Authoring of a mixed reality assembly instructor for hierarchical structures

Conference Paper · November 2003

DOI: 10.1109/ISMAR.2003.1240707 · Source: IEEE Xplore

Michael Haller
Fachhochschule Oberösterreich

Authoring of a Mixed Reality Assembly Instructor for Hierarchical Structures

Jürgen Zauner, Michael Haller, Alexander Brandl
amire@fh-hagenberg.at
Upper Austria University of Applied Sciences (MTD)
4232 Hagenberg, Austria

Werner Hartmann
wh@faw.uni-linz.ac.at
Institute for Applied Knowledge Processing (FAW)
4040 Linz, Austria

Abstract

Mixed Reality is a very useful and powerful instrument for the visualization of processes, including the assembly process. A Mixed Reality based step-by-step furniture assembly application is introduced. On the one hand context related actions are given to the user to install elements. On the other hand an intuitive way for authors to create new MR based assembly instructions is provided. Our goal is to provide a powerful, flexible and easy-to-use Authoring Wizard for assembly experts, allowing them to author their new assembly instructor for hierarchical structures. This minimizes the costs for the creation of new Mixed Reality Assembly Instructors.

1. Introduction

This is a common problem that many people have had before when assembling furniture: making sense of complicated handbooks, reference manuals or instructions. As described in [2] people simply do not have time to bother with instructions. Besides, they are just another unwanted annoyance. A great deal of inspiration for our work came from several papers and projects such as the proactive instructions for furniture assembly of [2], which showed a very innovative way that involved placing multiple sensors onto different parts of the assembly. As a result, the system as a whole could recognize user actions and thus monitor the whole assembly process. Webster et al. [3, 16] and Tang et. al. [14] have shown in their work that Augmented Reality improves the performance in assembly tasks and that computer assisted instructions are useful for complex assembly tasks. Schwald et a. presented in [12] an augmented reality system for the training and the assistance to maintenance in industrial environments. A marker based approach for assembling furniture is given in AEKI [11]. Our approach, likewise, uses Mixed Reality technology to solve assembly problems. By using ARToolKit [4], we attach reference markers to the various furniture parts thus allowing our system to determine the point and order of assembly. In today's assembly procedures the user has to switch between reading the instructions and actually assembling the elements. Mixed Reality allows a smooth combination of the real and the virtual world. Thereby, the user doesn't have to make a logical connection between the physical world and the virtual description of the instructions, which would greatly hinder him/her. A goal of this paper is to point out the easy authoring capabilities of the *Mixed Reality Assembly Instructor*. Haringer et. al. [5] described a very interesting tool, the PowerSpace. This authoring tool uses the functionality of a 2D presentation program (Microsoft PowerPoint) as the basis for the composition of 3D content. Instructions are usually described in a step-by-step procedure. Therefore, a PowerPoint approach with a slide-metaphor seems to be the right tool for authoring such an application. We use a similar approach - an authoring wizard - that allows for very intuitive usage. In addition, our authoring tool is based on the MR approach, too and it allows an interaction with the real world during the authoring process.

2. Overview

Today's assembly instructions are mostly linear, since they only describe *one* way to complete the task - or there is a limited number of ways for assembly. For beginners this might be appropriate, but for users with more experience this can be too restrictive and quite annoying. To reduce the complexity and to prevent end users from getting confused too much by a random sequence, we decided to use a straightforward approach of the assembly process. Therefore, we present only one well defined way of assembly. We support assembly instructions for hierarchical structures (as shown in figure 2). A desk for example is a hierarchical structure. Starting the assembly process we assemble the desk itself until we have to install the drawers. The assembly of these drawers must be completed first before installing them to the desk. We call elements like these drawers composed elements. Elements that can be installed without any further assembling are called simple elements.

The *Mixed Reality Assembly Instructor* is authored by another application. This application is the *Authoring Wizard* that allows a fast implementation and reconfiguration of the *Mixed Reality Assembly Instructor*. Subsection 2.1 and 2.2 give a more detailed description of the usage and the concept of our system.

2.1. Mixed Reality Assembly Instructor

Initially, our approach was an assembly application using an head mounted display (HMD); however, a tablet PC with a mounted camera will be more practical and more realistic for end users. Both approaches have their advantages and disadvantages. An HMD with its lower resolution allows the user to interact more freely and gives him/her the possibility to use his/her hands together with real tools (e.g. hammer) during assembly (cf. figure 1). In contrast, by using a tablet PC with its higher resolution the users are hindered in their possible degree of interaction, because at least one hand is required to hold the tablet PC. The choice of the device depends on the user's requirements. On the one hand we have a higher resolution and a limited interaction with the real world. On the other hand we have a lower resolution and good opportunity for interaction.

We use a mixture of traditional 2D and 3D technology in combination with Mixed Reality (including audio) to guide the user through the assembly process (cf. figure 5).

Figure 2 shows the assembly plan of a simple structure, consisting of four wooden boards (*E1'*, *E1"*, *E2'* and *E2"*). In step 1 board *E1"* is installed to board *E1'*. The result of this assembly is shown in step 2 (it is named element *E1*). Step 3 shows how two boards (*E2'* and *E2"*) have to be joined before the composed element *E2* can be mounted to the board in step 4. A fact of this plan is that there are always exactly two elements which are involved in one assembly step. We call one of them the base element, because it is the base for the installation of the other element (like the simple element *E1'* in step 1 and the composed element *E1* in step 4).

Figure 3 shows a general overview of the states that are necessary for the *Mixed Reality Assembly Instructor*. At the beginning of a structure assembly we always have to

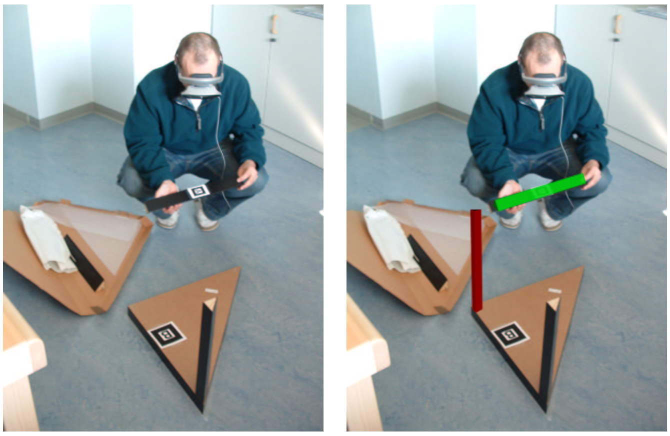

Figure 1. While in the left image the user can only see the real elements, the right image shows a scenario with augmented, highlighted virtual objects.

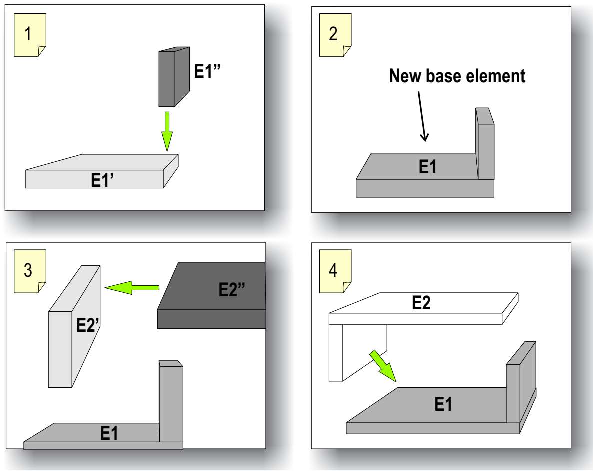

Figure 2. Steps 1, 2, 3 and 4 of the assembly instructions show how the boards are attached.

find the base element (state 1). In the example with the four boards this would be board *E1'*. When we have found it, the type of the next element has to be considered. In our case the next element is a simple element (board *E1"*), which leads us from state 2 to state 3. This board has to be placed (state 4) and installed (state 5) on board *E1'*. The installation of the board is confirmed by the user and brings us back to state 2, where we have to consider the next element's type once again. Before we proceed with the next element it is important to mention that the composed element *E1* is the

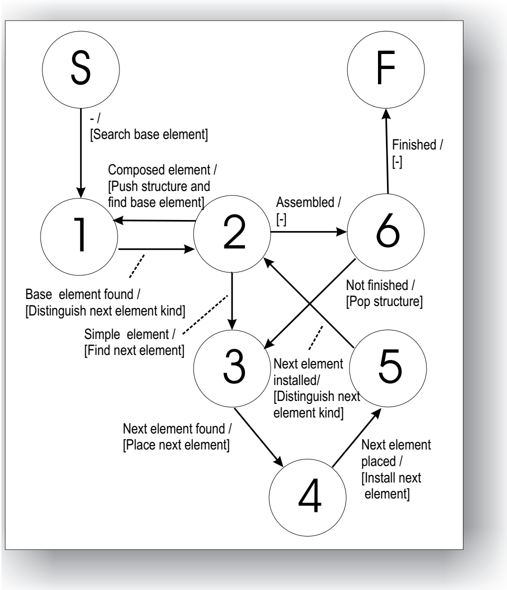

Figure 3. State diagram of the Mixed Reality Assembly Instruction.

base element after the installation of *E1"* on *E1'*.

We have described the installation of a simple element. But the next element in the example is the composed element *E2*. Therefore, we push the current structure onto a stack (stack based recursion) and return to state 1 (the base element of *E2* is *E2'*). The assembly of board *E2"* to *E2'* is done as previously shown. After the installation has been confirmed by the user, the composed element *E2* is assembled and we go to state 6. Because we have not finished the assembly of the whole structure, we have to pop the structure from the stack and proceed with the placement and installation of *E2* to *E1* (state 3, 4 and 5). Afterwards we have assembled the whole example structure (state 6) and therefore the assembly process is finished. It is worth noticing that not all states are shown in the figure (so, for example, states of lost markers are not depicted).

When the user has to find an element a 2D user interface is used, containing a textual description and a schematic overview of the desired element (cf. figure 4). To focus on the right element during the placement it is shown highlighted like the board seen in figure 5. But who tells the user how to mount it? Our approach for this problem is to augment reality with a small animation of the placement process. The augmented animation shows a path animation of the element from the current position to the position where it should be mounted. Moreover, the application automatically gives the user more detailed installation instructions (visually and via audio). Even actions of preparing the elements (adding dowels, cams, or screws and the usage of the nail-holder) are shown in this way.

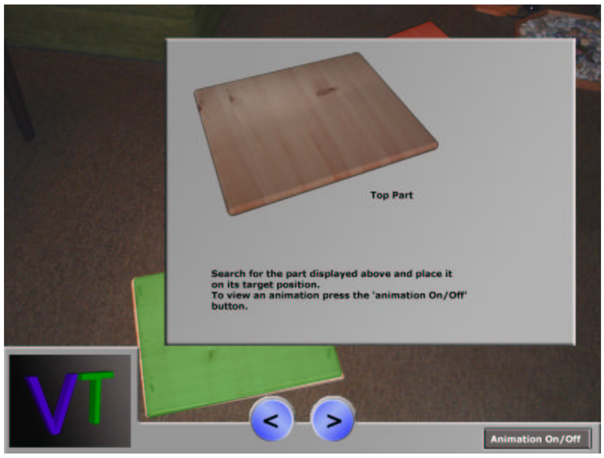

Figure 4. An intuitive 2D user interface guides the user through the assembly instructions.

Figure 5. An augmented, highlighted board helps the user to find the right boards.

Undetectable elements are often forgotten in the assembly process. As our approach is based on a marker detection system, our detection system is limited to flat surfaces. Small parts and round elements are not appropriate, but very often they are essential in the assembly process. In our case we decided to show this kind of elements relative to the other elements that have some mounted markers (cf. figure 6). Moreover, the user gets a more detailed description of what to do with the small elements (e.g. where to mount the screws etc.).

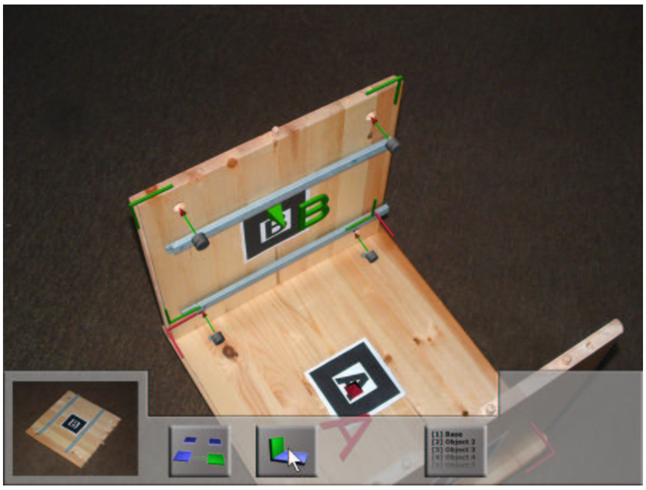

Figure 6. The author gets more detailed information about the small elements, which are mounted relative to the board.

2.2. Authoring Wizard

As already mentioned, the *Mixed Reality Assembly Instructor* can be authored in a very easy way, whenever the sequence changes. People without any programming skills but who are experts in their respective field must have an authoring tool that enables them to describe and plan an assembly procedure in Mixed Reality. Our first idea was to use a component based approach, where the authors define a set of components and the connections between them (like the GraphEdit of Microsoft's DirectShow [10]). This approach is good for experts who have experience with the component based programming paradigms. This tool, of course, is not limited to authoring Mixed Reality applications. Therefore, we call it a *generic authoring tool*. Our authors have more experience in writing assembly instructions, but in general they have no programming experience. Hence, the generic authoring tool would not be the best solution for them. Our conclusion is that a tool with a maximum flexibility and a minimum complexity is needed (the *Authoring Wizard*). It guides the author through the authoring process as described below.

In our example the author has to implement a simple MR Assembly Instructor (cf. figure 2). Once again we have four simple wooden boards (*E1'*, *E1"*, *E2'* and *E2"*). At the beginning the author can define the relationship between an ARToolKit marker and the real element. Naturally, not all 'simple' elements have to be defined at the beginning. Whenever the author creates a new element, it is stored in a repository.

Now, the author can start with the authoring process. Initially, the author has to start with the base element. With a simple 3D mouse click, the author selects one of the real elements. The first selected element automatically becomes the base element. In our example it would be board *E1'*.

Afterwards, the author picks the next element, places it (cf. placement tool of section 3.7) in its correct position/orientation and confirms this task. As mentioned before, the author can create new 'simple' elements whenever he/she wants. In the example above this could be done for *E2'* and *E2"*. Moreover, our system allows for the composing of elements. The boards *E2'* and *E2"* have to be composed and then mounted to the current base element (in our case the composed element *E1*).

Thanks to the *Authoring Wizard* we offer a very flexible and re-configurable system that allows a fast implementation of such an MR application.

3. Implementation

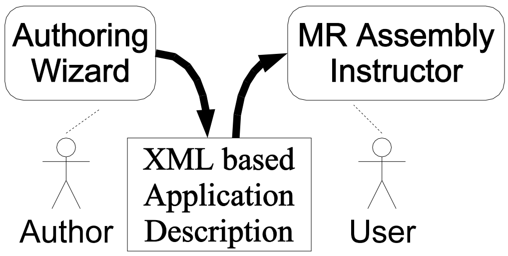

Figure 7. Relationship between the *Authoring Wizard* and the MR Assembly Instructor.

Figure 7 gives an overview of the relationship between the *Authoring Wizard* and the *MR Assembly Instructor*. The author uses the *Authoring Wizard* to create an XML based application description file of the *MR Assembly Instructor*. The chosen framework starts the application by loading this XML file.

3.1. Framework

The *MR Assembly Instructor* and *Authoring Wizard* are based on the component oriented AMIRE (Authoring MIxed REality) [1] framework which is a project funded by the EU/IST (IST 2001-34024). Short introductions to the framework concepts are given in [7] and [8]. It is designed for authoring purpose and MR applications. Some of the most important features are:

- **A generic configuration mechanism** of components by so-called properties. A property is able to represent

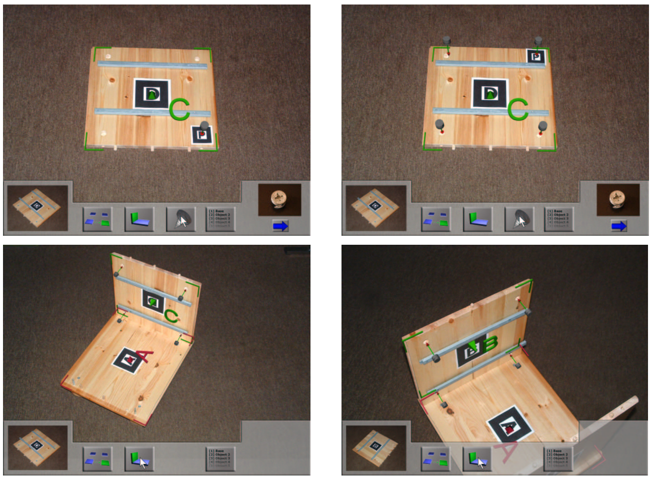

Figure 8. The expert assembles in a step-by-step approach the different parts. The undetectable and detectable elements are built together with the authoring wizard.

data of a base type and structured data. The typing of the structured properties is extendable in an object-oriented way.

- **The communication between components** is based on two kinds of slots, namely in- and out-slots. A connection between an in- and an out-slot is the only way for a component to communicate with another one. Due to this fact the interface complexity between components is reduced to a minimum.

- **The framework provides base conventions** for 2D and 3D components such as a 3D and 2D picking mechanism, a user interaction mechanism or placement and alignment. Only with these conventions is a generic authoring tool which uses MR for the authoring process possible.

- **A prototype oriented approach** is used to create new component instances. This means we have prototypes of specific components and an interface to make clones of each prototype. Two kinds of component prototypes are available. The first one is a native kind, completely written in C++ and packed into dynamic libraries such as the DLL format of Microsoft Windows systems or the shared object format of UNIX systems. The second kind is based on the existing set of prototypes and is called a composed component prototype. It consists of a component network, an export list of slots and a configuration export for the components in the network. A **composed component prototype** is handled like a native prototype. Hence the author does not see any difference between using instances of a native and a composed component prototype.

- Authoring an application without generating and compiling additional C++ sources requires the **dynamic loading of libraries**. The AMIRE framework provides an interface to load and to replace a library at runtime. Currently, two kinds of libraries are supported, namely C++ based libraries and XML based libraries of composed components.

- **Persistence** of an application is supported by an XML based file format. Such an XML file contains a list of library dependencies, the component instances and the connections between them. Furthermore, an XML format for libraries of composed component prototypes is defined.

3.2. Object detection

The underlying tracking implementation is abstracted by the definition of an object detection component. Hence, the detection of markers or the use of position and orientation information of a magnetic tracking system are not used directly. The definition describes the slots of such an object detection component as seen in figure 9:

- An object detection system must emit a boolean signal on an out-slot called *detected*. This signal reflects the detected state of an object. True is emitted when the object is detected and false otherwise.

- Furthermore, its placement relative to the camera must be emitted on an out-slot called *transformation* as an OpenGL transformation matrix, which represents the transformation between the camera and the reference point of an object. The matrix must be suitable for a projection matrix like the one created by *gluPerspective* [17].

- The definition of the object is provided by an in-slot called an *object*. The type of this in-slot depends on the implementation of the object detection system.

3.3. Marker based object detection

The currently implemented object detection is marker based and uses a modified ARToolKit. It is inspired by the marker based indoor pathfinding solution of Kalkusch et. al. [9], which uses multiple markers to track the position and orientation of a person in a building. The problem in this tracking approach is similar to tracking a structure. In both cases markers are often occluded by other parts (cf. figure

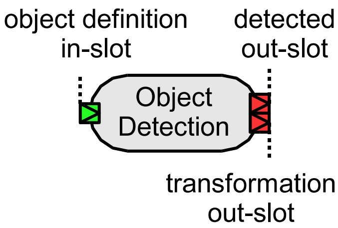

Figure 9. Component interface of an object detection system.

10). This problem is solved by using multiple markers to detect and track an object. Therefore, the marker based object detection system needs an object definition, which is given by a tuple list. Each tuple maps a marker to the relative transformation between the marker and the object reference point. This mapping is used for the calculation of the relative transformation matrix between the camera and the object's reference point. ARToolKit version 2.65 or below operates with a projection matrix rotated 180 degree around the x-axis. This of course also affects the creation of the transformation matrix between the camera and the marker. The projection matrix of ARToolKit and the transformation matrix provided for a detected marker had to be modified to implement the definition of the object detection.

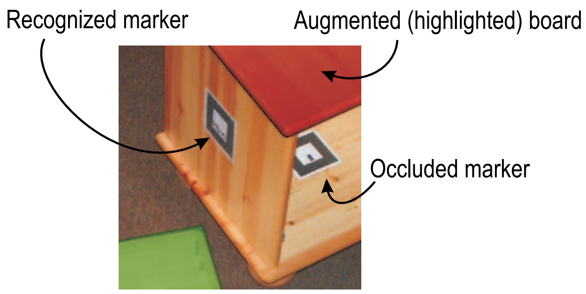

Figure 10. Partially occluded marker.

In the *MR Assembly Instructor* only two objects must be detected at once: the base element and the next element that should be installed to the base element. The object definition of the latter is constant. But the object definition of the base element depends on the elements already installed.

3.4. Data structure

The assembly plan is held in a tree structure called structural element (as shown in figure 11), containing the following:

- The description of a composed element consists of a list of structural elements. For simple elements this list is empty.
- An optional object definition for the object detection component as described in the section 3.2.
- A detectable flag indicates the availability of the above object definition.
- The element placement relative to the base element is stored as a transformation matrix which conforms to the object detection system definition.
- The geometry is described as a filename pointing to a 3DS file. It is loaded by the open source library Lib3DS [6].

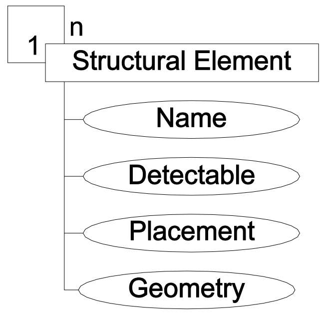

Figure 11. Hierarchical data structure of the assembly plan.

3.5. User interface components

In the following we give a short overview of the user interface components of the *MR Assembly Instructor*:

- Schematic overviews of elements and images of tools such as a hammer or a screw driver (as seen in figure 12 and 13) are implemented by using an image component, which uses the open source library DevIL [18] to load and display a configurable image.
- Textual descriptions and requests of the application and authoring wizard are implemented by using a text component. It displays an optional background and a 2D text, which is rendered by using the open source library freetype [15]. Both the text and the background are configurable.

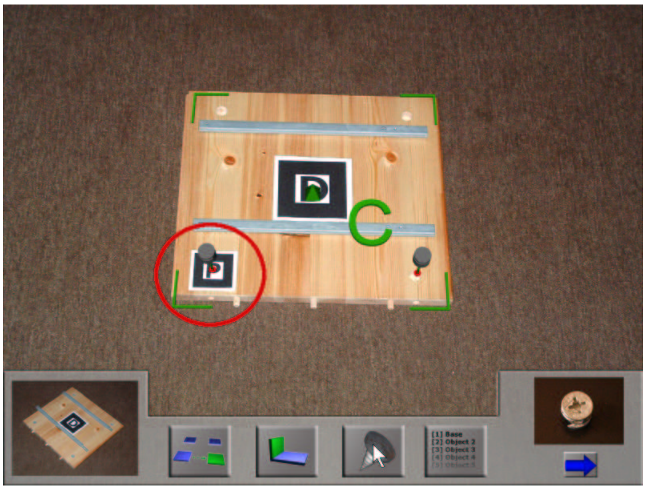

Figure 12. The author gets more detailed information about the small element.

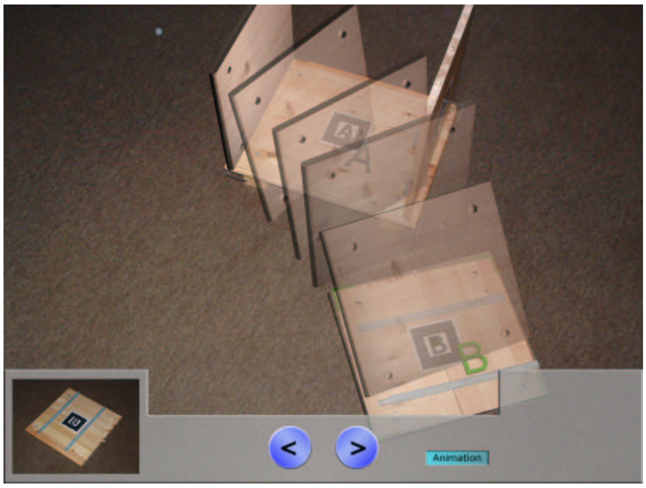

Figure 14. A small interpolation shows the correct placement of the board in a very intuitive way.

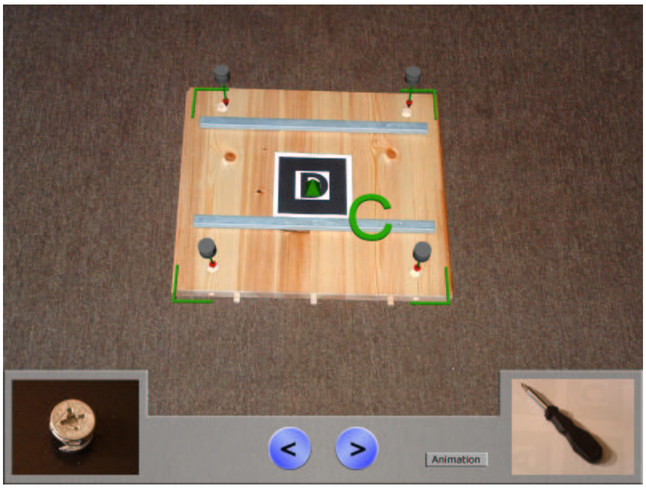

Figure 13. The author gets more detailed information about the tools and installation.

3.6. Placement animation

The animation of the placement is done by using an interpolation between two matrices (see figure 14). The first matrix represents the transformation between the camera and the base element. The second matrix is the transformation matrix between the camera and the next element. The interpolated matrix is used to display the geometry of the next element.

The interpolation between these two matrices is done by dividing the problem into an orientation interpolation and a linear position interpolation. The orientations of both matrices are converted to their quaternion representation and interpolated by the quaternion slerp interpolation, which Shoemake presented in [13].

3.7. Placement tool

The geometry for the element is created with a standard modelling tool, such as 3D Studio Max or Maya. Three functions are needed to place a geometry relative to an object reference point:

- Scaling,
- Rotation, and
- Translation.

The placement tool supports the authoring wizard with these functions. A wireless three-button mouse was integrated as an input device (figure 15). By pressing a button, the user activates a function and stores the initial orientation and position of the marker and the element until the button is pressed once again:

- The translation is activated by pressing the first button
- The second button allows for object scaling. It is proportional to the length of the difference vector between the marker's initial and actual position.
- Finally, the third button activates the rotation of an element. The marker's orientation alteration is calculated by comparing the marker's initial orientation with

Figure 15. Placing elements with the placement tool.

its actual orientation. It is directly applied to the element's initial orientation. This rotation mechanism allows the user to manipulate the element orientation without having any basic knowledge of rotation transformations. Unfortunately, a marker based solution of the placement tool has the disadvantage that the maker can become invisible. Therefore, this transformation in some cases has to be done in more than one step (e.g. turning an element upside down). Using a magnetic or an inertial tracker for the orientation of the placement tool would eliminate this problem.

3.8. Mixed Reality Assembly Instructor

The *Mixed Reality Assembly Instructor* is implemented as a component network and stored as an AMIRE application description file. This file is automatically generated by the *Authoring Wizard* when the author has finished the authoring process of the structure.

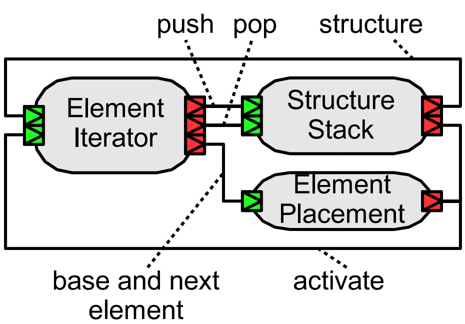

Figure 16. Simplified component network of the MR Assembly Instructor.

In figure 16 we have illustrated a subset of the component network, containing the three essential composed components, namely the *element placement*, the *structure stack* and the *element iterator component*. All three components are activated by their in-slots and become inactive when they have finished their task.

- The *element placement component* is responsible for the placement of all kind of elements, also including composed elements which were already assembled. The in-slots of this component provide the object definitions of the base element and the next element. The component network for the placement of detectable elements is shown in figure 17.

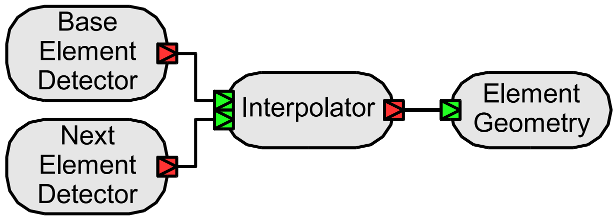

Figure 17. Animation component network.

After the placement and installation of the element an impulse is emitted on an out-slot to reactivate the *element iterator* and the *element placement component* itself becomes inactive.

- The *structure stack component* is responsible for storing the unfinished structures. It is used for the recursion of the assembly process.

- The *element iterator component* is responsible for the iteration of the tree structure. It pushes the current structure and the iteration state onto the stack and pops it back when the assembly of a composed element is finished.

3.9. Authoring Wizard

The *Authoring Wizard* has - like the *Mixed Reality Assembly Instructor* - a component network approach and it is based on the AMIRE framework.

The overview of this network illustrated in figure 18 shows the five essential composed components, namely the *activity selection*, the *element definition*, the *element repository*, the *structure container*, and the *placement editor component*.

- The *activity selection component* is responsible for what to do next. Therefore it contains user interface

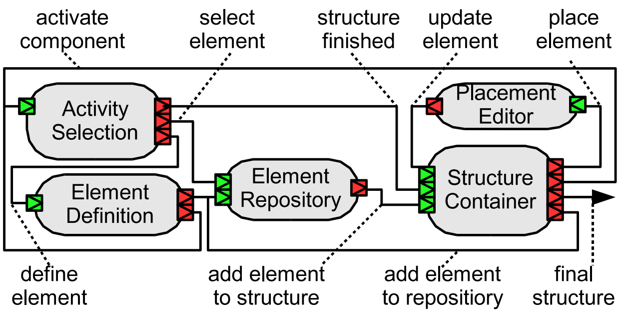

Figure 18. Simplified component network of the Authoring Wizard.

components, which allow the author to activate the definition of new elements, the selection of an existing element from the element repository and the definition of a new structure. It is activated by an activation in-slot. After the next activity has been chosen by the author, the component becomes inactive.

- The *element definition component* allows the author to define new elements by creating an object definition for the object detection system. Therefore, the implementation of this component strongly depends on the underlying object detection system. It also has to be replaced when another object detection system is used. Each new defined element is added to the element repository. Then the *activity selection component* is reactivated.

- The *element repository component* stores all types of elements. It provides one in-slot to add elements and another one to select elements. The first one is passive (it does not activate the component). But the second one activates the component and starts the selection process. The author can decide if he/she wants to select an undetectable or a detectable element. Undetectable elements are selected by choosing from a list as seen in figure 19. Detectable elements are selected by picking them with the placement tool. The selected element is added to the current structure by emitting it on an out-slot. After the selection the *element repository component* becomes inactive.

- The *structure container component* holds the current structure. It is activated either when a new element is added to the structure or when it gets the finished signal from the *activity selection component*. If the structure is finished, it is stored in the *element repository*. Otherwise, a new element is added and has to be positioned by sending the element to the *placement editor component*.

- The *placement editor component* configures the placement of the element. Thus, the author first installs the element. The placement configuration of detectable elements is calculated by using the camera relative transformation matrix of the base element and the new element. However, the configuration of undetectable elements cannot be calculated without any further tools. The placement tool is used to get a transformation matrix like the one of detectable elements. When the configuration of the placement is finished, the element in the structure is updated by the configured one.

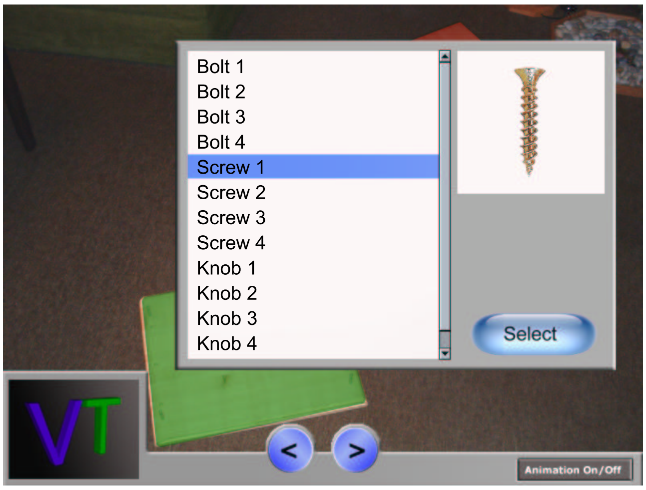

Figure 19. Author selecting an undetectable element.

4. Results and future work

The *Mixed Reality Assembly Instructor* and the *Authoring Wizard* have been tested with furniture. The user is guided step-by-step through the furniture assembly process in a very intuitive and proactive way. A misplaced element for example is virtually rotated or moved until it fits. This helps the user to avoid mistakes. The placement of markers on large flat surfaces poses no problem. Furniture, however, also contains parts without a flat surface that is large enough for markers, for example screws and tubes. Although the occlusion of markers during the assembly process is a problem, this is temporarily solved by using more markers for one element. Nevertheless, a more accurate and stable object detection system, such as a magnetic tracking system or an image based object detection system without markers, would be useful.

We came to the conclusion that the generic authoring tool (cf. 2.2) is a task for application engineers, but it seems to be too complicated to use it for non programming experts. Nevertheless, applications of a specific kind can be authored by experts in the application field by using a domain specific wizard.

In the future, it has to be shown that the *Mixed Reality Assembly Instructor* is also applicable to other application fields such as assembling large machines. Furthermore, element variations have to be considered, such as different colors or drawers with several kind of knobs. These variations could possibly be stored as a linked list. Therefore, the data structure (cf. section 3.4) has to be extended by a field containing the next element variation.

Furniture assembly is a good example to demonstrate the *Mixed Reality Assembly Instructor*, but it is not limited to this. During the assembly process of a complex machine for example it would be useful to have an assistant based on the presented *Mixed Reality Assembly Instructor* prototype. Unfortunately, such large machines normally have a lot of small and round parts, for which the placement of markers is impossible. Finally, when there is a technology object recognition that is also able to detect small and round parts, we have to show that the use of the *Mixed Reality Assembly Instructor* is possible in these fields.

5. Acknowledgements

The work described in this paper has been undertaken in the context of the AMIRE project, which consists of the following partners: C-Lab, Guggenheim Museum Bilbao, Fachhochschule Hagenberg, FAW, Fraunhofer-AGC, Labein, OMV, Talent Code and University of Helsinki. AMIRE is partly supported by the European Commission under the IST program (project 34024).

References

[1] AMIRE Consortium. Amire - authoring mixed reality, 2002. http://www.amire.net.

[2] S. Antifakos, F. Michahelles, and B. Schiele. Proactive Instructions for Furniture Assembly. In *Proceedings of the The Fourth International Conference on Ubiquitous Computing (UbiComp 2002)*, G"oteborg, Sweden, September 2002.

[3] S. Feiner, A. Webster, B. MacIntyre, and T. H"ollerer. Augmented Reality for Construction, 1996. http://www1.cs.columbia.edu/graphics/projects/arc/arc.html.

[4] H. Kato, M. Billinghurst, B. Blanding, and R. May. ARToolKit. Technical report, Hiroshima City University, December 1999.

[5] M. Haringer and H. T. Regenbrecht. A pragmatic approach to augmented reality authoring. In *Proceedings of the The First IEEE International Augmented Reality Toolkit Workshop (ART02)*, September 2002.

[6] J. E. Hoffmann. The 3d studio file format library (lib3ds), 2001. http://lib3ds.sourceforge.net/.

[7] M. Haller, J. Zauner, W. Hartmann, and T. Luckeneder. A generic framework for a training application based on mixed reality. Technical report, Upper Austria University of Applied Sciences (MTD), 2003.

[8] M. Haller, W. Hartmann, and J. Zauner. A generic framework for game development. In *Proceedings of the ACM SIGGRAPH and Eurographics Campfire*, Snowbird, Utah, USA, June 2002. ISBN 3-8322-0241-2.

[9] M. Kalkusch, T. Lidy, M. Knapp, G. Reitmayr, H. Kaufmann, and D. Schmalstieg. Structured visual markers for indoor pathfinding. In *Proceedings of the The First IEEE International Augmented Reality Toolkit Workshop (ART02)*, September 2002.

[10] Microsoft. About the directshow filter graph editor, March 2000. http://www.microsoft.com/Developer/PRODINFO/directx/.

[11] T. Pintaric. IMS Showcases Augmented Reality Technology at SIEMENS Forum Vienna, 2002. http://www.ims.tuwien.ac.at/ thomas/siemensforum.html.

[12] B. Schwald and B. de Laval. An Augmented Reality System for Training Assistance to Maintenance in the Industrial Context. In *The 11th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision'2003*, Plzen, Czech Republic, 2003.

[13] K. Shoemake. Animating rotation with quaternion curves. In *Proceedings of SIGGRAPH'85*, July 2002.

[14] A. Tang, C. Owen, F. Biocca, and W. Mou. Comparative effectiveness of augmented reality in object assembly. In *Proceedings of ACM CHI'2003*, April 2003.

[15] D. Turner. The freetype project, 2003. http://www.freetype.org/.

[16] A. Webster, S. Feiner, B. MacIntyre, W. Massie, and T. Krueger. Augmented reality in architectural construction, inspection and renovation. In *Proceedings of ASCE Third Congress on Computing in Civil Engineering*, pages 913–919, Anaheim, CA, June 1996.

[17] M. Woo, J. Neider, T. Davis, and D. Shreiner. *OpenGL reference manual: the official reference document for OpenGL, Second Edition*. Addison-Wesley Developers Press, 1997. ISBN 0-201-46140-4.

[18] D. Woods and J. Joplin. Devil - a full featured cross-platform image library, 2003. http://openil.sourceforge.net/.