

Believable Performance Agents for Interactive Conversations

Nathan D. Nichols, Kristian J. Hammond, David A. Shamma, Sara H. Owsley

Intelligent Information Laboratory
Northwestern University
2133 Sheridan Road, Room 3-320
Evanston, Illinois 60208
+1 (847) 467-6924

{ndnichols, hammond, ayman, sowsley}@cs.northwestern.edu

Keywords

Network Arts, Emotion, Blogs, Media Arts, Culture, World Wide Web, Software Agents

1. INTRODUCTION

As computers become more integrated into our everyday lives, they will need to be able to interact with us within the context of our world as well as theirs. While it is unlikely that we will ever want all interactions with a computer to mimic dialogs with other people, it is clear that they will need to be able to engage in coherent, compelling conversations with people who are not thinking of them as machines. Our effort, and the system described below, is aimed at approaching this goal by creating a framework for believable performance agents within the content of interactive theatrical experiences.

While making scripted movies with digital actors has become more common—if still difficult—there has been much less progress made in performers that interact convincingly and dynamically in real time with an audience. To that end, we have developed DigImp, a platform for performance agents that can persuasively interact with an audience as a whole, individual audience members, and other human performers. In support of these goals, DigImp is designed to leverage information about the task at hand, the current local context, and the information available on the internet that can be used to support a broader context of human/machine interaction.

2. FRAMEWORK

DigImp builds on our previous work in Network Arts [2, 1] to create a full interactive on-stage production. With DigImp, we have a system that can welcome an audience, discuss current news stories with them, and react and respond appropriately to what it hears (including handling gracefully and believably situations where the speech recognizer has difficulty understanding the audience.) As



Figure 1: As a starting point, we created a Half-Life 2 mod for a basic stage set with an avatar actor (HL2's Alyx Vance). The actor was augmented with dynamic text to speech abilities using Microsoft SAPI and Neospeech voices to synchronize the spoken audio to the actor's visemes.

we progress with the project, our performers will be able to interact in new types of conversations and contexts as well as eventually performing as a full cast member in an improvisational theater setting. To do this, we have modified three relatively turnkey solutions. For graphics, we use the Source game engine, the same engine used for the recent game *Half-Life 2*. For speech recognition and speech generation, we use Microsoft Speech API 5.1. These commodity technologies were originally designed for different tasks than for which we are using them and so we have had to design our core, connecting system with these weaknesses in mind. Our system effectively ties together the strengths of these otherwise impressive technologies while mitigating their weaknesses.

The *Half-Life 2* engine is designed to provide clever super soldiers and aliens, not a friendly theater MC. Knowing these limitations, we pulled out the existing AI and replaced it with our own high-level scripting commands. To allow the avatar to gesture and express body language in a normal, humanlike way, we have implemented a basic markup language. With this language, the avatars are able to perform arbitrarily complicated expressions. These commands can be generated externally; we have a predefined “welcome the audience script” for example. They can also be generated internally; on an emphasized “you” for instance, our performer may decide to point at the audience. Together the speech and actions are coordinated under a simplified command and control language (SCML) that may be expressed remotely across a standard TCP/IP socket. Small macros can be defined in SCML to expose high-level commands like *LookAt*, *SayHello*, and *AppearPleased*. Furthermore, we can actually embed SCML calls in the text-to-speech stream to synchronize spoken words and physical gestures. With this amount of control over the performer’s bodily presence, she is able to reflect physically the same ideas she is communicating verbally. She can point at herself when discussing herself, for example, or smile genuinely when she senses the audience is pleased. As we continue to develop her physicality, we will have a system that autonomously uses gestures and body language to help convey its ideas.

Microsoft’s speech recognizer was intended to be a computerized note-taker, listening to and recording whatever is said by a voice for which the system is trained. However, we need it to respond to the raucous yelling of a crowd, so we create our own grammars algorithmically and do a little reasoning to make sense of the answers we receive from the audience. In an improvisational context, conversational abilities (such as getting a popular news topic suggestion from the audience) are essential. Fortunately, the task constraints of the improvisational theater allow for an untrained speech recognition system to perform well with the addition of a lightweight reasoner to interpret the often garbled responses of a shouting crowd. While the speech recognition system itself rarely recognizes many words accurately, we are able to use its basic output to externally reason about what is actually being said. In most cases, when we ask the audience a question, we know the types of answers we expect to hear. If our performer asks, “What’s something in the news that makes you mad?” she can safely expect to hear an answer like “Bush,” “Iraq,” or “taxes.” Our system is able to automatically parse multiple online sources such as news sites to create dynamic grammars of expected responses to her questions. Our speech recognition system is built with its limits in mind. Because we are trying to build performers, not perfect speech recognizers, our actors are not afraid of asking for clarification. She can also reframe questions if she is unable to get an intelligible response to a previous query; for example, if she cannot understand

any answers to “Have you seen any good movies lately?” she may simplify the question to “Has anybody seen *Crash*?” Even when her speech recognition fails, our performer comes across as a person, not a computer with disappointing speech recognition abilities.

The SAPI text-to-speech (TTS) engine is designed to provide strictly comprehensible speech. Since the voice has a strong tendency to drone on, we analyze the words the performer is speaking and apply the existing capabilities of the TTS engine in order to give our actors a more human-like cadence. For instance, we use a system originally developed for Buzz to find the highest-affect words in a sentence; we can then audibly emphasize these important words, turning “You expect me to do what?” into “You expect ME to do WHAT?” We are also developing a list of proper pronunciations for difficult-to-pronounce words and phrases; when reading from a blog, for example, she may pronounce the abbreviation “FL” as “Florida.” These convincing speech patterns, coupled with her automatic physical gestures, make for a more believable experience.

Finally, we also use a few other techniques to increase the credibility of the performer’s conversations. We use simple face-tracking software and a basic webcam to inform her of how many people she is talking to and where they’re physically located. This allows her to change speech autonomously (“you” can become “you all” when she’s speaking to a group), look directly at people she’s speaking to, and “wake up” when someone approaches her. We are also implementing a series of Basic Environmental Evaluators (BEEs) that allow her to have a low-level understanding of her environment. For example, because she knows the current time, she can greet people with “Good morning,” “Good afternoon,” or “Good evening” appropriately. Or, if she knows it’s unseasonably chilly out, she may thank the audience for venturing out into the cold to see her. This information may not change the core content of her interactions, but it does give her a more convincing air.

3. CONCLUSION

We are building DigImp and the SCML language to allow a performer to hold her own in an improvisational theater setting; her abilities will range from chatting with the audience before the show to participating in improv games as a full member of the cast during. Although her performances are limited currently, we have built a solid foundation for tracking and responding to conversations she is involved in as well as applying subtle physical, audio, and contextual details that make her performances more convincing and lifelike. As we continue to extend her interactive abilities, as well as building more high-level scripted conversations, we look forward to her first full show in a public theater.

4. REFERENCES

- [1] M. Ruberry, S. Owsley, D. A. Shamma, K. Hammond, J. Budzik, and C. Albrecht-Buehler. Affective behaviors for theatrical agents. In *Proceedings of Intelligent User Interfaces Workshop on Affective Interactions: The Computer in the Affective Loop*, San Diego, 2005.
- [2] D. A. Shamma, S. Owsley, K. J. Hammond, S. Bradshaw, and J. Budzik. Network Arts: Exposing cultural reality. In *Alternate track papers & posters of the 13th international conference on World Wide Web*, pages 41–47. ACM Press, 2004.