Towards a Structured Design of Augmented Reality Applications

Arnd Vitzthum
University of Munich, Department of Computer Science, Media Informatics Group, Amalienstrasse 17, 80333 Munich, Germany
e-mail: arnd.vitzthum@ifi.lmu.de

Abstract

Mixed Reality (MR) and especially Augmented Reality (AR) technologies provide high potentials for future applications. However, a lack of concepts and tools for a structured design of AR systems can be noticed. Our approach to address this problem is a visual language for the abstract specification of AR applications, called SSIML/AR. We plan to extend this language to enable the description of relations between AR user interface elements. Such relations can provide information about the user's current actions.

CR Categories: D.2.2 [Software Engineering]: Design Tools and Techniques---Computer-aided software engineering (CASE), Object-oriented design methods, User interfaces; I.3.6 [Computer Graphics] Methodology and Techniques---Languages; H.5.1 [Information Interfaces and Presentation] Multimedia Information Systems---Artificial, augmented, and virtual realities; H.5.2. [Information Interfaces and Presentation] User Interfaces---Graphical User Interfaces.

Keywords: Augmented Reality, AR user interface, task modelling, 3D software design, scene modelling, SSIML, SSIML/AR

1 Introduction

Augmented Reality (AR) plays a key role within the field of Mixed Reality (MR). AR technologies provide high potentials for domains such as medicine or assembly, maintenance and repair. In AR, the real world around the user is enriched with virtual information. Augmented Reality Systems superimpose real objects with virtual information in 3D and in real-time [1].

The development of an AR system often poses a challenge to the developers. In the following we present three important reasons for this situation.

1. The creation of 3D content often requires the use of complex 3D authoring environments.
2. Program code and 3D content have to be integrated. Since 3D content and code are commonly developed by different authors using completely different tools, it is possible that code and content become inconsistent.
3. Real objects have to be integrated into the AR user interface. Virtual objects have to be aligned properly with real objects.

Until now, most AR research effort was spent into base technologies such as tracking and rendering. AR systems are often developed at implementation level using low-level toolkits such as the ARToolkit [2]. Thus, reuse of more complex AR software components is rare. Programming AR applications at a low level of abstraction can become an error-prone and time-consuming task, in particular if the applications become larger.

Some research projects such as the Designers Augmented Reality Toolkit (DART) [3] provide high-level AR authoring tools but concentrate on the support of non-programming experts. However, more sophisticated applications (e.g. in task focused domains such as medicine) can not be developed without deep programming knowledge. Thus, it is necessary to facilitate AR development also for programming experts. Concepts and tools for an abstract design of AR applications above the implementation level are still needed.

Unfortunately, there are only few approaches which address a structured design of AR applications. For instance, ASUR [4] is a notation for the specification of AR systems at an early stage of the development process. However, ASUR does not specify how to achieve a transition to the implementation level. The Augmented Presentation and Interaction Language (APRIL) [5] enables the definition of AR presentation flows via UML [6] statecharts. Some design aspects, such as the structure of 3D content, are not considered in APRIL.

In traditional software engineering, the de-facto standard for software design is the Unified Modeling Language (UML). However, without extensions the UML is not suitable for the design of AR systems. For instance, the UML does not provide elements which explicitly represent real physical objects (for details please refer to [7]).

2 An Approach for an Abstract Design of AR User Interfaces and Applications

Our approach to address the problems mentioned above is a visual language for the abstract pre-implementation design of AR applications. This language is called SSIML/AR [7]. SSIML/AR is based on the Scene Structure and Integration Modelling Language (SSIML) [8]. This language is an extension of the UML. SSIML models 3D content structures using a scene graph-oriented notation. It provides model elements e.g. for modelling virtual objects and groups in a virtual scene. In addition, it is possible to specify relations between application classes and the 3D scene.

Primarily, SSIML/AR extends SSIML with model elements which represent real (physical) objects. Therewith real and virtual objects can be integrated in a scene. Furthermore, SSIML/AR enables the specification of relations between scene elements (real and virtual objects) and AR system components such as tracking and rendering components. For a seamless transition to the implementation level, code can be generated from the SSIML/AR models.

In particular, SSIML/AR supports the development of AR applications from task-focused domains such as maintenance and repair. Therefore sequences of tasks the user has to perform to solve a problem can be modelled. For each task, SSIML/AR allows specifying the information which is presented to the user (see section 2.4).

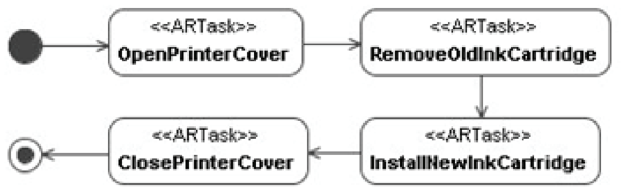

Figure 1: The sequence of user tasks in the printer cartridge installation scenario

2.1 Example Scenario

In order to illustrate the basic concepts of SSIML/AR we have chosen a scenario from the domain of maintenance. For demonstration purposes the scenario was simplified as far as possible. In the scenario the user is supported by an AR system in exchanging the cartridge of an inkjet printer. To achieve her or his goal the user has solve a sequence of tasks:

1. The user has to open the printer cover (task: *open printer cover*).
2. The user has to localize and remove the empty ink cartridge (task: *remove old ink catridge*).
3. The user has to install the new printer cartridge by attaching it to the cartridge holder (task: *install new ink catridge*).
4. The user must close the printer cover again (task: *close printer cover*).

2.2 Taskflow Model

In SSIML/AR the sequence of tasks is modelled with an UML activity diagram (figure 1). Every task is represented by an action in the activity diagram. Note that it is possible to decompose tasks hierarchically or to model optional tasks. The hierarchical decomposition is useful for more complex applications with a large number of tasks.

2.3 Scene Model

The structure of the AR user interface is modelled in a SSIML/AR *scene graph* (also called *scene model*). In contrast to implementation-level scene graph architectures such as Java3D [9], SSIML/AR allows specifying a scene at an abstract design level. Implementation details such as specific transformation values are not included into the scene model.

Figure 2 shows a screenshot of the UML tool NoMagic MagicDraw [10] (see also section 2.6 about tool integration). The scene model for the printer maintenance scenario is depicted in the lower right window in figure 2. Different types of SSIML/AR

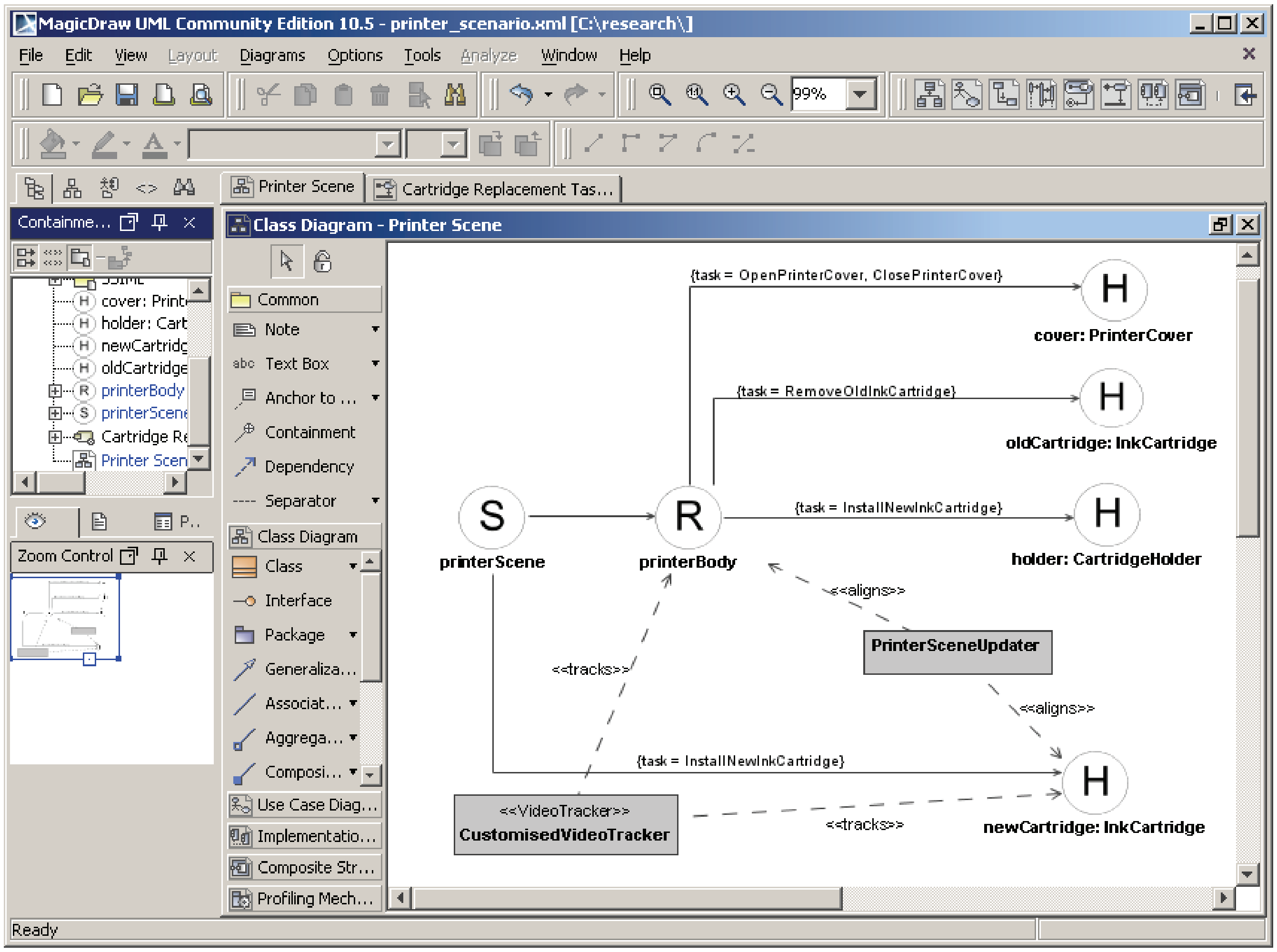

Figure 2: Screenshot of the UML tool MagicDraw [10] containing the SSIML/AR scene model for the printer cartridge installation scenario

nodes are visible. The *scene* node (S) is the root of the scene graph. This node type is adopted from SSIML which is the basis of SSIML/AR. The node with the name *printerBody* is a so-called *real object* node (R). *Real object* nodes represent physical (tangible) objects. A physical object can be manipulated by the user in the real world. For instance, the user can move the printer from one place to another. Position and orientation of a real object can be tracked by a tracking device. In the virtual three-dimensional space a real object can serve as container for other objects (i.e. a group element), although it has no visual representation (i.e. no 3D model or other virtual information is associated directly with a real object). Tracking data is mapped to the transformation values of the group element representing the real object in the virtual space.

The nodes *cover*, *oldCartridge*, *newCartridge* and *holder* represent *hybrid objects* (H). A hybrid object has a visual equivalent in the real world as well as in the virtual world. The coordinates of the real part of a hybrid object are mapped to the coordinates of the corresponding equivalent in the virtual world. For instance, the real empty ink cartridge is superimposed with a virtual 3D cartridge model. This allows directing the user's attention to the empty cartridge in the second task. The type of the 3D model follows the node's name (*InkCartridge*).

Note that there also exist node types for pure virtual objects which can be integrated into a SSIML/AR scene graph, e.g. to present textual information to the user. In addition, subgraphs of a SSIML scene model can be encapsulated in special nodes. This facilitates the management of more complex scene structures. For a detailed description of the several different SSIML and SSIML/AR node types please refer to [8] and [7].

2.4 Task-dependent Information Presentation

The information presented to the user by the AR system depends on the user's current task. For example, the information presented to support the user in locating and opening the printer cover should be different from the information which instructs the user how to remove the empty ink cartridge. Thus, besides defining the structure of the scene representing the AR user interface, SSIML/AR allows specifying which information is rendered for a special task. Therefore we provide the concept of *task-constrained edges*. If a parent and a child node are connected via a task-constrained edge, the child node will only be rendered if one of the tasks specified in the task-constrained edge is the current task of the user. For instance, in figure 2 a task-constrained edge containing the tasks *OpenPrinterCover* and *ClosePrinterCover* is modelled from the node *printerBody* to the node *cover*. This means that the virtual information associated to the hybrid object *cover* will only be rendered if the user works currently on the task *OpenPrinterCover* or *ClosePrinterCover* (see taskflow diagram in figure 1). Since a task-constrained edge may contain more than one task, an object may be rendered for more than one specific task.

2.5 Class-Node Interrelations

In SSIML/AR, two AR specific relation types between application classes (represented by UML classes) and scene nodes can be modelled: <<*tracks*>> and <<*aligns*>> relations.

A <<*tracks*>> relation between a tracker class and a real or hybrid object node expresses that the tracker class can deliver data containing position and orientation of the associated object. Such transformation information can be transferred to other application components, e.g. components which generate context data or components which are responsible for the rendering of virtual objects. In the model in figure 2, the class *Customised-VideoTracker* calculates the positions and orientations of the objects *printerBody* and *newInkCartridge*.

<<*aligns*>> relations may be specified between virtual objects and classes which update the transformation values of these objects according to the values calculated by a tracker class. An example for an <<*aligns*>> relation is the relation between the class *PrinterSceneUpdater* and the node *newCartridge* in figure 2.

2.6 Tool Integration

The realisation of SSIML/AR as UML profile (i.e. as an extension of the UML) facilitates its integration into UML tools. For instance, we have integrated SSIML/AR into the UML case tool NoMagic MagicDraw [10]. Figure 2 represents a screenshot of MagicDraw containing a SSIML/AR model. The tool integration also allows exporting SSIML/AR models into the XML Metadata Interchange (XMI) format [11]. The XMI format is supported by many UML case tools. XMI-encoded models are a suitable basis for translating the models to platform specific code using XSL Transformations (XSLT) [12]. The generation of code skeletons from SSIML/AR models eases the transition from the design to the implementation level. An approach to map the models to code is presented in [7].

3 CURRENT RESEARCH

Currently, it is not possible to specify the actions which trigger the transitions between consecutive user tasks with SSIML/AR. A simple method is to switch from one task to the next when the user presses a key. However, it would be desirable that the AR system recognizes the completion of a task automatically.

In an AR system, object relations can exist between real objects or between real and virtual objects. Object-object relations are e.g. collision or proximity. Object-object relations can trigger task transition events. For example, if an AR system recognizes a collision of two tracked physical objects it can generate a corresponding collision event. In an AR application from the domain of assembly and maintenance such an event could be used to indicate that two objects have been joined together by the user (such as the new printer cartridge and the cartridge holder in the cartridge installation scenario – see section 2).

Also, camera-object relations could be analyzed by the system to trigger transitions between tasks. This can be considered as a special case of an object-object relation since a camera is also a real object. For instance, when the user in the cartridge installation scenario opens the printer cover and the empty cartridge inside the printer is recognized by a video-based tracking system, an event could be generated in order to switch to the *RemoveOldInkCartridge* task.

Thus, a point of current interest is the specification of object-object relations which describe the interaction between the user and the AR system in the context of applications from task-focused domains. A further question is how such relations can be mapped to platform specific code.

4 SUMMARY

In this paper we underlined the potential of MR and especially AR technologies for future applications. It can be noticed that there is a lack of concepts and tools which support a pre-implementation abstract specification of AR systems. We also presented three main challenges when developing AR applications: The creation of 3D content, the integration of 3D content into applications and the integration of real physical objects into AR user interfaces.

Our approach to address the mentioned problems is a visual modelling language for the semi-formal specification of AR-

Systems: SSIML/AR. We plan to extend the language in order to allow specifying relations between real and virtual objects which are part of the AR user interface.

References

[1] Ronald T. Azuma. A Survey of Augmented Reality. *Presence: Teleoperators and Virtual Environments*, Vol. 6, No. 4, pages 355-385, August 1997.

[2] ARToolkit: http://www.hitl.washington.edu/artoolkit/

[3] Blair MacIntyre, Maribeth Gandy, Steven Dow, and Jay David Bolter. DART: A Toolkit for Rapid Design Exploration of Augmented Reality Experiences. *Proc. UIST 2004*, pages 197-206, 2004

[4] Emmanuel Dubois, Paulo Pinheiro da Silva, and Philip D. Gray. Notational Support for the Design of Augmented Reality Systems. *Proc. DSV-IS 2002,* pages 74-88, 2002

[5] Florian Ledermann. An Authoring Framework for Augmented Reality Presentations. Diploma thesis, Vienna Technical University, 2004

[6] OMG. UML Superstructure Specification, Version 2.0, 2005

[7] Arnd Vitzthum. SSIML/AR: A Visual Language for the Abstract Specification of AR User Interfaces. *Proc. 3DUI 2006*, pages 135-142, 2006

[8] Arnd Vitzthum, and Andreas Pleuß. SSIML: Designing Structure and Application Integration of 3D Scenes. *Proc. Web3D 2005*, pages 9-17, 2005

[9] Sun Microsystems. The Java 3D API Specification, Version 1.3, June 2002

[10] NoMagic MagicDraw: http://www.magicdraw.com/

[11] OMG. XML Metadata Interchange (XMI) Specification. Version 2.0, 2003

[12] W3C. XSL Transformations (XSLT), Version 1.0, 1999