3D Character Modeling in Virtual Reality

Szilárd Kiss [S.Kiss@cs.utwente.nl]
Department of Computer Science - Language, Knowledge and Interaction group,
University of Twente, Postbus 217, 7500 AE Enschede, The Netherlands

Abstract

The paper presents a virtual reality modeling system based on interactive web technologies. The system's goal is to provide a user-friendly virtual environment for the development of 3D characters with an articulated structure. The interface allows the modeling of both the character's joint structure (the hierarchy) and its segment geometry (the skin). The novelty of the system consists of 1) the combination of web technologies used (VRML, Java and EAI) which provides the possibility of online modeling, 2) rules and constraints based operations and thus interface elements, 3) vertices and sets of vertices used as graphics primitives and 4) the possibility to handle and extend hierarchies based on the H-anim structure elements.

Keywords: VR, VRML, H-anim, modeling, WWW.

1 Introduction

The purpose of our system is to transform the geometry and hierarchy of articulated characters into visible, visually editable entities. For this purpose we use the VRML[13] language and on the H-anim[7] specification. The H-anim structure elements "joint" (for the hierarchy) and "segment" (for the geometry) are used in the system as basic components of the articulated characters. The segment geometries are based on a regular grid of vertices (reasons and advantages of this see sections 3.3 and 3.4).

The system is aimed to be simple to use, such that users would be able to learn to use it quickly. The complexity of the geometry and hierarchy is user-handled. Since creating 3D characters is a specialized modeling task and our aim is to provide operations on a low level of detail that is still suited to the casual user, an expressive and minimal user interface was built on top of the modeling operations.

The system consists of a VRML virtual environment, a Java applet and the EAI[3] component put together and used as an interactive system that is able to dynamically create and update the edited character's different visualizations (hierarchy, geometry). Since it is an applet that requires inter-plugin communication, it can be used online and offline within a web browser.

2 Background

Complex modeling systems like Maya (PLE), GMax (simpler version of Studio Max), Blender, CosmoWorlds (VRML modeler), etc. tend to be very general and thus present an overwhelming interface to the novice or casual user. They certainly exceed in possibilities our system, which aims among other to eliminate the steep learning curve usually associated with state of the art modeling tools. Our system specializes in building articulated hierarchies and the related geometries (for our virtual environments [11]), and the constraints of this task allow the use of a simplified interface.

Our system breaks the interface - environment duality: since we model a character at a time without modeling its surrounding environment, the entire interface, with the exception of a few file-oriented buttons, is situated inside the virtual environment. This allows camera freedom (the user can move around freely) which is enhanced by positioning widgets (researched by [2], [4], [5] among others) and different data visualizations, making multiple views unnecessary (multiple views are not very appropriate for current on-line technologies). For the data visualizations, we separate the hierarchy from geometry, and we use a semantically linked and rule-based hierarchy and modeling approach.

We try to achieve selection indirection. Our system, in contrary with GMax, Milkshape3D for instance, provides direct selection functionality: the type of selection is handled automatically by using non-overlapping selection sensors instead of explicit selection type buttons.

Specialized avatar creation tools (for on-line environments) like Avatar Studio and Avatar Lab are based on prebuilt body components which can be parametrically altered for longer/shorter, thinner/thicker, etc. results. Geometry complexity is fixed, as is the number of segments. They do not provide flexibility in hierarchy construction and geometry complexity.

Part of the novelty of the system is its complete imple-

Proceedings of the Sixth International Conference on Information Visualisation (IV'02)
1093-9547/02 $17.00 © 2002 **IEEE**

mentation within the VRML and Java (extended with EAI) web-based technologies. This allows modelling to be performed online, and the whole system could be transformed into a collaborative editing environment if it is connected to a central database.

At last but not least, modeling systems have high computing requirements that are usually accessible only for professional users. Our system is more modest, its little over 200KB code runs on much more modest requirements.

3 Framework

The system consists of two interacting components embedded in a html hypertext file. The 3D environment (VRML world) holds the static environment, the widgets (buttons and other controls) and the containers for the dynamically created VRML content that visualize the different aspects of the edited character: hierarchy, whole character or selected segment, all equipped with sensors for selection handling.

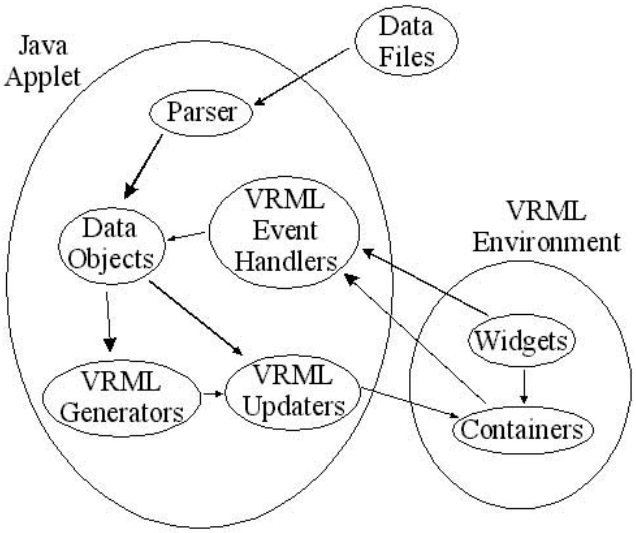

Figure 1. System components

Next to the VRML component there is a Java applet which stores all the data that is assigned to the modeled character, and specifies the methods to create and update the dynamic content of the VRML component using the EAI communication channels. A diagram of the connections between the elements of the two components is shown in figure 1.

3.1 VRML component

Besides the static environment, there are three VRML containers that hold the dynamic scene components. Their functionalities are:

- "Character Container" - a container for the whole character which holds all the segment geometries of the current character. The segment geometries are selectable.
- "Hierarchy Container" - a container for the hierarchy which holds a visual representation of the joint positions and connections of the current character. Each joint and terminal point is selectable.
- "Segment Container" - a container for a single segment which represents the geometry of the currently selected segment. All primitives are selectable.

The content placed in these containers is created and updated by the Java component (see figure 1), which listens also to the events produced by the control structures from the container. These updates are initiated with the widgets/buttons of the VRML component. In addition, widgets from the VRML component can also change properties of the containers directly (the position and orientation, used for preview and interface managing purposes).

The system is functionally divided in two editors. The first one is the hierarchy editor (described in more detail in section 3.7) which represents the position of joints in the character and the connection between the joints (the joint hierarchy) with selection sensor boxes respectively joint relation lines. This editor permits the creation and modification of the underlying hierarchy of the character and influences the content of the hierarchy container and the content of the whole character container.

The second editor is a geometry editor which is used to visualize and edit the surface of single segment geometries by representing the surface coordinates with sensor boxes and coordinate rings and columns with sensor linesets. This editor influences the container of the selected segment and the whole character container, thus every change in a single segment of a character is repeated on the whole character visualization for data consistency. This editor is described in more detail in section 4.1.2.

3.2 Java applet component

To enable a high level of interactivity, the system is capable of reading the data in and consequently modifying the read in data. This needs behavior specification (links between the Java and VRML components) that are dynamically created, changed and updated during the modeling process. One important task is managing the communication between the dynamic VRML content and the Java component. This communication is also dynamically defined, since it must link to dynamically (run-time) created VRML entities (nodes or events). This is achieved using node and event retrieval based on indexes calculated from the data quantities stored in the Java component.

3.3 Graphics Primitives

The graphics primitives used in modeling packages are usually coordinates, edges and polygons, they provide the base for such operations as beveling, extruding, etc., applied to arbitrary models [10]. Our system is more restrictive in the model range. We require that the models must be regular. This is required for automatic surface handling and for reduced data size, since indexes are not stored, but automatically generated. To maintain the regular grid constraint, ring (similar to the contours used by [1]) and column primitives were introduced (collections of coordinates). Enabling coordinate complexity changes (addition and removal) only for ring and column primitives, the regularity criteria is maintained. Position changes are enabled for all primitives: single coordinates, rings and columns as well. The primitive sensors (boxes in case of vertices and line sets in case of rings and columns) are placed a bit outwards the actual geometry coordinates to ensure their visibility.

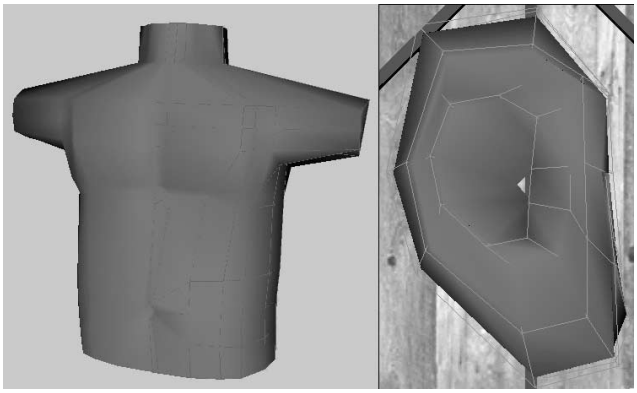

Figure 2. Example geometries

To show that the primitives used are enough for the purpose of character modeling, the following geometry examples were modeled with our system (see figure 2): a human torso and an ear. Further example geometries can be found in [8] and [9].

3.4 VRML constraints

The system is built upon the open VRML format. This enables 3D interactivity and web based interaction. VRML, although the ”Language” word is part of its name, is more a file definition format than a programming language. This file format uses the graphics primitives coordinate, (multi)line and polygon, the latter ones building on coordinate arrays and indexes. This also influenced the ring and column primitives used in the system, since these are collections of coordinates that can be applied directly in geometry composition.

VRML is interpreted, and its interpreters are freely available browser plugins. Standalone viewers are also available, but those do not always support the Java platform for programming the interaction and most certainly do not support the EAI method for programming, since that is based upon inter-plugin communication. In this context, the hardware considerations are addressed within the browser companies, and our system relies on them for on-screen display.

3.5 Direct manipulation

Most modern graphics packages use direct manipulation methods to eliminate indirection: the target and the operation are situated in the same spot. This is fine in a modeling package where there are a few two-dimensional views (GMax, Maya PLE, Blender, MilkShape3D, etc.), where the mouse's two dimensional handling possibilities can be directly mapped. However, our manipulation approach is slightly different since we use solely a three dimensional view. We apply direct manipulation in the case of selections and positioning/viewing widgets, which are in turn indirected in the mentioned modeling packages. However, we do not exclude the possibility of direct editing, and we are researching the possibilities of direct editing widgets.

3.6 H-anim(+)

The H-anim specification provides a standard way to describe human-type characters (avatars). It specifies a number of 93 (in version 1.1) joints with their corresponding segments. The joints and segments have standardized names and there is measured data available about joint positions and flexibility. For different purposes, anchor coordinates can be defined for segments (e.g. for glasses, tie, watch, etc.) in the form of sites. Displacers can be used to deform the segments, when additional animation is required besides the joint rotation method.

There are variations of hierarchy complexity that that are suggested by the H-anim standard for H-anim type characters. They are called Level Of Articulation (LOA), and there are a number of variations of them. There are joints that are required by the specification for a minimal H-anim conformance, but other than that different variations can be built.

In our system there is no restriction on the names, positions and hierarchy that could be built, thus there is no H-anim validity checking. However, if one uses an existing H-anim model just to adjust the 3D character's geometry or to extend it (if not already a full H-anim hierarchy) by naming the new joints and segments in a H-anim compliant way, then the result will be a H-anim compliant character. This approach gives the possibility to extend the H-anim character space to non-human characters with more/less than two

legs, two arms, one head and to add different extensions: tails, fins, wings, etc.

In our system, we only use the rules between the H-anim components defined by the specification, and disregard the rules of defined joint names and connections. We do this for the obvious reason that we do not want to restrict the user to human models. Users might want to build avatars in the form of animals and other entities (TelePresence's multi-user software had an avatar in form of a wooden board).

The hierarchy rules are:

a) A joint can have multiple joint children

b) Every joint has one and only one segment child

c) All (except the root) joints have one and only one parent joint.

3.7 Hierarchy

The user has the possibility to use and modify predefined H-anim and other hierarchy variants or to create new types of hierarchies by adding or removing joints. A few example hierarchies are presented in figure 3.

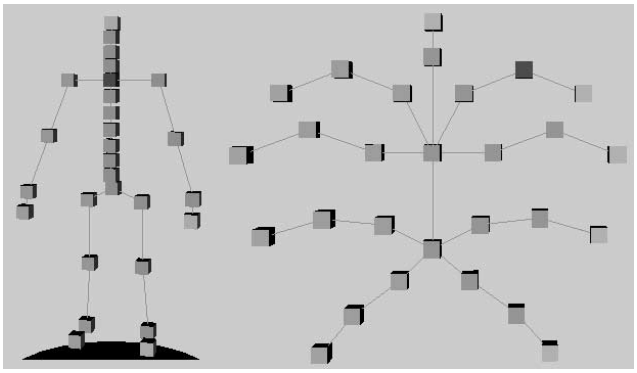

Figure 3. Example hierarchies: human and spider

The Hierarchy operations mentioned (visible in figure 5) bear the marks "Joint" and none of them uses "Segment". This is a consequence of the rules stated in 3.6. Since each joint has one and only one segment child, they do not influence at all the hierarchy if their parent joint is already a part of the hierarchy. Thus they are handled automatically in the background.

Hierarchy changes are reflected on the segment geometries. Hierarchy expansion creates new segments and their geometries, hierarchy refinement divides segment geometries and finally collapsing hierarchy joints means concatenation of geometries. Modifying the joint positions results in the automatic modification of the geometry in cases when the parent-child relationships make it possible. The exception appears at multiple and non-symmetric children segments, when there is no clear way of choosing appropriate geometry changes for all child relations without the danger of interfering with the user's preferences. When the relationships allow the automatic geometry modifications, the geometry of the selected joint (and eventually it's symmetric) and the geometry of the joint's parent joint deform to follow the new joint position. This is very advantageous in case of human ankles, knees, elbows or when the torso is segmented, by automatically keeping the relation between the affected geometry segments. This way, different poses can be produced quickly.

3.8 Geometry

Each hierarchy segment defines a piece of geometry, and the character mesh is the sum of these segment geometries. Currently in our system the coordinates are stored separately in each segment geometry, but they can be stored also in an unified "Coordinate" VRML node (the H-anim specification allows both methods, but recommends the later one). The geometries of the hierarchies of figure 3 are presented in figure 4.

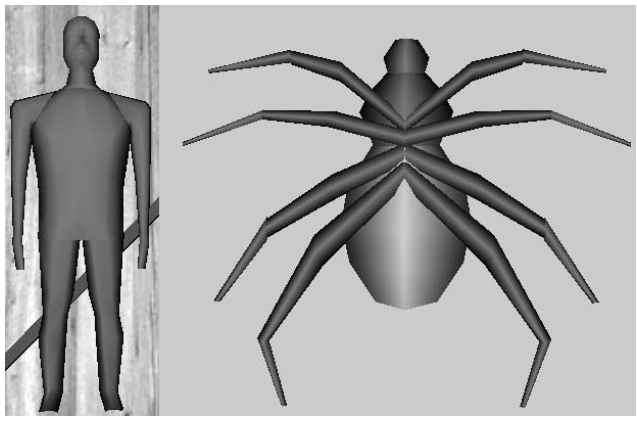

Figure 4. Example characters: human and spider

4 User Interface

The interface consists of two components: a Java part (only a few Buttons) and the VRML interface (see figure 5). The Java Buttons are used simply to trigger file opening and saving operations, while the VRML interface is used for all the editing operations. It is our intention to integrate the Java Buttons functionality into the VRML environment next to the editing functionalities for adding consistence to the system, eliminating a double interface. Since the goal is to create a single character at a time without modeling the target environment, it is possible to integrate the edited character and all the widgets directly into a VRML world where a real-time rendered preview can be obtained about the character at all times.

To aid simplicity, only relevant widgets/buttons are visible in the interface, and they are visible all at once (even those deactivated due to different selection contexts). This approach helps by revealing the available modeling options, as well as the selection consequences.

4.1 VRML interface

The VRML environment contains the main editing interface. In this environment there are three different visualizations of the edited character: a full character visualizations (the collection of segments, noted with 1 in figure 5) which serves for preview and for segment selections, a hierarchy visualizations (the joint structure, noted with 2) that serves for editing the hierarchy and a segment geometry visualizations (a single segment geometry, noted with 3) that serves for editing the geometry of the selected segment. These three visualizations are displayed simultaneous in the environment when a single segment geometry is edited (with the hierarchy visualization set aside, but still viewable for reference), but when the hierarchy is edited, the segment geometry visualization is discarded, since the geometry changes introduced by the hierarchy editing process can be viewed on the whole character as well. The segment and joint selections are interlinked (equivalent) due to the 1:1 relationship (see section 3.6) and the editing mode determines the active selection. This type of visualization is used as a compromise between displaying relevant information and overwhelming a casual user with too much information.

There are two editing modes that can be switched by a single button click (no.4 in figure 5): editing the hierarchy or editing the geometry of a selected segment. Each of them have specific interface components (described in sections 4.1.1 and 4.1.2), but there are also universal widgets. The first of the two universal widgets is the editing widget (no.6 in figure 5). It consists of six arrows representing the six degrees of freedom (color coded). They are sensor enabled, and clicking on them would mean that the currently selected coordinate(s), whether they represent a joint position, a single point or a set of points from a segment geometry, will be displaced in the direction the clicked arrow points into. The other universal widget, the unit selection widget (no.5 in figure 5) allows the selection of values for the magnitude of all the modeling operations. The bigger this value, the coarser the modeling operations will be. The edit widget and the two editor's different functions use this value for coordinate calculations.

4.1.1 Hierarchy editing mode interface

When a hierarchy file is loaded, a visualization of the joint hierarchy (skeleton) is generated from the information found within the file. The joints are represented by green-red (red when selected) sensor boxes and their parent-child relationship is represented by line segments connecting the related joints. There are also Terminal sensors and terminal lines, which represent the segment lengths of terminal joints. These sensor boxes are colored turquoise-yellow (yellow when selected) and provide the displacement functionality for the modeled character's endpoints. These sensors do not participate in hierarchy modifying functions.

The only local widget of the hierarchy editor is a pedestal (the rotation widget), on which the hierarchy representation is placed. By dragging the pedestal, the user can position and preview the skeleton (and at the same time the whole character) from any desired position along the Y rotation axis.

The hierarchy building functionalities of the editor are accessible using the function buttons. There are four function buttons defined:

"Insert Joint" button. If this button is activated, it adds to the hierarchy a joint as the parent of the selected joint and as the child of the selected joint's former parent. The insert operation inserts a new joint between a joint child-parent relationship (and not between a parent-joint relationship) to take advantage of the specific property of one joint having one parent versus the ambiguous property of one joint having possible multiple joint children. The functionality is enabled for all joints except the root joint, which does not have a parent to which to add the new joint. According the symmetry rules, if the new joint's parent is an internally symmetrical joint, then the new joint will be also internally symmetrical, else if the parent joint is an externally symmetrical joint, then another joint is added to the parent joint's symmetrical joint, and the two new joints will be the externally symmetrical joints of each other.

"Add Terminal Joint" button. This function button is enabled only if terminal joints are selected. It extends the hierarchy with one additional joint using from the former terminal joint and segment their direct properties (coordinates) and their indirect properties (length, orientation) to generate new properties. The same symmetry rules apply as with the previous function button.

"Bifurcate Joint" button. Clicking on the button will add two joints as the children of the currently selected joint. These new joints will split the hierarchy line at the selected joint, allowing more diversity in the hierarchy. The function is enabled with every joint selected, thus this function can be used for hierarchy refinement and for extending the hierarchy as well (by applying the function to internal respectively terminal joints). The same symmetry rules apply with this function also, but with the slight change that in case of external symmetry, two pairs of joints are added, one pair for each of the selected joint respectively for its externally symmetrical joint.

"Remove Joint" button. This function performs joint removal operations, with the scope of reducing the complexity of hierarchies. It is enabled with every joint selection except the root joint and removes the joint from the hierarchy, but preserves the geometry associated with the joint removed and appends that geometry to the parent joint's geometry (segment concatenation is performed).

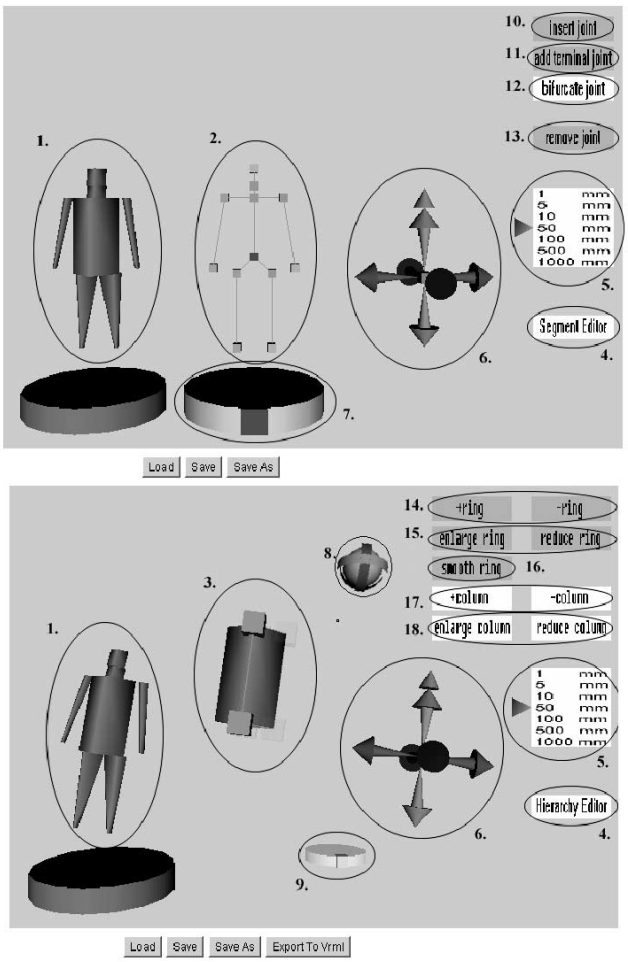

Figure 5. The editing environment in two stages, on top with the hierarchy editor active, on the bottom with the segment editor active. The elements are: 1.Whole character, 2.Hierarchy representation, 3.Selected segment, 4.Editor selection buttons (superimposed), 5.Unit widget, 6.Edit widget, 7.Hierarchy rotation widget, 8.Segment rotation widget, 9.Zoom widget, 10-13.Function buttons: Insert Joint, Add Terminal Joint, Bifurcate Joint, Remove Joint; 14-18.Function buttons: Add/Remove Ring, Positive/Negative Ring Scale, Smooth Ring, Add/Remove Column, Positive/Negative Column Scale.

4.1.2 Segment editing mode interface

There are widgets that are local to the segment geometry editor, and these are a rotation widget and a zoom widget. The rotation widget or tool is represented as an ensemble of three color-coded circular stripes which have rotation sensors linked to them. By dragging these stripes, the user can view the geometry from all angles and can position the geometry as desired in the course of the editing process. Since the rotation axes are separated in the widget, a precise handling is possible. The rotation is also conveyed to the edited character geometry and to the editing widget (arrows). This way, not only the geometry is positioned, but the axes of the editing arrows and the actual axes of the vertices remain aligned.

Another tool present in the geometry editor is the zoom tool, which eliminates the difficulties of editing too large or too small objects by zooming them to a comfortable size level.

The remaining elements of the geometry editor are a number of function buttons. Their functionality, similarly to the hierarchy editor functions, is to transmit their activation status to methods in the Java applet that will calculate and effectuate the actions related to the buttons. The button functionalities are:

"+ ring" or "+ column". Add a new ring or new column right after the current ring respectively column selection.

"- ring" or "- column". The selected ring respectively column vertices will be removed from the geometry.

"enlarge ring" or "reduce ring". The selected ring is enlarged or reduced and this introduces changes in the geometry thickness locally.

"enlarge column" or "reduce column". A column scale operation acts on every vertex of the selected segment geometry. This means that the geometry will become longer (taller) respectively shorter, while the thickness does not change.

"Smooth Ring". The contour line of the selected ring's vertices are smoothed.

The geometry editing interface is described as a separate tool with more detail in [8] and [9].

5 Data handling

The system uses a simple text format to store hierarchy, joint, respectively segment information. The main hierarchy file is a text file representing hierarchy relationships for one single character. In this file, for each joint there is a line record specifying the joint's name, the name of it's child segment, the name of it's parent joint and for later use lists for child Site and Displacer names. The internal hierarchy information stored in joint and segment objects is extracted from this main file.

The joint and segment files (there is one separate file for each joint and segment) are in a subdirectory which is specified by the hierarchy file's name. This method of file handling was used for the possibility to integrate at a later stage other entities of the H-anim standard: sites and displacers, as well as other extensions, for instance animation data files.

File segmentation could also provide the possibility to interchange the different segments/joints to compose new characters from existing components. The joint and segment files are simple text files which store the properties of their owners.

Finally, the system has the option to export the edited character in H-anim style into a VRML file specified by the user. It uses the Humanoid, Joint and Segment nodes style specified by the H-anim specification. Java3D export can be also included, since the H-anim specification is file format independent and our Java3D geometry export functions are already used in a separate geometry editor tool.

6 Symmetry

The system handles symmetries automatically to make the modelling task easier. The types of symmetries used is internal and external symmetry, applicable for both joints and segments. Since the H-anim standard specifies the default character position as facing the viewer (it is looking into +z direction) the symmetry plane can be considered the YZ plane where X=0 (defined by the root joint and other internally symmetric joints).

6.1 Internal symmetry

In case of internal symmetry, the joint's segment child has geometry coordinates that are the symmetric of each other respective to the symmetry plane. This is relevant to the geometry editing: when an internally symmetric segment is edited, its control structure is built only for half of the geometry coordinates, the other coordinates are automatically adjusted to maintain the symmetry. Additionally, joint centerpoints and segment coordinates that are on the symmetry plane, are not allowed to "escape" from that plane. Regarding the hierarchy editing operations, internal symmetry influences the created new joints as follows:

- if a single joint is added and the parent joint is an internally symmetric joint, the new joint will be also internally symmetric

- if two joints are inserted to an internally symmetric parent joint, the new joints will be the externally symmetric joints of each other.

6.2 External symmetry

External symmetry means that a joint and its segment has a joint-segment pair whose center point and coordinates are the symmetrical of their center point and coordinates. With external symmetry, the hierarchy controls are enabled only for one side of the hierarchy and the operations effectuated on them are mirrored to their externally symmetric joints. In case of segments, the control structure now covers all the geometry and as in case of joints, the operations effectuated on them are mirrored on the externally symmetric segment's coordinates. The hierarchy rule for external symmetry is: all the inserted joints will have an externally symmetric joint regardless of the parent joint's symmetry type.

7 Conclusions

In our environment, the user does not have to handle the complexity of commercial [professional] graphics modeling packages for shape control, nor the limits of predefined body components present in specialized modeling packages in case the user needs a customized human character or other types of characters than human. However, next to the direct manipulation possibilities, presents a degree of indirection for joint center and geometry coordinatre displacements (a separate edit tool), which can be remedied by using specialized widgets.

This paper states a useful goal and provides an interesting and useful solution. While it is not the most powerful tool imaginable, it meets the goal of easy usability and provides a unified and consistent design. The design works within the context of VRML and EAI, leveraging appropriately off of what is supplied by these technologies.

As an advantage from using regular geometries, the system can be easily converted into using NURBS-based geometry instead of polygon based geometry [6] and even lattice-based geometries could be used [12]. Since the regular grid coordinates satisfy the control point requirements of NURBS surfaces, they can be easily converted into control points (using appropriate dimensions, the surface will even pass through the control points as in case of polygons, retaining the general shape of the geometry). Then the same primitive operations (vertex, ring and column operations) apply to the NURBS control points, providing a high level manipulation possibility (both for complexity and surface).

8 Future work

One aspect of the system which can be refined is the manipulation aspect. Currently, editing is done with indirect manipulation, using a clickable editor widget targeted to the selected primitive. The editor widget/unit widget combination can be replaced with a directly manipulated editing widget, for instance one integrated into the selection sensors.

On a more technical note, the visualization of parent-child relations could be improved to show the direction of the relations (which is the parent and which is the child) for instance with the traditional diamond shape. The current line representation was chosen for simplicity.

The system is being extended to include more functionalities. These planned extensions will provide animation possibilities (including deformations via Displacers), Sites (additional elements of the H-anim specification) and textures. In the case of texturing a per-coordinate texture manipulation approach is used in a separate geometry editing module that is being integrated into the system.

References

[1] C. Babski and D. Thalmann. A Seamless Shape For HANIM Compliant Bodies. In *VRML '99, Proceedings of the fourth symposium on The virtual reality modeling language*, pages 21–28, Paderborn, Germany, 1999. ACM Press.

[2] E. A. Bier. Skitters and Jacks: Interactive 3D Positioning Tools. In *Proceedings 1986 ACM Workshop on Interactive 3D Graphics*, pages 183–196, Chapel Hill, North Carolina, New York, october 1986.

[3] Chris Marrin - Silicon Graphics, Inc. External Authoring Interface Reference. http://www.graphcomp.com/info/specs/eai.html, 1997. Proposal for a VRML 2.0 Informative Annex.

[4] D. B. Conner, S. S. Snibbe, K. P. Herndon, D. C. Robbins, R. C. Zeleznik, and A. van Dam. Three-dimensional widgets. In *Computer Graphics (1992 Symposium on Interactive 3D Graphics)*, volume 25(2), pages 183–188, 1992.

[5] J. Döllner and K. Hinrichs. Interactive, Animated 3D Widgets. In *Proceedings of Computer Graphics International CGI'98*, Hannover, Germany, 1998.

[6] H. Grahn, T. Volk, and H. J. Wolters. NURBS in VRML. In *Proceedings of the Web3D-VRML 2000 Fifth Symposium on Virtual Reality Modeling Language*, pages 35–43. ACM Press, 2000.

[7] Humanoid Animation Working Group Web3D Consortium. H-Anim: Specification for a Standard VRML Humanoid, version 1.1. http://www.h-anim.org/Specifications/H-Anim1.1/, 1999.

[8] S. Kiss. A simple, visual mesh editor. CTIT Technical Report series, No. 01-27, ISSN 1381-3625, 2001. Technical Report.

[9] S. Kiss. Web Based VRML Modeling. In *IV2001 Information Visualisation*, pages 612–617, London, England, 2001.

[10] G. Maestri. *Digital Character Animation 2*, volume 1 - Essential Techniques. New Riders Publishing, 1999.

[11] A. Nijholt and H. Hondorp. Towards Communicating Agents and Avatars in Virtual Worlds. In A. de Sousa and J. C. Torres, editors, *EuroGraphics 2000 - Short presentations*, pages 91–95. EuroGraphics, 2000.

[12] A. Wakita, M. Yajima, T. Harada, H. Toriya, and H. Chiyokura. XVL: A Compact And Qualified 3D Representation With Lattice Mesh and Surface for the Internet. In *Proceedings of the Web3D-VRML 2000 Fifth Symposium on Virtual Reality Modeling Language*, pages 45–51. ACM Press, 2000.

[13] Web3D Consortium. VRML97: The Virtual Reality Modeling Language. http://www.web3d.org/technicalinfo/specifications/vrml97/, 1997.