CU Animate

Tools for Enabling Conversational Interaction with Animated Characters

Jiyong Ma, Jie Yan and Ronald Cole

Center for Spoken Language Research
University of Colorado at Boulder, Campus Box 594
Boulder, Colorado 80309-0594, USA, http://cslr.colorado.edu
{jiyong, jie, cole}@cslr.colorado.edu

ABSTRACT

In this paper, we describe CU Animate: a set of software tools for researching full-bodied three-dimensional animated characters, and for controlling and rendering them in real time. Presently, eight complete characters are included with the system. Each character has a fully articulated skeleton, a set of viseme targets for the phonemes of English, a tongue that moves to target states for each phoneme, and a coarticulation model that controls the movements of the articulators between phonemes during speech production. A set of authored tools enables designers to create arbitrary animation sequences. A text markup language enables authors to control facial expressions and gestures during dialogue interaction and narration of text. CU Animate has been integrated into the Galaxy architecture within the CU Communicator system, enabling mixed-initiative conversational interaction with animated characters.

1. INTRODUCTION

The objective of our research is to enable animated computer characters to engage in natural face-to-face conversational interaction with users. To achieve this goal, it is necessary to synthesize accurate visible speech. But to produce the desired experience, it is also important to synthesize realistic, graceful and contextually appropriate facial expressions, eye movements, and head, hand and body movements while the character is speaking, and also while the character is listening to the user.

Our research is being conducted within an ambitious project called the Colorado Literacy Tutor, funded by grants from the Department of Education, NSF and NIH, which aims to provide powerful learning tools in which students converse with intelligent animated agents that behave much like effective teachers. The animated agents described in this article are being incorporated into interactive books—multimedia learning environments for helping children learn to read and acquire knowledge through reading. To enable conversational interaction with animated characters within these books, CU Animate has been integrated into the CU Communicator advanced dialogue system, which uses the Galaxy hub-server architecture. This architecture enables character animation to be integrated into spoken dialogue interaction through communication among CU Communicator’s technology servers—speech recognition, natural language understanding, language generation, speech synthesis and dialogue management. Our eventual goal is to enable teachers and students to develop their own animated productions that include spoken dialogues with animated characters. Thus, CU Animate is designed to enable easy control of embodied animated agents and their behaviors in interactive books. Teachers and students in Boulder Colorado are now testing interactive books incorporating animated characters such as those shown in Fig 1.

In the remainder of this article, we describe the key components of CU Animate. These include facial animation; hand gesture animation; incorporating animation into virtual environments; authoring tools for creating animation sequences; and tools for marking up text to control animation during speech production. We conclude with a description of plans to distribute CU Animate as a freely available toolkit for animation research.

2. FACIAL ANIMATION

2.1. Visible speech

Visible speech refers to the movements of the lips, tongue and lower face during speech production by animated agents. Visible speech in CU Animate is produced by morphing between viseme targets. Sixteen visemes were designed for each character. Each viseme is represented by a particular configuration of the lips, tongue and lower face for a set of phonemes with similar visual outcomes. For each phoneme, a tongue target was designed. Tongue posture control consists of 24 parameters manipulated by sliders in a dialog box. One 3d tongue model is shown in Fig 2a and Fig 2b.

Coarticulation [1] is modeled through a combination of rules and smoothing across prior and subsequent phonemes. A few heuristics have been incorporated into the rendering scheme to improve the visible speech; e.g., a rule requires targets to be reached at the onsets of stop consonants; this rule guarantees that a closure is seen before the release of a plosive consonant; smoothing across phonemes could prevent the closure target from being reached at faster speech rates. Once the rules have been applied, a time-varying linear low pass filter is applied to the preceding five phonemes and succeeding five phonemes. While the current system produces natural-looking visible speech, it is not accurate enough for use in some language training tasks. Research is now underway to develop more accurate visible speech.
The system uses the audio channel as the synchronization clock. The audio module sends a signal to the visual module, ensuring that audio and visual speech is synchronized.

Figure 2: (a) Side view of a 3d tongue model (b) Top view of the 3d tongue model (c) Eight Facial Zones

2.2. Facial expression

Realistically animating different types of facial expressions is an extremely challenging task. The face is a complex collection of muscles that pull and stretch the skin in a variety of ways. To generate realistic facial expressions, we need to understand the underlying anatomy of the human face and how muscle movements affect non-verbal behaviors. While Ekman and Friesen [2] identified six universal facial expressions for expressing sadness, anger, joy, fear, disgust and surprise, people actually produce thousands of different expressions [2].

To generate a large number of facial expressions, we enable independent control of separate facial components, as shown in Fig. 3. These include left and right eyebrows; left, right, up and down eyeball movements, up and down eyelid movement, nose size and mouth shape. In order to realize independent control of these components in CU Animate, we first designed thirty-six facial expression morph targets for each of the eight characters. Among the thirty-six facial expressions, the six “universal” facial expressions were designed based on optical analyses of movies of these expressions done at CMU [3]. We then enabled separate control of individual facial components within the 3d geometry of the models. Finally, we designed a user interface for manipulating these separate components.

The separated facial components include: left/right eyebrow; left/right/up/down eyeball; up/down eyelid; nose size, mouth shape, etc. To achieve precise control of facial features, we divided the face into 8 independent regions, as shown in Fig 2c. Independent interpolation parameters were used for each region.

2.3. Eye gesture

Eye movement patterns can be defined by the direction of gaze, the point or points of fixation, duration of eye contact and circular movement. The polygons associated with each eye, eyeball and eyebrow can be controlled independently, as with head movements. Eye blinks can accentuate linguistic content, as well as satisfy the biological need to lubricate the eyes. In general, there is at least one blink per utterance [4]. It is clear that with regard to this project, the ability to control the movement of the eyes is essential for added realism. The CU Animate markup language (CU AML) tags provide a subset of both head and eye movements to allow further realism in character animation.

2.4. Smoothing facial expressions

Smoothing algorithms are necessary to make transitions between facial expressions more natural. Three types of smoothing algorithms were designed to meet this requirement: (1) An “easy in or easy out” algorithm, used to control animation speeds at different times so that the animation is more realistic; (2) Kochanek-Bartels cubic splines, in which three parameters such as tension, continuity and bias are used to produce smooth motion; and (3) B-Splines.

3. BODY ANIMATION

CU Animate uses a parameter driven skeleton/bone model for the generation of lifelike gestures. Fig.4 shows the skeleton/bone structure [7]. The skeleton/bones are considered as rigid objects. Each bone is driven by the specific joints’ rotation parameters referenced by the three rotating axes. The
movement of the skeleton/bone is controlled by the pre-defined rotation parameters set for each joint.

... animation speed and route as a specific sequence of key frames. Here one key frame is described by one particular body movement.

3.1. Multi-level gesture description module

We have designed a description module that controls the behaviors and actions of virtual characters. The module is structured at multiple levels and is operated according to parameters. Our multi-level gesture description module has a three-level structure. The first level, the hand shape transcriber, is used to build the hand shape data. The second level, the sign transcriber, relies on the hand shape database and allows users to specify the location and motion of the two (left and right) arms. The third level, the animation transcriber, generates realistic animation sequences according to the target frames generated from the sign transcriber. Further details about each of the three modules and their corresponding user interfaces are described below.

3.1.1. The hand shape transcriber

The hand transcriber allows users to specify hand shapes. We provide both a low-level parameter control interface and a high-level trajectory control interface to create diverse hand gestures. By using these control interfaces, users can select one or more fingers and move them to a desired position. Slider bars specify the configuration of the selected fingers.

Low-level parameter controller: The parameter model in CU Animate is designed to drive the underlying kinematic skeleton of the character consistent with physiological considerations. The skeleton comprises 22 degrees of freedom (DOF) in 15 joints of each hand: (wrist (2); thumb (first joint (3), second joint (1)); index finger (first joint (2); second joint (1), third joint (1)); The index finger, middle finger, ring finger and little finger have the same structure. Here one slider bar is designed for one DOF of each joint, thus 15 slider bars are used to specify the position and orientation of the finger. The advantage of low-level parameter control is that it provides direct and precise manipulation of each finger joint. The disadvantage is that this requires specifying an excessive number of degrees of freedom in a coordinated way. The solution to this problem is now proposed.

High-level trajectory controller: In order to give the user an easier way to create hand gestures, a higher level controlling mechanism is needed. To this end, a set of commands are defined to describe one specific hand gestures using six trajectories: "spread", "bend", "hook", "separate" "yaw", "pitch". By using this high level concept, a motor control algorithm automatically performs an internal simulation of the hand structure that reflects the desired trajectory, and then translates the trajectory into low-level DOF parameters to drive the articulated hand models.

Hand shape library: To further increase authoring efficiency, we developed a primary hand shape library for users. Based on the ASL (America Sign Language) dictionary, a total of 20 basic hand shapes were selected for the primary hand shape library. By pre-storing some commonly used hand gestures in the hand shape library, the user’s editing efficiency will be improved greatly by using them directly or generating new gestures through some small modifications. The library was designed to be extensible. Users can create and add new hand shapes to the library very easily.

3.2. The body posture transcriber

The body posture transcriber is built on top of the hand shape transcriber. It allows users to specify the body posture in terms of hand shape, location and orientation for both hands and arms. The user can select the left/right hand shape from the hand shape library and then adjust the corresponding rotation parameters of the body components to create particular body postures. We provide an interface to edit the corresponding body components: pelvis, waist, neck, left clavicle and right clavicle. Here 3 DOF are defined in each joint of the above body components and slider bars are used to specify the particular positions and orientations. Fig. 5 shows some body posture examples. A library was provided to store the body postures.

3.3. The animation transcriber

In order to generate natural-looking body animation sequences, the animation transcriber enables the user to define the animation speed and route as a specific sequence of key frames. Here one key frame is described by one particular body
posture status (according to the particular rotation angles of each bone joint). Given several key frames of the body movement, a cubic splines-based interpolation algorithm [8] generates the animation sequence according to the hierarchical structure of the body. By using this algorithm, the cubic splines will provide a cubic interpolation between each pair of key frames with varying properties (described as Tension, derivation, continuities) specified at the endpoints. To generate one body movement sequence, the user is required to specify the total number of frames and provide the corresponding key frames of the body movement interactively. In CU Animate system, we provide an interface to edit the total number of frames as well as each key frame to create the animation sequence. Similarly, a library is also provided to store the animation sequences. This library includes several commonly used animation sequences such as bowing, “thumbs up,” clapping and others. Users can easily make modifications to these sequences or create new ones.

4. VIRTUAL ENVIRONMENT

CU Animate provides tools to construct image-based virtual environments. By providing some scene effects, this tool makes it possible for users to create various virtual environments for the animated characters, such as a plain scene with solid background, a half transparent scene or even special effects such as fog, clouds, rain, or snow. All these features in the toolkit were designed independently so that users can build up complex virtual environments simply by choosing one effect or the combination of several effects. Fig. 6 shows two examples.

![Figure 6: CU Animate virtual environment](image)

5. MARKUP LANGUAGE

CU Animate Markup Language, CU-AML, provides application developers with easy-to-use yet flexible and powerful means to control all behaviors of animated characters by marking up text. For example, CU-AML enables designers to control facial expressions and gestures of animated characters while narrating text; during conversations between animated characters; in response to arbitrary user behaviors in learning tasks; and during conversational interaction with users in mixed-initiative dialogue systems. For example, in spoken dialogue interaction, user utterances that receive a low confidence score may produce a puzzled look by the animated agent while she scratches her head.

CU-AML tags follow a defined structure like HTML. They have a specific purpose and affect the input text in a predetermined manner. The text currently includes tags for controlling facial expressions and gaze control; eye blink control; eye movement control; head gesture control and hand gesture control. Many useful features of character animation are realized using the CU-AML, and additional features are developed as needed.

6. INTERFACE

Visible speech movements of CU Animate characters are synchronized automatically with either synthetic or natural recorded speech. Festival [5] has been integrated into CU Animate for both Spanish and English. Automatic phonetic alignment of recorded speech uses CSLR’s Sonic speech recognition system [6].

CU Animate was developed on a PC platform using Visual C++ and Open GL libraries. To enable CU Animate to be platform independent, APIs for JNI (Java Native Interface) were designed so that JNI can invoke the APIs designed with C++.

7. SUMMARY

CU Animate is a working system that controls 8 animated characters. Each character is able to make hundreds of emotions by parametric control. The real time rendering engine in the system animates the models. Both the characters and animation system have been designed for maximum flexibility and control. Once CU Animate has been tested and documented, it will be distributed to university researchers free of charge.

8. ACKNOWLEDGEMENTS

This work was supported in part by NSF CARE grant EIA-9996075; NSF ITR grant IIS-0086107, and Interagency Education Research Initiative Grant REC-0115419. The findings and opinions expressed in this article do not necessarily represent those of the granting agencies.

9. REFERENCES