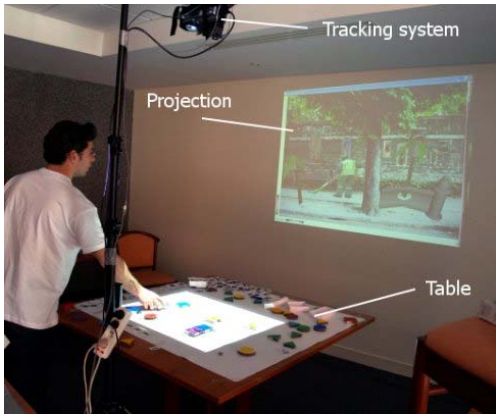

Figure 22 – Overview of the Color Table

A video camera is placed above the table and captures the current state of the system. It tracks the positions, colors and sizes of the different colored shapes placed on the table.

Two projections allow reflecting the current state of the system. A horizontal one is situated directly on the table and serves therefore as augmentation of the tangible table. It is mainly used to provide additional information such as feedback to the user. A vertical projection is placed in front of the users and shows a visualization of the situation the users are creating. The individual parts of the Color Table are explained in Figure 22.

The system of the Color Table includes different tangible interaction modules that may be used independently:

- Colored Objects Interface
- Barcode Interface
- Rotating Color Table

Colored Objects Interface

The colored shapes, currently cylinders of 7 different colors, may be repositioned on the table in order to interact with the system. Their positions, sizes and colors are used to manipulate the state of different elements of an application.

In this way, the colored objects interface can for example be used to control virtual objects in a mixed reality scene. The colored shapes are linked to various types of digital content that may be moved within space by moving the corresponding physical objects on the table. Each color defines a different virtual object. Small, green triangles can be attached to cylinders in order to manipulate the orientation of 3D models. Figure 23 shows the colored shapes in use.

Figure 23 – The colored shapes may be repositioned on the table.

Barcode Interface

The barcode interface enables users to easily access elements of the Hyper Media Database by reading in dedicated barcodes. These elements can be 3D models, 2D partly transparent pictures and sound files to be used within the application, or commands controlling other parameters of the application.

Depending on the purpose of the application, the barcodes may be presented differently to the users. Figure 24 shows the arrangement of barcodes on separate sheets within a booklet (left), and on a cardboard (right).

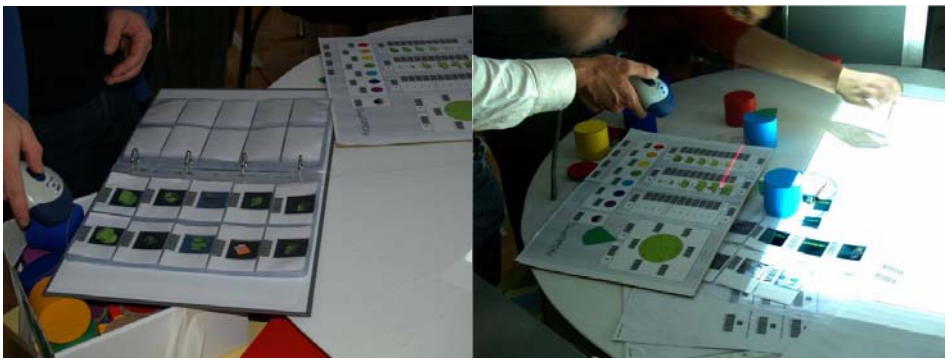

Figure 24 – Users may read barcodes in order to select elements of the Hyper Media Database.

Rotating Color Table

The rotating Color Table (see Figure 25) is a tangible interface manipulating of the orientation of the viewpoint within a 3D environment. The disk, on which the colored shapes are lying, may be rotated in order to rotate the whole scene around the user.

Figure 25- The Rotating Color Table at the Vienna Workshop

4.7.2 Specification

Hardware and OS	4 Dell Laptops (IP M Processor, 2.13 Ghz) Windows XP
Software	• JAVA 1.5.0_05 • JMF 2.1.1e • Atelier Framework (see 2.2.5) • Hyper Media Database • Apache Tomcat 4.1 • MySQL 4.0.13 • OpenTracker 2.0 (see 2.1.1) • Studierstude 4.0 (see 2.5.3) • OpenCV beta 5
Core Features	• tangible Interaction through manipulation of colored objects • barcodes to activate commands and selecting content • rotating color table for navigation.
Status	Technology Probe
Intended users	5-7 users coming from various fields
Showcases	This prototype is available for all interested showcases and is used in WP6
Relevance beyond project	

4.7.3 Testing / Evaluation

The demonstrator and the included technology probes have been tested at 4 Workshops:

- St. Anne, Paris, Workshop (June 15-16, 2006)

Figure 26 - Workshop at St. Anne, Paris

- GB16, Vienna, Workshop (September 25-26, 2006)

Figure 27 - Workshop GB16, Vienna

- MCIS, Venice, Workshop (October 8, 2006)
- Vienna Workshop with architects (November 16, 2006)

Figure 28 - Workshop with architect, Vienna

The demonstrator has also been shown at the Beginner's Day at Vienna University of Technology

4.8 MMS Media Extractor

4.8.1 Description

The MMS Media Extractor demonstrator is a smartphone application which can be used to extract media (text, images, video, sound) from a MMS (Multimedia Messaging System) message. This media and associated metadata can then be automatically sent over Bluetooth™ using OBEX protocol to any device that supports these protocols. Currently there is one component (a WP5 deliverable) which is capable of receiving these files. The component is depicted in Figure 29), on the left side of the figure (right side showing how the media is then used). The concrete plan is to use the MMS Media Extractor with the WP7 scenarios Forage, and generally in any other showcase in IPCity where generation and entry of user generated media with mobile phones is feasible.

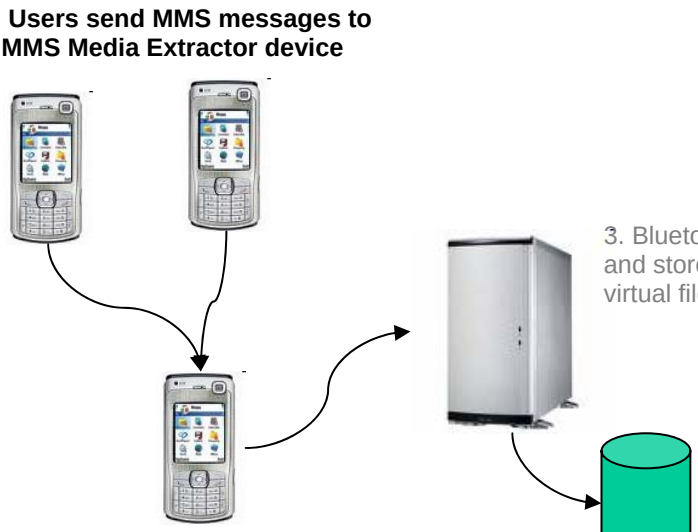

Figure 29 - MMS Media Extractor and examples of related server side components.

The MMS Media Extractor on the phone side consists of a Symbian application programmed with C++. The application monitors the phone's messaging inbox for MMS (multimedia) messages. As a message arrives, the attachments (text, images, videos, sound files, etc) are extracted. Also, a meta-information file, describing the user that sent the message and the sent files is created. This meta-information file along with the multimedia and text files are sent over Bluetooth™ using OBEX (Object Exchange) protocol to a PC. In future, more meta-information in addition to the user can be added to enhance the functionality of the application (e.g. location based on CellID or GPS, nearby Bluetooth device names).

4.8.2 Specification

Hardware and OS	Nokia N70, Symbian OS v. 8.1a, Series60 2nd Ed FP3
Software	• MMS inbox reader • BT/OBEX sender
Core Features	Extracts contents of multimedia messages and sends the contents along with a metafile to a PC side system for further processing
Status	Beta prototype
Intended users	Any users who need to import media by using camera phones
Showcases	WP7, others
Relevance beyond project	Usable in other similar contexts

4.8.3 Testing / Evaluation

The component been tested but further integration testing will be done during December 2006/January 2007.

4.9 Extendable DataMatrix reader

4.9.1 Description

Using camera phones to read and interpret two dimensional barcodes for various purposes is currently a popular topic in both research and in practical applications. Two dimensional barcodes can be used to access data and services, like bus time tables, product information etc. There are standards for the two dimensional barcodes, but implementations are usually application specific and closed. This implementation uses a publicly availabe DataMatrix standard (Figure 30). Thus there are no need to pay for any licenses when using the DataMatrixes. For more information, see e.g. http://en.wikipedia.org/wiki/Datamatrix.

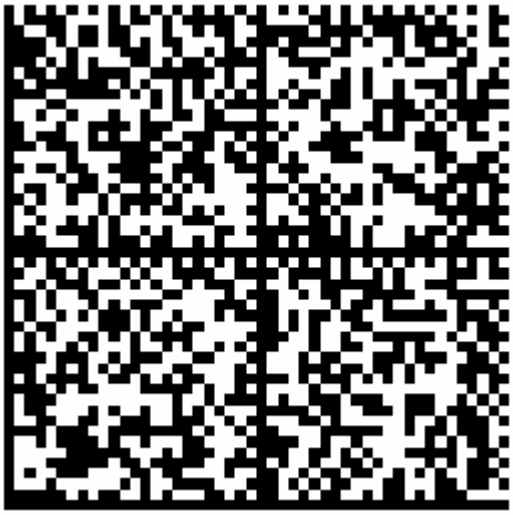

Figure 30: An example of a DataMatrix barcode.

This implementation is based on the design of extendibility. The application is designed based on a plugin architecture. This enables us to extend the functionality of the system so that the actual content of the read barcode specifies also the intended interpretation and handling of the barcode data. It is designed that the produced barcode can contain a URN (Uniform Resource Name) or a URL (Uniform Resource Locator). The URL is used in most currently known implementations. The URL can be a link to a web page or a web service, such as a product description or a location specific information such as a timetable for a specific bus stop at the current time. In our design the barcode can – in addition to URLs – contain a URN; an application specific code, followed by the application specific data. This enables us to create applications and services on the phone that are not dependent on the network or services on the network. The URN is embedded in the beginning of the barcode data, followed by the application data.

As the barcode is read the application first checks if the code contains a URL or a URN. If the code contains a URL, the browser on the device is directed to that page. If the code contains a URN, the plugin repository on the phone is searched for the plugin component that supports handling of that specific URN. If one is found, the remaining of the data is passed on to that plugin component. New plugin components can be installed on the device to support new types of extensions. The handling of the data is plugin (application) specific.

As an example, the application can be used to provide location specific data and services to the user without accessing the phone network. For example, the barcode could contain an instruction for the application to capture an image using the camera on the phone and send it to a specific recipient (e.g. the MMS or Email Entrance; see WP5 demonstrators). Or the barcode could instruct the phone to get a random media file over Bluetooth™ from a nearby Bluetooth enabled media server – a story that a user left about a specific spot in the city (relates to the City tales showcase).

Currently the demonstrator is in early beta stage; the image capturing and analysis is on final development and testing stage, as the plugin architecture is in early design and implementation phase.

4.9.2 Specification

Hardware and OS	Nokia N70, Symbian OS v. 8.1a, Series60 2nd Ed FP3
Software	Image analysis component, data handler plugin components
Core Features	Reads two dimensional bar codes and passes the data to application specific plugin component for handling
Status	Beta prototype
Intended users	Showcase users
Showcases	All showcases
Relevance beyond project	Will be published as an open source implementation

4.9.3 Testing / Evaluation

Image capturing and analysis works currently in beta stage. Plugin components will be developed starting January/February 2007.

5 Dissemination

5.1 Publications

Ohlenburg, Jan, Broll, Wolfgang, and Lindt, Irma, *"DEVAL – AR/VR Device Abstraction Layer Implementation"*, accepted as publication in Proceedings of Universal Access in Human-Computer Interaction (UAHCI) 2007, 2007.

5.2 Workshops

During the following workshops the results of the Color Table have been demonstrated and have been evaluated:

Workshop at Sainte Anne, Paris, June 15-16, 2006

The Color Table producing simple mixed reality scenes, the texture brush and the Urban Sketcher have been presented and discussed as design concepts in the psychiatric hospital in Sainte Anne. In this first workshop only the architects engaged at Sainte-Anne, some hospital staff including two old professors of psychiatry, and additional urban planners were included.

Workshop at the Vienna Urban Renewal Office, Vienna, September 25-26, 2006

The rotating Color Table producing see-through augmentations and panorama augmentations, as well as the Urban Sketcher and the 3D image reconstruction have been shown as early prototypes in the urban renewal office of Vienna's 16th district. Participants in this workshop were members of IPCity from TUW, TUG, UOulu, and UMVL on the one hand, members of the urban renewal office itself, as well as two collaborating architects, an urban sociologist, and two representatives of local authorities.

The Beginner's trail at TUW, October 2, 2006

The rotating Color Table in use with a panorama augmentation has been shown at the TUW within the scope of a presentation of our department to new students.

MCIS Workshop, San Servolo, Venice, October 8, 2006

In this workshop, the Color Table has been presented outdoors, in combination with a see-through augmentation. Participants have been young artists and people with a background in CSCW and philosophy. They were asked to select images and sound for preparing a scene or sequence of scenes that would allow them to explore aspects of presence.

Vienna Workshop, November 16, 2006

The rotating Color Table producing see-through augmentations and panorama augmentations has been showed at TUW. The barcode application has been improved in order to provide more control to the users. Participants of the workshop have been with visitors from Oslo University (mainly colleagues from the interaction design field).

6 Appendix

The Requirement Analysis (see Section 3) has been conducted based on a questions catalogue to be filled out by all showcases. On the following pages, the questions catalogue can be found.

Questions Catalogue for IPCity Showcases

1. Infrastructure

1.1 What programming languages do you intend to use?

- C/C++
- C#
- Java
- Other?

1.2 What programming languages are not feasible for your? Explain why.

- C/C++
- C#
- Java
- Other?

1.3 What scripting languages do you want to use?

- Phyton
- Perl
- JavaScript
- Inscene scripting
- Other

1.4 Would you prefer to use scripting languages exclusively for application development?

1.5 What software environments do you intend to use?

1.6 What hardware platforms and OS is your showcase most likely to run on?

- PC/Windows XP
- PC/Linux
- PDA/Windows Mobile 5.0
- Smartphone/Symbian
- Smartphone/J2ME (Version?)
- PSP

1.7 What hardware platforms and OS would you like your showcase to run on – if available?

- PC/Windows XP
- PC/Linux
- PDA/Windows Mobile 5.0
- Smartphone/Symbian
- Smartphone/J2ME (Version?)

- PSP

1.8 If you intend to use PDAs, which are the requirements concerning memory size, processing power, 3D accelerator chips and screen size?

1.9 If you intend to use PDAs, which features does it have to provide?

- WiFi
- GMS/GPRS
- Camera
- GPS

1.8 Which PDAs have you used before? What are the pros and cons?

1.10 If you intend to use mobile phones, which are the requirements concerning memory size, processing power and screen size?

1.11 If you intend to use mobile phones, which features does it have to provide?

- SMS
- MMS
- GPRS
- 3G
- Camera
- Live video
- Color screen
- Touch sensitive screen
- GPS
- WiFi

1.12 Which mobile phones have you used before? What are the pros and cons?

1.13 Do you have other requirements concerning hardware infrastructure?

2. Network architecture

2.1 What kind of network architecture will you use in your showcase?

- Client/Server
- Peer-to-Peer
- Multicast groups
- Ad hoc connections

2.2 Connectivity / Disconnectivity. Are clients most of the time connect via internet to the application?

2.3 Will disconnectivity be exploited within the showcase?

2.4 Which connection types are you planning to use?

- LAN
- WiFi
- WiMax
- GSM
- GPRS
- UMTS
- Bluetooth
- Infrared
- Other?

3. Messaging

3.1 What kind of data sharing support does your showcase need?

- Message passing
- Chat
- Streaming media (audio and video)
- SMS/MMS
- Replicated application state
- VoIP
- Remote object invocation (CORBA, COM, RMI)
- Other

3.2 What kind of data needs to be distributed?

- Files (Scripts, Images, Audio Clips, Video, Scenes)
- Passive data types (integers, floats, structs…)
- Language objects (C++ or Java objects)
- Other

3.3 How fast does the data need to be distributed between clients or clients and server?

- Within milliseconds
- Within seconds
- Within minutes
- Other

3.4 What kind of message passing to you intend to use?

- Push (subscribers receive messages automatically)
- Pull (clients have to query for new messages)

3.5 How often does the application need to send data and how big is the expected data size?

3.6 How many processes simultaneously need to share data?

3.7 Will data need to be distributed between processes written in different programming languages or processes running on different OS?

4. Cross-Reality Interaction Tools

4.1 If you want to be able to exchange devices for interactions, which devices would you like to exchange?

- GPS and 6DOF Tracker (e.g. ARToolkitPlus)
- Speech commands and Gestures and Keyboard

4.2 Do you want to combine devices to form a new input device? If yes, can you think of any?

- ARToolkitPlus and inertial tracker
- GPS and inertial tracker
- …

4.3 Are you scenario settings likely to change (i.e. in case you cannot also rely on the same hardware/infrastructure setup)?

4.4 What kind of interactions would you like to use in your showcase?

- Speech
- Gestures
- Tangible user interface
- Wands
- Scribble
- Keyboard and/or Mouse
- Other?

4.5 Do you want to create and evaluate new interaction mechanisms? If yes, how do you plan to do this?

4.6 Which MR interactions have you used in other projects? What are your experiences?

4.7 Questions about Ambient Interfaces…

5. Mixed-Reality Infrastructure

5.1 Where will your showcase be operating?

- Indoors (Constrained/Unconstrained)
- Outdoors (Constrained/Unconstrained)
- Indoors/Outdoors (Constrained/Unconstrained)

5.2 Do you intend to use Augmented Reality in your showcase? If yes, on which devices do you plan to use AR applications?

- Workstation
- Desktop/Laptop (not worn)
- Laptop/Tablet PC (carried, worn on a backpack)
- PDA
- Portable game console
- Smartphone
- Other?

5.3 If you intend to use AR in your showcase, what kind of AR are you planning to use?

- Video augmentation
- See-through HMD
- Projection-based AR
- Other?

5.4 Which devices are you planning to use within your showcase, no matter for what purpose?

- Workstation
- Desktop
- Laptop
- Tablet PC
- PDA
- Portable game console
- Smartphone
- Mobile Phone

5.5 Do you intend to use head-worn displays? If yes, to which devices to you want to connect it to?

- Workstation
- Desktop
- Laptop/Tablet PC
- PDA
- Portable game console
- Other?

5.6 Which kind of output devices do you intend to use?

- Monitor
- Projection Wall
- Public displays
- HMD (mono/binocular/stereo)
- Audio
- Actuators (Vibrators)
- Others?

5.7 Do you intend to use position tracking of users and/or objects? If yes, what information you do require?

- Full 6DOF position and orientation.
- GPS position.
- GPS position combined with orientation.
- GSM cell.

5.8 If you intend to track users/objects, which technologies are you planning to use?

- Marker-based tracking.
- GPS tracking.
- Inertial tracking.
- RFID.
- Marker-less tracking.

5.9 If you intend to use positioning of users/objects, on which devices do you want to use positioning?

- Workstation
- Desktop
- Laptop
- Tablet PC
- PDA
- Portable game console
- Smartphone
- Mobile Phone

5.10 What would be a realistic acceptable latency for user/object tracking? In case this differs for different devices, please specify.

5.11 What would be the maximal acceptable latency for user/object tracking? In case this differs for different devices, please specify.

5.12 Do you expect tracking to be needed in unconstrained environments?

5.13 Do you intend to use different tracking approaches for different environments? E.g. GPS outdoors, CV-base tracking indoors?

5.14 Do you expect the tracking to switch seamlessly between different environments? I.e. ubiquitous tracking.

5.15 Do you have experiences with different tracking systems, which apply to your showcase? Please specify.

6 Summary of Showcase

Please describe your initial showcase scenario(s) as detailed as possible. In order to extract other requirements, which were not covered by the previous questions.

References

[1] Abrams, M. *"Device-Independent Authoring with UML".* Proc. W3C Workshop Web Device Independent Authoring, 1999

[2] Abawi, D., Dörner, R., Haller, M., and Zauner, J. *"Efficient Mixed Reality Application Development"*. Proceedings of the 1st European Conference on Visual Media Production. London, March, 2004.

[3] Broll, W., Lindt, I., Ohlenburg, J., Herbst, I., Wittkämper, M., and Novotny, T., *"An Infrastructure for Realizing Custom-Tailored Augmented Reality User Interfaces"*, IEEE Transaction on Visualization and Computer Graphics, Vol.11,No.6, November 2005, pp. 722-733.

[4] Grimm, P., Haller, M., Paelke, V., Reinhold, S., Reimann, C., and Zauner, J., *"AMIRE – Authoring Mixed Reality"*. The First IEEE International Augmented Reality Toolkit Workshop, Darmstadt, Germany, September, 2002.

[5] Haringer, M. and Regenbrecht, H. T., *"A pragmatic approach to Augmented Reality Authoring"*, ISMAR'02, Darmstadt, Germany, September/October 2002.

[6] Höllerer, T., Feiner, S., and Pavlik, J. *"Situated Documentaries: Embedding Multimedia in the Real World"*. Proceedings of the ISWC'99 (International Symposium on Wearable Computers", San Francisco, CA, October 18-19, 1999.

[7] Kato, H., Billinghurst, M., Blanding, B., and May, R. *"ARToolKit"*. Technical Report. Hiroshima City University. December 1999.

[8] Kurose, J. F., and Ross, K., *"Computer Networking: A Top-Down Approach Featuring the Internet"*. Addison-Wesley, 2nd edition, 2002.

[9] Ledermann, F. and Schmalstieg, D., *"APRIL: A High-Level Framework for Creating Augmented Reality Presentations"*. Proc. IEEE Virtual Reality, 2005

[10] Lindt, I., *"Exchangeability of 3D Interaction Techniques"*, IEEE VR 2005 Workshop on New Directions in 3D User Interfaces, Bonn, 2005.

[11] Ohlenburg, J., Herbst, I., Lindt, I., Fröhlich, T., and Broll, W. *"The MORGAN Framework: Enabling Dynamic Multi-User AR and VR Projects"*, Proc. ACM Symp. Virtual Reality Software and Technology (VRST 2004), *pp. 166-169*, 2004

[12] Pesce, M. D., *"Programming Microsoft DirectShow for Digital Video and Television"*, Microsoft Press, 2003.

[13] Sandor, C. and Reicher, T., *"CUIML: A Language for the Generation of Multimodal Human-Computer Interaction"*, Proc. European UIML Conf., 2001

[14] Tayler II, R.M., Hudson, T.C., Seeger, A., Weber, H., Juliano, J., and Helser, A.T., *"VRPN: A Device-Independent, Network-Transparent VR Peripheral System"*, Proc. ACM Symp. Virtual Reality Software and Technology (VRST 2001), 2001

[15] Trewin, S., Zimmermann, G., and Vanderheiden, G. *"Abstract User Interface Representations: How well do they Support Universal Access?"* In Proceedings of CHI 2003, 2003.

[16] Zauner, J., Haller, M., Brandl, A., and Hartmann, W. *"Authoring of a Mixed Reality Assembly Instructor for Hierarchical Structures"*, ISMAR'03, Second International Symposium on Mixed and Augmented Reality, 2003.

[17] Beckhaus, S., Lechner, A., Mostafawy, S., Trogemann, G., and Wages, R. *"alVRed - Tools for storytelling in virtual environments"*. Internationale Statustagung Virtuelle und Erweiterete Realität, Leipzig, Germany, 2002.

[18] Billinghurst, M., Kato, H., and Poupyrev, I. *"The Magic-Book - Moving Seamlessly between Reality and Virtuality"*. IEEE Computer Graphics and Applications, vol. 21(3), 6-8, 2001.

[19] Bimber, O., Fröhlich, B., Schmalstieg, and D., Encarnacao, L. M. *"The virtual showcase"*. IEEE Computer Graphics and Applications, 21(6):48-55, 2001.

[20] Clark J. XSL transformations (XSLT) version 1.0 - W3C recommendation. http://www.w3.org/TR/xslt, 1999.

[21] Conway, M., Pausch, R., Gossweiler, R., and Burnette, T. *"Alice: A rapid prototyping system for building virtual environments"*. Proceedings of ACM CHI'94 Conference on Human Factors in Computing Systems, volume 2, 295-296, 1994.

[22] Fuhrmann A., Prikryl, J., Tobler, R., and Purgathofer, W. *"Interactive content for presentations in virtual reality"*. Proceedings of the ACM Symposium on Virtual Reality Software & Technology.

[23] Kretschmer, U., Coors, V., Spierling, U., Grasbon, D., Schneider, K., Rojas, I., and Malaka, R. *"Meeting the spirit of history"*. Proceedings of VAST 2001, Athens, Greece, 2001.

[24] Ledermann, F. *"An authoring framework for augmented reality presentations"*. Master's thesis, Vienna University of Technology, 2004.

[25] MacIntyre, B., Gandy, M., Dow, S., and Bolter, J. *"DART: A Toolkit for Rapid Design Exploration of Augmented Reality Experiences"*. Proceedings of User Interface Software and Technology (UIST'04), Sante Fe, New Mexico, 2004.

[26] MacWilliams, A., Sandor, C., Wagner, M., Bauer, M., Klinker, G., and Bruegge, B. *"Herding Sheep: Live System Development for Distributed Augmented Reality"*. Proceedings ISMAR 2003, Tokyo, Japan, 2003.

[27] Object Management Group. *"Unified modeling language (UML)"*, version 1.5. http://www.omg.org/technology/documents/formal/uml.htm, 2003.

[28] Reitmayr G. and Schmalstieg, D. *"OpenTracker - an open software architecture for reconfigurable tracking based on XML"*. Proceedings of IEEE Virtual Reality 2001, pages 285-286, Yokohama, Japan, 2001.

[29] Schmalstieg D., Fuhrmann, A., Hesina, G., Szalavari, Zs., Encarnacao, L. M., Gervautz, M., and Purgathofer, W. *"The Studierstube augmented reality project. PRESENCE - Teleoperators and Virtual Environments"*, 11(1).

[30] Stoakley, R., Conway, M., and Pausch, R. *"Virtual reality on a WIM: interactive worlds in miniature"*. Proceedings on human factors in computing systems, 265-272, Denver, USA, 1995.

[31] Strauss, P. and Carey, R. *"An object oriented 3D graphics toolkit"*. Proceedings of SIGGRAPH '92, 1992.

[32] Tramberend H. *"Avocado: A distributed virtual reality framework"*. Proceedings of IEEE Virtual Reality 1999, 1999.

[33] VRML Consortium. *"VRML97 specification"*. Specification 147721:1997, ISO/IEC, 1997.

[34] Web3D Consortium. X3D specification website. http://www.web3d.org/x3d/specifications/.

[35] 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.

[36] 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.

[37] V. Maquil. *"Automatic generation of graphical user interfaces in Studierstube"*. https://www.ims.tuwien.ac.at/publication detail.php?ims id=140, Juli 2004.

[38] http://www.xulplanet.com/, http://www.mozilla.org/projects/xul/

[39] Salminen T. and Riekki J. *"Lightweight middleware architecture for mobile phones"*. Proc. 2005 International Conference on Pervasive Systems and Computing, Las Vegas, NE, 147-153, 2005.

[40] Repo, P. and Riekki J. *"Middleware support for implementing context-aware multimodal user interfaces"*. Proceedings of the 3rd international conference on Mobile and ubiquitous multimedia. pp. 221 - 227, 2004.

[41] Ishii, H., and Ullmer, B. *"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.

[42] Ullmer, B., Ishii, H., and Glas, D. *"mediaBlocks: Physical containers, transports, and controls for online media"*. Computer Graphics, 32(Annual Conference Series):379–386, 1998.

[43] Ullmer, B., and Ishii, H. *"The metadesk: Models and prototypes for tangible user interfaces"*. In ACM Symposium on User Interface Software and Technology, pages 223-232, 1997.

[44] Camarata, K., Do, E. Y.-L., Johnson, B. R., and Gross, M. D. *"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.

[45] Weiser, M. *"The computer for the 21st century"*. In Scientific American, volume 265 of 3, pages 94-104, 1991.

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

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

[48] Fitzmaurice, G. W. *"Graspable User Interfaces"*. PhD thesis, University of Toronto, 1996.

[49] Suzuki, H., and Kato, H. *"Algoblock: a tangible programming language, a tool for collaborative learning"*. In Proceedings of 4th European Logo Conference, pages 297-303, Aug. 1993.

[50] Adaptive Communication Environment, http://www.cs.wustl.edu/schmidt/ACE.html, (13.04.2006)

[51] SGI OpenInventor, http://oss.sgi.com/projects/inventor, (13.04.2006)

[52] UIML Homepage, http://uiml.org, (13.04.2006)

[53] VRPN: virtual reality peripheral network, 2005, U. of North Carolina at Chapel Hill. http://www.cs.unc.edu/Research/vrpn/.

[54] Henning, M. and Vinoski, S., "*Advanced CORBA Programming with C++"*. Addison-Wesley, 1999.

[55] Spiczak, J., DiMaio, S., Reitmayr, G., Schmalstieg, D., Burghart, CR., Samset, E., "Multi-Modal Event Streams for Virtual Reality" MultiMedia Computing and Networking Conference 07 (San Jose, CA)

[56] Greenhalgh, C., Izadi, S, Mathrick, J., Humble, J., and Taylor, I. *"ECT: A Toolkit to Support Rapid Construction of UbiComp Environments"*. In Proceedings of UbiSys 2004, 2004.

Acknowledgements and Further Information

IPCity is partially funded by the European Commission as part of the sixth framework (FP6-2004-IST-4-27571

For further information regarding the IPCity project please visit the project web site at:

ipcity.eu