Available online at www.sciencedirect.com
SCIENCE DIRECT
Journal of Network and Computer Applications 29 (2006) 83–104
Journal of NETWORK and COMPUTER APPLICATIONS
www.elsevier.com/locate/jnca

Cards-to-presentation on the web: generating multimedia contents featuring agent animations

Yukiko I. Nakano[a,*], Toshihiro Murayama[a], Masashi Okamoto[b], Daisuke Kawahara[b], Qing Li[b], Sadao Kurohashi[b], Toyoaki Nishida[c]

[a]*Japan Science and Technology Agency (JST), Atago Green Hills MORI Tower 18F, 2-5-1 Atago, Minato-ku, Tokyo 105-6218, Japan*

[b]*Graduate School of Information Science and Technology, The University of Tokyo, 7-3-1 Hongo, Bunkyo-ku, Tokyo 113-8656, Japan*

[c]*Graduate School of Informatics, Kyoto University, Yoshida-Honmachi, Sakyo-ku, Kyoto 606-8501, Japan*

Received 1 October 2004; accepted 1 October 2004

Abstract

With the goal of supporting the knowledge circulation and creation process in a society, we have studied story-based communication in a network community. On the basis of this research motivation, this paper proposes a web-based multimedia environment called Stream-oriented Public Opinion Channel (SPOC), which enables novice users to embody a story as multimedia content and distribute it on the Internet. The system produces digital camera work for graphics and video clips and automatically generates agent animations according to linguistic information in a text. The findings of our evaluation experiments show that SPOC is easy for novice users to learn and use, suggesting that this system can reduce a user's cost in creating multimedia content and encourage communication in a network community.

1. Introduction

Multimedia content, such as movies and TV news stories, used to be produced solely by the corporate mass media. Thanks to the significant advances in media and network

* Corresponding author. Tel.: +81 3 5404 2871; fax: +81 3 3432 8851.
E-mail addresses: yukiko@ristex.jst.go.jp (Y.I. Nakano), tmuraya@ristex.jst.go.jp (T. Murayama), okamoto@kc.t.u-tokyo.ac.jp (M. Okamoto), kawahara@kc.t.u-tokyo.ac.jp (D. Kawahara), liqing@kc.t.u-tokyo.ac.jp (Q. Li), kuro@kc.t.u-tokyo.ac.jp (S. Kurohashi), nishida@i.kyoto.ac.jp (T. Nishida).

1084-8045/$ - see front matter © 2005 Elsevier Ltd. All rights reserved.
doi:10.1016/j.jnca.2004.10.006

technologies over the last decade, multimedia equipment (e.g. digital still/video cameras) and Internet access have become available for personal use. This has allowed ordinary people to express their own stories as multimedia content and distribute it on the Internet.

It is still not very easy, however, for nonexpert users to produce their own multimedia content. In embodying their stories as multimedia content, users need to learn how to edit multimedia materials, such as graphics, video clips, audios, and animations. For example, previous studies suggest that synchronized speech and agent animations make learning activities more effective (Craig and Gholson, 2002; van Mulken et al., 1998), but it is almost impossible for ordinary users to create detailed designs for agent animations to be synchronized with speech. In addition, they also need to have the skill to set up streaming of the resulting content to distribute it on the network. Even for expert users, these tasks take enormous effort and time.

Our research goal is to provide an all-in-one web-based application enabling users to easily create and distribute multimedia narrative content so as to facilitate story-based communication within a network community. To accomplish this goal, we propose a multimedia environment, called Stream-oriented Public Opinion Channel (SPOC), and an animated agent system, called Conversational Agent System for neTwork applications (CAST).

First, SPOC is a server system providing the following functions: (1) automatic generation of multimedia story content by integrating speech, graphics, video clips, and agent animations; (2) broadcasting of the contents on a network; and (3) display of such multimedia content on a web browser.

CAST, working as a component of SPOC, creates a storyteller or presenter agent in SPOC. It determines the agent's nonverbal behaviors automatically according to linguistic information in a text. Because of its embodied representation, the animated agent can display nonverbal behaviors (e.g. gestures, eye movements) with its face and body. Therefore, the animated presenter agent is capable of utilizing the same communication modalities as a human in face-to-face communication.

In the rest of this paper, first, we describe the background of this study. Then, we explain the SPOC mechanisms, illustrating how these mechanisms embody a story as multimedia content. We also discuss the results of our preliminary evaluation experiments to ensure that this system is easy to use for nonexpert users. After describing the relevant previous work, we finally summarize the contributions of this work and discuss our future direction.

2. Background

Stories can transfer tacit knowledge and help people understand a collection of events as a coherent unit. One study of cognitive psychology showed that the bulk of human knowledge and memory is communicated and encoded in story form (Schank and Abelson, 1995). As stories seem to play a central role in human memory by providing an organizing structure for new experiences and knowledge, storytelling has been studied in a number of disciplines, including linguistics, psychology, artificial intelligence, human-computer interaction, learning environments, and knowledge management (Aylett,

1999; Bruner, 1990; Lawrence and Thomas, 1999; Mateas and Stem, 2000; Mott et al., 1999).

On the basis of these previous studies, we have proposed the concept of 'Social Intelligence Design' for a network community (Nishida, 2002). Social intelligence design employs story-based communication and conversation to establish mutual understanding and create knowledge in a society (Isaacs, 1996). To support the process of evolving and circulating social intelligence, we have already developed some web applications. Public Opinion Channel (POC) is a participatory broadcasting system that broadcasts questions, opinions, and discussions arising in a community (Fukuhara et al., 2003). This system has the following functions to support knowledge management in a community: (1) interaction with the system by posting messages and stories; (2) viewing of conversational presentations by two embodied agents; and (3) use of a 'Knowledge Card', consisting of a short text (a few sentences) and a graphic image, as the information unit distributed to the community; and (4) collection and classification of information by using keywords contained in a Knowledge Card.

As an extension of the POC system, our next system, EgoChat (Kubota et al., 2002), employs an agent-based approach to facilitate the process of circulating conversational information within a community. Previous studies suggest that animated avatar agents can play an important role in communication in a network community. Avatar agents facilitate seamless communication in video-mediated communication (VMC) (Nakanishi et al., 1996), as well as encourage communication in collaboration systems (Takahashi and Takeda, 2001). On the basis of these works, EgoChat enables personalized, peer-to-peer asynchronous communication. In this system, an embodied agent acting as a virtualized ego talks on a user's behalf. The user can have a conversation with any virtualized ego on the system by using a speech interface. This motivates users to enjoy interaction with the system. In addition, EgoChat supports an information-circulating process within a community by helping the users to generate, improve, integrate, and delete Knowledge Cards.

Our basic idea of story-based communication for network communities has been successfully implemented in these systems. However, the content generated by these systems is quite limited in its expressiveness, since the presentation content consists of a static graphical image and agent animations with a limited range of actions. To improve the expressiveness of content, first, more dynamic and lively visual materials are preferred, because it is hard to keep audience's attention with static visual content (Kraft, 1986). Second, our previous animated agents were not capable of displaying nonverbal communicative behaviors (Cassell et al., 2001). Even when graphics or video clips contain necessary and sufficient information, the presence of a lively speaker presenting well-prepared contents is much more appealing (Andre et al., 2000). To enable an animated agent to perform meaningful actions as a presenter, a more sophisticated agent system is necessary. Addressing these issues, we designed our new system, SPOC, by focusing on the following capabilities:

1. Streamed video clips are available as visual materials.
2. Camera work, such as zoom and pan, is automatically applied to visual materials, including both graphics and video clips.

3. The gestures and facial expressions of an animated agent are automatically selected and generated.

The following sections describe the details of SPOC's implementation and show how this system exceeds the capabilities of previous systems.

3. SPOC

The SPOC components and its functions are illustrated in Fig. 1. SPOC users can (a) edit a SPOC program with the Knowledge Card Editor, (b) play a program through the SPOC Viewer, and (c) ask the system a question. User can post questions while watching a program. If the Question-Answering (QA) Module (Kiyota et al., 2002) finds an answer for a question, it sends a SPOC program with the answer back to the Viewer, and the program is played on the Viewer. In the following subsections, in addition to describing the system architecture and the Knowledge Cards, we focus on the (a) editing and (b) playback functions.

3.1. SPOC system architecture

To provide an environment for creating, editing, posting, and playing multimedia content without any software installation, all the system components run on the server side. Fig. 2 shows an overview of the SPOC architecture. It consists of a web server and three back-end servers: the database server, the streaming server, and the speech synthesis server.

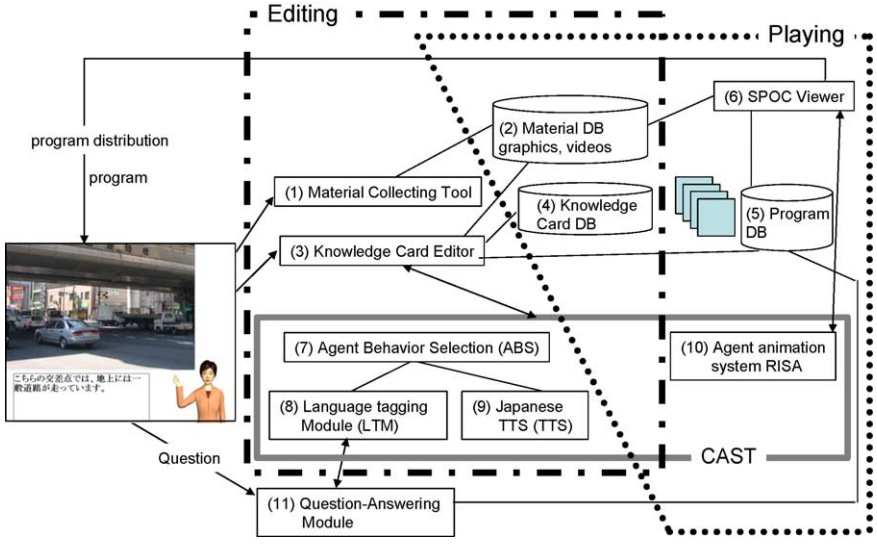

Fig. 1. SPOC functions and components.

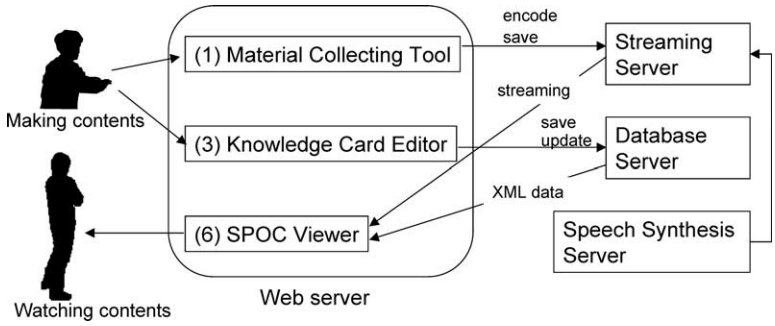

Fig. 2. SPOC system overview.

The web server, using session management, provides three web applications to the users: the Material Collecting Tool, the Knowledge Card Editor, and the SPOC Viewer. Each is connected to the back-end servers. The streaming server converts and saves video clips, which are uploaded by users through the Material Collecting Tool, as streaming data. It also serves streams of video clips and speech sounds to the users through the SPOC Viewer. The database server maintains XML data for constructing multimedia presentations, which are created and modified by the users through the Knowledge Card Editor. The speech synthesis server creates audio files for agent speech by accessing a text-to-speech engine.

Employing this system architecture, SPOC enables users to enjoy a web-based multimedia content service without installing any software, such as a multimedia authoring tool, animation software, or a text-to-speech engine.

3.2. Editing SPOC contents

In SPOC, a user's story is embodied as a SPOC program, which is like a TV program. Users can create their own programs by using the Knowledge Card Editor (Editor), and they can post programs on the web. Visual materials, such as graphics and video clips, are uploaded and encoded through the Material Collecting Tool and then stored in the Streaming Server.

A SPOC program consists of a sequence of Knowledge Cards (Cards), as shown in Fig. 3. Each Card is like a scene in a TV program, and a user edits Cards one by one. Thus, a Card is a building block for composing a story. Users can create different stories by changing the order of the Cards. A snapshot of the Editor is shown in Fig. 4. In editing a Card, a user only needs to do the following two things:

(I) Edit visual materials by first selecting a file from a menu, and then specifying the zoom scale and the position of the focused area. For example, in Fig. 4, the user has focused and zoomed in on Target A in the original picture. The user can do this procedure intuitively by manipulating a GUI (zoom bar). If the selected material is a

Fig. 3. Sequence of knowledge cards representing a SPOC program.

movie file, the user can extract part of the video clip by specifying the start and end frames in the video.

(II) Type the text to be uttered by the animated agent.

Step (I) specifies the camera work, which is automatically generated by the SPOC Viewer, and step (II) triggers automatic script generation by CAST for the agent behavior. These steps are described in detail in Sections 3.5.1 and 3.3, respectively.

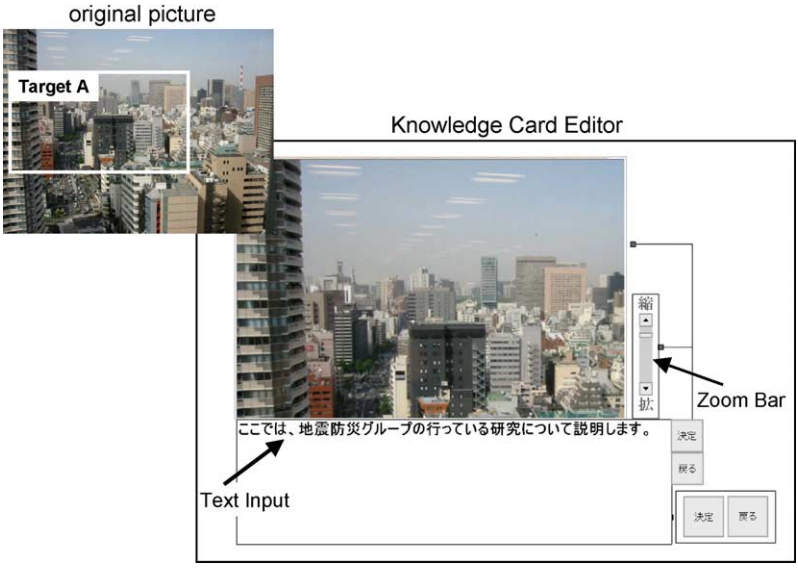

Fig. 4. Knowledge card editor.

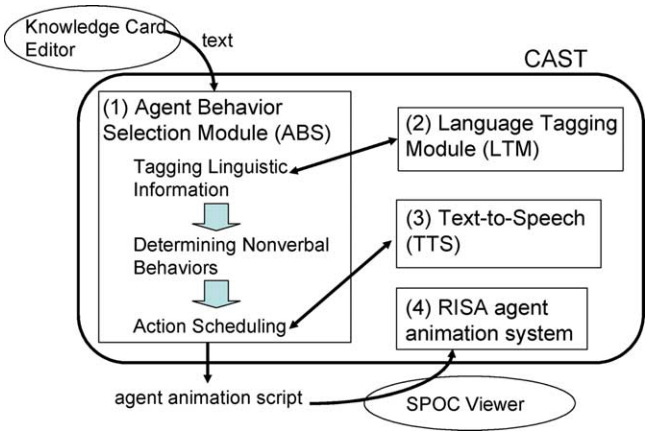

Fig. 5. CAST architecture.

3.3. Determining agent behaviors

This subsection describes an animated agent system, CAST. It is triggered by the Editor to calculate an agent animation schedule and produce a synthesized voice for the agent.

As shown in Fig. 5, CAST consists of four main modules: (1) the Agent Behavior Selection Module (ABS), (2) the Language Tagging Module (LTM), (3) a Text-to-Speech engine (TTS), and (4) a Flash-based character animation system, RISA (RIStex animated Agent system). When CAST receives a text input, it sends the text to the ABS. The ABS selects appropriate gestures and facial expressions according to linguistic information calculated by the LTM. Then, the ABS obtains timing information by accessing the TTS, and it calculates a time schedule for the set of agent actions. The output from the ABS is a set of animation instructions that can be interpreted and executed by the RISA animation system.

3.3.1. Agent behavior selection module

The input to the ABS is plain text, while the output is a set of instructions for RISA. While the system architecture for the ABS is similar to that described by Cassell et al. (2001), we propose more detailed gesture generation rules specifically for Japanese. Unfortunately, there has been little work in computational linguistics on how to identify and extract linguistic information from text in order to generate gestures. Addressing this problem, we have designed and developed a mechanism that determines appropriate agent behaviors according to the linguistic information contained in Japanese text. For this purpose, the following mechanisms are necessary:

- Annotating Japanese linguistic information useful for determining nonverbal behaviors.
- Assigning nonverbal behaviors by applying behavior selection rules to the annotated Japanese text.

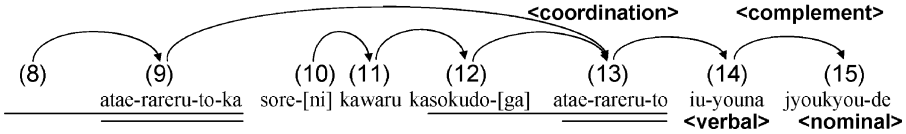

Fig. 6. Sample analysis of syntactic dependency and gesture occurrence.

We describe these processes in detail below.

3.3.1.1. Tagging linguistic information. We have collected and analyzed presentations by seven people and identified nine features ((a)–(i) below) as factors for predicting gesture occurrence. Fig. 6 shows a sample of our data. Gesture strokes occur at the double-underlined positions. This example shows that gestures co-occur with parallel phrases constructing a coordination. The details of the linguistic issues related to the features and the results of data analysis are described by Nakano et al. (2004).

We use a Japanese syntactic dependency analyzer (Kurohashi and Nagao, 1994) as the LTM to parse the input text, and we calculate the following nine features for each *bunsetsu* unit.[1,2]

(a) If it is a nominal phrase (NP), is it modified by a clause or a complement?
(b) If it is an NP, what type of postpositional particle marks its end (e.g. 'wa' (topic marker), 'ga', 'wo')?
(c) Is it a wh-interrogative?
(d) Have all the content words in the *bunsetsu* unit been mentioned in a preceding sentence? (judgment of old/new information)
(e) Is it a constituent of a coordination?

In addition to these five features, we noticed that some lexical entities frequently co-occurred with a gesture in the human presentation data that we collected. Therefore, we used the syntactic analyzer to annotate additional lexical information based on the following questions.

(f) Is the *bunsetsu* unit an emphatic adverbial phrase (e.g. very, extremely), or is it modified by a preceding emphatic adverb (e.g. very *important* issue)?
(g) Does it include a cue word (e.g. now, therefore)?
(h) Does it include a numeral (e.g. *thousands* of people, 99 times)?
(i) Does it include a demonstrative (e.g. this, these)?

For example, *bunsetsu* (9) in Fig. 6 has the following feature set:

[1] A '*bunsetsu* unit' in Japanese corresponds to a phrase in English, such as a noun phrase or a prepositional phrase.

[2] To avoid the effects of parsing problems, errors in the syntactic dependency analysis were manually corrected for about 13% of the data.

Table 1
Gesture decision rules

Case ID	Case			Frequency per bunsetsu unit
[C1]	Quantity of modification	(a) NP modified by a clause		0.382
[C2]		Pronouns, other type of NPs	(b) Case marker = "wo" and (d) new information	0.281
[C3]	(c) WH-interrogative			0.414
[C4]	(e) Coordination			0.477
[C5]	Emphatic adverb	(f) Emphatic adverb itself		0.244
[C6]		(f) Following an emphatic adverb		0.350
[C7]	(g) Cue word			0.415
[C8]	(h) Numeral			0.393
[C9]	(i) Demonstrative			Deictic gesture
[C10]	Other (baseline)			0.101

{Text-ID: 1, Sentence-ID: 1, Bunsetsu-ID: 9, Govern: 8, Depend-on: 13, Phrase-type: VP, Linguistic-quantity: NA, Case-marker: NA, WH-interrogative: false, Given/New: new, Coordinate-with: 13, Emphatic-Adv: false, Cue-Word: false, Numeral: false}.

The text ID of this *bunsetsu* unit is 1, the sentence ID is 1, and the *bunsetsu* ID is 9. This *bunsetsu* governs *bunsetsu* 8 and depends on *bunsetsu* 13. It conveys new information and together with *bunsetsu* 13 forms a parallel phrase.

3.3.1.2. Assigning nonverbal behaviors. Next, for each *bunsetsu* unit, the ABS decides whether to assign a nonverbal behavior. The decision rules for gestures, listed in Table 1, are based on our empirical results (Nakano et al., 2004). For example, *bunsetsu* unit (9) shown above matches case C4 in Table 1, where a *bunsetsu* unit is a constituent of a coordination. In this case, the system assigns a gesture to the *bunsetsu* unit with a 47.7% probability. In the current implementation, if a specific gesture for an emphasized concept (e.g. a gesture animation expressing 'big') is defined in the gesture animation library, it is preferred to a 'beat gesture' (a simple flick of the hand or fingers up and down) (McNeill, 1992). If a specific gesture is not defined, a beat gesture is used as the default.

3.3.1.3. Adding highlights to gestures. As an optional function in CAST, we have also built a component called the Enhancement Generator that automatically adds highlighting animations to agent gesture animations after the gestures are determined by the ABS (Li et al., 2004). We propose two types of highlighting methods to emphasize synchronization between verbal (speech) and nonverbal (gesture) behaviors.

(1) *Superimposition with beat gesture*. Beat gestures simply emphasize one part of an utterance without representing the meaning of a word. To visualize synchronization between the emphasized words and a beat gesture, the Enhancement Generator adds a superimposition of the emphasized words to the agent's beat gesture animation. Fig. 7(a) shows an example of superimposition with a beat gesture.

Fig. 7. Examples of (a) superimposition and (b) illustrative animation representing 'increase'.

(2) *Illustrative animation with metaphoric gesture*. When a specific shape of gesture is assigned to a metaphoric gesture, it is emphasized by illustrative animations, such as an arrow and a line. If the emphasized concept implies motion or movement, such as 'increase' or 'decrease,' the direction of the movement is illustrated by an arrow animation. An example of highlighting the concept of 'increase' is shown in Fig. 7(b). If the emphasized concept expresses a static state, a motionless picture is used to emphasize the gesture. For example, when the agent is performing a 'long' gesture, a rectangle shape is shown near the agent's hands to emphasize the length.

The output of the ABS is stored in XML format. The type of action and the start and end of the action are specified by XML tags. In the example shown in Fig. 8, the agent performs contrast gestures at the second and sixth *bunsetsu* units and a beat gesture at the eighth *bunsetsu* unit. The latter gesture is accompanied by a highlighting animation, which is specified by an Enhance tag. In this example, a superimposition of the emphasized word, 'jyoukyou-de', is added to the beat gesture animation. The ABS also assigns facial

```
[1]   shindo-ga
          <Gesture_right type="contrast" handshape_right="stroke1@2">
[2]       atae-rareru-to-ka
          </Gesture_right>
[3]   sore-ni
[4]   kawaru
[5]   kasokudo-ga

          <Gesture_right type="contrast" handshape_right="stroke2@2">
[6]       atae-rareru-to
          </Gesture_right>
[7]   iu-youna
           <Enhance type="superimposition" text="jyoukyou-de" layout="right">
          <Gesture_right type="beat" handshape_right="stroke1">
[8]       jyoukyou-de
          </Gesture_right>
          </Enhance>
      …
```

Fig. 8. Example of CAST output.

expressions. An eyebrow raise co-occurs with a gesture. A head nod and blink occasionally occur at the end of an utterance.

3.3.2. Action scheduling

After determining the nonverbal behaviors, the next step is to generate a time schedule to be executed by the animation system. To synchronize an agent's speech with nonverbal behaviors, the Scheduling Module in the ABS accesses the TTS engine to obtain the timing information for each phoneme (phoneme type, start time, and duration) and the *bunsetsu* boundaries. At this point, a synthesized voice produced by the TTS engine is saved in the streaming server. A viseme[3] for lip-sync process is assigned according to the phoneme type. The output of the Scheduling Module is formatted as a set of instructions to be executed by the RISA animation system. Each command in the instruction set specifies an action type and the start time of the animation. An example of an instruction set is shown below:

```
<START AID="A669" ACTION="GESTURE_RIGHT" TYPE="DEICTIC"
HANDSHAPE_RIGHT="POINTING" SRT="2.88">
<START AID="A671" ACTION="EYEBROWS" SRT="2.88">
<START AID="A673" ACTION="VISEME" TYPE="D" SRT="2.88">
⋮
<START AID="A679" ACTION="VISEME" TYPE="O" SRT="3.25">
<START AID="A680" ACTION="VISEME" TYPE="E" SRT="3.36">
<STOP AID="A671" ACTION="EYEBROWS" SRT="3.40">
<STOP AID="A669" ACTION="GESTURE_RIGHT" TYPE="DEICTIC"
HANDSHAPE_ RIGHT="POINTING" SRT="3.40">
```

START or STOP at the beginning of a command indicates whether the command starts or stops the action. AID indicates the action ID. The ACTION attribute specifies the type of action, such as GESTURE_RIGHT or VISEME. For VISEME, the viseme type is specified by the TYPE attribute. For example, TYPE="D" indicates that the lip shape for the "D" sound should be used. SRT specifies the time at which the command should be executed. For example, in the action AID="A669", a right-hand pointing gesture starts at 2.88 s, and the hand returns to the original position at 3.40 s. Finally, the animation instruction set is sent back to the Knowledge Card Editor and saved in XML format.

3.4. Structure of a knowledge card

Next, we reconsider the whole process of producing content by describing the structure of a Knowledge Card. As a result of the two-step Editor procedure described in Section 3.2, a Knowledge Card is automatically generated and saved in the Knowledge Card DB in XML format. An example of the XML code is shown in Fig. 9.

[3] A viseme is a generic facial image that can be used to describe a particular sound. A viseme is the visual equivalent of a phoneme or unit of sound in spoken language.

```
<?xml version="1.0" encoding="Shift_JIS" ?>
<cards id="sgkc4lwPh4SEl9js" user_id="wall">
<card>
<id>sj8SWPc9mnBrS4x1</id>
<box>1</box>
<image>
<imageurl>http://000.000.00.000:8080/sgkc4lwPh4SEl9js/s1.jpg</imageurl>
<xscale>65.9958</xscale>
<yscale>65.9958</yscale>
<xpos>0.0</xpos>
<ypos>0.0</ypos>
<inipos>0</inipos>
<endpos>0</endpos>
</image>
<AGENT SCENE="sj8SWPc9mnBrS4x1" UtrID="0">
<Action ID="1650000" Srt="0.0" />
<Action ID="1680000" Srt="0.0" />
<Action ID="18500" Srt="0.0" />
<Action ID="188" Srt="0.02002" />
<Action ID="189" Srt="0.06108102000000001" />
<Action ID="188" Srt="0.13021110102000003" />
<Action ID="189" Srt="0.21342431212102" />
<Action ID="188" Srt="0.289713736433141" />
<Action ID="186" Srt="0.3240374501695742" />
<Action ID="185" Srt="0.4282236789904709" />
<Action ID="18500" Srt="0.5467699026694613" />
                    .
                    .
                    .
<comment>ここでは、地震防災グループの取り組んでいる、
                事実の明示化に関する研究を紹介します。</comment>
"I will present our research on fact visualization, which is conducted in the Earthquake
Disaster Prevention Research Group."
```

Fig. 9. Example of a Card in XML format.

＜CARDS＞ represents the beginning of a new program. This consists of CARD elements. A ＜CARD＞ element is the building block of a program and is composed of ID, BOX, IMAGE, AGENT, and COMMENT elements. ＜ID＞ specifies the ID of the CARD. ＜BOX＞ specifies the order of the card in a program. An ＜IMAGE＞ element consists of several sub-elements specifying the visual material in detail. ＜IMAGEURL＞ specifies the URL address where the graphics and video clips are stored. ＜XSCALE＞ and ＜YSCALE＞ specify the horizontal and vertical zoom scales, respectively, as percentages (%). ＜XPOS＞ and ＜YPOS＞ specify the horizontal and vertical positions, respectively, of the material in a display. ＜INIPOS＞ and ＜ENDPOS＞ specify the respective start and end frames of video material. In the case of a graphic image, the value of the data is '0'. The data for these tags are specified while editing the visual material (Section 3.2). An ＜AGENT＞ element contains the set of animation instructions generated by CAST (Section 3.3). Finally, ＜COMMENT＞ indicates text in the card.

When the SPOC Viewer plays a Card, it interprets the XML tags and displays all the materials according to the instructions. By repeating this process, SPOC generates a multimedia presentation from a sequence of Cards. The details of this process are described in the next subsection.

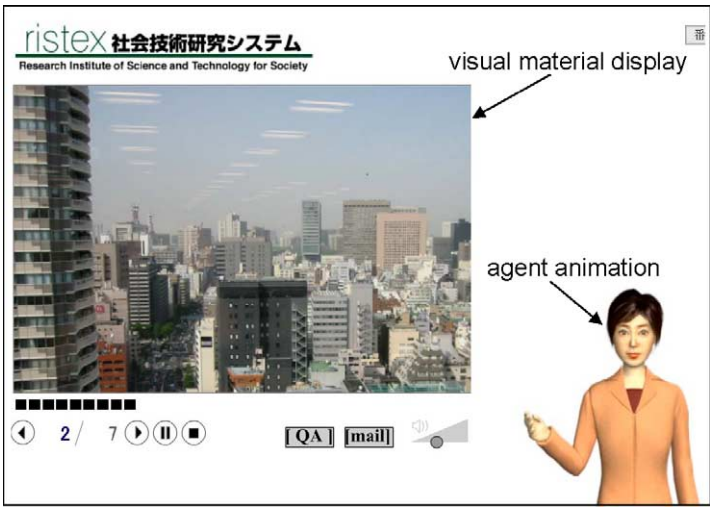

Fig. 10. Screen shot of the SPOC viewer.

3.5. Viewing SPOC contents

Using the SPOC Viewer (Viewer), users can watch programs posted by other community members. The input to the Viewer is a set of Knowledge Cards created by the Editor. The Viewer produces an audio-visual stream by playing each Card in a program one by one. A screen shot of the SPOC Viewer is shown in Fig. 10. The Viewer consists of a visual material display and an animated agent. The visual material display shows a graphic image or a video clip. The agent animations are generated by the Flash-based animation system RISA, described in Section 3.5.2.

3.5.1. Automatic camera work generation

The Viewer automatically generates digital camera work. In playing a program, the Viewer compares two consecutive cards. If both cards use the same original visual material, camera work is automatically applied. The Viewer calculates the difference in zoom scale and focused area position between the Cards. It then gradually changes the zoom scale and the focused area from one Card to another.

For example, as shown in Fig. 11, Cards 1 and 2 use the same image. Card 1 is focused on Target A. This information is saved in a Knowledge Card XML:

```
<XSCALE>150</XSCALE>
<YSCALE>150</YSCALE>
<XPOS>20</XPOS>
<YPOS>100</YPOS>
```

Likewise, Card 2 is focused on Target B:

```
<XSCALE>220</XSCALE>
<YSCALE>220</YSCALE>
```

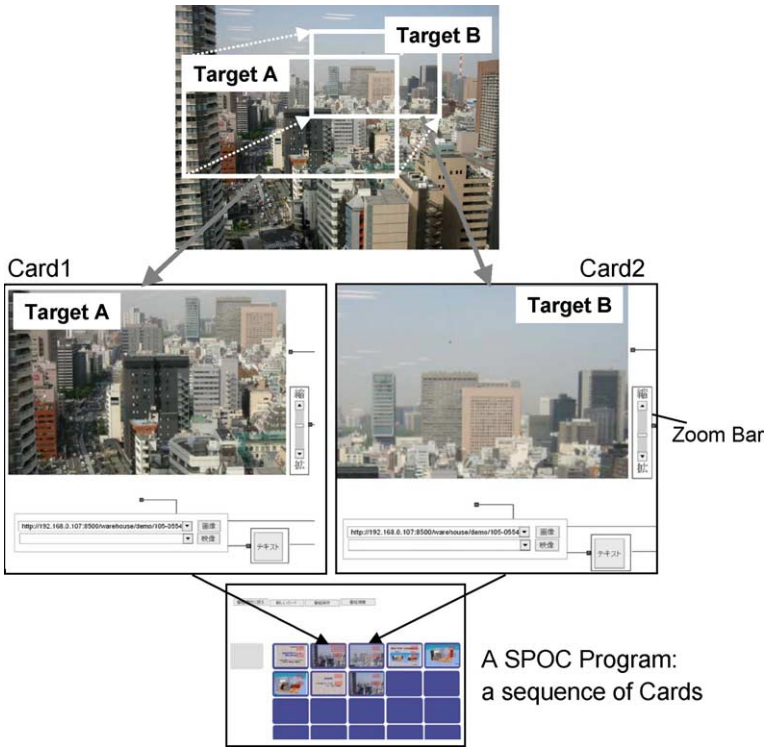

Fig. 11. Example of camera work in the SPOC viewer.

<XPOS>200</XPOS>
<YPOS>40</YPOS>

Assume that Cards 1 and 2 are arranged side-by-side in the program, and that the drawing sampling rate is 1/12 s. The Viewer then draws the visual material every 1/12 s while changing the X position of the material by $(200-20)/12=15$ pixels and the Y position by $(40-100)/12=-5$ pixels. The same algorithm is applied to calculate the scale of the material, which is gradually changed from 150 to 220%. As a result, it appears in the Viewer as if a camera moves from Target A to Target B while zooming in on Target B. This technique can also be applied to a video clip.

The advantage of this method is that users need not design the camera work itself. They simply need to change the scale and position of a picture intuitively by manipulating a GUI. In TV programs, camera work (e.g. zoom, pan, tilt) is frequently used to improve the comprehensibility of a program. With the technique proposed here, such useful camera work is automatically generated in SPOC programs.

3.5.2. Playing agent animations

When all the animations and the synthesized voice are ready to play, the Viewer sends a cue to start playing them in a synchronized manner. To control the animated character

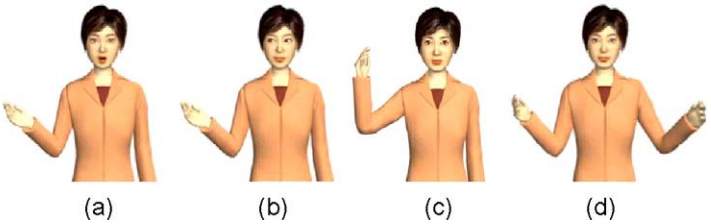

Fig. 12. Screen shots of RISA (a) performing a beat gesture, (b) looking away, (c) pointing at the visual material display, and (d) performing a 'big' iconic gesture.

through a web-based application, we implemented the RISA animation system in Macromedia Flash. Snapshots of RISA gesturing and changing her facial expression are shown in Fig. 12.

The basic idea of this animation system is to construct an agent animation by assembling small animations of each body part. The agent body is divided into 12 parts: head, two eyebrows, two eyes, two eyeballs, mouth, two arms, and two hands. Small pieces of animations are defined for each body part (e.g. moving the left eyebrow up 30 degrees, moving the right arm in front of the body). The total number of actions in the library is over 300, including reverse actions for the hand gestures. By combining these animations, various kinds of agent behaviors can be produced.

The SPOC Viewer generates agent animations by executing the agent action script specified in the <AGENT> element of a Knowledge Card XML. Fig. 13 shows how the SPOC Viewer synchronizes the animations with speech. The agent speech is saved in the streaming server as an MP3 audio file, and it is played through an MP3 audio player. The Viewer accesses the MP3 player to get the current position in the audio file. If the

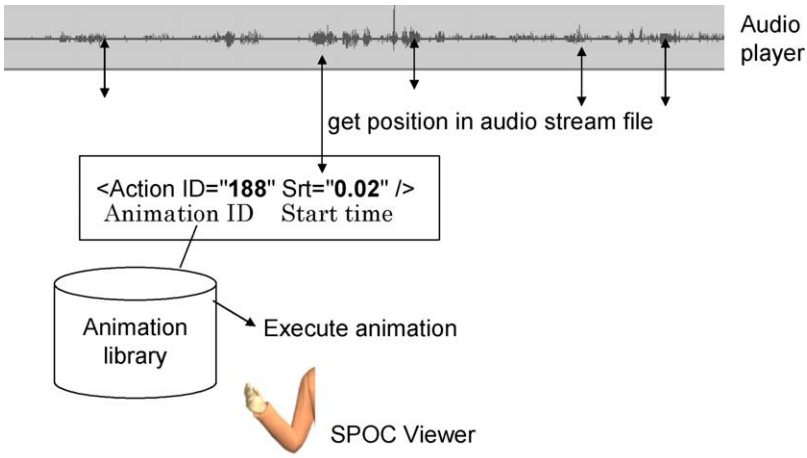

Fig. 13. Synchronization of audio and animation.

current time matches the start time of an action, the Viewer picks up the animation for the action ID from the animation library and executes it. In addition to actions co-occurring with speech, RISA also performs some idle actions, such as looking away and blinking, during silent periods.

4. Evaluation

Our main concern in designing the SPOC system is to help users create multimedia story content without writing a script for a whole program. To accomplish this goal, we have proposed a very simple Knowledge Card editing interface. In this section, based on the results of two experiments, we examine whether SPOC is actually simple and easy enough for novice users to create multimedia content.

4.1. Experiment 1

As a preliminary study, four subjects (two men and two women) learned how to edit a Card, and the times required for them to perform tasks were measured. In this experiment, the subjects were required to edit Cards correctly according to recipes for the Cards.

4.1.1. Procedure

At the beginning of a session, an experimenter instructed the subjects on how to use the Knowledge Card Editor. After the instruction, each subject performed a task by herself/himself.

4.1.2. Task

The subjects' task was to make a 1-min SPOC program about earthquake simulation with seven Knowledge Cards. For each Card, a recipe was provided to the subject. The recipe specified the name of a graphics/movie file to be selected, the text to be typed in, and a picture showing the zoom scale and the focused area. The average length of text in the Cards was 30 characters. The recipes for Cards 2 and 3 are shown in Fig. 14. These two Cards used the same graphics, but the zoom scales and focused areas were different. Thus, digital camera work would be applied to these Cards. Likewise, camera work for a video clip would be applied to Cards 5 and 6.

4.1.3. Results

The mean length of the instruction session was 4:32, and the average time for the subjects to perform the task was 13:30. Thus, on average, a subject made a 1-min program in 13.5 min, after 4.5 min of instruction. Moreover, all of the subjects edited one Card in less than 2 min. These results suggest that learning the Card editing process was quite easy for all of the subjects. The operation time, however, does not indicate the users' impressions of the software. We therefore addressed this issue in Experiment 2.

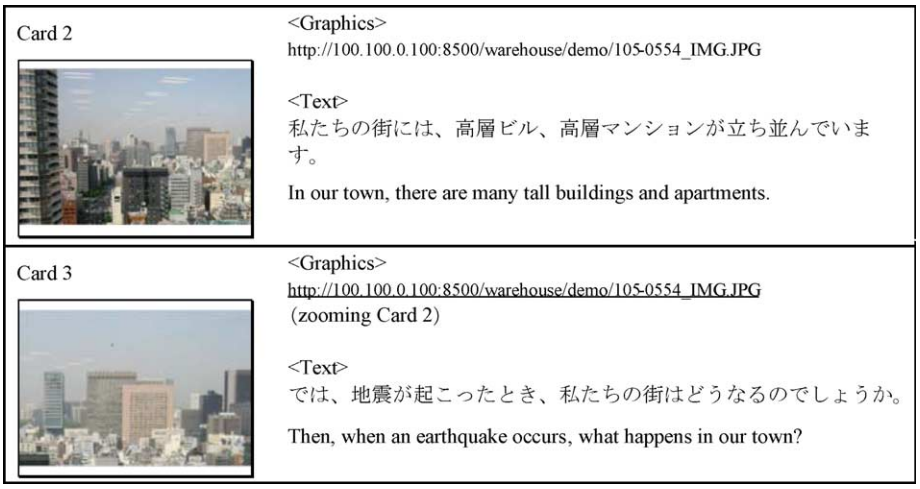

Fig. 14. Examples of knowledge card recipes.

4.2. Experiment 2

To examine users' subjective impressions of SPOC, we gathered another set of subjects (nine people: two women and seven men) and asked them to answer a questionnaire after creating an original SPOC program.

4.2.1. Procedure

At the beginning of a session, each subject was provided the same instruction as in Experiment 1. Each subject then edited a short practice program, which consisted of two Cards using the same graphical material and one Card using a video clip (i.e. Cards 2, 3, and 6 in Experiment 1).

4.2.2. Task

After practicing, the subjects were asked to create their own original program about 'Tokyo' with at least four Cards. As visual materials, 23 graphics files and 7 movie clips of Tokyo were provided. After creating their programs, the subjects answered a questionnaire asking about their impressions of watching the programs and using the SPOC system. The questions are listed in Table 2. The subjects answered these questions on a four-point scale.

4.2.3. Results

The mean length of the programs was 1 min, 13 s (SD = 10.8 s) with 6.2 Cards. All of the subjects used video clips as well as graphics. On an average, a video clip was used in 1.8 Cards per program. The users' general impressions of the software are shown in Fig. 15. The subjects felt that the sequence of operations was not complicated (Q1, Q2), nor did it take time to learn (Q3). They also felt that SPOC was useful, and that it was comfortable and enjoyable to use (Q4, Q5). They also exhibited positive attitudes about

Table 2
Questions about users' general impressions of SPOC

(Q1) The sequence of operation procedures was not complicated
(Q2) SPOC was easy to use
(Q3) It would not be long before I could use this software perfectly
(Q4) Creating a program with SPOC was enjoyable
(Q5) SPOC is useful
(Q6) SPOC can convey information comprehensibly
(Q7) SPOC can convey information accurately
(Q8) If I have the chance, I would like to use this software again
(Q9) If I familiarize myself with this software, I will be able to make more interesting programs than I made this time

using the software again (Q8, Q9). An interesting finding is that the subjects judged SPOC as conveying information more comprehensibly (Q6), rather than accurately (Q7). In addition to the general impressions, we also asked about usability of each operation. We found that some operations, such as file selection and time range specification for a video clip, still need to be improved.

4.2.4. Discussion

The results strongly support our approach. After receiving brief instructions, the subjects succeeded in not only editing Cards correctly according to recipes, but also in creating their own programs. In Experiment 2, all of the subjects created highly original, well-structured programs. The stories expressed the subjects' opinions about Tokyo, as well as their personal lives and experiences there. We did not, however, examine how well these programs would have been received by community members. Through further work, we need to evaluate SPOC programs from the recipient point of view and examine how they affect communication in a community.

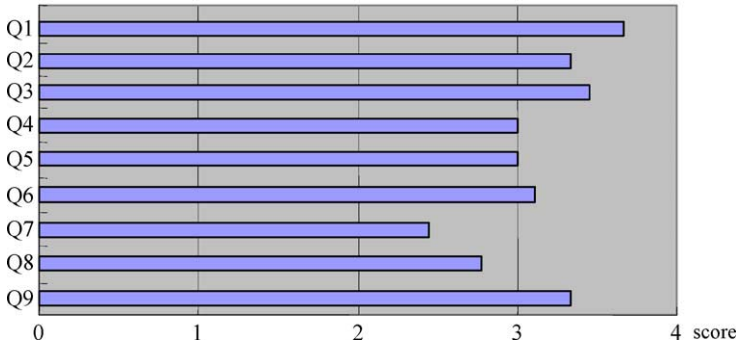

Fig. 15. Results of the subjective evaluation.

5. Related work

5.1. Methods for multimedia contents generation

Multimedia content generation has been studied mainly from two different approaches: the scripting approach, and the AI-based approach. We thus compare these approaches with our method used in SPOC and CAST.

In the scripting approach, content developers create scripts for multimedia content by using markup languages, such as SMIL (The Synchronized Multimedia Integration Language), MPML (Ishizuka et al., 2000), and TVML (Hayashi et al., 1999). SMIL is a markup language for describing synchronized multimedia in general. MPML defines a set of tags for controlling a Microsoft Agent. TVML is a markup language for creating a TV program with a background setting, character animations, and camera work. Although these markup languages provide specifications for designing multimedia content in detail, content creators need to be skillful and patient enough to describe every single piece of the content, including when and how camera work is applied, and when and what type of nonverbal behaviors an animated agent should perform.

The other method is the AI-based approach, in which multimedia content is generated using artificial intelligence techniques such as planning (Andre et al., 1999; Rickel and Johnson, 1999; Shaw et al., 1999). This approach has developed highly sophisticated methods for automatic presentation generation. However, it also requires content creators to be skilled at describing relationships between events and actions in logical form.

In contrast, the combination of SPOC and CAST enables users to create multimedia content without scripting all the details or learning logic and AI theory. SPOC automatically generates camera work by using the size and position information of the visual material in a Knowledge Card. CAST calculates an agent's action script from text by analyzing linguistic information. Using this environment, users can easily create multimedia content with less effort than in the scripting approach and the AI-based approach.

In addition, SPOC employs Knowledge Cards as the building blocks of programs, while other markup languages provide GUI-based program editors for editing the whole sequence of a program (e.g. TVML Editor). We suggest that the Card interface makes it easier for nonexpert users to construct programs from small pieces. On the other hand, a program editor allows more professional users to design programs in more detail. Thus, choosing an appropriate system according to the purpose of communication is important.

5.2. Web-based presentation agent

With the goal of providing a media technology that does not depend on either computer performance or platform, we have implemented our system as a web application.

Because web-based applications have become so popular, a newscaster agent on a web TV has actually been implemented for commercial use (ANANOVA). This system employs a scripting approach, so that all of the agent's behaviors are described by content designers. As more basic research, multimodal web-based presentation generation has been studied based on the AI approach. For example, in PPP persona (Andre et al., 1997),

multimodal help instructions are generated and presented on the web by an animated agent. Adele, developed at USC, is a pedagogical agent working on a web-based medical education system (Shaw et al., 1999). Note that these systems are designed to help users learn something by watching multimedia content. In contrast, SPOC-CAST aims to help users not only in viewing multimedia content but also in creating their own content.

6. Conclusion and future work

This paper has described a web-based multimedia environment, SPOC. The system allows nonexpert users to create multimedia story content, like a TV program, and to play programs posted by other community members. SPOC can use both video clips and graphics as visual materials, and automatically generates digital camera work from these materials. Moreover, by integrating CAST, it automatically determines and generates agent animations based on linguistic information in a text. Our evaluation experiments showed that SPOC is easy to learn and use for creating and playing programs. These results suggest that SPOC can contribute to reducing the volume of user tasks in creating multimedia content, while also encouraging users to spend more time engaged in communication with other community members.

Although the evaluation experiments focused on content creation, evaluating SPOC from the audience's perspective will be an important future work. For example, it is important to investigate whether a SPOC program can affect a viewer's attitude, and how viewers respond to a program that they have watched. Such bi-directional evaluation will be necessary to improve the communication functionality of this system. In addition, it is also necessary to evaluate the effectiveness of nonverbal agent behaviors in actual human-agent interaction. We expect that if CAST can generate nonverbal behaviors with appropriate timing for emphasizing important words and phrases, users will perceive agent presentations as lively and comprehensible. An important future direction for our research will be conducting a user study to examine this hypothesis.

In addition, we plan to enhance CAST by incorporating more general discourse-level information, though the current system does exploit cue words as a very partial kind of discourse information. For instance, gestures frequently occur at discourse boundaries. Pushing and popping a discourse segment (Grosz and Sidner, 1986) may also affect gesture occurrence. Therefore, by integrating a discourse analyzer into the Language Tagging Module (LTM), more general discourse information can be used in the CAST mechanism.

Acknowledgements

We thank the anonymous reviewers for reading this paper. Special thanks also to Ken'ichi Matsumura for his advice on the evaluation experiments. This study was supported by Mission-oriented Program I in RISTEX, which was established under the joint auspices of the Japan Science and Technology Corporation (JST) and the Japan Atomic Energy Research Institute (JAERI).

References

ANANOVA. http://www.ananova.com.

Andre E, Rist T, Muller J. WebPersona: a life-like presentation agent for the World-Wide Web. Proceedings of IJCAI97, Nagoya, Japan 1997.

Andre E, Rist T, Muller J. Employing AI methods to control the behavior of animated interface agents. Appl Artif Intell 1999;13:415–48.

Andre E, Rist T, Mulken Sv, Klesen M, Baldes S. The automated design of believable dialogues for animated presentation teams. In: Cassell J et al, editor. Embodied conversational agents. Cambridge, MA: MIT Press; 2000. p. 220–55.

Aylett R. Narrative in virtual environments—towards emergent narrative. In: Proceedings of the narrative intelligence, AAAI Fall symposium 1999. AAAI Press; 1999. p. 83–6.

Bruner JS. Acts of meaning. Boston: Harvard University Press; 1990.

Cassell J, Bickmore T, Campbell L, Vilhjalmsson H, Yan H. More than just a pretty face: conversational protocols and the affordances of embodiment. Knowl Based Syst 2001a;14:55–64.

Cassell J, Vilhjalmsson H, Bickmore T. BEAT: the behavior expression animation toolkit. In: Proceedings of SIGGRAPH 01, Los Angeles, CA. ACM Computer Graphics Press; 2001. p. 477–86.

Craig SD, Gholson B. Does an agent matter? The effects of animated pedagogical agents on multimedia environments. In: Proceedings of ED-MEDIA 2002: world conference on educational multimedia, hypermedia and telecommunications; 2002. p. 357–62.

Fukuhara T, Fujihara N, Azechi S, Kubota H, Nishida T. A network-based interactive broadcasting system for supporting a knowledge-creating community. In: Howlett RJ et al, editor. Internet-based intelligent information processing systems. Singapore: World Scientific; 2003. p. 227–68.

Grosz B, Sidner C. Attention, intentions, and the structure of discourse. Comput Linguist 1986;12(3):175–204.

Hayashi M, Ueda H, Kurihara T. TVML (TV program making language)—automatic TV program generation from text-based script. Proceedings of Imagina'99 1999;31–42.

Isaacs WN. The process and potential of dialogue in social change. Educ Technol 1996;Jan./Feb.:20–30.

Ishizuka M, Tsutsui T, Saeyor S, Dohi H, Zong Y, Prendinger H. MPML: a multimodal presentation markup language with character agent control functions. In: Proceedings of agents2000 workshop 7 on achieving human-like behavior in interactive animated agents, Barcelona, Spain; 2000. p. 51–5.

Kiyota Y, Kurohashi S, Kido F. 'Dialog navigator': a questions answering system based on large text knowledge base. Proceedings of the 19th international conference on computational linguistics (COLING 2002), Taipei 2002;460–6.

Kraft R. The role of cutting in the evaluation and etention of film. J Exp Psychol Learn Mem Cogn 1986;12(1): 155–63.

Kubota H, Kurohashi S, Nishida T. Virtualized-egos using knowledge cards. Proceedings of seventh pacific rim international conference on artificial intelligence (PRICAI-02) WS-5 international workshop on intelligent media technology for communicative reality (IMTCR2002), Tokyo, Japan 2002;51–4.

Kurohashi S, Nagao M. A syntactic analysis method of long Japanese sentences based on the detection of conjunctive structures. Comput Linguist 1994;20(4):507–34.

Lawrence D, Thomas JB. Social dynamics of storytelling: implications for story-base design. In: Proceedings of narrative intelligence, AAAI Fall symposium 1999. AAAI Press; 1999.

Li Q, Nakano Y, Okamoto M, Nishida T. Highlighting multimodal synchronization for embodied conversational agent. Proceedings of the 2nd international conference on information technology for application (ICITA 2004) 2004.

Mateas M, Stem A. Towards integrating plot and character for interactive drama. In: Proceedings of social intelligent agents: the human in the loop symposium, AAAI Fall symposium series, AAAI Press; 2000.

McNeill D. Hand and mind: what gestures reveal about thought. Chicago, IL: The University of Chicago Press; 1992.

Microsoft Agent. http://www.microsoft.com/msagent/default.asp.

Mott BW, Challaway CB, Zettlemoyer LS. Towards narrative-centered learning environments. In: Proceedings of narrative intelligence, AAAI Fall symposium 1999, AAAI Press; 1999.

Nakanishi H, Yoshida C, Nishimura T, Ishida T. Free walk: supporting casual meetings in a network. Proceedings of ACM CSCW'96 1996;308–14.

Nakano YI, Okamoto M, Kawahara D, Li Q, Nishida T. Converting text into agent animations: assigning gestures to text. In: Proceedings of human language technology conference of the North American chapter of the Association for Computational Linguistics (HLT-NAACL 2004), Companion Volume, Boston; 2004. p. 153–6.

Nishida T. Social intelligence design for web intelligence. IEEE Comput Spec Issue Web Intell 2002;35(11): 37–41.

Rickel J, Johnson WL. Animated agents for procedural training in virtual reality: perception, cognition and motor control. Appl Artif Intell 1999;13(4–5):343–82.

Schank R, Abelson R. Knowledge and memory: the real story. In: Wyer W, editor. Advances in social cognition, vol. VII. Hillsdale, NJ: Lawrence Elbaum; 1995. p. 1–86.

Shaw E, Johnson WL, Ganeshan R. Pedagogical agents on the web. In: Proceedings of the third international conference on autonomous agents; 1999. p. 283–90.

Takahashi T, Takeda H. TelMeA: an asynchronous community system with avatar-like agents. In: Proceedings of the eighth IFIP TC.13 conference on human–computer interaction (INTERACT 2001); 2001. p. 190–7.

The Synchronized Multimedia Integration Language (SMIL). http://www.w3.org/AudioVideo/.

van Mulken S, Andre E, Mfiller J. The persona effect: how substantial is it? In: Proceedings of people and computers XIII: Proceedings of HCI'98; 1998. p. 53–66.