XNAgent: Authoring Embodied Conversational Agents for Tutor-User Interfaces

Andrew M. Olney, Patrick Hays, & Whitney L. Cade

Institute for Intelligent Systems & Department of Psychology
365 Innovation Drive
Memphis, Tennessee 38152
{aolney,dphays,wlcade}@memphis.edu
http://iis.memphis.edu

Abstract. Embodied conversational agents are virtual characters that engage users in conversation with appropriate speech, gesture, and facial expression. The high cost of developing embodied conversational agents has led to a recent increase in open source agent platforms. In this paper, we present XNAgent, an open source platform for embodied conversational agents based on the XNA Framework. By leveraging the high-level class structure of the XNA Framework, XNAgent provides a compact implementation that is suitable both as a starting point for the development of a more advanced system and as a teaching tool for AI curricula. In this paper we describe how we created an embodied conversational agent in XNA using skeletal and morph animation, motion capture, and event-driven animation and how this process can facilitate the use of embodied conversational agents in the Generalized Intelligent Framework for Tutoring.

Keywords: XNA, ECA, GIFT, agent, HCI, conversation, interface, tutoring

1 Introduction

It is well known that we unconsciously and automatically interact with computers using social norms [1]. Embodied conversational agents (ECAs) capitalize on this phenomena as characters with human-like communicative capabilities. By doing so, ECAs leverage pointing, gestures, facial expressions, and voice to create a richer human-computer interface. As a result ECAs have been used in diverse AI applications, including education [2], where they form an important part of the tutor-user interface.

ECAs combine research in discourse, computer animation, speech synthesis, and emotion. Consequently ECA systems tend to be costly to build [3] As a result, in the past decade, a great deal of tutoring research has used closed-source platforms such as Microsoft Agent [4], adapted commercial/open source game engines [5], or low-level libraries like OpenGL [6]. These approaches present different types of challenges. Game engines usually have support for basic character animation but lack native lip-sync and fine animation control, and game engines come with a complex API with many features that may not be relevant for education research, e.g. bullet/explosion physics or first-person shooter perspective. Conversely low-level libraries have no similar irrelevant complexity but require designing the AI from the ground up. Given the challenges of both the game-engine and low-level routes, recent researchers have released open source platforms for ECA development [7, 8, 9, 10] based on either game engines or low-level libraries.

The design and development challenges described above for ECAs are manifest in the development of computer-based training environments and have recently been addressed by the Generalized Intelligent Framework for Tutoring Framework [11]. One of the design goals of the Generalized Intelligent Framework for Tutoring (GIFT) is to provide authoring capability for the creation of computer-based training components. One such component is the tutor-user interface, which in modern intelligent tutoring systems often uses an ECA. Accordingly, in this paper we present an open source solution to ECA development that meets the design goals of the GIFT Framework. Rather than use a game engine with its inherent complexities or a low-level library that requires a large investment of initial development, we present an ECA platform that combines the best of these using Microsoft's XNA framework [12]. By providing high-level libraries, a runtime environment for managed code (C#), free development tools, and extensive support in the form of code samples, official forums, and commercially available books at all levels, the XNA framework provides a solid foundation for ECA development. In this the following sections we describe how we implement the face and body of XNAgent using skeletal and morph animation via vertex shaders, motion capture, and event-driven animation. At each step the content creation pipeline is outlined to illustrate how XNAgent may be adapted to new AI contexts. We conclude by considering the design goals of the GIFT Framework and how they are addressed by XNAgent.

2 Face

The face of an ECA can be considered independently of the body in terms of speech, emotions, and facial expressions. The classic reference for facial expression is the Facial Action Coding System, which uses the anatomy of the face, primarily in terms of muscle groups, to define facial action units [13]. While it is certainly possible to create "virtual muscles" and animate with them, a number of other real-time approaches exist which give satisfactory results [14]. Perhaps the most well-known and widely used facial animation approach is morph target animation.

In morph target animation, a version of the head is created for each desired expression. For example, one version for smiling, frowning, or a "w" lip shape. Each of these shapes becomes a target for interpolation, a morph target. If two morph channels exist, e.g. a neutral face and a smiling face, the interpolation between them can be described by the distance between matching vertices across the two faces. In practice, this distance is often normalized as a weight such that a weight of 1 would push the neutral face all the way to happy. The advantage to using morph target animations is that each morph target can be carefully crafted to the correct expression, and then mixtures of morph targets can be used to create huge number of intermediate expressions, e.g. smiling while talking and blinking.

FaceGen Modeler, by Singular Inversions, is a popular software package for creating 3D faces that has been used in psychological research on gaze, facial expression, and attractiveness [15]. FaceGen Modeler contains a statistical model of the human face with approximately one hundred and fifty parameters to vary face shape and texture. Using FaceGen Modeler, a virtual infinite variety of human faces can be created by manipulating these parameters, and for a given custom face FaceGen Modeler can output thirty-nine morph targets including seven emotions and sixteen visemes (the visual correlates of phonemes used for lip-sync). XNAgent uses FaceGen Modeler output, so a correspondingly large variety of faces can be implemented in XNAgent.

Since XNA does not provide native support for morph targets, we have implemented them using vertex shaders. A shader is a program that runs directly on the graphics card. In XNA, shaders are written in High Level Shader Language that resembles the C programming language, and the shaders compile side by side with C#. To implement morph target animation, XNAgent's vertex shaders operate on each vertex on face and perform bilinear interpolation (interpolation on two axes). Thus there are three versions of the XNAgent head loaded at any particular time: a neutral head that was skinned with the body (see Section 3), a viseme head for the current viseme, and an emotion/expression head for the current emotion. It is possible to have more channels for additional morphing, and these are easily added if necessary.

XNAgent utilizes a dynamic, event-driven animation system for facial expressions. Three categories of facial animation are currently supported, including blinking, lip-sync via visemes, and facial expressions. Blinking is implemented using a model of blinking behavior in humans [16] in its own thread. Because the most salient feature of blinking is perhaps that the eyelids cover the eyes, XNAgent imitates blinking through texture animation rather than morph target animation. In texture animation the texture of the face is switched quickly with another version of the face. In the case of blinking the two textures are nearly identical except the blink texture's eyes are colored to match the surrounding skin, thus simulating closed eyes.

Lip-syncing through morph target animation is controlled by the agent's voice, i.e. a text-to-speech synthesizer. Some speech synthesizers generate lip-sync information during synthesis by producing visemes, the visual correlates of phonemes. Each viseme unit typically includes the current viseme and the viseme's duration. In a viseme event handler, XNAgent sets the current viseme morph target and its duration using these values. In the Update() loop, the viseme's time left is decremented by the elapsed time. In the Draw() loop, the viseme morph is expressed with a weight based on the remaining time left. Thus the lip sync remains true independently of the framerate speed of the computer running XNAgent and linearly interpolates between visemes.

Morphing expressions like emotions require a more flexible approach than viseme animations. For example, a smile can be a slow smile that peaks at a medium value, or a rapid smile that peaks at an extreme value. To capture these intuitions, our expression morph animation has parameters for rise, sustain, and decay times, with a maximum weight parameters that specifies what the maximal morph will be during the sustain phase. Currently these three phases are interpolated linearly.

3 Body

Non-facial movements, or gestures, appear to greatly differ from the face greatly in terms of communicative complexity, forming sign language in the extreme case. Our approach is therefore to model the entire body as a collection of joints, such that manipulating the values of these joints will cause the body to move. This common approach to animation is often called skeletal, or skinned animation [17].

In skinned animation a character "shell" is first created that represents a static character. An underlying skeletal structure is created for the shell with appropriate placement of joints and placed inside the shell. The shell and skeleton are then bound together such that a transformation on the underlying skeleton is mirrored in the shell; this result is known as a rigged model. Once a model is rigged, it may be animated by manipulating the skeleton and saving the resulting joint position data. Every saved movement creates a keyframe, and when these keyframes are played back at a rapid rate (e.g. 30 fps) the rigged model will carry out the animated action. Alternatively motion capture technologies can extrapolate joint position data from naturalistic human movement. In this case the resulting animation is still a keyframe animation.

In order to create a body for XNAgent, we used several software packages to form what is commonly known as a 3D authoring pipeline. At each stage of the pipeline there are multiple available techniques and software packages, making navigating this space a complex process. In brief, there are three major phases to creating a body with gestures, namely model creation, rigging, and animation. Model creation can be extremely difficult for non-artists without initial materials to work from. To facilitate the process of body creation, we used the face model generated by FaceGen Modeler together with the FaceGen Exporter to export the face model to the Daz Studio software package. This process seamlessly combines the face and body models and auto-rigs the body with a skeleton. Daz Studio allows for comparable customizations of the body (gender, size, shape) as FaceGen does for the face. In addition, Daz Studio comes with a variety clothing and accessory packs that can be applied to the body in a drag and drop manner. In effect, several hundred hours of 3D authoring can be accomplished by a novice in less than an hour.

In order to create realistic animations, we primarily used the low-cost iPi Desktop Motion Capture system from iPi Soft. The simplest camera configuration for this system uses the Microsoft Kinect camera. Once the motion capture has been recorded by iPi, it can be merged and edited using AutoDesk 3DS Max, where ultimately it is exported for XNA using the kw X-port plugin. A complete description of this process is beyond the space limitations of the current discussion, but a full tutorial, including software installer and step by step slides, is available from the corresponding author's website[9].

In order to achieve similar functionality to interpolating visemes, skinned animation clips require mechanisms for blending and mixing. Simply put, blending is end to end interpolation, like a DJ fading from one song to the next. Mixing breaks the animation into components and plays them simultaneously, like a DJ taking the beat from one song, vocals from another, and playing them together. Blending and mixing can be done simultaneously if clips are playing in different regions of the skeleton

[9] http://andrewmolney.name

while being blended with other clips in those same regions. XNAgent uses the Communist Animation Library [18] to perform blending and mixing. Currently in XNAgent the skeleton is divided into center, left side, right side, and head regions. These regions are used to represent the following tracks: idle, both arms, left arm, right arm, and head. Animations are assigned to tracks at design time and then played with weights according to what other animations are currently playing in their track. For example, the idle animation consists of motion capture of a person standing and slightly swaying. Some degree of the idle animation is always playing in all the tracks, but when other animations are played in those tracks they are played with a higher weight. Thus lower priority animations like idle will be superseded by higher priority animations in a relatively simple manner.

Animations are triggered in XNAgent by inserting animation tags into the text to speak, either dynamically or manually. When the TTS encounters the tag, it schedules the animation immediately. The mixing properties of the animation are specified in the tag to create new versions of animations, similar to morphing. For example, since the idle animation is always playing, it can be given more weight relative to an arm gesture to create a "beat" gesture [19]. Thus a normal full arm extension animation can be dampened arbitrarily using weighting, bringing the arm closer to the body with increasing weight. In addition, the speed of the animation clip can be modulated to control for the appropriate speed of the beat gesture, since beat gestures are often quick and fluid.

Although XNA has some level of built in support for skinned animations, combining skinned animations with morph target animations requires a custom vertex shader. In XNAgent there are two vertex shaders that operate separately on the head and body of the agent. The head shader applies morphing to calculate a new vertex position and then applies the transformation defined by skinning. This allows the head to be applying morph targets (e.g. speaking) while also nodding or shaking. The second vertex shader focuses strictly on the body and so does not require morphing.

4 Working with XNAgent

One of the most important aspects of any ECA is its ability to integrate into an AI application. Game engines typically don't support integration well and rather present a fullscreen interface for the game, as does XNA. Although text input and other user interface functions can be carried out inside XNA, they are difficult because XNA doesn't provide the native support commonly expected by GUI designers. For example, key presses in XNA are interpreted based on the framerate of the game, meaning that a normal keystroke will produce a double or triple production of letters or numbers. To address the integration issue, XNAgent provides an XNA environment inside a Windows form control. That means that adding XNAgent to an interface is as simple as selecting the XNAgent control from the Visual Studio toolbox and dropping it on a form. The primary method to call on the control is Speak(), which processes both text to speech and animation tags as described in previous sections. In summary, the process for using XNAgent is (1) create a 3D model using the authoring pipeline described above (2) import the model to XNAgent (3) call XNAgent from your application using the Speak() method. We have previously integrated XNAgent into the Guru intelligent tutoring system shown in Figure 1 and conducted a number of experiments [20].

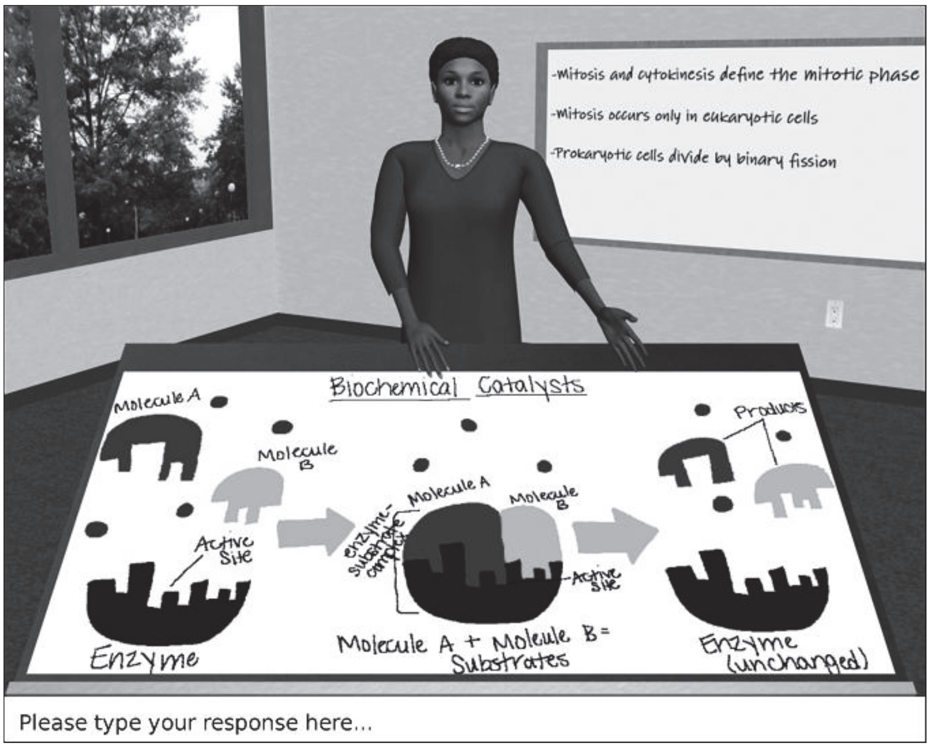
-Mitosis and cytokinesis define the mitotic phase
-Mitosis occurs only in eukaryotic cells
-Prokaryotic cells divide by binary fission
Biochemical Catalysts
Molecule A
Molecule B
Active Site
Enzyme
Enzyme-Substrate complex
Molecule A
Molecule B
Active Site
Molecule A + Molecule B = Substrates
Products
Enzyme (unchanged)
Please type your response here...

Figure 1: XNAgent running in the Guru intelligent tutoring system.

We argue that XNAgent fulfills many if not all of the design goals for GIFT authoring components [11]. XNAgent decreases the effort of authoring ECAs through its 3D authoring pipeline. Similarly it decreases the skills required for authoring ECAs by making authoring a drag-and-drop process, rather than a pixel-by-pixel process. XNAgent's animation framework allows authors to organize their knowledge about pedagogical animations and helps structure pedagogical animations. Perhaps most importantly in a research environment, XNAgent supports rapid prototyping of ECAs with different properties (gender, size, or clothing) for different pedagogical roles (teacher, mentor, or peer). XNAgent supports standards for easy integration with other software as a Windows form control. By cleanly separating domain-independent code from specific 3D model and animation content, XNAgent promotes reuse. Finally XNAgent leverages open source solutions. Not only is XNAgent open source, but every element in its 3D authoring pipeline either has a freeware version or is free for academic use. Moreover, the recent version of MonoGame, an open source implementation of XNA, promises to make XNAgent cross platform to desktop and mobile devices beyond the Windows desktop.

5 Conclusion

In this paper we have described the XNAgent platform for developing embodied conversational agents. Unlike existing ECA platforms that require either low level graphics programming or the use of complex game engines, XNAgent is written using a high level framework (XNA). Our contribution to this research area is in showing how to implement appropriate speech, gesture, and facial expression using skeletal and morph animation via vertex shaders, motion capture, and event-driven animation. We argue that the XNAgent platform fulfills most of the authoring design goals for GIFT with respect to authoring ECAs. It is our hope that XNAgent will be used by adopters of GIFT to facilitate creation of dialogue based tutoring systems that use ECAs.

6 Acknowledgments

The research reported here was supported by the Institute of Education Sciences, U.S. Department of Education, through Grant R305A080594 and by the National Science Foundation, through Grant BCS-0826825, to the University of Memphis. The opinions expressed are those of the authors and do not represent views of the Institute or the U.S. Department of Education or the National Science Foundation.

7 References

1. Nass, C., Steuer, J., Tauber, E.R.: Computers are social actors. In: Proceedings of the SIGCHI conference on Human factors in computing systems: celebrating interdependence. CHI '94, New York, NY, ACM (1994) 72–78
2. Dunsworth, Q., Atkinson, R.K.: Fostering multimedia learning of science: Exploring the role of an animated agent's image. Computers & Education 49(3) (November 2007) 677–690
3. Heloir, A., Kipp, M.: Real-time animation of interactive agents: Specification and realization. Applied Artificial Intelligence 24 (July 2010) 510–529
4. Graesser, A.C., Lu, S., Jackson, G.T., Mitchell, H., Ventura, M., Olney, A., Louwerse, M.M.: AutoTutor: A tutor with dialogue in natural language. Behavioral Research Methods, Instruments, and Computers 36 (2004) 180–193
5. Rowe, J.P., Mott, B.W., W. McQuiggan, S.W., Robison, J.L., Lee, S., Lester, J.C.: CRYSTAL ISLAND: A Narrative-Centered learning environment for eighth grade microbiology. In: Workshops Proceedings Volume 3: Intelligent Educational Games, Brighton, UK (July 2009) 11–20
6. Lester, J.C., Voerman, J.L., Towns, S.G., Callaway, C.B.: Deictic believability: Coordinating gesture, locomotion, and speech in lifelike pedagogical agents. Applied Artificial Intelligence 13 (1999) 383–414
7. Damian, I., Endrass, B., Bee, N., André, E.: A software framework for individualized agent behavior. In: Proceedings of the 10th international conference on Intelligent virtual agents. IVA'11, Berlin, Heidelberg, Springer-Verlag (2011) 437–438

8. Heloir, A., Kipp, M.: Embr — a realtime animation engine for interactive embodied agents. In: Proceedings of the 9th International Conference on Intelligent Virtual Agents. IVA ’09, Berlin, Heidelberg, Springer-Verlag (2009) 393–404
9. de Rosis, F., Pelachaud, C., Poggi, I., Carofiglio, V., De Carolis, B.: From Greta’s mind to her face: modelling the dynamics of affective states in a conversational embodied agent. International Journal of Human-Computer Studies 59(1-2) (2003) 81–118
10. Thiebaux, M., Marsella, S., Marshall, A.N., Kallmann, M.: SmartBody: behavior realization for embodied conversational agents. In: Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems - Volume 1, Estoril, Portugal, International Foundation for Autonomous Agents and Multiagent Systems (2008) 151–158
11. Sottilare, R.A., Brawner, K.W., Goldberg, B.S., Holden, H.K.: The generalized intelligent framework for tutoring (GIFT). Technical report, U.S. Army Research Laboratory â” Human Research & Engineering Directorate (ARL-HRED) (October 2012)
12. Cawood, S., McGee, P.: Microsoft XNA game studio creator’s guide. McGraw-Hill Prof Med/Tech (2009)
13. Ekman, P., Rosenberg, E.: What the face reveals: basic and applied studies of spontaneous expression using the facial action coding system FACS. Series in affective science. Oxford University Press (1997)
14. Noh, J., Neumann, U.: A survey of facial modeling and animation techniques. Technical Report 99-705, University of Southern California (1998)
15. N’Diaye, K., Sander, D., Vuilleumier, P.: Self-relevance processing in the human amygdala: gaze direction, facial expression, and emotion intensity. Emotion 9(6) (December 2009) 798–806
16. Pelachaud, C., Badler, N., Steedman, M.: Generating facial expressions for speech. Cognitive Science 20 (1996) 1–46
17. Gregory, J.: Game engine architecture. A K Peters Series. A K Peters (2009)
18. Alexandru-Cristian, P.: Communist animation library for xna 4.0 (December 2010)
19. McNeill, D.: Hand and mind: what gestures reveal about thought. University of Chicago Press (1992)
20. Olney, A., D’Mello, S., Person, N., Cade, W., Hays, P., Williams, C., Lehman, B., Graesser, A.: Guru: A computer tutor that models expert human tutors. In Cerri, S., Clancey, W., Papadourakis, G., Panourgia, K., eds.: Intelligent Tutoring Systems. Volume 7315 of Lecture Notes in Computer Science., Springer Berlin / Heidelberg (2012) 256–261

Authors

Andrew Olney is presently an assistant professor in the Department of Psychology at the University of Memphis and the associate director of the Institute for Intelligent Systems. His primary research interests are in natural language interfaces. Specific interests include vector space models, dialogue systems, grammar induction, robotics, and intelligent tutoring systems.

Patrick Hays is a research assistant at the University of Memphis. He is a recent graduate with a BA in Psychology. Patrick's work focuses on 3D animation, 3D modeling, graphic arts, and human-computer interaction.

Whitney Cade is a graduate student in the Department of Psychology at the University of Memphis. Her research interests include intelligent tutoring systems, expert tutors, pedagogical strategies, 3D agents, and machine learning.