UPGRADE is the European Journal for the Informatics Professional, published bimonthly at <http://www.upgrade-cepis.org/>

UPGRADE
The European Journal for the Informatics Professional
http://www.upgrade-cepis.org
Vol. VII, issue No. 2, April 2006

Publisher
UPGRADE is published on behalf of CEPIS (Council of European Professional Informatics Societies, <http://www.cepis.org/>) by **Novática** <http://www.ati.es/novatica/>, journal of the Spanish CEPIS society ATI (*Asociación de Técnicos de Informática,* <http://www.ati.es/>)

UPGRADE monographs are also published in Spanish (full version printed; summary, abstracts and some articles online) by **Novática**

UPGRADE was created in October 2000 by CEPIS and was first published by **Novática** and **INFORMATIK/INFORMATIQUE**, bimonthly journal of SVI/FSI (Swiss Federation of Professional Informatics Societies, <http://www.svifsi.ch/>)

UPGRADE is the anchor point for **UP**ENET (**UP**GRADE European NETwork), the network of CEPIS member societies' publications, that currently includes the following ones:
• **Informatik-Spektrum**, journal published by Springer Verlag on behalf of the CEPIS societies GI, Germany, and SI, Switzerland
• **Mondo Digitale**, digital journal from the Italian CEPIS society AICA
• **Novática**, journal from the Spanish CEPIS society ATI
• **OCG Journal**, journal from the Austrian CEPIS society OCG
• **Pliroforiki**, journal from the Cyprus CEPIS society CCS
• **Pro Dialog**, journal from the Polish CEPIS society PTI-PIPS

UPGRADE **Newslist** available at
<http://www.upgrade-cepis.org/pages/editinfo.html#newslist>

ISSN 1684-5285

Monograph of next issue (June 2006)
Software Licenses
(The full schedule of **UP**GRADE is available at our website)

Monograph: Virtual Environments
(published jointly with Novática*)

Guest Editors: *Jesús Ibáñez Martínez, Carlos Delgado-Mata, and Ruth Aylett*

UPENET (UPGRADE European NETwork)

* This monograph will be also published in Spanish (full version printed; summary, abstracts, and some articles online) by **Novática**, journal of the Spanish CEPIS society ATI (*Asociación de Técnicos de Informática*) at <http://www.ati.es/novatica/>.

Creating Three-Dimensional Animated Characters: An Experience Report and Recommendations of Good Practice

Michael Nischt, Helmut Prendinger, Elisabeth André, and Mitsuru Ishizuka

This paper provides a brief overview of state-of-the-art graphics techniques and tools for modelling and animating highly realistic and expressive characters. All techniques are explained with reference to our ongoing project on the creation of life-like characters for real-time applications, performed jointly with a professional Japanese character animator. Based on our experience with currently available software for digital content creation, we will also suggest methods of good practice and give recommendations.

Keywords: Animation, Good Practices, Japanese-style Characters, Three-dimensional Characters, Virtual Human Characters.

1 Introduction

With the ever growing need for delivering high-quality content and information effectively, we can observe an increased awareness of the huge potential for employing sophisticated life-like characters (or virtual humans) in sales, advertisement, education, research promotion, and of course, gaming and entertainment, including edutainment and infotainment [1]. However, the integration of highly realistic characters into a virtual environment typically involves significant effort. A core step in this process is to transfer the three-dimensional virtual human (or character) models and animations from Digital Content Creation (DCC) tools into a format that can serve as input for the target application. In short, a designer starts out with creating the visual appearance of the character. After that, the model's geom-

Authors

Michael Nischt is a Student Researcher in Computer Science and Multimedia at the Augsburg University, Germany. Since 2001, he has been supervising practical courses on computer games and animated agents. Currently, he is completing his Master's thesis on an agent-based interface to graphics engines. From the very beginning of his studies, the animation and rendering of virtual characters has been one of his main research interests. While conducting practical projects in this area, he has achieved comprehensive experience in utilizing both OpenGL and DirectX as well as their shading languages for hardware accelerated graphics programming. Furthermore, he has contributed a full article to the fourth volume of the ShaderX series, which is a collection covering state-of-the-art programming techniques for today's graphic-cards. <Michael.Nischt@monoid.net>

Helmut Prendinger is Associate Professor at the National Institute of Informatics (NII), Tokyo, Japan. He previously held positions as a JSPS (Japan Society for the Promotion of Science) Research Associate and JSPS Post-doctoral Fellow at the University of Tokyo, Ishizuka Laboratory. Earlier he worked as a junior specialist at the University of California, Irvine, USA. He received his MA and PhD degrees from the University of Salzburg, Dept. of Logic and Philosophy of Science and Dept. of Computer Science, Austria. His research interests include artificial intelligence, intelligent multi-modal interfaces, and affective computing, in which areas he has published more than 70 papers in international journals and conferences. He is a co-editor (with Mitsuru Ishizuka) of the book "Life-Like Characters. Tools, Affective Functions, and Applications" that appeared in the Cognitive Technologies series of Springer in 2004. <helmut @nii.ac.jp>

Elisabeth André is a Full Professor of Computer Science and Chair of the Laboratory for Multimedia Concepts and Applications at Augsburg University, Germany. She received a Diploma Degree and a PhD. in Computer Science from Saarland University, Garmany. Prior to her university appointment, she worked as a principal researcher at the German Research Center for Artificial Intelligence (DFKI GmbH) in Saarbrücken. Her research interests include conversational embodied agents, affective computing, intelligent multimedia interfaces, and the integration of vision and natural language. She is on the editorial board of various international journals, such as Artificial Intelligence Communications (AICOM), Cognitive Processing (International Quarterly of Cognitive Science), Universal Access to the Information Society (UAIS), Autonomous Agents and Multi-Agent Systems (JAAMAS).
<andre@informatik.uni-augsburg.de>

Mitsuru Ishizuka is a Professor of the Graduate School of Information Science and Technology, University of Tokyo, Japan. Previously, he worked at NTT Yokosuka Laboratory and Institute of Industrial Science, University of Tokyo. During 1980-81, he was a visiting Associate Professor at Purdue University, USA. He received his BSc, MSc and PhD. degrees in electronic engineering from the University of Tokyo. His research interests are in the areas of artificial intelligence, multimodal media with lifelike agents, Web intelligence, and next-generation Web foundations. He is a member of IEEE, AAAI, Japanese Society for Artificial Intelligence (currently, President), IPS Japan, IEICE Japan, among others. <ishizuka@i.u-tokyo.ac.jp>

etry is extended with one or more parametrizable deformation techniques, for example an underlying skeleton. Using those techniques, a virtual human can be an animated in a comfortable way, which is quite often done by setting keyframes at specific times. Finally, the computerized data needed by the application, is exported into a dedicated file format.

In view of the steps just described, it can be a challenging task to add new human models to a virtual scene or exchange them with more appropriate ones that match the scenery of the target application. In our case, for instance, we aimed at creating two unique Japanese-style characters performing a (virtual) presentation, which implied certain constraints on their style of expression and set of gestures, and differences to western-style models. The sections in this paper provide both an experience report and a list of solution for animating highly-realistic and expressive virtual humans.

2 Brief Overview of Transferring Characters from Animation Tools to The Target Applications

When evaluating libraries dealing with the creation of virtual humans, we noticed that they are often deployed with self-developed special purpose tools, which for example are used to define facial animation parameters, although the DCC tool that was used to create the model offers the same functionality. In our experience, the main problem with such in-house tools is that neither the artist nor the programmer feels comfortable when working with them. It is hence preferable to let the artists work with the tools most convenient for them, which will likely result in saving time creating a model and a better quality. For this reason, our design strategy is to make ample use of the design tools' rich functionality. However, we observed that getting familiar with the respective SDK (Software Development Kits) of the modelling tools, which are essential to export the additional data directly, can result in even more effort than simply providing an in-house tool.

Another challenge was the absence of standard file-formats dealing with animated characters. As to static geometry it would not have been a problem to find an appropriate format. However, we needed one capable of skeletal subspace deformation (a.k.a. soft-skinning) and morph-targets (a.k.a. blend-shapes), which we use among others for the (dynamic) facial animation. During our evaluation of the few existing formats containing the needed information, we encountered two issues: either there was no public available export plug-in for the DCC tools we use, or the file-formats were not designed for dynamically changing virtual scenarios similar to computer/video games. In the following paragraphs, we report on our experience with three standard formats, all of which meet the requirements mentioned above.

At first we considered using X3D [2], which includes the H-Anim [3] standard and is therefore capable of describing virtual characters. Unfortunately, we did not find any free available DCC plug-ins, which could export all of the required features, although they are included in the format. For example, some could handle the skeletal animations well, but did not support facial animation in terms of

```
<Mesh>
 <Vertices count="5" maps="1" elements="2">
  <Map>0 3 1 2 3 2</Map>
  <VertexElement name="position">
   <Coordinates dimension="3" count="5" unique="4">
    8.0 -12.5 ... <Split/> 2
   </Coordinates>
  </VertexElement>
  <VertexElement name="texture" map="0">
   <Coordinates  dimension="2" count="4" unique="3">
     <File name="model.bin" offset="0" real="float" natural="short"/>
   </Coordinates>
  </VertexElement>
 </Vertices>
 <Primitive groups="2">
  <Triangles count="2">0 1 2  0 3 4</Triangles>
  <Triangles count="4" material="brick">
     <File name="model.bin" offset="18" natural="int"/>
  </Triangles>
 </Primitives>
</Mesh>
```

Figure 1: An Example Geometry File.

Displacer objects defined by H-Anim. Moreover, we discovered that the format itself was not perfectly suited to our application requirements, since it combines both geometry and runtime behavioral descriptions in a one single format, which is adequate for its purpose – defining interactive web- and broadcast-based 3D content – but not for our targeted distributed agent-based architecture.

Other alternatives included the two 3D asset exchange formats, COLLADA [4] and Fbx [5]. Since their primary purpose is to transfer complete virtual sceneries from one DCC tool to another, they both contain all the required information. On the other hand, neither of them is designed for real-time applications that would require a fast loading mechanism. We therefore do not recommend them as direct input. However, we would like to mention that they are great content holders and all the information needed by individual applications could in principle be extracted and transferred into more optimized or specialized formats. Similar to the problems mentioned regarding X3D, we noticed the issue of the unavailable exporter (although we expect this problem to be solved in the near future).

Summing up, we believe that while all of the above mentioned file formats are well suited for their main purpose, they are not adequate for optimized and real-time input. Nevertheless, they might all serve well as a foundation, that is, they can be used to extract and transform the needed data (assuming that the functionality of the available exporter eventually increases). Up to now, however, there is no way out other than working with the SDK of a DCC tool for writing one's own export plug-in or export script. At the time of writing, the first author of this paper has implemented exporters for both Maya [6] and 3ds Max [7], which are the most common animation software programs for industry professionals. We further decided in favor of their embedded scripting languages (mel and maxscript), because the implementation process is usually faster and the export functionality is typically not a performance critical part.

3 Assembling Virtual Humans

As previously mentioned, file-formats targeting real-time applications or managing populated virtual worlds have specific demands. Here, we briefly describe how our format and the loading mechanism are designed to accomplish those demands. We will then continue with describing the individual modules required for our virtual human models.

In contrast to most existing formats that are either human readable or have a binary representation, we advocate a hybrid format. More specifically, we allow for swapping the geometrical data into binary files while storing only the related semantic and the remaining data in an XML-based file. The benefits of the eXtensible Markup Language (XML) are clear, as it not only allows us to change descent values by hand, but also to take advantage of existing, highly optimized parsers available for any platform. Utilizing those parsers, the format can be extended with further nodes and attributes without violating syntactical constraints. On the other hand, formats related to computer graphics usually store a huge amount of geometrical data, which, if embedded in XML, may slow down reading as well as increase the memory consumption. This drawback is especially critical for programs that require loading new models at run time.

We further think it to be a good practice to ship an SDK with a format, as it is also the case for Fbx. This decreases the workload of developers, as they do not have to write their own parsers and other utilities. Accordingly, we programmed a set of interfaces corresponding to our XML-Schemas (XSD) as well as a file-loading mechanism. Since parsers only map the binary files instead of loading the data into the memory, another benefit of the hybrid format became apparent. Using the interfaces to transfer the geometrical data into an application dependent representation is highly time-efficient, as almost no intermediate structures are used.

With the general architecture as a basis, the following sub-sections will discuss the individual modules necessary for animating the characters.

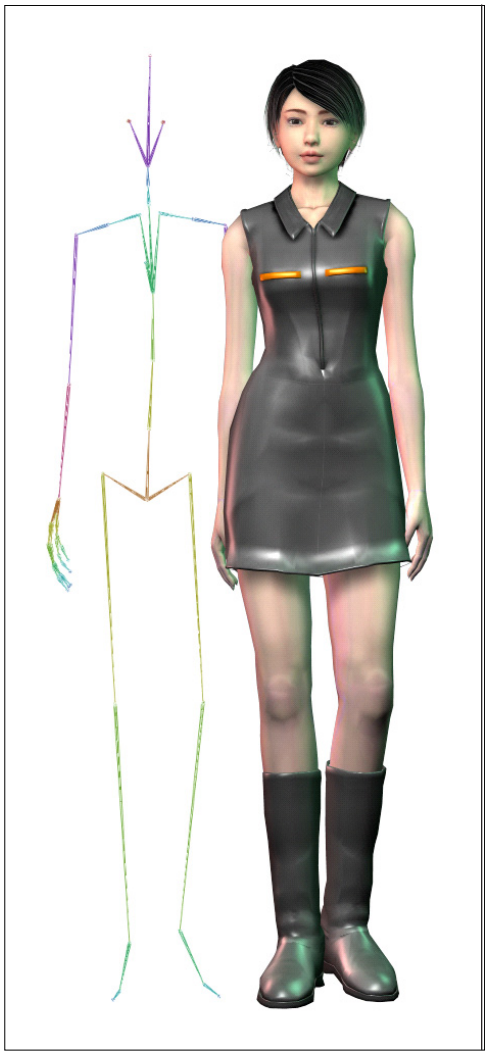

Figure 2: The Female Japanese Model and its Skeleton.

3.1 Geometry

Obviously the geometry of a virtual model, which defines its shape, constitutes the foundation of a character model. Consequently a large number of different geometrical representations for three-dimensional models have been developed in the history of computer graphics. Probably the most common ones are polygonal meshes, parametric surfaces (e.g. NURBS), subdivision surfaces, and meta-ball modelling. Thus most DCC tools offer more than one technique for creating objects; however, today's graphic-cards are mainly optimized for processing triangles. Due to this fact and because all modelling tools have the built-in functionality to convert the other supported representations to polygons, we decided to focus on triangle meshes. Nevertheless, the format can include other hardware supported primitives like *quads*, *polygons*, *lines* or simply *points*, because all of these topologies can be represented as lists of natural numbers.

The remaining geometrical data is divided semantically in a similar way as the hardware abstracting APIs (Application Program Interfaces) do, and can therefore easily be compared to Direct3D's Vertex-Elements or OpenGL's Vertex-Attributes. Although these allow us to use scalar and simple vectors with dimensions up to four (for some primitive types), we decided to deal with the floating points variants only, in order to keep the implementation simple. Probably the main difference compared to other implementations is that we keep the information about which positions are duplicated, because two faces referencing them share a hard-edge as well as the duplicated texture coordinates occurring when two neighboring faces have different materials assigned. Maintaining this knowledge allows for an optimized deformation, since only the unique coordinates have to be transformed. By way of example, Figure 1 shows a partly swapped geometry file.

3.2 Materials

After defining the shape of a virtual object, illuminating it properly is key to the illusion of three-dimensionality on a flat screen. The surface properties determining the reflection, refraction and scattering of an incoming light ray are therefore described as the object's material. It is actually even possible to assign multiple materials to a single mesh, by associating them with the individual faces. Furthermore, various types of more or less different lighting models were developed in the past and many of them are supported by today's DCC tools. However, hardware accelerators used to be limited to a single one.

Fortunately, this has changed during the last years and the current programming model has evolved from cumbersome assembly languages to more high-level ones (for instance, Microsoft's HLSL, High Level Shader Language; OpenGL's GLSL, GL Shading Language; or nVidia's CG). As a result, our materials mainly consist of such shading scripts and their input parameters, which are either, attribute types, scalars as well as vectors and matrices of small dimensions, or textures that fetch their data mainly from bitmaps. Our current implementation uses the Phong Illumination Model exclusively, but we already are working on others – for example to improve the realism of the characters' hair.

3.3 Deformers

While the geometry and its associated materials are perfectly sufficient when dealing with rigid bodies only, human models are intrinsically non-rigid. Hence techniques for deforming the geometry according to intuitive parameters are needed. In the case of real-time animated characters, mainly two techniques are employed: skeletal subspace deformation and morph-targets. The first one uses an underlying skeleton to transform the geometry's coordinates (see Figure 2 for a bone setup that we used for our characters). In order to prevent creasing around joints, a single coordinate is usually modified according to the movement of more than one bone. Defining their influence weights well inside the DCC tool is therefore extremely important for achieving a good visual quality. While this technique works well for body movements such as gestures, morph-targets are also well suited for simulating smaller, more local displacements, in particular in the face. Here, each

Figure 3: Three Different Emotional Expressions of the Japanese Male Model. From Left to Right: Sadness, Surprise, Joyful.

```
<Expressions>
 <!— Phoneme - Viseme  Mapping —>
 <Sum id=”Er”> <!— Viseme —>
  <Scalar id=”EH”/> <!— Phoneme —>
  <Scalar id=”ER”/>
  <Scalar id=”EY”/>
 </Sum>
 ..
 <!— Viseme - Facial-Feature Mapping —>
 <Sum id=”JawOpen”> <!— Facial Feature —>
  <Product>
   <Scalar>0.2</Scalar> <!— Influence Factor —>
   <Scalar id=”EE”/> <!— Viseme —>
  </Product>
  <Product>
   <Scalar>0.3</Scalar>
   <Scalar id=”Er”/>
  </Product>
  ..
 </Sum>
 ..
</Expressions>
```

Figure 4: Excerpt of a Phoneme-To-Viseme-To-Facial-Feature File.

morph-target has typically a feature associated with it, like ‘left_eyebrow_up’, which can be compared to MPEG4’s (Moving Picture Experts Group) facial animation parameters (FAPs), or an emotional state like ‘surprise’ (see Figure 3). It is important to observe that both can be computationally expensive for highly detailed characters. As a result, it is a good practice to layout the data in a way that the deformation algorithms can exploit the parallel execution capabilities of current and upcoming Multi-Core CPUs and GPUs (Graphics Processing Units). We advise a representation similar to Vertex-Elements that were mentioned in the section on geometry. This approach supports compatibility with the graphics pipeline’s geometry stage of future hardware, or use of other advanced techniques like those described in [8].

3.4 Animations

Having the geometry of a character extended with the deformers as mentioned above, the only remaining task is to change their parameters over time. A common technique is to define a set of key-frames for specific time points, which are to be interpolated in order to obtain smooth animations. A good practice is to use either uniform distributed key-times, which can be implicitly defined by their quantity, or to use an ordered set of floating point values between zero and one. The main benefit of both alternatives is that the key-times stay invariant, changing the animation’s duration. Key-values are usually grouped semantically into so-called channels, which can be compared to a scalar-valued array. For example, the key-frames corresponding to a joint’s rotation are distributed over three channels, where each one contains the rotation angles about one of the axes of the 3D-space. Alternatively each channel can be associated with its own key-times or share them with others. Finally, we associate non-overlapping partitions of an animation with a dedicated (application-dependent) semantics. For instance, gesture animations can be divided into a starting, performing and finishing phase. This is especially useful if the middle part can be played as a loop. Regarding speech animations containing phoneme channels, we advise proportioning according to the spoken words.

4 Extensions – Animation Transitions and Lip Synchronization

The implementation of the modules as described in the previous sections provides a fairly nice foundation for a character animation system. Obviously, various extensions are needed in order to increase the flexibility of the system. In this section, we will list some of them. First, we might want to manipulate the neck and eye joints directly, based on target coordinates or objects the character should focus (see [9] for a detailed description).

An especially worthwhile feature is automatically generated transitions from one animation to another. This is done by comparing the rotation angles of the individual joints at two specific times in order to compute the transition time. More precisely, we use the end of the performing phase of the first gesture and the start of the second gesture’s performing phase. However, if the two poses of the skeleton are too different, we play the first gesture until completion, which in our case is always a neutral gesture and afterwards start the other from its beginning.

Finally, we wanted our characters to speak with a proper lip synchronization based on weighted phonemes, which come either directly from a text-to-speech (TTS) system or are extracted from audio files by using a tool like LipSync [10]. The challenge here is how to map the phonemes to the facial features of an individual model, which can vary by having different morph-targets for the individual models. For general purpose, we recommend all of the sixty-eight Facial Definition Parameters (FDPs) defined by the MPEG4 standard (see [11] for more details). However, creating blend-shapes for all of them is time-consuming and not always necessary. We therefore confined the number of blend-shapes to twenty meta-expressions, which is completely sufficient for our purpose of achieving realistic and natural lip synchronization. In order to keep our implementation independent of the facial features defined for a particular model, we are using simple mathematically expressions to first map the phonemes to visemes (mouth positions corresponding to phonemes) and then map these to the facial parameters according to individual influence factors for each model. Figure 4 shows two sample expressions embedded into an XML file.

5 Conclusions

This paper walked the reader through a series of steps required for the creation of highly realistic and high-quality life-like characters. We discussed state-of-the-art graphics techniques and related tools for handling all aspects character modelling and generation, including geometry, materials, deformers, animation, as well as between-gesture transition and lip synchronization.

A task complementary to that of defining and animating a visual character is to provide intuitive means for directing the behavior of one or more characters. We previously developed the Multimodal Presentation Markup Language (MPML) as an easy-to-use scripting language for controlling cartoon-style characters in web and mobile applications as well as the ASIMO humanoid robot from Honda (see [12] for an overview). Our current efforts are dedicated to developing a MPML-style authoring language, tentatively called MPML-3D, that will allow digital content creators to effortlessly generate attractive presentations and interactions with the three-dimensional characters described in this paper.

Acknowledgements

The first author was supported by an International Internship Grant from NII (National Institute of Informatics, Japan) under a Memorandum of Understanding with the University of Augsburg, Faculty of Applied Informatics, Germany. The research was supported by the Research Grant (FY1999-FY2003) for the Future Program of the Japan Society for the Promotion of Science (JSPS), by a JSPS Encouragement of Young Scientists Grant (FY2005-FY2007), and an NII Joint Research Grant with the University of Tokyo (FY2005).

References

[1] Helmut Prendinger and Mitsuru Ishizuka (Eds.). Life-Like Characters. Tools, Affective Functions, and Applications, Cognitive Technologies Series, Springer, Berlin Heidelberg, 2004.
[2] X3D Specification, <http://www.web3d.org/x3d/>.
[3] H-Anim Specification, <http://www.h-anim.org/>.
[4] COLLADA Specification, <http://collada.org/>.
[5] Autotdek's Fbx, <http://www.autodesk.com/fbx>.
[6] Autotdek's 3ds Max, <http://www.autodesk.com/3dsmax>.
[7] Autotdek's Maya, <http://www.autodesk.com/maya>.
[8] Michael Nischt and Elisabeth André. Real-Time Character Animation on the GPU, In Wolfgang Engel (Ed.), ShaderX4: Advanced Rendering Techniques , Charles River Media, pp. 47-56, 2006.
[9] Jeff Lander. To Deceive is To Enchant: Programmable Animation, In Game Developer magazine, CMP Media LLC (May 2000).
[10] LipSync Tool, <http://annosoft.com/lipsync_tool.htm>.
[11] Jörn Ostermann, Face Animation in MPEG-4. In Pandzic, I. S. and Forchheimer, R. (Eds.) MPEG-4 Facial Animation - the Standard, Implementation and Applications , pp. 17-56, John Wiley & Sons., Chichester England, 2002.
[12] Mitsuru Ishizuka and Helmut Prendinger. Describing and generating multimodal contents featuring affective lifelike agents with MPML. New Generation Computing 24, pp. 97-128, 2006.