A. Egges, G. Papagiannakis, N. Magnenat-Thalmann. An Interactive Mixed Reality Framework for Virtual Humans. Cyberworlds 2006, EPFL, Switzerland, November 28-29, IEEE Computer Society. November 2006.

An Interactive Mixed Reality Framework for Virtual Humans

Arjan Egges, George Papagiannakis, Nadia Magnenat-Thalmann
MIRALab - University of Geneva
7 route de Drize, 1227 Geneva, Switzerland
{egges,papagian,thalmann}@miralab.unige.ch

Abstract

In this paper, we present a simple and robust Mixed Reality (MR) framework that allows for real-time interaction with Virtual Humans in real and virtual environments under consistent illumination. We will look at three crucial parts of this system: interaction, animation and global illumination of virtual humans for an integrated and enhanced presence. The interaction system comprises of a dialogue module, which is interfaced with a speech recognition and synthesis system. Next to speech output, the dialogue system generates face and body motions, which are in turn managed by the virtual human animation layer. Our fast animation engine can handle various types of motions, such as normal key-frame animations, or motions that are generated on-the-fly by adapting previously recorded clips. All these different motions are generated and blended on-line, resulting in a flexible and realistic animation. Our robust rendering method operates in accordance with the previous animation layer, based on an extended for virtual humans Precomputed Radiance Transfer (PRT) illumination model, resulting in a realistic display of such interactive virtual characters in mixed reality environments. Finally, we present a scenario that illustrates the interplay and application of our methods, glued under a unique framework for presence and interaction in MR.

1 Introduction

Over the last few years, many different systems have been developed to simulate scenarios with interactive virtual humans in virtual environments in real-time. The control of these virtual humans in such a system is a widely researched area, where many different types of problems are addressed, related to animation, speech, deformation, and interaction, to name a few research topics. The scope of applications for such systems is vast, ranging from virtual training or cultural heritage to virtual rehabilitation. Although there is a variety of systems available with many different features, we are still a long way from a completely integrated system that is adaptable for many types of applications. This is not only because of the amount of effort that is required to integrate different pieces of software, but also because of the real-time constraint. The latter becomes especially an issue when many different components need to work together.

The true challenge of such systems is for a person to be able to naturally *interact* with the virtual humans in the virtual as well as in the real scene, for example by using a speech recognition interface. A lot of research has been done to develop chatbots that are mostly focused on a simple stand-alone application with a cartoon face [1], however only a few systems succeed to link interaction with controlling 3D face and/or body motions played in synchrony with text-to-speech [4, 10, 12]. The main problem with such systems is that they are far from ready to be used in Mixed Reality applications, which are much more demanding than a stand-alone application. For example, a highly flexible animation engine is required that not only plays animations in combination with interaction, but that is also capable of playing simple key-frame animations as part of a predefined scenario. This also means that a dynamic mechanism is required that allows to switch between playing key-frame animations as part of a scenario, and animations related to the interaction (such as body gestures and facial expressions) without interrupting the animation cycle. Furthermore, Mixed Realities require a more elaborate rendering method for the virtual humans in the scene, that uses the global illumination information, so that virtual humans that augment a reality have lighting conditions that are consistent with the real environment. This is another challenge that this work takes upon, since consistent rendering in MR contributes to both feeling of presence as well as realism of interaction in the case of virtual characters in MR scenes. To the best knowledge of the authors—and as it is highlighted in Section 2—there are no such systems appearing in the bibliography up to date.

In this paper, we propose simple and fast methods for three important components (Interaction, Animation and

Rendering), that elevate some of the issues discussed above. Our approaches are specifically tailored to work as components of a mixed reality real-time application. Our fully integrated system, includes speech recognition, speech synthesis, interaction, emotion and personality simulation, real-time face and body animation and synthesis, real-time camera tracking for AR and real-time virtual human PRT rendering, and is able to run at acceptable speeds for real-time (20-30fps) on a normal PC. As a part of the work presented in this paper, we will show the system running different scenarios and interactions in MR applications.

2 Background

A growing number of projects are currently based on AR integrated platforms, exploring a variety of applications in different domains such as cultural heritage [31, 19, 20], training and maintenance [30] and edutainment-games [28, 27]. Special focus has recently been applied to system design and architecture in order to provide the various AR enabling technologies a framework for proper collaboration and interplay. Azuma [2] describes an extensive bibliography on current state-of-the-art AR systems and frameworks. However, few of these systems take the modern approach that a realistic mixed reality application, rich in AR virtual character experiences, should be based on a complete VR Framework (featuring game-engine like components) with the addition of the "AR enabling Technologies" like a) Real-time Camera Tracking b) AR Displays and interfaces c) Registration and Calibration. Virtual characters were also used in the MR-Project [27] where a complete VR/AR framework for Mixed Reality applications had been created. Apart from the custom tracking/rendering modules a specialized video and see-through HMD has been devised.

We believe that as PRT methods can produce the most realistic and physically principled real-time GI effects up to date, they are ideal for VHs as they allow a wealth of GI effects to be simulated: area lights, self-shadowing, interreflections, BSSRDF with subsurface scattering, all important elements for realistic depiction of VHs. The last GI effects (interreflections, BSSRDF) are not covered in this thesis but are fully supported within the PRT context [26, 11]; our assumptions thus cover and extend the basic PRT method for diffuse, low-frequency convex and concave VH surfaces lit by area lights.

Mixing such aesthetic ambiences with virtual character augmentations [5] and adding dramatic tension has resulted in an exciting new edutainment medium. Balcisoy [3] also presented a novel system (one of the first in the bibliography) to present interactive virtual humans in AR. In these systems, the animation of the humanoid is generally based on scripted animations. However, a truly *interactive* virtual human requires a high level of control over the body postures and gestures. There are several research efforts that try to control such types of character movements. The Greta system is an Embodied Conversational Agent (ECA) simulator that includes a dialogue system, emotions and a facial expression synthesizer [22]. Hartmann et al.[10] present an extension of the Greta system that automatically generates hand and arm gestures from conversation transcripts using predefined key-frames. Another well-known system that can produce gesture animations from text, is the MAX system, developed by Kopp and Wachsmuth[12]. Most of these systems use procedural methods to generate the gesture motions, resulting in rather stiff motions. In order to overcome this problem, recent work starts to integrate motion capture based techniques to generate new motions while still retaining the flexibility of procedural motion synthesizers [8]. This approach synthesizes full body motions as a blended combination of *idle motions* and *gesture motions*. The idle motions provide for a continuous motion for any character of any required length.

In combination with these idle motions, gesture motions are blended in. We have developed a method for automatic dependent joint motion synthesis in real-time [7]. Following this approach, existing gesture synthesizers can be used to produce the basic gesture tracks for a few joints, and our method automatically adds the motions of dependent joints, resulting in a more natural looking motion.

Currently, no AR system exists that can handle such kinds of complex full body motions, interaction, speech recognition/synthesis, illumination registration, geometrical registration, skin deformation, facial animation, and more, in a mobile setup. In this paper, we will present such a system, building on a previously developed framework called VHD++ [23].

In Section 3, we will give an overview of that system. Then, we will describe how this system was extended with animation and interaction capabilities in Section 4. Section 5 will show our global illumination model that is used to light the scene. Finally, in Section 6 we will show an example AR scenario that is succesfully handled by our system.

3 VHD++

Our MR-Rendering, animation and interaction system is incorporated in the VHD++ [23] component-based framework engine developed by VRLab and MIRALab. This framework allows quick prototyping of VR-AR applications featuring integrated real-time virtual character simulation technologies, as depicted in Figure 1 and the OpenSceneGraph [18] real-time scenegraph rendering API. The key innovation of VHD++ is focused in the area of component-based framework that allows the plug-and-play of different heterogeneous human simulation technologies

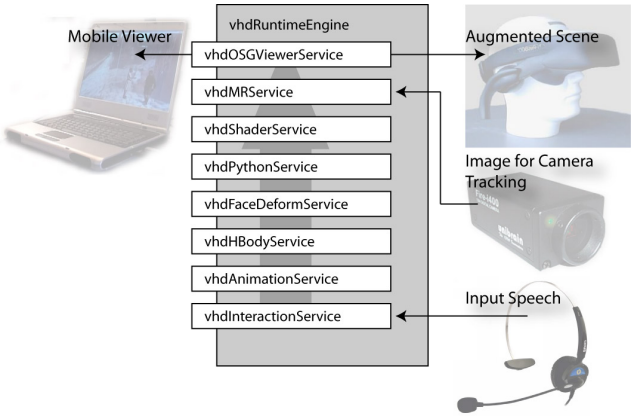

Figure 1. Overview of the VHD++ MR framework.

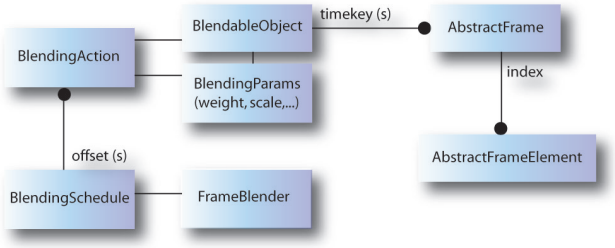

Figure 2. Overview of the blending engine data structure.

such as: Real-time character rendering in AR (supporting real-virtual occlusions), real-time camera tracking, facial simulation and speech, body animation with skinning, 3D sound, cloth simulation and behavioral scripting of actions.

The software architecture is composed of multiple software components called services, as their responsibilities are clearly defined. They have to take care of rendering of 3D simulation scenes and sound, processing inputs from the external VR devices, animation of the 3D models and in particular complex animation of virtual human models including skeleton animation and respective skin and cloth deformation. They are also responsible for maintenance of the consistent simulation and interactive scenario state that can be modified with python scripts at run-time.

For animation and interaction, we have developed two services. The *vhdAnimationService* contains all the animation related components: motion synthesis, blending, loading and saving animations and so on. The *vhdInteractionService* uses the animation service to control a virtual character: it includes speech recognition and synthesis, a dialogue manager and emotion/personality simulation. For the global illumination and rendering, an *vhdOSGShaderService* has been developed, that implements the PRT shading approach. In the following sections, we will give an overview of each of these services.

4 Real-time Animation and Interaction

4.1 Animation

In this section, we will present our animation engine, called MIRAnim. The main architecture of the animation engine is a multi-track approach, where several animation streams need to be blended into a final animation. There has been quite some research in motion blending. Perlin [21] was one of the first to describe a full animation system with blending capabilities based on procedurally generated motions. There are several researchers who have used weight-based general blending to create new animations [32, 24]. There have also been several efforts to apply motion blending not directly on the joint orientation domain. For example, Unuma et al. [29] perform the motion blending in the Fourier domain and Rose et al. [25] used spacetime optimization to create transitions that minimize joint torque. Kovar et al. [13] use registration curves to perform blending. Their approach automatically determines relationships involving the timing, local coordinate frame, and constraints of the input motions. Blend-based transitions have been incorporated into various systems for graph-based motion synthesis [24, 14].

The goal of our animation engine is to provide for a generic structure that allows for the implementation of different blending strategies. This is especially important, since our animations use different representations, depending on the application. Additionally, in the final system, we will need to perform blending operations on both body and face animations, which are two completely different animation formats that require different blending strategies. The approach that we will present in this Section is suitable for any of the previously discussed blending approaches. A large set of blending tools, for example time warping, splitting, fading, and so on are available. The advantage of using this generic approach is that once a blending tool has been defined, it can be used for any type of animation, regardless of its structure. In order to be able to use these blending tools, only an interface needs to be provided between the data structure used for blending, and the original animation structure. An overview of the blending engine is provided in Figure 2.

The basic structure used in the blending engine is the so-called `BlendableObject` interface. A blendable object is the representation of an animation that can be blended with other animations. A blendable object is defined as a function $A : t \rightarrow F$, where t is a timekey $\in [t_s, t_e]$ with

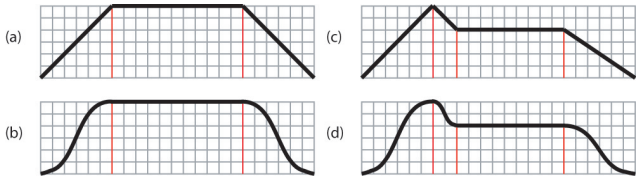

Figure 3. Various basic curve structures are available, such as (a) linear or (b) cubic fading, and (c) linear or (d) cubic attack-decay-sustain-release.

$0 \leq t_s < t_e < \infty$, and F is the corresponding keyframe of the animation. A frame in the blending engine is called an `AbstractFrame`. An abstract frame consists of a number of elements, called `AbstractFrameElement` objects. For example, in the case of body animations, an `AbstractFrameElement` could be a list of 4 floating points, representing a quaternion rotation. In the case of facial animation, the abstract frame element could be a list of 1 floating point, representing a FAP value in the MPEG-4 standard[16].

For each animation track, blending parameters are defined using the `BlendingParams` interface. The most basic parameter used for blending is a weight value to be used for blending. In addition to a list of weights, the `BlendingParams` interface also provides for a list of scalings. The evolution of the weights and scalings over time is governed by a parametrizable curve. Figure 3 shows some examples of curves that can be chosen. Different types of `BlendingParams` objects can be multiplexed, and custom blending parameter objects can be defined.

The `BlendingParams` object, together with the `BlendableObject`, form a `BlendingAction` object. The final step in obtaining a mix of different `BlendingAction` objects requires a structure that allows for activating and deactivating different animations according to the blending parameters. This structure is called a `BlendingSchedule`. Each blending action is associated with a timekey, which defines the time that the blending action should start. The `BlendingSchedule` is again a blendable object. This allows for performing animation blending on different levels, with local blending parameters.

The animation service is built around the blending engine, and it contains specific implementations for controlling both face and body animation. The service can blend several different types of motions, including real-time idle motions and key-frame animations. The service also includes an integrated player, that plays and blends scheduled animations in a separate thread for all the humans in the scene.

The animation service is controlled either through a GUI, or through a Python script. The use of Python scripts allows for a complete control over many different parts of the environment, such as audio playing in synchrony with animations or camera motions. The following example shows a simple script that controls the body motions of two characters and activates several audio signals as well (crowd cheering, and prerecorded speech signals). The script activates prerecorded actions (such as 'creon_wrk') and actions linked with the motion synthesizer (such as 'creon_idle'). These different actions are blended on the fly and played on the characters.

```
# global variables
Antigone="Antigone_vhd_occ_rec"
Creon="Creon_vhd_occ_rec"
cam01Creon_crowd="root.Cam01Creon_crowd"

# start the sound
sndService.sndmpPlayMedia(cam01Creon_crowd)

# no facial animation
animService.initPlayerScope_face(False)

# start the player
animService.start_player()
animService.activateAction_body
  (Antigone,"antigone_idle")

# creon monologue
animService.activateAction_body(Creon,
  "creon_wrk")
voiceService.activateAction(Creon,
  "CreonSpeech.Cam01CreonP1",1.0)
animService
  .waitUntilActionFinished_body(Creon,
  "creon_wrk",-4.0)
animService.activateAction_body(Creon,
  "creon_idle")

# Antigone answers to Creon
animService.cutOffAction_body(Antigone,
  "antigone_idle",3.0)
animService.activateAction_body(Antigone,
  "antigone_wrk")
voiceService.activateAction(Antigone,
  "AntigoneSpeech.Cam01CreonP3",1.0)
animService
  .waitUntilActionFinished_body(Antigone,
  "antigone_wrk",-4.0)
animService.activateAction_body(Antigone,
  "antigone_idle")

sndService.sndmpStopMedia(cam01Creon_crowd)
```

4.2 Interaction

As a higher level control mechanism on top of the animation service, we have developed the *interaction service*. This service mainly handles the spoken interaction between a user and any virtual human in the scene. The core of the interaction service is a dialogue manager [6], that responds to the user according to a predefined script that follows a similar approach as Alice [1]. In order to provide for a more natural interaction, we have integrated the Microsoft Speech SDK (SAPI5.1) [17] into our MR framework, so that Automatic Speech Recognition (ASR) as well as Text-to-Speech (TTS) is available.

From the response text generated by the dialogue manager, the facial speech animation is created automatically [6] using the timing information obtained from the TTS system. Additionally, the responses generated by the dialogue manager contain XML tags, which define specific face and body gestures that should be displayed in synchrony with the speech animation. An example of such a tagged response is given as follows:

```
<begin_gesture id="g1" anim="shake_head"/>
Unfortunately, I have <begin_gesture id="g2"
anim="raise_eyebrows"/>no idea <end_gesture
id="g2"/> what you are talking about.
<end_gesture id="g1"/>
```

Within each gesture tag, an animation ID is provided. When the gesture animation is created, these animations are loaded from a database of gestures, also called a Gesticon[15]. Next to the animation itself, the Gesticon also contains information related to the scope of each animation (which part of the body it should play on) and the weights of the joints. Each animation is stretched so that it is in synchrony with the speech, and it is blended with the other running animations by the animation service.

Because the same service is used for both scripted and interactive animation, it is possible to first play a script as shown in the previous section, and then dynamically switch to animation controlled by the interaction service. Since the animation playing itself is continuously handled by the animation service, the character animation is not disrupted when this switch is made.

5 Real-time Shading

In order to enhance the consistency of illumination of the real scene and the virtual characters, we have chosen to extend the Precomputed Radiance Transfer(PRT) illumination model, in order that is applicable to the multi-segmented, deformable and animatable hierarchies that the MIRAnim system is controlling. The main issue that we resolved was to allow for the multiple segments that the virtual skeleton consists of (due to different joints, segments, cloth, hair meshes) to respond to a high dynamic range area light that can be captured from the real scene, similar to the method described in [19]. Thus starting from the Rendering Equation approximation in diffuse PRT Methods [26, 11]:

$$L_0(x) = \frac{K_d}{\pi} \int_U L_s(l) T^{DSM} \, dU$$

where

$$T^{DSM} = V(x,l) \cdot \max(n \cdot l, 0)$$

we define a new visibility function $V(x,l)$ which checks visibility of the current vertex position against not only the current geometric mesh that resides but against a Proxy mesh M':

$$M' = \sum_{i=0}^{n} S_i$$

Where S_i are all the Mesh Segments of the skeleton hierarchy of the virtual character. However, not all segments are desired to be able to cast or receive self-shadowing and other GI effects. Thus the construction of the proxy mesh M' is based according to the 'type' of the mesh segment, tagged as *Receiver* or *Occluder* during the artist mesh preparation loop:

Receiver can receive but cannot cast self shadowing to other objects of the Proxy Mesh

Occluder can cast but cannot receive self shadowing from other objects of the Proxy Mesh

This is a disjoint set, i.e. an element cannot participate in both groups. If a 'Receiver' segment is found, it is excluded from the Proxy Mesh, thus being transparent to ray casting hits. If an 'Occluder' segment is found, then this segment is excluded from further dPRT processing and calculation proceeds to next segment. However, this segment is part of the Proxy Mesh and candidate for ray-casting hits by other segments. Figure 4 illustrates the VH result of DSM with and without the Receiver-Occluder specifications. The main artifacts appear between the Skull and Hair segments which is a typical case of VH arrangement, as separate meshes due to separate creation processes and materials involved. However, with the application of our Receiver-Occluder set of elements, this artifact is avoided, as shown in Figure 4.

We employ a Voxel Ray Tracer and Uniform Grid acceleration structure for offline ray-tracing for PRT self-shadowing tests. This uniform grid is being initialized in a first pass with this Proxy Mesh while tested in a second pass, similarly as before with rays originating from every vertex of every segment in the skeleton hierarchy. The main difference is that in the simple Diffuse Shadowed Transfer from Sloan et al. [26], the ray tracer was initialized with containing only the parent mesh of the current vertex x for which rays arriving from directions l where tested.

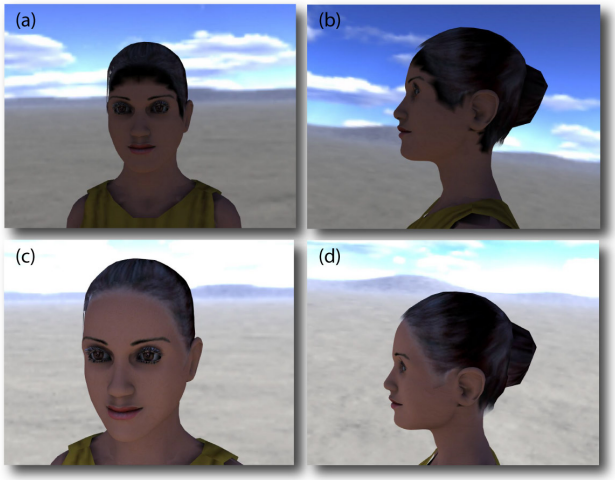

Figure 4. Image without (a, b) and with (c, d) Receiver - Occluder set processing for dPRT.

Figure 5. Tensor Distance Ray-Hit Relaxation Criteria for DSM Transfer and dPRT. Images shown with (a) and without (b) tensor distance threshold applied.

This simple and efficient scheme of our Proxy mesh, results in our Diffuse Shadowed Merged Transfer, that allowed us for correct global illumiantion effects such as self-shadowing and successful application of diffuse PRT methods for multi-mesh virtual humans.

As shown in Figure 5, in the case of multi-segmented, multi-material skeleton hierarchies, even with our Diffuse Shadowed Merged Transfer, incorrect self-shadowing is performed due to the proximity of the different mesh segments. It is usually manifested as wrong dark coloring in the vertices in the boundary edges of different skeleton segments, as these segments are modeled adjacent in the skin bind pose, but as they are separate segments, still maintain a distance between them, not visible to the eye but large enough for ray casting. Thus one relaxation criteria that we have been applying with positive results is the checking for the tensor distance between the ray origin and the ray-triangle intersection point, against a value e corresponding to the minimum distance between the different mesh segments. E.g. for the H-Anim [9] structure this corresponds to $1.71 * 10^{-4}$ m. Hence all false hit-point candidates returned by the ray tracer below this e value correspond to hits between the segment boundaries and thus ignored.

Another issue for the shadow feeler visibility rays is that polygons adjacent to the current vertex ray origin are going to be coplanar to many rays originating from that vertex. Thus any ray tested against a polygon that included the ray origin will at best return a hit at that vertex origin. This also leads to incorrect shadowing so it is important that we exclude the polygons that contain our ray origin, from the M' Proxy Mesh. Thus in the construction of M' we create full Face, Edge, Vertex information adjacency in order to account for this special case. This second criteria has been already known to the ray-tracing community and similar approaches have been discussed.

Since the final shading information is stored per vertex in the H-Anim virtual human hierarchy, it is independent of the underlying animation approach. Therefore the MIRAnim system is easily integrated within our MR framework to animate the skeleton of the virtual humans with shading according to the blending schedule that is defined (see Section 4.1). In Figure 6 shows the difference between the various shading strategies. At the far left a standard Phong-based model as opposed to our PRT method with Diffuse Shadowed Merged Transfer.

6 Results

For our AR case study we have extended the system described in [19] by allowing for interaction with a virtual human in MR. This aims to allow visitors of ancient Pompeii to be equipped with a mobile AR guide and experience real-time digital narratives of ancient virtual life coming to life within their natural 'real' Pompeian environment. The two crucial components for realizing these AR experiences are a) the feature based camera tracker and b) the MR framework for virtual life simulation and AR registration. The work presented in this paper provides three new components in the previous framework: a) procedural animation, b) interaction and c) MR rendering based on PRT methods. As our original storytelling experience has been intended to revive the life in ancient Pompeii, Italy in an AR simulation manner, a real-size paper maquette of the walls of the Thermopolion of Vetutius Placidus has been recreated in the lab as depicted in Figure 7. The employed hardware platform was based on a P4 3.0 GHz mobile workstation, with a NVIDIA Geforce5600Go graphics card and Unibrain firewire web camera attached on an I-Glasses HMD. The resulting performance was approximately 30fps for 12.000

Figure 6. Standard Phong-based shadowing (a) in comparison with our PRT method with Diffuse Shadowed Merged Transfer (b,c).

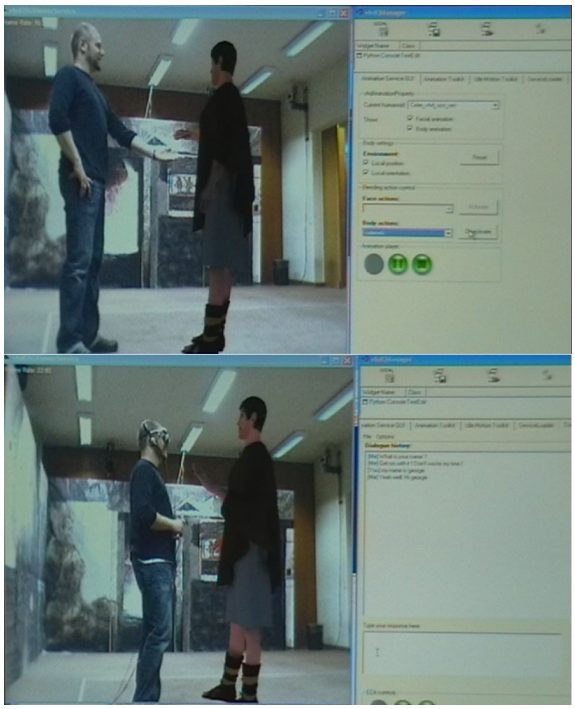

Figure 7. Real human and Interactive Virtual Human in a Mixed Reality.

polygons, depending on the performance of the markerless feature-based camera tracker.

7 Conclusion

In this paper, we have presented a Mixed Reality framework for Interactive Virtual Humans. We have presented a flexible interaction and animation engine, in combination with a robust real-time rendering engine that uses a global illumination for real-time PRT extension for virtual humans. The animation engine allows to switch dynamically between interaction and scenario playing, without interrupting the animation cycle. All components have been integrated as plug-ins in a modern framework, and both VR and AR applications exhibit very suitable performance for real-time MR applications. Finally we illustrate that the amalgam of animation, rendering and interaction with virtual characters, particularly in real scenes, could pave the way for a new suite of applications in augmented reality and a wider adoption of this technology that still lacks a clear application domain.

As with any MR application, many features can still be added to increase the realism and the presence of the users in the scene and that will be the focus of our continuing work in this area. First, our virtual human animation system currently does not consider interaction with objects in the MR scene. This would include picking up virtual objects, or perhaps an integration with a haptic device in order to interact with objects from the real world. Also, a look-at behaviour would be an interesting extension, which would drastically improve the sense that the virtual human is aware of the user in the scene. Finally, we are currently investigating the inclusion of more elaborate real-time simulation and rendering of real-time virtual hair and clothing in the MR scene, while still ensuring that the system can operate at an acceptable framerate on a commodity desktop PC.

8 Acknowledgements

This research has been supported through the EU funded projects EPOCH (IST-507382) and HUMAINE (IST-507422). We would also like to thank Nedjma Cadi and Alessandro Foni for designing the environment and the characters.

References

[1] Alice chat bot. http://www.alicebot.org/. Accessed November 2005.

[2] R. Azuma, Y. Baillot, R. Behringer, S. Feiner, S. Julier, and B. MacIntyre. Recent advances in augmented reality. *IEEE Computer Graphics and Applications*, 21(6):34–47, November/December 2001.

[3] S. S. Balcisoy. *Analysis and development of interaction techniques between real and synthetic worlds*. PhD thesis, EPFL, 2001.

[4] J. Cassell, H. Vilhjálmsson, and T. Bickmore. BEAT: the Behavior Expression Animation Toolkit. In *Proceedings of SIGGRAPH*, pages 477–486, 2001.

[5] M. Cavazza, O. Martin, F. Charles, S. Mead, and X. Marichal. Users acting in mixed reality interactive storytelling. In *2nd International Conference on Virtual Storytelling*, 2003.

[6] A. Egges, S. Kshirsagar, and N. Magnenat-Thalmann. Generic personality and emotion simulation for conversational agents. *Computer Animation and Virtual Worlds*, 15(1):1–13, 2004.

[7] A. Egges and N. Magnenat-Thalmann. Emotional communicative body animation for multiple characters. In *First International Workshop on Crowd Simulation (V-Crowds)*, pages 31–40, 2005.

[8] A. Egges, T. Molet, and N. Magnenat-Thalmann. Personalised real-time idle motion synthesis. In *Pacific Graphics 2004*, pages 121–130, 2004.

[9] H-ANIM Humanoid Animation Working Group. Specification for a standard humanoid. http://www.h-anim.org/. Accessed May 2006.

[10] B. Hartmann, M. Mancini, and C. Pelachaud. Formational parameters and adaptive prototype instantiation for mpeg-4 compliant gesture synthesis. In *Computer Animation 2002*, pages 111–119, 2002.

[11] J. Kautz, J. Lehtinen, and P.-P. Sloan. Precomputed radiance transfer: Theory and practise. In *ACM SIGGRAPH 2005 Course Notes*, 2005.

[12] S. Kopp and I. Wachsmuth. Synthesizing multimodal utterances for conversational agents. *Computer Animation and Virtual Worlds*, 15(1):39–52, 2004.

[13] L. Kovar and M. Gleicher. Flexible automatic motion blending with registration curves. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 214–224, 2003.

[14] L. Kovar, M. Gleicher, and F. Pighin. Motion graphs. In *Proc. SIGGRAPH 2002*, 2002.

[15] B. Krenn and H. Pirker. Defining the gesticon: Language and gesture coordination for interacting embodied agents. In *Proceedings of the AISB-2004 Symposium on Language, Speech and Gesture for Expressive Characters*, pages 107–115, University of Leeds, UK, 2004.

[16] S. Kshirsagar, S. Garchery, and N. Magnenat-Thalmann. Feature point based mesh deformation applied to MPEG-4 facial animation. In *Proceedings Deform2000*, pages 23–34, Geneva, Switzerland, November 2000.

[17] Microsoft Speech SDK version 5.1 (SAPI5.1). http://www.microsoft.com/speech/download/sdk51/. Accessed May 2006.

[18] Openscenegraph. http://www.openscenegraph.org/. Accessed May 2006.

[19] G. Papagiannakis, A. Foni, and N. Magnenat-Thalmann. Practical precomputed radiance transfer for mixed reality. In *Proceedings of Virtual Systems and Multimedia 2005*, pages 189–199. VSMM Society, 2005.

[20] G. Papagiannakis, H. Kim, and N. Magnenat-Thalmann. Believability and presence in mobile mixed reality environments. In *IEEE VR2005 Workshop on Virtuality Structures*, 2005.

[21] K. Perlin. Real time responsive animation with personality. *IEEE Transactions on Visualization and Computer Graphics*, 1(1), 1995.

[22] I. Poggi, C. Pelachaud, F. D. Rosis, V. Carofiglio, and B. D. Carolis. Greta: A believable embodied conversational agent. In O. Stock and M. Zancanaro, editors, *Multimodal Intelligent Information Presentation*, volume 27. Springer Verlag, 2005.

[23] M. Ponder, G. Papagiannakis, T. Molet, N. Magnenat-Thalmann, and D. Thalmann. Vhd++ development framework: Towards extendible, component based vr/ar simulation engine featuring advanced virtual character technologies. In *Proceedings of Computer Graphics International (CGI)*, pages 96–104. IEEE Computer Society Press, 2003.

[24] C. Rose, M. Cohen, and B. Bodenheimer. Verbs and adverbs: Multidimensional motion interpolation. *IEEE Computer Graphics and Applications*, 18(5):32–48, September/October 1998.

[25] C. Rose, B. Guenter, B. Bodenheimer, and M. Cohen. Efficient generation of motion transitions using spacetime constraints. In *Proceedings of ACM SIGGRAPH 1996, Annual Conference Series*, pages 147–154, August 1996.

[26] P. P. Sloan, J. Kautz, and J. Snyder. Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. In *Proceedings of ACM SIGGRAPH 2002*, pages 527–536. ACM Press, 2002.

[27] H. Tamura. Mixed reality: Future dreams seen at the border between real and virtual worlds. *IEEE Computer Graphics and Applications*, 21(6):64–70, November/December 2001.

[28] B. Thomas, B. Close, J. Donoghue, J. Squires, P. De Bondi, M. Morris, , and W. Piekarski. Arquake: An outdoor/indoor augmented reality first person application. In *Proceedings of Symposium on Wearable Computers*, pages 139–146, 2000.

[29] M. Unuma, K. Anjyo, and T. Tekeuchi. Fourier principles for emotion-based human figure animation. In *Proceedings of ACM SIGGRAPH 95, Annual Conference Series*, pages 91–96, 1995.

[30] L. Vacchetti, V. Lepetit, M. Ponder, G. Papagiannakis, P. Fua, D. Thalmann, and N. Magnenat-Thalmann. Stable real-time ar framework for training and planning in industrial environments. In S. K. Ong and A. Y. C. Nee, editors, *Virtual Reality and Augmented Reality Applications in Manufacturing*. Springer-Verlag, 2004. ISBN: 1-85233-796-6.

[31] V. Vlahakis, N. Ioannidis, J. Karigiannis, M. Tsotros, M. Gounaris, D. Stricker, T. Gleue, and L. Daehne, P.and Almeida. Archeoguide: An augmented reality guide for archaeological sites. *IEEE Computer Graphics and Applications*, 22(5):52–60, 2002.

[32] D. Wiley and J. Hahn. Interpolation synthesis of articulated figure motion. *IEEE Computer Graphics and Applications*, 17(6):39–45, 1997.