Sam YEATES [1], **Eun-Jung HOLDEN** [2], **Robyn OWENS** [3]
[1 2 3] Department of Computer Science & Software Engineering,
The University of Western Australia, 35 Stirling Hwy, Crawley WA 6009 Australia
[1] samy@cs.uwa.edu.au
[2] eunjung@cs.uwa.edu.au
[3] robyn@cs.uwa.edu.au

REAL-TIME 3D GRAPHICS FOR HUMAN MODELLING AND TEACHING SIGN LANGUAGE

Abstract

The Auslan Tuition System provides a flexible, visual Australian Sign Language (Auslan) educational tool. In order to accurately represent sign language animations, we have designed a generic Human Modelling System. Implemented in cross-platform object-oriented C++, the Human Modelling System consists of three modules: the core Human Modelling Module for model construction, manipulation and forward kinematics; the Model Rendering Module for displaying model configuration visually using OpenGL; and the Model Interpolation Module, for providing flexible partial-keyframe interpolation and animation control. These modules form the basis of the Auslan Tuition System, and allow the display of categorised sign phrases, interactive fingerspelling and contextual sign dialogue examples. The user has complete viewing control of sign display and animation, running on modest hardware.

keywords: *HUMAN MODELLING, SIGN LANGUAGE ANIMATION, INTERACTIVE TUTORIAL*

1 INTRODUCTION

The Australian deaf community use a sign language called Auslan. Signers use a combination of techniques to communicate, including hand and arm position and orientation, motion of the hands, and facial expression. Auslan is different from other sign languages, including Signed English, although it is related to British sign language.

In addition to the unfamiliarity of gestural languages, the different grammmar structures of Auslan make learning difficult for native English speakers. The best sign language learning environment is one in which students can observe signs, and be given immediate feedback on their sign learning progress. This is the method used by an Auslan teacher. While sign reference books contain large vocabularies, the inaccuracy inherent in representing motion in a single diagram can make reproducing signs difficult for sign learners. To overcome this limitation, pre-recorded analogue video, or digitised movies, provided on CD-ROM or streamed online, are viable alternatives [1]. However, stored digital video sequences require a large amount of space, and while each sign is stored with its complete motion, signs cannot be combined and smoothly animated to form phrases.

We have developed an Auslan Tuition System with the aim of addressing these issues. Built to run on domestic PCs, it stores signs as parametric data, allowing run-time generation of sign movement through interpolation and display of computer graphics primitives. A static pose is represented by kinematic joint angles of the human upper body. An animated sign is then generated by sequencing a set of key framed poses. The system displays signs by interpolating joint angles, calculating modelled joint positions using robot kinematics, and rendering the resulting 3D human model in real-time. This technique allows smooth animation between and within stored signs, as well as more flexible control of the 3D model display.

The interface contains fingerspelling and Auslan number tutorials, sets of categorised sign phrases, and examples of Auslan dialogue showing sign language in context. It allows some degree of interaction where the user can input English text for fingerspelling translation, or input a number for signing, and provides full control of sign playback progress and viewing angle.

Signs are created using a sign editing interface where body segments are graphically manipulated, and then stored in a parametric animation format. This technique allows for quick visual sign creation, and for simple modification of current signs and addition of signs to the system.

In this paper, we present our general Human Modelling System, and detail its core forward kinematic, rendering and interpolation modules. We then describe the Auslan Tuition System built upon our Human Modelling System. We conclude by giving a summary of proposed additions and ongoing development in this area.

2 HUMAN MODELLING SYSTEM

The Human Modelling System uses a parametric human animation technique to display 3D graphics of human upper body motion. While primarily designed to display sign language, the system is general enough to be used for further research into human modelling.

The Human Modelling System consists of three functional modules. The Human Modelling Module is the core component of the system. It provides for input and output of XML definitions describing the hierarchical object being modelled, modifications of model joints within the subsequently generated internal kinematic tree, and forward kinematic calculations to determine global joint positions and rotations. The Rendering Module displays the position of joints of the above kinematic tree graphically, while the Model Interpolation Module provides for input, output and high-level control of animations between a set of model keyframes. Thus the system is flexible and easily extended as modules can be modified, or replaced entirely with minimal effect on the rest of the system.

The system is implemented in platform-independent C++, being source-code compatible on both Linux/Unix and WIN32 architectures. Making use of object-oriented design principles, many components of the system are designed to allow for custom extensions, by means of subclassing existing components to provide further functionality.

2.1 Human Modelling Module

The human body can be simplified and modelled as a purely rotational hierarchical kinematic tree. To calculate the positions of joints in 3D space for further analysis or rendering (forward kinematics), joints are individually modelled by storing a fixed positional offset from the immediate parent joint and a variable joint rotation. A full kinematic tree is constructed by linking joints according to a defined topology. The topology used to animate upper body sign language in our system has 39 nodes.

The Human Modelling Module parses a HumanModel XML file, and builds a corresponding kinematic tree by creating and linking node objects. The HumanModel XML file contains a number of XML Node elements, each of which contains information about joint offsets, initial joint rotations and joint linkages. Using object-oriented design principles, custom node subclasses can provide enhanced functionality within the forward kinematic and positioning system. The Node element *type* attribute is used to specify a specific node subclass to instantiate. Each node subclass must store joint rotation and translation parameters, as well as implement functions to position, calculate forward kinematics, and input/output XML, allowing custom object variables and state to be written to or read from HumanModel XML files.

A human upper body pose is represented by a set of joint angles, which are stored in a HumanPosition XML file. After constructing a kinematic tree by importing a HumanModel XML file, the body pose can be manipulated by changing individual joint angles. These angles can be saved into, and later re-applied from a HumanPosition XML file. Forward kinematic processing operates recursively down the tree, and can be initiated from any node. Updates to global rotation and translation of each node are stored as object variables within each node.

2.2 Rendering Module

The Rendering Module provides routines to display a stored human pose graphically. Each node within the kinematic tree structure is associated with a polygonal model representing the surface of the body segment directly following the joint. The node model can then be rendered by transforming it to the node's local coordinate space, and rendering it. The entire tree is rendered using a depth-first traversal from the root node, and rendering each node encountered as shown in figure 1.

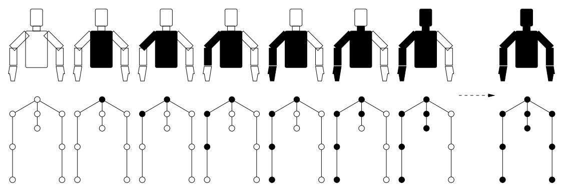

Fig. 1. Rendering the human upper body model. The Rendering Module renders the human upper body model by depth-first traversal of the kinematic tree. When a node is encountered, its associated polygonal model is displayed on the screen. Once the traversal is complete, the entire model has been rendered.

The Rendering Module uses OpenGL calls to render polygonal models, and provides other features such as joint click selection detection. A custom node rendering callback can be passed to the rendering method, which allows effects such as node highlighting, and partial node transparency.

2.3 Model Interpolation Module

To achieve understandable motion, smooth changes from one defined body pose to the next are required. The Model Interpolation Module provides interpolation, keyframing and animation control features for the Human Modelling System.

In previous work, we have focussed on determining the effectiveness of both euler and quaternion interpolation methods [2]. Representing rotations using euler angles has problems associated with gimbal lock, and issues relating to the large number of joint rotation order representations. In terms of interpolation, there is no simple euler space interpolation along the shortest path between two rotations. By using quaternions these problems can be avoided, and shortest-path interpolation between rotations can be performed using spherical linear interpolation (SLERP).

While more advanced interpolation methods involving quaternion splines [3] can provide smoother animation, we have found that simple SLERP interpolation, combined with careful generation of keyframes, provides sufficiently smooth and natural movement. The direct analytic nature of the simple interpolation involved ensures that processing times are much quicker than more sophisticated interpolation methods, and ensures that a greater range of interactive features is possible on modest hardware. The design of this module allows the use of custom interpolation code with other keyframe sequencing commands.

Partial Keyframing

The Model Interpolation Module provides partial-keyframe animation. The system reads and writes keyframe information from KeyframeSequence XML files. Standard simple computer keyframing techniques mirror traditional hand-drawn keyframing — that is, the position of the entire model must be specified for each keyframe. Partial keyframing removes the need to define superfluous keyframes, by maintaining a list of joint positions and times that the joint positions change, rather than storing a sequence of whole model keyframes. Each joint in a keyframe sequence must have a minimum of two positions, however joints need not have the same number of keyframed positions, and these positions need not be specified at the same times for each joint. This flexible technique allows for more powerful sign creation, as fine movements of one part of the body do not force definition of intermediate positions for joints that have movement over a greater time. An example for the sign “welcome” is illustrated in figure 2.

An additional benefit of partial keyframing is the ability to merge independent adjustment sequences with main sequences. Our system uses this technique to animate a questioning look, by moving the head and neck slightly towards the end of a sign animation. In this instance, the imported adjustment sequence only has keyframes defined for the head and neck, so that the main sign animation (arms and hands) can continue unaltered. This technique also reduces the number of sequences that must be defined, especially where there are only minor alterations between different signs, for example in Auslan numbers, where arm movements are the same for different values in the tens, hundreds or thousands, and the handshape alone specifies the specific number within each range.

3 AUSLAN TUITION SYSTEM

The Auslan Tuition System uses the Human Modelling System to render and animate sign language. The tuition system aims to provide an interactive, visual environment targeted to the learning needs of new Auslan signers, improving on traditional sign language references by providing both a better sign representation, and a more interactive educational experience.

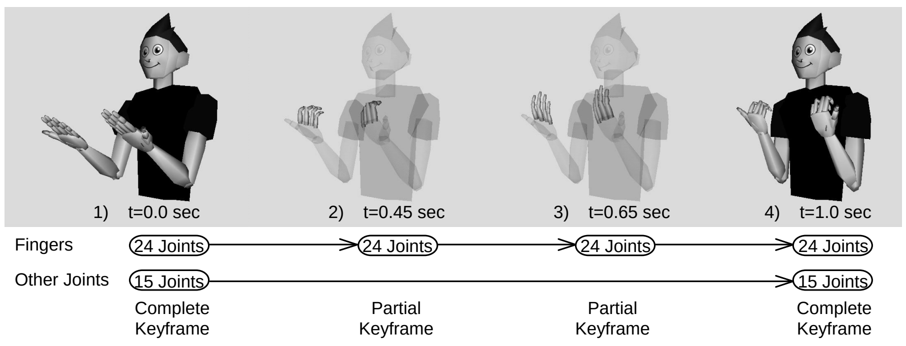

Fig. 2. The Auslan sign for welcome is a familiar gesture that is also used by the wider community, often to say "come in". Partial keyframing allows more effective sequencing of this sign, and other similar signs, as the smooth arm motion does not have to be interrupted when sequencing the more intricate finger motion. In this diagram, stored joints are drawn normally, while interpolated joint positions are rendered as semi-transparent.

There are several components of sign language that must be effectively communicated in order for a learner to understand and replicate a previewed sign. Signs comprise handshape, location, orientation, movement, and facial expression. Of primary importance is the shape made by the arms and hands. Sign handshapes are applied to hands depending on whether signs are one handed, double handed or two handed. In one handed signs, only the primary hand moves and has an associated handshape. Double handed signs use the same handshape on both hands, whereas two handed signs have two different handshapes. There are 38 major handshapes in Auslan, with a further 28 variants. Location of the hands is an important feature, and includes positions of contact between the hands and body, and between the two hands, while orientation refers to the direction in which the palms and hands point. These shape attributes of a sign must be clearly visible to the user. Movement is also a key aspect of sign language — movements must appear natural, and be clear enough to allow understanding and replication by the user. The Auslan Tuition System is able to effectively demonstrate signs with these attributes, but can not yet display facial expression.

Targeted Educational Content

Sign language literature and systems can be roughly categorised as either educational or reference material. While reference tools provide detailed sign information, they do not effectively target important signs or sign phrases for beginning signers to learn, or encourage step-wise exploration of sign content. Traditional education tools may provide targeted content, but rarely provide detailed sign information or a sufficient visual representation of signs. Our sign tutorial section combines signs to form phases, and groups these phrases into progressive categories. The interface also highlights the current word being signed, and provides links to detailed information about the individual signs. Figure 3(a) shows a sign being animated from a sign list.

User Control & Interactive Features

Both position and movement are necessary to correctly produce signs. Familiar "VCR-style" playback controls, as well as simulated frame-by-frame positioning of the animation, allow specific sections of signs to be examined more closely.

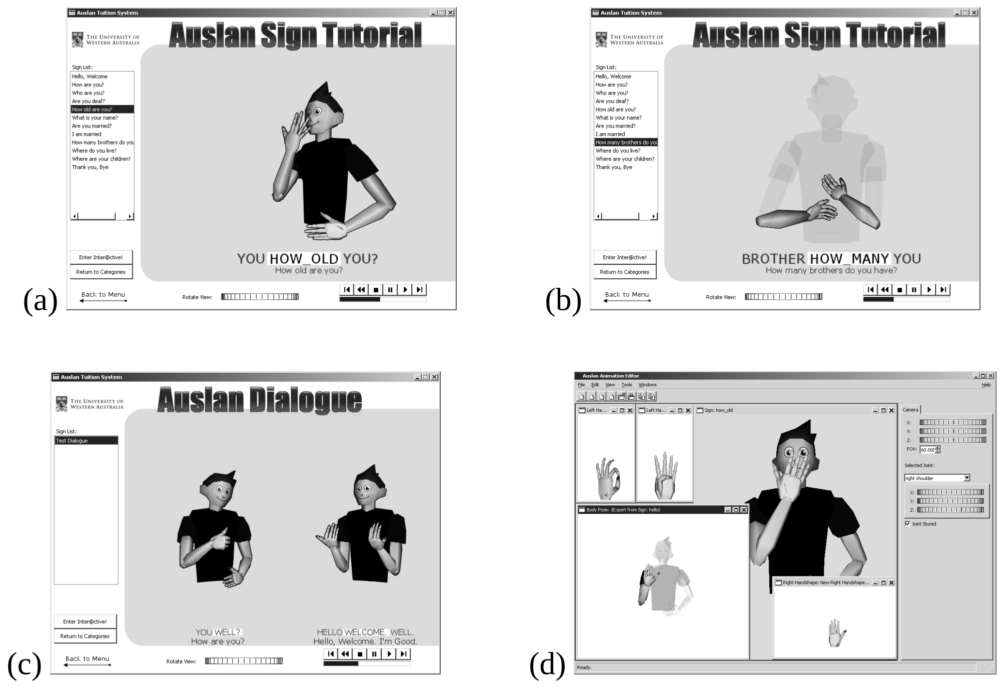

Fig. 3. Screenshots of Tuition System features. Figure 3(a) shows an animation of a sign from a sign category list. Figure 3(b) shows a sign being animated. The playback controls, visible on the lower-right corner of the screenshot, allow control of sign progress, while the view roller has been used to change the viewpoint so that animation is seen through the back of the model. Figure 3(c) shows a contextual Auslan conversation between two animated models. Figure 3(d) shows the sign editor application used to create signs.

Traditional sign references show a sign being produced “front on” as if watching a third party demonstrate a sign. Our system allows the user to rotate the displayed model, so that the figure can be viewed from different angles. When facing away from the user, the back and upper arms of the model become semi-transparent. This first-person view, which is also used by parents to teach deaf children how to perform sign language, provides a beneficial educational alternative for users of the system. Figure 3(b) shows a screenshot of a sign being animated from this first-person view.

The flexible rendering and interpolation system used to display signs has allowed us to easily add interactive features, which enhance the learning experience of the user. Fingerspelling is used for proper nouns, where there is no translation for an English word, or where the sign for an English word is not known. Our system is able to render an English phrase input by the user as fingerspelled Auslan. The system can also sign any number between 1 and 9999.

Displaying Sign Language in Context

Similarly to spoken languages, Auslan usage changes depending on conversation context. We have created an Auslan dialogue section, that allows the user to view an Auslan conversation between two rendered signers, showing the animated signs, sign names, and English meanings concurrently. This interface helps to highlight situations where signs presented in the Tutorial section would differ in the context of a conversation. A screenshot of this feature is given in figure 3(c).

Modification of Tutorial Contents

To build signs for the Auslan Tuition System, we have designed a Sign Editing Interface. The interface allows the user to create static poses, by visually selecting joints of the model to rotate, then using euler angle modification controls to position them. A screenshot of the sign editing interface is shown in figure 3(d). The system provides pre-defined positions for the major Auslan handshapes as a starting point for creating poses.

Static poses can be saved, or sequenced into sign animations. Speed of the animation is controlled by changing the timing of sequenced keyframes. Signs are saved to disk as KeyframeSequence XML files.

The Tutorial System loads and parses SignCategory and SignDialogue XML files to populate sign tutorial and sign dialogue categories and sign lists. These XML files store English descriptions, and sign control strings that set which signs to load at specific times. A simple parser reads, loads and sequences signs, loading sign keyframes directly from KeyframeSequence XML files stored within the working folder of the Auslan Tutorial System executable. Adding signs to the system is therefore a simple exercise — first, the sign is created using the sign editor, and saved into the correct directory on the disk, then the corresponding SignCategory or SignDialogue XML file is modified to include a phrase with the newly created sign. New categories of signs can also be added by modifying the SignCategory XML file.

4 SUMMARY & ON-GOING DEVELOPMENT

We have presented both the generic Human Modelling System, and an application based on the modelling system, the Auslan Tuition System.

The Human Modelling System consists of three modules: the Human Modelling module, the Node Rendering module and the Model Interpolation module. These form a base implementation of functions useful for research into human modelling, or other applications involving forward kinematics and display of fully rotational kinematic trees.

Combining the features of these modules, the Auslan Tuition System provides an interactive, flexible and extensible interface for learning sign language. The system offers flexible sign display and allows simple editing and addition to stored signs.

An ongoing task is to improve the vocabulary of the system. Other possible improvements include adding display of facial expressions and lip movement, employing smarter interpolation schemes including collision detection/ obstacle avoidance, changing the rendering of the system so that the model is drawn with a smooth mesh, and including a parser designed to automatically translate English grammar to Auslan.

Acknowledgement We would like to thank Nick Lowe and James Strauss for their involvement, and thank Prof. Michael Arbib and the West Australian Deaf Society for their input. This work is supported by the Australian Research Council.

REFERENCES

[1] T. A. Johnston, editor. *Signs of Australia: A new dictionary of Auslan (the sign language of the Australian Deaf community)*. North Rocks Press, 2nd edition, 1998.

[2] N. Lowe, J. Strauss, S. Yeates, and E. J. Holden. Auslan jam: A graphical sign language display system. In *Proceedings of the 6th Annual Conference on Digital Image Computing Techniques and Applications*, pages 98–103, 2002.

[3] R. Ramamoorthi and A. H. Barr. Fast construction of accurate quaternion splines. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, pages 287–292, 1997.