

Agents That Talk And Hit Back: Animated Agents in Augmented Reality

István Barakonyi, Thomas Psik, Dieter Schmalstieg
Vienna University of Technology
{ bara | tomp | schmalstieg } @ ims.tuwien.ac.at

Abstract

AR Puppet is a hierarchical animation framework for Augmented Reality agents, which is a novel research area combining Augmented Reality (AR), Sentient Computing and Autonomous Animated Agents into a single coherent human-computer interface paradigm. While Sentient Computing systems use the physical environment as an input channel, AR outputs virtual information superimposed on real world objects. To enhance man-machine communication with more natural and efficient information presentation, this framework adds animated agents to AR applications that make autonomous decisions based on their perception of the real environment. These agents are able to turn physical objects into interactive, responsive entities collaborating with both anthropomorphic and non-anthropomorphic virtual characters, extending AR with a previously unexplored output modality. AR Puppet explores the requirements for context-aware animated agents concerning visualization, appearance, behavior, in addition to associated technologies and application areas. A demo application with a virtual repairman collaborating with an augmented LEGO® robot illustrates our concepts.

1. Introduction

Body and facial gestures as well as speech are familiar and widely accepted means of human communication. Animated characters, often with autonomous and affective behavior, have proved to be very useful in man-machine communication since they are able to exploit and deliver information through multimodal channels and thus engage the user in a natural conversation. Autonomous agents have been actively researched in recent years as an interface to computerized systems bridging the communication gap between man and computer, and the real and virtual world. Augmented Reality (AR) applications share the same goal through enhancement of the real environment with useful virtual information, where virtual objects appear to coexist with the real world.

Software applications utilize various input and output channels for interaction (see Figure 1). Autonomous

agents are capable of making their own decisions based on their perception of the environment.

If placed in an AR scenario, these agents form a Sentient Computing system [1] since they maintain a model of the real world obtained through sensors, which influences their behavior. Using physical properties such as pose, velocity, temperature or light as input interaction channels, agents are allowed to react to changes in the environment in accordance with the users' perception. Autonomous, emergent behavior is a novel feature in AR, while awareness of real world attributes is yet unexploited by autonomous agents.

Virtual animated characters enhance human-computer interaction with natural communication symbols such as body and facial gestures. Although AR supports the collaborative involvement of real humans, "human agents" are not always available due to spatial, temporal or financial constraints, therefore synthetic agents must be used. Anthropomorphic agent representations can be particularly useful in situations where human assistance would be normally required.

While traditional desktop agents base their behavior on virtual inputs from a mouse, keyboard or by speech recognition, AR applications track and exploit qualities of the physical environment. The output channel in classic AR scenarios superimposes virtual information on top of real world objects.

We argue that real world attributes can become novel input and output communication modalities and allow new behavioral patterns for animated agents in AR environments that were not possible previously with desk-

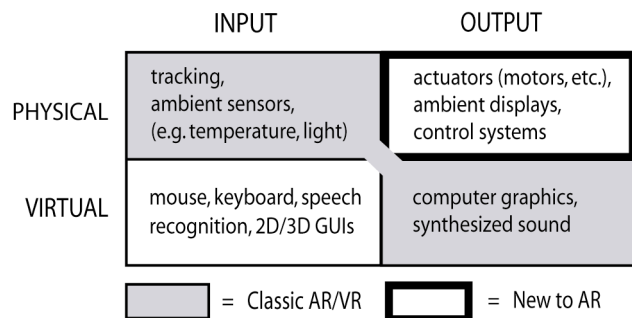


Figure 1. Input and output methods

top-based agents. By turning physical objects into interactive, responsive entities collaborating with both anthropomorphic and non-anthropomorphic virtual characters, AR can explore output channels hitherto unexploited. Physical objects become agents that not only talk back but also “hit back”.

Our research combines the aforementioned domains of AR, Autonomous Animated Agents and Sentient Computing, yielding a novel, coherent human-computer interface paradigm, which we call Augmented Reality Agents. We explore the implications of AR scenarios on animated agents concerning their appearance, behavior, application areas and associated technologies. After describing related research and our animation framework AR Puppet, we demonstrate our concepts with a sample application.

2. Related work

Virtual Reality (VR) is a more mature field than AR and has consequently already explored new interaction techniques involving animated agents, therefore it provides many useful ideas. One of the outstanding VR examples is the Jack animation system from Noma et al. [2] that allows animated virtual human figures to be used in a wide range of situations from military trainings to virtual presentations. The Improv system [3] creates a novel interactive theater experience with real-time virtual actors on a virtual stage. The autonomous pedagogical agent of Rickel and Johnson [4] called Steve operates as a virtual trainer in an immersive VR environment presenting complex interactive, educational machine maintenance scenarios.

Although users of these VR systems may have a strong sense of coexistence with virtual objects, they lack a connection to the real environment, as provided in AR systems. An early AR application providing character support is the ALIVE system [5], where a virtual animated character composited into the user's real environment responds to human body gestures on a large projection screen. This type of display separates the user's physical space from the AR environment, which demands carefully coordinated user behavior. The Welbo project [6] features an immersive setup, where an animated virtual robot assists an interior designer wearing an HMD. However, the character lacks a tangible physical representation and can only interact with virtual objects. The idea of the Steve agent recurs in an AR setting in the EU project, STAR [7], which aims to enhance service and training in real factory environments using virtual humans. Their robust machine maintenance scenario is a similar idea to our demo application, however, their

system acts as an animated guided presentation, not a responsive interactive system.

AR has started to step beyond the usual instructional and presentational domain and is now being used to explore new application fields, for which animated agents open new perspectives. MacIntyre et al. [8] make the point that a new media, such as AR, starts to gain wider public acceptance once it enters the game, art and entertainment domain. Their interactive theater experience places prerecorded video-based actors into an AR environment. The characters do not possess any autonomy, as their behavior is scripted, and interaction is limited to changing viewpoints and roles in the story. Cheok et al. [9] also experiment with Mixed Reality entertainment with live captured 3D characters, which enable real persons' telepresence in a Virtual or Augmented Reality setting but without any control of the environment. Cavazza et al. [10] place a live video avatar of a real person into a Mixed Reality setting, and interact with a digital storytelling system with body gestures and language commands.

Balcisoy et al. [11] experiment with interaction techniques with virtual humans in Mixed Reality environments, which play the role of a collaborative game partner and an assistant for prototyping machines. ARquake [12] recreates the famous first-person shooting game in a real campus setting using a mobile AR setup, where the user has to shoot virtual monsters lying in ambush behind real buildings, and uses a tangible interface to fire virtual weapons. All three systems above exploit real world properties to control the virtual world, however, physical objects always act as passive background, rather than active performers.

Animated characters and agents are appearing more and more frequently on mobile devices. The portability of a PDA or a mobile phone offers dynamic characteristics that enable agents to step out from static environments like a computer screen, and exploit mobile features such as current location and context. An early appearance of context-aware interface agents on mobile devices can be found in the work of Mase et al. [13]. They employed simple 2D characters as tour guides to deliver location-based information on portable PCs and PDAs. Gutierrez et al. [14] use a PocketPC device to control the appearance and body posture of animated 3D characters of a large VR framework through standard MPEG-4 parameters. Kruppa et al. [15] introduce a “multi-device” presentation agent that is able to “jump” from one device to another (e.g. from a PDA to a large display) to draw the attention to a certain feature. As described later, we use PDAs as multi-purpose interaction devices that serve simultaneously as platforms for agents to appear and an interface to dynamically control their behavior.

3. AR implications on animated agents

With the exception of the DART system by MacIntyre et al. [16], which is an authoring framework for AR applications enhancing a commercial multimedia authoring tool, all of the related researches are bound to a single application and technology, and lack a general approach to create a reusable framework. Moreover, none of them consider physical entities as equal, active partners of virtual characters in dialogs, as they were predominantly used as passive objects like 3D pointers, tracking aids or interaction devices.

Our main goal is to create a set of software components that allow easy enhancement of AR applications with animated agents. Additionally, we want to turn physical objects like a robot, a printer or a digital piano into context-aware, interactive responsive agents that perform various tasks with digital actors, virtual presenters and other synthetic visual elements.

Augmented Reality raises new challenges and offers novel features that can be exploited by animated agents. These implications can be divided into two categories: representation and behavior.

3.1. Representation

AR agents are embodied as three-dimensional virtual or physical objects. They share users' physical environment, in which they can freely move using all 6 degrees of freedom. Virtual agents in AR scenarios appear to have a solid, tangible body that can be observed from an arbitrary viewpoint, thus becoming integral parts of the physical environment. Virtual objects are typically animated characters but are not necessarily anthropomorphic. In some AR applications a fully fledged virtual human can be more distracting than a simple animated arrow that may communicate more information. Therefore, we experiment with various character forms.

A novel and exciting new aspect of AR agents is that physical objects like a printer, a digital piano or an interactive robot can be turned into intelligent, responsive entities that collaborate with virtual characters. If we track and monitor relevant physical attributes and process this data, attribute changes can generate events that can be interpreted by other agents and application logic. Using network packets, infrared messages, MIDI code sequences or other means of low-level communication, physical objects can not only be queried for status information but can also be controlled by external commands. Therefore physical objects act as input and output devices in AR spaces.

The combination of the real and virtual representation results in the augmented representation. This assumes the presence of a physical representation and only

superimposes necessary virtual information. Virtual agent representations may have an associated tracked physical object, which serves as a tangible control interface, while this is a prerequisite for augmented agent representations. Screenshots in Figure 2 illustrate three different representations of a LEGO robot.

The augmented representation also helps overcome the problem of correct visual occlusion. This means that we should ensure that physical objects placed between the user's viewpoint and virtual agents appear to cover parts of the virtual objects behind. This issue can be easily supported by tracking the occluding object's pose and associating it with an augmented representation that only renders an approximate virtual model into the depth buffer at the right location. Figure 3 provides illustration.



Figure 2. Physical, augmented and virtual representations of a LEGO® Mindstorms robot

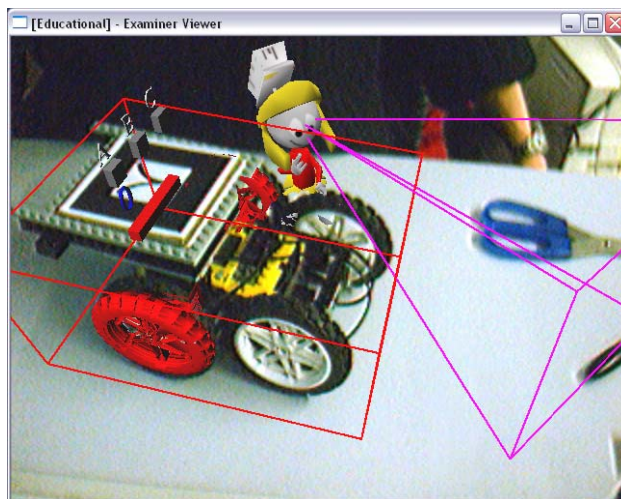


Figure 3. An augmented real LEGO robot detects collision with a virtual cartoon character (screenshot). Note the correct occlusion by the physical robot.

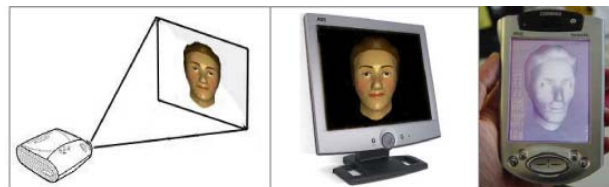


Figure 4. Virtual presenter agent appearing on different displays and devices

In AR scenarios users are mobile, traveling between different physical locations and hardware setups, therefore they require cross-platform, mobile assistants. AR agents can “live” on several devices and displays, like HMDs, projection screens, PDAs (see the talking head agent in Figure 4) or more recently mobile phones. Each has its own local coordinate system placed into the global coordinate system of the user’s physical environment. AR agents are able to smoothly travel between devices and coordinate systems.

3.2. Behavior

An AR agent interacts in real-time with other agents in the same or remote AR spaces, with users working with collaborative applications, and the applications they are embedded into. In addition to their capability of executing scripts, they possess a certain autonomy, which means they watch and automatically react to changes in the properties of AR spatial objects.

The scheme represented in Figure 5 depicts the interaction flow of autonomous agents within an AR scenario. The agent monitors the physical and virtual world, facilitated by physical and virtual sensors. While processing the information from light, push, angular or temperature sensors is obvious, implementing a perception capacity for virtual entities is non-trivial. Examples of virtual sensors include the following:

Agents can “see” users or other agents when their bounding box intersects with the viewing volume associated with the agent’s virtual eyes, which may be a single frustum or several frustums, a box, or even unbounded space. Once in the viewport, agents start observing the position, orientation and hence velocity of users, interaction devices, agents and other physical and virtual objects. AR applications are also associated with a physical position and orientation since their working volume usually augments only a subspace of the real environment.

The agent can “hear” a sound object if the sound source’s propagation volume intersects with the agent’s hearing volume (typically spheres).

Touching is modeled by collision detection, therefore we need to properly calculate bounding volumes for both physical and virtual items. Figure 3 shows the visualization of a virtual touch sensor for a real object (see the bounding box of the LEGO robot) and the virtual eyesight of a virtual cartoon character (see the wireframe viewing frustum). Some physical entities such as displays may not make use of a precise bounding box but instead a predefined “hotspot” area not related to physical boundaries, which triggers events once something is inside.

Agents can be equipped with **object and application-specific sensors** that examine application attributes, GUI input, and internal state information of virtual objects (e.g. the emotional state of a virtual human) and physical objects (e.g. an error message of a printer).

Perception is followed by processing of incoming information. With the assistance of an internal simulation model, the agent performs actions in response to input events. Traditional multimodal output channels can be opened between users and agents such as non-verbal communication (facial and body gestures), speech synthesis and recognition. However, AR offers novel, compelling modalities involving pose, velocity and status information of objects. The physical location agents inhabit, the direction they are looking into and the object they control all convey important context. These new modalities enable a wide range of new behavioral patterns, such as the following:

1.) The user places a character into the physical working volume of an application. The character receives an event with the identity of the user and the application, and loads the user’s application-specific profile and the state in which she last left the application. The character continues to work with this application.

2.) A virtual presenter is working with a user in an immersive AR setup and wears an HMD. She decides to work in another room with a projection screen suitable for a larger audience. She takes a pose-tracked PDA, moves it close to the character and “picks it up”. The character continues to “live” on the PDA screen until it is carried over to the projection screen in the other room. It then becomes aware of the new environment and jumps to the projection screen, where the same application is running, maintaining the state of the user’s work.

3.) A machine in a large PC cluster starts malfunctioning. Firstly, a virtual repairman character identifies the computer in the cluster room, then leads the human operator to the physical location. Once in the vicinity, the repairman points out the possible sources of error on the machine itself. An explanation is only begun once the operator looks at the repairman, in order to ensure appropriate attention and focus. The machine sends feedback to the repairman when it is back to its normal state.

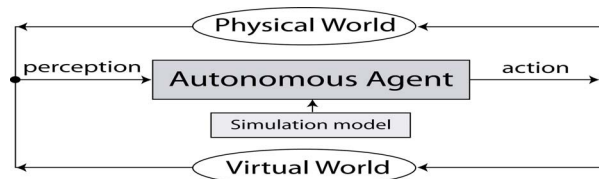


Figure 5. Autonomous behavior scheme

4. AR Puppet framework

One of the authors had the chance to admire the famous water puppet theater (“mua roi nuoc”) in Hanoi, Vietnam [17], where exceptionally skilled puppeteers animated a group of puppets using hidden, underwater controls while performing Vietnamese legends. Although the puppeteers focused only on their own controlled puppets, they always stayed perfectly synchronized since they followed well-prepared instructions from a choreographer. The choreographer received instructions from the director, who actually breathed life into the legends told by storytellers.

In digital storytelling it is common to use a hierarchical structure similar to that used in a theater [18] since these terms, which often represent complex system components, are familiar even to non-technical people. Although the comparison is not novel, we found that tasks to control AR agents can be divided into discrete groups which closely match the layers of a puppet theater’s multilevel structure. We therefore borrowed the stage metaphor for AR spaces, story metaphor for applications, puppet metaphor for AR agents and puppeteer, choreographer and director metaphors for various control logics. Interaction is performed by the storyteller, who also assures that the story proceeds in the desired direction. These components build up our hierarchical animation framework (see Figure 6), which we call AR Puppet. Each component’s role in controlling our agents is now briefly explained.

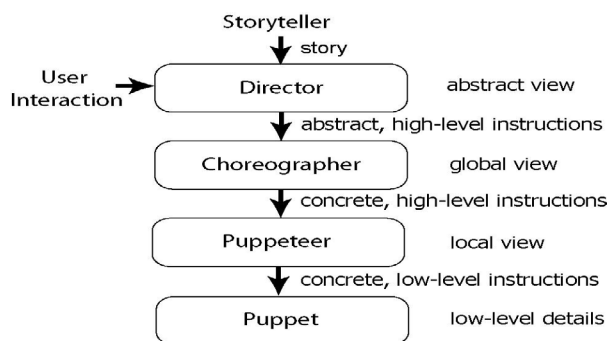


Figure 6. Overview of the AR Puppet framework

4.1. Puppet

The bottommost component is the puppet level. A puppet stands for one representation of an AR agent, which, as described in Section 3.1, can be physical, virtual or augmented, and may appear on various platforms ranging from an HMD to a PDA. Although they all may require different implementation, low-level

communication and visualization (e.g. level of detail), they belong together since they refer to the same entity, namely an AR agent.

We created some animated agents to experiment with the framework including an affective facial agent, a character based on the Quake2 game’s MD2 format, an augmented LEGO Mindstorms robot and a skeleton-based animated character built on the open source library Cal3D [19]. The latter character is capable of the import and display of high-quality animation exported from 3D Studio MAX’s Character Studio, allowing for unlimited animation possibilities. It also allows direct access to bones, which permits inverse kinematics and the linking of objects to joints, for example to pick up and carry an object.

4.2. Puppeteer

On the next level the puppeteer is the component that groups puppets together and controls a selected set of agent representations at the same time. It knows exactly “which strings to pull”, that is how to implement higher-level instructions for each puppet to obtain the desired effect. The puppeteer has the following responsibilities:

- Providing a **unified command interface**, which allows scripting of physical and virtual objects. A default implementation is provided for a predefined set of commands like go to, turn to, point at, etc., which can be overloaded by derived objects.
- Customization of default parameters of **virtual sensors** (e.g. viewing frustum parameters, hearing sphere radius, bounding volume).
- Support for **new tracking modalities** like pose and internal status
- Support for **idle behavior**
- **Command adaptation**

Command adaptation means that high-level puppeteer commands need to be tailored to the capabilities of its puppets. This is necessary in the following cases:

Switching between representations: If one agent representation becomes unavailable, it needs to be turned off and another (or several) turned on. If the physical representation of a machine is broken or malfunctioning, we can switch to its virtual representation simulating the appearance and behavior of the real object.

Communication: Commands have to be transmitted to puppets, which requires low-level communication management (open/close/recover connection, send/receive data) with mobile and remote devices using various protocols (TCP/IP, IrDA, MIDI etc.).

Device adaptation: A character has more freedom to move around when it appears on an HMD than on the screen of a mobile device, therefore certain motions cannot be performed. Instead of moving and pointing to a 3D location, a PDA agent could just give a visual or audio hint about its whereabouts.

Animation parameter adaptation: A high-level motion command requires adaptation of animation parameters. For instance if a character has to move to a distant location but a walking animation would appear unnatural within the allowed time interval, a running or flying animation sequence should be triggered.

Motion constraint adaptation: The puppeteer facilitates the puppets' adaptation to motion constraints like terrain and path following.

4.3. Choreographer

While puppeteers focus only on their respective puppets, the choreographer has a general overview of all puppeteers and their attributes. This level does not deal with character-specific details but uses high-level commands like "go to my printer and point at the paper tray that has become empty". In a highly dynamic environment such as AR users move around and work with different applications, objects are displaced and devices may malfunction, low-level information like absolute coordinates and internal status constantly change. Consequently this dynamic information needs to be hidden from users and applications, as they deal solely with abstract names and spatial references. The tasks of the choreographer are centered around resolving these references:

- **Command parsing and substitution:** The component parses object attribute references in an *object.attribute* format, which substitutes the current value in the command sent to puppeteers. For example: "go near printer.position, point at printer.lastError.position"
- **Motion planning:** The choreographer is aware of all objects between the source and target of a moving agent, therefore it is able to plan the motion to avoid these obstacles.
- **Feedback for synchronization:** If feedback is sent whenever a group of puppeteers finishes command execution, multiple agents in the same application can wait for one another, thus maintaining synchronization. For example a virtual repairman stops drawing attention to the printer's paper tray once as it is refilled.

4.4. Director

The director represents the level of application logic and interaction. This component drives the "story", the application, forward based on choreographer events and feedback, user interaction and scripted behavior. Although all agents can be configured statically before running an application, in an AR scenario we are aiming at dynamic configuration.

An innovative way to configure AR agents is using the **Personal Universal Controller (PUC)** [20]. All agents need to provide an XML-based description of their relevant, configurable attributes and their supported commands together with the command syntax. All characters run a PUC service, they are listening to incoming connections from PUC clients. A PUC client is able to render a Graphical User Interface (GUI) to control attributes that are described in the description, and are implemented on various devices and platforms including PCs, PDAs and smartphones. Figure 7 shows a control GUI rendered on a PocketPC. By checking the "skeleton" control checkbox (marked with an ellipsoid) the user changes the rendering mode from mesh to skeleton mode allowing observation of the underlying bone structure. One PUC service can accept multiple clients, which allows collaborative, multi-user configuration.

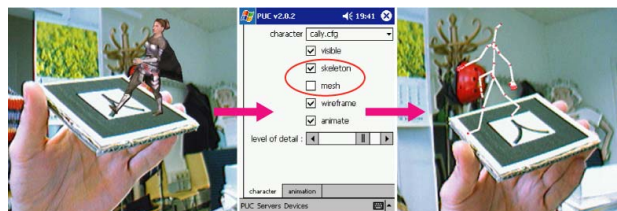


Figure 7. Animated character bound to a tangible optical marker and controlled by a PUC generated interface on a PocketPC (monitor + PDA screenshots)

Mobile devices implementing the PUC technology provide an intuitive way to configure AR agents as the user can simply walk up to an agent in her physical environment, and connect to it in order to query its PUC description, and then tweak attributes using the GUI that has been generated on the fly. If the character exposes the commands it understands together with their syntax in the XML description, a template can be generated with which the user can directly test the effect of script commands.

4.5. Storyteller / Author

The storyteller is a meta-component representing the author creating the application or story, and has only an abstract view of the components and the story flow.

4.6. Integration with applications

It has always been a challenge for interface agents to monitor the current state of the application in which they are embedded, and the behavior of the users they are interacting with, without modifying the application itself. One of the most powerful aspects of the AR Puppet framework is the easy way current applications and users can be monitored, which is grounded in the architecture of our AR platform Studierstube [21], a middleware allowing for a wide range of distributed collaborative multi-user AR applications.

Both Studierstube and AR Puppet have been built on the multiplatform high-level 3D graphics API Open Inventor. Inventor is based on a scene graph database, where all entities are scene graph objects interacting with one another via input and output attributes or “fields”. AR applications, users and components of the AR Puppet are all parts of the same hierarchical scene graph, therefore they can monitor one another’s fields and immediately respond to changes. A disadvantage of our AR platform is the lack of support for legacy applications. Interfaces to legacy applications must be implemented on a case-by-case basis.

Users have fixed, standard attributes like current tracked pose and display type. Their actions can be monitored through tracked interaction devices. Animated agents in the AR Puppet framework are thus always aware of the application users’ pose and behavior.

Applications and agents should be prepared to interact with one another. If an application is to have the possibility to host agents, then it must supply them with functionality to exist:

- **Dynamic addition and mobility of agents**

A choreographer component has to be added to the scene graph, which can dynamically add/remove, enable/disable agent representations supporting mobility.

- **Monitoring the application’s current state**

Agents need to be aware of the current state of applications. This internal state has to be mapped to a vector of attribute values, which can be easily observed by agents and the director component. Therefore we need to mark relevant attributes in the application’s scene graph with a special tag, which is a quick and mechanical task in Open Inventor. When placed into an application, the agent can easily retrieve and query the marked attributes. Figure 8 shows the mark-up of our demo application for an agent.

5. AR Lego – a pilot application

Our pilot application demonstrates the features of AR Puppet and implements a traditional AR scenario:

machine maintenance with a remote operator. Two AR agents are employed to educate an untrained user to assemble, test and maintain machines composed of active (engines and sensors) and passive (cogwheels, gears, frames) parts. The two agents are a real, augmented LEGO Mindstorms robot and a virtual repairman. An expert located at a remote site communicates with the local user and monitors her progress with the help of our in-house AR videoconferencing system described in [22], which serves as a remote augmented window onto the user’s physical environment. This window displays a live video stream of a tracked camera placed in the user’s workspace and renders audio captured by a microphone. Audio and video data are transmitted to the remote expert via the videoconferencing module. Tracking data used to render virtual content on the local user’s display is sent separately, so that the repairman character and virtual LEGO tiles are correctly superimposed on top of the live video stream on the remote side as well. The local user can freely move the camera around, giving different viewpoints for observation.

The remote operator has no direct control over the local physical environment, as her presence is only virtual. This causes difficulties in correctly identifying and highlighting objects in the local user’s workspace. Reference to objects in the real environment can be achieved either by using an awkward 3D pointer or by the user’s active resolution of ambiguous verbal spatial references like “mount the large cogwheel on right of the front axis” or “turn the robot around and trigger the leftmost push sensor”. As pointed out by Pfeiffer et al. [23], humans use several competing reference frames in task-oriented dialogs including a user-oriented, communication partner-oriented and object feature-oriented view. This ambiguity imposes undesired and tedious synchronization between the two users.

5.1. Virtual repairman

Animated AR agents are a good choice for solving this issue, since they can serve as autonomous telepointers and explain about other objects with expressive, natural gestures in the vicinity of significant physical locations. By monitoring the attributes of the LEGO assembly application’s interface object (see Figure 8), the AR Puppet framework is able to automatically generate behavior for the animated virtual repairman (see Figure 9).

The assembly application’s interface object provides the agent with all the relevant information about the next building block to be mounted. It outputs the current construction step so that the agent is aware of the user’s progress, i.e. what was and needs to be constructed. Based on relevant information the appropriate LEGO block is generated, which is then linked to the agent’s hands.

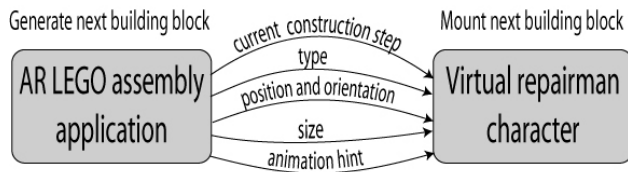


Figure 8. Application attributes controlling agent behavior

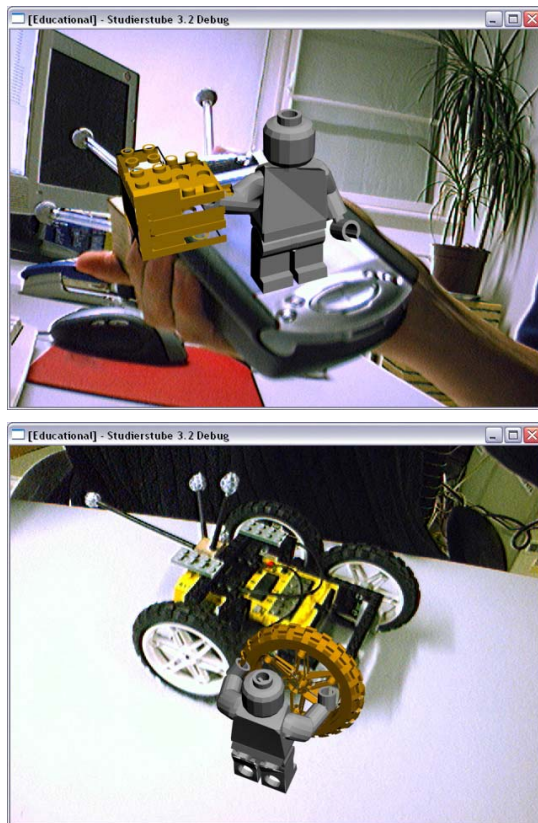


Figure 9. Virtual repairman as an assembly assistant: a) Introducing the next building block b) Mounting the next block on the actual physical robot

If the size parameter of the block is below a certain threshold, the object is linked to and carried in one hand, otherwise it is held in both hands because the object has been perceived as “heavy”. The position of the next block allows the movement from the agent’s current location to the tile’s target location to be planned without bumping into the already constructed model. The block’s suggested orientation instructs the agent to put the block into the correct pose before mounting.

Finally, the virtual block is added to the real robot using either a default gesture or special gesture sequences (e.g. turn around, push in, twist) if an animation hint is provided. There are steps that do not include any LEGO

blocks, e.g. instructing the user to turn the construction around if the robot’s relative orientation hides important details for the next step.

5.2. LEGO robot

Engines and sensors (push, light, rotation, temperature etc.) are important components of the robot since misconnection or misconfiguration would lead to erroneous behavior. It is desirable that the user can quickly verify whether they are connected and functioning correctly. By turning our physical robot into an AR agent, these tests become intuitive. The robot agent is able to accept commands from the assembly application and synchronize them with the repairman. This means that as soon as the user finishes mounting an engine, the robot attempts to turn the engine on. In case of incorrect behavior, the current or previous construction steps have to be repeated. Sensors are equally important entities. As soon as they are mounted, the user is invited to test them, for example by pushing, holding a color plate, etc. while watching the results of the sensor query commands. Structure and appearance cannot be verified since most LEGO tiles are passive. However, the current model can be shown to the expert using the augmented videoconference window, who can remotely give advice and hints using natural gestures and voice instructions.

LEGO-like tangible interfaces for real-time interaction have already inspired researchers [24]. Using a LEGO Mindstorms robot appears to be an appropriate choice to build and test machine prototypes for the following reasons:

- It is easy to build and export a precise virtual counterpart of new robot models using a free LEGO CAD editor [25], therefore there is a smooth modeling pipeline.
- A “central brain” unit, called RCX, is able communicate with a PC running the AR Puppet framework with infrared commands using a communication tower, which makes controlling and monitoring straightforward. The RCX can set and query engine parameters like on/off state, voltage and direction, configure and query various sensors, and send status information like acknowledgement of a finished command, battery power and other useful data.
- LEGO is fun to work with and is familiar to many people.

To correctly overlay virtual information, we need to track the current construction’s pose in the physical environment. Mounting the tracking marker structure on the RCX is an obvious choice since this is the basic building block present in all interactive robots. It contains

important control buttons and an LCD display, which should not be covered by tiles, so there is always free space for the mounting of markers, which are integrated into LEGO bricks.

5.3. Interaction

Both the local and the remote user are able to interact with the assembly application and the LEGO robot in real-time using a tracked PDA, which acts as a multi-purpose interaction device (see Figure 10). Firstly, it serves as a tangible, physical interface to move the virtual repairman and virtual LEGO tiles around in the physical environment. However, it also renders a PUC-based GUI on the PDA screen, which can be used to move between the assembly steps and control the LEGO robot's engines and sensors. The brightness of the PocketPC's display also allows easy viewing by HMD users. The PDA has a wireless network card for cable-free interaction and a tracking marker structure mounted on the back of the case.

We exploit the fact that we are aware of the current pose of the PocketPC to create the following gestures in the application:

- 1.) The virtual repairman holds the next block in the construction while standing on the local user's PDA. The user can freely observe it from all angles by moving the device around. As soon as she places the PDA near the real robot, the character moves from the PDA's local coordinate system into the physical LEGO robot's local coordinate system and starts explaining where to place the LEGO tile and how.

- 2.) After the explanation is finished, the user can pick up the character again with the PDA, which binds it to the PDA again and commands it to display the next building block.

- 3.) The local user places her PDA with the repairman near the tracked monitor displaying the videoconferenc-

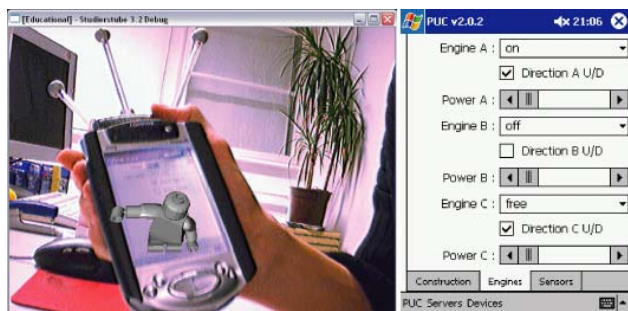


Figure 10. Tracked PocketPC as a multi-purpose interaction device: (left) Tangible interface in a screenshot of the AR LEGO application (right) PDA screen capture of the LEGO robot's control GUI

ing window. This serves as an indication to the system that she wants to "send" the current block over to the remote user for further observation. The virtual model of the next block is copied and immediately appears on top of the remote operator's PDA, and can be freely moved around on the operator's display.

5.4. Work environment and interaction

The local user sits at a table with a box of spare LEGO bricks. Near the box is the monitor showing the remote operator's videoconference window. The local user wears a tracked helmet equipped with an HMD and a headset for voice communication, and holds her interaction device in her hand. The room is equipped with a high-quality optical tracking system, which tracks the user's head, her PocketPC interaction device, the RCX of the LEGO robot and the camera for videoconferencing.

The remote user has a desktop-based AR setup consisting of an ordinary monitor and a webcam to optically track an ARToolKit marker mounted on the PDA. The monitor displays the local and the remote user's videoconference window. Figure 11 shows a snapshot of the local user's work environment.

6. Conclusion and future work

We have created a hierarchical animation framework for Augmented Reality agents called AR Puppet, which combines several research areas into a single coherent human-computer interface paradigm. We have made the following contributions:

- 1.) AR Puppet is the **first general framework** for autonomous animated agents that has been developed specifically for AR applications. The framework has been built on a powerful middleware, which allows experimentation with a wide range of applications, tracking technology, platforms and displays.



Figure 11. Work environment of local user

2.) The framework has examined **agent-specific aspects of AR**. New modalities enabled by the physical environment have been exploited in animated agent behavior and autonomous behavior has been added to an AR application at low cost by **marking relevant attributes** of an application scene graph. Agents can easily monitor and thus react to user interaction and changes in the application state.

3.) **Physical objects** can be turned to **intelligent, responsive agents** and used as input and output devices in AR environments. Consequently, the agents of AR Puppet can not only talk back but “hit back” as well. A unified command interface allows physical and virtual objects to be scripted in the same way.

4.) AR Puppets are **dynamically configurable**, their attributes and command fields can be intuitively controlled **through mobile devices**.

We believe that animated agents bring new challenges and fresh perspectives for AR. In the near future we plan detailed evaluation of AR agents examining the following aspects:

- **Psychology:** effectiveness of various gestures, amount of autonomy allowed, amount of anthropomorphic features in appearance and behavior
- **Usability:** display and interaction types, user acceptance
- **Interface:** size of the character, placement of the character on the right spot: informative, not obtrusive

Another important aspect is the implementation of agent memory. This means that relevant samples of application states and agent decisions can be stored in a database, and later recalled by a query. Thereby AR agents become persistent and remember their obtained knowledge and the users’ latest application profiles.

7. Acknowledgement

This system was sponsored by the Austrian Science Fund FWF (contract no. Y193) and the European Union (contract no. IST-2001-34204). Special thanks go to Joseph Newman for his zero tolerance paper review.

8. References

[1] M. Addlesee, R. Curwen, S. Hodges, J. Newman, P. Steggle, A. Ward, A. Hopper, “Implementing a Sentient Computing System”, *IEEE Computer* 34 (8), 2001, pp. 50-56.
 [2] T. Noma, L. Zhao, N. Badler, “Design of a Virtual Human Presenter”, *IEEE Comp. Graphics & Applications*, 20(4), 2000.
 [3] K. Perlin, and A. Goldberg, “Improv. A System for Scripting Interactive Actors in Virtual Worlds”, *Proc. of SIGGRAPH '96*, 1996, pp. 205-216.

[4] J. Rickel, W. Johnson, “Steve: A pedagogical agent for Virtual Reality”, *Proc. of the 2nd International Conference on Autonomous Agents*, 1998, pp. 332-333.
 [5] P. Maes, T. Darrell, B. Blumberg, A. Pentland, “The ALIVE System: Wireless, Full-body Interaction with Autonomous Agents”, in *ACM Multimedia Systems*, 5(2), 1997, pp.105-112
 [6] M. Anabuki, H. Kakuta, H. Yamamoto, H. Tamura, “Welbo: An Embodied Conversational Agent Living in Mixed Reality Space”, *CHI 2000, Extended Abstracts*, 2000, pp. 10-11.
 [7] L. Vacchetti, V. Lepetit, G. Papagiannakis, M. Ponder, P. Fu, “Stable real-time interaction between virtual humans and real scenes”, *3DIM 2003*, Banff, AL, Canada, 2003.
 [8] B. MacIntyre, J. D. Bolter, J. Vaughan, B. Hannigan, E. Moreno, M. Haas, M. Gandy, “Three Angry Men: Dramatizing Point-of-View using Augmented Reality”, *SIGGRAPH 2002 Technical Sketches*, San Antonio, TX, 2002.
 [9] A. D. Cheok, W. Weihua, X. Yang, S. Prince, F. S. Wan, M. Billingham, H. Kato, “Interactive Theatre Experience in Embodied and Wearable Mixed Reality Space”, *Proc. of International Symposium on Mixed and Augmented Reality (ISMAR'02)*, Darmstadt, Germany, 2002.
 [10] M. Cavazza, O. Martin, F. Charles, S.J. Mead, X. Marichal, “Interacting with Virtual Agents in Mixed Reality Interactive Storytelling”, *Proc. of Intelligent Virtual Agents*, Kloster Irsee, Germany, 2003.
 [11] S. Balcişoy, M. Kallmann, R. Torre, P. Fua, D. Thalmann, “Interaction Techniques with Virtual Humans in Mixed Environments”, *Proc. of Intern'l Symposium on Mixed Reality*, Tokyo, Japan, 2001.
 [12] W. Piekarski, B. Thomas, “ARQuake: The Outdoor Augmented Reality Gaming System”, *ACM Communications* 45(1) 2002, pp. 36-38.
 [13] K. Mase, Y. Sumi, R. Kadobayashi, “The Weaved Reality: What Context-aware Interface Agents Bring About”, *Asian Conference on Computer Vision*, Taipei, 2000.
 [14] M. Gutierrez, F. Vexo, D. Thalmann, “Controlling Virtual Humans Using PDAs”, *Proc. of the 9th International Conference on Multimedia Modelling (MMM'03)*, Taiwan, 2003.
 [15] M. Kruppa, A. Krüger, “Concepts for a combined use of Personal Digital Assistants and large remote displays”, *Proc. of Simulation and Visualization 2003*, Magdeburg, Germany, 2003, pp. 349-361.
 [16] B. MacIntyre, M. Gandy, “Prototyping Applications with DART, The Designer’s Augmented Reality Toolkit”, *Software Technology for Augmented Reality Systems Workshop (STARS 2003)*, Tokyo, Japan, 2003.
 [17] M. Florence, R. Storey, “Vietnam”, *Lonely Planet Publications*, 2001, pp. 216-217.
 [18] U. Spierling, D. Grasbon, N. Braun, I. Iurgel, “Setting the scene: playing digital director in interactive storytelling and creation”, *Computers & Graphics* 26(1), 2002, pp. 31-44.
 [19] Cal3D project website, <http://cal3d.sourceforge.net>
 [20] J. Nichols, B. A. Myers, M. Higgins, J. Hughes, T. K. Harris, R. Rosenfeld, M. Pignol, “Generating Remote Control Interfaces for Complex Appliances”, *CHI Letters: ACM Symposium on User Interface Software and Technology*, Paris, France, 2002, pp. 161-170.
 [21] D. Schmalstieg, A. Fuhrmann, G. Hesina, Zs. Szalavári, M. Encarnação, M. Gervautz, W. Purgathofer, “The Studierstube Augmented Reality Project”, *PRESENCE -Teleoperators and Virtual Environments*, MIT Press, 2002.
 [22] I. Barakonyi, T. Fahmy, D. Schmalstieg, “Remote Collaboration Using Augmented Reality Videoconferencing”, *Proc. of Graphics Interface 2004*, London, ON, Canada, 2004, pp. 89-96.
 [23] T. Pfeiffer, M. E. Latoschik, “Resolving object references in multimodal dialogues for immersive virtual environments”, *Proc. of VR 2004*, Chicago, IL, USA, 2004, pp. 35-42.
 [24] E. Sharlin, Y. Itoh, B. Watson, Y. Kitamura, L. Liu, S. Sutphen, “Cognitive Cubes: A Tangible User Interface for Cognitive Assessment”, *Proc. of CHI 2002*, Minneapolis, Minnesota, USA, 2002, pp. 347-354.
 [25] MLCAD website, <http://www.lm-software.com/mlcad/>