Building Discount ECAs for the Study of Rapport

Ivan Gris, David Novick, Diego A. Rivera, Mario Gutierrez, Adriana Camacho, Alex Rayon
The University of Texas at El Paso
El Paso, TX 79968
(915) 747- 8959
ivangris4@gmail, novick@utep.edu, darivera2@miners.utep.edu,
mgutierrez19@miners.utep.edu, accamacho2@miners.utep.edu, amrayon2@miners.utep.edu

ABSTRACT
In this paper, we report our experience in using off-the-shelf software and hardware to build embodied conversational agents in immersive environments. Our underlying research focuses on paralinguistic factors in building rapport between humans and ECAs, and conducting this research requires ECAs that can (a) produce sophisticated full-body animation and (b) interact with humans across multiple sessions. Our approach to building these ECAS includes motion capture for defining animations, blending of animations to produce more realistic and complex movements, recognizing compound gestures, and defining and implementing an XML-based interpreter for human-ECA interaction scenes.

Categories and Subject Descriptors
I.2.11 [**Distributed Artificial Intelligence**]: Intelligent agents

General Terms
Design, Human Factors

Keywords
Full-body virtual agents

1. INTRODUCTION
Our research group studies human-agent dialog, with a particular focus on paralinguistics and building of rapport over time. For this work, we need fully functional embodied conversational agents (ECAs), created within the modest budget of a typical university research team, that can (a) incorporate sophisticated full-body animation and (b) interact with humans across multiple sessions. This paper reports how we created a platform for ECAs with greater naturalness and modifiability of animation than that provided by shared platforms, at a significant discount relative to commercial systems, and with lightweight scripting through which multiple interaction scenarios can be easily written and refined.

Using systems such as Unity and the Kinect, we were able to complete the platform after about 18 months of work—some focused, some trial-and-error. Today we are able to create new agents, which differ from each other in terms of looks, movement, domain, and functionality, in less than a month. Our ECAs have full-body naturalistic animation based on motion-capture and blending. New interactional scenarios can be developed and tested in very little time; about 20 scenes for our newest agent were developed in about three months. Because we use a Kinect sensor and a prosumer-grade game engine that has an accessible monthly subscription service, the marginal out-of-pocket expense of developing a new agent is less than $200.

In this paper we describe our approach for creating modular ECAs that provide a virtually unlimited set of animations, full-body gesture recognition, and lightweight scripting of scenarios. We briefly review other agents and platforms. We present how we create and process our animations, and how we address the challenge of creating movements for our agents that loop, blend, transition and even combine with each other to ensure that each gesture is unique and no animations are repeated. We present two examples of advanced intelligent agents capable of multimodal interaction that we developed using our platform. Finally, we conclude with limitations, future improvements, and potential applications of our system.

2. BACKGROUND
Researchers in the field of intelligent multimodal interaction have developed impressive ECAs that serve as aspirational models for our discount approach. Notable examples, among many, include:

- REA, a real estate agent who can help you choose a house that fits your needs and personal taste based on specific questions and chit-chat [1].
- Louise, a virtual nurse who talks patients though the hospital discharge process and assesses their understanding of medical instructions [2].
- Ada and Grace, responsive virtual-human museum guides [3].
- Sergeant Blackwell, a U.S. Army soldier who answers a wide range of questions, including open-ended questions [4].

Powerful platforms for development of ECAs are available (e.g., the Virtual Human Toolkit [5]), based on standards, such as FML and BML, and frameworks that integrate them, such as SAIBA [6]. A drawback of this approach is that, to produce an animation through BML, developers can control only specific, predefined body parts (head, torso, gaze, body, lips, legs, gesture and speech). In addition, a script needs to be created with all the rules and parameters per animation [7]. VHT provides semi-automated animation, such as for gestures in speech, but our underlying research on paralinguistics and rapport required animation that could be more naturalistic, more dynamic, and more parameterized (e.g., in terms of gesture amplitude). As a result, we developed our own (discount) platform for building ECAs.

3. ANIMATION
In our platform, animations for ECAs are developed with motion capture and combined with animation graphs.

3.1 Motion Capture
Motion capture involves recording and tracking the movements of a person and saving the movement files to be applied to a 3D model. This is usually an expensive process requiring a large studio, dozens of cameras and tracking suits, and expensive

software. This process is typically time consuming and often requires post-processing to remove undesired movements from the recorded animations.

We needed a much faster, much simpler approach to motion capture. Developing our agents' animations by hand would not be sufficient. First, creating animations manually requires a person with expertise in 3D modeling and animation programs, and the animations can take a long time to develop, especially if you need enough variety to prevent users from noticing repeated animations. Second, our research focuses on naturalistic movement, so we preferred our agents to express actual human movement rather than an animator's idea of this movement.

Our solution consists of using two Kinects, one placed in front and one to the side of the person modeling the movements. We record and process the movements using iPi Soft [8], a small, inexpensive applications that records the depth field from one or two Kinects. This approach is "markless," which means it does not require the actors to wear a suit or any type of sensors) and saves the animation files. Figure 1 shows rendering and pre-visualizations of movements we recorded.

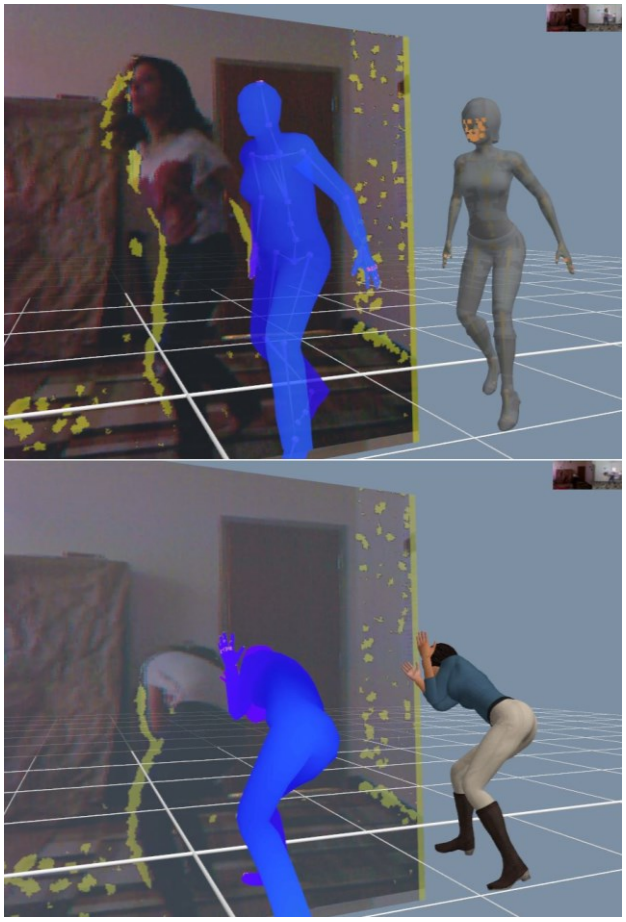

Figure 1. Rendering and pre-visualizations of recorded animations.

For some scenarios, our agent had to produce special movements and gestures, such as would be used in climbing a cliff. For these movements, we left the confines of our lab to go on-location, such as at the climbing wall of the university's Recreation and Fitness Center (see Figure 2).

Figure 2. One of the authors recording the climbing animation using the university's climbing-wall facilities.

Our approach does have two disadvantages. First, the animation requires significant processing time after being recorded. On average, it takes an hour to process ten seconds of animation. Second, the system is only capable of dealing with full-body motion capture under optimal conditions and continuous calibration. This is a process to which we have grown accustomed, but it means that we do not have motion capture for facial movement. As a result, we create our facial animations manually. But on the plus side, iPi Soft is 100 times less expensive than any other solution we could find, and it is good enough to get the job done.

3.2 Animation Graphs

Once the animations are recorded and available, we use Unity 4, as our game engine for displaying the agents and their environments. Mecanim is a system within Unity that enables us to create an almost unlimited array of animations.

Animations are played by a state graph that makes decisions depending on specific parameters. These parameters specify when an animation should start, end or blend with another animation. Multiple animations can be blended to obtain a completely different animation in real time and give the user the illusion that the agent never moves in exactly the same way twice.

Transitions between animations are also controlled by this system using a graphical representation of the states in the user interface, so animations can only transition between connected animation states.

Finally, animations are divided into layers that can control different parts of the body, so multiple animations can be played at the same time and affect different limbs of the agent. For example, the agent can have a blinking and talking animation on the face, an animation of explanatory gestures on the arms, and a walking animation from the hips down. The animations are played when Mecanim receives information about the specific animation to be played, the length of the dialog that the agent will say (only if applicable; the length is also necessary to activate the lip-sync function), and the position where the agent should be (in case there is any translation movement within the virtual scene, e.g. walking). Figure 3 shows Mecanim's graphical representation of the transitions between animation states; layers are listed on the top left and parameters on the bottom left.

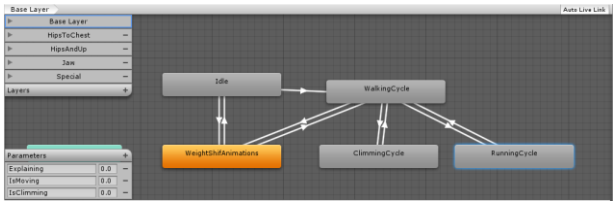

Figure 1. Graphical representation of the transitions between animation states.

By using our motion capture approach and combining and transitioning animations in Mecanim, we can create an almost limitless supply of unique animations that give users the impression that they are seeing unique, non-repeating movements.

These animations can be applied to any humanoid character. Figure 4 shows a Mecanim configuration example for one of our agents.

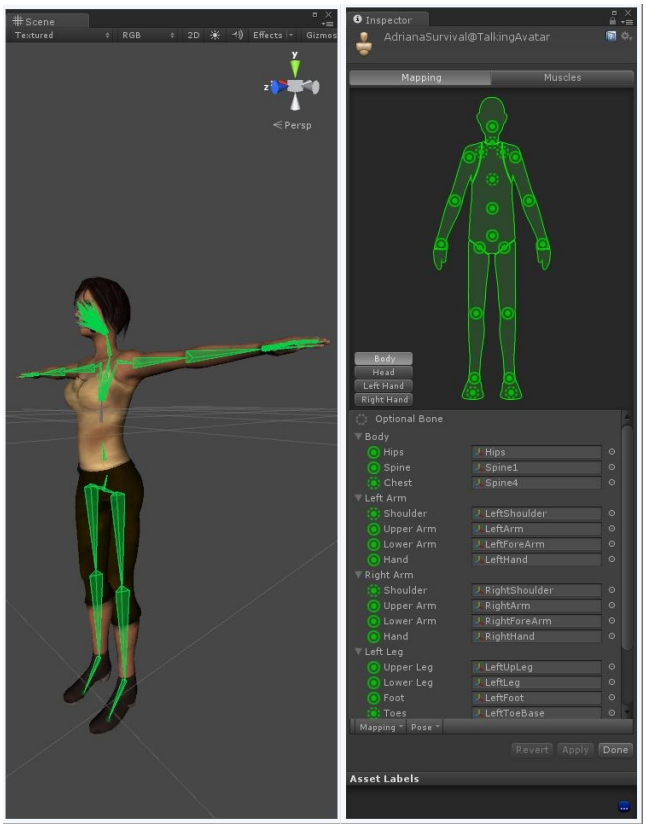

Figure 2. Mecanim configuration example.

4. GESTURE RECOGNITION

Part of our research in paralinguistics for ECAs involves tracking the similarity of gestures between human and ECA. As a result we needed a gesture-perception counterpart to our movement-expression animation.

In our approach, we recognize gesture using a Kinect and an application that we developed that tracks the 20 joints captured by the Kinect and calculates the angles between them. Our application then tracks the angles of the relevant joints for a gesture (e.g., wrists, elbows and shoulders in Figure 5). If the angles fall within the range of a predefined set of joint angles, we detect the pose in real time and message the agent to react accordingly if appropriate.

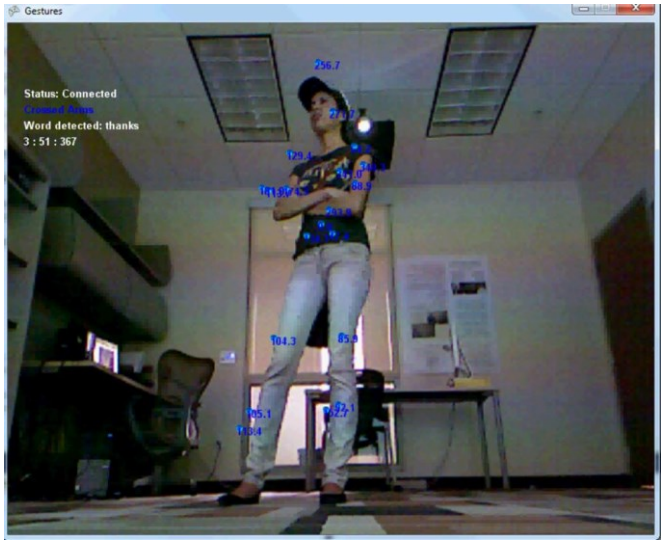

Figure 3. The gesture recognition system detecting the crossed-arms pose.

To generate our pose library, our system is able to take screenshots of subjects performing a particular pose. These record the position of all user-specified joints. Several subjects perform the same pose. When we have enough subjects, the system calculates the average angles and creates an appropriate margin of error. This can be integrated into a pose library, which enables the system to recognize immediately the newly added position in any person using the system. To avoid poses overlapping, such as detecting two different poses simultaneously in the same participant, we create a filter in the end application that permits the recognition of only the most probable or appropriate pose. This context-specific pose recognition is a gesture analogue for context-constrained recognition of speech; we reduce effective perplexity by limiting the number of items that can be recognized.

We have used this system as both a real-time interaction element and to partially annotate known behaviors. For poses or gestures in the pose library, the system can log each pose, record the video, and include timings [9].

5. LIGHTWEIGHT SCRIPTING

Because our research studies the development of human-ECA rapport over time, we have to produce multiple scenarios for each agent. Our agent currently under development, which interacts with a human in a jungle-survival story, is designed for periodic interaction over days or even weeks. The agent has about 20 individual scenarios, each about four to six minutes long. While our agents' scenes are refined through usability and playability testing (see [10]), development and revision of the scenes has to be lightweight.

Accordingly, we needed a system for authoring and interpreting interactions that would enable us to increase the number and length of the interactions with little or no coding. Toward this end, we developed a dialog interpreter that parses an XML document and links dialogs through conditions, similar to an if-then-else statement. The file also includes the human's responses expected at specific parts of the scenario. Figure 6 presents a brief example of the XML files for a scenario in which the agent asks the human about the weather.

SomeScene.xml	SomeSceneAnswers.xml
<Scene> Scene 1 </Scene> <Condition> It's a beautiful day, right? </Condition> [<Yes> But it looks like it'll rain later </Yes> <No> Yeah, those clouds will ruin it </No>] Let's get to the cave before it rains	[Yes,Yeah] [No,I don't think so]

Figure 4. XML file example for a simple question.

After the interpreter processes the file, it builds a dialog tree that represents the relationships of the dialogs through the scenario, enabling the system to follow the continuity of the story based on the human's verbal and physical responses. These responses are constrained to the context of the story that the agent and the user are experiencing. Figure 7 shows an abstract representation of the dialog tree for the scenario from Figure 6.

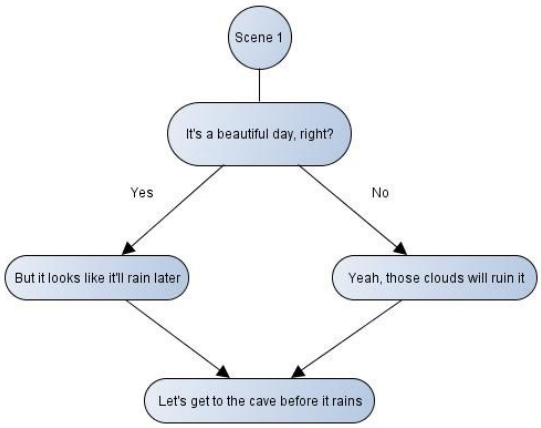

Figure 5. Simplest form of the dialog tree.

Dialog trees do not have to be divided into positive and negative answers. In fact, they can represent the paths for any condition. In the scenarios we are writing for the jungle-survival story, the interaction typically converges at an intended point. For example, the agent might ask the human if the human knows how to start a fire. If the human answers yes, then the agent asks the human to teach the agent how to start a fire; if the human answers no, then the agent teaches the human. Thus we designed our dialog representation so that the dialog tree can diverge or converge at certain states and can have a series of nested conditions.

6. CONCLUSION

We believe that our platform can be used to implement a series of full-body 3D agents for many different domains and purposes, without great expense. In particular, it can enable researchers to develop applications that make use of complex full-body gestures and that recognize relatively complex gestures. Our lightweight approach to authoring and interpreting dialog enables developers to prototype and refine interaction scenarios rapidly.

Our system requires expertise with the different software packages needed to deploy the agent. These packages include Autodesk Maya, the Unity3D game engine, and its Mecanim functions. Our approach also requires some basic knowledge of C# to code the activities and gestures. Our system lacks facial motion capture and facial gesture recognition, and it is not able tto track the movement of individual fingers.

Our future work is geared toward creating additional modules that enhance the behavioral capabilities of our agents, with a specific focus on the paralinguistics and relationship-building. We are currently working on a module that enables agents to remember simple elements of previous interactions. Agents will be able to refer to this information at appropriate times during the dialog, which will we expect to increase users' perceptions of rapport based on shared experience.

7. REFERENCES

[1] Cassell, J., Bickmore, T., Billinghurst, M., Campbell, L., Chang, K., Vilhjálmsson, H., & Yan, H. (1999, May). Embodiment in conversational interfaces: Rea. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems* (pp. 520-527). ACM.

[2] Bickmore, T. W., Pfeifer, L. M., & Jack, B. W. (2009, April). Taking the time to care: empowering low health literacy hospital patients with virtual nurse agents. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 1265-1274). ACM.

[3] Swartout, W., Traum, D., Artstein, R., Noren, D., Debevec, P., Bronnenkant, K., ... & White, K. (2010, January). Ada and Grace: Toward realistic and engaging virtual museum guides. In *Intelligent Virtual Agents* (pp. 286-300). Springer Berlin Heidelberg.

[4] Leuski, A., Patel, R., Traum, D., & Kennedy, B. (2009, July). Building effective question answering characters. In *Proceedings of the 7th SIGdial Workshop on Discourse and Dialogue* (pp. 18-27). Association for Computational Linguistics.

[5] Hartholt, A., Traum, D., Marsella, S. C., Shapiro, A., Stratou, G., Leuski, A., & Gratch, J. (2013, August). All together now: Introducing the virtual human toolkit. In *International Conference on Intelligent Virtual Humans* (Edinburgh, UK).

[6] Vilhjálmsson, H., Cantelmo, N., Cassell, J., Chafai, N. E., Kipp, M., Kopp, S., ... & Van Der Werf, R. J. (2007, January). The behavior markup language: Recent developments and challenges. In *Intelligent virtual agents* (pp. 99-111). Springer Berlin Heidelberg.

[7] Kopp, S., Krenn, B., Marsella, S., Marshall, A. N., Pelachaud, C., Pirker, H., & Vilhjálmsson, H. (2006, January). Towards a common framework for multimodal generation: The behavior markup language. In *Intelligent virtual agents* (pp. 205-217). Springer Berlin Heidelberg.

[8] http://ipisoft.com/.

[9] Gris, I., Novick, D., Gutierrez, M., & Rivera, D.A. (in press). The "Vampire King" (version 2) corpus. In *Proceedings of LREC 2014 Workshop on Multimodal Corpora*.

[10] Novick, D., Vicario, J., Santaella, B., Gris, I. (in press). Empirical analysis of playability vs. usability in a computer game. In *Proceedings of HCI International 2014*.