To appear in *Computer Animation and Simulation '99* (Eurographics Series). Vienna, Austria: Springer Verlag.

Requirements for an Architecture for Embodied Conversational Characters

J. Cassell, T. Bickmore, L. Campbell, K. Chang, H. Vilhjálmsson, H. Yan

Gesture and Narrative Language Group
MIT Media Laboratory
E15-315, 20 Ames St, Cambridge, Massachusetts
{justine, bickmore, elwin, tetrion, hannes, yanhao}@media.mit.edu

Abstract

In this paper we describe the computational and architectural requirements for systems which support real-time multimodal interaction with an embodied conversational character. We argue that the three primary design drivers are real-time multithreaded entrainment, processing of both interactional and propositional information, and an approach based on a functional understanding of human face-to-face conversation. We then present an architecture which meets these requirements and an initial conversational character that we have developed who is capable of increasingly sophisticated multimodal input and output in a limited application domain.

1 Introduction

Research in computational linguistics, multimodal interfaces, computer graphics, and autonomous agents has led to the development of increasingly sophisticated autonomous or semi-autonomous virtual humans over the last five years.

Autonomous self-animating characters of this sort are important for use in production animation, interfaces and computer games. And increasingly their autonomy comes from underlying models of behavior and intelligence, rather than simple physical models of human motion. Intelligence also increasingly refers not just to the ability to reason, but also to "social smarts" – the ability to engage a human in an interesting, relevant conversation with appropriate speech and body behaviors. Our own research concentrates on the type of virtual human that has the social and linguistic abilities to carry on a face-to-face conversation, what we call Embodied Conversational Agents.

Embodied conversational agents may be defined as those that have the same properties as humans in face-to-face conversation, including:

- The ability to recognize and respond to verbal and non-verbal input
- The ability to generate verbal and non-verbal output.
- The use of conversational functions such as turn taking, feedback, and repair mechanisms.
- A performance model that allows negotiation of the conversational process, and contributions of new propositions to the discourse.

Our current work grows out of experience developing two prior systems—"Animated Conversation" [5] and Ymir [14]. Animated Conversation was the first system to automatically produce context-appropriate gestures, facial movements and intonational patterns for animated agents based on deep semantic representations of information, but did not provide for real-time multimodal interaction with a user. The "Ymir" system focused on integrating multimodal input from a human user, including gesture, gaze, speech, and intonation, but was only capable of limited multimodal output in real time in an animated character called "Gandalf".

We are currently developing a conversational character architecture which integrates the real-time multimodal aspects of Ymir with the deep semantic generation and multimodal synthesis capability of Animated Conversation. We believe the resulting system will provide a reactive character with enough of the nuances of human face-to-face conversation to make it both intuitive and robust. We also believe that such a system provides a strong platform on which to continue development of embodied conversational agents.

2 Motivation

There are a number of motivations for developing conversational character interfaces, including:

> *Intuitiveness.* Conversation is an intrinsically human skill that is learned over years of development and is practiced daily. Conversational interfaces provide an intuitive paradigm for interaction, since the user is not required to learn new skills.
>
> *Redundancy and Modality Switching:* Embodied conversational interfaces support redundancy and complementarity between input modes. This allows the user and system to increase reliability by conveying information in more than one modality, and to increase expressiveness by using each modality for the type of expression it is most suited to.
>
> *The Social Nature of the Interaction.* Whether or not computers look human, people attribute to them human-like properties such as friendliness, or cooperativeness [8]. An embodied conversational interface can take advantage of this and prompt the user to naturally engage the computer in human-like conversation. If the interface is well-designed to reply to such conversation, the interaction may be improved.

In this paper we will first present a summary of the salient features of human face-to-face conversation, and how these drive the design of an architecture which is able to control an animated character who participates effectively in this kind of interaction. We then present an architecture that we have been developing to meet these requirements and describe our first conversational character constructed using the architecture – Rea.

3 Human Face-to-Face Conversation

Embodied conversation relies on a number of different *modalities* such as speech, prosody, hand gestures, facial expression and head movements. The speaker employs these channels in parallel, combining modalities as needed for appropriate elaboration, while the listener simultaneously produces multi-modal feedback and contentful responses in a similar way. The speaker and listener accomplish the switching of roles through a sequence of overlapping turn-taking behaviors where the parallel nature of the communication channels, and short timescales of the relevant behaviors provide a seamless transition.

The behaviors that directly contribute to the content delivery or the organization of the conversation are termed *conversational behaviors* and are the surface form of the exchange. Typical conversational behaviors include head nods, glances to the side, raising eyebrows, and speaking. But it is also important to identify the *functions* that these conversational behaviors serve. Typical discourse functions include *conversation initiation*, *giving and taking turns*, *giving and requesting feedback,* and *breaking away.* The same conversational behavior can contribute to the realization of different discourse functions and the same discourse function can be implemented using different combinations of conversational behaviors. For example, head nods can indicate agreement, or simply attention; and to indicate agreement a listener may nod or say "uh huh."

To further clarify the type of roles discourse functions serve, the contribution to the conversation can be divided into *propositional information* and *interactional information*. Propositional information corresponds to the content of the conversation and includes meaningful speech as well as gestures, facial expression, head movements and intonation used to complement or elaborate upon the speech content. Interactional information consists of cues that affect the conversational process and includes a range of nonverbal behaviors as well as regulatory speech such as "huh?" "Uh-huh".

4 Architectural Requirements

The construction of a computer character which can effectively participate in face-to-face conversation as described above requires a control architecture which has the following features:

· *Multi-Modal Input and Output* – since humans in face-to-face conversation send and receive information through gesture, intonation, and gaze as well as speech, the architecture also should support receiving and transmitting this information.

· *Real-time* –The system must allow the speaker to watch for feedback and turn requests, while the listener can send these at any time through various modalities. The architecture should be flexible enough to track these different threads of communication in ways appropriate to each thread. Different threads have different response time requirements; some, such as feedback and interruption occur on a sub-second timescale. The architecture should reflect this fact by allowing different processes to concentrate on activities at different timescales.

· *Understanding and Synthesis of Propositional and Interactional Information* – Dealing with propositional information requires building a model of user's needs and knowledge. Thus the architecture must include both a static domain knowledge base and a dynamic discourse knowledge base. Presenting propositional information requires a planning module to plan how to present multi-sentence output and manage the order of presentation of interdependent facts. Understanding interactional information, on the other hand, entails building a model of the current state of the conversation with respect to conversational process (who is the current speaker and who is the listener, has the listener understood the speaker's contribution, and so on).

· *Conversational Function Model* – Explicitly representing conversational functions provides both modularity and a principled way to combine different modalities. Functional models influence the architecture because the core modules of the system operate exclusively on functions (rather than sentences, for example), while other modules at the edges of the system translate input into functions, and functions into outputs. This also produces a symmetric architecture because the same functions and modalities are present in both input and output.

5 Related Work

Other researchers have built embodied multimodal interfaces that add dialogue and discourse knowledge to produce more natural conversational characters. For example, Olga is an embodied humanoid agent that allows the user to employ speech, keyboard and mouse commands to engage in a conversation about microwave ovens [3]. Olga has a distributed client-server architecture with separate modules for language processing, interaction management, direct manipulation interface and output animation, all communicating through a central server. Olga is event driven, and so only responds to user input and is unable to initiate output on its own. In addition, Olga does not support non-speech audio or computer vision as input modalities.

Olga uses a linear architecture in which data flows from user input to agent output, passing through all the internal modules in between. Nagao and Takeuchi [8] suggest a different approach. Their conversational agent is based on the subsumption architecture by Rodney Brooks [4]. In this case the agent is based on a horizontal decomposition of task-achieving behavior modules. The modules each compete with one another to see which behavior is active at a particular moment. Thus there is no global conversational state or model and the conversational interaction arises from the interplay between the different behavioral layers. Their agent responds to speech and gaze information, but coordination of the input analysis and output generation is also an emergent behavior, so precise control is impossible. The end result is that user input and agent output are decomposed according to task behaviors rather than conversational function.

Lester et al. [7] do generate verbal and non-verbal behavior, producing deictic gestures and choosing referring expressions as a function of the potential ambiguity of objects referred to, and the proximity of those objects to the animated agent. This system is based on an understanding of how reference is achieved to objects in the physical space around an animated agent, and the utility of deictic gestures in reducing potential ambiguity of reference. However, the generation of gestures and the choice of referring expressions (from a library of voice clips) are accomplished in two entirely independent (additive) processes, without a description of the interaction between the two modalities. Likewise, Rickel and Johnson [10] have their pedagogical agent move to objects in the virtual world that it inhabits, and then generate a deictic gesture at the beginning of the verbal explanation that the agent provides about that object.

Work by the Thalmanns [13] and by Badler [2] has concentrated on smooth and natural behaviors for virtual humans who exist in a virtual environment, and can interact and converse with other virtual characters. The Thalmanns' work on digital actors, in particular, has begun to address the challenges of mapping human facial conversational behaviors onto graphical characters. The demands of virtual spaces are quite different, since the embodied character must respond to other characters, rather than real humans. However, the issues involved in generating appropriate and realistic conversational behaviors are similar.

Our current approach derives from previous work by a student in our research group on the Ymir architecture [14]. In this work the main emphasis was the development of a multi-layer multimodal architecture that could support fluid face-to-face dialogue between a human and graphical agent. The agent, Gandalf, recognized and displayed interactional information such as gaze and simple gesture and also produced propositional information, in the form of canned speech events. In this way it was able to perceive and generate turn-taking and backchannel behaviors that lead to a very natural conversational interaction. This work provided a good first example of how verbal and non-verbal function might be paired in a conversational multimodal interface.

However, Gandalf had limited ability to recognize and generate propositional information, such as providing correct intonation for speech emphasis on speech output, or a gesture co-occurring with speech. In contrast, "Animated Conversation" [5] was a system that automatically generated context-appropriate gestures, facial movements and intonational patterns. In this case the domain was conversation between two artificial agents and the emphasis was on the production of non-verbal propositional behaviors that emphasized and reinforced the content of speech. The system did not run in real-time and since there was no interaction with a real user, the interactional information was very limited.

The approach we use combines lessons learned from both the *Gandalf* and *Animated Conversation* projects. In the next section we present a conversational function based architecture for developing embodied conversational interfaces. Following that we describe Rea, the first conversational humanoid based on this architecture.

6 Conversational Humanoid Architecture

Based on our previous experience with Animated Conversation and Ymir we have been developing an architecture that handles both real-time response to interactional cues and deep semantic understanding and generation of multimodal inputs and outputs[1]. At a high level, our architecture is partitioned into: an Input Manager, which is responsible for collecting inputs across modalities; an Action Scheduler, responsible for synchronizing output actions across modalities; and components which handle the real-time interactional functions and longer-term deliberative responses such as content understanding and synthesis. The full breakdown of the architecture is shown in Figure 1. The modules communicate with each other using KQML, a speech-act based inter-agent communication protocol, which serves to make the system very modular and extensible. Each of the modules in the architecture are described next.

6.1 Modules

6.1.1 Input Manager

The Input Manager obtains data from the various input devices, converts it into a form usable by other modules in the system, and routes the results to the Understanding Module. Interactional information is also forwarded directly to the Reaction Module to minimize system response time.

[1] This architecture has been developed in conjunction with the Conversational Characters project at Fuji-Xerox Palo Alto Laboratory.

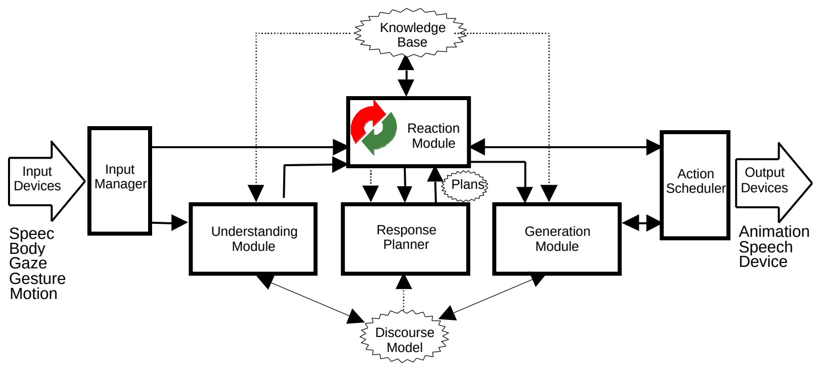

Fig. 1. Detailed Conversational Character

The Input Manager will typically receive information from devices which provide speech text, user gesture, location, and gaze information, and other modalities. In all cases the features sent to the Input Manager are time stamped with start and end times in milliseconds. In our current implementation, the Input Manager also bundles co-temporal input events into aggregate semantic representations (e.g., a user utterance and accompanying gestures) for the Understanding Module to process.

6.1.2 Understanding Module

The Understanding Module is responsible for fusing all input modalities into a coherent understanding of what the user is doing. The Understanding Module receives inputs from the Input Manager and can access knowledge about the application domain (Static Knowledge Base) and the current discourse context (Discourse Model) to help it interpret the inputs. For example, if the user gestures while the character is speaking it is interpreted as a *wanting turn* function, whereas if a gesture is detected while the user is speaking it is taken as a speech-accompanying gesture.

6.1.3 Reaction Module

The Reaction Module is responsible for the "action selection" component of the architecture, which determines what the character should be doing at each moment in time. The Reaction Module receives asynchronous updates from the Input Manager and Understanding Module, and uses information about the domain (Static Knowledge Base) and current Discourse State to determine the action to perform.

The Reaction Module currently responds to interactional cues based on a set of states. The system starts up in the *NotPresent* state and remains there until a user is detected, at which time it transitions to *Present*. Once the user and the character have exchanged greetings (or other similar cues) the system transitions into turn-taking, represented by *UserTurn* and *ReaTurn* states. The *Conclude* state is used to handle user interruptions of the character, allowing it to continue to the end of a phrase boundary before giving the turn back to the user. The *Interrupt* state is entered if the system detects that the user has turned away. We anticipate adding more states as we begin to explore multi-sentential, mixed-initiative dialog.

6.1.4 Response Planner Module

The Response Planner is responsible for formulating sequences of actions, some or all of which will need to be executed during future execution cycles, to carry out desired communicative or task goals.

6.1.5 Generation Module
The Generation Module is responsible for realizing discourse functions output from the Reaction Module by producing a set of coordinated primitive actions (such as speech or gesture generation, or facial expression), sending the actions to the Action Scheduler for performance, and monitoring their execution.

6.1.6 Action Scheduling Module
The Action Scheduler is the "motor controller" for the character, responsible for coordinating output actions at the lowest level. It takes a set of atomic modality-specific commands and executes them in a synchronized way. This is accomplished through the use of event conditions specified on each output action which define when the action should be executed.

6.2 Fulfillment of Architectural Requirements
We feel that the architecture described meets all of the requirements for an embodied conversational character that can participate face-to-face conversation with a human. It is capable of reacting to and producing inputs and outputs across multiple modalities by mapping specific features of these modalities into conversational functions and using a uniform knowledge representation format (KQML) throughout the system. It can run in real-time, by providing immediate responses to interactional cues, and decoupling processes such as content understanding and synthesis, which can take seconds of response time. The architecture is able to work with both interactional and propositional information, in fact most KQML frames used within our implementation have slots for both kinds of input interpretations or output specifications present. The presence of the Understanding and Generation Modules in the architecture are specifically to enable the separation of channel-specific features from conversational functions, allowing the Reaction and Response Planning Modules to deal entirely at the functional level of abstraction. Finally, the use of a common KQML representation throughout, coupled with the disciplined use of functional descriptors allows the system to be very extensible with respect to input and output modalities, and modular with respect to plugging in new modules which implement alternative theories of discourse.

7 Implementation
Rea ("Real Estate Agent") is our first instantiation of the architecture described above. Rea is a computer generated humanoid that has a fully articulated graphical body, can sense the user passively through cameras and audio input, and is capable of speech, facial display, and gestural output. The system currently consists of a large projection screen on which Rea is displayed and which the user stands in front of. Two cameras mounted on top of the projection screen track the user's head and hand positions in space. Users wear a microphone for capturing speech input. A single SGI Octane computer runs the graphics and conversation engine of Rea, while several other computers manage the speech recognition and generation and image processing (Figure 2). The system is implemented in C++ and CLIPS, a rule-based expert system programming language.

7.1 A Sample Interaction

Rea's domain of expertise is real estate and she acts as a real estate agent showing users the features of various models of houses that appear on-screen behind it. The following is a excerpt from a sample interaction:

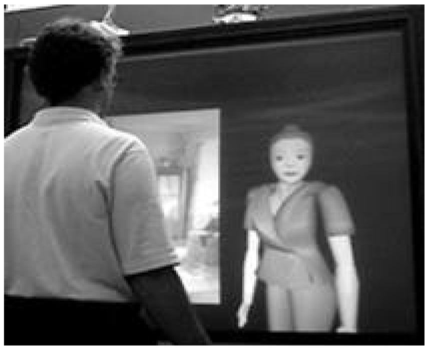
Fig. 2. User interacting with Rea

Lee approaches the projection screen. Rea is currently turned side on and is idly gazing about. As the Lee moves within range of the cameras, Rea turns to face him and says "Hello, my name is Rea, what can I do for you?" "Hi, I'm Lee, I'm looking for a place near MIT." Rea replies "I have a house in Somerville" after briefly looking up and away while pondering. "Sounds good, tell me about the house". A picture of a house appears on-screen behind Rea."This is a nice Victorian with a large garden" Rea says while producing a curved gesture, indicating that the garden surrounds the house. Rea continues "It has two bedrooms and a large kitchen ..." Lee raises his hands into space, indicating his intention to take the turn, so Rea yields the turn to Lee. "Tell me about the bedrooms," Lee says. An image showing the master bedroom appears."The master bedroom is furnished with a four poster bed and…" "Where is the bathroom?". Lee says ,interrupting Rea in a mid-sentence. "It is next to the bedroom" Rea replies, placing her hands close to each other to show the adjacency arrangement. And the house tour continues…

Rea is able to describe the features of a house while also responding to the users' verbal and non-verbal input. When the user makes cues typically associated with turn taking behavior such as gesturing, Rea allows herself to be interrupted, and then takes the turn again when she is able. She is able to initiate conversational repair when she misunderstands what the user says, and can generate combined voice and gestural output. Rea's speech and gesture output is generated in real-time. The descriptions of the houses that she shows, along with the gestures that she uses to indicate and describe those houses are generated using the SPUD natural language generation engine [12], modified so as to also generate natural gesture.

7.2 Input Sensors

The input manager currently receives three types of input:

- Gesture Input: STIVE vision software[1] uses two video cameras to track flesh color and produce 3D position and orientation of the head and hands at 10 to 15 updates per second.
- Audio Input: A simple audio processing routine detects the onset, pauses, and cessation of speech.
- Grammar Based Speech Recognition: Speech is also piped to a PC running IBM's ViaVoice98, which returns text from a set of phrases defined by a grammar.

Data sent to the Input Manager is time stamped with start and end times in milliseconds. The various computers are synchronized to within a few milliseconds of each other using NTP (Network Time Protocol) clients. This synchronization is key for associating verbal and nonverbal behaviors. Low level gesture and audio detection events are sent to the reaction module immediately. These events are also stored in a buffer so that when recognized speech arrives a high-level multimodal KQML frame can be created containing mixed speech, audio and gesture events. This is sent to the understanding module for interpretation.

7.3 Output System

The multi-modal and real-time requirements call for a careful design of the output system. In particular, a conversational character needs a perfect coordination between speech and nonverbal behavior such as gesturing. The slightest mismatch will not only look unnatural, but could in fact convey something different from what was intended. The modularity and extensibility requirement has enforced well defined interfaces between the various components of the output system and has inspired the implementation of a plug-in style motor skill mechanism.

The output system consists of three main components: a Scheduling Component, an Animation Component, and a Rendering Component. The Scheduler receives requests for the activation of various behaviors from the Generation Module. The requests include interdependencies among the behaviors, such as requirements about one behavior finishing before another one starts. The Scheduler is therefore responsible for successfully sequencing pending behaviors. The Animator assigns a behavior ready to be executed to a motor skill that then becomes responsible for animating the joints of the model by communicating with the Renderer (Figure 3).

7.3.1 Scheduler

A behavior description, along with its preconditions and manner of execution are sent to the Scheduler in a KQML message. The Generation Module typically sends the Scheduler a set of behaviors that together, when properly triggered, are meant to carry out a single function, such as an invitation to start a conversation. The Scheduler can be instructed to notify the Generation Module through KQML callback messages when certain events occur, such as completion of an output behavior sequence.

Execution of behavior in the Scheduler is event-driven because it is often difficult to accurately predict output behavior execution timings, making it impossible to plan out completely synchronized execution sequences in advance. In addition, some behaviors can produce meaningful events while they are being executed (e.g., the speech synthesis behavior can produce an event after each word is produced), and

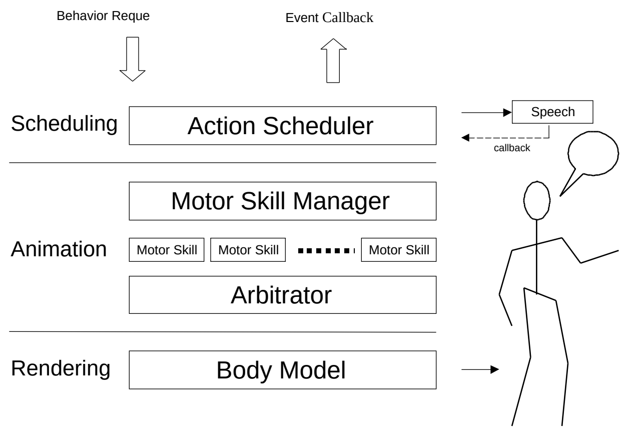

Fig. 3. The three layers of the Output System: Scheduling, Animation and Rendering.

thus allow other behaviors to be started or stopped when these events occur. Figure 4 shows an example of an event-driven plan executed by the Action Scheduler with dependencies among the individual behaviors.

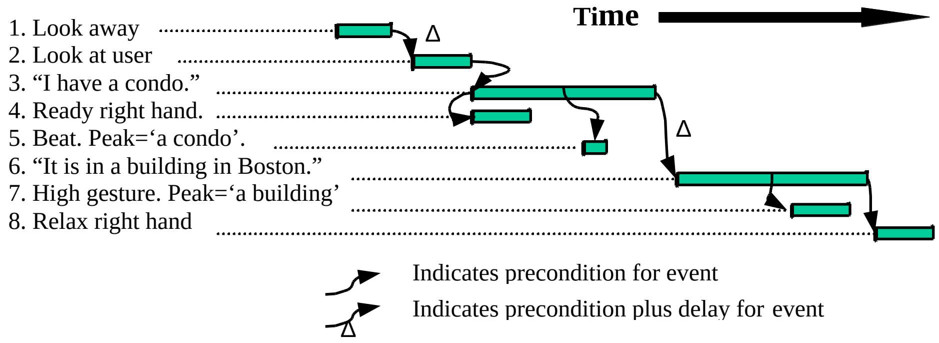
Time
1. Look away
2. Look at user
3. "I have a condo."
4. Ready right hand.
5. Beat. Peak='a condo'.
6. "It is in a building in Boston."
7. High gesture. Peak='a building'
8. Relax right hand
Δ
Δ
Indicates precondition for event
Indicates precondition plus delay for event
Δ

Fig. 4. Example of synchronized speech and gesture output by the Action Scheduler.

The specification sent to the Action Scheduler contains a description of each individual behavior to be executed (a ":content" clause), along with a precondition for the start of the behavior (a ":when" clause) and an optional symbolic label (":id") which can be used in the preconditions of other behaviors. Figure 5 shows the KQML input specification for the plan shown in Figure 4.

```
[(action :id H_AWAY :when immediate
         :content (headlook :cmd away :object user))
(action :id H_AT :when (offset_after :event H_AWAY.END :time 00:01.50)
         :content (headlook :cmd towards :object user))
(action :id S_CONDO :when (after :event H_AT.END)
         :content (speak :content "I have a condo."))
(action :when (after :event S_CONDO.START)
         :content (rightgesture :cmd ready))
(action  :when (after :event S_CONDO.WORD3)
         :content (rightgesture :cmd beat))
(action :id S_BLDG :when (offset_after :event S_COND.END :time 00:01.00)
         :content (speak :content "It is in a building in Boston."))
(action  :when (after :event S_BLDG.WORD4)
         :content (rightgesture :cmd compose :trajectory vertup :hand bend))
(action  :when (after :event S_BLDG.END)
         :content (rightgesture :cmd relax))]
```

Fig. 5. Action Scheduler KQML input specification for the plan shown in Figure 6.

The Action Scheduler works by managing a set of primitive behavior objects, each of which represents a set of animations (e.g., "right arm gestures"). When a behavior is commanded to start it first acquires the body Degrees Of Freedom (DOF) that it requires, such as the set of the right arm and hand joints. It then goes into a starting phase in which it can perform initialization, such as moving the arm into a ready position. Most of the behavior's actions are carried out in the update phase, which ends when the behavior reaches a natural stopping point, when it is explicitly commanded to stop, or when some other behavior preempts it by grabbing one or more of its DOFs. Before returning to idle, a behavior can go through an ending phase in which it can perform any wrapup operations needed, such as returning the arm to its rest position.

7.3.2 Animator

When the Scheduler has a non-verbal behavior ready for execution, it passes its description over to the Animator. Actions not involving the character's body are executed directly, for example verbal behavior is sent to the speech synthesizer. The Animator checks with the Motor Skill Manager to see if a motor skill capable of handling the request has registered with it. The task of animating joints of the model was broken up into separate motor skills in part because the different skills called for different methods of animation. Motor skills range from straight forward ones, such as those executing a single head nod, to more elaborate ones such as those employing Inverse Kinematics for pointing at objects or playing key-frame animation. When a motor skill is activated, it asks the Arbitrator for the body DOFs it needs to modify. If two skills ask for the same DOF, the one with the higher priority captures it. Depending on the implementation of particular skills, the losing skill can keep trying to capture the DOF. This feature is useful for instances where a continuous behavior is momentarily interrupted by an instantaneous one, such as when the character is tracking the user with it's gaze, and gets asked to glance up and away (higher priority). When the glance is completed, the tracking automatically resumes. The Arbitrator is responsible for keeping track of DOFs in use and allocating them to skills that request them. All skills can access information about the environment, including virtual objects and the perceived user position through a shared World. Motor skills such as for controlling facing can therefore accept names of objects as parameters.

7.3.3 Renderer

The rendering engine is abstracted away from the animator by introducing a BodyModel layer that essentially maps a DOF name to the corresponding model transformation. We have implemented a BodyModel that interfaces with a VRML scene graph rendered using OpenInventor from TGS. The naming of the character's DOFs follows the H-Anim VRML Humanoid Specification for compatibility [11].

8 Conclusion

In this paper we have argued that a new approach to intelligence and autonomy is needed in the design of virtual humans and other autonomous animated characters. This approach goes beyond the insights about reasoning offered by classical AI, and beyond the focus on believability advocated by classical graphical animation. In this new approach, autonomy comes from underlying models of social and linguistic intelligence that allow our autonomous animated agents to be able to carry on realistic conversations with humans, using both speech and other visual modalities. We argue that the nature of human face-to-face communication imposes strong requirements on the design of embodied conversational characters, and have described how our architecture satisfies these requirements.

We demonstrated our approach with the Rea system. Increasingly capable of making an intelligent content-oriented – or *propositional* – contribution to the conversation, Rea is also sensitive to the regulatory – or *interactional* -- function of verbal and non-verbal conversational behaviors, and is capable of producing regulatory behaviors to improve the interaction by helping the user remain aware of the state of the conversation. Rea is an embodied conversational agent who can hold up her end of the interaction.

9 Future Work

Implementing multimodal embodied conversational characters is a very complex undertaking, and we have an extensive research agenda of conversational competencies to add or improve on. Finally, the least reliable components of the system are the individual modality feature detectors, and we are continuing to refine and extend these to provide user gaze direction, facial expression, and gestural form.

References

1. Azarbayejani, A., Wren, C., Pentland A. Real-time 3-D tracking of the human body. In *Proceedings of IMAGE'COM 96*, Bordeaux, France, May 1996.
2. Badler, N., et al. A Parameterized Action Representation for Virtual Human Agents. *Proceedings of the 1st Workshop on Embodied Conversational Characters 1998, 1-8.*
3. Beskow, J. and McGlashan, S. Olga - A Conversational Agent with Gestures, In *Proceedings of the IJCAI'97 workshop on Animated Interface Agents - Making them Intelligent*, Nagoya, Japan, August 1997
4. Brooks, R.A. A Robust Layered Control System for a Mobile Robot. *IEEE Journal of Robotics and Automation.* 2 (1), 1986, 14-23.
5. Cassell, J., Pelachaud, C., Badler, N.I., Steedman, M., Achorn, B., Beckett, T., Douville, B., Prevost, S. and Stone, M. Animated conversation: rule-based generation of facial display, gesture and spoken intonation for multiple conversational agents. *Computer Graphics (SIGGRAPH '94 Proceedings)*, 1994, 28(4): 413-420.
6. Cassell, J. and Thórisson, K. The Power of a Nod and a Glance: Envelope vs. Emotional Feedback in Animated Conversational Agents. *Journal of Applied AI*, in press.
7. Lester, J., Towns, S., Calloway, C., and FitzGerald, P. Deictic and emotive communication in animated pedagogical agents. In *Proceedings of the Workshop on Embodied Conversational Characters.* 1998.
8. Nagao, K. and Takeuchi, A. Social interaction: multimodal conversation with social agents. *Proceedings of the 12th National Conference on Artificial Intelligence (AAAI-94)*, (Seattle, WA, August 1994), AAAI Press/MIT Press, 1994, vol. 1, 22-28.
9. Reeves, B. and Nass, C. The Media Equation: How People Treat Computers, Television, and New Media Like Real People and Places. Cambridge University Press, 1996.
10. Rickel, J. and Johnson, L. Task-oriented Dialogs with Animated Agents in Virtual Reality" in Workshop on Embodied Conversational Characters. In *Proceedings of the Workshop on Embodied Conversational Characters.* 1998.
11. *Specification for a Standard VRML HumanoidVersion 1.0.* http://ece.uwaterloo.ca/~h-anim/spec.html
12. Stone, M. *Modality in Dialogue: Planning, Pragmatics, and Computation*. PhD Thesis, University of Pennsylvania, 1998.
13. Thalmann, N.M., Kalra, P., and Escher, M. (1998). Face to virtual face. *Proceeding of the IEEE*, 86(5), 870-883.
14. Thórisson, K. R. *Communicative Humanoids: A Computational Model of Psychosocial Dialogue Skills.* PhD Thesis, MIT Media Laboratory, 1996.