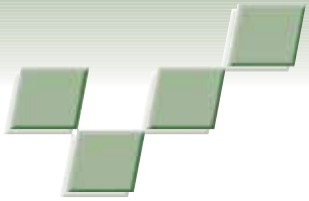


Managing Complex Augmented Reality Models

Dieter Schmalstieg, Gerhard Schall, Daniel Wagner, and István Barakonyi
Graz University of Technology

Gerhard Reitmayr and Joseph Newman
University of Cambridge

Florian Ledermann
Vienna University of Technology

Mobile augmented reality requires georeferenced data to present world-registered overlays. To cover a wide area and all artifacts and activities, a database containing this information must be created, stored, maintained, delivered, and finally used by the client application. We present a data model and a family of techniques to address these needs.

To successfully present world-registered information displays, mobile augmented reality (AR) applications require a detailed model of the user's real environment composed of both visual and nonvisual information. Therefore, models for AR are more difficult to produce and maintain than models typically used for VR, which concentrates solely on the visual fidelity of a purely virtual model. Both AR and VR models are frequently based on measurements taken from real environments, such as architectural models. But unlike many VR models, AR models require semantic interpretation of the environment. For example, AR uses geometry not only for visualization purposes, but also for handling occlusions, rendering of artificial shadows, interaction, and vision-based tracking of real objects. Therefore, the structure of AR models is more complex than that of VR models.

Besides structural complexity, the model scale is also significant. The long-term goal of mobile AR is to let users move unconstrained throughout a wide area, and to continuously provide assistance for a wide variety of tasks. This requires coverage of the whole area and all the possible contained tasks in the underlying AR model. Scaling AR models to such wide area-modeling coverage is only practical by leveraging legacy databases, such as existing digital maps.

In this article, we summarize our work over the last five years to build a tool chain based on the Studierstube framework.[1] We address many of the issues in the management of large-scale, general-purpose AR models. Throughout our work we have been inspired by recent trends in online information systems, in particular Web-based applications. Similar to the Web, open standards can enable the delivery of complex data sets from a wide variety of sources to a large number of AR clients. The clients execute interactive AR applications, and these systems interpret the data and transform it when necessary. We are leveraging architectural patterns and existing tools where possible—specifically, we have based our modeling efforts on the lingua franca of the Web, XML.

An AR modeling pipeline

We organize the management of AR models along the lines of a conventional information-processing pipeline, which has as its main stages acquisition, storage, delivery, and use of the data. This organization separates creation and use of AR data into distinct phases. We imagine that in the near future, mobile AR users will participate in the AR database creation, either actively or passively, by providing updates online. However, such user participation requires a critical mass of users, so initially we have concentrated on high-quality tools for noninteractive content management.

Data acquisition

The first problem to address when building an AR database is how to get data into the database. We describe three principal methods of data acquisition: conversion of data from legacy databases; surveying of a physical area[2] and its contained artifacts; and computer-assisted authoring of location-based, interactive content using a designated authoring tool.[3]

Data storage

The main challenge of the data repository is not so much the actual storage technology, but to find out how AR models must be structured so that they are sufficiently flexible to address a wide range of AR applications and lend themselves to real-time use in location-based applications. We designed the data model and query mechanism around XML formats, which have proved more flexible than conventional relational databases and lend themselves to incremental prototype design. The hierarchical nature of XML documents conveniently fits established hierarchical modeling techniques for geometric models. The associative addressing used in XML is suit-

Published by the IEEE Computer Society

able for attaching semantic information to the model without restricting the scope of possible applications.

Data delivery

This stage is concerned with making the data accessible to clients. Speaking in database jargon, we require a query language to describe the required piece of information, and custom database views, which present the data in a way that a given client can directly use. This view generation must happen at runtime based on client queries and represents an important and potentially complex part of the application logic. The principal advantages of on-the-fly transformation of data are that complex data replication is avoided, and network bandwidth use can be optimized. We use Extensible Stylesheet Language Transformation (XSLT) stylesheets and a powerful, context-sensitive scene graph extension for the view generation.[4]

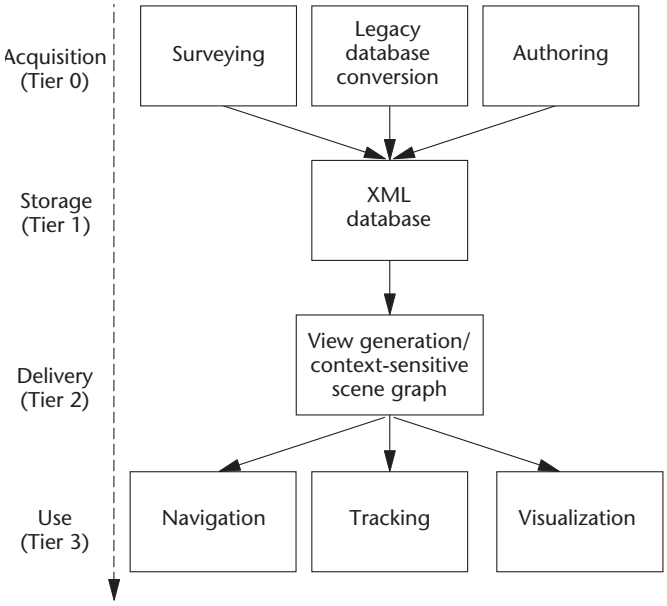
Data use

Mobile AR clients are responsible for presenting the data to the user and enabling interaction. A data-driven approach ensures that a single application for viewing the 3D model (an AR browser) is sufficient for a wide range of applications. Model data helps configure all relevant subsystems of the client application, including 3D visualization, tracking, and the user interface. We use an enhanced scene graph for visualization. The scene graph is based on Coin3D (see http://www.coin3d.org), a free implementation of the Open Inventor API. Custom scene graph extensions embed visualization techniques and user interface methods directly in the scene graph.[5] Consequently, the scene graph is self-contained and a generic AR scene graph browser can execute arbitrarily complex AR applications. The scene graph embeds all the application logic, and is produced on the fly through view generation, so applications are effectively delivered on demand to the client. Besides the scene graph, other subsystems of the AR client also rely on dynamic model delivery from the database. For example, the tracking subsystem requires device configuration data[5] and georeferenced features for optical tracking, while the navigation subsystem relies on a topological model of the surroundings.

We organized the database, as described previously, as a classic 3-tier model involving a database server as tier 1, an application server creating the views as tier 2, and the client as tier 3. The offline acquisition can be seen as tier 0 (see Figure 1). Each tier involves tools and techniques specific to AR models. While storage of the generic model in tier 1 is independent of individual applications, the remaining tiers 0, 2, and 3 are not. Therefore, the general techniques for modeling and storage are explained in the next section without reference to an application. Our description later of two major case studies involves an outdoor[4] and an indoor[6] AR navigation system we have built, demonstrating implementations of tier 0, 2, and 3 components.

Data model and storage

When developing an appropriate data model several important design issues need to be considered, which are pointed out in the subsequent section.



1 The management of augmented reality models is organized as a pipeline consisting of acquisition, storage, delivery, and use stages. Each stage requires specific technologies, presented in the separate sections of the article.

Model language

The core data model employed in our work consists of a structured collection of objects in the physical environment with associated information—taking place at the tier 1 (storage) stage. We designed an XML dialect called Building Augmentation Markup Language (BAUML) to cover both indoor and outdoor 3D environments, including topological information to derive navigation hints and georeferenced semantic information.[4] See the "Built on XML" sidebar on the next page for further details on XML use.

We based the model on an object-oriented approach. The base classes are generic object, spatial object (adding pose information and geometrical representation), and spatial container (adding a child subelement to aggregate other objects for hierarchical composition). We can easily extend the schema to special applications since legacy applications can provide reasonable fallback behavior for unknown object types by considering their base class. Alternatively, we can attach to every object annotations containing free-form XML subscene graphs.

The system interprets the XML tree comprising the model in the standard geometrical way, by defining a child's pose relative to its parent. However, the XML format affords the definition of other than spatial relations by using relational techniques such as referring to object IDs and annotations.

For outdoor use, the model introduces buildings and waypoints as distinct features. We can attach waypoints to either a specific building location (usually the main entrance) or locate them along a footpath. The approach associates buildings with address codes, and a network of pedestrian routes connects the waypoints.

We structured the indoor model into rooms and portals (that is, doorways), implying building topology to support indoor navigation. The model explicitly sup-

ports fiducial markers—from ARToolKitPlus (see http://www.studierstube.org/handheld_ar/artoolkitplus.php)—as special spatial objects to support an indoor tracking system.

We can annotate points or regions of interest with free-form text or by specifying external multimedia files through MIME types. These annotations can display landmarks or tourist information. The model makes use of keywords for all objects as a simple extensible means for filtering the annotations.

We can view an AR model as a special case of a geographic information system. However, when we started our work, no suitable XML standard for 3D GIS existed. Likewise, the XML standard for 3D graphics, X3D, is oriented toward conventional computer graphics and even less suitable for our purposes. We therefore eventually designed our own XML dialect. Recently we have begun to design a new version of our model specification as an application profile (extension) of the new Geographic Markup Language version 3 standard (see http://www.opengeospatial.org/standards/gml). However, the work reported in this article builds on the original custom XML dialect.

Built on XML

A key observation toward our current design was that the most widely established data model for network computing is XML. Using XML as a model core has a number of advantages for complex information systems. XML's features include the following:

- self-documenting format that describes its structure as well as data types and meanings of values;
- simultaneously human- and machine-readable format;
- presentation of many basic data structures such as lists, trees, and records;
- weak type system (when not enforcing a schema) that supports quick changes in data structures and thereby assists rapid prototyping; and
- structures that can be addressed using standardized methods, such as XPath.

The fact that a high-quality, open source implementation for many XML technologies exists and high-quality tools for creating, modifying, and validating XML documents are available adds to the benefit of using XML.

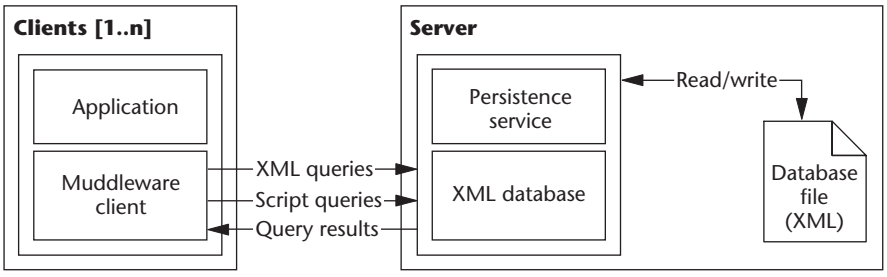
Many geodatabases and online information systems are built on SQL databases. We chose XML over SQL for several reasons. SQL is suited to huge amounts of data in flat structures. In the case of AR models, data for all the heterogeneous locations, artifacts, and so on require less space but is highly structured, which is a natural strength of XML. With XML, we can easily extend existing structures with new attributes and child elements without breaking existing code.

Database server

For the storage and delivery of the AR models—at the tier 1 (storage) stage—we have developed an XML database called Muddleware (see http://www.studierstube.org /handheld_ar/muddleware.php). This database is a real-time store for XML documents that provides persistence and that clients can address associatively using XPath. A large number of clients (we have successfully simulated 200 simultaneous clients) can connect to the server by using C++, Java, or XPath queries. Muddleware stores the data as a document object model (DOM). Muddleware supports atomic operations, a simple scripting language, and a publish-and-subscribe mechanism for database updates. Figure 2 gives an overview of the Muddleware architecture.

XPath allows queries by pattern matching and allows convenient addressing of multiple levels in the hierarchical DOM structure. It's a purely functional language, which makes it easy and fast to implement. We chose it over XQuery because it's sufficient for our query purposes and more efficient to implement.

Persistence is obtained by continuously journaling changes of the DOM to disk. The requirement to hold the DOM in memory limits DOM to a few 100 Mbytes, not counting multimedia files, which are stored as URL references to a separate Web server. This capacity is sufficient for our current models of up to 1 km^2 for outdoor applications, but is not a long-term solution. Therefore, we have experimented with the commercial XML database Tamino (see http://www.softwareag.com/de/products/tamino/), again using XPath as the query language. Performance was acceptable, although database updates were not really satisfactory. However, Tamino precludes rapid prototyping by requiring defining a schema in advance to prepare the internal structure. We therefore decided to continue the development of Muddleware, which not only suits our performance requirements but is also suitable as a real-time blackboard for collaborative AR applications.



2 The Muddleware real-time XML database processes XPath queries and returns XML fragments to an arbitrary number of clients.

Outdoor tour guide

The canonical tour guide application was our first choice as an example scenario for outdoor AR. The rich cultural heritage of Vienna as well as the availability of reasonably accurate digital map material from the city administration made this an obvious test case. Our AR tour guide—Signpost—should support navigation and information browsing in 3D.

The hardware setup for the mobile AR application originally consisted of a backpack system with a notebook equipped with various sensors, and a see-through, head-mounted display. We

3 **The evolution of mobile augmented reality hardware from (a) backpack systems to (b) Ultra-Mobile PCs, (c) personal digital assistants, and (d) smart phones.**

achieved tracking through a combination of differential GPS and an inertial orientation tracker, while GPRS and WiFi provided wireless network connectivity. Indoors, the system employs ARToolKitPlus fiducial tracking from the head-mounted camera. Since 2003, we have replaced the backpack system with handheld Microsoft Ultra-Mobile PC and Windows CE devices. All these devices run the same Studierstube software framework (see Figure 3).

Visualization and user interface

The system presents information—at the tier 3 (use) stage—to the user on the head-mounted display. This information either appears as graphical objects rendered to fit into the natural environment or as text, images, and 3D objects providing a heads-up display. The system draws graphical objects to enhance and complement the user's perception of the natural environment. It can represent abstract information, alternative representations of real objects, or highlighted real structures. The heads-up display provides a GUI consisting of typical 2D widgets such as buttons, lists, and text fields and that provides other status information. Figure 4 shows a typical view through the user's display.

The user can control a cursor within the 2D user interface in the display's upper right corner with a touch pad that is either worn on the belt or handheld. She can switch between different application modes, such as navigation, information browsing, and annotation. Each mode presents a number of individual panes to provide control of parameters and other options related to the current task. A general heads-up display at the bottom of the view presents generic information, such as the current location, selected target location, distance to the target, and an orientation widget modeled after a compass.

Interaction with the environment either occurs implicitly using the user's location and orientation or explicitly using the viewing direction. Some functions use a ray-picking interaction that the viewing direction determines to select objects or establish object placement. In this case, a crosshair displayed in the heads-up display identifies the exact location of the intersection in screen space.

Navigation

In the navigation mode—taking place in the tier 3 (use) stage—the user selects a specific target address or a desired target location of a certain type, such as a supermarket or a pharmacy. The system then computes

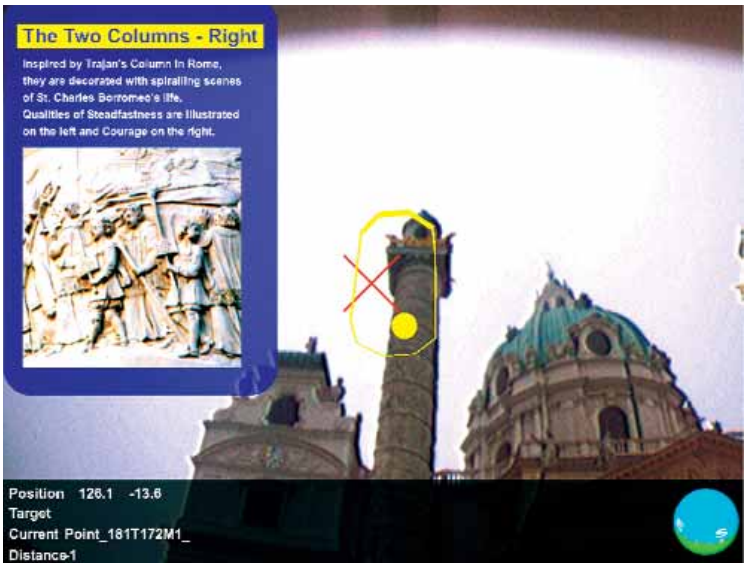
4 **A visualization of the path to the selected target with clipping on known objects.**

the shortest path in a known network of possible routes. Moreover, it's interactive and reacts to the user's movements by continuously recomputing the shortest path to the target, if the user goes astray or decides to take another route.

The system displays the information as a series of waypoints visualized as cylinders standing in the environment. Arrows connect these cylinders to show the direction the user should move between the waypoints. Together they become a visible line through the environment that is easy to follow (see Figure 4). The user can enable an additional arrow that points directly from her current position to the next waypoint. Buildings can clip the displayed geometry to enable additional depth perception cues between the virtual information and the real world. Finally, the system provides simple directional information if the user can't perceive the next waypoint because she is looking in another direction.

Information browsing

The information browsing mode—tier 3 (use) stage—presents the user with location-based information. Location-referenced information icons appear in her view, and she selects them by looking at them. A pop-up presents associated information. The application conveys historical and cultural information about sights in Vienna. For example, frescos on a building facade carry infor-



5 The user selects the column by looking at it, triggering display of the content.

mation on the sculptor who created them and the picture's topic. Statues give information on the background of the person they represent. In the current application, we use geometric representations of building parts to annotate these with cultural information. The annotated hotspots appear to the user as outlines of parts of the building. A virtual ray is cast through the center of the display and intersected with the hot spots. The first icon hit triggers the display of information associated with it (see Figure 5).

A user might not be interested in all types of available information or might get overwhelmed by a large number of locations that trigger information displays. Therefore, the system describes objects and content with a keyword set, and the user can select interesting information by selecting a subset of these keywords. The user will only see information matching the selected keywords.

Context-sensitive scene graph

To facilitate the development of the Signpost application, we wanted to have a tool that allows scene graph use not only for graphical rendering, but also as a geometric database for other purposes—for example, to determine the location of a particular historical artifact for which cultural annotations are available. To this end we developed a context-sensitive scene graph—related to tier 2 (delivery).[5] A set of context parameters maintained during scene graph traversal allows parameterization and repurposing subscene graphs in various ways. The system maintains context parameters—modeled as key and value pairs—as part of the traversal state, which makes them independent of the scene graph structure.

By using pointers as parameters, we can insert template subscene graphs multiple times during the traversal. In contrast to a conventional graph structure, the binding of child nodes to their parents happens late, during the traversal itself, so the system can change nodes for each traversal and provide a flexible way of assembling complex scene graphs.

The scene graph will adapt the visual appearance of its contained objects to dynamically changing requirements, and even compose subgraphs on the fly. A particular visual representation is simply an instance of a template subgraph combined with a specific choice of context parameters. The system creates combinations of content and visual interpretation during traversal only, which is the actual moment in time the user requires them. The context-sensitive scene graph helps configure the scene graph of the second part of Signpost, the information browsing component.

Using the context-sensitive scene graph in Signpost

The information browsing component requires three different representations of all information items:

- a highlighting geometry registered in 3D to the information location (here, a building part outline),
- a picking geometry to intersect the user's viewing ray, and
- the information overlay presented to the user (here, 2D billboards with text and images or 3D objects).

The component makes extensive use of the context-sensitive scene graph concept to organize these different presentations per object.

To allow selection of items based on keywords or other means, the scene graph contains a flat structure of subgraphs, each representing a single object. By traversing subsets of the flat hierarchy, the system only renders selected objects. Each subgraph contains three branches that contain the rendering and geometric information for each of the three modes (see Figure 6). A context-sensitive switch selects which of the branches is traversed.

During highlighting rendering, a context parameter (vis_type) selects the first branch that produces the outlines to represent locations with information. The picking traverses the same scene graph but sets vis_type to select the picking geometry to intersect with. If the system finds an intersection, it selects the corresponding item and traverses its subgraph again with vis_type configured to render the overlay.

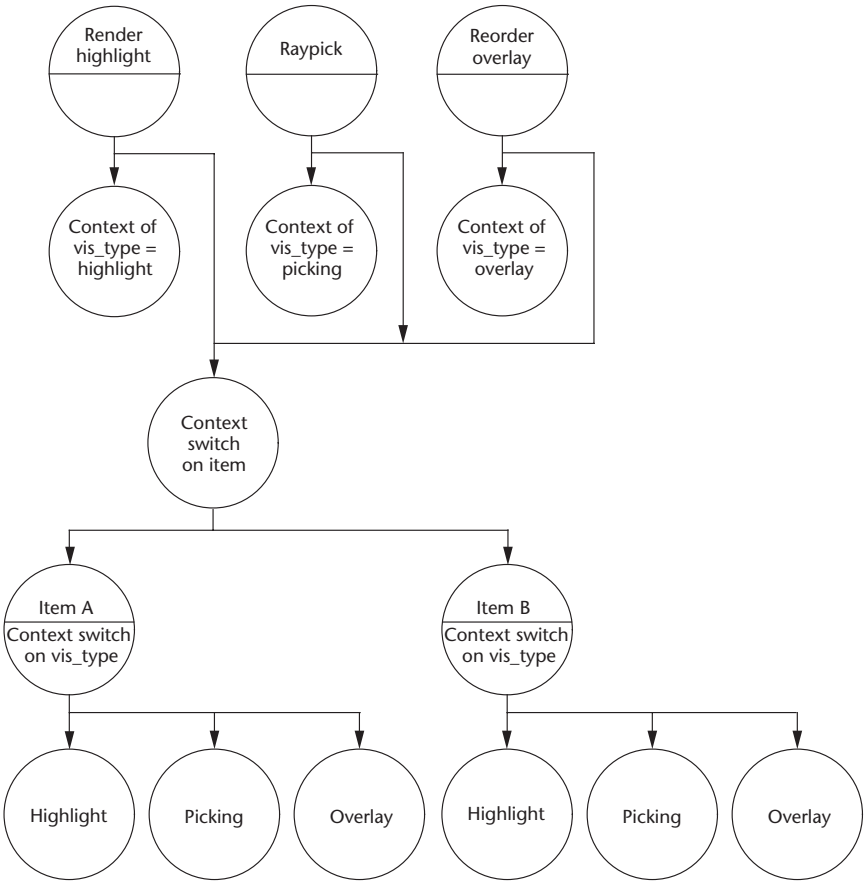
View generation

The overall application's implementation is solely based on an intricately choreographed and interconnected node set in the scene graph—these include user interface widgets, navigation, artifact visualization, and annotation overlay. The scene graph stores not only visual information, but also more general information such as waypoints, route edges, or geometrical objects representing the waypoints and edges. The system maintains correspondence between these data items by using shared indices and context-sensitive scene graphs.

As the complexity of creating these descriptions for larger data sets increases dramatically, manual authoring becomes error prone and consequently, the system needs automated methods to create the lists. The auto-

matic view generation—in tier 2 (delivery) of the pipeline—bridges the gap between the XML database model and the complex scene graphs. It uses a dedicated XSLT processor to generate the more verbose and redundant representation as a scene graph from the concise and compact BAUML model. For example, the stylesheet not only generates geometric descriptions for buildings and waypoints but the route information for the navigation node. The system converts annotation elements for buildings to the context-sensitive highlighting/picking/overlay subscene graphs. It then generates the user interface panel and all connections to and from the 3D scene along with the geometric content of the scene.

The presentation can be customized during translation by specifying alternative template geometries. The user can then directly download the resulting scene graph and use it in the general-purpose AR scene graph viewer, which is essentially an interpreter for a complex semantic model.



6 Information items traverse different subgraphs for rendering, picking, and content display. A context parameter set before traversal selects the right branches through the whole graph.

Legacy database conversion

One of the objectives of our modeling efforts was to provide a single consistent source of data for all AR applications—occurring at the tier 0 (acquisition) stage. This step requires converting and integrating information coming from external sources into the database without introducing undesirable artifacts during the conversion. For our outdoor model, Vienna city administration donated 2D GIS data in a VRML format containing footprints of several dozen buildings.

Colleagues at Vienna University of Technology supplied us with a network of accessible routes for pedestrians in GML2 format. They derived this model from the general map of Vienna and represented it as an undirected graph. We georeferenced each node in this graph and used the nodes as waypoints in the navigation model. For each building, we defined an address point and included it into the path network to construct a path to each address. As this route network was available in an XML-based format, a simple XSLT transformation script was sufficient to incorporate this data into the model. Furthermore, we derived annotation information, such as businesses located at certain addresses, from the general map of Vienna. This information is connected to address points in the spatial database. We manually added information concerning historic sites in the environment including multimedia annotations (text and images) because no comprehensive digital source was available. Recent trends such as Semapedia and Google Earth promise to make significant amounts of georeferenced data available through community efforts.

We found it necessary to compute the intersection of the 3D model data and the navigation graph, as we had derived the relevant input data from two nonidentical sections of the city map. We achieved this by computing a subset of the model within a given bounding box and then repairing the internal structures of the navigation graph to make sure the data was still coherent after trimming. The maintenance tool directly reads from and writes to the common data model.

Indoor tour guide

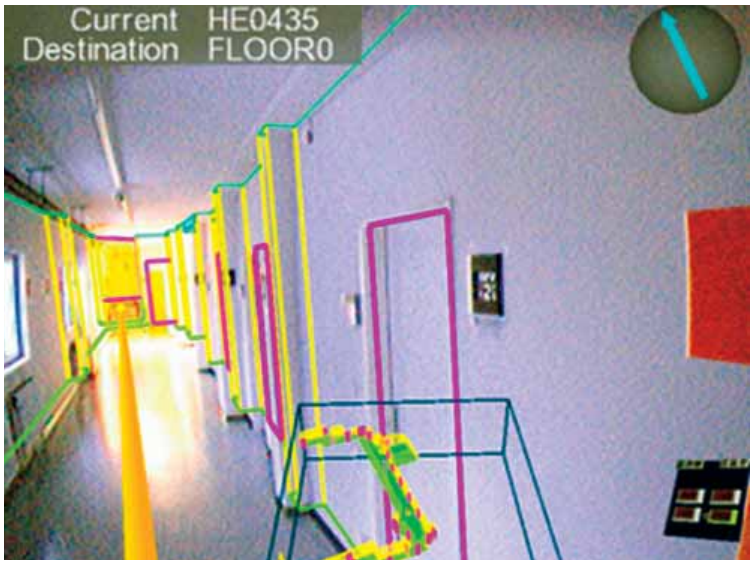
Indoor Signpost is the indoor counterpart to the previously described outdoor navigation application. It provides augmentation of the room geometry for navigation using superimposed wireframe models and transparent highlighting of relevant objects such as the suggested exit.[6]

This system provides a world-in-miniature approach, giving a zoomable bird's-eye view of the immediate surroundings. The normal mode of operation is again to choose a destination, then follow directional arrows, now using doorways and portals as waypoints. In addition, indoor Signpost displays annotations attached to individual rooms or building sections on the basis of user demand (see Figure 7 on the next page).

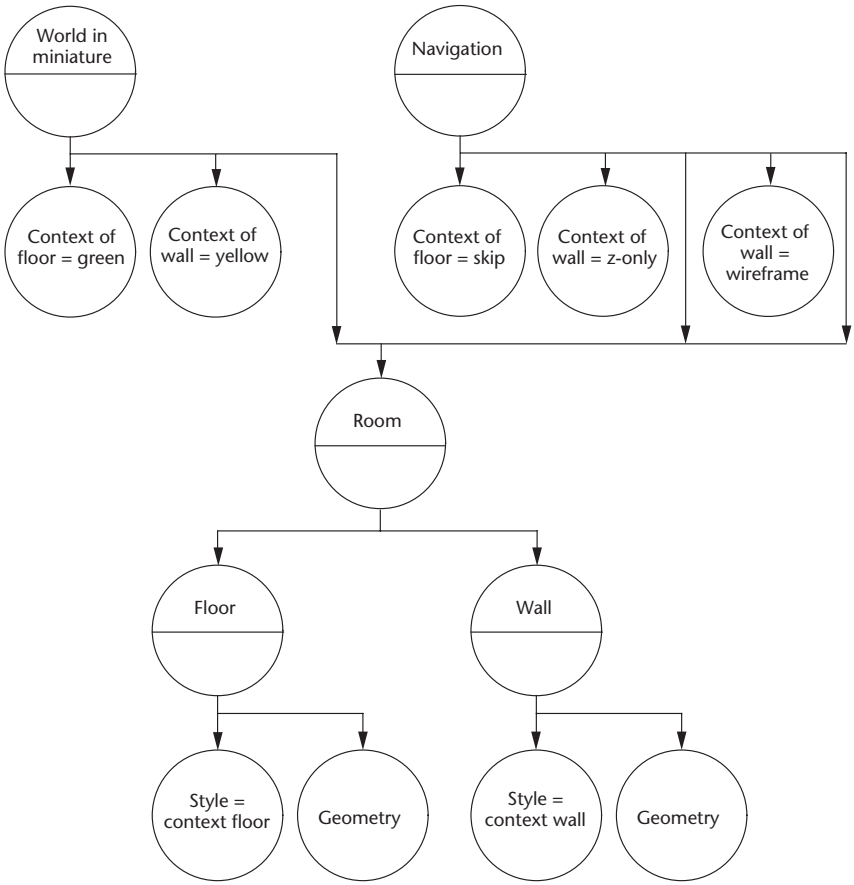
Visualization

The different visualization methods—occurring at the tier 3 (use) stage—for the wireframe augmentation, world in miniature, and navigation portal highlighting all reuse a single scene graph. A dedicated model com-

ponent provides a template scene graph for rendering to client components. These adapt the model rendering using a set of context parameters that the model component defines. The template scene graph evaluates the parameters to set rendering styles for individual parts such as walls, floor, ceiling, or doorways.



7 Indoor Signpost: the overlay shows a world in miniature and an arrow points to the suggested exit highlighted in yellow.



8 The scene graph contained in a room node uses context-sensitive traversal to select between different rendering styles. Clients of the room node set specify context parameters to their own rendering style and traverse the common scene graph.

The template scene graph contains a flat node structure corresponding to individual rooms grouped under a single building node, similar to the structure used in the Signpost information browsing application. Each room node contains different parts for wall, ceiling, floor, and portal geometry. These parts insert individual subgraphs from the context into their structure to set rendering parameters such as color, render style, transparency, and z-buffer testing.

A model component client obtains a reference to the top-level building node. It sets up context nodes to provide rendering styles for walls, floor, ceiling, and portal geometries. The client then traverses the scene graph, which results in different visual output depending on the style context setting. For example, the navigation component achieves a two-pass, color-coded, hidden line style by rendering the building node twice: once for rendering phantom geometry to the z-buffer only, and a second time for wireframe rendering (see Figure 8).

Authoring

We embedded interactive storytelling—at tier 0 (acquisition)—into indoor Signpost using the Augmented Reality Presentation and Interaction Language (April). April is an XML scripting language that supports convenient authoring of a wide range of AR applications and physical scenarios.[3] It provides elements to describe hardware (displays and tracking), content, temporal organization, and interaction.

April organizes content in relation to the visibility, location, and behavior of the objects over time. The system describes the dynamic flow of events using Unified Modeling Language (UML) state charts represented as XML Metadata Interchange (see http://www.omg.org/technology/documents/formal/xmi.htm). Driven by user input or the time, a runtime state machine transitions from one state to another one, triggered by object behaviors as described in the state chart.

We can compose applications in April using graphical tools for producing UML state charts, geometry, animations, and other multimedia files. We can encapsulate existing Studierstube (see http://www.studierstube.org/) applications as April components and use them as objects in April along with the built-in basic objects. April is therefore useful as a metatool that adds interactive content exploration to existing AR tools.

Animated guide

We used April to add an animated tour guide (see Figure 9) to indoor Signpost for storytelling and an AR puppet framework[7] for animation. The animated character pro-

vides assistance to find selected destinations and provides location-specific explanations of various rooms' contents using body gestures—for example, looking toward, pointing at, asking the user to follow, and so on—as well as 2D and 3D visual elements and sound. It appears to walk up real stairs and go through real doors and walkways, thus further enhancing integrity with the user's physical environment.

View generation

Like the outdoor application, indoor Signpost relies on view generation—taking place at tier 2 (delivery)—to translate the compact BAUML model into the necessary input files for the application subsystems. In the case of the animated tour guide, this comprises a context-sensitive scene graph including the guide animations, the storyboard control logic, and the tracking subsystem configuration. The system generates all necessary data sets using April's XSLT scripts. The animated tour guide is actually the first case requiring translation of relevant information using two XSLT stages, first translating BAUML to the indoor Signpost data, then translating this intermediate result into an April application that the AR scene graph browser can load and execute.

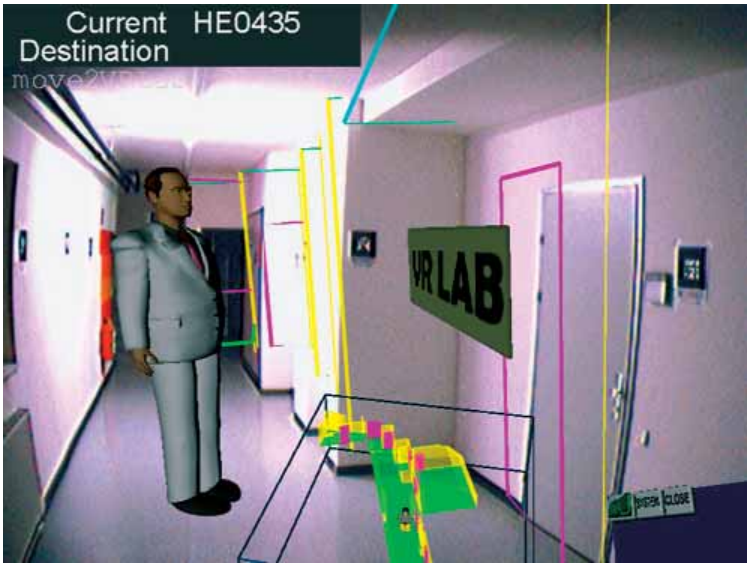
We based the tracking subsystem on OpenTracker, a tool for managing multiple trackers and processing their data.[5] OpenTracker's data flow is based on a graph configuration specified in XML. Indoor Signpost uses OpenTracker to compute a global position from the fiducial marker tracking, which reports only local coordinates. Indoor Signpost derives this data from the BAUML model so the subsystem's frames of reference can never accidentally diverge.

Surveying

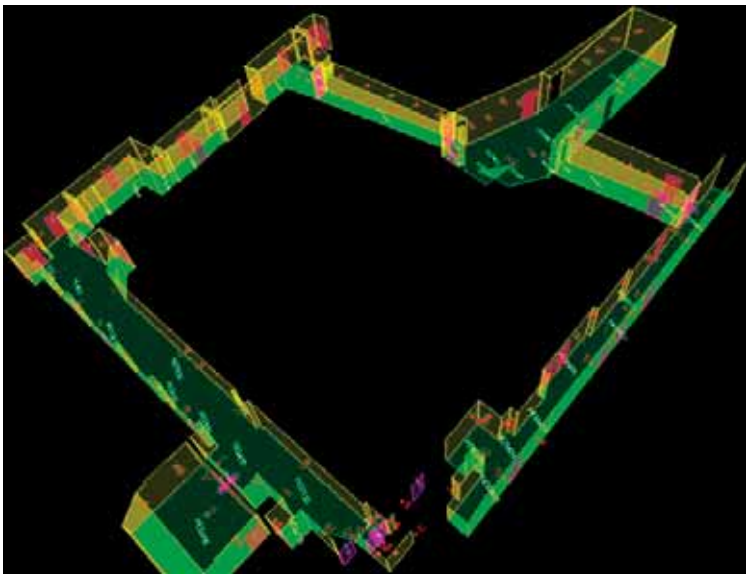
Our original intention for creating an indoor model for AR—for the tier 0 (acquisition) stage—was to work from architectural blueprints and extrude the data to obtain 3D. Unfortunately, the qualitative and quantitative inaccuracy of the blueprints proved significant enough to render the plans completely useless for our purposes, except that it taught us the difference between as-planned and as-built documentation.

Because no accurate architectural plans were available, we used a Leica Total Station TPS700 theodolite to survey 3D building geometry (walls, floor, and fiducials). We repeatedly set up the theodolite at a salient position in the middle of a building segment and measured relative angles and distances of feature points (room corners and fiducial markers). Visibility constraints limited the surveying to one room at a time. To obtain a uniform global model, we chained surveying information together by incorporating a sufficient number of overlapping points when surveying adjacent rooms.

The theodolite transmitted batches of measurements to a computer via a serial interface. A Matlab script converted the measurements into BAUML format, which we then loaded into a custom BAUML editor for immediate quality inspection. After we completed the surveying, we transformed the data into a common global reference frame. We manually classified which points constituted a floor, wall polygon, or fiducial marker.



9 The indoor tour guide helps the user navigate the building.



10 Surveying a large indoor model such as the one shown is tedious; we later used a mobile robot to perform semiautomatic surveying.

The overall procedure for obtaining a high-quality, indoor model turned out to be quite labor intensive. The model in Figure 10—which covers long hallways, approximately 20 rooms, and several hundred fiducial markers—consumed several hundred working hours. For surveying larger models, we have therefore worked on automated surveying using an ActiveMedia PeopleBot mobile robot equipped with a Sick LMS 200 laser range sensor and a 2-megapixel camera with a wide-angle lens.

The robot surveys fiducial markers using a shape-from-motion approach. It selects two nearby frames from the image sequence and calculates the essential matrix for each image pair. For every landmark, we create a 3D reconstruction using the appropriate point matches. We assume that the landmarks are planar and compute a robust plane fitting in 3D. The method's details are available elsewhere.[2]

Related Work

Typical augmented reality demonstrations work with small data sets that have been entered manually and don't require data management. Consequently, little work exists on data management techniques for large AR models.

For example, Höllerer et al. describe the use of a database and description-logic-based metadata to store a building floor model annotated with metadata for navigation target selection.[1] The sentient computing project uses a Corba runtime infrastructure to model a live environment as distributed software objects where locations and attributes of objects are updated permanently.[2]

Work on AR authoring[3] and in situ AR modeling[4,5] addresses database creation, but doesn't provide a general-purpose AR database model and isn't concerned with storage and retrieval issues. Ubiquitous computing recognizes the utility of geometric information beyond a basic notion of location as well.[6] The Nexus project specifically deals with the software architecture required for ubiquitous location-based applications and provides abstract interfaces for location data to such applications.[7] However, it doesn't focus on AR applications interacting with complex information structures.

The purpose of geographic information systems is to store and manipulate large-scale, geometric databases.[8] However, the current data sets are mostly 2D. Building high-quality functional urban 3D models is a demanding task.[9]

Recent work on AR software architectures has emphasized component-based, distributed architectures,[4,10] which help handle the increasing complexity of AR applications. This complexity stems from the fact that AR applications have diverse requirements—such as rendering, tracking, registration, and networking—that systems need to address simultaneously. Also mobile AR software has to operate in a dynamically changing environment. However, componentization only addresses the complexity of code, but not that of data. Specifically, componentization, as such, does not address the need to provide consistent views on the application data to multiple components. For example, a system might require the same geometric model for rendering and tracking. If the system can't fully duplicate the model, it will need additional data management tools. The main article focuses on such tools.

References

1. T. Höllerer et al., "Steps toward Accommodating Variable Position Tracking Accuracy in a Mobile Augmented Reality System," *Proc. 17th Int'l Joint Conf. Artificial Intelligence. Workshop Artificial Intelligence in Mobile Systems*, Morgan Kaufmann, 2001, pp. 31-37.
2. J. Newman, D. Ingram, and A. Hopper, "Augmented Reality in a Wide Area Sentient Environment," *Proc. 2nd IEEE and ACM Int'l Symp. Augmented Reality* (ISAR), IEEE CS Press, 2001, pp. 77-86.
3. S. Güven and S. Feiner, "Authoring 3D Hypermedia for Wearable Augmented and Virtual Reality," *Proc. 7th Int'l Symp. Wearable Computers* (ISWC), IEEE CS Press, 2003, pp. 118-126.
4. W. Piekarski and B. Thomas, "Tinmith-Metro: New Outdoor Techniques for Creating City Models with an Augmented Reality Wearable Computer," *Proc. 5th Int'l Symp. Wearable Computers* (ISWC), IEEE CS Press, 2001, pp. 31-38.
5. N. Navab et al., "Scene Augmentation via the Fusion of Industrial Drawings and Uncalibrated Images with a View to Markerless Calibration," *Proc. 2nd IEEE and ACM Int'l Workshop Augmented Reality* (IWAR), IEEE CS Press, 1999, pp. 125-133.
6. B. Brummit and S. Shafer, "Better Living through Geometry," *J. Personal Ubiquitous Comp*, vol. 5, no. 1, 2001.
7. K. Rothermel and A.A. Leonhardi, "Maintaining World Models for Context-Aware Applications," *Proc. 8th Int'l Conf. Advances in Communication and Control* (COMCON-8), W.R. Wells, ed., Optimization Software, 2001, pp. 235-245.
8. S. Shekhar et al., "Data Models in Geographic Information Systems," *Comm. ACM*, vol. 40, no. 4, 1997, pp. 103-111.
9. M. Lancelle and D. Fellner, "Current Issues on 3D City Models," *Proc. Image and Vision Computing*, 2004, pp. 363-369; http://www.cgz.tugraz.at/CGV/people/lancelle/3dcitymodels/3DCityModels_Lancelle_IVCNZ04_final.pdf.
10. B. MacIntyre et al., "DART: A Toolkit for Rapid Design Exploration of Augmented Reality Experiences," *Proc. ACM Symp. User Interface Software and Technology* (UIST), ACM Press, 2004, pp. 197-206.

While the pose of fiducial makers can be reliably reconstructed using automated methods, this surveying approach currently does not produce room geometry. Moreover, in this initial experiment, we manually controlled the robot using a joystick. However, we did achieve a significant speedup over the fully manual methods. We are currently working on a method for fully autonomous robot-based surveying including room geometry, and on improving accuracy by global bundle adjustment.

Future work

Work in progress addresses the migration to a novel database format firmly based on recent standards such as Geographic Markup Language version 3. This will allow us to better incorporate assets from legacy databases such as GIS systems. A particular application area that we are currently addressing is in a new project called Vidente (see http://www.vidente.at/) together with GIS solution provider Grintec to visualize artifacts that are not directly visible, such as building or subsurface infrastructure.[8] Part of this application will also perform database corrections and editions while using the mobile AR client outdoors. This poses new challenges as the inverse mapping of converting 3D user input (for example, 3D picking) back to the original GIS database must be supported. Finally, we want to extend our database to handle models of larger physical size. ■

Acknowledgments

This work was sponsored by the Austrian Science Fund FWF under contract no. Y193 and by the Österreichische Forschungsförderungsgesellschaft FFG under contract no. Bridge 811000.

References

1. D. Schmalstieg et al., "The Studierstube Augmented Reality Project," *Presence*, vol. 11, no. 1, 2002, pp. 32-54.

2. G. Schall et al., "Construction and Maintenance of Augmented Reality Environments Using a Mixture of Autonomous and Manual Surveying Techniques," *Proc. 7th Conf. Optical 3-D Measurement Techniques*, vol. 1, A. Grün and H. Kahmen, eds., 2005; http://www.icg.tu-graz.ac.at/Members/schall/optical3dpaper.pdf/download.
3. F. Ledermann and D. Schmalstieg, "April—A High Level Framework for Creating Augmented Reality Presentations," *Proc. IEEE VR*, IEEE CS Press, 2005, pp. 187-194.
4. G. Reitmayr and D. Schmalstieg, "Data Management Strategies for Mobile Augmented Reality," *Proc. Int'l Workshop Software Technology for Augmented Reality Systems*, IEEE CS Press, 2003, pp. 47-52.
5. G. Reitmayr and D. Schmalstieg, "OpenTracker—A Flexible Software Design for Three-Dimensional Interaction," *Virtual Reality*, vol. 9, no. 1, 2005, pp. 79-92.
6. G. Reitmayr and D. Schmalstieg, "Location Based Applications for Mobile Augmented Reality," *Proc. Australasian User Interface Conf.* (AUIC), IEEE CS Press, 2003, pp. 65-73.
7. I. Barakonyi, T. Psik, and D. Schmalstieg, "Agents that Talk and Hit Back: Animated Agents in Augmented Reality," *Proc. 3rd IEEE and ACM Int'l Symp. Mixed and Augmented Reality* (ISMAR), IEEE CS Press, 2004, pp. 141-150.
8. G. Schall et al., "Handheld Geospatial Augmented Reality Using Urban 3D Models," *Proc. Mobile Spatial Interaction Workshop, Computer–Human Interaction Conf.*, Springer, 2007.

Dieter Schmalstieg *is a professor of computer graphics and VR at Graz University of Technology, where he directs the Studierstube research project. His research interests include augmented reality, VR, and ubiquitous computing. Schmalstieg has Dipl.-Ing. and Dr.tech degrees from the Vienna University of Technology. Contact him at schmalstieg@icg.tugraz.at.*

Gerhard Schall *is a research assistant and PhD student at Graz University of Technology. His research interests include mobile augmented reality, pervasive computing, and wearable computing. Schall has a Dipl.-Ing. in telematics from Graz University of Technology. Contact him at schall@icg.tugraz.at.*

Daniel Wagner *is a research assistant and PhD student at Graz University of Technology where he is the main researcher behind the Handheld Augmented Reality project. His research interests include augmented reality and real-time graphics on mobile devices. Wagner has a Dipl.-Ing. in computer science from Vienna University of Technology. Contact him at wagner@icg.tugraz.at.*

István Barakonyi *is a research associate at Imagination GmbH in Vienna. His research interests include stationary and mobile augmented reality applications, embodied autonomous agents, affective computing, and man–machine interfaces. Barakonyi has an MS in computer science from the Budapest University of Technology and Economics and a Dr.techn. degree in computer science from the Vienna University of Technology. Contact him at bara@icg.tu-graz.ac.at.*

Gerhard Reitmayr *is a research associate in the Machine Intelligence Lab at the University of Cambridge. His research interests include augmented reality user interfaces, computer vision techniques for localization and interaction, and collaboration in ubiquitous computing environments. Reitmayr has Dipl.-Ing. and Dr.techn. degrees in computer science from Vienna University of Technology. Contact him at gr281@cam.ac.uk.*

Joseph Newman *is a research associate at the University of Cambridge, and is concluding a PhD at the Graz University of Technology. His research interests include the convergence of mixed and augmented reality with ubiquitous and pervasive technology. Contact him at jfn20@ cam.ac.uk.*

Florian Ledermann *is a research assistant and PhD student at the Vienna University of Technology. His research interests include augmented reality, user interface design, usability testing, and content creation for ubiquitous systems. Ledermann has a Dipl.-Ing. in computer science and design from the Vienna University of Technology. Contact him at ledermann@ims.tuwien.ac.at.*