Template Based Authoring for Augmented Reality based Service Scenarios

Christian Knöpfle[1], Jens Weidenhausen[1],
Laurent Chauvigné[2], Ingo Stock[2]

[1]Fraunhofer-IGD
{knoepfle,weidenh}@igd.fhg.de

[2]BMW Group
{laurent.chauvigne,ingo.stock}@bmw.de

ABSTRACT

As of today one of the major application domains for Augmented Reality is the service and maintenance of manufactured products, e.g. cars, planes, large machinery. In this area, Augmented Reality is used to support the service personnel in carrying out their repair tasks by augmenting context-sensitive, additional, virtual information in their field of view. One major problem is the creation of such information. In this paper the authors describe a concept and its realization, which heavily simplifies the creation of AR based manuals and even allows people without special AR and IT skills to carry out this task. The core idea is template based authoring: The editor of an AR based manual describes scenes at a very high and abstract level for example "remove the v-belt" or "dismantle acoustic cover of the engine". The tool transforms this description into VRML-based statements including animations (e.g. transformation of geometrical objects, use of PositionInterpolators). Furthermore the authors describe the realization of the tool and its integration in an AR environment.

CR Categories and Subject Descriptors: I.3.7, I.3.4, J.2.5
Additional Keywords: Augmented Reality, Virtual Reality, Authoring

1 INTRODUCTION

In many industrial areas it is a known fact that the products are getting more and more complex and the development time is getting shorter with each new product. A technology that heavily supported this trend was Virtual Reality.
But this trend has not only an effect on the development and production, but also on the service and maintenance of these products ("after sales"). The consequence: Training of the service personnel has to be more intensive and the documentation more extensive. Since the available time gets shorter and shorter, creation of appropriate documentation is getting more challenging. This also applies for the training. In the near future the available time will be too short to carry out these tasks in an appropriate manner.
Consequently there is a big need for a technology, which supports the service and maintenance task for complex products: Augmented Reality (AR).
For the past years, Augmented Reality[1] has matured a lot and as of today there are many companies evaluating or already using this technology in their application areas. The prominent domain is the one mentioned above, the service.
There are several AR-systems available, developed by research institutes, universities and commercial companies (HIT-Labs ArToolkit [2], the studierstube system [7] or the ARVIKA AR-Browser [[9]]). All of these systems are able to augment the user's field of view with virtual information and have a more or less decent tracking system to determine the users position and viewing direction for the correct registration between the real and virtual world.
But one of the major issues is not covered by those systems: How can content be created for augmented reality applications easily, especially for augmented reality based manuals for service tasks?
As there is a lack of such tools creation is tedious, time consuming and expensive. It is worth to point out that the existence of AR authoring tools is crucial for the success of the technology in the targeted application domain (surely in other areas too) and sooner or later not become a thing of the past.
In this paper we will describe our approach Template Based Authoring, the core concepts, the design, implementation and integration in a work environment of an authoring tool for AR applications in the area of service and maintenance assistance. Before we go into detail, we will describe the previous work in the next chapter.

2 PREVIOUS WORK

Assistance in service and manufacture tasks can be regarded as a classic topic of augmented reality applications. In the early 90s Feiner [4] presented a system that assists the user in the maintenance of a laser printer.
Caudell and Mizell [3] described an augmented reality assistance for the task of manufacturing wire bundles for aircrafts. A more recent example is the ARVIKA project [9], where augmented reality assistance applications were developed for a broad range of production and service scenarios. Albeit the problem of creating mixed reality service instructions is addressed insufficiently.
So far only few prototypical authoring systems exist that require no programming skills for authoring augmented and mixed reality content.
The MARS Authoring Tool [5] allows to associate multimedia content with geographic locations. The author, in this case a journalist, can interactively place documents in a 3d model of the real environment. Due to the diverse application domain a specific support for technical documentations is not provided.
The Approach described by Haringer [6] is based on the 2D presentation tool Microsoft PowerPoint. A technical documentation editor can compose work steps by arranging elements in 2D on PowerPoint slides. The specialized 3d editor Powerscape processes the PowerPoint XML output and allows further spatial arrangements. Although the PowerPoint slide metaphor nicely fits the workflow structure, 3d animated work steps are only supported faintly.
The AMIRE Authoring Wizard [8] is aimed to support an editor in composing assembly instructions for hierarchical structures.

The instructions itself are put together in the mixed reality environment using interactive placement tools.
We can conclude that there is a lack of authoring tools for AR based content which can be used by people without decent AR and IT skills.

3 TEMPLATE BASED AUTHORING

In this chapter the concept of template based authoring will be described. First an overview is provided and then the different components of the concept presented in more detail. At the end some implementation issues are discussed.

3.1 Analyzing real world scenarios

Our major goal was to develop an authoring tool, which can be used by people, who are familiar with traditional authoring tools and the machinery and tasks they have to describe but have very little AR and 3D graphics knowledge. Additionally we can't presume advanced programming skills.
We started with analysis of typical service tasks and tried to find a generalization or patterns, which might be helpful for design of such a tool. Typical service tasks are:

- Change oil filter
- Change battery
- Replace v- or drive belt
- Replace gear box

All of these tasks are built upon "atomic" work tasks (operations), which have to be carried out in a specific order. For example in the battery scenario, the worker has to carry out 8 atomic operations:

1. remove the earth cable from the battery negative pole
2. remove the positive cable from the battery positive pole
3. release the screws, which are fixing the battery
4. remove the battery from the engine bay
5. insert new battery to engine bay
6. fasten the screws, which are fixing the battery
7. attach the positive cable to the battery positive pole
8. attach the earth cable to the battery positive pole

In this example, one could immediately see, that most operations are similar (e.g. remove) and only differ in their parameterization (e.g. cable and pole).
To verify this observation we investigated five different and complex service scenarios, one in the interior, one in the trunk, and three in the engine compartment. The longest scenario consists of 75 operations and a mechanic needs about one hour to perform it. The goal of this scenario is to replace an adjustment unit of the engine. In total, the five service tasks comprise of 300 operations. We classified this operations and assigned them to abstract classes.
The following three classes represent the operations which are the most common ones:

Release and fastening
Nearly all parts are connected to another part by using a fastening. The necessary parameters for operations of this class are the following:

- The kind of fastening (screw, clamp, bolt, etc.)
- The tool needed for dismantling (screwdriver, ratchet, wrench, etc.).

For the Tool itself, several parameters are needed:

- The specific configuration of the tool (e.g. size of an Allen wrench)
- The initial position of the tool relative to the part
- The movement of the tool (e.g. ratchet behaves differently than a screwdriver)
- The movement of the part, when the tool is operated
- The connection point between the tool an the part

Remove and insert part
After the fastening of a specific part is released, the part can be removed. In many cases it is necessary to know how to remove the part. Therefore following parameters are important:

- The motion for removing the part (e.g. a linear motion or rotation for removing a lid)
- The axis of rotation
- The end point of the motion
- The location where the mechanic should touch the part

Unlock and lock plug
In most technical products a lot of control units are installed. These units are connected by wire to other control boxes, actuators or sensors.
The necessary parameters for this class are as follows:

- The kind of plug
- The kind of fastening (screw, clip, etc.)
- The tool needed for unlocking (e.g. hand)

Additional textual information is needed for all classes (e.g. maximum torque which should be applied to a nut).
We classified the 300 operations of the 5 scenarios and 256 of them belong to the three classes mentioned above. We had "release fastening"-type operations 69 times, "remove part" 126 times and "unlock plug" 61 times. Thus 85% of the operations were similar. They only differed in their parameterization.
We can conclude that typical service scenarios are based on atomic operations, which have to be carried out in a predefined order. Most operations are similar, they only differ in their parameterization.

3.2 The core concept

This observation built the foundation for our core concept of an authoring tool for Augmented Reality based service manuals.
Such atomic, parameterizable operations are in some way similar to the template concept in C++, thus we coined the term *template based authoring*.
It is important to note that still the most important demand is the fact that the author doesn't have to take care of the graphical representation of such an operation. Based on the information given by the author the graphics must be generated automatically. Thus the graphics must be part of the implementation of each operation / template. In the following chapter we will discuss the templates in more detail.
In order to make templates work smoothly together with objects of the 3D-scene, we added an abstraction layer to all these objects. This layer is described in chapter 3.4.
Since a single template encodes only one operation, we also need the possibility to arrange multiple operations in a temporal order or as a sequence. The arrangement and transitions between operations will be described in chapter 3.5.
The major advantage of this approach is the fact that this concept fits very well in the way professional technical writers work and think today. The major task of such an author is to think about the procedures and operations a worker has to carry out. He also defines the tools and in case of a complex disassembly path the best motion of the part. In terms of the authoring tool the user has to instantiate a template. To parameterize it and define a temporal order of the instantiated templates.

3.3 Templates

When the author defines an operation *remove screw using a screwdriver*, the graphical echo should be similar to the following:

The screwdriver is shown and placed above the screw. Then the screwdriver is animated, turning the screw upwards. After dismantling, the screw and screwdriver should vanish.

It is important to point out that the reverse operation (*screw in* of a screw) differs not only in the direction the animation is played back but also in the visibility of the objects. In the remove-operation the screw is part of the scene and visible from the very beginning. In the screw-in-scenario the screw should be added to the scene, thus is invisible until the screwdriver appears and the animation is played back.

We evaluated several other operations and for all of them animation and visibility of associated parts are depending on the template. Therefore the template is responsible for controlling animation and visibility.

To enable the template to carry out these tasks, we defined a minimal API, which has to be supported by every template:

- set_fraction(float f): This member function is called by the application and provides a float value in the range [0;1]. The animation of the parts is controlled by this value. A ‘0’ is equivalent to the beginning of the animation and a ‘1’ to the end of the animation.
- reset(bool b): This function informs the template, that the scene is set-up and will be presented to the user. For example a screw-in template will switch off the screw when the function is called.
- animate(bool b): This function informs the template, that it has to begin (b==true) or end (b==false) the animation (similar to set_fraction(0) and set_fraction(1.)). For example the screw-in template will switch on screw and tool when called with b==true.

As the template concept distinguishes between generic templates and specific parameter sets it strongly supports the idea of reusability. A programmer implements the potentially needed templates just once and stores them into a database. The editor just selects the appropriate templates from this predefined set to model a given operation or work task.

3.4 Abstraction Layer for Scene Objects

Almost every template accesses objects in the scene, (parts and tools) and performs several operations on it, e.g. animating and changing visibility. To guarantee that templates are able to operate on any object in the scene and to simplify the task of developing new templates, all objects in the scene should follow given guidelines.

To develop reasonable rules, we analyzed the different scenarios and also took into account the available data at the end user site (BMW AG). In fact we can distinguish between three types of objects:

- Parts (e.g. engine hood): A part is defined in the car coordinate system
- Standard parts (e.g. screws): A standard part is defined in its local coordinate system and is loosely connected to the part it belongs to. These objects reside in a different database than the parts
- Tools (e.g. ratchet or custom tools): Tools are defined in their local coordinate system and are stored in a third database.

First we defined the following general modeling guidelines for all objects:

- Objects must be modeled in millimeters
- Cartesian coordinate system with z as up vector is used
- All objects are stored in VRML97 format.
- Exactly one object is stored in a single VRML97 file.

There are no additional specific model rules for parts.

An important information for tools is the contact point, where a (standard) part and the tool will be connected to each other. The contact point must be at the origin of the tools local coordinate system. Any tool has to be defined accordingly (see Fig 1).

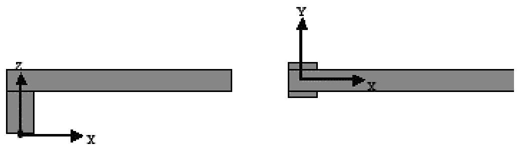


Fig. 1: Contact point of the tool

Tools also define a specific movement, e.g. a screwdriver rotates around a single axis and a ratchet has a back and forth movement. This behavior must be defined together with the tool. Since the tool movement is not implicitly identical to the movement of the connected part (e.g. the back and forth movement of a ratchet), a second animation must be defined, which is then used to move the part.

In order to animate a tool, the tool abstraction layer offers a function, which is similar to the one described in chapter 3.3. The current state of the animations, which depends on the function-value are accessible via a defined API. This allows to propagate new position and orientation values to the part, which is connected to the tool.

Until now we only had screws and connectors as standard parts. For correct interaction between standard parts and tools, the geometric model of the part must be oriented in that way that the contact point is at the origin of the parts coordinate system. The following figure illustrates this fact.

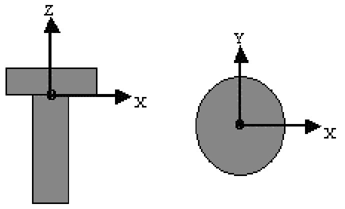


Fig. 2: Coordinate system of a standard parts

Additionally the position and orientation of each standard part in the car coordinate system must be supplied. See chapter 3.6.2 for implementation issues.

Besides the modeling rules, each part should offer a well-defined set of operations. In our current implementation the only function offered is set_visibility, which allows to hide and show the object.

3.5 Temporal Order of Operations

Another important issue is the temporal order already mentioned above. A simple approach would be that each operation is played back once and then the user has to issue the command “next” (via speech recognition or other user interface). This approach has several drawbacks in terms of usability:

When the operation is played back only once and the user didn’t understand it, he has to issue another command. This might be cumbersome for him.

Imagine a scenario where a lot of screws has to be removed. If the task is simple and no specific order has to be maintained, the user has to issue “next” a lot of times, which will certainly bother him.

Consequently we need a more sophisticated control which bears the needs of a professional mechanic in mind. We can assume that such a mechanic has good knowledge about the nature of most operations and that he is familiar with general service tasks. Thus not all tasks and operations he has to carry out are at the same level of complexity (which might as well be true for people of other professions).

In our concept we distinguish between the following entities:

- Template (Operation)

- Work step
- Task

Operations were already discussed in chapter 3.3. In the following chapters Work step and Task will be presented.

3.5.1 Work Step

A *work step* combines multiple templates and defines a time span. The sequence of the operations can be defined freely, thus each template has a starting time within the time span of the work step. This allows to play back several operations at the same time, overlapping or as a sequence. The work step is looped over and over until the user issues a command like “next”. See the following figure, which illustrates this concept.

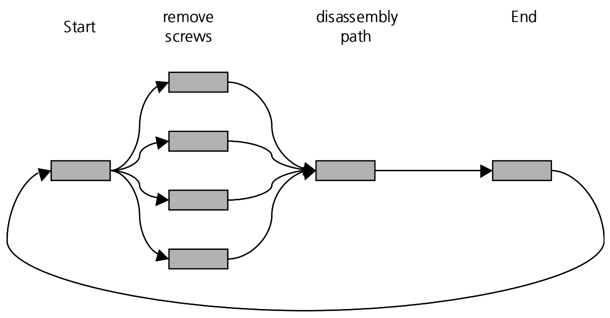


Fig. 3: A work step

A work step is used, when the operations are easy to understand and several of them can be presented to the user without overloading his cognitive abilities.

3.5.2 Task and Workflow

A task combines multiple work steps and represents a full service task e.g. replacing the gearbox. The order of the work steps is not time based but event based. In the most simplest scenario the event is the “next” command triggering the next work step to be played back (linear sequence).
But the user interaction is not limited to the “next”-command. Sometimes the user has to inform the system about the outcome of a work step and then the appropriate next work step is displayed. The following figure illustrates this fact. Work steps are represented by boxes, branches by diamond shapes.

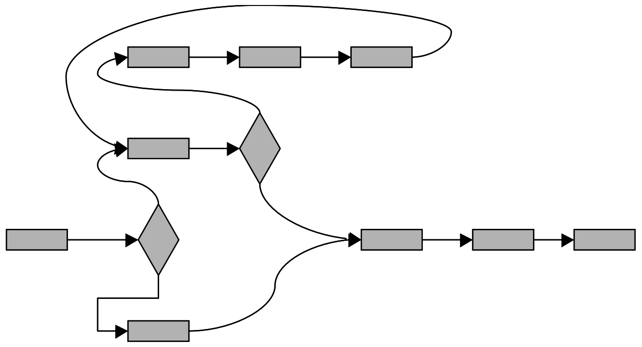


Fig. 4: Task and Workflow

Due to space constraints the workflow engine will be omitted in this paper.

3.6 Implementation issues

The implementation of the generic concept was based on our Mixed Reality System AVALON ([[10]]), which is a joint development of Fraunhofer-IGD and ZGDV e.V.. AVALON is a 100% compatible VRML97 client and offers several extensions needed for Virtual and Augmented Reality applications. This system is based on the Open Source Scenegraph OpenSG ([[11]]).
The described generic concept of templates can be easily transferred to an object oriented approach. Templates which are the actual operations (e.g. “remove a specific part with a specific tool”) are comparable to the concept of classes. They offer a specific behavior and a well-defined programming interface. Parameterization of templates (e.g. define the specific part and tool) is similar to the instantiation of classes and invocation of set-methods. Needless to say that parameters can be changed at run-time too.
VRML97 ([13]) offers mechanisms supporting this concept. In VRML97 everything is a node, which is in a object oriented way simply a class. So the idea was to use the node concept to realize templates.
To create new nodes, VRML97 offers the concept EXTERNPROTO. An EXTERNPROTO defines an interface (via named fields) and an implementation, which itself consists of VRML97 nodes including ECMAScript-nodes. Script nodes are very helpful for defining the behavior of such a new node. The advantages of EXTERNPROTO are:

- compatibility with any VRML97-browser
- extensive documentation is available
- rapid development

Of course, one could use the proprietary C++ interface of a browser, but this will result in nodes, that are not compatible to any other browser. On the other hand the C++ interface offers much more freedom and flexibility.
Our implementation of templates and the temporal order is fully based on the EXTERNPROTO mechanism. The authoring tool itself uses the C++ interface of AVALON.

3.6.1 Templates

The functions mentioned in chapter 3.3 are declared and implemented as eventIn fields. Additional parameters, which have to be provided by the user are implemented in the same way (e.g. referencing a part). Those fields are directly linked to a Script-node, where the functionality of the template resides.
For the automatic creation of the parameterization dialogue window in the authoring tool (see chapter 4.1), we additionally have to encode plaintext descriptions of the eventIn-fields. This is done by using specific MFString-fields, storing the name of the eventIn-Field and the corresponding textual description. This data is provided by the template programmer.
This template visualizes the removal of standard parts, e.g. screws, by using a tool. The only information which must be provided are the part via set_object(), the tool via set_tool() and the direction of the animation (unscrew or screw-in).

3.6.2 Parts, Standard Parts and Tools

Every object in the scene is wrapped with an EXTERNPROTO node implementing the concepts described in chapter 3.4. The actual geometry is defined through an Inline-node. Via the EXTERNPROTO interface the wrapped object can be moved around and its visibility changed. This is actually realized with a small scenegraph consisting of Switch- and Transform-nodes within the EXTERNPROTO node.

3.6.3 Temporal order

For the realization of the temporal order we implemented two different nodes. A TimeGroup node, which provides the global TimeSensor for the whole work step. Only one TimelineGroup node exists per work step. Furthermore we implemented Timeline nodes, which are associated to operations and convert the global time provided by the TimelineGroup node into the local time

(t∈[0;1]) needed by the templates. They also call the animate()-function of the templates.
The TimelineGroup node offers an interface to start and stop the animation of the whole work step. This node triggers the reset()-function of all templates of the work-step.

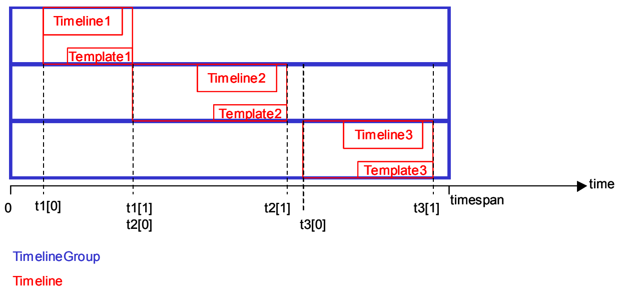


Fig. 5: Temporal order

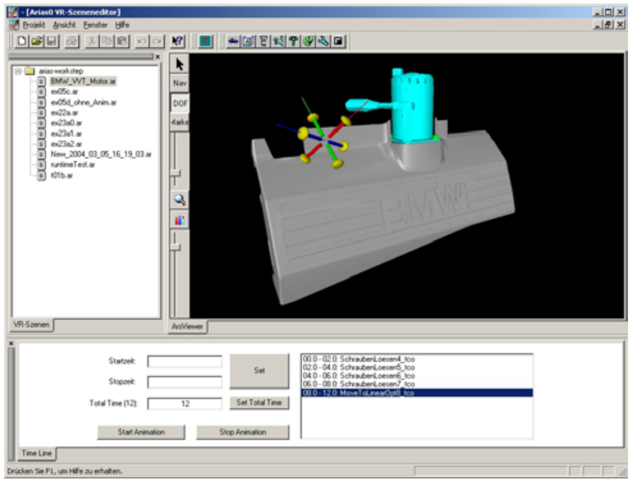
In Fig. 5 the relation between the node types is illustrated.

4 Realization

In the following we will describe the authoring tool, which is based on the concepts described above. Furthermore we introduce the run-time environment used for playing back the created AR manuals.

4.1 Authoring Environment

Obviously the concept of template based authoring can be used without a decent GUI by editing VRML-files using Parallelgraphics VRMLPad [12] or any other text editor.
But since we cannot expect that technical writers of manuals are willing to work this way, we developed a GUI, which supports the creative process of putting together Augmented Reality based manuals.
The GUI was developed under Windows 2000/XP using AVALON, Qt 3.3 and the Workbench environment provided by Siemens AG.
The GUI is divided into three parts (see Fig. 6).



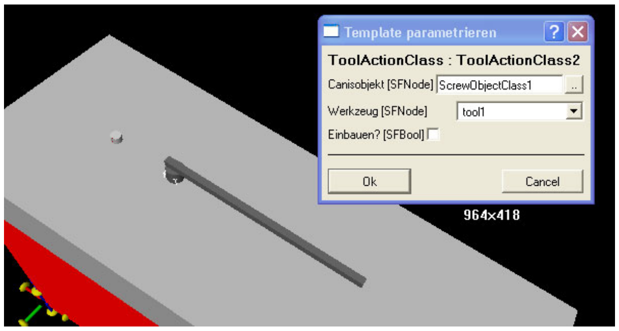
arias-work.step
BMW_VVT_Motor.ar
ex05c.ar
ex05d_ohne_Anim.ar
ex22a.ar
ex23a0.ar
ex23a1.ar
ex23a2.ar
New_2004_03_05_16_19_03.ar
runtimeTest.ar
t01b.ar

Total Time (12): 12

00.0 - 02.0: SchraubenLoesen4_tco
02.0 - 04.0: SchraubenLoesen5_tco
04.0 - 06.0: SchraubenLoesen6_tco
06.0 - 08.0: SchraubenLoesen7_tco
08.0 - 12.0: MoveToLinearOpt0_tco

Fig. 6: The Authoring tool

On the left hand side the user finds a tree-view where he can browse through the various parts and tools. By double clicking the selected part or tool is loaded into the system.
The biggest area of the GUI is covered by the AVALON viewer window. In this window the 3D scene is displayed. A simple viewer concept allows the navigation within the 3D scene. The user is able to toggle between different interaction modes (e.g. select) for this windows via buttons located at the left hand side of the window. The modes will be explained later in this chapter.
The third area in the lower part of the GUI is the timeline editor. Here the user is able to arrange the instantiated templates in a temporal order by defining start and end time of the template as well as the overall time of the whole work step.
Normally the user of the system will work the following way. First he loads in the parts and tools he will need to define the work step. Since he does the authoring in a purely virtual environment, he might load other parts, which are not relevant for the work step itself but for orientation purposes.
Then he starts adding templates to the scene by browsing the tree-view and double clicking the appropriate template name. The template is loaded into the system and the GUI opens up an automatically generated dialogue window. Here the user definable parameters of the template are displayed. In Fig. 7 one can see this dialogue window.



ToolActionClass : ToolActionClass2
Canisobjekt [SFNode] ScrewObjectClass1 ...
Werkzeug [SFNode] tool1
Einbauen? [SFBool]

Fig. 7: Parameterize a template

The information needed for constructing the window are stored in the template itself. There the authoring tool finds the name of the EventIn-field to store the value, its value type and a textual description (see chapter 3.6.1). Depending on the value type, different GUI elements are used. The mapping from value to GUI element is straightforward for most cases, except SFNode, SFRotation and SFTranslation.
In SFNode-fields objects of the scene will be stored. Here we have to distinguish between parts and standard parts on one hand and tools on the other hand.
When the AVALON window is in select mode, parts can be selected through point and click. A reference to the selected part will be copied to the dialogue window when the "accept" button is clicked. In Fig. 7 the accept-button can be found in the second line and is marked with "…".
Tools are not initially part of the scene and do not have a fixed location. Thus they cannot be selected by point and click but via a drop down box filled with all currently loaded tools.
For some templates it is necessary to define a position and rotation in space (SFRotation and SFTranslation fields), e.g. for tracking markers or disassembly paths. Of course it is possible to type in the coordinates or quaternion values, but this is not usable. Thus we developed the concept of the DOFSensor. The DOFSensor allows to define rotation and translation in space graphically. The DOFSensor is a 3D-object of the scene. The DOFSensor is the the red-green-blue object in Fig. 6. When the viewer window is in DOF mode its position and orientation can be manipulated. Clicking on an object in the scene, positions the DOFSensor at the clicked location. Clicking and dragging one of the colored tubes allows axis-aligned translation, while clicking and dragging of the yellow ellipsoids allows rotation of the DOFSensor. By clicking the accept-button of a SFTranslation or

SFRotation field in the dialogue window, the current values of the DOFSensor are copied to this field.
After the template is parameterized, start and stop time can be defined with the timeline editor. A playback functionality allows the user to preview the work step he created. In Fig. 8 one can see the result of the parameterization shown in Fig. 7.

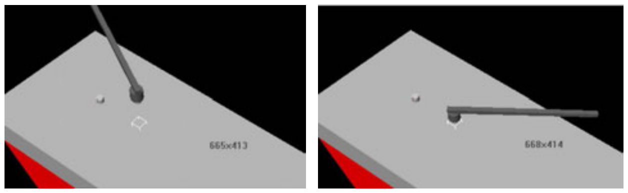


Fig. 8: Playing back a template

The authoring tool also allows to create the workflow of a service task. The workflow defines the connectivity and transition between the work steps. Due to space constraints this topic will be omitted.

4.2 Runtime Environment

The service instructions created with the authoring tool are presented to the service technician using the AR-Browser augmented reality runtime system, developed in the ARVIKA project [9]. The AR-Browser is realized as an internet browser plug in, that can be easily integrated into web-based applications. The ActiveX-Plugin is configured and controlled through its scripting interface. Hence the application logic can be implemented in JavaScript embedded in the web page. Application logic as well as the content can be easily adapted to the workers current context. The content can be retrieved from the local file system as well as from a distant server via http protocol.
The scene descriptions are handled by the AVALON VRML-engine, a fully compatible and complete VRML97 implementation. The rendering of the virtual environment and, in case of a video see-through display, the video background is performed by the open source scene-graph OpenSG [10]. The authoring tool’s output consists of a XML File describing the work environment of a particular work step. The description contains all templates and their parameterization, so the runtime systems loader can instantiate the required VRML-Templates with the appropriate parameters.
In addition to the graphical work instructions the scene description contains a suitable tracking configuration that is applied to the built in marker based video tracking. This tracking technology enables the fast and robust determination of the users position and viewing direction even on mobile hardware. Every marker is described by his position and orientation and a unique 4x4 pattern. They can be either used for the tracking of the user with respect to the environment or dynamic objects like automotive parts that are removed during the service task. While tracking the real part a phantom geometry representing the parts shape is used to secure correct occlusions between real and virtual scene parts.
The user interaction in this scenario is very simple. The technician’s attention for his real world task is not disturbed too much. By utilizing speech control he can navigate hands free through the sequence of task descriptions. Where possible the transition between task steps are triggered automatically by the detection of revealed markers or by external signals, for example generated by a car diagnosis system.

5 APPLICATION SCENARIOS

Concerning the application we focused on Augmented Reality for car service. Our test scenario was the replacement of the BMW Valvetronic-servomotor. In this chapter we first describe the authoring process and how the templates are used. Then we outline the hardware set-up for the run-time, which will be later used at the dealer's garage.

5.1 The authoring process

To realize this scenario we initially developed several templates. An important design goal was that these templates are as generic as possible so they can be reused for other scenarios too.
The first and most important template is the ConnectToolAndPart-Template, which is used for visualization of assembly / disassembly of screws using an adequate tool. The template needs as argument a tool, a standard part and a hint whether assembly or disassembly should be shown. Since the tool provides motion data for the part, the only thing the template has to do is to establish a connection between tool and part. In fact, the eventOut-field for motion data of the tool is routed to the eventIn-field for motion data of the part. Furthermore the part provides the information about its position and orientation in car coordinate system to the tool. Thus the tool can be positioned in the correct way. During animation, the tool gets the fraction-value from the template and updates the position and orientation of the tool. Then it transmits the new position and orientation of the part via routes to the part.
The second template was the LinearMotion-Template. It takes a part and a position and orientation in space as input values. Depending on the fraction-value a new position and orientation of the part is calculated. This is done via linear interpolation between start position / orientation and the provided input values.
With only those two templates the whole service task could be realized. The following screenshots show the five work steps of the repair instruction.
For working on the engine you have to remove the acoustic cover first of all. This step consists of two partial stages.
Before removing the cover several screws have to be released. The mechanic needs the information about where the screws are und which tool is necessary. This is done using the ConnectToolAndPart-Template four times and set the screws and a ratchet as input values. The templates are played back at the same time (see Fig. 9).

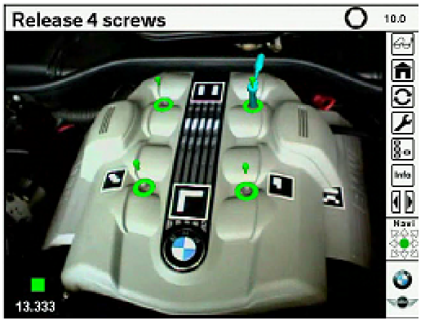


Fig. 9: Release screws

Now the cover can be removed. An augmented CAD-object and two virtual human hands show how to handle the cover the best way. Here we use the LinearMotion-Template to move the cover (see Fig. 10).

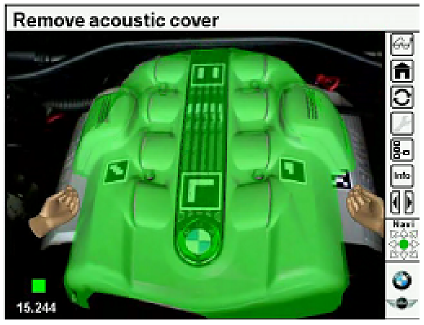


Fig. 10:Remove acoustic cover

After all covers have been removed the servomotor can be dismantled. The first step is to unlock the plug. The mechanic needs to know how the plug is fastened and in which way it can be released. This is shown by a virtual human hand. Again we use the ConnectToolAndPart-Template with the plug as part and the hand as tool (see Fig. 11).

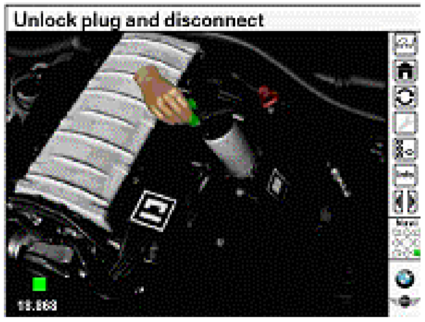


Fig. 11:Unlock plug and disconnect

The servomotor itself is fastened with four screws. The Augmented Reality based repair instruction shows the required tool, the bolt head and the positions of the four screws. Four ConnectToolAndPart-Templates are used for this step (see Fig. 12).

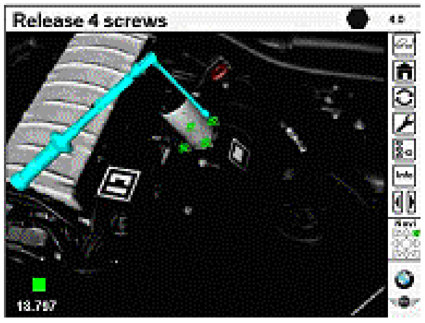


Fig. 12: Release 4 screws

Finally the servomotor can be removed by rotating until it is no longer engaged in the spline teeth of eccentric shaft. An augmented arrow points into the correct rotating direction. The animation includes an occlusion geometry permitting that only the visible part of the arrow is shown. Again ConnectToolAndPart-Template was used. Here the tool is not a real tool, but a "container", which provides rotational motion data (see Fig. 13).

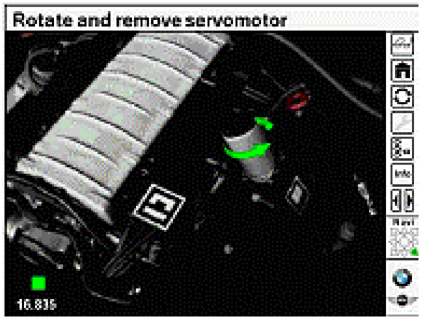


Fig. 13:Rotate and remove servomotor

After the servomotor was removed, the steps described above are played back in reverse to show the assembly.
One motivation to use this scenario as an example was the fact that two years ago it was already created by hand using standard 3D modeling and animation tools. So we were able to get an impression of the speed-up by using template based authoring.

5.2 The run-time setup

After authoring the work steps, the scenario was saved to disk and later played back using the runtime environment described in chapter 4.2.
For real world tests it is not only necessary to please the technical writers of manuals but also to ease the usage of the AR technology for the mechanics. Therefore the developers at BMW designed a very light and robust hardware set-up.
The mechanic is wearing a small camera, a very light head mounted display and a headset including a microphone and an earphone. The video and audio data is transmitted wirelessly to a computer server. The transmission is realized via radio communication, which is built in a very small and light module. All the needed technology is embedded in a BMW service jacket.
On the server itself the described runtime environment is running. The video image is directly fed to the AR browser. The audio data is sent via a speech recognition module to the AR browser too. The AR browser analyzes the image using decent tracking algorithms and calculates the position of the mechanic in relation to the detected marker. With this information the browser is able to augment the image with additional information (e.g. arrows, tools, virtual human hands etc.). It renders a new image consisting of the video background and the additional virtual objects and sends it back to the mechanic using the radio connection. Via the earphone the mechanic gets a feedback or further information like warnings.
In Fig. 14 a mechanic is shown repairing a component of the engine. He looks into the engine compartment and gets the necessary information about what to do next. Augmented objects guide him through the repair instructions.

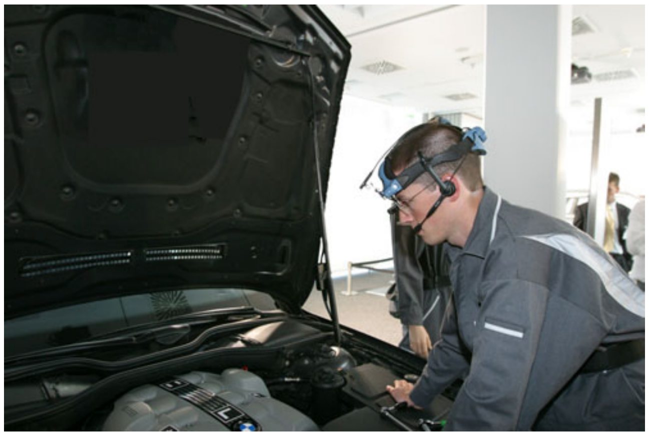


Fig. 14: Set-up on CeBIT 2004

Within the next few months we will run field tests using this kind of set-up and evaluate the usability of the technology.

6 Summary

Template based authoring can help to reduce the development time of animation based repair instructions. From our scenario we can conclude that the time needed to create an AR manual can be reduced dramatically compared to standard modeling tools. Furthermore this concept allows people without decent IT skills to create such manuals. So technical writers of traditional manuals could also be able to create AR based repair instruction by using the system we developed.

The creation of the templates still needs highly skilled IT professionals. But from our experience we can say that the number of different templates is fairly low. We guess that around 25 to 30 of them have to be developed until 95% of all service tasks within a car could be covered.

Another advantage is the fact that templates standardize the visualization of the instructions. If the developer of templates follow a given style guide, the creators of the AR manuals doesn't have to think about this issue any more. In fact they are even very limited in changing it.

Nevertheless the first prototype of the authoring tool shows some limits. This is mainly because of the restricted interface to the scene objects. For example it is currently not possible to change the transparency of parts or tools.

The actual implementation of the DOFSensor also showed some limitations. It was not very intuitive to position or orient the graphical object. Furthermore accuracy was fairly low. Since the positioning of tracking markers relies on the DOFSensor, this task was tedious to carry out too.

The developed concepts could also be used for service manuals, where still images and text are replaced by videos ("VR based manuals"). By adding an easy to use camera animation editor and video export function, one could easily create such service manuals.

In the near future we will focus our work on improving the usability of the authoring tool itself. Furthermore we will separate the object description and information from the actual VRML file and store this information in a XML-based data format.

Another important issue will be the development of a style guide for AR based manuals. Here we will have to work together with researchers from HCI and usability engineering. We believe that the insights provided by such a guide will not only be useful for AR manuals but for the whole AR community.

7 Acknowledgment

We would like to thank the whole ARIAS0 team with people from Siemens AG A&D ATS/4 in Nürnberg-Moorenbrunn, BMW AG in München and Fraunhofer-IGD in Darmstadt. Furthermore J. Behr from ZGDV e.V. for the AVALON support.

8 References

[1] Azuma, R., Baillot, Y., Behringer, R., Feiner, S., Julier, S., MacIntyre, B. Recent Advances in Augmented Reality. *IEEE Computer Graphics and Applications 21*, 6 (Nov/Dec 2001), 34-47.

[2] Billinghurst, M., Kato, H." Collaborative Augmented Reality", Communications of the ACM, July 2002, Vol. 45, No. 7, pp. 64-70.

[3] Caudell, T. P., Mizell, D. W.: "Augmented Reality: An Application Of Heads-Up Display Technology to Manual Manufacturing Processes", Proceedings of Hawaii International Conference on System Sciences, 1992, 659-669.

[4] Feiner, S., MacIntyre, B., Seligman, D. "Knowledge-Based Augmented Reality", Communications of the ACM, 36(7):53-62, 1995.

[5] Güven, S., Feiner, S. " Authoring 3D Hypermedia for Wearable Augmented and Virtual Reality", ISWC 2003. White Plains, NY, October 21-23, p. 118-126.

[6] Haringer, M., Regenbrecht, T. "A pragmatic approach to Augmented Reality Authoring", in Proceedings of ISMAR 2002, Darmstadt 2002 Pages:237 – 245

[7] Schmalstieg, D., Fuhrmann, A., Hesina, G.: "Bridging Multiple User Interfaces With Augmented Reality", In 3rd Int'l Symposium on Augmented Reality, pp 20-29, Munich, Germany, 2000

[8] Zauner J., Haller M., Brandl A., Hartmann W., "Authoring of a Mixed Reality Assembly Instructor for Hierarchical Structures", In Proceedings of ISMAR 2003, Tokyo, 2003.

[9] Friedrich, W. "ARVIKA - Augmented Reality for Development, Production and Service", In Proceedings of ISMAR 2002, Darmstadt 2002, 3-4

[10] AVALON. Avalon Project Home Page, http://www.zgdv.de/avalon/

[11] OpenSG. OpenSG Project Home Page, www.opensg.org

[12] Parallelgraphics VRMLPad product information, http://www.parallelgraphics.com/products/vrmlpad/

[13] The Virtual Reality Modeling Language, International Standard ISO/IEC 14772-1:1997, 1997