HANIME
An H-Anim Compliant Avatar Editor for NVEs

Christos Bouras
Research Academic Computer Technology Institute (CTI), Greece and
Computer Engineering and Informatics Department (CEID), University of Patras, University Campus, GR 26504, Patras, Greece
bouras@cti.gr

K. Kartsakalis
Computer Engineering and Informatics Department Department (CEID), University of Patras, University Campus, GR 26504, Patras, Greece
kartsaka@ceid.upatras.gr

V. Triglianos
Computer Engineering and Informatics Department Department (CEID), University of Patras, University Campus, GR 26504, Patras, Greece
triglian@ceid.upatras.gr

Th. Tsiatsos
Department of Informatics, Aristotle University of Thessaloniki, University Campus, GR 26504, Thessaloniki, Greece
tsiatsos@csd.auth.gr

Keywords: H-Anim, Graphical User Interface, Networked Virtual Environments, VRML, X3D.

Abstract: This paper addresses the problem of a visual H-Anim avatar editor. Although H-Anim is a very promising standard for interchangeable avatars and avatar's animation, the lack of an effective yet simple to use visual avatar editor could repel users from H-Anim compliant Networked Virtual Environments. Therefore this paper presents an H-Anim compliant editor, called HanimE, for creating avatars for Networked Virtual Environments. This editor is entirely based on a graphical user interface and allows users to customize H-Anim avatars, in a simple yet effective manner. The avatars created by HanimE can be integrated in various H-Anim compliant Networked Virtual Environments regardless of the technology they are built on, since they are X3D files.

1 INTRODUCTION

The popularity of Networked Virtual Environments (NVE's) has rendered them as part of everyday life for many online users. Online users are represented by avatars in the virtual worlds (Singhal & Zyda, 1999). These avatars, in most cases, are three dimensional (3d) representations of a human like model. Users tend to develop a psychological bond with their avatars. One of the main tasks in an NVE is the integration of humanoid avatars augmented with humanoid animations. This could be useful for the users to transfer their avatars from one NVE to another. In many types of NVEs, such as Massively Multi-User Online Role-Playing Games, users are able to control and change the appearance of their avatars (Yee, 2006). Usually, the users exploit this functionality to customize their avatars in order to reflect their real or virtual personality. The virtual characters are starting to be widely used in user

interfaces in order to improve the human–computer communication (Del Puy Carretero et al., 2005).

The creation of virtual humans that are compatible with many different NVE platforms is a challenging task, since the creation of humanoid animation is a complex task, which usually requires particular skills and training (Buttussi et al., 2006). To that direction the H-Anim specification is proposed. H-Anim is now included in X3D standard, and it describes humanoids as an hierarchically organized set of nodes. The contribution of this paper is a multi-platform Java based H-Anim compliant editor for modifying the appearance of avatars. This editor is called HanimE (H-anim avatar Editor). H-Anim specification is by definition geared towards flexibility, compatibility and simplicity. However, the variety of the available VRML/X3D editors, are not suitable for non-experienced users concerning the creation of humanoid avatars. Users who have limited knowledge about the fundamental concepts behind x3d (such as the usage of "nodes" and "routes" as well as the definition of geometries) cannot easily create H-Anim avatars. Most frameworks available are either geared towards the experienced 3d artist exploring the capabilities of VRML, or they support the H-Anim avatars creation by offering tools for developing animation and simulation of movement. Either way, this makes things complex and discouraging for the average user who is seeking a graphical tool to modify the visual properties of his/her avatar, such as colors, skins and virtual items, before importing it to the NVE of his/her choice. Given the limited number of H-Anim compliant avatars, the creation of new avatars is a strenuous activity which requires hours of work even for the experienced 3d artist. This is a limitation in the broad adoption of H-Anim specification. Furthermore, there is a significant number of NVE communities that are not providing a large number of available avatars to their potential users. For example open source or research projects cannot afford to hire 3d designers for this task. This discourages users from participating to such a community since users want to project their individuality through their avatar in a virtual world. A convenient solution to this problem is to provide one type of avatar that can be customized in a graphical user interface by the users according to the his/her preferences. This solution not only saves resources, but it also allows users differentiate their avatars.

This paper is structured as follows. The next section presents similar tools. Next we are describing the main design characteristics of HanimE and the functionality it offers. Afterwards, we are presenting implementation and interoperability issues and the way we have faced them. Finally, some concluding remarks and planned next steps are briefly described.

2 RELATED WORK

Most of the tools for visual H-Anim avatar editing are either focused to specific tasks or complicated for the average user. On the one hand this restricts their integration to NVEs. On the other hand, it discourages the average user from using them. Wang and Ressler (2007) present a tool for viewing and manipulating CAESAR (Civilian American and European Surface Anthropometry Resource) generated bodies. However, this work does not apply to the average user neither from the perspective of functionality nor from the perspective of ease of use.

Cobo and Biery (2002) presented a Web3D toolbox for creating H-Anim compatible actors. The toolbox supports avatar modelling through primitives, meshes and NURBS, joint and facial animation. It relies on Shout3D, which is a proprietary and commercial platform for the visualization of the avatars. This limits its portability and ease of integration to NVEs. In addition it requires that the user has 3D modeling skills and intermediate knowledge of the H-Anim standard, in order to use it effectively. HanimE editor presented in this paper differs in two ways. First of all, it is based on non-proprietary plug-ins, which allows easy integration to third party platforms. Secondly, it is crafted in a way that the basic avatar processing tasks are performed in a way that no H-Anim specific knowledge is required from the user.

3 MOTIVATION AND RATIONALE

Our solution aims at bridging the gap between the user's demands for simple avatars creation and the compatibility with H-anim standard. For that reason we have implemented HanimE, that is offering to the avatar creators a graphical interface to create their avatars. All the editing takes place on a purely visual level, providing the user with all the sufficient graphical tools required so that s/he can see any change on his avatar instantly. Possible changes include the change of the color of avatar's shoes, the exact fitting of sun glasses on the avatar's head, or the complete replacement of a body part with a new

one. Any coding and/or technical issue concerning H-anim compatibility is transparent to the users. HanimE supports the following features:

- Automatic or manual H-Anim avatar loading.
- Manipulation of the Material Node properties such as colour, transparency etc.
- Automatic or manual H-Anim avatar scaling.
- H-Anim avatar rotation.
- Texture mapping.
- Virtual items library.
- Adding and adjusting size and orientation of virtual items on an avatar.
- Save X3D Scene to file.

HanimE is organized in three parts: (a) a visualizer; (b) library containing virtual items/ H-Anim body parts; and (c) a material node editor. This organization aims in guiding the user to the desired result through simple steps that are easy to follow.

The **visualizer** is the first user interface screen that appears when H-AnimE is launched. An avatar can be loaded both automatically or manually. Finally, from that screen a user can save his/her avatar. The visualizer is an extended VRML/X3D browser window that deals with the tasks of loading an avatar, visualizing it and exporting it to a file. In addition this window triggers the virtual items/ H-anim body parts library window and/or the material node editor. The loading portion is accomplished by a custom fully featured X3D/VRML parser. This parser enumerates various aspects of an VRML/X3D file such as characters, lines H-Anim Joints, Segments Humanoids, Displaces and Sites. Moreover, it calculates the Level of Articulation of an avatar as well as whether it is H-Anim compliant or not. Additional features include the VRML version of the file, syntax errors check, ROUTEs and PROTOs syntax check, Segments to Joints ratio check, orphan Joints check etc. When a user wants to add geometry to an avatar, either in the form of an item/accessory or in the form of an H-Anim compliant body part, the aforementioned library is triggered. In cases where the user needs to modify existing geometry, the material node editor is invoked.

The **virtual items/ H-Anim body parts library** is a library that hosts both virtual items, such as hats, sunglasses etc., and H-Anim compliant body parts, for example arms and legs. These can be imported, scaled, transformed and adjusted to an avatar. This library features both integrated and user added items. Users can add and instantly preview items to the library, from ordinary VRML/X3D files. As long as they have selected the items they want to import, these items are imported to the 3D scene that visualizes the avatar. Subsequently, the material node editor is launched to adjust the properties of the items or modify the avatar.

The **material node editor** consists of three panes. The first pane is a 3D previewer in which the user previews the changes on the avatar during the editing. The second pane holds a visual representation of the avatars joints, where each joint is represented by a small circle. Users can specify the joint that they like the item to bind to. The selected joint is highlighted and the item is bound to the selected joint, resulting in smooth item motion when the joint is animated. Finally, the last component is a tabbed pane that allows the users to adjust the position, the color and the texture properties of an avatar or a virtual item. These actions are performed in a visual way using graphical controllers. Furthermore, the user can enter a numerical value for the field s/he is changing, e.g. the diffuse color of a material.

In addition, in order to ensure portability, the editor is written in Java (and thus platform independent) and the processed avatar can be exported as an X3D or VRML file. Both characteristics allow the migration of the user's avatars to his/her favourite H-Anim compliant NVE platform.

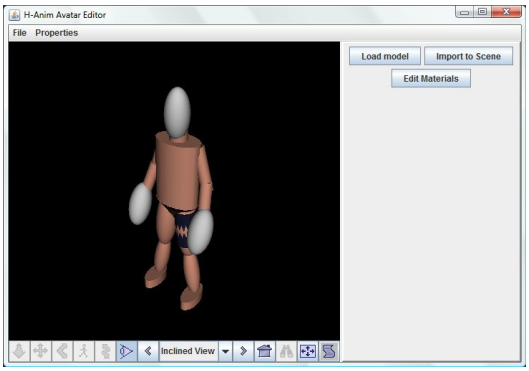

Figure 1: Main screen with automatic avatar loading.

4 AVATAR EDITING PROCESS

This section describes the avatar editing process. The function of the utility presupposes the existence of a VRML file that contains a valid H-Anim avatar, structured according to the H-Anim 1.1 specification, with the list of Joints named according to the full description of the specification. We also presuppose that available to the user is a long list of items written in VRML, each one contained in it's own VRML library, although that is not strictly necessary for the editor to function.

The operations supported by the editor on the avatar are: material editing, item/ body part fitting. As far as materials are concerned users can modify material properties such as color and transparency. Item/body part fitting describes the process of loading items or body parts to the Avatar Scene, selecting the Joint to which they will be attached and relocating them correctly so that they fit in the scene accordingly. The operation of the editor is based on the steps depicted in Figure 2.

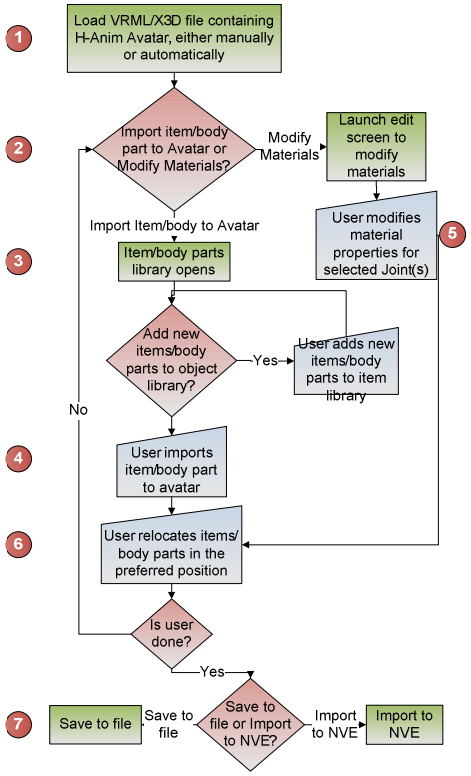

Figure 2: Mechanism Flow chart.

Step 1. Avatar loading: User loads the avatar on the startup editor screen, either manually, or by setting the editor to automatically load the avatar each time s/he launches the editor (Figure 1).
Step 2. **Selection of editing function**: The user chooses the editing function (a) import of an item or a body part to the avatar, or (b) editing the material nodes of the avatar. In the second case, steps 3 and 4 are omitted and the editor goes straight to step 5.
Step 3. Selecting items from virtual items library: User is shown the library of available items and body parts (Figure 3).

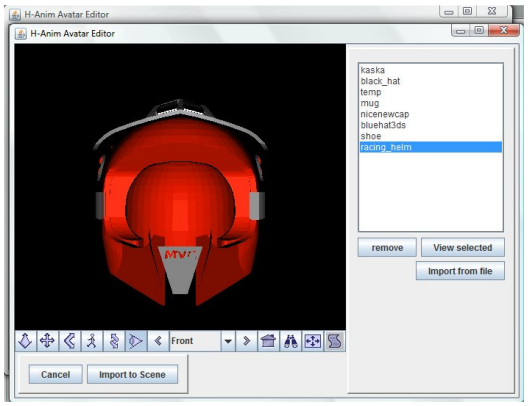

Figure 3: Virtual Items and Body Parts Library.

This library contains personal items/body parts for each user. Each user can fully manage this library by adding new items, body parts, removing old ones and/or selecting/viewing the items s/he wishes to import. This has to be a collection of non H-Anim files.
Step 4. Importing items from items library: User imports selected items/body parts to his/her H-Anim avatar. Both the avatar and the imported items/body parts appear on the next screen (the edit screen, Figure 5 and Figure 6), with the item given a set of tools for relocation on the X, Y, Z axis, scaling or rotation.

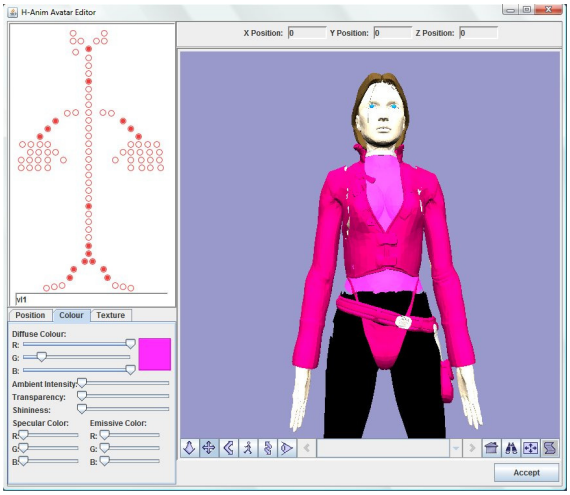

Figure 4: Changing Colors.

Step 5. **Editing the material nodes of the avatar**: The edit screen contains a view of the Scene (the H-Anim avatar, along with the imported items, if any), and a panel showing the H-Anim Joint Hierarchy. In this panel, the Joints used by the avatar are highlighted and selectable. Below this panel is a set of tools by which the user can modify the material properties of any selectable (highlighted) Joint.

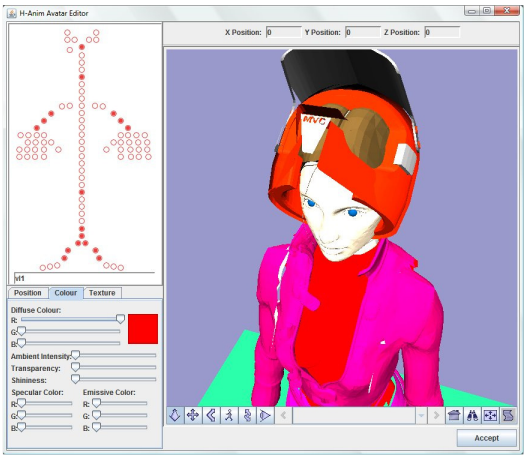

Figure 5: Adjusting a helmet to an avatar.

These tools appear as part of the "Colour" tab (Figure 4) of this area, and allow the changing of the material attributes: diffusion color field, transparency, shininess etc. Any changes occurring to the material appear instantly on the Scene.

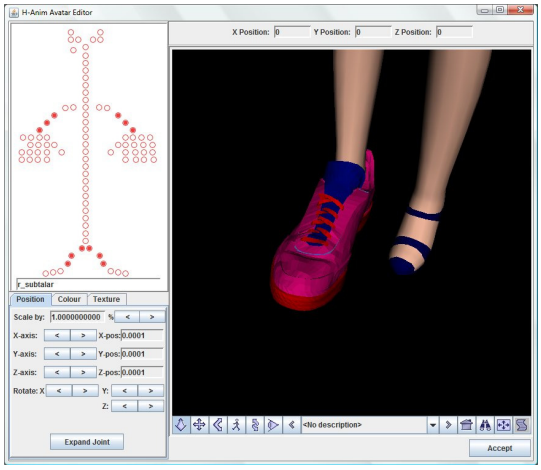

Figure 6: Adjusting a shoe to an avatar.

Step 6. **Relocation of items**: The "Position" tab of the same area is only enabled if user chose to import an item on step 2, otherwise this panel is empty. The list of tools for the relocation of the imported item is provided here, as explained in step 4. It should be noted that the user must select the Joint to which the item will be attached (e.g. a hat item would be attached to a node corresponding to the head). The editor provides the option for automatic relocation of the item to match the coordinates of the selected Joint. However, the user can take it one step further using the relocation tools to fit his item exactly where he wants it to be before attaching it to the selected node. Once that happens, the item is moving along with the rest of the H-Anim avatar, following the corresponding ROUTEs of the Joint.

Step 7. **Save/Export**. The user can choose to save the avatar's file, or to export it to the NVE for which the editor is to be used.

5 IMPLEMENTATION AND INTEROPERABILITY ISSUES

One of the major goals of the editor is to be portable, while complying to open and widely used standards. In order to obtain portability the whole application is programmed in Java that makes it executable on virtually every platform for which a Java Virtual Machine exists. The manipulation and visualization of the X3D models was developed using the Xj3D Toolkit, a Java base toolkit for X3D and VRML content. However, the usage of Xj3D entailed some problems.

One of the major issues with the use of the Xj3D toolkit is the complete absence of an integrated "Save Scene To File" function, that would take full responsibility for extracting all the valuable data from an X3DScene Object in Java and writing them in the correct format in an output file. The solution to this issue is a challenging one, considering the fact that not all Nodes in an X3DScene are named, and that the Xj3D package does not allow the user to read values from certain field types, therefore rendering the implementation of a "Save Scene to File" function from scratch nearly impossible. Our solution to this problem is based on the fact that the user makes use of already existing X3D files, both for the avatar and the items to be imported, which can be parsed by Java and treated as simple text files following the appropriate X3D format. When the applications needs to save changes to the loaded file, it opens up the loaded file using the tools contained within the java.io.* package used for reading/writing to a file as text. Subsequently the application scans the file for the appropriate position(s) to save the new data and carefully removes any old data that has to be replaced, with the edited file being stored in the same folder used by the utility and the original file intact, only to be replaced with the new one when the user clicks on either the "Save" or "Save As" button.

The editor is designed to be used as a stand-alone application for creating and editing avatars. However, in case that the NVE designer chooses to integrate HanimE to a NVE platform, the following have to be considered: An NVE administrator integrating HanimE to his NVE platform would have to provide access to three screens either prior to the

importation of the user avatar to the NVE's world, or as an extra editor once the user is inside the NVE. These three screens are the HanimE project Frames (using javax.swing.JFrame): the startup screen, the item library screen and the edit screen, which connect with each other. In any case, the NVE can read the edited H-Anim avatar with the new values either directly, from the x3d Browser contained within the startup and edit screens as an X3D Scene, or by reading the corresponding saved filed after the user has edited his avatar. In the latter case, the editor automatically generates a backup VRML file in the same folder as the editor, which can be used as a direct reference.

6 CONCLUSIONS – FUTURE WORK

An H-Anim compliant avatar editor was presented in this paper. The focus of the authors is to provide a portable easy to use graphical editor for avatar customization, while at the same time comply with the H-Anim specifications. The application is developed upon portable and open standardized technologies to allow platform migration, such as Java and Xj3D. It features a friendly easy to use graphical interface that the average user can handle with ease. The editor supports functions such as adding H-Anim body parts and items to an avatar, avatar rotation, changing colors and more, and can save the prefered state of an avatar as an X3D file. Furthermore, an VRML/X3D parser utility presents useful information to the end user.

Among our future plans is the enhancement of the editor with new features. More specifically, we intend to integrate the work presented in Bouras et al. (2009) that will make possible the addition of H-Anim animations to the user's avatar. Moreover, an animation authoring tool will provide animating capabilities to the average user. A user will be able to add, remove and adjust animations to an avatar, as well as to create custom animations through a visual wizard. Furthermore, an avatar creator tool that will allow the creation of custom avatars is to be integrated to the next version. The tool will follow the same principles HanimE follows, that is efficiency combined with ease of use. The user will have a set of primitives and body parts that he will be able to modify and combine in order to form his/her avatar. Finally, to improve H-Anim interoperability, an H-Anim syntax validation and correction tool will be incorporated to the editor. It will parse VRML/X3D files imported by the user and it will validate them against the H-Anim standard. In case that the file has errors or incompatibilities with the H-Anim standard, the tool will try to repair the file or will suggest solutions to fix the problems. We believe that the above enhancements will offer a powerful tool to the average user and will enforce the position of the H-Anim standard, eliminating some of the difficulties that prevent it from being widely spread.

REFERENCES

Bouras, C., Chatziprimou, K., Triglianos, V., Tsiatsos Th. 2009. A framework for H-Anim support in NVE'. In proceedings of the *International Conference on Computer Graphics Theory and Applications 2009* Lisboa, Portugal, February 5-8.

Buttussi, F., Chittaro, L., Nadalutti D. 2009. H-Animator: a visual tool for modeling, reuse and sharing of X3D humanoid animations. In proceedings of the *Eleventh international Conference on 3D Web Technology* Columbia, Maryland, April 18-21, 2006.

Cobo, M., Bieri, H. 2002. A Web3D Toolbox for Creating H-Anim Compatible Actors. *Computer Animation 2002*, pp.120.

Del Puy Carretero, M., Oyarzun, D., Ortiz, A., Aizpurua, I., Posada, J. 2005. Virtual characters facial and body animation through the edition and interpretation of mark-up languages. *Computers & Graphics*, Volume 29, Issue 2, April 2005, Pages 189-194, ISSN 0097-8493, DOI: 10.1016/j.cag.2004.12.003.

Humanoid Animation Working Group (2004). H-Anim.http://h-anim.org.

Singhal, S., Zyda, M. 1999. Networked Virtual Environments: Design and Implementation. *Addison Wesley*, Reading, MA.

Wang, Q. and Ressler, S. 2007. Generation and manipulation of H-Anim CAESAR scan bodies. In proceedings of the *Twelfth international conference on 3D web technology*. Perugia, Italy, April 15-18, pp. 191-194.

Yee, N. 2006. The Psychology of Massively Multi-User Online Role-Playing Games: Motivations, Emotional Investment, Relationships and Problematic Usage. *Avatars at Work and Play, Book Series Computer Supported Cooperative Work*, Volume 34, ISSN 1431-1496, Springer.