The International Journal of Virtual Reality, 2006, 5(2):25-30

The Virtual Human Platform: Simplifying the Use of Virtual Characters

Christian Knöpfle and Yvonne Jung

Abstract— **In this paper, we will explain our approach to create and animate virtual characters for real-time rendering applications in an easy and intuitive way. Furthermore we show a way how to develop interactive storylines for such real-time environments involving the created characters. We outline useful extensions for character animation based on the VRML97 and X3D standards and describe how to incorporate commercial tools for an optimized workflow. These results were developed within the Virtual Human project. An overview of the project is included in this paper.**

Index Terms— **Animation, Real-Time Rendering, virtual characters**

I. MOTIVATION

The simulation of virtual characters is a complex topic in research for many years. Such a simulation comprises many research aspects, ranging from digital storytelling, artificial intelligence to animation and real-time rendering. The ultimate goal is to create virtual human beings, which look, act and react like we do.

But why is research on virtual characters so important? Because virtual characters could be the next generation user interface, which helps people to understand and use machines, computers, any other electronic device in an easier way. Especially when it comes to complex tasks, virtual characters can be much more helpful than today’s user interface. The reason is simple: if we are able to communicate with the machine in the same way we do with other human beings, an ability we train every day, nobody would have to learn new interaction paradigms, concepts etc. and could use his native skills. The machine would adopt to the user and not vice versa (as it is today). Such virtual characters could even help us improve our communication skills. An example is management training with the focus on the social interaction with employees. The Virtual Character could show how to do it right or could be the “sparring partner” for the user of the application.

Still a lot of research has to be carried out to come close or even reach the ultimate goal, and many remarkable results in all areas related to virtual characters have been made.

But if we look at industry, the deployment of virtual characters is very limited; computer games industry seems to be the only user of the technology. The major reason is that the creation of characters is tedious, expensive and needs very skilled and talented character designers to achieve convincing results. One can say that only the computer games industry has the appropriate budgets and the willingness to invest.

In 2002 we started to work on the Virtual Human research project [16] and until now we realized three complex demonstrations. Beside the technical challenges we planned to work on, other problems came up, which sometimes took us more time to solve and made the realization of the demonstrations very tedious. These problems were mainly the same, which hinders the deployment in industry: The creation of the virtual character model, its animations and behavior. Thus we started to develop methods and tools, which made our live easier and ease the use of virtual characters for a wide range of application types. In this paper we describe one way to create and use virtual characters much easier.

The paper is organized as follows: First we describe the Virtual Human project and the developed technology platform. Then we focus on the creation of the virtual character geometry and animation. Then we describe how the actual story and the behavior could be created. We finish with a summary and outlook for future work.

II. VIRTUAL HUMAN PROJECT

Virtual Human has been initiated as research project funded by the Federal Ministry of Education and Research in Germany. The global aim of Virtual Human is to combine Computer Graphics technology provided by the INI-GraphicsNet (Fraunhofer IGD, ZGDV Darmstadt and TU Darmstadt) with speech and dialogue processing technology provided by the German Research Center for Artificial Intelligence (DFKI) in order to develop methods and concepts for “realistic” anthropomorphic interaction agents. In addition, the third major project partner Fraunhofer IMK is responsible for the domain model of the Virtual Human application scenario.

Whereas interactive storytelling techniques are primarily used for the dialogue and narration engine as control unit of the Virtual Human run-time environment, computer graphics technology is used for realistic rendering and animation of virtual characters within the Virtual Human rendering platform.

The Virtual Human project started in November 2002; an early demonstration has been set up for summer 2003 demonstrating the basic principles of Virtual Human components. Another major step was the first integrated Virtual Human demonstration presented at the CeBIT 2004 exhibition fair in Hannover. It was an eLearning scenario with two virtual

Manuscript Received on August 20, 2006

Christian Knöpfle is with the Fraunhofer Institute for Computer Graphics IGD (address: Fraunhofer-IGD, Fraunhoferstr. 5, 64283 Darmstadt, Gernamy; e-mail: christian.knoepfle@igd.fraunhofer.de).

Yvonne Jung is with the Fraunhofer Institute for Computer Graphics IGD e-mail: yvonne.jung@igd.fraunhofer.de.

characters, a teacher and a pupil (see Fig. 4). Astronomy was the topic and it was possible to adjust the behavior of the pupil from friendly to nasty. The teacher reacted accordingly. Since CeBIT 2004 went very well, the consortium focused his work on the next scenario. There new challenges could be tackled, e.g. more interactivity, more involved characters (real and virtual) and the flow of the story at a much faster pace.

The second scenario was developed with the football championship 2006 in mind, which took place in Germany. The game based scenario was called ZAMB, an abbreviation for "Zweiundachtzigmillionen Bundestrainer" (82 million soccer coaches). Since every German believes that he knows best how to set up the soccer team, we built a game where he could try for his ideas. The user puts the soccer players on different positions (keeper, striker etc.) on a virtual soccer ground; two virtual characters commented on his actions and provided feedback and hints in real-time (see Fig. 4). This scenario put a great burden onto the dialogue and story engine, because there were numerous possibilities of setting up the players resulting in a broad variety of different actions of the virtual characters, which have to be calculated and set-up in real-time. The first results were shown at the INTETAIN 2005 conference in Madonna di Campiglio / Italy and the final result was shown at CeBIT 2006.

In the next sections we describe the core components of the Virtual Human platform and then focus on the control language PlayerML and the Avalon graphics system.

A. *The Core Components of the Virtual Human Platform*

From the technical point of view, Fig. 1 provides an overview of the major components of the Virtual Human platform and indicates partner responsibilities:

• The content layer consists of a domain model providing geometry, story models and character models, pedagogic models, media, etc. Furthermore, Virtual Human editors are used to create application scenarios and stories. The output of the content layer is a story board.

• The narration engine consists of a story engine [3] controlling a scene engine [13] and an improvisation module [9]. The outputs of the story engine are directions coded in Direction ML for the dialog engine [5]. Here, dialogues among virtual characters are generated during run-time and sent to the player component as Player ML scripts.

• Finally, the player component based on the Mixed Reality System Avalon [1] creates/renders the 3D environment and sends corresponding data to the visualization platform.

• Possible peculiarities of visualization platforms range from simple monitors or web-based application scenarios up to high-level rendering applications to be shown on a Powerwall. A more in depth description of the narration and dialogue engine can be found in [11].

B. *PlayerML – Player Markup Language*

The Player Markup Language (PlayerML) can be seen as a high level language to control virtual characters. No specific graphics knowledge is needed to use the language; the possible actions of the virtual character and its parameters are defined at

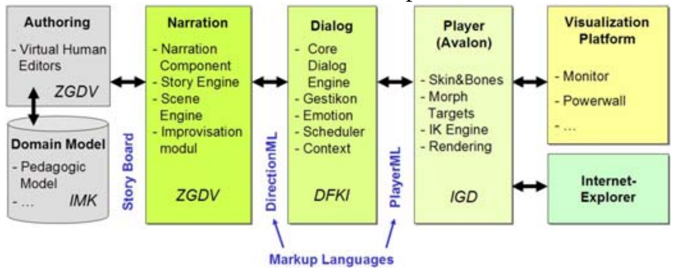


Fig. 1. Virtual Human Platform – Component based Architecture.

start-up of the system and can be used in any order and sequence. Thus an author could simply write a PlayerML script, send it to the player and then the graphics would react in the anticipated way.

PlayerML is a XML based Markup Language which takes up concepts from the RRL (rich representation language) [14]. PlayerML defines a format for sending instructions (commands) from the dialog-manager to a 3D virtual reality system (VR player Avalon [1]). Additionally it defines a message format which can be sent to or from a player.

PlayerML scripts are strictly scene-based. A *scene* describes the three dimensional space containing objects and virtual characters as well as all possible actions for objects or characters (e.g. movement, animation, interaction).

At the beginning of a new scene all static objects and characters are defined by *sceneDefinition* scripts. During the scene *actions scripts* describe all run-time dependent actions depending on their temporal appearance. PlayerML distinguishes between *SceneActions*, *CharacterActions* and *WorldQueries*. Hereby, *SceneActions* represent actions depending on the virtual environment (e.g. fade-in or fade-out of Objects). *CharacterAnimations* are actions which a virtual character can carry out (e.g. talking, smiling, walking, talking). Altogether, PlayerML is an abstract specification language. It is independent of the implementation of the VR player and the virtual environment.

In Virtual Human, PlayerML is used as descriptive interface markup language between a dialog creating environment (dialog engine) and the VR player and synchronizes all steps of dialog-generation. In the first step of the dialog generation, a very common representation of the dialog is generated. In the next step additional information (e.g. emotion, timing for lip-synchronization) augments the dialog-skeleton. In the third step, the dialog is enhanced by using fitting animations like mimic and other animations. The result is a script with a full definition of the actions and their temporal order. In a complex scenario, such an *action script* encodes duration of 5 to 20 seconds, meaning that the player receives many scripts in a short time. Nevertheless, such action scripts could encode a much longer duration.

The following example shows a slightly simplified PlayerML actions script. The animation tags refer to preloaded animations, which are referenced by their name. In complete PlayerML the tag sentence would contain a list of phonemes including their duration, which are mapped by the system to facial animations, as well as a source URL of an audio file, which contains generated speech.

```
<playerML id="s1">
    <actions>
        <characterAct id="ca2">
            <character refName="Sven"/>
            <animation id="a3" refName="pride"/>
                <sentence id="s4">
                    <text>This sentence shall be spoken proudly.
                    </text>
                </sentence>
            <animation id="a5" refName="progress"/>
            <temporalOrder>
                <seq>
                    <par>
                        <act refId="a3" begin="0" dur="4000"/>
                        <act refId="s4" begin="300" dur="3700"/>
                    </par>
                    <act refId="a5" begin="0" dur="2000"/>
                </seq>
            </temporalOrder>
        </characterAct>
    </actions>
</playerML>
```

Fig. 2. PlayerML sample script.

C. *Avalon Player*

Avalon is a component based Virtual and Augmented Reality system developed and maintained at Fraunhofer IGD and ZGDV. Within Avalon the behavior of the virtual world is defined by a scene graph following and extending the concepts of VRML/X3D [2]: The scenegraph describes the geometric and graphical properties of the scene as well as its behavior.

Each component in the system is instantiated as a node, which has specific input and output slots. Via connections between the slots of the nodes (called routes), events are propagated, which lead to state changes both in the behavior graph and usually in the visible representation from render frame to render frame.

The rendering backend of Avalon is OpenSG [8], an open source high performance and high quality scenegraph renderer. One major advantage of the architecture of Avalon is that new functionality can be integrated quite easily by adding new nodes.

One might wonder why PlayerML is not based on X3D concepts, because X3D is able to encode behavior too (in this case the behavior of the virtual character). In fact one of our first realization of PlayerML was based on X3D. Every action was encoded as a node and a route network was used to encode the temporal order (the *<par>* and *<seq>* elements in the example script). Theoretically it worked, but even very simple scripts were highly complex and hard to understand at the end. Therefore we “invented” a new language, where a scene could be encoded in single script.

D. *VRML / X3D Extensions for Character Animations*

The X3D based H-Anim standard [7] defines node types, which allow to store and play back virtual characters based on skins and bones. The standard is implemented in Avalon.

As soon as we have several different animations, which could be played back in varying order, the whole set-up gets complex and crowded with routes and interpolators. Maintaining and extending such a scenario is cumbersome.

To make things simple and also to embed it into our Virtual Human platform, we added three different nodes to the Avalon system: *AnimationContainer*, *TimedAnimationContainer* and *TimelineComposer*. Their definition is shown in Fig. 3.

```
PROTO TimedAnimationContainer
  field              SFString    name        ""
  field              MFNode      targets     []
  field              MFString    fieldnames  []
  field              MFNode      interpolators           []
  field              SFFloat     duration    0

PROTO AnimationController
  field              SFString    name        ""
  exposedField       MFNode      animationContainer      []

PROTO TimelineComposer [
  exposedField       SFString    command     ""
  eventOut           SFString    message
  exposedField       MFNode      animationController     []
```

Fig. 3. Proto definitions of the newly introduced nodes.

The *TimedAnimationContainer* stores all animation data for a specific animation, e.g. “wave-hands”. The animation interpolators are stored in the *interpolators* field. The targets they are acting on are stored in the *targets* field. Targets can be e.g. *HAnimJoint* for skin and bone animation or *HAnimDisplacer* for morph target based animation. Since there is not a unique mapping of interpolator-type to a specific field of the target, the field must be provided too (stored in fieldnames). The whole duration of the animation is given in the duration field.

The *AnimationController* stores all *TimedAnimation Container* nodes, which belong to a character or object. Thus we have now containers for a single animation as well as a container for the available animations of a specific character or object. Because of the generic design of these nodes, their use is not limited to character animation.

With these nodes, it is much easier to debug and maintain larger set-ups than using the X3D/H-Anim nodes only. Now all nodes belonging to a single animation are linked to one node. All needed routes are created automatically.

Finally the *TimelineComposer* is the PlayerML interface node, where PlayerML commands/scripts are sent to and executed. It has a reference to all available Animation -Controller nodes (all possible animations and actions).

III. A SIMPLIFIED PROCESS FOR CREATING NEW SCENARIOS

The two Virtual Human scenarios were very complex to set-up and since they represent and include the top notch results of the Virtual Human project, they are building on the newest technology. The most time consuming tasks were (a) creating the geometries, textures, shaders and animations of the virtual character, and (b) feeding all story related information into the dialogue and narration engines.

But there are a lot of other application areas, where such high-end technology is not needed, not affordable or not these applications we thought about another approach for creating interesting and useful scenarios with virtual characters. Still the creation of the character model and set-up of the story was the most tedious task. In the following we will describe this workflow in more detail.

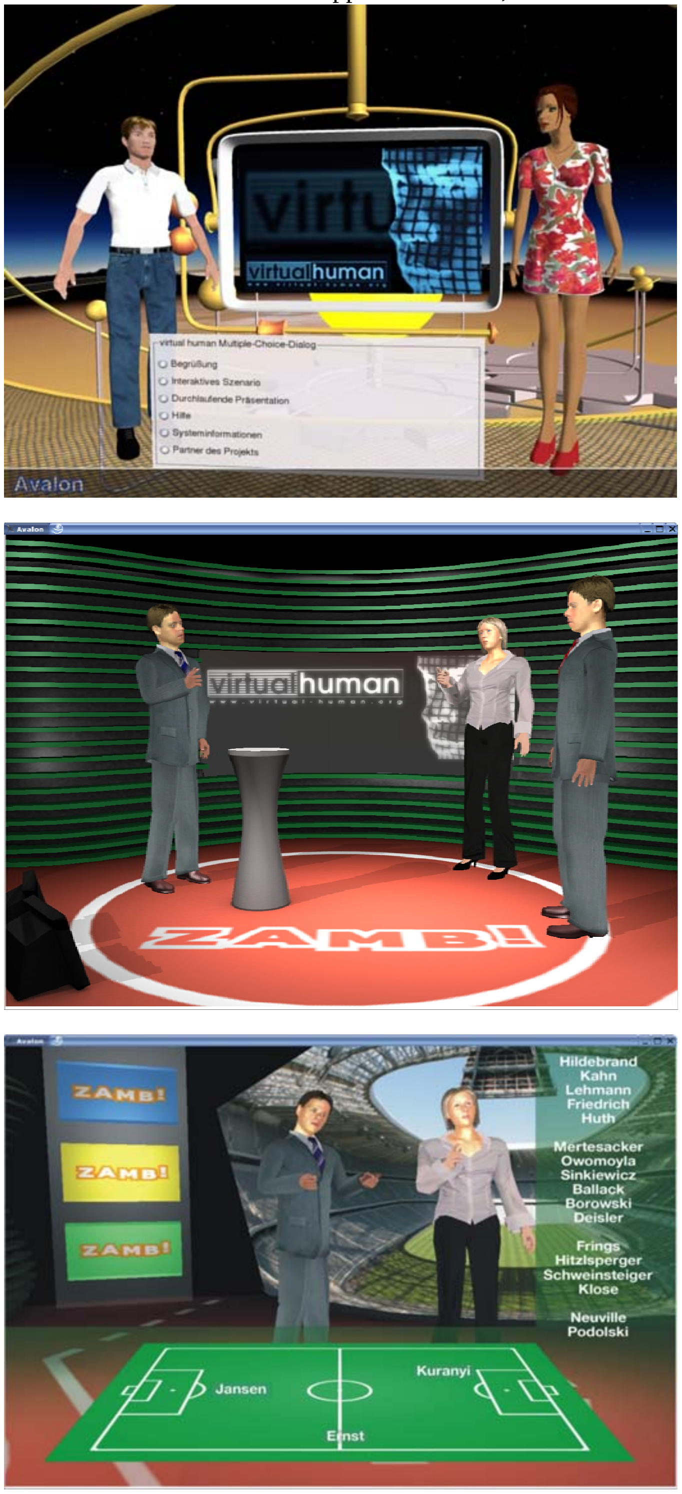


Fig. 4. Upper: The Virtual Human Demonstration at CeBIT 2004. (Background model courtesy of rmh new media GmbH, Character models courtesy of Charamel Software GmbH) Middle, Lower: The Virtual Human Demonstration ZAMB! at the INTETAIN05 conference (Background model courtesy of rmh new media GmbH, Character models courtesy of Fraunhofer-IMK). See Color Plate 9 & 10 & 11.

A. Creating Mesh Geometry and Animations Easily

To create a new virtual character, first its visual appearance must be modeled using a 3D modeling package like 3DSMax® or Maya®. This comprises of the creation of the polygon mesh as well as applying textures for good visual quality. It needs time and talented designers to create good character models.

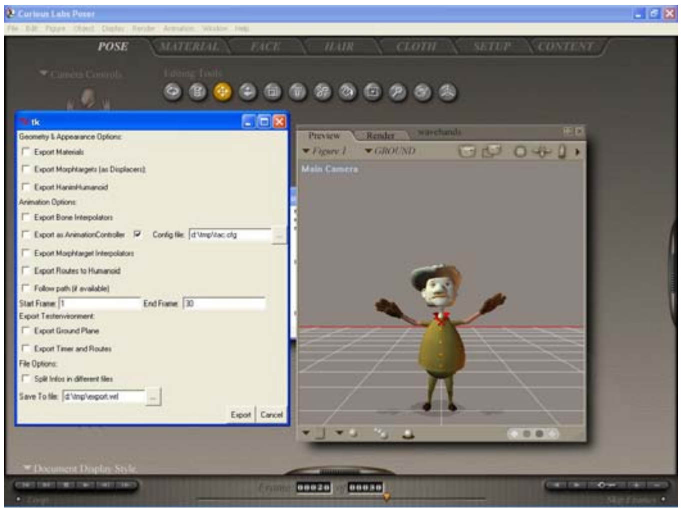
To animate such virtual characters in real-time, the skins and bones approach are used. Here the skin is the geometrical hull (polygon mesh) of the virtual character, which is rendered to the screen and comprises the visual appearance of the character. A linked bone system is attached to the skin, which is not rendered but used for animating the character. Each bone influences parts of the skin and a designer only has to animate the bones to create an animation of the whole character. A good introduction into skin and bones can be found in [17]. Such animation can be done by hand or based on motion capture data. Such data delivers the best visual quality but still needs to be cleaned and optimized by the designer.

As an alternative, character geometry can be created by 3D scanning of real persons, an approach which is also investigated in the virtual human consortium by Fraunhofer-IMK. Unfortunately there is still a lot of manual work needed to polish the scanned mesh, mainly reducing mesh complexity in order to get it usable for real-time animation and rendering. The animation of such a polygon mesh is still a problem. We could summarize that creation of the geometry for a virtual character as well as the accompanying animation (walking, gestures, mimics) is very time consuming and tedious. It needs talented designers to build 3d models from scratch.

To not reinvent the wheel, we analyzed a variety of commercial products for character creation. One tool, which seemed to us as fairly straightforward to use is Curious Lab's Poser™ [4]. Poser offers characters ranging from comic to realistic. Furthermore commercial companies offer additional character models at low price. Slight adaptations of the characters can be easily carried out in Poser. Most characters offer a wide range of morph targets (for visemes and emotions) which can be simply applied to form more complex animations. Even animating a character by hand is straightforward.

Unfortunately its export options are very limited and erroneous. For example to use the created animations and characters in 3DSMax the complete model must be adapted and revised. The H-Anim output only delivers full-body morph meshes, which is not suitable for real-time, interactive scenarios.

The good news is that Poser™ features a Python-based API, which allows accessing nearly all internal data structures. Therefore we used Poser as a base and implemented an export-feature (see Fig. 5), which allowed us to directly load the mesh and animation data into the Avalon system without any additional work to carry out. The output is X3D and H-Anim compliant but uses additional features of Avalon like Namespaces to easily split nodes in different files. The export utilizes the additional X3D nodes described in the sectoin above.



Geometry & Appearance Options:
- [ ] Export Materials
- [ ] Export Morphtargets (as Displacers)
- [ ] Export HanimHumanoid

Animation Options:
- [ ] Export Bone Interpolators
- [ ] Export as AnimationController ✓ Config file: d:\tmp\tac.cfg
- [ ] Export Morphtarget Interpolators
- [ ] Export Routes to Humanoid
- [ ] Follow path (if available)

Start Frame: 1 End Frame: 30

Export Testenvironment:
- [ ] Export Ground Plane
- [ ] Export Timer and Routes

File Options:
- [ ] Split Infos in different files

Save To file: d:\tmp\export.wrl

Export Cancel

Fig. 5. Poser™ user interface and option window of our export plugin.

The basic idea is now to load a character into Poser and adapt it to one's needs. Then all the different gestures, walks, mimics the character should carry out are modeled in Poser. For this we use the Poser timeline and arrange these entire animations one after the other, e.g. waving hand from frame 0 to frame 30, raising head from frame 31 to frame 45. Thus we need only a single Poser-file for all animations. After some experience, a user can easily create 30+ animations within a single day.

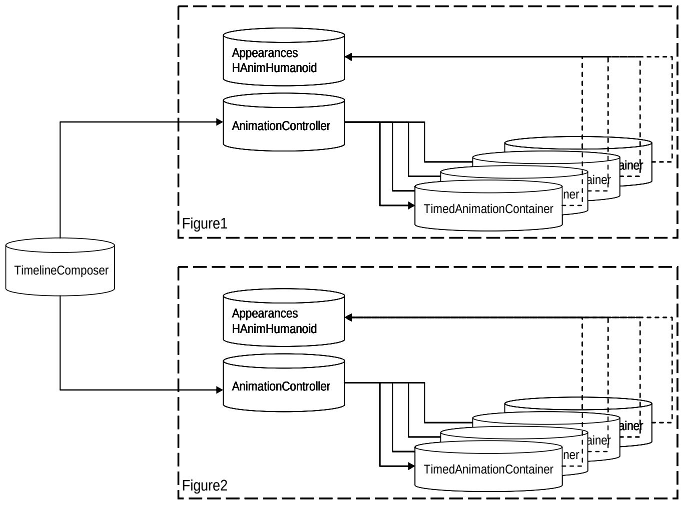


Fig. 6. Sample setup.

Finally the exporter dumps the character geometry as a *HAnimHumanoid* node into a file and each animation into separate files as a bunch of interpolators stored in a *TimedAnimationContainer* node. Furthermore *Animation Controllers* are created and initial PlayerML scripts for the setup are created. The Fig. 6 shows a setup for two characters with four animations each. All references between the nodes are generated automatically by the exporter.

B. *Online Generated Animation Data*

But the creation of animation data can still be tedious, for example if one needs pointing gestures in many different directions. For this we added direct ways for online creation of special animation data:

• Pointing and "look-at" gesture: Here we use inverse kinematics on the bone model to create convincing pointing gestures. The user only has to define a target the character should point to. The current implementation is based on geometric algebra [8].

• Walking: To let the character walk from one place to another we are currently porting the algorithms developed by [12] to Avalon. As soon as it is available, the user of the system will not need to create walking-animations, because they can be generated online. The base data needed by the algorithm to compute walk-cycles can be easily and automatically exported from Poser™.

The advantage of animation data created during run-time of the system is obvious: The creator of the character does not have to care about walking and pointing animations, but the author of the story could still issue "walk" and "point-at" commands.

C. *Creating the Digital Story*

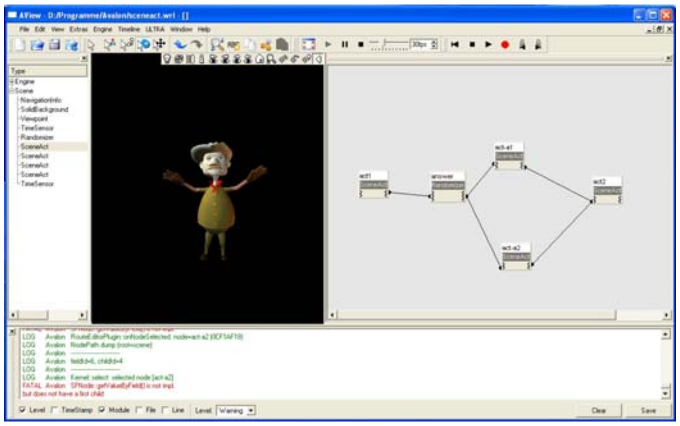
For the creation of the digital story we have several methods at hand. The most advanced ones in the Virtual Human consortium are those developed by ZGDV (narration engine) and DFKI (dialogue engine and management). Those engines have big potential when it comes to non-linear storytelling and adaptive dialogues. On the other hand authoring these engines is not trivial and takes a lot of effort to get a decent result. But for non-linear stories it is the best way to go.

But there are many applications where non-linearity and adaptive dialogues are not wanted or are not necessary. For such cases we developed tools allowing us to put together storylines in a very simple and intuitive way. As described earlier, a story can be described with PlayerML. PlayerML allows to define when and what a character / object in the scene does something. Furthermore how / when the user can interact with the virtual environment. The naïve approach would be to write one big PlayerML-script, where the whole story is described. This would allow no interaction by the user and there would be no chance to change the flow of the story in any way. The result would be similar to a film.

Therefore we choose a different approach. The idea is to define short acts and transform them into PlayerML scripts. A short act could be for example a dialogue between two characters on a given topic. Such PlayerML scripts will be stored in special *SceneAct*-Nodes (could be easily realized as X3D-PROTO), which are connected by routes. The routes define the flow of the story. As soon as such a node gets a trigger on its input field, its PlayerML script will be executed.

By adding other nodes in the route graph, we easily can add some non-linearity and possibilities for user interaction to the story. For example if we want to add randomized answers of a character triggered by a user interaction, we simply could add a Randomize node to the route graph. The user interaction triggers the Randomize node. To the output field of this node several *SceneAct* nodes are connected (via routes). The Randomize node chooses one of them and forwards the signal to it and the associated PlayerML script gets executed.

Beside the technological foundation for the story playback, we also implemented a first version of a graphical user interface for putting route graph (story graph) and the PlayerML scripts together. These GUIs are embedded in our Aview authoring framework (see Fig. 7).



```
LOG    Avalon  RouteEditorPlugin: onNodeSelected: node=act-a2 (0CF1AF18)
LOG    Avalon  NodePath dump (root=scene)
LOG    Avalon  ---------------
LOG    Avalon  fieldId=6, childId=4
LOG    Avalon  ---------------
LOG    Avalon  Kernel: select: selected node [act-a2]
FATAL  Avalon  SFNode: getValueByField() is not impl.
but does not have a first child
```

Fig.7. Authoring the story graph using AView

D. Runtime

After authoring is finished, the whole scene can be saved as VRML/X3D file and then played back using the Avalon rendering system. The user can interact with the system in the way it was defined during the authoring process.

IV. SUMMARY AND OUTLOOK

In this paper we presented our approach for an easy deployment of virtual characters in scenarios of medium complexity. We have shown how commercial tools can be adapted to allow fast and simple creation of character geometries and animations as well as integration into a real-time rendering system using an ISO standard. Furthermore we described a method to build simple interactive, non-linear stories. All these developments were integrated into the Virtual Human platform, which also supports highly complex scenarios.

We believe that having optimized workflows is a key issue for the wide adoption of virtual characters and our first steps in this direction seem to be promising. In the future we would like to optimize the creation, animation of virtual characters as well as implementation of storylines. This could be either by adding more automatisms for creating geometry and animation as well as by integrating other commercial tools.

ACKNOWLEDGMENT

The work described in this paper was carried out in the Virtual Human project, which is funded by the bmb+f (research ministry of the German government) and supported by the people of the Virtual Human consortium.

REFERENCES

[1] Avalon Virtual Reality System, http://www.ini-graphics.net/~avalon

[2] J. Behr, P. Dähne, M. Roth. Utilizing X3D for Immersive Environments, Web3D 2004 Symposium, Monterey, 2004

[3] N. Braun. Automated Narration – the Path to Interactive Storytelling, *Proceedings NILE*, Edinburgh, Scotland, pp.38-46, 2002

[4] Curious Labs Poser™, http://www.e-frontier.com/article/articleview/1597/1/281?sbss=281

[5] Patrick.Gebhard, Michael.Kipp, Martin.Klesen and Thomas. Rist.Adding the Emotional Dimension to Scripting Character Dialogues, in Proc. of the 4th International Working Conference on Intelligent Virtual Agents (IVA'03), Kloster Irsee, 2003

[6] S. Göbel, O. Schneider, I. Iurgel, A. Feix, C. Knöpfle and A.Rettig. Virtual Human: Story-telling & Computer Graphics for a Virtual Human Platform, on *Proceedings of Technologies for Interactive Digital Storytelling and Entertainment 2004*, Darmstadt, Germany

[7] H|Anim Humanoid Animation Working Group, http://www.h-anim.org

[8] Dietmar. Hildenbrand. Geometric Computing in *Computer Graphics* using Conformal Geometric Algebra, in *Computers & Graphics* 2005, vol.29, no. 5, October, 2005

[9] I. Iurgel. Emotional interaction in a hybrid conversational group. In: Prendiger, H. (ed.): International Workshop on Lifelike Animated Agents. Working Notes in Proceedings PRICAI-02, Tokyo, Japan,pp.52-57, 2002

[10] OpenSG, an open source scenegraph renderer, http://www.opensg.org

[11] N.Reithinger, P.Gebhard, M.Löckelt, A.Ndiaye, N. Pfleger and M. Klesen. VirtualHuman:Dialogic and Affective Interaction with Virtual Characters, To appear in: Maybury M. et al (eds.), Proceedings of the International Conference on Multimodal Interfaces (ICMI'06), November 2-4, 2006, Banff, Alberta, Canada.

[12] Sang Il Park, Hyun Joon Shin, Tae Hoon Kim and Sung Yong Shin. On-line motion blending for real-time locomotion generation, in *Computer Animation and Virtual Worlds,* vol.15, pp.125–138, 2004

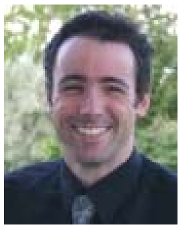
[13] O.Schneider, N.Braun. Content Presentation in Augmented Spaces by the Narration of Interactive Scenes, *Proceedings AVIR*, Geneva, Swiss, pp. 43-44, 2003

[14] Synchronized Multimedia Integration Language (SMIL 2.0), http:// www.w3.org/TR/smil20

[15] The NECA RRL, Information on Neca's Rich Representation Language, http://www.ai.univie.ac.at/NECA/RRL

[16] Virtual Human Project, http://www.virtual-human.org

[17] Weber, Jason. Run-Time Skin Deformation, in *Proceedings of the 2000 Game Developers Conference*



Christian Knöpfle received his diploma from the Fachhochschule in Darmstadt in 1995. From 1996 to April 2000 he worked at Fraunhofer-IGD as a researcher in the virtual reality group. He also has a rich experience as project leader in national as well as international projects. Since April 2000 he is the head of the virtual reality group. Since then he focused more on project acquisition and management. He received his PhD in 2003 from the Technical University Darmstadt.