Integrated Project on Interaction and Presence in Urban Environments

FP6-2004-IST-4-27571

ipcity.eu

First Prototypes of Interaction Tools

Deliverable D4.2

Doc-Id:	**D 4.2**
Version:	**1.0**
Author(s):	**Jan Ohlenburg, Markus Sareika, Denis Kalkofen, Gerhard Reitmayr, Iris Herbst, Giulio Jacucci, Valérie Maquil, Antti Juustila, Wolfgang Broll**
Date:	**2008-02-07**
Status:	**Ready for Review**
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 continued developing the first year's demonstrator and started a couple of new technologies, e.g. the Multi-Touch Display and the Augmented Map Table. Basis for the work in this year have been the results from the showcase evaluations and review and redesign discussions of the consortium.

This document summarizes the scientific and technical achievements in workpackage 4 during the second year. According to the internal report I4.3 which includes a redesign analysis for all year one demonstrators for WP 4, first prototypes have been developed which mostly have already been tested within several showcase.

Intended Audience

This document is intended to all partners of the project, the reviews for the second project's phase, the EU commission and the public.

1 Workpackage Objectives

Success Criteria Phase II	(1) The tools and services developed within the current working period (a)/since project start (b)/based on existing technology (c) (2) Significant progress in the development of technologies of each major building block (3) Each technology requested by at least two showcases and actually used by at least one showcase (4) Each technology developed is contributing to the overall project objectives (5) Technology used or requested by other projects or third parties and to what extend (6) Submission of at least six conference and/or journal papers (at least one for each of the five major building blocks) (7) Identify new technologies developed within the project not available elsewhere
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.
Results Phase II	The following tools are new developments within phase II (1a): • **Multi-Touch Display:** Collaborative Touch Display for concurrent multi-user interaction • **Mobile Media Collector:** Mobile device and application for collecting, browsing, and saving location specific media • **Location Based Media Browsing on Paper Maps:** Map augmentation by location based media on a smartphone The following tools from phase I have been extended (1b): • **Interaction Prototyping Tool:** Template mechanisms for easy reuse of existing interaction techniques, a

	visual programming environment. • **AuthOr:** PDA version, new Overlays, Code redesign • **MRIML:** Modeling Language for Mixed Reality Interfaces • **ColorTable:** Revised interaction modules, new workspace organization • **OpenVideo:** Switching between multiple dynamic videos, sending and receiving multiple videos The following pre-existing tools have been extended (1c): • **Augmented Map Table:** Tabletop AR environment for augmenting maps with dynamic information • **OpenTracker:** Object-oriented approach to access tracking devices, and to fuse, to filter and transform their input data. The following tools have been requested by other projects or third parties (5): • **DEVAL:** CoSpaces, Deutz, Exploar Number of submissions to conference and/or journals (6): **6** full conference papers, **3** workshop papers, **3** posters New technology developed in workpackage not available anywhere else (7): • Multi-Touch Display • Augmented Map Tracking • ColorTable • Mobile Media Collector
Evaluation Results Phase II	…
Objectives Phase III	• 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. • Tangible User Interfaces: Clarifing design issues related to the collaborative creation of mixed-reality configurations and making use of material and spatial properties in designing both, physical interface, as well as multiple and simultaneous interactions. Developing and applying methods allowing users to rapidly learn, use and understand the interactions with the ColorTable to gain a basic understanding of the user's presence related to interaction. • Augmented map table: Integration with the existing application components Urban Sketcher and Color Table. The map table will present a tangible interface to various elements manipulated in the Urban Sketcher and Color Table, such as drawing planes and placeholder objects. • 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. • MapLens (augmented maps on mobile devices over paper maps): development of the mobile client on Symbian OS smartphone in collaboration with HIIT/TUG/CAM. Field trials planned in Fall 2008. • Mobile media collector: design and development of the tool for collecting and browsing location based media using mobile devices. First prototype implementation planned in Fall 2008. • UrbanSketcher Interface Streamlining: UrbanSketcher will also undergo extensive user interface redesign, e.g. usability improvements, strengthening the collaborative properties, enhanced visual and/or audio feedback of user action, or additional functions to handle the assumed ground plane in the MR scene. Additionally developing of sketching/3d modelling tools. • Multitouch large screen. Set up of an infrared tracking based touch screen with rear projection, Development of a catalogue of gestures to manipulate virtual objects using multiple fingers.

2 Review Report Feedback

Based on the feedback of the reviewers we have taken the following actions:

- WP 4 and WP 5 have described our shared vision about of the technology developed within the project and present our shared vision in both deliverables D4.2 and D5.2 (see Section 3).
- The reviewers had the concern that we did not have been ambitious enough. In order to make our achievements more visible and point out which technologies are new, which are continued pre-existing work and which technologies are going beyond the current state-of-the-art, we have clearly listed all tools developed within this year in the table of Section 1. There we also point out, that a number of technologies are not in that form available anywhere else.
- The remark that we only continue pre-existing technologies, is not true and in order to make this evident, we have separated all technologies developed in year 2 into the following categories, which are listed in Section 1: Pre-existing technology, technology started with IPCity, technology started within year 2.
- Each technology developed now states the related work in this deliverable and compares our achievements against the current state-of-the-art.

3 Conceptual Infrastructure Framework for Interaction and Presence Experience

From a technical point of view, enabling presence and experience in mixed reality environments requires a multi-layer approach. Firstly, providing the general infrastructure (hardware and software) and services to realize MR systems. Secondly, the provision of higher-level tools for authoring MR environments and supporting the realization of MR user interfaces. Thirdly, the development of the actual MR application including application-specific features and tools. Figure 1 further clarifies the concept between the various building blocks of MR technology and gives an overview of interconnections between the developed tools and senses and sensations of presence.

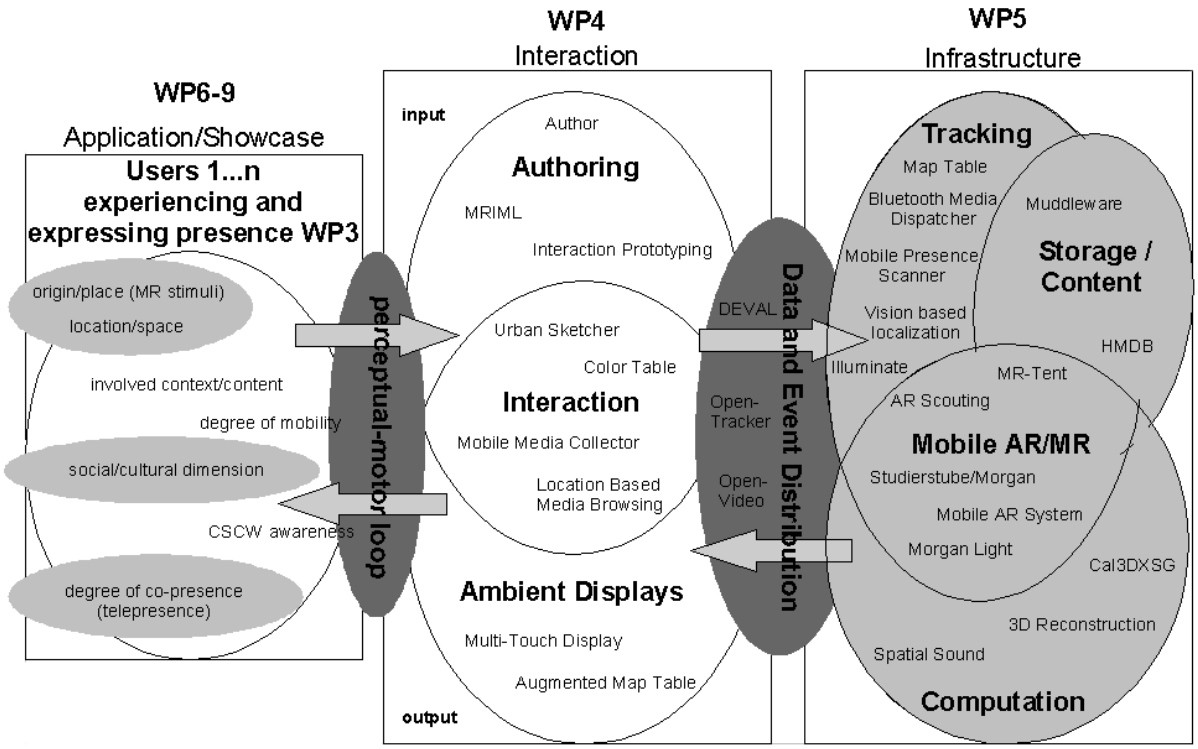

Figure 1: Conceptual Infrastructure Framework Overview

The first section of Figure 1 represents the user in his current context of environment and intention. This is the state of presence or co-presence which is experienced and defines an entry point for mixing reality. The initial situation imposed by a showcase environment implicitly defines suitable characteristics for interface technology. Furthermore the application of software tools and building blocks, which define the infrastructure, allows to dynamically route the exchange of information not only between two users but enables 1-to-n as well as n-to-1 communication, while several modalities supported by the hardware interface can be involved in the communication process.

Interaction between human and machine is possible through these hardware interfaces by connecting senses thus leading to co-presence experiences. Various feedback channels engage individual users and integrate their expressions thus allowing co-construction of presence. The ability to mix the experienced reality at an arbitrary scale is only limited by the capabilities of the underlying infrastructure (Figure 1 right). Therefore integration and the application of open interfaces are essential for large scale collaboration. The cross-reality interaction tools research work package (Figure 1 middle) 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 supported by the underlying infrastructure.

Interaction, presence and mixed reality in urban environments are complex phenoma. In contrast to classical research on presence, the phenomena considered in IPCity have collaboration an essential property. From a technical point of view, Mixed Reality was initially described as a continuum by Milgram. Independently, Weiser examined ubiquity, which is obviously important for a project operating in urban space, such as IPCity. These considerations were always kept distinct. The recent publication of Newman et. al.[1] suggests to organize ubiquitous MR applications in a two-dimensions Milgram-Weiser continuum (Figure 2) taking the quantity and density of spatial distribution in to account. This approach is able to better represent configurations where multiple input and/or output devices are interconnected to contribute to MR-systems blurring the border to ubiquitous computing.

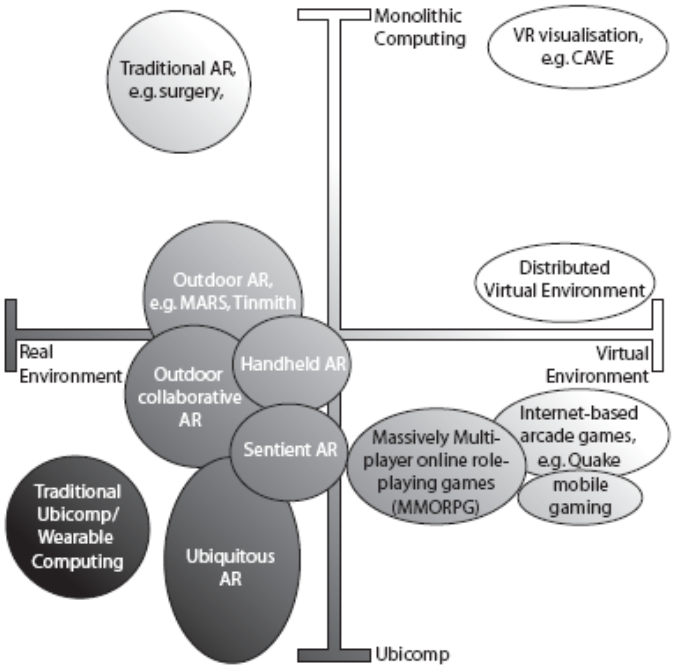

Figure 2 - Milgram-Weiser-Newman Continuum

We felt that even the Milgram-Weiser continuum is not sufficiently expressive when it comes to the representation of co-presence in MR. Among the multiple dimensions of presence that have surfaced in our and related research are at least spatial presence (e.g., perceptual immersion, sense of being there), sensory presence (perceptual realism), engagement (involvement), social presence (co-presence). Consequently Figure 3 suggests a four dimensional continuum enclosing reality, mobility, collaboration and ubiquity. A similar 3-dimensional taxonomy, covering immersion, collaboration and mobility, has been proposed by Broll [2] (2002).

---

[1] Joseph Newman, Alexander Bornik, Daniel Pustka, Florian Echtler, Manuel Huber, Dieter Schmalstieg, Gudrun Klinker: Tracking for Distributed Mixed Reality Environments Proceedings of IEEE Virtual Reality Workshop on Trends and Issues in Tracking for Virtual Environments, Charlotte NC, USA, March 2007.

[2] Broll, W.: “Collaborating in Mixed Realities”. 3D FORUM, The Journal of Three Dimensional Images Vol. 16, No. 4, (Dec. 2002): 135-140. Also in Proceedings of HC 2002 – the Fifth International Conference on Humans and Computers (Sept. 11 – Sept. 14, 2002), Aizu, Japan, 138-143.

The technology we are developing in work packages 4 and 5 represent samples or probes at interesting positions in this very complex conceptual design space. Together they represent the necessary building blocks required by Mobile Mixed Reality applications for users engaging in mixed reality co-presence.

The conceptual design space guides the development and leads to decisions made to address specific aspects out of the design space but also to provide solutions at a possibly large scale. For example, handheld devices have the potential to provide a strong mobile interface, where as stationary technologies have their strength in face to face collaboration. The combination of various types of input devices by interfacing their infrastructures closes the gap between different levels of scale and enriches the overall communication process.

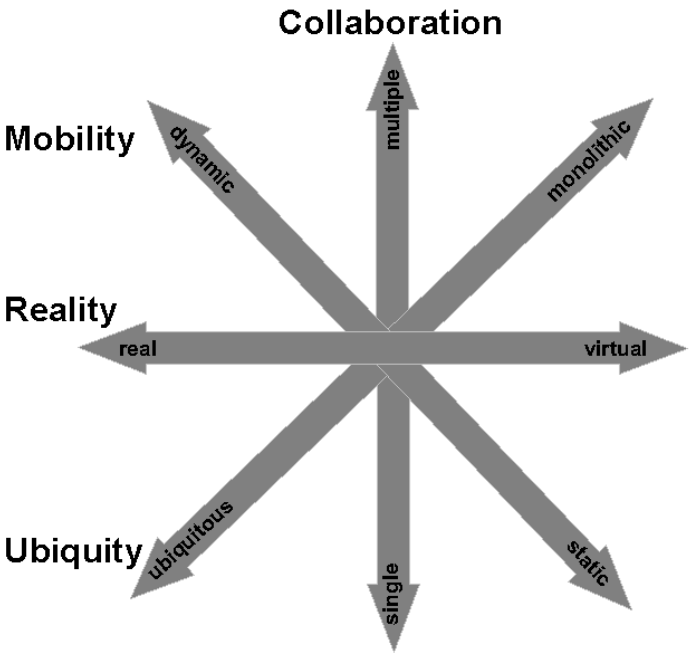

Figure 3 - Multidimensional continuum spanning a design space

The showcases – due to their differences – serve as a good cross section through the design space of Mixed Reality applications. In the spirit of high ubiquity, we favor a building block approach over monolithic solutions. The overall shared vision by the technical work packages WP 4 and WP 5, is to provide all the necessary building blocks required by modern (Mobile) Mixed Reality applications and additionally provide tools to support the design, authoring, direction and evaluation process of its content, user interfaces and interactions. The showcases – due to their differences – serve as a good cross section through the whole area of Mixed Reality applications. In order to concentrate only on those technologies, which are not unique to only one type of application scenario, we have defined requirements for technologies developed in the technical directed work packages. These requirements for all tools and services are:

- has to be required by at least two showcases,
- must actually be used by at least one showcase,
- must be flexible enough to be used in other showcases and even in other projects.

Since we do not believe that a single tool is able to provide all required aspect of a service, our developed technologies can be overlapping in regards to functionality. The application is hence able to pick the most appropriate set of technologies, which serves its requirements best.

The middle block of **Figure** 1 shows the major building blocks related to Cross-Reality interaction and authoring tools. It has been divided into the following parts:

- Authoring

- Interaction
- Ambient Displays
- Data and Event Distribution

They are visualized as four bubbles inside the middle block providing an overview on to the developed interaction and authoring tools which are needed to create an information space that lives around the users cross reality presence and are described in the next chapter in detail.

4 Cross-Reality Interaction and Authoring Building Blocks

4.1 Authoring

Authoring tools support the developers creating and administrating content and interaction for MR applications and they define one of the major building blocks. We have quite a number of different tools for different tasks of the authoring phase. Due to their diversity all of them provide unique features. *AuthOr* provides functionality for registering an application to a specific area by map augmentation. *Interaction Authoring* is supported by a graphical editor for defining interaction and application behavior. Apart from that, the HMDB can be managed and administrated using a *Web Interface* and the *Urban Sketcher* … MRIML on the other hand is a markup-language for MR user interfaces.

4.2 Interaction

Within the Interaction building block is about technologies and tools that either provide new interaction techniques or enable application developers to author, customize and evaluate such techniques. In MR applications it is very important to quickly prototype and evaluate new MR interactions, since no standard has yet been defined. The *Interaction Prototyping Tool* including its new graphical editor allows to define interactions and application behavior in a very forward and simple way. On the other hand technologies such as the *ColorTable*, the *MapLens*, the *Augmented Map Table*, and others provide very powerful customized new interaction methods, which are not only limited to a single application scenario, but can also be used for other applications. Similarly, AuthOr, the Mobile Media Creator use available digital maps in order to visualize application specific information, such as events or locations, and as such allow mobile MR applications to be registered with the application area.

4.3 Ambient Displays

The use of ambient displays becomes more and more important for modern MR applications, especially due to the increasing number of public displays. The building block Ambient Displays respects this trend. Our first effort in this area is the technology *Multi-Touch Wall*, which is used to create displays in public spaces. It allows multiple users to interact with the dynamic content.

4.4 Data and Event Distribution

Data and event distribution is another key building block for all MR applications, since these tools provide the means of distributing information between different components; additionally they provide data and device abstraction for easier access to a large variety of heterogeneous interaction devices. 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. OpenTracker and OpenVideo provide similary functionality but they are restricted – and thus more specialized – to specific device classes, namely tracking devices and video devices respectively. MMS Entrance provides access to different kinds of media.

5 Year 2 Prototypes

The year 1 initial prototypes have been re-designed, developed, tested and evaluated according to D1.5 *Detail work plan for Month 12-30* (see Figure 4). Hence the work for each prototype is described in relation to the different phase, after a general introduction the review and re-design phase is describe, followed by the development & research phase and finally the testing and public demonstration phase as well as the evaluation phase.

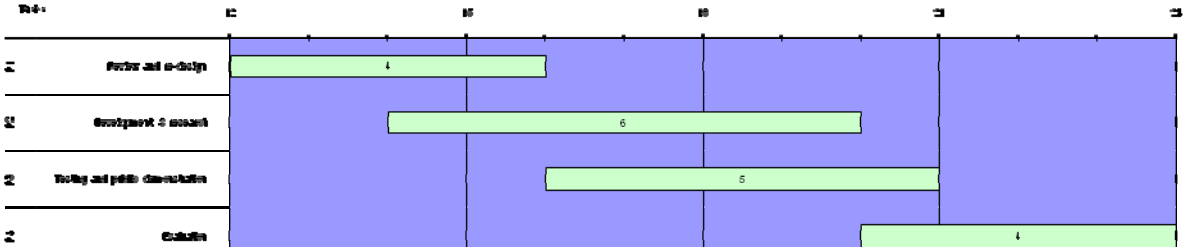

Figure 4 - Detail work plan for Month 12-24.

5.1 Re-design of interaction tools

Based on the first year early prototypes developed in the work package Cross-Reality Interaction and Authoring Tools, we received feedback from the showcases for further developments. Additionally, based on this feedback and test results and observations all tools have undergone a re-design process. As outcome of this process, some tools only need to add functionality, while others need major code restructuring for better portability, e.g. AuthOr. All design decision taken can be found in the following sections.

Apart from the re-design process we have also decided working on two new authoring and interaction tools requested by the showcases. The Multi-Touch display – a large interactive display surface – will primarily used by the Large Scale Events showcase, but is also planned for the Urban Renewal showcase demonstrators. The display is important for the showcases as it supports collocated, collaborative actions on digital, virtual objects. Another request formulated by all showcases is the need to stream audio and video data during the demonstration activities for several reasons. Since no available solution meets all the requirements, especially for Augmented Reality (AR) and Virtual Reality (VR) applications, we saw a strong need for conducting research in this area. Audio and video streaming is important for AR and VR applications in many ways, e.g. augmented video streams allow spectators to see what a user perceives in AR or the possibility to attach video streams regardless of their locality to objects in the 3D environment. We will follow two different approaches, OpenVideo and an integration of audio and video streaming into DEVAL, in order to allow all showcases the maximum amount of flexibility, while ensuring that both solutions will be able to interoperate seamlessly. This way, advantages and disadvantages can be seen much easier and we believe that both approaches will benefit from each other.

5.2 Interaction Prototyping Tool

When developing Virtual Reality (VR) and Augmented Reality (AR) applications the majority of time is spend during the interface and interaction technique development phases [Bowman2004]. There are several reasons for this:

- Firstly, a large variety of individual 3D input devices exists. Therefore, it often is not clear in advance, which input devices will be used for a particular application or task.
- Secondly, in contrast to 2D desktop environments, which rely on the WIMP metaphor, there are no standard interaction techniques which can be applied across

applications. Instead they often depend on the input devices available and the user's preferences as well as the specific application environment.

- Thirdly, many users lack experience of MR/VR tasks, devices and interaction techniques. As a result the user interface and interaction techniques often require frequent modification in order to ensure they are suitable for the tasks and are user friendly.

One possible solution is the provision of a predefined set of reasonably advanced user interface techniques, which can be accessed by an appropriate API. However such approaches often restrict the developer to exactly this predefined set of interaction techniques and modifying those or creating new ones often requires a substantial amount of programming skills.

We use a component based approach to specify interaction techniques, object behaviors, abstract devices, or even whole application prototypes. We distinguish three fundamental types of usage:

- Application prototypes, which are typically independent of a specific 3D scene and allow us to access and manipulate 3D scene data, device input and output as well as other MR system components
- Scene graph related interaction techniques and object behaviors, which are typically attached to one or several scene graph objects and thus modify their behavior in regard to user input, time, or other object. Nevertheless in our current approach those objects in fact reside outside the actual scene graph. This is realized by a standard set of API functions, each scene graph has to provide.
- Device combiners and adaptors, which are not directly related to 3D scene graphs, but rather allow for creating new virtual devices out of existing ones or modifying and simulating device input and output

While originally designed and developed to support prototyping of interaction techniques and behaviors, our approach is not limited to those and may even be used for non-3D environments.

5.2.1 Review and re-design

In year 1 of the project, a pure text-based approach was developed, allowing the specification of interaction and behavior components by a simple text file. Thus, specific user interface components and interaction techniques as typically applied by the showcases could be specified, and interpreted and executed by the realized behavior engine. While the approach proved to be quite powerful and flexible regarding its overall capabilities, it was stated that this may not be sufficient for more complex user interfaces or applications, making extensive usage of such mechanisms.

This may particularly be true when a certain degree of complexity is reached, where a text-based representation may not allow the developer to keep track of the individual components, their connections, and their interoperation. In order to overcome these limitations and to expand the original benefits of the approach to more complex MR environments, a couple of new and modified mechanisms was specified during the re-design period:

- The addition of new components allowing for simpler use of time-dependent behavior by reducing complexity
- The support for virtual timers to realize more flexible time-dependent behavior based on virtual time strands
- The addition of key-value-mapping components to allow for realization of typical 3D user interaction techniques by a rather small but powerful components

- A complete re-design of the data storage concept
- The realization of master behaviors and behavior templates and behavior libraries for easier reuse of existing realizations
- The development of a graphical programming environment for faster development and better representation of interaction prototypes

5.2.2 Development & research

Our approach combines synchronous control and data flows with asynchronous event and network distribution. It is based on a predefined yet extendable set of components representing: integral part of object behavior, user interaction, and device handling. Several of these components are combined into a so called interaction or behavior object. These objects react to event input from input devices and other similar objects, or depend on the elapsed time or are synchronized with per frame scene updates.

Communication between such objects and with remote objects, devices, or scene graphs is handled by (asynchronous) events. The objects may specify appropriate interfaces for incoming and outgoing events. We distinguish between system events used for global services such as picking or device input and output, and user defined events allowing for sending arbitrary data. System events form an event hierarchy, which allows us to accept inherited events (e.g. an interface waiting for a 3-DOF orientation event may also accept a 6-DOF event). In contrast to the external event communication mechanism, the internal data and control flow between the individual components is implemented by a (synchronous) signal/slot mechanism. Thus, data fields of the local components are connected by a dataflow graph, while their execution dependencies follow the control flow graph.

The approach is based on a text/XML-based description of the objects, the components, as well as the data and control flow.

Components for Interaction, Behavior and Devices

In our approach we currently support the following groups of components for modeling interaction techniques, object behaviors and device input and output:

- Base components
- Execution components
- Time-dependent components
- Key-value mapping components
- Device components
- Interaction components

Base components allow (1) for receiving and sending events, (2) for querying data from other system components or scene graph objects, and (3) to register for notifications on new input, object modification, or state changes.

Execution components support the evaluation of conditions, calculations, the execution of statements or even scripting.

Time-dependent components allow for the execution (1) at specific time stamps, (2) for a particular period, or (3) for a series of intervals. They support individual time bases. In addition to time strands based on the real (system) time, strands originating from the scene file loading time and the behavior definition file loading time exist. Further, virtual time bases may be defined, allowing for an arbitrary manipulation of the virtual time including its speed and direction. Thus, a speed factor of 1.0 is represents a progress similar to the real time, while a speed factor of -1.0 implies a reversed time, a speed factor of 0.5 represents a slow

motion (the virtual time is half as fast as the real time), and a factor of 2.0 simulates a fast forwarding environment. This allows for easy adaptation of the time-dependent scene behavior such as animations, etc. according to the needs of the application. An example of this is provided in section 5.

Key-value mapping components support the mapping of key values to data values. Individual components exist to support interpolations, animations, and general mappings as well as finite state machines.

Device components are optimized to be used within device combiners and adaptors, i.e. for modifying and combining data of devices independent of a particular scene graph or application. They allow for transforming, filtering and extrapolating data values.

Interaction components are two-fold: Some of them are used for mapping 2D input (e.g. from a mouse pointer) to 3D planes or lines, while others support 3D interaction mechanisms such as picking and collision detection.

Additionally, data storage components and network components have been (re-)designed, but not yet fully be developed. The data storage components, which have been re-designed completely will provide full support for storing and retrieving data temporarily (in memory) or persistently (in files). They will become available in Phase III.

Reusability

In order to reduce the effort for realizing similar interaction techniques our approach provides a couple of mechanisms to support the reusability of already defined interaction and behavior objects. These are:

- Master Behaviors
- Behavior and component templates
- Includes / modules

Defined objects may either be of global scope (e.g. when part of an application or a device adaptor/combiner) or may be defined as master behaviors. The latter are loaded into our behavior engine but will not be instantiated until actually attached to an object (e.g. a scene graph node or any other VR/AR system component). The objects it is supposed to attach to and the conditions under which a specified attachment condition is evaluated may be defined for each object individually. The objects to attach to are specified by a list of addresses, which typically will include wildcards. A typical attachment condition is for instance to re-evaluate the specified objects upon addition of new object (e.g. shapes). After re-evaluation an instance of the master behavior will be attached to those objects, which were not on the list before, while it is removed from those missing on the new list. An example of this mechanism is when a gravity behavior is supplied to all objects in a scene.

In addition to this mechanism we developed two template mechanisms: one for complete interaction and behavior objects and one for individual components. The first one allows for specifying a full object with additional parameters for initialization. Upon instantiation the parameters are then used to create the actual object. Depending on the scope used this can either be a global object or a master behavior (allowing for attachment to several objects). The individual components of a template are invisible and cannot be accessed directly within the instantiated object. A similar mechanism exists for component templates. Beside parameters, the component template mechanism also defines the data and control flow interfaces of the new template component. Those are mapped to appropriate equivalents of the underlying sub-components. This allows us to combine several components into a new component template, which then can be used similar to built-in components. This mechanism is for instance used to supply the standard picking and collision detection components, which are based on extended base components. Again the interface of the component template finally hides the original components completely from the user.

In addition to these mechanisms our approach supports a standard include mechanism, which may be used to include existing standard interaction and behavior objects or templates and objects previously defined by the author into an environment. This mechanism allows the provision of interaction modules (such as set of mouse-based manipulation techniques or support for tangible user interfaces, etc.) or 3D widgets (in combination with appropriate 3D objects such as bounding boxes – see also examples in section 5.2.4).

Graphical Editor

Realizing interaction techniques and user interfaces with the aid of interaction and behavior objects provides as major benefit in that no usual programming skills are necessary. Modeling is component-based and therefore comparatively easy to learn – especially for users who are not familiar with programming or scripting languages. Nevertheless the task of creating behavior objects remains complex.

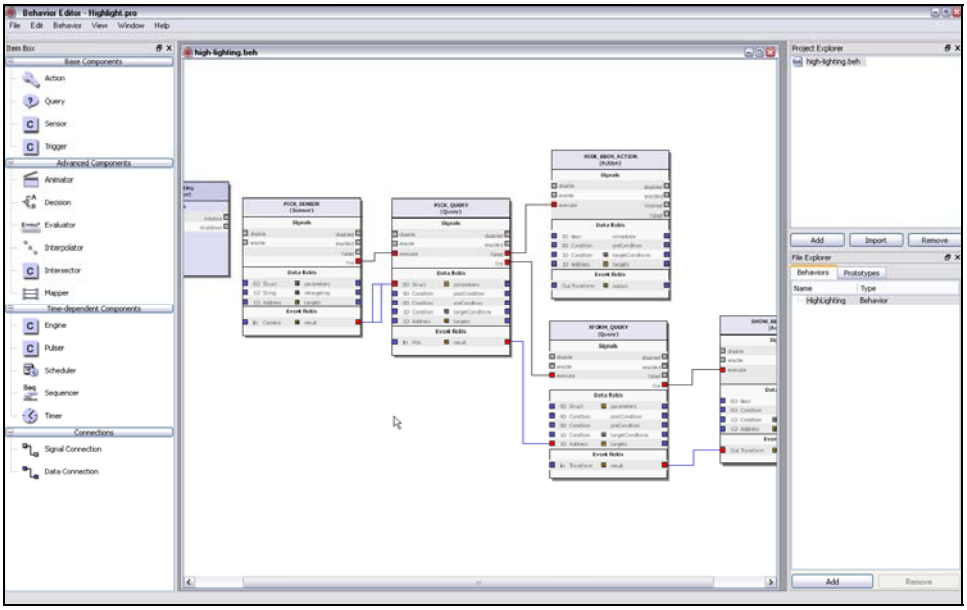

Figure 5 - Visual programming environment for modeling interaction and behavior objects

To date text / XML-based component descriptions were edited in a standard text editor. However, in large applications with lots of different and more extensive behavior objects, editing the description in this way did not provide an appropriate solution. In particular it was difficult to keep track of the entire application behavior. To simplify the creation of interaction techniques, application prototypes and object behaviors, we developed a visual programming environment that enables the user to assemble interaction and behavior objects from the set of predefined components.

The authoring tool allows the user to model behavior objects more intuitively and visualizes the whole network of components, i.e. the individual components as well as the data and control flow connecting them. This allows the user to get a good overview of the relationship between the individual components, thus allowing even for rather complex behavior objects to be understood quickly. Visual programming can also be used to speed up the development process and provides easy access to 3D user interface creation for beginners who are not familiar with programming. Another benefit of the visual programming environment is its flexibility. It can easily be extended to support new components by simply editing an XML based configuration file.

The user interface of the authoring tool is divided into different areas (see Figure 5). The main menu provides functionality for loading and saving as well as different view options. The interaction and behavior objects are saved in text / XML files and organized in XML-based projects within the authoring tool. Thus many interaction and behavior objects can be loaded at once. This is especially useful for more complex AR and VR scenarios as it makes the management of many interaction and behavior objects more convenient.

The item box on the left side contains the visual items (e.g. the available components). The visual programming environment provides a graphical representation of each available component and even supplies further information of each component on demand. In the centre of the workspace one or several composition areas can be arranged. Each composition area displays the network of components corresponding to a specific interaction or behavior object.

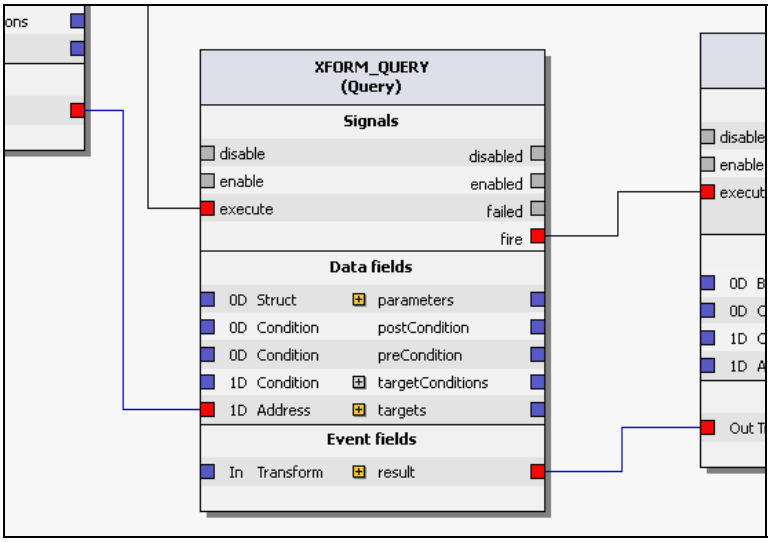

Figure 6 - Setting up the control flow and data flow connections between the individual components

The process of creating interaction techniques or object behavior by visual programming can generally be divided into four steps:

- Selection of the appropriate components
- Configuration of these components
- Specification of the data flow and the control flow
- Testing

At first the user selects the required components from the available component types. Adding a component to the composition window is carried out by dragging the component from the item box to the desired position in the composition area. In the next step the components can be configured by modifying their attributes and event interfaces. When specifying event interfaces for predefined system events (e.g. Speech, Mouse, 6-DOF-Tracker, etc.) those may directly be selected from a list. In addition the user is also supported in defining user-defined event interfaces.

The connections defining the data and control flow can be specified visually in an intuitive way by linking the components with the appropriate connection items (see Figure 6). Finally the created results can be tested by executing them within the AR/VR-environment. The steps can be repeated as often as necessary to achieve the desired effects.

The environment also supports the reusability mechanisms of the approach. New template objects can be created and are displayed within a separate area of the programming environment. The drag & drop mechanism is also used here to instantiate those templates. Additionally they may be opened for editing and modifying the individual components. The support for component templates is implemented in a similar way, but those are displayed within a special area of the component window.

Rather than implementing the visual programming environment as a separate and independent tool, which simply reads and writes interaction and behavior object description files, we decided to build the environment on top of the behavior engine. In addition to a unified parsing and description error checking system, this approach also allows for easy

checks on correct data types and type castings. Since all objects and components modeled are created and modified within the behavior engine as edited, errors are detected immediately and run-time errors are prevented. Another important reason for this implementation approach was that it will allow us to extend the visual programming environment for debugging purposes during runtime.

Implementation

The behavior engine is entirely written in C++ and has primarily been implemented for use within the AR/VR MORGAN framework. For that reason it is partially based on MORGAN support components. It is available on all MORGAN platforms including mobile devices such as PDAs and smart phones (based on Windows Mobile). The implementation however can easily be used within 3rd party software as well. We foster this by a unified interface which realizes access to and from all external components. While such components may be rather arbitrary, special support for 3D scene graphs, I/O devices and 3D viewers/applications is provided. In addition to implementing a rather small set of interface methods (e.g. for realizing the attachment of interaction and behavior objects to scene graph nodes), scene graphs have to provide a set of additional mapping tables in order to allow for unified access to 3D shapes independent of the exact scene graph structure.

5.2.3 Related Work

There has been much research into tools, toolkits, and mark-up languages for defining 3D and in particular Mixed Reality user interfaces. This research can generally be divided into the following categories:

- authoring tools – usually providing a 2D or 3D graphical authoring environment,
- component based approaches – specifying a pre-defined set of interaction components, interaction techniques, or 3D widgets,
- scene graph related approaches – extending scene graphs or scene graph description languages by nodes to realize interaction techniques, object behaviors, or scripting, and
- user interface description languages or user interface markup languages, as well as
- arbitrary combinations of the above.

In addition to the above items there are a couple of tools **for device handling and device** abstraction. These tools support some of the above functionality in order to create virtual or extended devices. One well known example is OpenTracker [Reitmayr2001].

Authoring tools provide a high-level environment for defining VR and AR applications. Examples of such tools include DART [MacIntyre2004], which is based on Macromedia Director, providing several predefined behaviors as well as support for fiducial markers and video captures. Another example is the CATOMIR Mixed Reality authoring tool [Zauner2004], which has been developed as part of the AMIRE [Abawi2004] framework. It uses 3D widgets as the main interaction components and provides a graphical authoring user interface. A major restriction is its limitation to desktop environments. In contrast to this iaTAR [Lee2004] provides an immersive authoring environment. It is based on physical and virtual props, which represent the 3D content and user interface elements. While all these approaches are based on components and allow for a rather easy creation of MR applications and user interfaces, they are rather restricted regarding the input devices supported. The interaction techniques are pre-defined by the user interface elements (components), and extending those either requires a significant effort (e.g. programming in Lingo for DART) or is even impossible without modification of the actual tool.

In contract to those approaches, our approach uses rather low-level components, which can easily be assembled and configured to support new devices and services, or to realize new interaction techniques.

Other component-based approaches using dataflow graphs as used by our approach include BodyElectric [Lanier1993], ICON [Dragicevic2001], and Unit [Olwal2004]. In contrast to the first two approaches, our approach aims to support potentially fully distributed environments. Unit allows for multiple abstraction layers of interaction techniques as well as for the distribution of units among several computers. The individual layers however are not encapsulated within individual objects - thus not allowing for a dynamic mapping or templates as used within our approach.

Several approaches exist, which are related to or closely coupled to 3D scene graphs (especially X3D/VRML). Among them are Behavior3D [Dachselt2003], YABLE [Burrows2005] and SSIML/Components [Vitzthum2006]. In contrast to these approaches, our approach while allowing for easy attaching to scene graph objects, is actually independent of a specific scene graph. Therefore, our approach additionally allows for scene graph independent device manipulation and application realization.

Another approach is the use of user interface markup or description languages. Examples include APRIL [Ledermann2005], which has been developed for the Studierstube MR system, using an XML-based description language for defining MR presentations. It allows for easy description of media objects and related user interactions and behaviors. In order to access individual devices it relies on OpenTracker [Reitmayr2001]. Another example is the UIML [Abrams1999] based CUIML [Sandor17], which is part of the DWARF MR framework. It can be transformed by XSL into VRML to realize 3D user interfaces. Finally InTml [Figueroa2002] represents another XML-based specification language for VR interfaces, which allows for defining VR applications by connecting various components. Components represent individual interaction techniques, devices, and scene objects exist.

In contrast to our approach, these description languages typically focus on the definition of the MR user interface rather than new interaction techniques. Further they are rather focused on specific domains (e.g. APRIL on presentations) or limited to specific interaction techniques (e.g. CUIML on techniques supported by VRML, InTml on pre-defined components).

While the authoring tools and even some of the description languages provide support for visual editing, visual programming of interaction techniques and behaviors is more often found in commercial products such as professional game engines (e.g. the C4 Script Editor, the Simbionic Visual Behavior Editor, etc.) or VR toolkits such as Virtools. However, those systems are typically restricted to predefined rather high-level behaviors.

Abawi, D.F., Dörner, R., Haller, M., Zauner, J. *Efficient mixed reality application development*. Proceedings of Visual Media Production, 2004. (CVMP). 1st European Conference on 15-16 March 2004 Page(s):289 – 294, 2004.

Abrams, M., Phanouriou, C., Batongbacal, A.L., Williams, S.M., and Shuster, J.E. *UIML: An Appliance-Independent XML User Interface Language*. In Proc. of the Eighth International World Wide Web Conference, Elsevier, Toronto, Canada, pp. 1695-1708, 1999.

Bowman, D., Kruijff, E., LaViola, J., and Poupyrev, I. *3D User Interfaces: Theory and Practice*, Addison-Wesley, Boston, 2004.

Burrows, T., and England, D. *YABLE—yet another behaviour language*. In Proc. of the Tenth international Conference on 3D Web Technology (Bangor, United Kingdom, March 29 - April 01, 2005). Web3D '05. ACM, New York, NY, 65-73, 2005.

Dachselt, R., and Rukzio, E. *Behavior3D: an XML-based framework for 3D graphics behavior*. In Proc. of the Eighth international Conference on 3D Web Technology (Saint Malo, France, March 09 - 12, 2003). Web3D '03. ACM, New York, NY, 101-ff, 2003.

Dragicevic, P., and Fekete, J.D. *Input Device Selection and-Interaction Configuration with ICON*. In Proc. IHM-HCI 2001. Frontiers, Lille, France, Springer Verlag. 543-448, 2001.

Figueroa, P., Green, M., and Hoover, H.J. *InTml: a description language for VR applications*. In Proc. of the Seventh international Conference on 3D Web Technology (Tempe, Arizona, USA, February 24 - 28, 2002). Web3D '02. ACM, New York, NY, 53-58, 2002.

Lanier, J., Grimaud, J.J, Harvill, Y., Lasko-Harvill, A., Blanchard, C., Oberman, M., and Teitel, M.. *Method and system for generating objects for a multi-person virtual world using data flow networks*. United States Patent 5588139, 2003.

Ledermann, F., and Schmalstieg, D. *APRIL A High-Level Framework for Creating Augmented Reality Presentations*. In Proc. of the 2005 IEEE Conference 2005 on Virtual Reality (March 12 - 16, 2005). VR. IEEE Computer Society, Washington, DC, 187-194, 2005.

Lee, G.A, Nelles, C., Billinghurst, M., and Kim, G.J. *Immersive Authoring of Tangible Augmented Reality Applications*. In Proc. of the Third IEEE and ACM international Symposium on Mixed and Augmented Reality (Ismar'04) - Volume 00 (November 02 - 05, 2004). ISMAR. IEEE Computer Society, Washington, DC, 172-181, 2004.

MacIntyre, B., Gandy, M., Dow, S., and Bolter, J.D. *DART: a toolkit for rapid design exploration of augmented reality experiences*. In Proc. of the 17th Annual ACM Symposium on User interface Software and Technology (Santa Fe, NM, USA, October 24 - 27, 2004). UIST '04. ACM, New York, NY, 197-206, 2004.

Olwal, A., and Feiner, S. *Unit: modular development of distributed interaction techniques for highly interactive user interfaces*. In Proc. of the 2nd international Conference on Computer Graphics and interactive Techniques in Australasia and South East Asia (Singapore, June 15 - 18, 2004). S. N. Spencer, Ed. GRAPHITE '04. ACM, New York, NY, 131-138, 2004.

Reitmayr, G., and Schmalstieg, D. *An open software architecture for virtual reality interaction*. In Proc. of the ACM Symposium on Virtual Reality Software and Technology (Banff, Alberta, Canada, November 15 - 17, 2001). VRST '01. ACM, New York, NY, 47-54, 2001.

Sandor, C., and Reicher, T. *CUIML: A Language for the Generation of Multimodal Human-Computer Interfaces*. In Proc. of the European UIML Conference, 2001.

Vitzthum, A. 2006. SSIML/Components: a visual language for the abstract specification of 3D components. In Proceedings of the Eleventh international Conference on 3D Web Technology (Columbia, Maryland, April 18 - 21, 2006). Web3D '06. ACM, New York, NY, 143-151.

Zauner, J, and Haller, M. *Authoring of Mixed Reality Applications, Including Multi-Marker Calibration for Mobile Device*. In Proc. of 10th Eurographics Symposium on Virtual Environments (EGVE 2004), pp. 87-90, 2004.

5.2.4 Testing and public demonstration

We tested our above presented approach on several scenarios which are directly connected to issues in VR/AR. The gaze-based selection – used by the TimeWarp showcase – is described here in detail:

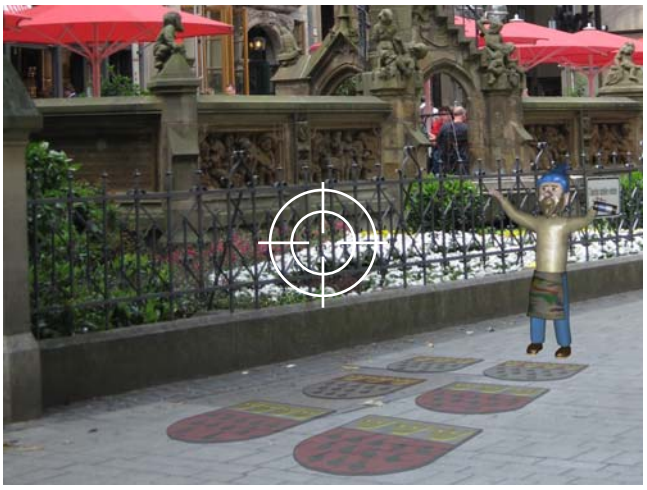

Figure 7 – Realization of a gaze-based selection mechanism using a crosshair for specifying the picking direction

One of the most important functionalities for users of VR/AR environments is the selection of 3D objects. The selection can be implemented with different techniques and input devices. One very intuitive way in AR is the use of a gaze-based mechanism. In this method a crosshair is placed in the center of the user's view. An object can then easily be selected by moving the (tracked) display (head-mounted display or handheld device) so that the crosshair covers the object to be selected. We use this method within the *TimeWarp* showcase. One of the challenges within this game-like application is that the user has the task of selecting the correct coat of arms of the city of Cologne (see Figure 7). We used our approach to realize a corresponding interaction object, which can be modeled by five components (see also Figure 8).

- A *Sensor* component for continuously receiving the current camera position and orientation from the 3D viewer.
- A *Picking* component, picking the object aimed at by the crosshair, each time new data arrives at the above Sensor (i.e. the viewpoint of the camera has changed). The picked object is then checked against a parameter provided, which identifies the id of the correct coat of arms.
- A *Query* component for querying the absolute transformation of the selected object in the 3D scene.
- An event issuing *Action* component to move a predefined highlighting box to the position of the selected coat of arms as determined by the query.
- A second *Action* component is used to hide the highlight box, if no coat of arms is picked

With the inheritance functionality the object behavior can easily be extended with additional components to provide more complex application logic e.g. for positive sound feedback if the correct coat of arms has been selected. The interaction object may also be used in arbitrary other scenarios or scene graphs as long as this kind of selection is used.

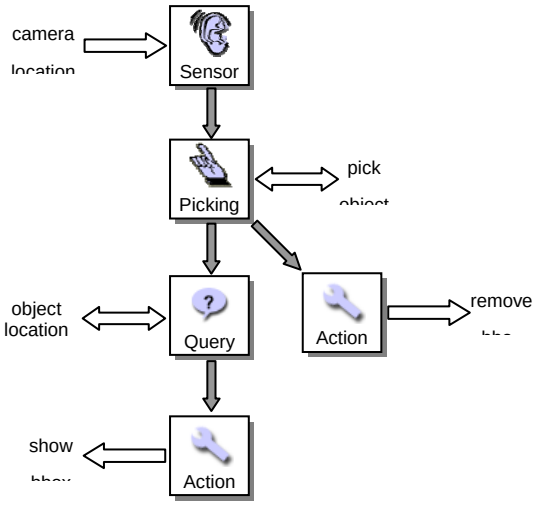

Figure 8 - Simplified view of interaction object components used for gaze-based selection including event interfaces (white arrows), and data and control flow (grey arrows)

5.2.5 Evaluation

A formal evaluation of the tool has not yet been conducted. However, the interaction prototyping tool will now be used within the TimeWarp showcase, which will include an evaluation by the developers. Further, it is anticipated to have a more formal evaluation of the tool (using the text form and/or the graphical programming environment) in comparison with a MR programming API.

5.2.6 Specification

Hardware and OS	Windows XP
Software	Morgan AR/VR Framework
Core Features	Universal tool for realizing • interaction prototypes, • MR user interfaces, • virtual devices, and • whole MR applications. Based on • universal interaction components • template mechanisms • graphical programming environment
Status	Extended prototype
Intended users	Showcase developers, MR developers, MR students
Showcases	TimeWarp (WP8), City Tales (WP9)
Relevance beyond project	The tool is intended for use beyond the scope of this project. It will be used in the EXPLOAR project funded by the EC through its Life-Long-Learning initiative.

5.3 AuthOr

AuthOr is a service to augment arbitrary maps with 2D graphical overlays, e.g. lines, images and text. Depending on the application scenario this service can be used for supporting the whole process chain of creating, authoring, orchestration and directing mobile MR applications (see Figure 9).

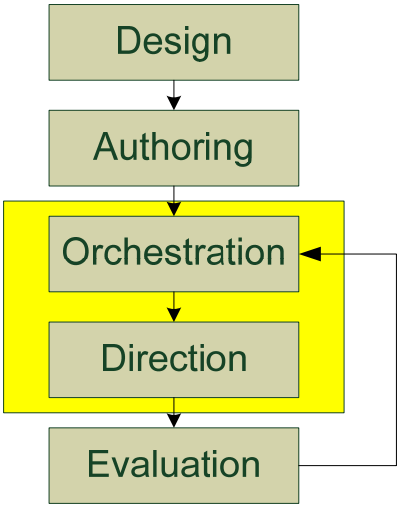

Figure 9 - The process chain of creating mobile MR applications. It is an interactive process after the design and the authoring phase, the application is tested and evaluated, afterwards, it is adapted again.

Traditional computer games do not relate in any way on the environment in which they are played, no work has to be done to setup the game or adopt it to the local environment the same applies to other MR applications using a fixed installation. This is not the case for mobile MR applications, thus the developers have to cope with new challenges. In particular, although the application design might be independent from the actual environment – the real world location – actions, items, events and virtual characters have to be closely integrated and require a strong relation to the current real environment. Such pre-runtime configuration and setup is known as orchestration and is an essential part of a mobile MR application. In case the mobile MR application has an event like character – where the users are equipped with special technology and the environment needs to be setup as well – it is also very crucial that such adaptations to the application can be performed during a session in order to influence the user interactions. Here we use the term direction for in game monitoring and adapting the user interactions.

AuthOr helps the developers to meet the challenges of orchestration and direction for mobile MR applications. Usage of AuthOr is not limited to the developers, but also can become a part of the application itself – e.g. TimeWarp – and hence support the whole process chain of creating authoring, orchestrating and directing for mobile MR applications.

5.3.1 Review and re-design

AuthOr has raised a very high interest by all showcases and will be used during phase 2 demonstrators by the showcases TimeWarp and CityTales.

Since the phase 1 demonstrator was only a prove of concept, the showcases have issued a number of requirements for further improvements and additional functionality. Most of these requirements are essential for the deployment within the phase 2 showcase demonstrators like the MIT developed by the TimeWarp showcase.

The first demonstrator of AuthOr has been developed for Windows PC and used Trolltech Qt for the user interface. For the second demonstrator we have to support additional mobile

platforms, because the MIT for TimeWarp will run on a Windows Mobile 5 PDA and the StreetBeat demonstrator of the CityTales showcase is running on a Smartphone. This requires separating the GUI (graphical user interface) from the core functionality for better portability; additionally we need to drop the usage of Qt due to the lack of portability for mobile platforms. The code needs to be rewritten in order to separate portable code – like calculating the current tiles and coordinate conversion functions – from less portable code, e.g. http downloads, image loading and displaying, user input.

Apart from the requirements for supporting PDAs and Smartphones the showcases have also requested for supporting different kinds of overlays, like images, paths, and text, and additionally to maps from Google Earth and Google Maps support for custom maps. Here a list of all requirements for AuthOr:

- PDA
- Smartphone
- Overlays
    - Images
    - Path
    - Text
- Overlay animations
- Integration into MIT
- Arbitrary custom maps

Figure 10 - AuthOr running on a Windows Mobile 5 PDA and a Smartphone

Image overlays can be used for representing users, locations, actions, or others and are either fixed in their size (in pixel dimensions) or they are fixed to GPS coordinates, i.e. they will scale together with the map. The same applies to text overlays. Path overlays on the other hand are a list of coordinates with form a line strip, the application developer may choose the appearance of the line, e.g. solid, dotted, dashed, and so on, together with the color and fill style. All overlays are dynamic, i.e. they can be created, updated and removed during runtime (see Figure 10).

There are also future plans for AuthOr, namely request for advanced features which we decide to support at a later phase. The request include animations for overlays, e.g. flashing, a 3D version for VR and AR applications and maybe in combination with height fields which are extracted from geo data services.

5.3.2 Development & research

Application orchestration and direction are major issues for mobile MR applications, since they allow for easy adaption of the application before and during the user session. Orchestration deals with all aspects of configuring and setting up the application for a certain session, while direction deals with monitoring and adapting the session while it is running. In general all aspects of the application that are configurable can be adapted during the orchestration phase and those which can change the state during a session can be adapted during the direction phase. This close relation between orchestration and direction suggests a tool that covers both aspects and is aware of the current phase.

Within our recent MR games, namely Epidemic Menace and Timewarp, we used a set of supporting tools for orchestration and direction.

Figure 11 - The orchestration board of the game Epidemic Menace.

Within Epidemic Menace, a pervasive AR game, we developed an orchestration board [Lindt et al 2006]. It uses a stylized map of the game area and provides functionality for configuring the teams, placing viruses, assigning devices to individual players (see Figure 11). The initial configuration of a session can be saved and used in the game engine. It was also used during the actual game play for receiving information about the current location of the players and the viruses and displaying this information on the map. The orchestration functionality was also enabled during the sessions allowing the game supervisors to add/remove players, place viruses and assign devices to the players, making the game flow adjustable. Due to the monitoring functionality of the board, it was additionally used within the game as an overview map for the teams, where they could observe the outdoor players.

For the Mobile MR Game Timewarp, we redesigned and reimplemented the orchestration board in order to meet the requirements stated above. While in Epidemic Menace a relocation of the game to a different location would have meant a great effort, AuthOr is much more general. One of the key features of AuthOr is hence augmentation of arbitrary maps with overlays (images, lines, text, …), e.g. for user positions, paths, items, events and so on. An interface to map services such as Google Maps, Google Earth and Virtual Earth allows playing the game everywhere in the world. Therefore, it is possible to register and display virtual objects and actions in the real environment. Further, the effort for adapting a session to a new location becomes reasonable. If used as an orchestration tool, the area can be defined, and arbitrary items can be placed. During a session it is used by the staff to supervise the user interactions and relocate, add or delete items, additionally it is provided to the users as a map tool, where they can see their current location on the map and interesting spots (see Figure 12). As the area becomes larger, zooming functionalities of the map are

essential in order to get an overview of the game, or a close up look for more detailed information. For efficient zooming support the maps can be specified for different levels

5.3.3 Related Work

Although MR Games have been developed for a couple of years, there are a lot of examples where the game does not relate to the environment or it has been set up for a specific environment. An example for environment unrelated games is *ARQuake* [Piekarski2002] and *ARPacman* [Cheok2004] for a fixed location game.

The mobile SMS game *Day of the Figurines* the staff managing and leading the game, used orchestration in order to manage and track the production of game play narratives during the game session. They state that orchestration is inseparably intertwined with the game play [Cabtree2007].

Epidemic Menace [Lindt2006], a cross media pervasive AR game, used orchestration tools for pre-game setup and in-game adaptations. In this game two teams have to fight a spreading viral menace. While the outdoor players have to fight the virus, the indoor players have the overview of the game are and communicate that to the outdoor players.

A good overview about the technology challenges of pervasive augmented reality games is given by Broll et al. [Broll2006], where the different aspects are closely reviewed and solutions are presented.

Broll, W., Ohlenburg, J, Lindt, I., Herbst, I, and Braun, A.-K. *Meeting Technology Challenges of Pervasive Augmented Reality Games*. In Proc. of ACM SIGCOMM NetGames 2006, 2006.

Cabtree, A., Benford, S., Capra, M., Flintham, M., Drodz, A., Tandavanitj, N., Adams, M., and Farr, J.R. *The Cooperative Work of Gaming: Orchestrating a Mobile SMS Game*. In Computer Supported Cooperative Work, 16(1-2), 2007.

Cheok, A., Goh, K. H., Liu, W., Farbiz, F., Fong, S. W., Teo, S. L., Li, Y., and Yang, X. *Human pacman: a mobile wide-area entertainment system based on physical, social, and ubiquitous computing*. Personal Ubiquitous Computing, 8(2), 2004.

Lindt, I., Ohlenburg, J., Pankoke-Babatz, U., and Ghellal, S. *A Report on the Crossmedia Game Epidemic Menace*. ACM Computers in Entertainment (CIE), 5(1), 2006.

Piekarski, W. and Thomas, B. *ARQuake: the outdoor augmented reality gaming system*. Communications of ACM, 45(1), 2002.

5.3.4 Testing and public demonstration

Figure 12 - The Mobile Information Terminal of TimeWarp. The map on the right shows the current location of the player and the screen on the left important game items.

Author has already been integrated into the showcase TimeWarp as part of the Mobile Information Terminal (MIT) (see Figure 12). The users can switch to the map on their PDA to see their current location in the city of Cologne and also the locations of the time portals.

Both the location of the user and the locations of the time portals are realized using image overlays. The MIT regularly receives the updated GPS location of the user and updates the locations of the user icon on the map and centers the map over the user.

The integration into an application like the MIT demonstrates how easy it can be used and why it is important to separate the core functionality and the GUI functionality.

The MIT was part of the evaluation and testing activities of TimeWarp.

Apart from that, AuthOr has been presented at the PEACH Summer School.

5.3.5 Evaluation

Evaluation of the MIT was more focused on presence aspects and no user studies have been conducted regarding the AuthOr part of the MIT.

5.3.6 Specification

Hardware and OS	Windows XP, Windows Mobile 5, Windows Mobile 6
Software	C/C++

Core Features	• Access to Google Earth tiles
	• Access to Google Maps tiles
	• Access to Virtual Earth tiles
	• Hybrid mode
	• Overlay support
	  o Lines
	  o Images
	• Separation of core functionality and GUI and OS specific functionality

Status	Stable prototype, which will be further developed.
Intended users	Showcases 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 outside of the project.

5.4 Mixed Reality Interface Modeling Language (MRIML)

MRIML (Mixed Reality Interface Markup Language) is an UIDL that allows non-developers to describe an interactive application with different types of interaction techniques, modalities of use and computing platforms. But MRIML cannot be rendered nor executed by its own, a rendering engine is required.

The showcase TimeWarp has used MRIML during phase 2 and provided feedback to improve the current specification.

5.4.1 Review and re-design

One of the objectives when designing the showcase TimeWarp was to provide an easy way to specify the content and interaction scheme for both systems (PDA based and mobile AR systems). One reason for this was that the map content on the PDA had to be synchronized with the location of time portals and markets which in turn had to result in a virtual objects

being displayed at the correct time. Instead of creating application and platform specific interfaces, we extended MRIML to specify the interface. MRIML is based on XML and comes with a large number of existing high-quality tools for editing and validating. Moreover, the format is self-documenting, human- and machine-readable, and elements can be addressed using methods such as XPath. In contrast to WIMP based user interface description languages MRIML fully supports three-dimensional content. UI elements are located in 3D space and can be of arbitrary media types. Element *binding types* allow us to place them relative to another element or view-stabilized (i.e. similar to a head-up display). Possible binding types supported by MRIML are:

- element related, i.e. the position is relative to parent's position
- world related, i.e. an absolute position in the real world
- view related, i.e. the element is always in the field of view independent from current player's position

The connection between game engine events or user input with the user interface is described with the MRIML element *Listener*. Listeners describe conditional signal-slot mechanisms, allowing for the specification of the game logic as well as the user interface logic. Listeners can react on game engine events such as proximity, data changes, or timeouts, and on user input events triggered by the individual input devices. The signals result in an action provided by the back-end (game engine or UI renderer) if the specified conditions are matched. An additional requested *Listener* was a *ProximityListener* to react on physical proximity. MRIML specifies now following *Listener* types:

- Focus Listener - triggers an event if parent is in focus, e.g. on mouse rollover
- Choice Listener - triggers an event if parent is chosen, e.g. on button pressed
- Text Listener - triggers an event if parent text changed, e.g. in TextInput
- Pose Listener - triggers an event if parent pose changed, e.g. via tracking
- Variable Listener - triggers an event if parent value changed, e.g. via tracking
- Proximity Listener - triggers an event if parent is close to 3D position, e.g. via tracking

Modeling generic user input events is quite difficult as mobile outdoor AR applications lack a standard input device. In TimeWarp pointers and buttons are supported. The device abstraction layer of Morgan allows for the easy exchange of the actual device.

Conditions may be expressed using XSLT/XQuery statements. When modifying the overall game state by user interaction, e.g. buying a tool at a market or picking up an item, this first results in a change of the local player data, which will then modify the displayed content of the local systems (e.g. highlighting of an item or playing a sound file).

5.4.2 Development & research

Within TimeWarp, an MRIML Renderer was implemented to demonstrate the capabilities of MRIML.

The MRIML Renderer parses and interprets an MRIML file. Depending on the type an MRIML element or attribute is mapped to a RSG or XSG node or field, or a binding of functions is realized. The current implementation supports different I/O devices such a pointing devices (viewpointer, mouse pointer) or discrete input devices like buttons, different types of media (X3D, wav, mp3, jpg) and a generic function binding.

5.4.3 Related Work

UIML – the User Interface Markup Language [Abrams1999] is an XML-compliant user interface description language, providing a rather universal interchange format, independent of specific modalities. As a meta-language it may be used to describe any type of user interface. To define a concrete user interface, a vocabulary of available elements must be specified. This approach was followed for the first realization of MRIML whereas its current implementation is independent of UIML.

CUIML [Sandor2001] is another UIML based user interface description language, which has been developed as part of the DWARF AR framework. It uses XSL transformations to convert user interface descriptions for instance into VRML or HTML. This mechanism is very similar to the one used in MRIML.

InTML [Figueroa2002] is another XML-based description language, supporting the definition of VR applications independent of particular environments. A component-based approach is used, where input/ouput devices, scene graph objects, and interaction techniques each may be represented by individual components. An application is then specified by interconnecting several components.

APRIL [Ledermann2005] finally, again is an XML-compliant description language supporting the authoring of AR applications. The approach introduces media objects and describes their interactions and behaviors. A hardware abstraction layer additionally allows the description of input and output devices. A specific virtual object may be rendered using different media according to the current runtime environment. APRIL has been developed as part of the Studierstube AR system.

Abrams, M. *Device-Independent Authoring with UIML*. In Proc. of the W3C Workshop on Web Device Independent Authoring, 1999.

Figueroa, P., Green, M., and Hoover, H.J. *InTml: a description language for VR applications*. In Proc. of the 7th International Conference on 3D Web Technology, Tempe, Arizona, USA pp. 53-58, 2002.

Ledermann, F., Schmalstieg, D., *APRIL: A High-level Framework for Creating Augmented Reality Presentations*. In Proc. of IEEE Virtual Reality 2005, 2005

Sandor, C. and Reicher, T. *CUIML: A Language for the Generation of Multimodal Human-Computer Interfaces*. In Proc. of the European UIML Conference, 2001.

5.4.4 Testing and public demonstration

MRIML was used in the showcase TimeWarp to configure the game content and interaction.

The first specification draft was developed in the project CONNECT (FP6-2002-IST-1-507844).

5.4.5 Evaluation

Evaluation of MRIML was focused on applicability and extensibility in the showcase TimeWarp and no formal evaluation of the language have been conducted.

5.4.6 Specification

Hardware and OS	Windows XP, Windows Mobile 5, Windows Mobile 6
Software	C/C++
Core Features	• Specification of MR UIs
Status	Specification Release 0.4
Intended users	Showcases and tools developer

Showcases	WP 8
Relevance beyond project	The specification is developed as a general one, that can be used outside of the project.

5.5 OpenTracker

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 (see also D4.1). Newman et al (2007) have integrated and extended OpenTracker with respect to ubiquitous tracking abilities.

5.5.1 Redesign decisions

During the second development phase we will improve previously implemented functionality and develop new OpenTracker features in order to achieve progress based on the redesign decisions. The needs for changes in the already implemented modules are stated below, followed by new parts to be implemented.

- **Camera Control – Zoom and Pan Tilt Unit**

The Zoom and Pan Tilt Unit (PTU) node needs additional functionality so calibration as well as dynamic relocating of the device can be done easily. Furthermore, a combination of an optical tracker and the PTU will be needed to allow on one hand for flexible movements of the camera within the tracked space while on the other hand to ensure an accurate acquisition of the current camera orientation.

- **Space Device and GoGo Interaction Modules**

Tests of this tool found that an automatic repositioning of the 3D pointer representation is needed to relocate the curser if lost by the user. Once the cursor is lost, meaning out of visual range it needs to be relocated in the current users view. This could be achieved by simply pressing a button on the Space Device.

- **Sys Mouse Sink**

To use the system mouse simultaneously with opentracker's 'SysMouseSink' and extension to the current sink's implementation is necessary. Furthermore the SysMouseSink should be extended so it can be used in setups using multiple displays.

- **Foot Pedal Interaction**

To allow for an interaction with the application during the usage of other tools a hands free interaction metaphor is required. An obvious solution is the usage of a Foot Pedal as Interaction device. Such a device should allow the user to trigger an action by stepping on the pedal. This device will e.g. be associated with a *clutch* mechanism for bringing selected, far away objects close to the user in the UrbanSketcher application. The original location of the object would be restored if a second trigger action is invoked by the user.

5.5.2 Development & research

- **Space Device and GoGo Interaction Modules**

An automatic repositioning of the 3D pointer representation was implemented to relocate the curser if lost by the user. Once the cursor is lost, meaning out of visual range it is relocated in the current users view on user request. This can be achieved by simply pressing a button on the Space Device or sending an appropriate OpenTracker event.

- **Sys Mouse Sink**

For the use of the system mouse simultaneously with opentracker's 'SysMouseSink' the node was extended. Furthermore the SysMouseSink can be used in setups using multiple displays now and has the capability to reposition and rescale the interaction region.

- **Foot Pedal Interaction**

Interaction with the application during the usage of other tools a hands free interaction metaphor was implemented. The obvious solution is the usage of a Foot Pedal as Interaction device. Such a device allows the user to trigger an action by stepping on the pedal. This device is associated with a *clutch* mechanism for bringing selected, far away objects close to the user in the Urban Sketcher application. The original location of the object is restored if a second trigger action is invoked by the user.

5.5.3 Related Work

The previous publications of Reimayer et al [Reitmayr2001] and Spiczak et al [Spiczak2007] outline the related work in the context of OpenTracker also mentioned in D4.1.

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.

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), 2007.

5.5.4 Testing and public demonstration

OpenTracker and especially the further developed nodes have been tested as part of the technology probes developed in WP6 in the Urban Sketcher Application. These aim to support groups of urban planners and diverse stakeholders in collaboratively envisioning urban change, using a set of mixed-reality technologies.

The different technology probes have been tested and demonstrated in 2 different workshops during phase 2:

- The hospital Sainte-Anne in Paris; 19th-21st of March 2007
- The planning of a new court house (TGI) close to the Seine and the Bibliothèque Nationale de France (BNF) in Paris; 17th – 19th of September 2007
- Demonstration within the scope of the exhibition "Draussen in der Stadt" in Vienna, 4th of October 2007.
- OpenLabNight, Graz, October 2007
- Demostration at ISMAR 2007 the 6th International Symposium on Mixed and Augmented Reality in Nara, Japan, 13th-16th of November 2007

5.5.5 Evaluation

Evaluation was done in the showcases especially WP6 within the UrbanSketcher Application.

5.5.6 Specification

Hardware and OS	Windows XP, Linux, Mac OS X
Software	▪ OpenTracker ▪ Studierstube AR/MR Framwork
Core Features	Data and event distribution
Status	Stable prototype which will be developed

	further
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, WP7 & WP9
Relevance beyond project	Can be used in appropriate context due to generic design

5.6 OpenVideo

OpenVideo is a general data integration and data processing library 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. To be able to easily support the requests of all the showcases, OpenVideo was designed to be as extensible and easily configurable as possible. It was furthermore decided to implement OpenVideo on the three main operating systems, namely Windows,Linux and Mac OS X.

5.6.1 Redesign decisions

In IPCity's first year demonstrator, OpenVideo was used to handle video information in the UrbanSketcher, ColorTable, ARScouting, Augmented Map Table and 3DReconstruction application.

Within the UrbanSketcher and the ColorTable application, the functionality of switching between viewpoints was carried out as an important feature to 'teleport' the user to arbitrary dynamic locations. OpenVideo, therefore needed an extension to offer this functionality of selecting its data 'online' from a list of available sources.

To switch between multiple dynamic video data, these applications of the urban renewal showcase requested also the ability to retrieve data from multiple ARScouts, which can be arbitrarily distributed in a sometimes even unknown network. This furthermore implies an extension to OpenVideo to offer the functionalities to send and receive multiple video streams even over a network.

To setup synthetic scenarios the showcases moreover requested the functionalities to display static background images as well as pre-captured video files. To meet these demands, OpenVideo needed to be extended to read dynamic data from avi sources as well as static image data from jpg, png and bmp file.

5.6.2 Development & research

OpenVideo implements a general dataflow system for video data based on a pipes-and-filters architecture [Buschmann1996] in a similar way OpenTracker [Reitmayr2001] is handling tracking data. However, in contrast to trackers and conventional sensors, video sources consume very high bandwidth and consequently require specific techniques to avoid undesirable behaviour such as frame drops or stalling. Existing video libraries such as Microsoft DirectShow [Pesce2003] achieve such efficiency by platform specific optimizations. Consequently, OpenVideo was designed as a wrapper library encapsulating efficient single-purpose video libraries and add a consistent scriptable interface as well as missing features such as network streaming, simultaneous delivery of video to multiple application objects, and a notification service for components interested in video updates.

OpenVideo encapsulates raw video data in video objects, which reside in shared memory and are passed around by reference. OpenVideo's functionality is therefore split into a core part, which is responsible for setting up and processing the runtime environment, and video objects. Using video objects have a number of advantages over a fixed frame structure. They can be annotated with arbitrary meta data, allowing simple and experimental multi-modal

data structures. For example, adding resolution and format description allow a receiver to deal with a variety of sources. A tracked camera or image probe can deliver a video stream where each frame is annotated with the corresponding pose information from the attached tracker. The video object also implements multi-buffer support with a buffer locking mechanism, which enables us to smoothly integrate and serve the video data from asynchronous threads. Using OpenVideo's multimodal capabilities it was possible to annotate different video streams with information about its source. Using this information a receiver is able to select during runtime its current data from the available pool of video streams which are supplied by OpenVideo.

OpenVideo now implements support for video streams which are delivered over a network. Therefore a video streaming server as well as a video streaming client was implemented and integrated into OpenVideo.

Next to live camera- or network sources, OpenVideo was extended to support data from video files as well as still images from jpg files. To easily extend the set of supported data in a consistent and backwards compatible way, OpenVideo's core classes have been implement based on the *Abstract Factory Pattern* [Buschmann1996] (see Figure 13) which provides a robust interface between the core architecture and new data dependent video nodes.

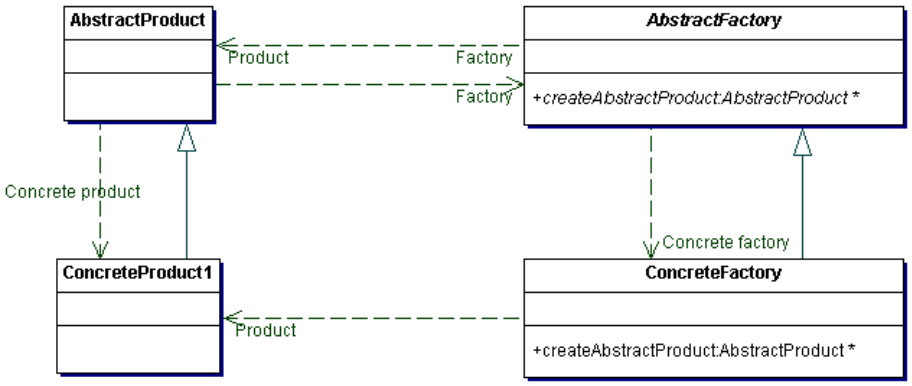

Figure 13 – Abstract Factory Pattern [Buschmann1996]

Different sources of image data require different strategies to retrieve its data. By now supporting a number of different kinds of video and image sources, a set of configurable graph traversal strategies needed to be developed. For example, while video data needs to be processed either triggered by the device driver or by using a predefined device depended update rate, an image source needs to be read into memory only at start time.

The OpenVideo core provides basic functionality to construct and update a directed acyclic graph data structure. OpenVideo now supports three different strategies to trigger an update of its data structure. First, a timer with a pre-defined update rate can be used to guarantee updates in a controllable and – depending on the real time capabilities of the operating system – in an almost equidistant time frame. Second, for a dedicated machine running OpenVideo, we have implemented support for using all available system resources to trigger updates of the graphical data structure. Finally, OpenVideo can be configured to wait for external signals to trigger its dataflow graph traversal.

5.6.3 Related Work

Video processing functionality is supported by a couple of software projects like VideoWrapper[3], GStreamer[4] or Directshow [Pesce2003]. Due to its main requirement, the capability of highly efficient video data handling, most of these libraries are implemented using platform specific optimizations which results in platform specific libraries. To be able to use the already existing optimizations on even multiple operating systems, we designed OpenVideo as a wrapper of existing libraries. By wrapping of external libraries into OpenVideo's nodes, we are able to selectively use a library's strength while at the same time ignoring its shortcomings.

Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., Stal, M. *Pattern-Oriented Software Architecture - A System Of Patterns*. John Wiley & Sons, Chichester, 1996.

Reitmayr, G., and Schmalstieg, D. *An open software architecture for virtual reality interaction*. In *VRST '01: Proc. of the ACM symposium on Virtual Reality software and technology*, pages 47–54. ACM Press, 2001.

Pesce, M. *Programming Directshow for Digital Video*, Microsoft Press Books, 2003

5.6.4 Testing and public demonstration

- Urban Renewal Workshop, Paris, March 2007
- Urban Renewal Workshop, Paris, September 2007
- Workshop "Draussen in der Stadt", Vienna, October 2007
- OpenLabNight, Graz, October 2007
- Significant Advances in Computer Science (SACS), Graz, November 2007
- International Symposium on Mixed and Augmented Reality (ISMAR), Nara Japan, November 2

5.6.5 Specification

Hardware and OS	Windows XP, Linux, Mac OS X
Software	OpenVideo
Core Features	Video image integration & distribution
Status	Stable prototype
Intended users	Showcase and tool 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, WP7 & WP9
Relevance beyond project	Can be reused in appropriate context due to generic design Contributes to MR research community

[3] http://videowrapper.sourceforge.net/

[44] http://gstreamer.freedesktop.org/

5.7 ColorTable

The main ambition of the ColorTable 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 shapes and sizes that may be placed and manipulated on the table. The users may thus stand around the table and interact simultaneously from different positions, as it is the case in conventional discussion situations.

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

The tangible table is augmented by a projection showing the momentary settings and tracking data. It provides directly information to the user about what is measured by the system, and augments the different tangible elements of the user interface by additional, digital information telling the user what they represent.

The system of the ColorTable includes different tangible interaction modules that may be used independently. The original modules where the colored objects interface, the barcode interface and the rotating viewpoint. In year 2, those modules have been ameliorated and expanded by a tangible selector, an info screen and a sketching interface.

5.7.1 Redesign decisions

Based on the feedback and evaluation provided on 4 workshops in year 1, and on 4 workshops in year 2, we took a number of redesign decisions concerning the different interaction modules of the ColorTable.

Tangible selector

We resolved users' problems with the configuration areas by introducing a separate workplace for selecting objects. The *tangible selector* (see Figure 14) consists of several disks, on which all the available color objects are represented as flat illustrations. When users want to select an object, they take the corresponding disk, put it onto a small rod, next to which a barcode reader has been mounted, and turn the disk until the right object is selected. RFID transponders are used to tag the different positions of the illustrations on the disk. When a certain object is selected, the RFID is captured by the reader and used as input for the *Input interpreter,* a component responsible for the communication between RFID and Barcode based interfaces and the Muddleware server.

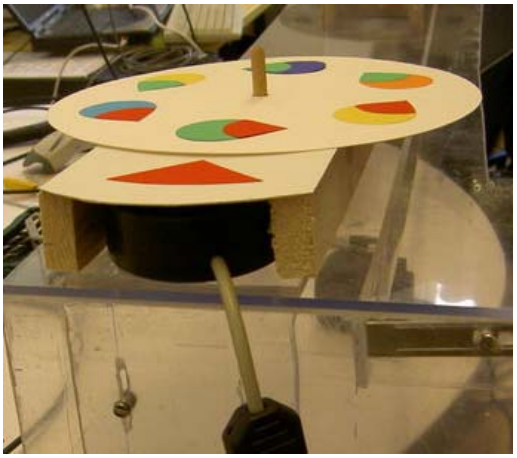

Figure 14: Selecting objects with the tangible selector

Info screen

We also decided to no longer project object attributes onto the table surface but to use a separate monitor as an info screen, on which users can see settings of selected objects. The info screen (see Figure 15) also gives feedback on which object currently is selected by the *tangible selector*.

Figure 15: The info screen shows additional information for the different objects.

Sketching interface

We developed a new tool allowing users to add their own paper sketches to the Hypermedia Database (see Figure 16). It includes a surface, where users can sketch with a conventional pen on a sheet of paper, and a web cam, placed above the surface, capturing the content. The tool automatically crops the background and stores the result in the Hypermedia Database in order to be available for other applications such as the barcode interface.

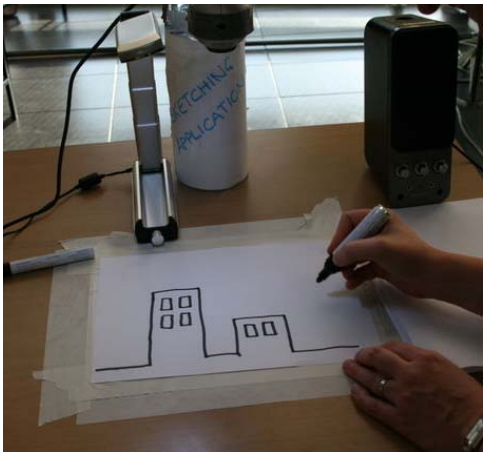

Figure 16: Adding paper sketcher to the HMDB with the sketching interface.

Rotating viewpoint

We decided to redesign the rotating color table and to separate the rotating infrastructure from the colored objects interface. A small turn-tilt plate is tracked and users can modify the orientation of the viewpoint by rotating the plate.

Workspace organization

The high number of new interaction modules of the ColorTable stressed the importance of organizing the workspace. All the material and devices needed should be within reach but not in the way. The size and shape of the table are important aspects to enforce or encourage collaboration. For this purpose, an additional workspace has been developed,

assembling two tangible selectors, two barcode readers, a rotating viewpoint, space for putting the colored tokens and the barcode sheets (see Figure 17). The two tangible selectors can each be used simultaneously along with one of the barcode readers. Input from the barcode application is assigned to the object currently selected by the respective tangible selector.

Figure 17: Experimenting on solutions for organizing the workspace

Redesigning the color objects

Tokens Version 1

The aim was to design and experiment with different color tokens (see Figure 18). The objects are smaller than the ones used for the 2nd Workshop in Paris. The objects are covered with full color paper, made out of cardboard with backfill inside to give them some weight.

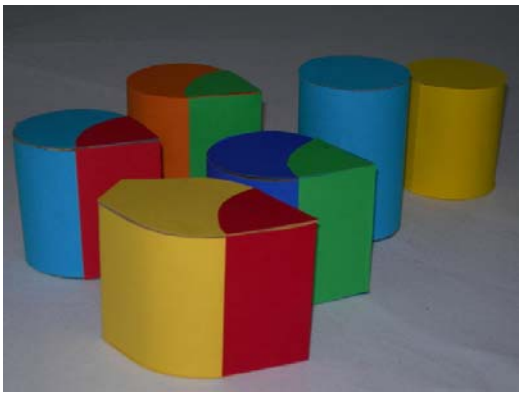

Figure 18: Experimenting with new formed color tokens

Additionally the new tokens provide the possibility to aim a specific direction and to see that within the form of the token itself. Therefore the objects are rotatable and provide means to try out a new way of interaction.

Tokens Version 2

New formal and material design issues are the base for the redesign of the tokens in version 2 (see Figure 19). The tokens are formally splitted into the graspable object with a haptic, attractive material (for these experiments we tried out wood) and the tracking color. The

tracking color itself is attached to and represents a specific content. Generally the size of the tokens is reduced comparing to all former tokens. The use of two basic geometric object forms (circular, rectangular) and the difference of token size and volumes are integrated in these experiments.

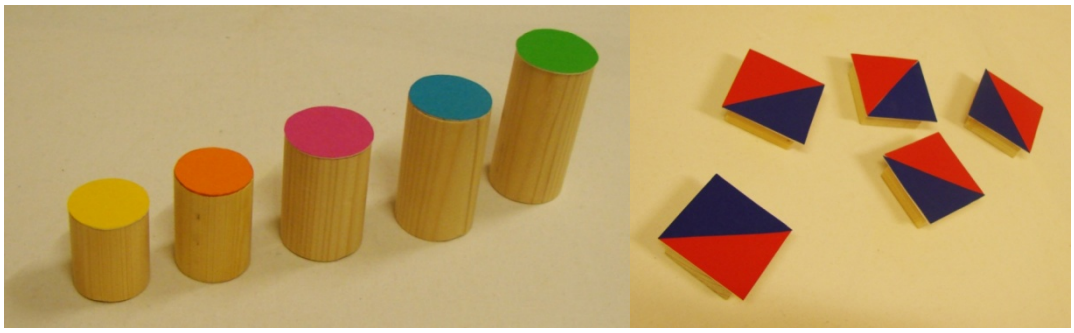

Figure 19: Experimenting with wooden tokens – (a) volume representing and (b) rectangles for different content

The tracking color gives implications for the tracking technology as additional information for the user. The forms represent variable content and are coded with the corresponding tracking color.

5.7.2 Development & research

By developing the ColorTable we try to clarify design issues related to tangible user interfaces:

- How to support users in the collaborative creation of mixed-reality configurations?
- How to make use of material and spatial properties in designing both, physical interface, as well as multiple and simultaneous interactions?
- How to develop and apply methods allowing users to rapidly learn, use and understand the interactions with the ColorTable?
- How to gain a basic understanding of the user's presence related to interaction?

5.7.3 Related Work

Quite a number of tangible tabletop systems deal with design issues related to tangible user interfaces and also report on the learning provided through engaging with users. The design of the *Envisionment and Discovery Collaborary (EDC)* is based on participatory design efforts with the use of physical simulations applied to community design with specific neighborhoods, which have been described in a series of publications. Arias et al. [Arias2000] emphasize how they have gained critical insights in the manipulation of physical objects and the need to support collaboration in these joint design sessions. Advanced prototypes of the *EDC* were evaluated e.g. by Eden et al. [Eden2002], with additional and detailed insights concerning its features.

The concept of *Tangible Tiles* [Waldner2006] has been evaluated in a series of user studies with a focus on learning how users perform relatively simple collaborative tasks involving digital imagery. The idea was to compare the tangible tiles system with a commercial touch screen and real paper prints. The *Luminous table* [Ishii2002] was developed as an extension of the *Urp* software, a system that supports a number of basic urban planning functions. The authors report on their observations of how the system was used in an urban design class and what design improvements they suggest for the future. *Tangible Viewpoints* [Malazek2002] is an interface for multimedia storytelling. The system has been used in different storytelling projects, and further development decisions have been taken based on user feedback.

While these research groups provide some insight into the process of developing a tangible user interface, the work on the ColorTable provides means to extensively deal with the

different design issues related to tangible user interfaces. The ongoing, iterative process of design-evaluation-feedback-redesign in a series of workshops help us to understand, discuss and improve problems and aspects of tangible interactions.

Arias, E., Eden, H., Fischer, G., Gorman, A., Scharff, E. *Transcending the Individual Human Mind – Creating Shared Understanding through Collaborative Design*. ACM Transactions on Computer-Human Interaction, Vol.7, No.1, pp.84-113, March, 2000.

Eden H., Hornecker E., Scharff, E. *Multilevel Design and Role Play: Experiences in Assessing Support for Neighborhood Participation in Design*. In: Proc. of DIS´02 (Designing Interactive Systems). London. ACM. pp. 387-392, 2002.

Ishii, H., Ben-Joseph, E., Underkoffler, J., Yeung, L., Chak, D., Kanji, Z., and Piper, B. *Augmented Urban Planning Workbench: Overlaying Drawings, Physical Models and Digital Simulation*. In Proc. of Ismar'02. ISMAR. IEEE Computer Society, Washington, DC, 203, 2002.

Mazalek, A., Davenport, G., and Ishii, H. *Tangible viewpoints: a physical approach to multimedia stories*. In Proc. of the Tenth ACM international Conference on Multimedia, Dec 01 - 06, 2002, Juan-les-Pins, France, MULTIMEDIA '02. ACM Press, New York, NY, 153-160, 2002.

Waldner, M., Hauber, J., Zauner, J., Haller, M., and Billinghurst, M. *Tangible tiles: design and evaluation of a tangible user interface in a collaborative tabletop setup*. In Proc. of OZCHI '06, vol. 206. ACM Press, New York, NY, 151-158, 2006.

5.7.4 Testing and public demonstration

The different interaction modules of the ColorTable have been tested as part of the technology probes developed in WP6. These aim to support groups of urban planners and diverse stakeholders in collaboratively envisioning urban change, using a set of mixed-reality technologies.

The different technology probes have been tested and demonstrated in 3 different workshops during phase 2:

- The psychiatric hospital Sainte-Anne in Paris; 19th-21st of March 2007
- The planning of a new court house (TGI) close to the Seine and the Bibliothèque Nationale de France (BNF) in Paris; 17th – 19th of September 2007
- Experimentation with simple rules commonly used in urban planning in Vienna, 13th of December 2007.

The ColorTable was also demonstrated within the scope of the exhibition “Draussen in der Stadt” in Vienna, 4th of October 2007.

5.7.5 Evaluation

The different workshop sessions were video-recorded, and transcripts of significant episodes were produced. We used several digital cameras to capture interesting situations and included saved images of visual scenes in our analysis. The details of participants’ interactions have been analyzed collaboratively in the team. After each workshop ideas for re-design have been discussed and implemented.

5.7.6 Specification

Hardware and OS	4 Dell Laptops (IP M Processor, 2.13 Ghz) Windows XP

Software	• JAVA 1.6 • JMF 2.1.1e • Muddleware • Hyper Media Database • Apache Tomcat 4.1 • MySQL 4.0.13 • OpenTracker 2.0 • OpenCV beta 5
Core Features	• tangible Interaction through manipulation of colored objects • barcodes to activate commands and selecting content • rotating viewpoint for navigation • tangible selector for selecting colored objects • info screen to visualize the settings of each object
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	

5.8 Mobile Media Collector

Architects and designers work frequently in the field. Field work may be related to starting a new design, revisiting the site the design is related to, for additional visions or comprehension of the context of the design. Also, during the construction (in architectural design) the site is often visited to check upon the implementation of the design. Especially with participatory design the participants of design are often met at the site. During the site visitation, especially in the beginning of the design process, the designer aims to grasp the surroundings, context for the design as effectively as possible. The designer may not have the possibility for frequent visits due to costs, long distances and the time required to invest in the visits.

The Mobile Media Collector (MMC) is a mobile device and a set of accompanying application(s) for supporting collecting, browsing, and saving location specific and directional media (using a digital compass) related to a urban design site. The tool enables the designer to:

- **Grasp the site** before the site visit, using zoomable maps and aerial imagery.
- **Planning a visit** to the site, by marking interesting places to visit or paths to drive/walk during the visit.
- Make **location specific notes** of the site using the map, including symbolic, textual, voice recordings, photograps or even small videos.
- **Collecting location specific media** (using GPS location, compass direction) and paths from the site.

- Creation of **2.5 dimensional views** of the visits, where the collected media is presented over a two dimensional map, in their correct location and compass direction in third dimension. Also, **panoramas** can be stiched from the images with GPS and compass direction data on a server side.
- **Create sketches**, both **on top of maps and photographs** taken from the site, enabling the user to create perspective drawings on top of photographs. Sketches are also zoomed as the maps are zoomed, and vice versa.

The MMC could also include some support for collaborative and presence features:

- In **collaborative** settings, the designer is also able to see other designers' notes, photographs, sketches, placemarks and paths, including the real time location of the user, enhancing the presence aspects in design.
- **Share** design related media and sketches with other designers both while on the site and in the studio or while preparing for a presentation to the stakeholders of the design.
- **Present** the initial design ideas to other participants in the design process, using the sketches on maps and photographs and the location specific placemarks, paths. The presentation is directed from the mobile device and viewed using either a projector or a large display.
- **Plan further visits** to the site, by providing placemarks, paths, media and sketches which can be used to pinpoint the areas that need more attention or are otherwise interesting for the designer.
- **Use the content** created with the tool in other tools and work settings as the content is stored to a HyperMedia Database (HMDB).

The MMC would have to be usable also in settings where a fast network connection is not available. Often, in field trips, outside city centers and main transportation links, there is no UMTS/WiFi coverage. Large maps do not load effectively over GPRS data networks. For this reason, the MMC will use preloaded map files on the device.

5.8.1 Redesign issues

Mobile Media Collector is a new tool, so no redesign issues yet.

5.8.2 Development and Research

There are many aids designers use in assimilation of the context and environment. Often designers use aids such as maps, aerial or satellite imagery, sketching and photograps (Figure 20) to memorize and collect the experience of the site for further reference. These are then later used in the studio to "revisit the site".

Figure 20: Using photographs and skeches on maps in design.

The material that was collected, including photographs, maps and also sketches perhaps made on top of photographs and maps, is used in the process (Figure 21). In the design, unrelated buildings, areas or sites can also be used as a reference. An unrelated building from a completely different context, even far away, can “give the feeling” of the intended design solution. This kind of inspirational material can also complement the material collected from the site.

Figure 21: Sketching on maps, early phase of design.

The initial design principles and ideas may also be reflected against the real, concrete environment by revisitations accompanied with sketches. This way the designer is able to find areas for improvement and see if the intended design solution would actually work or not. Especially in participatory design these initial sketches may also be discussed with the participants, either in the studio environment or at the site. For example, perspective drawings on top of the photographs from the site can be used to illustrate how a particular mass of buildings would fit a particular site.

5.8.3 The MMC tool concept

Before working on a specific design, the user is able to choose with which project she is working on. After this she can browse the maps, sketches and/or pictures or other media related to the project. Figure 22 of the mockup below illustrates the main feature of the MMC, the handling of location specific media and sketching over maps and photographs. Please note that the mockup size and layout is just used to characterize the functionality included in the concept – actual device can also be, for example, a smartphone.

Figure 22: The Field skecher map and media view.

In the figure above, the designer has selected a photograp of a specific place using the icons on the map, and is currently skecthing on the photograp. Both the map and the photo can also be viewed full screen, giving room for the browsing, viewing and manipulation.

The MMC can be used when travelling to the site (Figure 23). The designer can view the site map, plan the interesting places to visit and to take photographs of. In a collaborative setting, designers can negotiate on who goes where and what are the commonly interested sites and perspectives of the designers on the design task. If the car (van) has a flat display panel installed, the designers can share this in presenting their initial ideas for the visit, as well as when returning, discuss their afterthoughts.

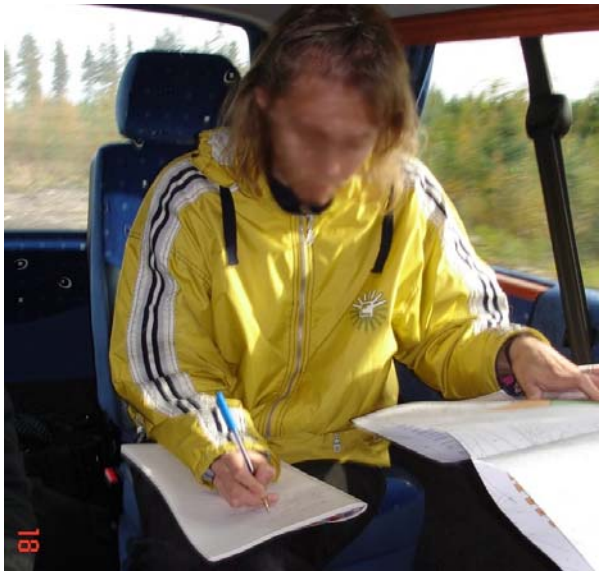

Figure 23: Planning the site visit using MMC in a car.

An important part of the visit is the collection of different types of media. The media helps the designer in grasping the essence of the site, by focusing on particular views and objects in the site from the perspective of the indivudial designer (see Figure 24). This media is location dependent. Also, it is important for the designer to be aware of perspectives and directions. The MMC supports these elements of grasping the site by storing the GPS location as well as the compass direction where the media was stored. The media can then be viewed in relation to the map, for example (see Figure 20).

Furthermore, the MMC is able to select photographs by GPS location, and then by analysing the compass direction of the photographs in the same location, stitch these photographs into a panorama, giving a more comprehensive view of that location to the designer.

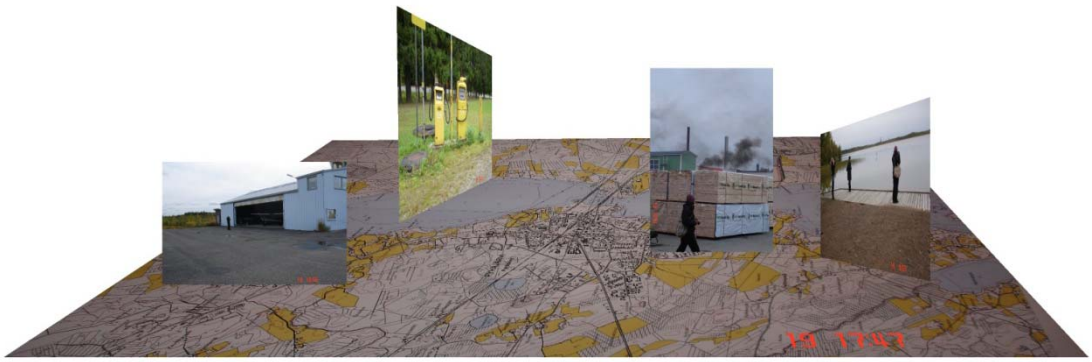

Figure 24: Crude mockup of the view of location specific, directional media placed on map.

In the studio, or in the collaborative meeting with participants of the design, the MMC is used as the designer's tool for mediating her vision of the site and the possible design solutions. The output of the MMC is projected on a wall or to a large display (Figure 25).

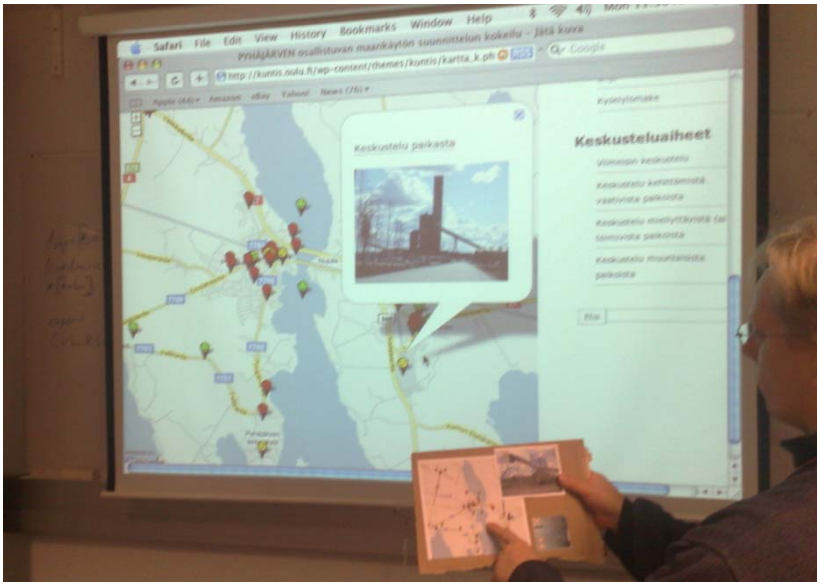

Figure 25: Presenting design ideas, views on a large display to stakeholders.

If many designers are participating in the process, the placemarks, paths and associated media (photographs, videos, sound notes), notes and sketches can be shared wirelessly and displayed at the same time. Alternatively the presenter can focus on individual content or content related to some specific area. The large display acts as a common table for discussing the ideas and forces affecting on the design.

Technically there are many options to implement the MMC (see Figure 26). The device could be a tablet PC (without a keyboard), as small and light as possible. User would have to carry the device on her arm, so weight is an important issue. We have also considered using an Internet tablet, such as the Nokia N810, but here the small screen size may be an issue. Using a smartphone would also be an option, having in-build camera and GPS device. Small screen size and limited input possibilities are the negative sides of using a smartphone. However, we can trial with different types of devices in the initial phases of the development and choose the final implementation platform based on user feedback.

		
Tablet PC without a keyboard	*Nokia N810 Internet tablet*	*Smart phone with a large screen*

Figure 26: Different types of user devices

MMC would also utilize either in-built or external GPS device and an electronic compass for specifying the location and direction of the media. Also an external camera chould be used, since the cameras integrated into in mobile devices (or webcams available for laptops) are usually too low quality for design purposes. Also a video camera and a microphone can be used to store video and sound notes from the field to avoid the cumbersome process of writing (either using a keyboard or touch screen) while standing and moving around in outdoor environment.

5.8.4 Related Work

The eDiary in Atelier IST project, collects a media path with location data. MMC is not only a data collection tool but also a tool for the initial phases of the design, supporting more complex gathering of design related media and also active sketching [Iacucci2004].

HyCon project (Interactive spaces). Technologically similar ideas for location specific media gathering and viewing. http://www.interactivespaces.net.

Iacucci, G., Kela, J. Pehkonen, P. *Computational support to record and re-experience visits*. Personal and Ubiquitous Computing. Springer London. ISSN 1617-4909. Volume 8, Number 2 / May, 2004, pp. 100-109, 2004.

Furthermore, three interaction tools from IPCity bear similar purposes being still from usage and implementation point of view, different. Urban sketcher supports sketching overlays in studio, using 3D technologies. The tool is focused on mixed reality sketching in stationery, not mobile, in the field settings. Scouting concept strives to support real time scouting supporting work in the Tent by streaming voice and/or video from immediate surroundings. MMC is, in comparison, more an independent tool for the desinger in the field, even though it also has some collaborative features. MMC would not include real time streaming of video or sound from the site to the studio. AuthOr supports location specific placemarks and paths. This tool is not so much oriented to support collecting location specific, directional (compass) media and sketching directly. Especially with AuthOr, there is a good possibility to reuse code, if similar implementation techniques are selected for MMC. Also regarding MapLens, there is the possibility for code reuse, if Symbian OS is selected as the implementation platform for MMC.

5.8.5 Testing and public demonstration

Detailed desing and implementation of MMC starts in January 2008, and the first prototype for field tests should be ready by September 2008.

5.8.6 Evaluation

Evaluation will happen with the first field tests during H2/2008.

5.8.7 Specification

Hardware and OS	Mobile phones, Internet tablets or Table PCs (without keyboard, portable)
Software	C++ or Java (CDC)
Core Features	Enables the user to collect and view location specific media, also from other users. Assists in taking grasp an area, supporting designers and architects in the initial design phase. Contains collaborative and sketching features.
Status	Initial design started.
Intended users	Designers, architects. With modification also other users interested in collecting/creating media in the field.
Showcases	WP6, WP9 (for collecting city tales)
Relevance beyond project	Would be usable and extensible over many usage scenarios.

5.9 Location Based Media Browsing on Paper Maps

This tool enables the user to view location based media on top of the map image projected on a smartphone camera screen. The map image is grabbed from the phone camera, as the user holds the camera on front of the map. The overlayed media can be various things, e.g. photographs, locations of other uses, event locations or other information related to any event.

The mail showcase for which the technology is developed is the WP7, where the concept is developed with the name MapLens.

5.9.1 Development & research

MapLens application is developed with C++ on Symbian OS v9. The application functionality is as follows:

1. Acquire the image of the map from the camera of the phone.
2. Analyse the camera image, extracting key features from the image.
3. Match the key features of the map to a database of features prestored on the mobile phone.
4. By using the feature database and the features of the visible area of the map, define the GPS coordinates of the corners of the map shown
5. Extract the location based data from an external database (the HyperMedia Database; see WP5) and displaying layered icons and media on top of the visible area of the map for the user.
6. Enabling the user to select visible overlayed data and viewing the details of data.

As the user moves the camera over the map, the content thus changes, depending on the area visible on the phone screen.

The first versions aim at 1-5 fpu speed, further improving this as the functionality is incrementally developed and the bottlenecks of the performance are found. Initially, the planned areas of optimization are:

- Prefetching the location specific media from the HMDB as soon as the map is found, on to the phone. This optimizes away the network communications part. However, depending on the scenario, additional fresh data needs to be periodically fetched from HMDB.
- Optimization of the image handlng by converting the jpg and other media formats to Symbian bitmap format (mbm) and saving them on the phone. While displaying the image data, the conversion from jpg etc. is no longer needed.

The image analysis is implemented in two complementary components. One component uses tags places on the augmented map for fast analysis. Another component does not need augmentation, but uses the features of the map (roads, rivers, any other distinctive elements on the map) for feature identification.

The location specific media on the maps are fetched from the HMDB using the new APP (Atom Publishing Protocol) interface of the HMDB, over HTTP protocol. The data format the HMDB provided is KML (http://code.google.com/apis/kml/). The KML export feature for the HMDB is currently under development and should be finished by February 2008. The media files are included in the KML file as hyperlinks to the media stored either into the HMDB or elsewhere on the Internet. The media files are fetched onto the phone after receiving the KML file.

The role of the HMDB is to act as a cache and a single source for the location based data displayed on the layers of the tool. Actual media and other data may either be directly stored by other applications to HMDB (like MMS Entrance, Bluetooth Media Dispatcher, eMailEntrace or other tools).

In some scenarios Flickr images are planned to be used in WP7. In these cases we strive to keep the simplicity of the media browser tool and still use only HMDB as the data source. For this purpose, we will build extension modules to HMDB which will prefetch the Flickr geotagged photographs for the geographical areas that are planned to be used with the tool (MapLens). The HMDB may either store the original photographs or just a link to the original photograph in Flickr, depending on the performance difference and the required performance.

5.9.2 Testing and public demonstration

The first version of the tool with limited functionality and restricted performance will be available in March 2008. First field tests are initially planned to be held in Fall 2008. Details of these will be planned during Q1/2008.

5.9.3 Specification

Hardware and OS	Nokia N95, Symbian OS v 9, S60 UI 3rd Edition
Software	C++
Core Features	Grabs mobile phone camera image, extracts features from the image, showing a map. Defines the area of the map visible on screen, gets location based media from remote database and displays media, icons on top of map.
Status	Implementation ongoing, first prototype ready by March 2008.

Intended users	Users interested in location based media, events.
Showcases	WP7, others.
Relevance beyond project	Would be usable and extensible over many usage scenarios.

5.10 Multi-Touch Display

Multi-touch display is a technology that is used to create displays in public spaces. The technology supports creation of multi-touch screens that are several meters wide and located either indoors or outdoors. The system can be used simultaneously by several users. The system is based on computer vision in the near-infrared range (700-1000 nanometers). The system is built to track hand movements on an interactive projection surface. It uses modern off-the-shelf components (FireWire cameras, lenses, filters, infrared illuminators) that enable construction of different custom-tailored systems. For a schematic view of the system see Figure 27. Figure 28 shows the finished prototype being used by two simultaneous users.

The software consists of two parts: 1) touch-display manager and 2) application layer (e.g. developed in WP7). The software is written with a modular approach in C++, splitting code into generic off-the-shelf components and application-specific components.

To detect contacts against the multi-touch-display the system uses an infrared camera coupled to a computer. The computer runs touch-display manager software that 1) captures images from the camera and 2) calculates touch-points from the images using computer vision methods. The software for the first task is platform-specific, while the second one is platform-independent. The touch-display manager is generic software that can be used in any touch-display application.

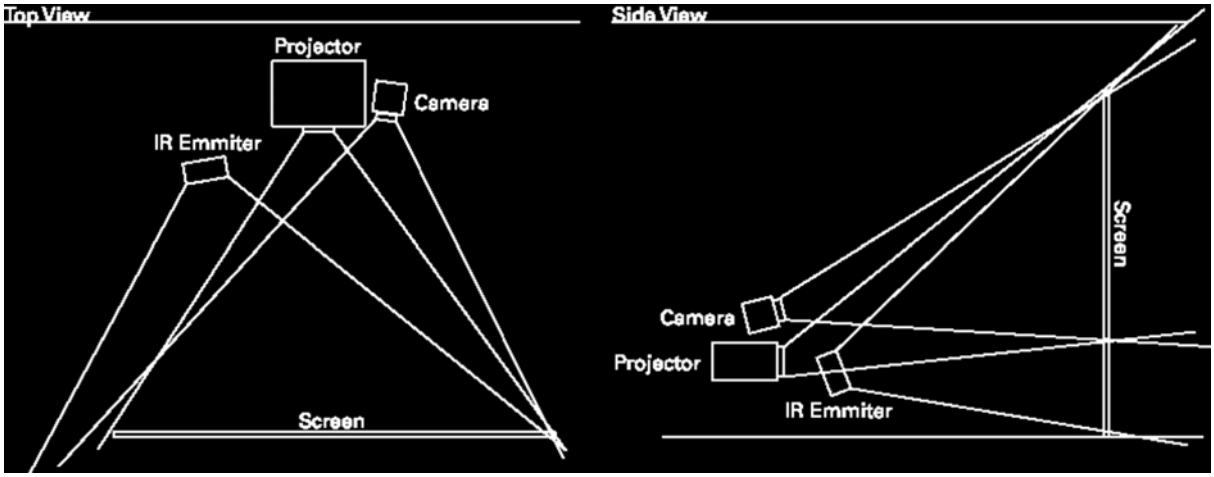

Figure 27 – Set up for multi-touch screens

5.10.1 Redesign decisions

This implementation of the software stack runs on the Linux platform. The applications developed are multi-threaded, so that the application thread can be separated from the touch-screen processing because slow application can hamper image processing and single-threading costs performance. Image processing is done in the background at a fixed rate (regardless of the application). The application sees the touch-screen as providing new fully-processed input samples at fixed rate.

Also, to handle larger screens, multiple cameras and projectors (multihead) are supported at the same time. Separate, dynamic calibration for each camera is implemented (a calibration application for setting up the projection parameters interactively has been developed) and

information is merged at the edge of the camera images. Each camera image is processed in a sepatate thread. Multihead keystoning is handled with OpenGL transformation matrix.

A new public API has been developed for the software. The original API simply gave access to the results of the processing, changing as changes were applied to the computer vision. A separate, stable API layer that provides all the important information from the computer vision has been developed that hides all the threading. It is also more simpler the header being less than 100 lines of C++.

Figure 28 - The first prototype of multi-touch screens

5.10.2 Development & research

The core of the system is a computer-vision based “TouchScreen” component, which has been implemented as a C++ library. This library is augmented by support library “TouchUI” developed in WP7 that simplifies the creation of applications for the touchscreen in the large-scale event showcase. The TouchScreen component has been developed to a point that it works in many real-life situations, both indoors and outdoors. The current efforts are directed at making the system more easily configurable so that it is easy to use in the various showcases.

The next tasks are documentation on about how to design the physical setup of a multi-touch display and on how to use the API in the showcase applications. Also more work is needed to make the hand-tracking algorithm more reliable in difficult conditions. Finally we need to package the touchscreen SDK in a form that can be easily installed by the project partners.

5.10.3 Related Work

Pre-existing research into touch sensitive screens has proposed several solutions, for example common products track one point of contact in two dimensions as a pointer. Current prototypes address large interactive walls, table tops, touch pads and augmented desks. These systems handle multiple inputs and to some extent can capture objects in the vicinity of the screen. The surface being tracked might be horizontal or vertical, with front or rear projection. Moreover the tracking system is embedded, or using computer vision with cameras in front or in the back of the surface. Different technologies are used for tracking. Generally electrode or antenna based solutions can position the object in 2D, and to some extent its distance, but do not provide enough information of the shape of the object. On the other hand camera and IR based tracking systems can offer high resolution information on the shape of objects but often suffer from occlusion and lighting problems.

Smart-Skin [Rekimoto2002] is based on capacitive sensing. It accurately tracks the position of the user's hands in 2D and also calculates the distance of the hands from the surface. Using arrays of antennas embedded in the touch surface DiamondTouch [Dietz2001] supports front-projection on a table. Each antenna transmits a unique signal and users have a separate receiver, connected to the user capacitively supporting distinguishing between simultaneous inputs from multiple users.

Cameras and computer vision techniques using IR tracking are used in TouchLight [Wilson2004], HoloWall [Matsushita1997] Designer's outpost [Klemmer2001], Barehands [Ringel2001] and the frustrated total internal reflection method (FTIR) [Han2005]. In these prototypes, objects that are on or very near the surface are tracked. This also allows video to be simultaneously projected on the surface. Different materials can be used on the surface to either track only point of touch or also sense objects near the surface.

MotionProcessor [Numazaki1998] uses a video camera with an optical IR filter for recognition, and IR lights to illuminate objects in front of the camera directly (using IR reflection). HoloWall [Matsushita1997] uses a rear-projection panel which is semi-opaque and diffusive, the user's shape or any other objects in front of the screen are invisible to the camera until they are close enough to the screen reflecting IR light and thus become visible to the camera. Hardware-wise this system is identical to ours, but Matsushita's publication lacks all implementation details, additionally our software techniques are signicantly different due to different goals.

Another set up is to put camera and IR tracking in front of the surface. The EnhancedDesk project [Koike2002] uses IR cameras to detect the 2D positions of all the fingertips of each hand for such tasks as two-handed drawing and GUI navigation, but their single camera setup cannot determine whether a finger is touching the table surface. Visual Touchpad [Malik2004] by using stereo vision with a camera in front of the surface can not only determine contact information, but also the distance of fingertips from the tabletop. Techniques can be used to extract finger orientation information and hand regions from the video images in real-time, and then transparently augment them over top of the graphical interface as a visual proxy for the user's actual hands. Usually two cameras are used and simple stereo calculations to detect objects but in [Wilson2005] only one camera is used.

Dietz, P. and Leigh, D. *Diamondtouch: A multiuser touch technology*. In UIST ’01: Proc. of the 14th annual ACM symposium on User interface software and technology, pages 219–226, ACM Press, New York, NY, USA, 2001.

Han, J.Y. *Low-cost multi-touch sensing through frustrated total internal reflection*. In UIST ’05: Proc. of the 18th annual ACM symposium on User interface software and technology, pages 115– 118, ACM Press, New York, NY, USA, 2005.

Klemmer, S.R., Newman, M.W., Farrell, R., Bilezikjian, M., and Landay, J.A. *The designers’ outpost: a tangible interface for collaborative web site*. In UIST ’01: Proc. of the 14th annual ACM symposium on User interface software and technology, pages 1–10, ACM Press, New York, NY, USA, 2001.

Koike, H., Xinlei, C., Nakanishi, Y-, Oka, K., and Sato, Y. *Two-handed drawing on augmented desk*. In CHI ’02: CHI ’02 extended abstracts on Human factors in computing systems, pages 760– 761, ACM Press, New York, NY, USA, 2002.

Malik, S. and Laszlo, J. *Visual touchpad: a twohanded gestural input device*. In ICMI ’04: Proc. of the 6th international conference on Multimodal interfaces, pages 289–296, ACM Press, New York, NY, USA, 2004.

Matsushita, N. and Rekimoto, J. *Holowall: Designing a finger, hand, body, and object sensitive wall*. In UIST ’97: Proc. of the 10th annual ACM symposium on User interface software and technology, pages 209–210, ACM Press, New York, NY, USA, 1997.

Morris, M.R., Huang, A., Paepcke, A., and Winograd, T. *Cooperative gestures: multi-user gestural interactions for co-located groupware*. In CHI ’06: Proc. of the SIGCHI conference

on Human Factors in computing systems, pages 1201–1210, ACM Press, New York, NY, USA, 2006.

Numazaki, S., Morshita, A., Umeki, N., Ishikawa, M., and Doi, M. *A kinetic and 3d image input device*. In CHI '98: CHI 98 conference summary on Human factors in computing systems, pages 237–238, ACM Press, New York, NY, USA, 1998.

Rekimoto, J. *Smartskin: an infrastructure for freehand manipulation on interactive surfaces*. In CHI '02: Proceedings of the SIGCHI conference on Human factors in computing systems, pages 113–120, ACM Press, New York, NY, USA, 2002.

Ringel, M., Berg, H., Jin, Y., and Winograd, T. *Barehands: implement-free interaction with a wall-mounted display*. In CHI '01: CHI '01 extended abstracts on Human factors in computing systems, pages 367–368, ACM Press, New York, NY, USA, 2001.

Wilson, A.D*. Touchlight: An imaging touch screen and display for gesture-based interaction*. In ICMI '04: Proc. of the 6th international conference on Multimodal interfaces, pages 69–76, ACM Press, New York, NY, USA, 2004.

Wilson, A.D. *Playanywhere: A compact interactive tabletop projection-vision system*. In UIST '05: Proc. of the 18th annual ACM symposium on User interface software and technology, pages 83–92, ACM Press, New York, NY, USA, 2005.

5.10.4 Testing and public demonstration

The technology is used in the large-scale event showcase WP7 and the WP6 Urban Renewal Showcase has expressed interest to use it as well. A first evaluation was crried out in March 2007 and is reported in the next section. Further test and public demonstration are within WP7.

5.10.5 Evaluation

To evaluate the practical relevance of the techniques we organized a user evaluation. The approach has been to evaluate a realistic scenario and avoid instructing users to follow predetermined tasks. To this end we asked users to browse their own photographs and create two collages out of them including sketches and text. The objective of the study was to evaluate the capability of the system in providing fluid gestural interaction and how easy such a system could be picked up by novices. The evaluation was done with 7 users, 2 females and 4 males between 24 and 40 of age and a 4 year old child.

The data collection used as primary data video based interaction analysis. In addition the initial phase of the session applied the think aloud protocol. Finally a semi-structured interview collected verbal user impressions of problems and successful features. Users were new to the system and were introduced in few short sentences to the session. No training or manual was available (neither printed nor inbuilt in the system) the researcher explained with one phrase each gesture and invited the user to create 2 or more picture collages. The questions for the study were twofold. On one hand, we set out to evaluate the capability of the system to successfully support the creation of satisfying collages. On the other hand the aim was to evaluate the appropriation of gestural interaction by analyzing the success and failure rates of gestures and inquiring about their cause. For the latter the coding in the data collection consisted of counting each gestural interaction: successfully tracked gestures that were intended, gestures performed that were not recognized, and recognition of not intended gestures.

A gesture starts when the fingers touch the screen and finishes when the fingers are lifted. These were the categories used for coding in the interaction analysis along with the percentage of occurrence: move 49.24%, pan 32.14%, rotate/ scale 9.31%, sketch/keyboard 2.45%, zoom 2.08%, explode 1.81%, unsuccessful 2.96%. As the percentages also reflect a good portion of time was spent in the sessions navigating the timeline to find pictures and moving (flipping) them to browse them. However in the percentages of zooming there are gestures to move picture to create the collage. Sessions lasted for around 30 minutes, and

users created either two or three collage. Collages generally included from 4 to 13 pictures, sketches and text.

Findings

Behind unsuccessful gestures. The overall percentage of gestures that failed (where the intended gesture of the user was not recognized) is fairly low for a prototype but still important. Careful analysis reveals that 79% of the unintended gestures were carried out with one finger resulting in sketching 96% of the time. The reason for this is that at the time of the evaluation the recognition of sketching gesture was implemented as a one finger touch activated after a couple of seconds of contact. Now we are implementing a new gesture for sketching where the hand is in the posture of holding a pen. By removing the ambiguity of one finger sketching 96% of the unintended gestures should disappear.

Intuitiveness and quick learning. The system required surprisingly little instruction. To deepen this aspect we had one child take part in the evaluation. A comparison of the results indicates that the 4 year old child did not have substantially more problems than the adults as the failed gestures were only 1.8% higher. There was also a much higher occurrence (18.6%) in rotating and scaling gestures as the child spent more time in playing with the dynamics of the pictures rather than constructing collages. Deliberate design choices in avoiding things like menus, modes (e.g. editing mode, annotating mode etc.) which are often found in similar multitouch systems appear to have contributed significantly to the intuitiveness of the system. The interface was so intuitive that users expected other intuitive features such as erasing by quickly rubbing their open hand like a board rubber to work. Moreover the failure rates decreased during the short sessions, in fact 75% of the failed gestures occurred within the first half of the sessions, with 43,75% occurring in the first quarter of the session. Simultaneous gestures Users quickly appropriated ways of using two gestures simultaneously. The amount of simultaneous gestures was as high as 16,21%. The simultaneous gestures included: 36.29% simultaneously rotating and scaling a picture with two hands, 29.14% simultaneously using one hand to pan the timeline and the other to move a picture, the same percentage 29.14% were occurrences were both hands were used to move two pictures at the same time, finally a 5.43% were gestures where scaling and rotating was coupled with moving the picture. The high percentage in simultaneous gestures serves to demonstrate the reliability and fluidity of the tracking technique that enabled users in doing truly multi-touch two-hand gestures.

Playfulness. What all users praised was the possibility to flip and throw pictures. This was especially made engaging by the “physics” implemented in the application that recognized the speed of gestures and animated objects accordingly. This enables very fast flipping of large piles of pictures. Also several uninteresting pictures could be eliminated quickly by applying enough speed in the gestures. Users reported that the system is not very well indicated for fine work as at times it was too sensitive but was very good for quickly flipping through large amounts of pictures. Another observation was the playful use of the child. In particular he made pictures very small to make it very easy to spin them at high velocity.

Moreover he would enlarge pictures very much to move a very large picture around the screen.

5.10.6 Specification

Hardware and OS	• Data Projector • Black and white Camera • Infrared lenses and filters • Infrared emittors • Multiple cameras and projectors are supported to handle larger screen (so far 2 FireWire cameras with 60fps and

	VGA resolution have been used with maximum of 4 projectors) • PC Hardware • OS: Linux
Software	The software consists of two parts: 1) touch-display manager and 2) application layer. The software is written with a modular, multi-threaded approach in C++. A high definition firewire camera with IR lense is used to track objects near the screen. The computer runs touch-display manager software that • captures images from the camera (platform-specific) • calculates touch-points from the images using computer vision methods (platform independent). This implementation of the software stack runs on the Linux platform. Image processing is done in the background at a fixed rate (regardless of the application). The application sees the touch-screen as providing new fully-processed input samples at fixed rate. Support for multiple screens and cameras Separate, dynamic calibration for each camera is implemented (a calibration application for setting up the projection parameters interactively has been developed) and information is merged at the edge of the camera images. Each camera image is processed in a separate thread. Multihead keystoning is handled with OpenGL transformation matrix. A separate, stable API layer provides all the important information from the computer vision.
Core Features	• Multiple point touch-screen interaction • Detecting points of contact • Detecting fingers • Identifying hands • Tracking hands on the screen at 60 Fps • Day and night mode
Status	Prototype

Intended users	Citizens and visitors pupils and as well as senior citizens.
Showcases	WP7
Relevance beyond project	This component has raised a lot of interest and has made possible the creation of a startup that commercializes products and services for multi-touch screens see D7.3.

5.11 Augmented map table

The augmented map table implements a tabletop augmented reality environment for enhancing paper-based artifacts, such as maps, with dynamic information and interaction capabilities. This system is a result of prior work by the University of Cambridge and will be included into showcases WP6 and WP8 to provide tangible user interfaces to the authoring systems deployed by the showcases.

5.11.1 Development & research

The augmented map table is a spacial augmented reality system. A camera-projector pair observes a table surface and track the location of one or more maps on the table. Projecting computer graphics onto the maps marks dynamic information such as waypoints, routes, icons and areas of interest. A set of tangible interaction tools in the form of rectangular pieces of cardboard are tracked as well. These tools provide access to geo-referenced information, alternative views on the map through magic lenses or user interfaces to active components. Examples are an image-browsing tool for pictures referenced to locations in the map, a magic lens for satellite imagery and a PDA that receives and displays component specific user interfaces (see Figure 29).

Figure 29 - (Top left) Overview of the augmented map system. The demonstration setup shows the river Cam flooding surrounding areas in the City of Cambridge, UK and related information. An image browser (top right) provides access to geo-referenced pictures, while a satellite view (bottom left) shows a different layer of data. A PDA-based tool (bottom right) displays a remote user interface to communicate with a helicopter.

5.11.2 Related work

Related work deals with the generic approaches of spatially augmented reality and tangible user interfaces. The metaDESK [Ullmer1997] and later work on Urp [Ishii22] created a city planner application where physical models representing a city scape were augmented with projected graphical information. Here, tangible tools were used to set parameters of simulations of shadows, sun light reflections or wind effects. The application was validated with a class of architecture students in a real course setting. The work on spatially augmented reality [Raskar1998, Raskar1999] showed that augmenting physical objects with projector based information is a valid alternative to more immersive systems using HMDs.

The Rasa project [McGee2000] describes a prototype environment to support work and decision making in military command posts relying on paper maps and post-its with hand written symbols. The authors tried to augment the established work practices and artifacts with computer supported interaction based on speech and character recognition, and simple localisations using Smartboard technology. In recent work they also investigate vision-based tracking methods [McGee2001].

Bobrich and Otto [Bobrich2002] present 3D overlays of digital elevation models over real maps with a video see-through augmented reality setup using fiducial based optical tracking. The interaction is limited to selecting different rendering modes. Moreover, the described approach replaces the original map image with a virtual model, instead of explicitly adding to it. Marking-up maps with electronic information was investigated by Reilly [Reilly2004]. Locations on maps were identified with RFID tags which are read by a handheld device. Then corresponding information was presented on the device.

The augmented map table differs from the above approaches in that it neatly combines spatially augmented reality to annotate the paper-based artifacts directly with tangible interfaces for manipulating both artifacts and tools. Manipulating and augmenting the artifacts retains their intrinsic tangible properties such that users can move and rotate the maps – and thereby control their view of them – as they are used to. The vision-based tracking solution also allows use of unmodified materials and simple tool shapes that do not require fiducial markers and are robust against partial occlusions by the users.

Bobrich, J. and Otto, S. *Augmented maps*. In Geospatial Theory, Processing and Applications, IAPRS Ottawa, Canada, 2002.

Ishii, H., Underkoffler, J., Chak, D., Piper, B., Ben-Joseph, E., Yeung, L., and Kanji, Z. *Augmented urban planning workbench: Overlaying drawings, physical models and digital simulation*. Proc. ISMAR 2002, pages 203--212, Darmstadt, Germany, 2002. IEEE.

McGee, D.R., Cohen, P.R., and Wu, L. *Something from nothing: Augmenting a paperbased work practice via multimodal interaction*. Proc. DARE 2000, pages 71--80, Helsinor, Denmark, April 12--14 2000. ACM.

McGee, D.R., Pavel, M., Adami, A., Wang, G., and Cohen, P.R. *A visual modality for the augmentation of paper*. Proc. PUI 2001, Orlando, Florida, USA, Nov. 15--16 2001. ACM.

Raskar, R., Welch, G., and Chen, W.-C. *Table-top spatially-augmented reality: Bringing physical models to life with projected imagery*. Proc. IWAR'99, San Francisco, CA, USA, October 20--21 1999. IEEE.

Raskar, R., Welch, G., and Fuchs, H. *Spatially augmented reality*. Proc. IWAR'98, San Francisco, CA, USA, November 1st 1998. IEEE.

Reilly, D. *Marked-up maps: exploring mixed media for group navigation and information gathering*. Proc. MUIA'04, Glasgow, Scotland, Sep. 13 2004.

Ullmer, B. and Ishii, H. *The metaDESK: Models and prototypes for tangible user interfaces*. Proc. UIST '97, pages 223--232, Banff, Alberta, Canada, October 1997. ACM Press.

5.11.3 Specification

Hardware and OS	The augmented map table operates under Linux on a standard desktop PC with state-of-the art CPU and graphics hardware. USB or Firewire cameras can be used as input.
Software	The system is written in C++ using the TooN numerical library, libcvd computer vision library and Coin scene graph library.
Core Features	Tracking of printed maps and rectangular interaction tools on a table surface, augmentation of maps and tools with projected graphics.
Status	Stable demonstrator to be integrated with other showcase technology.
Intended users	Any user
Showcases	Mainly WP6, potentially WP8 as user interface to authoring software.
Relevance beyond project	General user interface prototype for Command & Control type environments, such as first responder management.

6 Dissemination

Wolfgang Broll, Jan Herling, Lisa Blum. *Interactive Bits: Prototyping of Mixed Reality Applications and Interaction Techniques through Visual Programming*. To be published in Proc. of IEEE Symposium on 3D User Interfaces, 2008, IEEE, Piscataway, NJ, USA.

Wolfgang Broll, Irma Lindt, Iris Herbst, Jan Ohlenburg, Anne-Kathrin Braun, Richard Wetzel. *Towards Next-Gen Mobile AR Games*. Submitted to IEEE Computer Graphics & Applications, 2008.

Iris Herbst, Anne-Kathrin Braun, Rod McCall, Wolfgang Broll. *Interactive City Exploration through MR*. To appear at VR 2008, Poster Session, 2008.

Iris Herbst, Sabiha Ghellal, Anne-Kathrin Braun. *TimeWarp: An Explorative Outdoor Mixed Reality Game*. SIGGRAPH 2007 Poster, 2007.

Antti Juustila, Tanja Kangas, Toni Räisänen, and Kari Kuutti. *Bringing Urban Design Site to Studio by using a Remote Surveillance Camera.* In Proc. of the Workshop on Imaging the City, 2007.

Joseph Newman, Alexander Bornik, Daniel Pustka, Florian Echtler, Manuel Huber, Dieter Schmalstieg. *Tracking for Distributed Mixed Reality Environments*. Proceedings of IEEE Virtual Reality Workshop on Trends and Issues in Tracking for Virtual Environments, Charlotte NC, USA, 2007-March.

Valérie Maquil, Thomas Psik, Ina Wagner, and Mira Wagner. *Expressive Interactions Supporting Collaboration in Urban Design*. In: Proc. of GROUP 2007, Nov 4-7, Sanibel Island, Florida, USA, 2007.

Valérie Maquil, Thomas Psik, Ina Wagner. *The ColorTable – A design Story*. Tangible and Embedded Interaction 2008, Bonn, February 18 and 21, 2008 (accepted paper).

Jan Ohlenburg, Wolfgang Broll, and Irma Lindt. *Orchestration and Direction of Mobile MR Games*. *To be published at CHI 2008 Workshop - Urban Mixed Realities: Technologies, Theories and Frontiers*, 2008.

Jan Ohlenburg, Wolfgang Broll, and Irma Lind. *DEVAL - A Device Abstraction Layer for VR/AR*. In Proc. of the HCI 2007, Beijing, PR China, 2007.

Christian Pirchheim, Dieter Schmalstieg, Alexander Bornik. *Visual Programming for Hybrid User Interfaces*. In Proc. of MRUI'07, Charlotte, NC, USA, 2007.

Peter Peltonen, Antti Salovaara, Giulio Jacucci, Tommi Ilmonen, Carmelo Ardito, Petri Saarikko and Vikram Batra, *Extending Large-Scale Event Participation with User-Created Mobile Media on a Public Display*. Best Paper Award ACM MUM 2007 Mobile and Ubiquitous Multimedia Conference.

The Multi-Touch Display and the application of WP7 Citywall.org appeared on the media in television and magazines in Finland, Italy and the UK moreover this component has made possible the creation of a startup that commercializes products and services for multi-touch screens see D7.3 for detail.

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