AVRATAR: A VIRTUAL ENVIRONMENT FOR PUPPET ANIMATION

Alexander Redei
Department of Computer Science & Engineering
Univ. of Nevada, Reno
Reno, Nevada, 89557
redeia@unr.nevada.edu

Ed Tumbusch
Department of Computer Science & Engineering
Univ. of Nevada, Reno
Reno, Nevada, 89557
edtumbusch@sbcglobal.net

Josh Koberstein
Department of Computer Science & Engineering
Univ. of Nevada, Reno
Reno, Nevada, 89557
koberste@unr.nevada.edu

Sergiu Dascalu
Department of Computer Science & Engineering
Univ. of Nevada, Reno
Reno, Nevada, 89557
dascalus@cse.unr.edu

Fred Harris
Department of Computer Science & Engineering
Univ. of Nevada, Reno
Reno, Nevada, 89557
fredh@cse.unr.edu

ABSTRACT

When it comes to three dimensional computer animations, the use of live actors to control the movements of onscreen characters produces a realism that is unsurpassed. But so are the costs, and that is why Avratar, a virtual environment for puppet animation, uses widely available products utilizing motion capture technology to control a real time puppet in a world limited only by the imagination of the user. This paper introduces Avratar, a virtual puppet animation system, where users are able to utilize the natural movements of their own body to create dazzling real time performances or animated cinema. The UML-based software model of Avratar is presented in this paper, in addition to examples of application, and planned future work.

KEYWORDS: Virtual reality, animation, P5 glove, digital puppeteering, software engineering,

1 INTRODUCTION

Avratar is the brainchild of team triple-eight, which consists of the five authors listed on this document. The original inspiration was from Ed Tumbusch, who recalls seeing an animated bumper while watching TV. The bumper was a 3D digital character that would promote upcoming programs. The character was controlled via motion sensing gloves that were connected to a mainframe system. When a user moved the gloves an animated character mimicked their movements. When Ed described this to our team, we all liked the idea; and thus, Avratar: a virtual environment for puppet animation was born.

Digital animation methods have been available to the movie industry for years [1]. Real-time 3D animation methods are almost always passed in favor of cheaper, and less tedious methods. Consumers are unlikely to adopt Hollywood-style animation methods because they are too expensive and take too much time.

Actual consumer products that do some aspects of real-time 3D animation indeed exist, but are very limited in application and do not use graphics standards. Such an example of this would be iClone [2], a software that mimics the user via a digital avatar with the use of a webcam and a microphone

Users of Avratar are able to utilize a widely available and inexpensive input peripheral, the P5 glove, and standard 3D editing tools to transform natural hand and finger movements into sophisticated and fun digital animated cinema. Users can roll their own digital puppets by combining pre-existing models (head and hand puppet models) into a character they wish to animate. The user selects an appropriate environment (a cave, for instance), and the animation process begins. This consists of a multi-layered concept: meaning in each layer a user works with one item of an animation, until the user is satisfied and the layers are combined into a single animation.

As discussed above, there are several existing methods of modeling a digital puppet, but they are slow, tedious, and wildly expensive. Avratar delivers an affordable platform for digital puppeteering [3, 4] utilizing standards instead of custom solutions. The result is a new kind of digital animation that allows for a user, with little training and minimal expenses, to deliver dazzling animations.

The remainder of this document is structured as follows: Section 2 outlines related work in the field of digital puppeteering and shows the methods available. Section 3 details the functional and non functional requirements of Avratar in addition to use case models, in UML format [5]. High-level design is described in Section 4, Section 5 describes results with user

applications, and Section 6 details the future direction of Avratar. Our conclusions are presented in Section 7.

2 BACKGROUND

Digital animation methods have been available, and in high demand in the movie industry for years [1]. Digital animation is mostly done manually by the work of specialized artists, and true digital animation via motion capture is usually passed in favor of other techniques because of the costs. The methods used by the movie industry usually involve slow, tedious, and highly uncomfortable work (such as painting an actor's entire body blue or green, then attaching sensors like poka-dots to their bodies, and having them perform, naked, in front of a camera for hours). There is no consumer market with such methods; a consumer would not have the money, nor find the interest in this kind of digital animation.

One product that is consumer-oriented, called iClone [2], allows the user to create a three dimensional avatar complete with lip synchronization from a voice recording. Provided the user has a camera and a picture of his or her face, the iClone software will digitally mimic, through a 3D avatar, the user's facial movements based upon input from the camera and a microphone. This program does not provide for any customization and does not allow for user-controlled animation.

MotionBuilder [6], a product developed by AutoDesk Inc., allows a user to animate a 3D character in real-time by wearing motion capture devices. MotionBuilder, like Avratar, allows a user to create an environment, and use special effects. MotionBuilder, however, is not consumer-oriented and comes with a hefty price tag.

3 REQUIREMENTS MODELING

Following standard software engineering guidelines, the main functional and nonfunctional requirements of Avratar are presented below.

3.1 Functional Requirements

R01 The software shall have a complete GUI widget system implemented through CrazyEddie [7].

R02 The program's menu system shall be entirely rendered, that is, all menus should be in OpenGL [8] (for appearance, speed, and ease of use).

R03 The workflow will be separated into two phases: an initial recording phase, and an editing phase.

R04 The menu may contain an exportation tool where a scene/show can be exported into standard movie formats (.avi, .mpeg, etc).

R05 The user should be able to create, load, and modify custom models for use with 3D digital puppets.

R06 The software shall be able to take input from the P5 glove [9] and animate a model by using the 3D input peripheral.

R07 The software should be able to map P5 glove inputs to components (set which glove will control which object).

R08 The software should be able to map the sensitivity of P5 input to components in the scene (set the sensitivity of how fast each object will move).

R09 The software shall be able to record and set the position of sound through the OpenAL sound wrapper [10].

R10 The program may contain the functionality to be able to load and play user-specified music and sounds.

R11 The program shall be able to record the state of each object at any given time.

R12 The program should be able to take the state of each object and be able to modify it (editing in layers, so as to add to the fun of the movie).

R13 The user should be able to edit or switch the perspective (camera) of the scene either during recording or through the editor.

R14 The program may contain pixel shader effects that will modify graphical primitives of the scene (for instance, one could render the movie to look like a cartoon show).

R15 The program may contain a particle generation engine that can be set by the user to create sparks and particles in the scene.

R16 The program may contain the functionality to generate fog and modify the visibility based on the fog parameter in a scene.

R17 The program shall be able to save scenes for later use and load saved scenes.

R18 The program shall be able to play back previous recordings.

3.2 Non-Functional Requirements

N01 The program shall have many widgets and icons, and will refrain from using text with a preference toward easily understandable GUI elements.

N02 The GUI design will refrain from using layers of windows, and instead will favor a tabbed menu and editing system.

N03 The program should be able to run on any modern system (1.8+ Ghz P4, 512MB RAM) without stuttering or stalling.

N04 The program may be able to run on older systems (< 1.50 Ghz, 256MB).

N05 An average user with three hours of usage shall be experienced enough to create a movie scene.

N06 An average user with six hours of usage should be able to modify objects in the editor in layers.

N07 An average user with ten hours of usage may be able to create particle effects and use "shaders" (modify graphical primitives, such as rendering a cartoon).

3.3 Use Case Modeling

Avratar is a studio for 3D digital puppet animation, and as such it incorporates a fairly elaborate workflow (see Figure 1). In essence, digital animators and directors are the main users of avratar and utilize it either for a live puppet show, or a saved puppet show which can be played back later.

3.4 Description of Selected Use Cases

Load scene - This allows the user to load a pre-built environment into the avratar scene for animation.

Load model - This will take a model from most 3D modeling studios and import it into avratar.

Select model - This allows the user to recursively select models until they've built a character, which will be then be loaded into the avratar scene.

Load character - This allows a user to load a pre-built character, which is then available in the avratar scene.

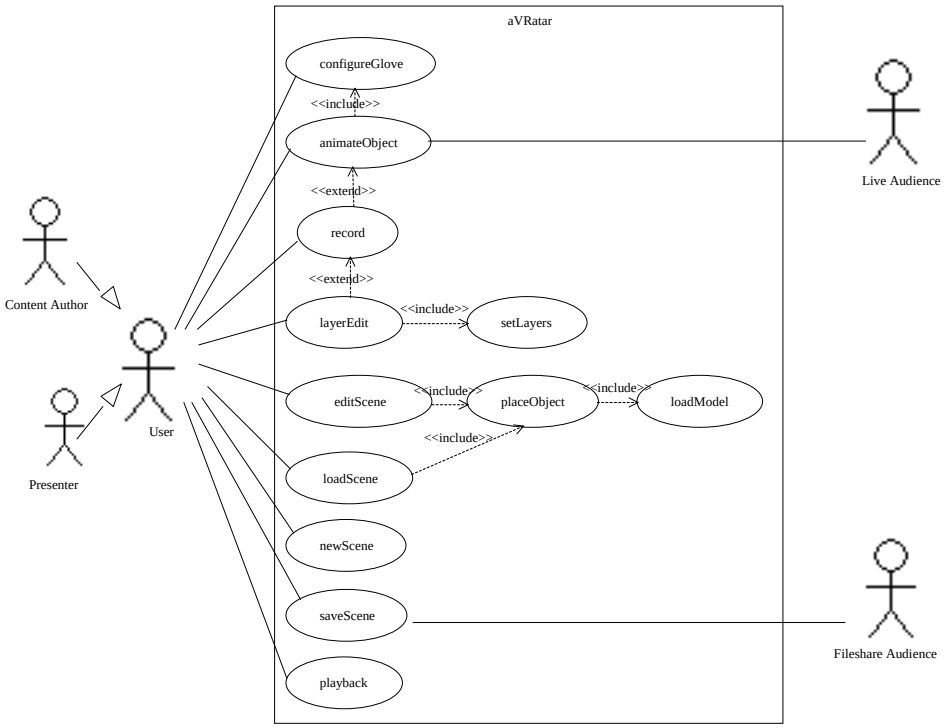

Figure 1. Avratar simplified use case model.

Record a layer - This uses input from the P5 peripheral glove to model the movements of an object. Each layer contains a special effect, sound, or an animation.

Edit a layer - Allows the user to override a layer with input from the P5 glove.

Remove a layer - Deletes the specified layer.

Record sound - This uses OpenAL to record sound into the avratar animation. The sound track can be overwritten if needed.

Apply special effects - This allows the user to apply special effects, such as cell-shading, to the scene.

Playback - Allows the user to playback a recorded scene.

4 DESIGN OVERVIEW

This section provides an overall perspective on how the system was designed and implemented. Avratar was designed as a virtual puppet animation studio, and as such we utilized standard wrappers for the 3D rendering, GUI, and sound system. The 3D rendering was achieved via Ogre [11], a LGPL cross-platform wrapper for Direct X and OpenGL. OpenAL [12], a LGPL sound wrapper, was used for the recording and playback of sound. Finally, the GUI was done via CrazyEddie [7], a LGPL GUI toolset with a library of pre-built widgets.

4.1 High Level Design

Avratar's main design can be divided into six components which are illustrated in Figure 2. These components are: the core, the playback system, the recording system, the scene editing system, the layer editing system, and the configuration system. The scene editing system is responsible for the environment, and the layered editing system works with the recording system to provide rich multimedia through layers. The configuration system provides real-time mappings for input from the P5 glove (for instance, mapping the current input to hand movements, or redirecting future passes to body animation).

4.2 Design Diagram

Figure 3 shows the organization and dependencies of the core and recording components of Avratar. In short, the core of Avratar consists of the modeling system which uses "bones" [13] to create character animations. Each bone is a fundamental element in a moving system; each bone must move as a single unit. The bones are connected via pointers that indicate the order bones are connected and how they behave when a neighbor is moved [14, 15, 16, 17]. The core of Avratar takes this data to create a digital puppet.

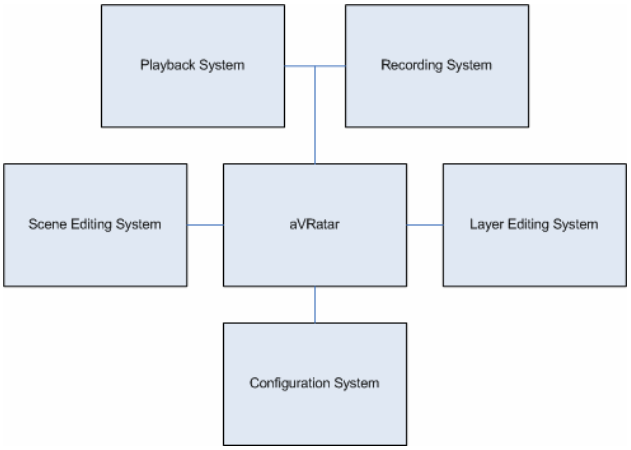

Figure 2. Avratar system level diagram.

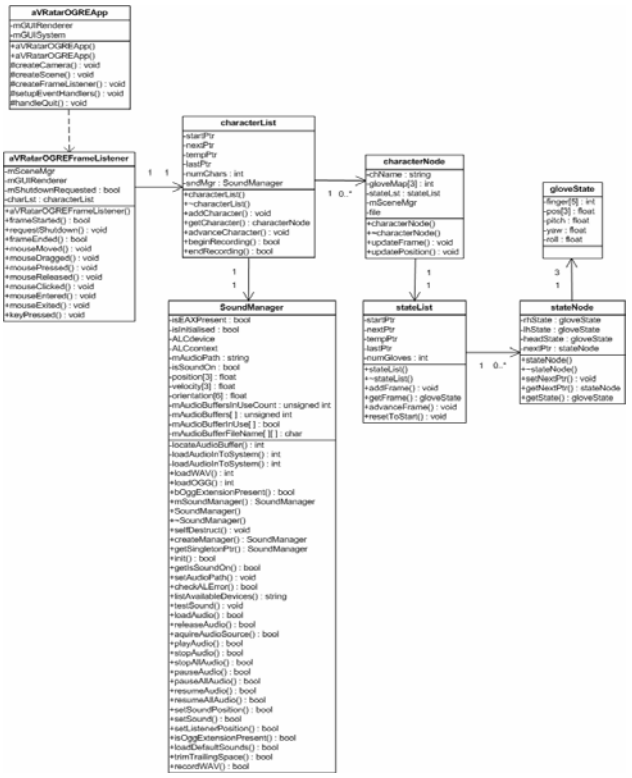

Figure 3. Avratar design diagram.

The recording system consists of multi-layered editing (each layer is stored as a node in a dynamic list), and multi-threaded sound recording and playback system. The sound system is performed on a separate thread in the application because it needs systematic access to I/O that cannot be guaranteed with a linear-programmed solution.

5 RESULTS: AVRATAR IN ACTION

This section provides an inside look into the Avratar system and the user interface with applications. Avratar utilized the CrazyEddie GUI wrapper [7], a LGPL product that provides a GUI platform through OpenGL, and also comes with a library of pre-built widgets. Custom GUI components were built to suit the needs of Avratar, for instance, the scene selector, which used a special render-to-texture GUI widget.

5.1 Scene Selector

This screenshot in Figure 4 represents what the user sees when prompted to select what scene they would like to use. The user will recursively select from the scene selector parts they want to use in their digital animation (head, hand models, environment, etc) until they've built an entire scene and are ready to start. At each stage, Avratar provides an advanced previewing feature by allowing the user to try out a model using direct input from the P5 glove in a separate view space.

Figure 4. Avratar scene selector.

5.2 Glove Mapper

The glove mapper (shown in Figure 5) screen is the heart of the layered editing approach utilized by Avratar. Every single controllable part of the character is listed in the glove mapper, along with a checkbox that controls its state. Each time a new layer is made, the checked items will be overwritten with the new input and the non-checked items will keep the previous animation. If no value has been recorded for an item, the default value of 0 is used.

5.3 Audio Options

This screen in Figure 6 represents what the user is presented with for audio options. The audio component of Avratar is driven via OpenAL [10], a LGPL sound wrapper from Creative. OpenAL provides a common platform for sound playback, recording, and processing. Avratar utilizes playback and recording via a multi-threaded sound system, and some digital processing. In figure 5, the user is able to determine if they want to record or overwrite the audio during the next pass of recording. In addition, the user has the option to mute or increase the gain of a previously recorded audio track.

Figure 5. Avratar input mapper.

Figure 6. Avratar sound configuration.

5.4 Special Effects

Ogre 3D [11], the wrapper used to program Avratar, supports a plethora of special effects. Since Avratar's research focus is in digital puppet animation, it supports the cell-shader special effect shown in Figure 7. Cell-shading is a special effect that renders a 3D model to look like a cartoon, a very well-known application of this is the character "bender" from Futurama (in certain action scenes he is rendered via cell-shading to look like a cartoon, although there's really a 3D model powering the animation).

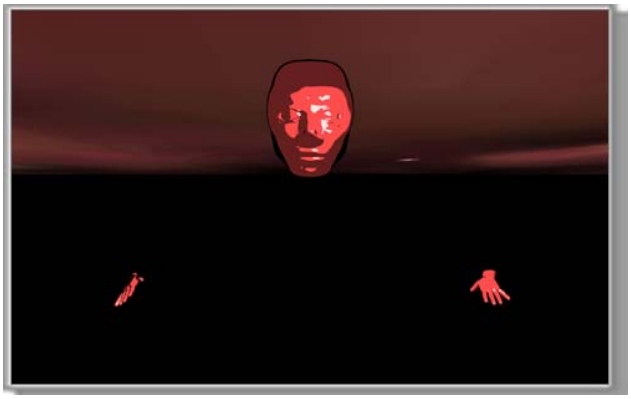

Figure 7. An Avratar puppet with cell-shading.

6 FUTURE WORK

Avratar is able to take input from an inexpensive, widely available peripheral, the P5 glove, and transform it into a tool as powerful as the user's imagination. Avratar supports models made in common editing studios, and some special effects.

In the future, we'd like to extend the special effects component of Avratar to include particle effects. Particle effects will provide Avratar with a new world of realism, things such as smoke, generated clouds, fog, sparks, and more will be supported. Particle effects are currently supported in Ogre 3D [11] (the 3D wrapper used by Avratar).

In addition, we'd like to extend the motion capture input devices supported by Avratar. For the first versions of Avratar it made sense to use the P5 glove for economy. However, digital artists and other users of Avratar might be able to make use of larger motion sensor arrays and more expensive equipment that are not yet supported by Avratar. A great application of this would be a full-body motion capture device [18]. This proposed extension would allow Avratar to capture input from a larger motion sensor array and support additional input peripherals.

7 CONCLUSIONS

Avratar is a real-time digital puppet animation studio that uses the inexpensive and widely available P5 glove [9] to control a real-time digital 3D puppet. Avratar has many applications in personal multimedia because it boasts an attractive combination of affordability and utility. Because of the focus on digital puppeteering, it's simple to use and users can create custom models on their favorite 3D editing platforms, then import them into Avratar. Planned enhancements for Avratar include adopting larger motion sensor equipment and a particle generation system.

REFERENCES

[1] Wired Magazine, *How Digital Animation Conquered Hollywood*,http://www.wired.com/wired/archive/14.03/animation.html, accessed May 1st, 2006.

[2] iClone website, http://www.reallusion.com/iclone/, accessed March 20th, 2007.

[3] George Latshaw. *The Complete Book of Puppetry*. Dover Publications, 2000.

[4] Richard Williams. *The Animator's Survival Kit*. Faber & Faber, 2002.

[5] Sommerville, I. Software Engineering, Addison-Wesley, 7th Ed., 2004.

[6] AutoDesk MotionBuilder website, *AutoDesk MotionBuilder*, http://usa.autodesk.com/adsk/servlet/index?id=6837710&siteID=123112, accessed March 20th, 2007.

[7] CrazyEddie GUI website *CrazyEddie GUI System*, http://www.cegui.org.uk/wiki/index.php/Main_Page/, accessed March 20th, 2007.

[8] OpenGL website, *The Industry-Standard for High Performance Graphics*, http://www.opengl.org/, accessed March 20th, 2007.

[9] Yahoo, *P5 Community Forum*, http://groups.yahoo.com/group/p5glove, accessed March 20rd, 2007.

[10] OpenAL website *OpenAL: Cross Platform 3D audio* http://www.openal.org/, accessed March 20th, 2007.

[11] Ogre 3D website *Ogre: Object-Oriented Graphics Rendering Engine* http://www.ogre3d.org/, accessed March 20th, 2007.

[12] Creative Inc., *Programmers Guide to OpenAL*. http://developer.creative.com/articles/article.asp?cat=1&sbcat=31&top=38&aid=51, accessed March 20th, 2007.

[13] 3D nuts. *Rigging Robotic Joints*. http://www.3dnuts.com/tutorials/robotrigging/rigging_robot_joints.shtml, accessed March 20th, 2007.

[14] Adi Bar-Lev, Alfred M. Bruckstein and Gershon Elber: *Virtual Marionettes: A System and Paradigm for Real-Time 3D Animation*, http://www.cs.technion.ac.il/users/wwwb/cgi-bin/tr-info.cgi?2004/CIS/CIS-2004-05, accessed March 20th, 2007.

[15] Siggraph, *Outline of the History of 3D Animation*, http://www.siggraph.org/education/materials/HyperGraph/animation/character_animation/motion_capture/history1.htm, accessed March 20th, 2007.

[16] Katherine Pullen and Christopher Bregler, *Motion Capture Animation*, http://mrl.nyu.edu/~bregler/papers375.pdf accessed March 20th, 2007.

[17] GameDev website *Cell Shading*, http://www.gamedev.net/reference/programming/features/celshading/, accessed March 20th, 2007.

[18] MetaMotion, *Full-Body Motion Capture Gypsy 5*, http://www.metamotion.com/gypsy/gypsy-motion-capture-system.htm, accessed March 20th, 2007.