H-Animator: A Visual Tool for Modeling, Reuse and Sharing of X3D Humanoid Animations

Fabio Buttussi *
HCI Lab
Dept. of Math and Computer Science
University of Udine
via delle Scienze, 206
33100 Udine, Italy

Luca Chittaro†
HCI Lab
Dept. of Math and Computer Science
University of Udine
via delle Scienze, 206
33100 Udine, Italy

Daniele Nadalutti‡
HCI Lab
Dept. of Math and Computer Science
University of Udine
via delle Scienze, 206
33100 Udine, Italy

Abstract

Humanoid animation is a complex task which usually requires particular skills and training. To simplify this process we propose a visual tool, called H-Animator, which aims to help animators (especially the novice ones) in modeling X3D animations for H-Anim humanoids. Besides easiness of use, achieved through intuitive metaphors and interaction styles, we aim at providing an architecture to facilitate the reuse and sharing of X3D content, allowing animators to build a wide archive of material to be reused.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation I.3.6 [Computer Graphics]: Methodology and Techniques—Interaction techniques H.5.1 [Information Interfaces and Presentation]: Multimedia Information Systems—Artificial, augmented, and virtual realities

Keywords: Visual modeling, H-Anim, X3D, animation database, reuse, sharing.

1 Introduction

Modeling humanoid animations is often a time-consuming and difficult task even for skilled animators. To speed-up and make this process easier, several researchers [Arafa and Mamdani 2003; Carretero et al. 2005; Davis et al. 2003; Huang et al. 2003a; Huang et al. 2003b; Kshirsagar et al. 2002; Perlin and Goldberg 1996; Yi et al. 2005] and companies [Alias-Wavefront 2005; Discreet, Autodesk, Inc. 2005; Virtlock Technologies, Inc. 2005] proposed solutions such as scripting animation languages and visual modeling tools for a wide variety of 3D formats (e.g., VRML, MEL, OBJ, 3DS). These visual tools tend to focus on a particular step of the animation process (e.g. humanoid mesh modeling, low-level or high-level animation modeling) or are designed for a specific context (e.g. dialogue animation, sign language animation) or for particular target users (e.g. novice, expert, 2D artist, 3D artist).

The H-Anim standard [Humanoid Animation Working Group 2004], now included in X3D, describes humanoids as an hierarchically organized set of nodes. In particular Joint nodes represent human articulations and Segment nodes represent the geometry and appearance of humanoid portions. Since an H-Anim animation consists of a set of rotations which are applied to Joint nodes, all the animations modeled for a specific humanoid can be reused with any other humanoid having similar anthropometric measures. The standard suggests default names and positions for the joints, but, since the actual positions in real humans vary due to e.g. sex and age, joint positions should be customized and the animations retargeted, if the animator needs to work with anthropometrically different humanoids.

Although one of the aims of the standard is to obtain exchangeable humanoids, there are actually few tools which let users easily reuse the animations they previously built [Huang et al. 2003a; Yi et al. 2005]. Moreover, reuse is usually limited to 3D content built inside a single team of animators, since there are no architectures which allow a widespread organized sharing of X3D H-Anim humanoids and animations.

Therefore, our research focuses on two main goals: (i) simplifying the modeling process, by creating a fast and easy-to-use visual tool which helps the user in each of the modeling phases, and (ii) promoting the sharing of animations, thanks to a public database integrated with the tool. To allow the modeling, reuse and sharing of X3D H-Anim humanoid animations, we propose in this paper H-Animator.

The paper is organized as follows. Section 2 surveys related work. Section 3 describes the H-Animator tool, focusing on the most interesting problems we encountered and the solutions we propose to solve them. In Section 4, we show an application of the tool to sign languages for the deaf. Section 5 concludes the paper and outlines future research directions.

2 Related work

Since humanoid animation is needed in a variety of application contexts, such as ergonomics, entertainment, simulation and sign

*e-mail: buttussi@dimi.uniud.it
†e-mail: chittaro@dimi.uniud.it
‡e-mail: nadalutti@dimi.uniud.it

language visualization, several proposals [Arafa and Mamdani 2003; Kshirsagar et al. 2002; Huang et al. 2003b; Perlin and Goldberg 1996; Carretero et al. 2005; Davis et al. 2003; Huang et al. 2003a; Yi et al. 2005] to simplify animation modeling have been made in the last years. The proposed solutions fall in two main classes: scripting animation languages and visual modeling tools. Other solutions, such as motion texture [Li et al. 2002], motion synthesis from annotations [Arikan et al. 2003] and motion graphs [Kovar et al. 2002], have been investigated in the literature. Anyway, these solutions were designed to work with motion capture data and the acquisition of high quality motion sequences requires expensive hardware, that is usually not an option for novice animators, i.e. the intended users of the tool we propose.

2.1 Scripting animation languages

A scripting animation language is a language to describe animations, specifying what and how a humanoid has to do. Most of these languages (e.g., VHML [Gustavsson et al. 2001], CML [Arafa and Mamdani 2003], AML [Kshirsagar et al. 2002]) focus on high level details such as humanoid behavior (i.e. the user can choose if the humanoid should be happy, angry, anxious, etc., affecting the way in which actions and gestures are performed) or synchronization between different actions (e.g. walk *while* observing the world around, jump *after* running) and between gestures and speech. Only a few scripting languages (e.g. STEP [Huang et al. 2003b], Improv [Perlin and Goldberg 1996]) consider also the low-level modeling of actions and gestures, instead of using a limited set of predefined ones. Interesting case studies (e.g. Greek Sign Language [Karpouzis et al. 2006], Tai Chi animation [Huang et al. 2003b]) prove that expert animators can achieve significant results using scripting languages. Anyway, like all programming languages, scripting animation languages need a considerable learning time before being used effectively, so novice animators may prefer more intuitive solutions.

2.2 Visual modeling tools

To exploit the intuitiveness of visual interaction, several interesting visual modeling tools [Davis et al. 2003; Carretero et al. 2005; Huang et al. 2003a; Yi et al. 2005] have been proposed in the literature. They differ for the phases of the modeling process they focus on, for the techniques used, for the intended users they are thought for. For example, Davis et al. [2003] proposed a tool that is specifically aimed at users who are skilled in 2D animation. Since an animation can be subdivided in a sequence of *postures* (i.e. a set of rotation values for the joints of the humanoid skeleton), the tool exploits users' abilities allowing them to draw 2D sketches of the postures the humanoid must take. For each 2D sketch, the system computes all the possible 3D postures which can be mapped into it and then, considering the entire sequence of sketches, the tool proposes a 3D animation. If the proposed animation is not the desired one, users can edit it, by changing the selected 3D posture for one or more sketches, choosing another mappable 3D posture in the computed set.
While Davis et al. targeted a specific kind of users, the solution proposed in [Carretero et al. 2005] is thought for a particular application context: the high-level tool they describe is specifically designed for dialogs and other animations where speech is predominant. Since it considers high-level details, such as synchronization between speech and gestures and behavioral animation, the authors structured the tool as a front-end for the VHML language [Gustavsson et al. 2001], combining the synchronization features of the scripting language with the usability of a visual tool. The tool lets the user choose one or more humanoids and then type the text they have to read. The user can also select the behavior of the humanoid by changing the color of the text and insert animations by dragging special icons between words. The tool is an interesting solution for dialog animation, but cannot be efficiently used for other kinds of animations, because the predefined animations and the interaction style are very context specific.
A general visual modeling tool, which also introduces the idea of animation "reuse", is proposed in [Huang et al. 2003a]. The proposed system is based on an animation database from which animations and 3D models can be retrieved. All the animations can be visually and interactively adjusted: for example the user can change a female walking animation to a male one, or change the destination the humanoid has to reach.
A database approach is supported also by [Yi et al. 2005]: this recent paper proposes a visual tool to create animations of sign language gestures (e.g. hand, arm, head movements) and signs (i.e. set of gestures corresponding to a word of the language). Since the same gesture can be reused in several signs and new signs can be obtained by composing existing ones, the authors use a database to store and reuse all the material built with the tool. Their final goal is to allow deaf people to "write" in their own language, composing sentences with the sign animations in the database.
Considering commercial solutions, Maya [Alias-Wavefront 2005] and 3DSMax [Discreet, Autodesk, Inc. 2005] are two widely used visual modeling tools. Both of them use proprietary file formats, but some exporters and plug-ins to simplify H-Anim modeling and animation have been proposed. In particular, [Amara et al. 2003] describes a Maya plug-in that allows the user to export H-Anim characters and body animations. To obtain anatomically realistic surfaces during animations, the plug-in generates a body animation table, that contains information to deform humanoid meshes during rotations. A wide set of plug-ins for 3DSMax is described in [Magnenat-Thalmann et al. 2004]. These plug-ins convert 3DSMax skinning, skeleton and motion data into H-Anim humanoids and animations. Another commercial tool which allows the users to create H-Anim humanoids and animations is VizX3D [Virtlock Technologies, Inc. 2005].

3 The H-Animator tool

Since we aim at simplifying the modeling of animations and promote the sharing of X3D content, we designed and implemented H-Animator, a visual tool for modeling, reuse and sharing of X3D humanoid animations. In H-Animator, animations are subdivided in two classes:

- *simple animations* are animations that can be defined by a few keyframes, such as a single action (e.g. pointing, jumping, kicking) or a simple gesture (e.g. moving the hands to emphasize speech or to communicate in a sign language);
- *complex animations* are sequences of simple animations.

Composing complex animations is easier and quicker if the animator can reuse previously stored simple animations [Huang et al. 2003a; Yi et al. 2005]. Extending this idea, we propose a *model and share* functionality: users can store the humanoids and simple animations they modeled on a database and immediately share them with a community of other animators, who can easily reuse the 3D material in their animations.
To guide the user in the modeling, sharing and reusing phases, we provided the H-Animator tool with three main functions:

- *Jointplacer* helps the user in producing an H-Anim humanoid,

using X3D meshes created with other 3D modeling tools; in particular, it allows to easily place the humanoid joints;

- *Keyframer* guides the user in the creation of simple animations, such as a single gesture or a particular action;
- *Composer* allows one to build complex animations using the simple ones created by the user or shared by other animators.

H-Animator can be used locally by single users or teams, but to achieve enhanced sharing functionalities can be connected to an animator community server, as shown in Figure 1. On the server, the most important component is the database which contains humanoids and simple animations; X3D files of humanoid meshes are saved in the network filesystem. Before uploading 3D material on the server, the user can choose the rights she grants on it, e.g., she can ask to be cited when her material is reused and she can allow or deny the alteration of the material.

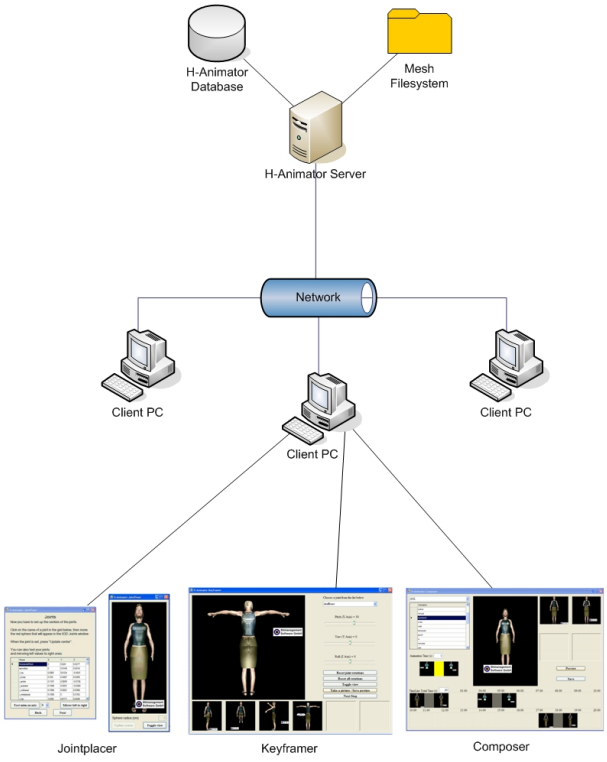

Figure 1: The H-Animator functionalities.

The first three following subsections describe each of the main functions mentioned above; the fourth briefly discusses license aspects for 3D content sharing, focusing on how our tool deals with content license management. The last subsection describes implementation issues.

3.1 Visually placing humanoid joints

The creation of a H-Anim humanoid can be divided in several steps: the meshes have to be manually modeled or acquired through 3D scanners and a level of detail (LOA) for the humanoid has to be chosen. If a segmented humanoid is needed, the meshes have to be subdivided and associated to the segments required for the chosen LOA. The last step before writing the X3D file is choosing the center position of each skeleton joint.
Mesh modeling and joint placement are two important and difficult steps. The first task can be efficiently carried out using a generic 3D modeling application, provided that it exports to X3D. For example, the meshes of the humanoid in the screenshots of this paper, were modeled with open-source software, i.e. Blender [Blender Foundation 2005] and the Makehuman plug-in [MHTeam 2005] for modeling humanoids from sizing parameters, a technique proposed in the literature by [Seo and Magnenat-Thalmann 2003].

Jointplacer allows one to load exported X3D meshes and focuses on joint placement. This is a crucial step: since joints are the centers of rotations, a wrong placement leads to unrealistic animations that can even cause segments to separate from the body. Figure 2 provides an example: starting from the same resting position, the right shoulder of the 3D humanoid is unrealisticly animated if the joint is unproperly placed, while it is animated as expected when the joint is in the correct position.

Figure 2: Two animations of the same humanoid with different shoulder center positions.

Jointplacer follows a wizard interaction style, guiding the user in the creation of the H-Anim humanoid, starting from existing X3D meshes. After choosing the desired level of articulation and associating the meshes with the corresponding H-Anim segments, an interface for the joint placement is loaded. This interface consists of two windows (Figure 3): the left one contains a table listing all joints with the default H-Anim names and positions; the right window embeds an X3D player, where the humanoid with the desired segments and default joints is shown.

When the user selects a joint in the table, a red sphere is placed in the embedded X3D player at the position that corresponds to the current value for the selected joint. The user can simply drag the sphere to place it in the desired position.
Unlike commercial applications, such as VizX3D [Virtlock Technologies, Inc. 2005], we decided to show only the sphere corresponding to the selected joint and we allow the user to change the size of the sphere itself. This prevents problems such as the impossibility to visually place the joint, due to the size of the markers, which cover entire segments of the humanoid.

Even with an easy-to-use interface, the placement of joints is not an intuitive task: dragging the markers is easier than specifying the joint center coordinates (at least for novice animators), but finding

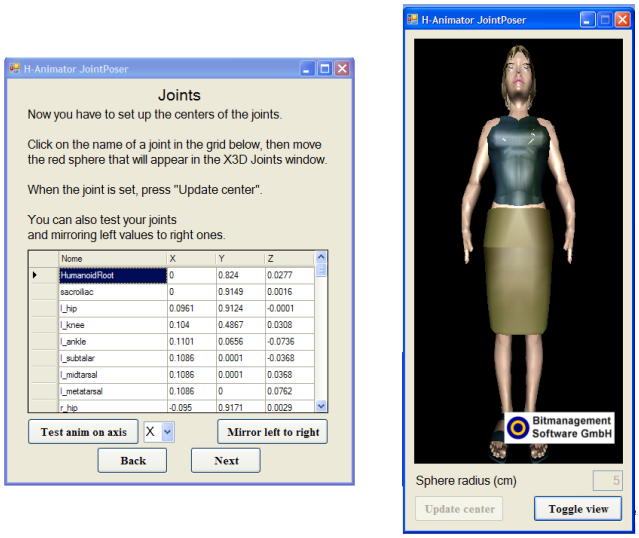

Figure 3: Jointplacer interface.

out a precise point position in a 3D space, trying to guess whether it will lead to realistic animations or not, is usually very difficult, even for expert animators. The user often learns that a placement is not as good as she thought only when she uses the humanoid in an animation, but correcting a joint position in the animation phase is very time-consuming: for example, using VizX3D, the animator learns about the wrong placement when she is in the preview window or when she is defining the animation postures; in each case she has to stop and go back to the main modeling interface; there she can correct the joint position and then she can return to the interface for modeling the animation or start the preview again. Since the user may not find the correct placement at the first try, she may have to switch between the interfaces more than once.
For these reasons, we provided our tool with an instant testing feature: whenever she wants, the user can start one of three predefined animations, consisting in the continuous rotation of the selected joint children around one of the three axis. These animations can be played also during joint placement: as the user presses the "Update" button, the joint center is updated with the current position of the sphere in the 3D scene, then the animation continues with the new center. This feedback is very important because it lets the user find the right placement by trying some positions and immediately evaluating the resulting animation.

After the joint placement, the user can store the humanoid in the local or community database to reuse it in her animations. Moreover, according to the rights she grants, other animators may use the humanoid in their applications.

3.2 Modeling simple animations

Animation techniques proposed in the literature to determine the postures can be divided in at least three categories:

- *direct kinematics* techniques ask the user to input the rotation values that have to be applied to the humanoid joints for each of the key-postures (i.e. the main postures the humanoid has to take during the animation);
- *inverse kinematics* techniques simplify the animation task requiring only the position of the end effectors (e.g. hands or feet) for each key-posture and automatically deriving rotation values;
- *motion capture* techniques acquire the data from a human actor using motion sensors or cameras.

Motion capture techniques are faster, but sometimes lack in precision, since some postures which are significant for the animation may not be acquired due to the sampling. On the contrary, both kinematics classes of techniques guarantee precision, since the animator inputs the data for all the important postures, but the determination of the values is a time-consuming task.

Considering the treatment of temporal constraints, animation techniques can be divided in:

- *per-key* techniques, if the rotation values and the time at which each posture must be taken are determined only for particular important postures;
- *per-frame* techniques, if the rotations are determined at constant intervals of time;
- *spacetime* techniques, if the timing information is calculated through a mathematical function that takes into account all postures simultaneously (see also [Gleicher 2001]).

Keyframer, H-Animator function for visual modeling simple animations, uses direct kinematics with a per-key approach, so the user has to choose the rotation values for each key-posture of the animation and the time at which each posture is taken. Since this may be a difficult task, our solution provides a set of specific interfaces to make animation modeling easier and quicker. Keyframe animation modeling can be divided into two sub-tasks [Terra and Metoyer 2004]: the specification of keyframe values (i.e. the rotations of the joints) and the keyframe timing (i.e. the specification of the time at which each posture is taken); Terra and Metoyer [2004] proved, studying novice users, that keyframe timing is more difficult than the specification of keyframe values and that a clear separation of these two sub-tasks leads to an easier modeling process. Therefore our tool presents two different interfaces for these sub-tasks.

To be intuitive for the user, we have based modeling on a photographer's metaphor: in the posing interface of Keyframer (Figure 4) the user poses the humanoid and then "takes pictures" of it. A data structure containing the rotation values is automatically associated to each picture, so, whenever the user clicks on a previous picture, the corresponding rotations can be loaded and applied to the humanoid joints in the X3D player embedded in the window.

In X3D and H-Anim, the rotation values have to be specified in the axis-angle notation using radians. This notation is usually unfamiliar to people without specific mathematics background, so we chose to let the user input the angle values in the more common Eulerian notation using degrees. The tool performs the trivial conversion between degrees and radians, but also the conversion between the axis-angle and the Eulerian notation, using quaternions and transformation matrixes. Even using degrees and the Eulerian notation, specifying the rotation values is not an intuitive task, so we let the user play with the values using the three sliders, labeled "Pitch", "Yaw" and "Roll", placed on the right side of the interface. As the user interacts with these sliders, the new rotation values are constantly applied to the chosen joint of the humanoid in the embedded X3D player on the left, giving a live feedback about the posture it will take. So the user can "pose" the humanoid and then, when the desired posture is obtained, she can "take a picture".

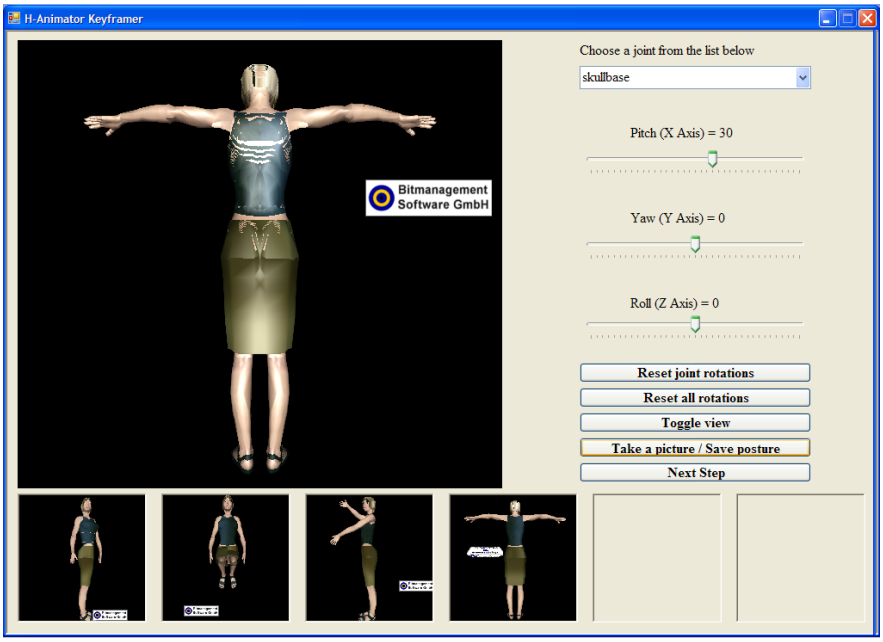

Figure 4: Keyframer posing interface.

After collecting the desired pictures, the user can switch to the timing interface (Figure 5). This time, the user has to choose a sequence of pictures she took and place them on a timeline.

This operation is extremely easy: she first clicks on a picture and the corresponding posture is immediately taken by the humanoid in the embedded X3D player; then she clicks the instant on the timeline where that posture has to be taken. This is also similar to the ordering process a 2D cartoon animator has to go through, facilitating the migration to 3D for this kind of users.
After timing the simple animation, the user can store it, so that it can be reused in complex animations.

To test ease of use of Keyframer, we carried out a preliminary informal evaluation with a few novice users. We gave them five photographs of an actor assuming different postures and we asked them to model the postures using the Keyframer posing interface. The mean time to model a posture was 3 minutes and 33 seconds.

Figure 5: Keyframer timing interface.

After they modeled all the five postures we introduced them to the Keyframer timing interface and we asked them to use the pictures of the humanoid postures to make a simple animation. The mean time to make the animation was 1 minute and 5 seconds. These results are encouraging, since the users had no previous experience in humanoid animation and it was the first time they used Keyframer.

3.3 Building complex animations

All the H-Anim humanoids created with Jointplacer and the animations modeled with Keyframer are automatically stored in the H-Animator database. Composer reuses this 3D content to quickly build complex animations. Figure 6 shows Composer interface: on the upper-left corner there is a combo-box to select the animation category (e.g. teacher's gestures, fitness exercises, American sign language); this is used to filter the list of available animations right below. When an animation is selected, an ad hoc user control, called animation control, is updated. This control changes its length according to the duration of the animation. Besides, the two pictures displayed on the animation control show the first and the last frame of the animation selected. After clicking the animation control, the user can place an image representing the simple animation in the timelines at the bottom, so she can easily compose a complex animation sequence with the simple animations she desires. At any moment, the user can see a preview of the animation in the X3D player placed at the center of the interface.

When the user needs a preview or wants to save the final animation, the chosen simple animations are retrieved from the database and the transitions between them are automatically generated by the tool. Transition generation is a problem for which the literature presents several solutions: a simple solution is to start and end each simple animation in the same neutral posture, so that no transition is needed. Anyway this solution may be good only if the humanoid can wait for some time in a given neutral posture, between each pair of subsequent simple animations. Otherwise, forcing a neutral posture may lead to unrealistic animations: for example, in a sign

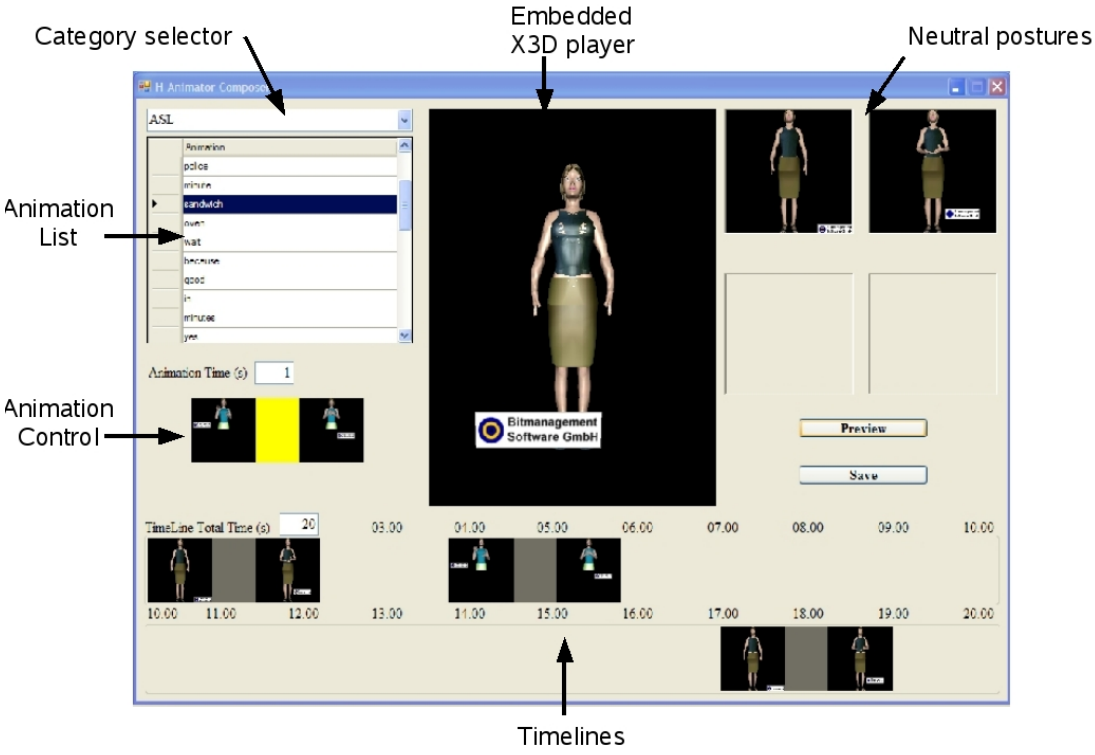

Figure 6: Composer interface.

language application there are no pauses between signs in the same sentence, so this solution must be avoided (see also [Zhao et al. 2000]). Interesting solutions, such as using finite state automata [van Zijl and Raitt 2004] or Parallel Transition Networks [Badler et al. 1999], were proposed for sign language applications or in the implementation of scripting languages, but we decided to choose a simpler semi-automatic linear interpolation technique: Composer retrieves the simple animations from the database and then builds a linear transition between the last posture of an animation and the first posture of the next one. The resulting transition is good when the postures can be linearly interpolated without collisions and compenetrations, otherwise it may be unrealistic: for example, if we consider an animation which ends with the hands behind the body and we want to concatenate it with an animation where the hands start in front of the body, the resulting linear transition passes through the body. When such unrealistic transitions occur, the user can choose an intermediate posture that has to be placed between the subsequent simple animations. Finding and correcting bad transitions is easy, since the user can test the animation in the embedded X3D player and she can also check the initial and final postures looking at the pictures on the animation controls.

3.4 Sharing the 3D content

Software on the web may be published under different licenses (demo, freeware, shareware, open-source), but such licenses cannot be applied to 3D material, since they are specifically written for software: for example, terms like "code" are not appropriate to describe animations and humanoids, because an animator may even not know that there is a code under the 3D content. To overcome this limitation, Creative Commons [2005] has recently adapted open-source licenses to artistic material, proposing a family of licenses, which differ for the rights the author grants.
We included the Creative Commons license system into the H-Animator tool. In H-Animator, after modeling a humanoid or a simple animation, the user has to set three attributes, which are used to generate the Creative Commons license corresponding to the rights the user grants. The attributes describe if it is required to cite the author, if the reuse is restricted to non commercial applications only and if the alteration of the content is denied. When a user wants to reuse a humanoid or a simple animation in a complex animation, similar attributes ask her if she agrees to cite the original author, to use the content only for non commercial applications, to preserve the original content. If she disagrees with some requirements, the list of reusable 3D content in appropriately filtered.

3.5 Implementation issues

H-Animator is written in C# using .Net Framework 2. To show animations and previews of X3D content, we use BS Contact ActiveX [Bitmanagement Software GmbH 2005] as an X3D embedded player. BS Contact was chosen because it supports all the X3D nodes we need to use and offers an interface which follows the SAI (Scene Access Interface) specifications of X3D, adding features such as a bitmap rendering function, which was used for making the pictures of the humanoid postures in Keyframer.
Using BS Contact, the interactions between the tool and the X3D scene can be implemented in three different ways: nodes and routings can be added directly to an existing scene, VRML code can be written and inserted in the 3D scene (addition of X3D code was not supported by BS Contact when we began to implement

our application), the loaded X3D world can be replaced by reading a new X3D file. The three methods present advantages and disadvantages: the direct way is the faster one if the developer needs to add only a few nodes or if she wants to change only some field values, since it does not require to read files or write strings of code; anyway, when a complex structure of nodes and routings has to be added, inserting external VRML code is faster than going through a long sequence of direct calls. Finally, the replacement of the 3D world should be used only when the existing scene has to be removed or when immediate response is not needed, since the replacement method requires longer loading times.
For example, the red sphere used in Jointplacer is inserted in a *Transform* node which also contains the *PlaneSensor* and the routings to move it, so we wrote a short VRML file containing these nodes that is added to the scene when the user selects a new joint or updates the joint center value. Otherwise, the calculation of the new joint position is performed by the C# application, reading the *position* field of the *Transform* node that contains the sphere directly through the SAI.
In Keyframer, all the X3D code for the previews is generated by the tool using the common *TimeSensor* and *OrientationInterpolator* nodes and it is then loaded inside the X3D player, since real-time response is not needed for this task. However, to give instant feedback when the user moves the sliders, we change the *rotation* field of the joint involved directly through the SAI.
In Composer, the most interesting programming issues are related to the transition animations: the tool reads from the database the final and initial orientations for the previous and the next animation respectively and then an *OrientationInterpolator* node is created for each joint whose values differ between an animation and the other. The concatenation is then achieved using *BooleanFilter* and *TimeTrigger* nodes, which activates an animation as soon as the previous one stops. The concatenation process takes about one second on recent PCs (depending on the number of animations to concatenate and the joints involved), so we load the new X3D scene replacing the older one, since we do not need immediate response.

4 Sign language application

An interesting application domain for H-Anim humanoids is given by sign languages, i.e. the visual communication languages used by deaf people. For example, [Sagawa and Takeuchi 2002] and [Karpouzis et al. 2006] describe architectures and tools that use H-Anim humanoids to teach sign languages, while [Zhao et al. 2000] presents an automatic machine translation system from written English to American Sign Language. As described in Section 2, [Yi et al. 2005] proposed a tool to create and reuse animations for sign languages, using a specifically designed database.
The deaf communities of different geographical areas use sign languages which differ in grammar and vocabulary. Each of these languages uses thousands of signs, so a single team of animators cannot produce all the animations even with a fast and easy-to-use tool. Therefore, the sharing capability becomes crucial: using Keyframer, deaf communities could populate their own vocabulary and easily share work among members. Besides, we provided our tool with a sign language version of the Composer function (Figure 7), which allows the user to compose animations of sign language sentences. The user can simply write sentences following the grammar of her sign language, using the animations of the words produced by her deaf community; then she can view the sentence gestures in the X3D player. She can change or correct the sentence and then she can save a X3D file, ready to be sent or published. Communicating with sentences in their own sign language is very important for deaf people, since writing in a language spoken by hearing people is like using a foreign language for them.

Figure 7: Sign language version of Composer interface.

The fast retrieval of the animations (about one second per sign language sentence, varying with the complexity of the sentence and the number of joints involved), combined with the automatic composition system and the sharing architecture, will allow to create new or simplify existing sign language applications. Examples are sign language chats, dictionaries, translation systems, humanoid signing of TV programs and teaching tools.
Moreover, sign language animations which follow the X3D H-Anim specifications can be displayed on PocketPC mobile devices using MobiX3D Player [Nadalutti et al. 2006], a mobile player that can display X3D scenes containing H-Anim humanoids. Using H-Animator together with MobiX3D Player will allow for innovative mobile sign language applications.

5 Conclusions and future work

In this paper, we proposed a visual modeling tool, called H-Animator. This tool aims to help both expert and novice animators in the modeling of H-Anim humanoid animations for X3D. We have based simple animation modeling on a familiar metaphor and we provide animation previews and immediate feedback about postures, while the possibility to reuse and share the 3D content is a result of the architecture based on a database. This last feature allows one to build humanoid and animation archives, simplifying the production and sharing of 3D content and applications, since an animator can search if the 3D material she needs is already on the database, instead of remodeling it. This is particularly important if huge databases are needed, as in the sign language application introduced in Section 4.
Obviously, H-Animator is not an alternative to commercial tools and plug-ins that allow for greater flexibility. Anyway,

these tools are not freely available and thought for expert animators, while H-Animator is publicly available (http://hcilab.uniud.it/h-animator/) and the results of the preliminary evaluation are encouraging with respect to its usability for novice animators.
We are currently working on a new release of H-Animator which will integrate new features. In particular, to simplify the posing task we are working to introduce inverse kinematics combined with a visual end effector manipulation system. Another important feature will be a retargeting interface to adapt the animations for anthropometrically different humanoids.

6 Acknowledgements

Our research has been partially supported by the Italian Ministry of Education, University and Research (MIUR) under the PRIN 2005 project "Adaptive, Context-aware, Multimedia Guides on Mobile Devices".

References

ALIAS-WAVEFRONT, 2005. Maya. http://www.aliaswavefront.com/en/products/maya/.

AMARA, Y., GUTIÉRREZ, M., VEXO, F., AND THALMANN, D. 2003. A maya exporting plug-in for mpeg-4 fba human characters. In *RICHMEDIA '03: Proceedings of the First International Workshop on Interactive Rich Media Content Production: Architectures, Technologies, Applications and Tools*, 121–130.

ARAFA, Y., AND MAMDANI, A. 2003. Scripting embodied agents behaviour with cml: character markup language. In *IUI '03: Proceedings of the 8th international conference on Intelligent user interfaces*, ACM Press, New York, NY, USA, 313–316.

ARIKAN, O., FORSYTH, D. A., AND O'BRIEN, J. F. 2003. Motion synthesis from annotations. *ACM Trans. Graph. 22*, 3, 402–408.

BADLER, N. I., PALMER, M. S., AND BINDIGANAVALE, R. 1999. Animation control for real-time virtual humans. *Commun. ACM 42*, 8, 64–73.

BITMANAGEMENT SOFTWARE GMBH, 2005. Bs contact vrml/x3d. http://www.bitmanagement.de.

BLENDER FOUNDATION, 2005. Blender. http:www.blender.org.

CARRETERO, M. P., OYARZUN, D., ORTIZ, A., AIZPURUA, I., AND POSADA, J. 2005. Virtual characters facial and body animation through the edition and interpretation of mark-up languages. *Computers and Graphics 29*, 189–194.

CREATIVE COMMONS, 2005. Creative commons licences. http://creativecommons.org.

DAVIS, J., AGRAWALA, M., CHUANG, E., POPOVIĆ;, Z., AND SALESIN, D. 2003. A sketching interface for articulated figure animation. In *SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, Eurographics Association, Aire-la-Ville, Switzerland, 320–328.

DISCREET, AUTODESK, INC., 2005. 3ds max. http://www.discreet.com/products/3dsmax/.

GLEICHER, M. 2001. Comparing constraint-based motion editing methods. *Graphical Models 63*, 107–134.

GUSTAVSSON, C., BEARD, S., STRINDLUND, L., HUYNH, Q., WIKNERTZ, E., MARRIOT, A., AND STALLO, J., 2001. Vhml working draft v0.3. http://www.vhml.org/documents/VHML/.

HUANG, A., HUANG, Z., PRABHAKARAN, B., AND C. R. RUIZ, J. 2003. Interactive visual method for motion and model reuse. In *GRAPHITE '03: Proceedings of the 1st international conference on Computer graphics and interactive techniques in Australasia and South East Asia*, ACM Press, New York, NY, USA, 29–36.

HUANG, Z., ELIËNS, A., AND VISSER, C. 2003. Implementation of a scripting language for vrml/x3d-based embodied agents. In *Web3D '03: Proceeding of the eighth international conference on 3D Web technology*, ACM Press, New York, NY, USA, 91–100.

HUMANOID ANIMATION WORKING GROUP, 2004. H-anim. http://h-anim.org.

KARPOUZIS, K., CARIDAKIS, G., FOTINEA, S., AND EFTHIMIOU, E. 2006. Educational resources and implementation of a greek sign language synthesis architecture. *Computer and Education, Special Issue in Web3D Technologies in Learning, Education and Training, In press*.

KOVAR, L., GLEICHER, M., AND PIGHIN, F. 2002. Motion graphs. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, 473–482.

KSHIRSAGAR, S., MAGNENAT-THALMANN, N., GUYE-VUILLÈME, A., THALMANN, D., KAMYAB, K., AND MAMDANI, E. 2002. Avatar markup language. In *EGVE '02: Proceedings of the workshop on Virtual environments 2002*, Eurographics Association, Aire-la-Ville, Switzerland, 169–177.

LI, Y., WANG, T., AND SHUM, H.-Y. 2002. Motion texture: a two-level statistical model for character motion synthesis. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, 465–472.

MAGNENAT-THALMANN, N., CORDIER, F., SEO, H., AND PAPAGIANAKIS, G. 2004. Modeling of bodies and clothes for virtual environments. In *CW '04: Proceedings of the 2004 International Conference on Cyberworlds (CW'04)*, IEEE Computer Society Press, Los Alamitos, CA, USA, 201–208.

MHTEAM, 2005. Makehuman. http://www.dedalo-3d.com.

NADALUTTI, D., CHITTARO, L., AND BUTTUSSI, F. 2006. Rendering of x3d content on mobile devices with opengl es. In *Web3D '06: Proceedings of the eleventh international conference on 3D Web technology*, ACM Press, New York, NY, USA.

PERLIN, K., AND GOLDBERG, A. 1996. Improv: a system for scripting interactive actors in virtual worlds. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, 205–216.

SAGAWA, H., AND TAKEUCHI, M. 2002. A teaching system of japanese sign language using sign language recognition and generation. In *MULTIMEDIA '02: Proceedings of the tenth ACM international conference on Multimedia*, ACM Press, New York, NY, USA, 137–145.

SEO, H., AND MAGNENAT-THALMANN, N. 2003. An automatic modeling of human bodies from sizing parameters. In *SI3D '03: Proceedings of the 2003 symposium on Interactive 3D graphics*, ACM Press, New York, NY, USA, 19–26.

TERRA, S. C. L., AND METOYER, R. A. 2004. Performance timing for keyframe animation. In *SCA '04: Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, ACM Press, New York, NY, USA, 253–258.

VAN ZIJL, L., AND RAITT, L. 2004. Implementation experience with collision avoidance in signing avatars. In *AFRIGRAPH '04: Proceedings of the 3rd international conference on Computer graphics, virtual reality, visualisation and interaction in Africa*, ACM Press, New York, NY, USA, 55–59.

VIRTLOCK TECHNOLOGIES, INC., 2005. Vizx3d. http://www.vizx3d.com.

YI, B., FREDERICK C. HARRIS, J., AND DASCALU, S. M. 2005. From creating virtual gestures to "writing" in sign languages. In *CHI '05: CHI '05 extended abstracts on Human factors in computing systems*, ACM Press, New York, NY, USA, 1885–1888.

ZHAO, L., KIPPER, K., SCHULER, W., VOGLER, C., BADLER, N. I., AND PALMER, M. 2000. A machine translation system from english to american sign language. In *AMTA '00: Proceedings of the 4th Conference of the Association for Machine Translation in the Americas on Envisioning Machine Translation in the Information Future*, Springer-Verlag, London, UK, 54–67.