Automatic Generation of Non-verbal Behavior for Agents in Virtual Worlds: A System for Supporting Multimodal Conversations of Bots and Avatars

Werner Breitfuss[1], Helmut Prendinger[2], and Mitsuru Ishizuka[3]

[1] University of Tokyo , 7-3-1 Hongo, Bunkyo-ku,
Tokyo, 113-8656, Japan
werner@mi.ci.i.u-tokyo.ac.jp
[2] National Institute of Informatics, 2-1-2 Hitotsubashi, Chiyoda-ku,
Tokyo, 101-8430, Japan
helmut@nii.ac.jp
[3] University of Tokyo, 7-3-1 Hongo, Bunkyo-ku,
Tokyo, 113-8656, Japan
ishizuka@i.u-tokyo.ac.jp

Abstract. This paper presents a system capable of automatically adding gestures to an embodied virtual character processing information from a simple text input. Gestures are generated based on the analysis of linguistic and contextual information of the input text. The system is embedded in the virtual world called second life and consists of an in world object and an off world server component that handles the analysis. Either a user controlled avatar or a non user controlled character can be used to display the gestures, that are timed with speech output from an Text-to-Speech system, and so show non verbal behavior without pushing the user to manually select it.

Keywords: Embodied Virtual Characters, Animated Agent Systems, Multimodal Output Generation, Multimodal Presentations, Virtual Worlds.

1 Introduction

Virtual agents represent a powerful human-computer interface, as they can embody behavior that a human may identify with [10], this ability may encourage users to engage in a more natural and immersive interaction and establish bonds with them [4].This paper describes an automatic non-verbal behavior generation system for virtual agents using linguistic and contextual information retrieved from text. It allows us to transform text into an agent behavior script enriched by eye gaze and conversational gesture behavior. The agents' gaze behavior is informed by theories of human face-to-face gaze behavior. Gestures are generated based on the analysis of linguistic and contextual information of the input text. The aim of our work is to generate non-verbal behavior automatically for conversations utilizing virtual agents, so that the user can focus on typing the text, which is then just feed into the system. A salient feature of our system is that we support behavior generation not only for the

A.A. Ozok and P. Zaphiris (Eds.): Online Communities, LNCS 5621, pp. 153–161, 2009.

role of the speaking agent, but also for listening agents, who might use backchannel behavior in response to the speaker agent. The increasing popularity of virtual worlds pushes the need for virtual characters either controlled directly by the user, called avatars, or controlled by a script, called bots.

This system can be used to provide natural gestures for both types and since all behaviors are generated automatically, there is no extra effort the user would have to contribute to increase the naturalness of the characters behavior and so provides a convenient method to have multimodal conversations in virtual environments. The speech is generated automatically by a plug-in [3] that uses Microsoft's SAPI to transform text messages in Second Life into speech output at the clients machine, Second Life own Voice over IP client can be used to relay that speech output back in world. So we are able to provide a multimodal conversations and dialogues in a 3d interactive environment that doesn't encumber the user with extra workload.

In the next part, the paper discusses related research, while section 3 describes the behavior generation and section 4 focuses on the application using it. Section 5 gives a brief future outlook and concludes the paper.

2 Related Research

The existing character agent systems already support the automated generation of some behaviors, such as automatic lip-synchronization most of this systems that focus on single agents , one of the first ones to do so was the BEAT system [2] it generates synthesized speech and synchronized non-verbal behavior for a single animated agent. It uses plain text as input, which is then transformed into animated behavior. First, text is annotated with contextual and linguistic information, based on which different (possibly conflicting) gestures are suggested. Next, the suggested behaviors are processed in a 'filtering' module that eliminates gestures that are incompatible. In the final step, a set of animations is produced that can be executed, after necessary adoptions, by an animation system. The system described in [8] generates both the language and deictic gestures of a robot-like virtual character for giving directions to a user.

A different approach, based on machine learning, is suggested in [7]. It was used in the COHIBIT system, where the author first has to provide a script containing the actions for two virtual characters. In the next step the author writes simple gesture rules using his or her expert knowledge. Using this corpus of annotated actions the system can learn new rules. In the third step the system suggests the most appropriate gestures to the author, which are, after resolving conflicts and filtering, added to the already existing ones. Finally it produces a script with the gesture behavior of both virtual characters. This work differs from ours in the sense that it uses input from the user making supporting taking some workload of him, but the generation is not entirely automatically as in our approach.

Many of the rules used in our system are derived from works of psychologists and linguists. Heylen [5] investigated the many different functions of gaze in conversation and its importance for the design of believable virtual characters. The gaze behavior of our agents is also informed by the empirically founded gaze models in [6, 9, 12]. Kendon [6] analyzed gaze behavior based on two-person dialogues and found that gaze is used to regulate the exchange between the speaker and listener. In [12] evaluates gaze behavior in multiparty environments, where four-person groups discussed current-affair topics in face-to-face meetings.

3 Behavior Generation

The Behavior generation in our system operates on the utterance level, for which certain rules are defined. The input we use thus consists of a simple text line. Based on contextual and linguistic information of the text, the behavior for the speaking and the listening agents is suggested. We do not only apply one layer of rules but also a second layer based on keyword and phrase spotting and a third layer, which is using iteration to align suggested gestures to each other. Also many of the gestures have a certain possibility of occurrence, which adds a random factor, so that the behavior of the gesturing character doesn't become redundant which would make the conversation less natural.

As eye gaze is one of our main features, we use a set of rules and algorithms to generate the appropriate patterns for both speaking and listening agent. The relationship between eye gaze, theme/rheme, and turn-taking was the focus in many psycholinguistic studies, we used those results to define an algorithm for controlling the gaze behavior of our two different roles.

Gesture generation is designed similar to gaze generation. Former studies say that 50% of the gestures humans use in a conversation are simple beat gestures. In accord with that finding, the standard gesture we use is a single beat, which is suggested whenever there appears a new word in the utterance. In the next step we identify words that can be played out by gestures that are more specific than the beat gesture. E.g., when the sentence contains the adjective "big", an iconic gesture ("show size") will be suggested, a gesture where our agents holds both hands with a certain distance between each other in front of his body. The information which gesture can be associated with what word is stored in a XSL based database, a typical entry is depicted by figure 1, all entries have the attributes priority and type of the gesture.

```
<xsl:templatematch= "//W[
.//@LEM='cu' or .//@LEM='bye' or
.//@LEM='goodbye' or .//@LEM='bye bye']">
  <gesture priority="3" type="GOODBYE">
    <xsl:copy>
      <xsl:apply-templates />
    </xsl:copy>
  </gesture>
</xsl:template>
```

Fig. 1. Entry for the goodbye gesture

To show a small example we take the sentence "This is a small example how to draw" and present the output of our system for the speaking agent (figure 2) and the listening agent(figure 3). The root node of the tree is always an utterance followed by a speech pause between the theme and rheme of the sentence. The nodes "Look away" at the beginning of the sentence and "Look at listener" define the gaze behaviour which is suggested by our algorithm (for further details on the gaze generation see [1]).

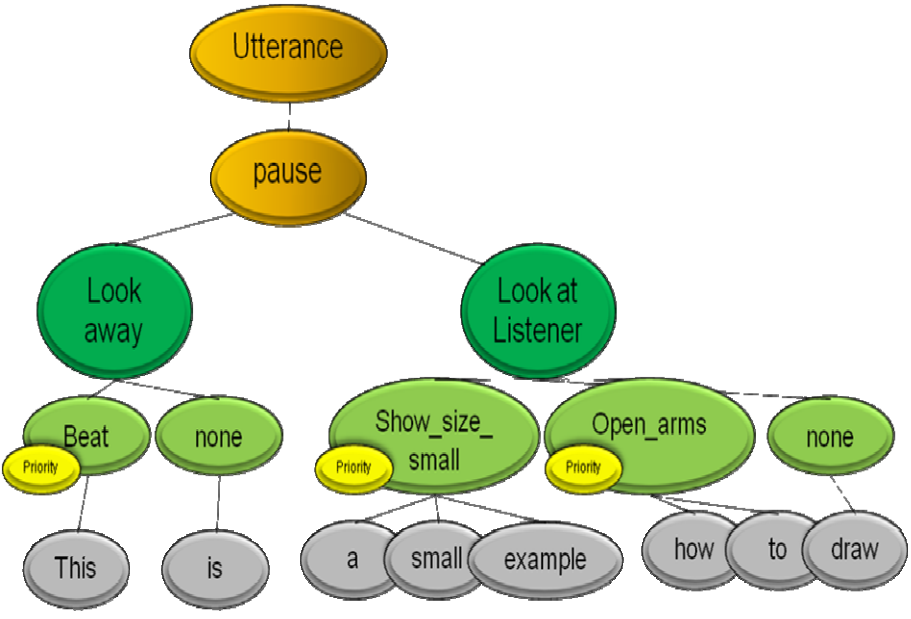

Fig. 2. A typical behavior tree for a speaking character

The gesture behaviour is generated according to dedicated gesture generation rules of the Behaviour Generation module. In this example, a beat gesture is selected to accompany the word "This", an iconic gesture (for describing something small) is suggested to co-occur with the phrase "small example" and a open-arms gesture along with the words "how to".

The behavior tree of the listener agent is generated similarly to that of the speaker agent. It is based also on the output from the Language Tagging module of the speaker agent, but applies listener behavior generation rules instead of speaker rules. Again, we start with root node "UTTERANCE". During the speaker's speech pause, no behavior for the listener agent is defined. The listener's gaze behavior is added similar to the speakers i.e. the listener is looking at the speaker when the utterance begins. Since the listener agent is paying attention to the speaker, it continues to look at the speaker also in the "rheme part" of the utterance. Thereafter, appropriate gestures are suggested for the listener agent. In our system, a head nod is a basic gesture type for the listener and appears often to signalize attention, so it is added whenever a basic speaker gesture like a simple beat occurs, for more complex gestures like a deictic gestures instead of adding a head node we adopt the gaze. This makes the listener behavior more natural and increases the overall communication quality between the two roles.

The last module combines the speaker and listener tree by adding the actions of both agents for every utterance into one MPML3D structure called "task". The MPML Script contains parallel and synchronized actions which can be started and ended at the beginning, middle, or end of a certain word. First we add all the actions

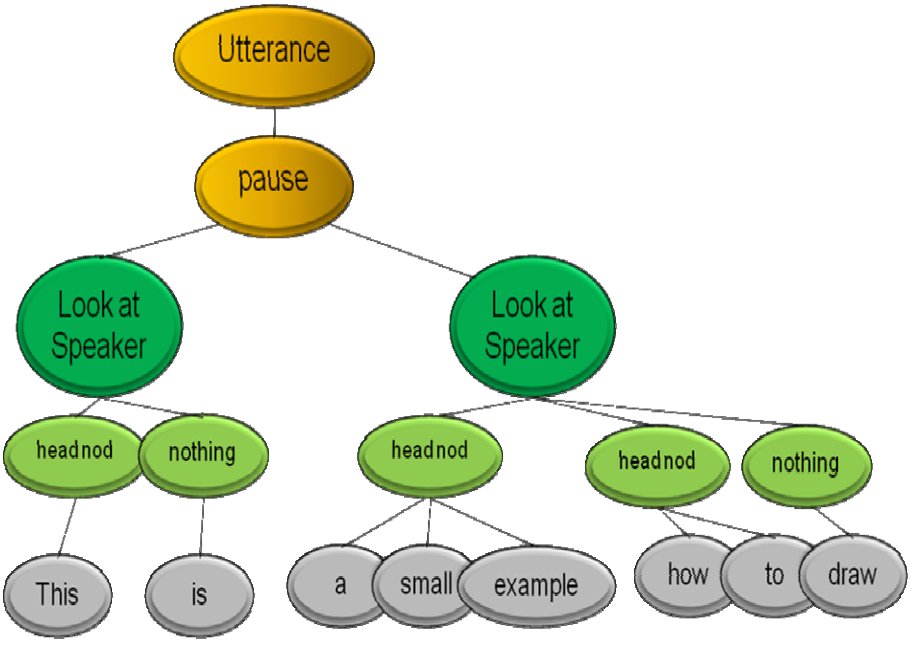

Fig. 3. A typical behavior tree for a speaking character

that should occur before the speaker starts to talk, mostly gaze behavior, like looking away from the speaker and idle gestures for the listener.

The next action that is added is speaking itself. In the following step, we add the gaze behavior, which has to be aligned with the appropriate words. Gaze is implemented by having the head turn to a certain direction. As the last level we add the gesture for the speaking agent and the listening agent.

4 Applications

The System itself consists of two parts one is the server part, where all of the text analysis is done and the client part that actually controls the agent and displays its behavior. The architecture of our system on the server side, as already mentioned, operates on three modules, a language module, the actual behavior generation module, and a module that generates a displayable script (MPML3D) and/or the messages that is sent back to the Second Life client . For this we choose a modular pipelined architecture to support future extensions. The code of the system is written in Java, and the XML format is used to represent and exchange data between modules.

The client side, consists of an object in Second Life containing the needed gestures in form of predefined animations and scripts that handle the communication as well as the activation of the animations. The scripts are written in the official Second Life scripting language, LindenScript.

1234-678 BEAT_TWO:5 POINT_LEFT:12 16

Fig. 4. A typical message from the server sent to the client

4.1 Server Side

The input for our analysis can either be a single text line forming a single string, which is taken from the chat message in Second Life, that the user enters, or a predefined dialogue script , depending whether the system is used for the instant messaging and so controlling an avatar or to generate a MPML-SL script to control bots. In the first case the interface between the Second Life Client and our Server is a simple Java Servlet that passes messages between both Systems. First when the message arrives it is analyzed like we described in the previous chapter, after that, since we only display a pair of gestures, the most fitting gestures are selected. For selection we use a priority system, where beat gestures are counted as the once with the least priority and metaphoric gestures with the highest , since they transport more meaning, the system also takes into account how often a gesture occurs in the utterance and adopts its priority accordingly. After the selection, the message to the client side is composed, it consists of the overall word count of the utterance, the gesture pair together with the index of the word at which they should be displayed (figure 4). The information of the timing is derived from the treelike structure we use for generation, and resembles the speaker behavior, since we have no control over other avatars in this scenario, the listener behavior is disregarded.

4.2 Client Side – AuGe System

In the first case the used avatar has to wear our AuGe Bands, an band like object generated in Second Life, once a chat message is typed in and the user hits the enter button , our object detects the new message and sends it to the Java Servlet which passes it on to the server. After that the system output is sent back to the users client and the new message is processed by the script inside the gesture bands. As shown in figure 4 the message contains 3 main parts.

At first we have the avatar ID to prevent other gesture bands in the proximity to accidently start the same gestures on their avatar. The second part consists of two gestures, the name of the gesture and timing information after the ":" which is the index of the word on which the animation should start. We decided to use only two gestures, one for the THEME and one for the RHEME part, or if a longer text is typed containing multiple sentences the two gestures with the highest priority since more gestures are troublesome as the animations often get delayed over overlay due to unpredictable lag or graphical issues on the user's side.

The last part is the word count of the whole text, and is used to time the lip movement, our system doesn't provide perfect lip synchronization, however since no lip movement looks very unnatural, we use a facial animation which opens and closes the mouth in a loop.

4.3 Client Side – MPML3D-SL System

Different to the AuGe Bands, we have a full script beforehand and no real time processing is necessary. Another big difference in this case is that we use non player

Fig. 5. Two virtual characters using AuGe Bands

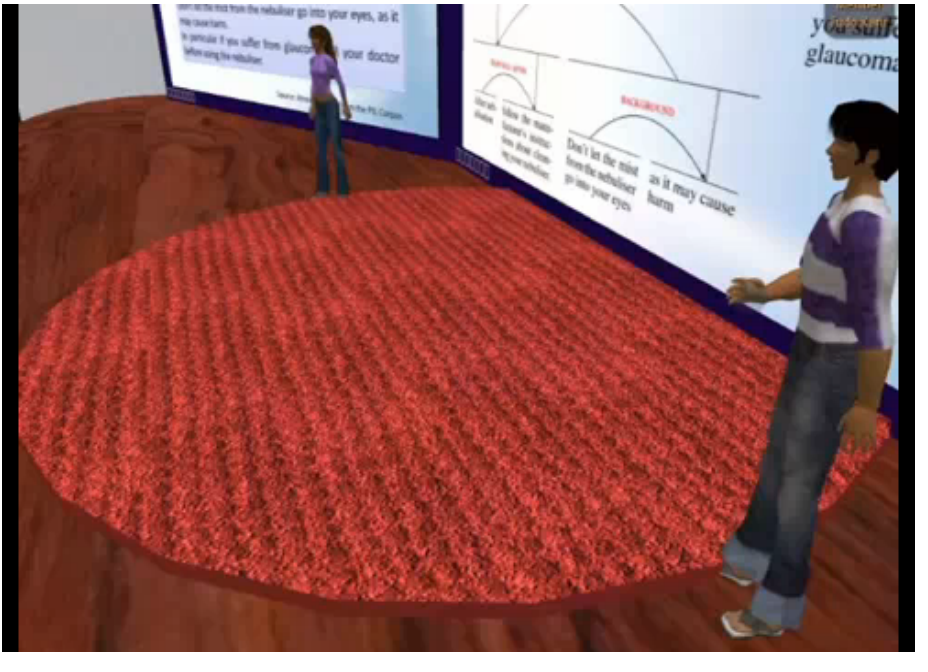

Fig. 6. Two virtual characters controlled by MPML3D-SL

controlled characters also called bots , so there is no interference with a user, also multiple character can be controlled at once , in case of our automatic behavior generation system we have a speaking agent and a listening agent. These bots have to be legal Second Life accounts, which have to be generated before hand, also a certain object has to be attached to them, it is called speech cube and is used for storing and playing the sound files according to the speech acts of the script. Once the character is all set up, the automatically generated script can be loaded using our MPML3D-SL player. The player consists of a backend, handling the script and a frontend that is actually controlling the bot. The frontend interacts directly with the Second Life servers at Linden Labs using a public API. (for more information on MPML3D-SL please refer to 11) This offers a better control of the characters which is more precise in terms of timing of gestures and speech output and also enables us to more actions like walking or sending text via the chat function. Figure 6 shows two virtual characters presenting information about the correct use of a drug.

5 Conclusion

In this paper we described a method to enrich communication in a virtual environment like the metaverse second life by adding gestures. We designed a system that is capable of processing the information in a text message associating non verbal behavior in form of gestures and animating an user controlled avatar in real-time. Our system addresses the problem of populating virtual worlds with “life”, such as agents acting in the roles of guides, clerks or sales staff. It can be used to easily add naturally acting life-like virtual characters to virtual spaces. The added believable nonverbal behaviors can improve the effectiveness of communication in those virtual worlds and provide a more immersive experience for users. And so help to make the emptiness of many virtual worlds a bit more lively and make conversations more enjoyable.

Future work will focus on improving the timing between gestures, lip movement and speech output, a crucial part of further enhancing the naturalness of our virtual characters. As well as adding more precise deictic gestures to our gesture repertoire, to refer to objects in a 3 dimensional world.

References

1. Breitfuss, W., Prendinger, H., Ishizuka, M.: Automatic generation of gaze and gestures for dialogues between embodied conversational agents. Int’l J. of Semantic Computing 2(1), 71–90 (2008)
2. Cassell, J., Vilhjálmsson, H., Bickmore, T.: BEAT: the Behavior Expression Animation Toolkit. In: Proceedings of SIGGRAPH 2001, pp. 477–486 (2001)
3. E.V.A. - Essential Voicechat Advancement by Jarek Dejavu (24.02.2009), http://www.shambles.net/pages/learning/ict/sltools/
4. Gratch, J., Wang, N., Gerten, J., Fast, E., Duffy, R.: Creating Rapport with Virtual Agents. In: Pelachaud, C., Martin, J.-C., André, E., Chollet, G., Karpouzis, K., Pelé, D. (eds.) IVA 2007. LNCS, vol. 4722, pp. 125–138. Springer, Heidelberg (2007)
5. Heylen, D.: Head gestures, gaze and the principles of conversational structure. International Journal of Humanoid Robotics 3(3), 241–226 (2006)

6. Kendon, A.: Some functions of gaze-direction in social interaction. Acta Psychologica 26, 22–63 (1967)
7. Kipp, M.: Creativity meets automation: Combining nonverbal action authoring with rules and machine learning. In: Gratch, J., Young, M., Aylett, R.S., Ballin, D., Olivier, P. (eds.) IVA 2006. LNCS, vol. 4133, pp. 230–242. Springer, Heidelberg (2006)
8. Kopp, S., Tepper, P., Cassell, J.: Towards integrated microplanning of language and iconic gesture for multimodal output. In: Proceedings of the Int. Conf. on Multimodal Interfaces 2004, pp. 97–104. ACM Press, New York (2004)
9. Peters, C., Pelachaud, C., Bevacqua, E., Mancini, M.: A model of attention and interest using gaze behavior. In: Proceedings of 5th International Conference on Intelligent Virtual Agents 2005, pp. 229–240 (2005)
10. Reeves, B., Nass, C.: The media equation: How people treat computers, television and new media like real people and places. CLSI Publications, Stanford (1996)
11. Ullrich, S., Bruegmann, K., Prendinger, H., Ishizuka, M.: Extending MPML3D to Second Life. In: Prendinger, H., Lester, J.C., Ishizuka, M. (eds.) IVA 2008. LNCS (LNAI), vol. 5208, pp. 281–288. Springer, Heidelberg (2008)
12. Vertegaal, R., Weevers, I., Sohn, C., Cheung, C.: Gaze-2: conveying eye contact in Group video conferencing using eye-controlled camera direction. In: Proceedings of the SIGCHI Conference on Human factors in Computing Systems (CHI 2003), pp. 521–528. ACM Press, New York (2003)