

MULTITALKER: BUILDING CONVERSATIONAL AGENTS IN SECOND LIFE USING BASILICA

BABA KOFI A. WEUSIJANA¹

*LIFE Sciences of Learning Center, University of Washington,
Seattle, WA 98195, USA*

ROHIT KUMAR²
CAROLYN P. ROSÉ

*Language Technologies Institute (LTI), Carnegie Mellon University,
Pittsburgh, PA 15213, USA*

Multi-User Virtual Environments (MUVes) such as Second Life can be great tools for educating people and researching how people learn. Utilizing conversational agents automated with artificial intelligence can further enhance MUVes. They can be used for tutoring students and they can be used to manage or consistently orchestrate environments for research purposes. However, most intelligent tutoring systems are designed for one-on-one conversations, whereas MUVes usually involve learners studying together in groups to leverage social aspects of learning. We have been working on conversational agents that can talk to learning groups in effective ways. Those at LTI have built the Basilica framework. Basilica is a system to facilitate rapid development of conversational agents that participate in collaborative environments. On top of Basilica, Dr. Weusijana of LIFE developed with LTI a Second Life conversational agent called MultiTalker. We consider our new technology an enabling tool for educators and learning researchers. We discuss its expected utility for educators and their learners, as well as future research directions.

1. Introduction

Utilizing artificially intelligent conversational agents can further enhance Multi-User Virtual Environments (MUVes) for learning. However, most intelligent tutoring systems are designed for one-on-one conversations, whereas MUVes usually involve learners studying together in groups to leverage social aspects of learning. We have developed a Second Life conversational agent called MultiTalker built on top of the Basilica framework [1] to facilitate rapid development of agents that chat with a student group. First, we discuss related work, from which we draw requirements for our own development effort. Next, we describe the Basilica framework itself. We then describe our experience of integrating Basilica with the Second Life MUVE in only one week's time, how we expect educators and students to utilize the system. We conclude with discussion and current directions.

2. Conversational Agents in MUVes

Conversational agents have already found their way into different Internet based environments for better or for worse. For example, chat rooms have become infested with marketing bots. At the same time, similar technology has helped automated administration of chat rooms.

Games can create a richer immersive experience for the player as DeSmedt and colleagues [2] suggest that non-player characters in video games can be given more personality by getting them to talk. Morris [3] describes a project to develop believable conversational agents in a game called Cluedo.

Besides web and video game based conversational agents, recently there has been interest in developing conversational agents in fully immersive virtual environments. Some of these environments are standalone. Usually these environments are built for specific purposes like education and training, entertainment, etc. The

¹ Work partially supported by the Learning in Informal and Formal Environments (LIFE) Sciences of Learning Center, a NSF funded Science of Learning Center that includes researchers from the University of Washington, Stanford University, and SRI International. See <http://www.life-slc.org> for more information.

² Work partially supported by the Office of Naval Research and the Pittsburgh Science of Learning Center (PSLC). PSLC is a NSF funded Science of Learning Center that includes researchers from Carnegie Mellon University and the University of Pittsburgh. See <http://www.learnlab.org> for more information.

Mission Game [4] shows the use of virtual conversational agents to engage a learner in a task based language learning situation. On the other side, however, are the persistent virtual environments, which are shared by several users. Second Life is an example of such a virtual environment. Novamenta's Cognitive Engine [5] renders intelligent behavior like conversation to characters (e.g. pets) in MUVes.

Agents in MUVes may play a variety of roles ranging from proactive participation in the environment to attract and engage users to responding to user requests or to observation of indicative behavior. In our view, agents are a collection of behavioral components and development of these can be decomposed into the development of individual behaviors and the establishment of the communication mechanisms between them. Instead of establishing a unified theory or a small set of theories of intelligence to facilitate all behavior, our current view calls for a heterogeneous mix of intelligent capabilities as demanded by the behavior, which can be mixed and matched. A framework that supports this decomposition can and should be designed to enable reuse of pre-developed behavioral components.

3. Basilica: A platform for building Conversational Agents

Our approach to developing intelligence adopts a decomposition stance. Each Intelligent behavior required for the development of a conversational agent is identified and developed as a separate component. We argue that this approach enables re-use of behavioral components. In addition, this facilitates integration by isolating the components required for the integration of the agent with external environments.

Basilica is an event-driven framework, which enables development of conversational agents by using two basic behavioral components, namely Actors and Filters. The components communicate using Events. The Actor component, as the name suggests displays behavior. Filters on the other hand observe behavior. Behavior and data are encapsulated into Events. For example, if an Actor component generates a text message to be shared by the other participants in the conversational interface, it broadcasts a `TextMessageEvent`. Similarly, if a human logs into the conversational interface, a `LogInEvent` is sent to all relevant filters. The Basilica framework implements a set of abstract software classes that correspond to components, events and other supporting elements of the framework such as channel independent communication, logging, and process management. Along with these abstract classes, the Basilica framework now has a growing set of reusable Actors, Filters and Events, which can be used to rapidly build custom conversational agents.

3.1. Design Motivations

The Basilica framework has been designed for professionals and students of software engineering and computer science who experiment with conversational interfaces. Considering this set of users, we compare with the existing set of technologies available for building conversational interfaces and we come across a plethora of scripting platforms each offering a set of basic operators, which can be combined to create a sequence of steps to be executed through a pre-defined engine. While this scripting approach enables a wide variety of users to build conversational systems, it lacks the versatility to enable our target users to create new behavior and modify existing behavior using their programming skills.

In order to retain the ease of the scripted approach, Basilica supports building wrapper components, i.e. actors and filters that can execute scripted behavior. Behavior created for the TuTalk dialog management system [6] can be incorporated into conversational agents built using the Basilica framework using the existing wrapper components. Dr. Rosé and others at the Pittsburgh Science of Learning Center (PSLC) have developed TuTalkAT, a graphical authoring tool for building dialog scripts. It may be observed that Basilica acts as a meta-framework for bringing together different frameworks that support different kinds of behavior without extensive re-implementation.

Supporting procedural behavior in an event-based framework like Basilica brings together two very different approaches of generating conversational behavior within the same framework. With this integration, we can view the conversational task as a space of graphs. Each graph represents a procedural behavior and graphs are triggered by events. The traversal of each graph once it is triggered is guided by the control strategy of the behavior associated with the graph. This has implications for the amount of effort involved in developing conversational applications of very different scales. In a procedural framework, the task would be represented by a single but potentially a much more complex graph. Using Basilica the complex graph can be divided into several smaller graphs, hence allowing distributed development with fewer dependencies and a greater possibility of reuse of behavior.

The event-driven approach adopted by Basilica has further advantages for building multi-modal conversational applications due to the need for different levels of synchrony for communication between peripherals and internal components.

3.2. Building Conversational Agents with Basilica

A Basilica component can be defined based on the events it observes and the operations it performs when it observes each of those events. These operations may include update of its beliefs, access to resources, and generation of events, among other things. Building a conversational agent under this framework is essentially an exercise in instantiating the desired actors and filters from a library of reusable components and establishing communication links, called Connections, between them. This exercise involves ensuring that all components are well connected to receive the events they expect. In some cases, the exercise may involve creating new actors and/or filters to generate or observe some new behavior.

4. Integrating Basilica with Second Life

In a recent development, we have built a conversational agent using our Basilica framework. This conversational agent can render conversational chat behavior to any object in Second Life. Further in this section, we discuss the design choices and final bridging of a Second Life object and the Basilica based conversational agent using a generic middleware component.

Figure 1. shows the Basilican graphical representation of the Second Life conversational agent. This agent displays two conversational behaviors in the current implementation. The PromptingActor prompts the user when the user touches the Second Life MUVE, which is considered to be a gesture of initiation.

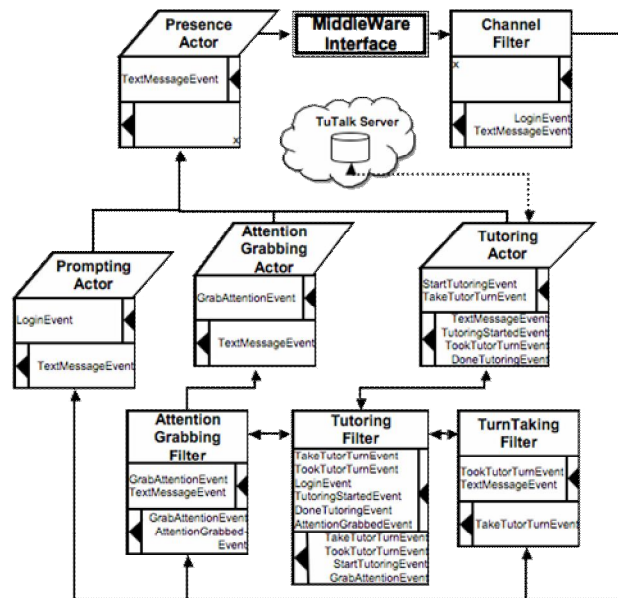


Figure 1: Graphical representation of the Second Life Conversational Agent

Before an instructional dialog begins, to grab the user's attention, the agent produces an AttentionGrabbing prompt. An example of an AttentionGrabbing prompt is "Now might be a good time for some reflection". The strategy to produce an AttentionGrabbing prompt is motivated from our experience with building conversational support in a collaborative learning environment [1]. Students tend to not notice the agent's instructional turn in between their turns and the tutor never gains the floor in the conversation. We think that an AttentionGrabbing prompt may be successful at getting the tutor the floor in a text based conversational environment.

The agent back-end discussed above communicates with its front-end in Second Life through a middleware component. The middleware is essentially a message-exchanger implemented as a HTTP servlet. Both the agent front-end and back-end implements an interface to communicate with the middleware. The

middleware performs the functions of session establishment, session maintenance, and reception, translation, and delivery of messages between objects connected in a session. In the current implementation only two objects can be connected in one session. However, this can be easily generalized.

Figure 2. shows, the middleware component and its interaction with the second life object and the Basilica conversational agent. Whenever a new front-end object is instantiated by a user, the object connects to the middleware to start a new session. The new session indicates the conversational agent to spawn a new instance of itself and act as the second object in the session. Henceforth, whenever the middleware receives a message, it translates the message and queues it up in the other object's message queue. The object can check with the middleware if a message is waiting for it and receive it.

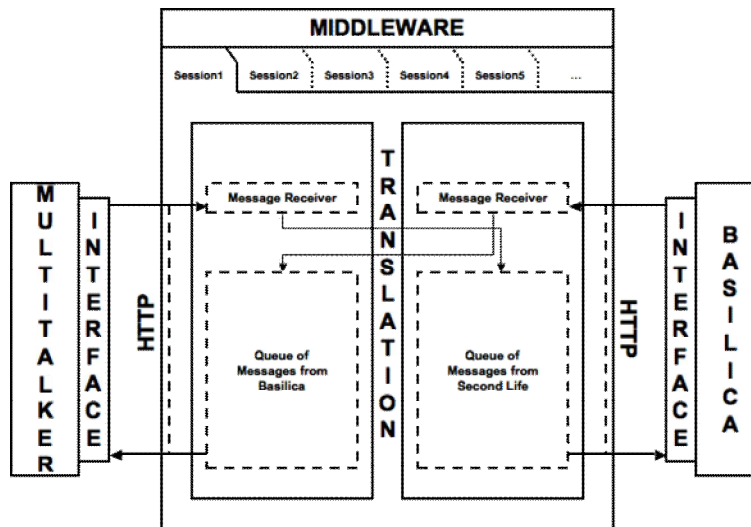


Figure 2: Middleware for Second Life / Basilica Integration

The restricted modes of communication that Second Life provides to its internal objects with the outside world was the primary constraint that led us choosing HTTP as the communication channel for the middleware that facilitates the Basilica/Second Life integration. However, other than the choice of the specific transport protocol, the middleware implements all the generic functionality required for most imaginable extensions in the exchange between the agent back-end and front-end.

The message translation in the current form is as simple as attaching additional appropriate headers to the payload before adding to the other object's queue. However, the generic nature of the middleware permits doing more sophisticated message translation. This facilitates integration between other environments and front-end components that may not be using Basilica specific messaging formats.

The PresenceActor and the ChannelFilter components of the conversational agent use the middleware interface to send messages to the front-end via the middleware. The PresenceActor is responsible for collecting the events from all other actors and sending them across through the middleware. The ChannelFilter constantly polls the middleware to check for messages from the front-end. Similar functionality is implemented in the Second Life object using the virtual environment's proprietary Linden Scripting Language (LSL).

The development and integration of the conversational agent was an experimental effort after we developed the first version of a conversational agent using Basilica for our CycleTalk [1] project. The CycleTalk project aims to develop an intelligent tutor, which participates in thermodynamic design tasks with students to help them learn the underlying concepts. Most of the Basilica components shown above in Figure 1, except the PresenceActor, ChannelFilter and the Middleware Interface, were used as-is for the Second Life conversational agent. The development was done by two of the authors largely using that previous work. We estimated that between 15 to 20 person-hours of work was spent on this effort. Table 1 below shows an estimated breakdown of prominent activities and time spent on them. This demonstrates the extensive reusability of Basilica components due to the isolation of individual behaviors and the separation of the agency of components from the environment.

Table 1. Development Activities and Time Spent.

Activity Type	Activity	Hours Spent
<i>Earlier Work</i>	<i>Development of CycleTalk Agent with 13 Basilica components</i>	<i>40</i>
<i>Earlier Work</i>	<i>Development of Second Life object</i>	<i>1</i>
New	Development of MiddleWare	4
New	Development of MiddleWare interface for Second Life object	2
New	Development of MiddleWare interface for Basilica	2
New w/ Reuse	Development of Conversational Agent Back-end using Basilica	4
Exchange	Understanding of code between the two developers involved	3
New	Debugging first versions for working integration	3
<i>Recurrent</i>	<i>Maintenance task due to changed requirements since stable version</i>	<i>4</i>

5. Using MultiTalker for Your MUVE Learning Experiences

Once the stages of making a MultiTalker are complete, there is still the work of deploying it and using it appropriately to help coach small groups of students. MultiTalker is primarily designed to tutor students while they try to work together on group homework assignments in a MUVE. A human educator is not expected to be present often if at all during these periods. In addition, the students are not supposed to primarily depend on the MultiTalker to tell them everything they must or might do. A MultiTalker is expected to suggest actions and offer reflective advice when group discussions stall or if progress towards the assigned goal seems to be staggering.

Students will typically activate a MultiTalker agent when they begin their group work. Students of a workgroup all touch the agent with their avatar in order to register that they are producers of chat text that the agent should “listen” to. When conversation dies down, becomes off-task, or unproductive, the agent waits a specified amount of time and interrupts with some reflective prompt, inviting members of the group to discuss some likely relevant aspect or their task. The agent will ask members of the group to answer questions and will confirm if answers are correct and follow-up with implications of the correct answers until productive student-student conversation resumes. This is hypothesized to provide effective metacognitive and procedural guidance for students to finish their group work. Figure 3 shows MultiTalker being used by a group in Second Life.

Figure 3: MultiTalker Being Used by Multiple Avatars in Second Life

6. Discussion and Directions

When MultiTalker was first developed, it used a tutorial script on thermodynamics and was simply rendered as a decorated box. We are giving MultiTalker a more anthropomorphic appearance and we are currently exploring using it for teaching groups of 2-3 students in the domain of laser photonics.

Moving forward with the development of conversational agents for MUVEs like Second Life, we have identified the following directions. Text and gestures is currently the prominent mode of interaction in Second Life and using speech and multimedia could enrich the interactive experience. In accordance with the redundancy principle [7], we may need to have chat text hidden while speech is being produced.

Our integration of conversational agents with a popular virtual world should also create opportunities for

new research in human-computer interaction and learning sciences, as we will be able to run experiments where the tutor's actions are controlled.

References

- [1] Kumar, R., Chaudhuri, S., Rose, C. P., "Leveraging social behavior for task success", Submitted to AERA 2009.
- [2] DeSmedt, B. and Loritz, D., "Can We Talk?", AAAI 1999 Spring Symposium on Artificial Intelligence and Computer Games, pp 28-31, 1999.
- [3] Morris, T.W., "Conversational Agents for Game-Like Virtual Environments", AAAI 2002 Spring Symposium on Artificial Intelligence and Interactive Entertainment, pp 82-86, 2002.
- [4] Johnson, W. L., "Serious use of a serious game for language learning", Proc. of the Intl. Conf. on Artificial Intelligence, Amsterdam, 2007.
- [5] Goertzel, B., "A pragmatic path towards endowing Virtually-Embodied AIs with Human level linguistic capability", IEEE World Congress on Computational Intelligence, Hong Kong, 2008.
- [6] Jordan, P., Ringenber, M., Hall, B., "Rapidly Developing Dialogue Systems that Support Learning Studies", Proc. of the ITS-06 Workshop on Teaching with Robots, Agents, and NLP, 2006.
- [7] Mayer, R. E. 2005. Principles for Reducing Extraneous Processing in Multimedia Learning: Coherence, Signaling, Redundancy, Spatial Contiguity, and Temporal Contiguity Principles. *The Cambridge Handbook of Multimedia Learning*:183-200.