Proposal of MMI-API and library for JavaScript

Kouichi Katsurada, Taiki Kikuchi, Yurie Iribe, and Tsuneo Nitta

Toyohashi University of Technology,
Toyohashi, Aichi 441-8580, Japan
{katsurada, nitta}@cs.tut.ac.jp,
kikuchi@vox.cs.tut.ac.jp
iribe@imc.tut.ac.jp

Abstract. This paper proposes a multimodal interaction API (MMI-API) and a library for the development of web-based multimodal applications. The API and library enable us to embed synchronized multiple inputs/outputs into an application, as well as to specify concrete speech inputs/outputs and actions of dialogue agents. Because the API and the library are provided for JavaScript, which is a commonly used web-development language, they can be executed on general web browsers without having to install special add-ons. The users can therefore experience multimodal interaction simply by accessing a web site from their web browsers. In addition to presenting an outline of the API and the library, we offer a practical example of the use of the multimodal interaction system, as applied to an English pronunciation training application for Japanese students.

Keywords: MMI-API, JavaScript, web-based multimodal interaction

1 Introduction

The use of web-based multimodal interaction is one of the most absorbing research topics in the area of multimodal interaction. Some multimodal description languages such as X+V [1] and XISL [2] have been proposed to describe speech interaction scenarios and are used together with HTML pages, while SALT [3] provides a set of tags that are used to embed a speech interface into HTML documents. Other languages such as SMIL [4], MPML [5], and TVML [6] define the synchronization of output media or the gestures of animated characters for the rich presentation of output media/agents. Although these languages have resulted in significant advances in web-based multimodal interaction, not many of them are widely used as practical systems.

One reason for this is the difficulty that application developers face in mastering a new description language. Although the above languages provide a strong foundation for developing multimodal applications, application developers must make considerable effort to learn them. If developers could instead use a well-known language, they could create many applications without such an investment of time and effort. The other difficulty presented by the currently available language is the complexity faced by users in the installation and compilation of an interpreter. For general web users, it is time-consuming to install new software when they use a new application. To avoid

this labor, it is desirable for multimodal applications to be executed without the use of any additional software.

Against this background, we propose an MMI-API for JavaScript and provide a library that can be executed on general web browsers. Because JavaScript is the most commonly used language for web development and provides rich descriptive power for controlling multimodal interaction, developers can create rich multimodal applications without needing to learn a new language. This enables users to experience multimodal interaction on the web through their browsers. The ease of use of these features will accelerate the use of multimodal interaction on the web.

A number of APIs and libraries have been proposed for the handling of speech and dialogue agents in JavaScript. Microsoft agent [7] is one approach to handling animated characters on the web. It provides an API to control the gestures of animated characters in JavaScript. It differs from our proposal, however, in that the characters need to be installed on the user's PC, whereas our API and library require no installation in order for the user to experience multimodal interaction. Other approaches to providing rich web contents using Flash and JavaScript utilize synchronized outputs of speech, music, and movies. Most of them, however, just provide output functionality. The purpose of our approach is to provide an API that can handle both output and input modalities, including speech interaction with dialogue agents. This distinguishes our approach from other approaches for delivering rich content through the web.

In the rest of this paper, we first discuss the requirements for the API in Section 2, and present an outline of the API specifications in Section 3. In Section 4, we show the system structure of the library. Next, in Section 5, we introduce a sample application, and then conclude our proposal in Section 6.

2 Requirements for MMI-API

A number of languages have been proposed for the development of multimodal interaction systems. SALT is a tag set that is used to add a voice interface to its mother language (which is usually presumed to be HTML). It provides tags for speech input/output, binding them to HTML forms, along with some other functions that are used in speech interaction. SALT tags are activated by the JavaScript described in its mother HTML document. X+V achieves multimodal interaction by using VoiceXML, which defines speech interaction, together with XHTML. VoiceXML and XHTML tags are bound by the original tags. The VoiceXML document is invoked by an XML event that occurs in the XHTML document. XISL is a language that is used to control a multimodal interaction scenario, which includes speech interface, HTML pages, and so on. In an XISL document, HTML is handled as one of the output/input modalities. SMIL, MPML, and TVML are languages that are used to control multimedia presentations or the gestures of dialogue agents. SMIL provides detailed synchronization of multimedia outputs such as audio and movies, whereas MPML offers a number of gestures that are performed by animated characters. TVML also controls the performance of animated characters for TV programs. Table 1 lists the features of these languages.

Table 1. Features of existing languages

	SALT	X+V	XISL	SMIL	MPML	TVML
multimodal inputs	Yes	Yes	Yes	No	Yes*	No
multimodal outputs	Yes	Yes	Yes	Yes	Yes	Yes
media synchronization	No	No	Yes**	Yes	Yes	Yes
dialogue agent	No	No	Yes	No	Yes	Yes
compatibility with HTML	Yes	Yes	Yes	No	No	No

* Some versions of MPML support speech interface [8].
** XISL provides limited input/output synchronization.

For an MMI-API, the handling of multimodal inputs/outputs is an essential function. At the same time, the synchronization of inputs/outputs and the gestures of dialogue agents play important roles in the development of rich applications with various input/output modalities. Another important feature of an MMI-API is compatibility with HTML. To create useful applications, data binding with HTML is a fundamental requirement. To summarize the above discussion, we defined the following requirements that an MMI-API should ideally satisfy.

1. It can handle multimodal inputs.
2. It can handle multimodal outputs.
3. It can control the synchronization of multimodal inputs/outputs.
4. It can control the gestures of dialogue agents.
5. It has a strong connection with HTML.

Because requirement 5 is a preexisting JavaScript feature, we sought to create an API that satisfies requirements 1-4.

3 Outline of MMI-API

Based upon the discussion in Section 2, we created the following specifications for an API in order for it to handle multimodal inputs/outputs. Table 2 shows the multimodal input/output functions provided by the API. It provides four types of input functions: `Input`, `seqInput`, `altInput`, and `parInput`; these handle unimodal input, multimodal sequential inputs, multimodal alternative inputs, and multimodal parallel inputs, respectively. As for outputs, the API provides three types of functions: `Output`, `seqOutput`, and `parOutput`; these handle unimodal output, multimodal sequential outputs, and multimodal parallel outputs, respectively.

These functions only partially satisfy requirements 1-3 because they do not define any details with regard to concrete inputs, outputs, and synchronization. These details are provided by the arguments given to the functions. These arguments are described in the JSON format [9], as listed in Figure 1. The JSON format is used to describe multiple pairs of properties and their values. Each property represents the type of modality, the conditions for accepting input, or some other options. Tables 3-5 show

the available properties of inputs, outputs, and gestures performed by the dialogue agents, respectively.

Table 2. Multimodal input/output functions

Type	Function	Outline
Input	`Input`	Accepts a unimodal (single) input
	`seqInput`	Accepts sequential inputs
	`altInput`	Accepts alternative (exclusive) input
	`parInput`	Accepts simultaneous inputs
Output	`Output`	Outputs a unimodal (single) content
	`seqOutput`	Outputs contents sequentially
	`parOutput`	Outputs contents simultaneously

Fig. 1. Descriptive example of multimodal input/output functions

```
// Input example
mmi.altInput({
     "type"   :"click",
     "match"  :"agent"
},{
     "type"   :"speech",
     "match"  :"./grammar/start.txt"
});    //start interaction with the agent
       //click the agent or talk to the agent


// Output example
mmi.parOutput({
     "type"   :"audio",
     "event"  :"play",
     "match"  :"./sound/isshoni.ogg",
     "options":{
           "begin":500
     }
},{
     "type"   :"agent",
     "event"  :"gesture",
     "gesture":"speak",
     "options":{
           "begin":500,
           "dur"  :5500
     }
});    //output the agent's speech
       //This example does not use TTS.
```

Table 3. Example of input properties and available values

Property	Value	Outline
"type"	"click"	Mouse click
	"mouseover"	Mouseover event
	"speech"	Speech from the user
	"keydown"	Key event
	:	:
"match"	HTML ID	Specify the ID in the HTML document ("type" is mouse event)
	"agent"	Operation to the agent ("type" is mouse event)
	Grammar file	Specify the grammar file ("type" is "speech")
	Character	Output contents simultaneously ("type" is key event)
	:	:
"options"		
"begin"	Time or event	Start time or event of input acceptance
"dur"	Time	Duration of input acceptance
"repeatCount"	Number	Number of repetitions
:	:	:

Table 4. Example of output properties and available values

Property	Value	Outline
"type"	"agent"	Agent's gesture
	"audio"	Output an audio file
	"browser"	Output a HTML page
:	:	:
"event"	"speech","gesture"	Agent's gesture ("type" is "agent")
	"play", "stop",…	Operation to audio ("type" is "audio")
	:	
"match"	HTML ID or path to media	Specify an output content ("type" is "browser", "audio", …)
"text"	Output text	Specify the output text ("type" is "agent", "browser")
"gesture"	Define a gesture	The gesture is performed by the agent
"options"		
"begin"	Time or event	Start time or event of presentation
"dur"	Time	Duration of presentation
"fadein"		Fade-in effect
"volume"	0.1 to 1.0	Volume control
:	:	:

Table 5. Gestures performed by dialogue agent

Type	Gesture	Outline
Facial expressions and gestures (24 gestures)	"happy"	Happy expression
	"sad"	Sad expression
	"blink"	Blink
	"speak"	Lip sync
	:	:
Hand gestures (15 gestures)	"pointup"	Point upward
	"grab"	Grab an object
	:	:
Body gestures (14 gestures)	"walk"	Walk
	"explain"	An explanatory posture
	:	:
User interaction (4 gestures)	"appear"	An agent appears
	"disappear"	The agent disappears
	:	:
Others (5 gestures)	"search"	Search for something
	"wait"	Wait
	:	:

4 Multimodal Library and System Structure

We provide a multimodal library together with the API. The library includes a JavaScript program for handling multimodal inputs/outputs, a Java applet program for recording speech input, and a Java servlet program for managing server-side applications such as a speech recognition engine, speech synthesis engine, and agent manager. Figure 2 shows the system structure of the multimodal interaction system.

In this system, a speech input from a user is recorded in the sound recorder module, which is executed on the web browser. It is then sent to the session manager on the web server through the JavaScript program that is executed on the web browser. The session manager sends it to the input manager, and the speech is then recognized by the speech recognition engine Julius [10]. The recognition result is sent to the JavaScript program through the input manager and the session manager.

The production of speech output and dialogue agent animation is ordered from the JavaScript program that is executed on the web browser. An output text and a command specified by the JavaScript program (using the API) are sent to the agent manager through the session manager, the output control manager, and the output manager. The agent manager produces a synthesized voice using the speech synthesis engine Aques Talk [11], and sends it to JavaScript, together with the images of the dialogue agent through the output manager and the session manager. The images of the dialogue agent are given in a gif format, and are converted into animation on the web browser.

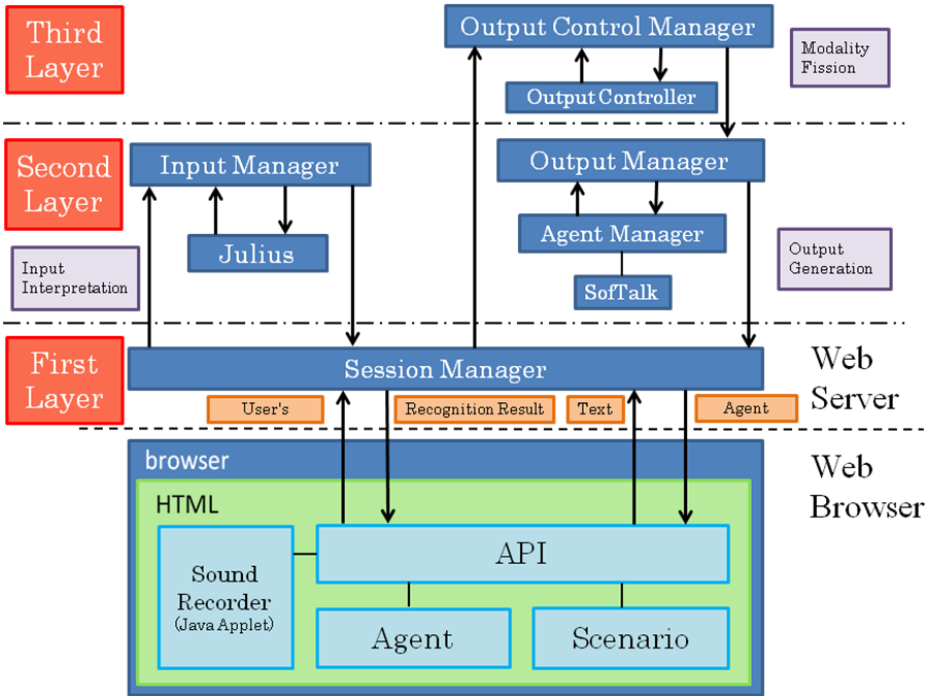

Fig. 2. Structure of multimodal interaction system

5 Sample Application Using developed API and Library

To confirm the usability of the developed API and library, we embedded multimodal interaction into a web-based English pronunciation training application for Japanese students [12] that was scripted using Action Script (A scripting language used for developing Flash software). The application recognizes the user's pronunciation of a phoneme and shows its manner and place of articulation (such as the shape of the mouth and position of the tongue) on the International Phonetic Alphabet (IPA) vowel-chart. The user can correct his/her pronunciation by modifying his/her mouth shape and tongue position according to the chart.

Figure 3 shows a screenshot of the application, and Figure 4 outlines its system structure. The character shown at the center of the figure is a dialogue agent to which the user speaks. The user can input commands to the system either by operating the mouse or by speaking to the agent. These commands are sent to the Flash program that is executed on the browser, after which they are sent to the server. The content is delivered to the user after synchronization of the dialogue agent's gestures, speech, and background music.

Fig. 3. Screenshot of English pronunciation training application

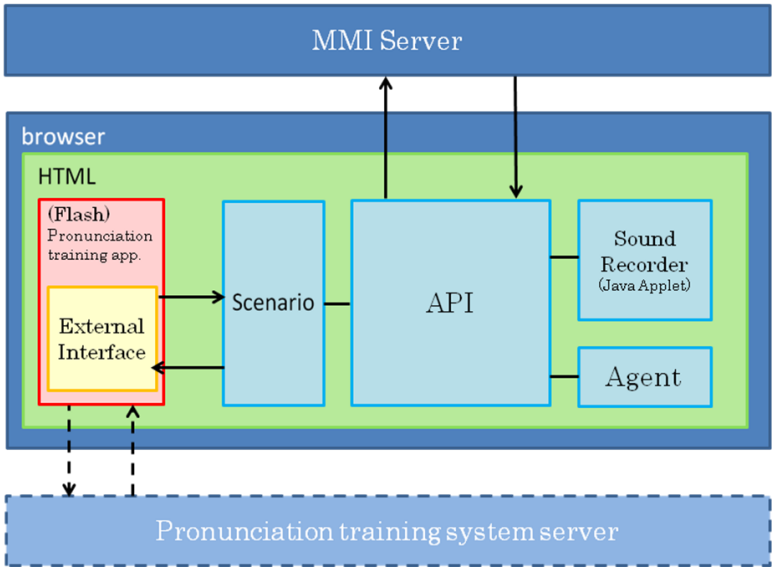

Fig. 4. Browser-side system structure of pronunciation training application

Using the proposed API and library, through the development process of this application, we confirmed a number of findings. Compared to the development of the application using the other languages, our API and library realize more detailed and complex control of interaction, including control of the timing of inputs/outputs and coordination with Flash or other APIs (such as Google map API). Because an application developer can now construct an interaction scenario only using JavaScript, he/she can regard the interaction scenario and the application as a single program unit. This feature enables developers to construct complicated interaction scenarios far more easily than they would use some other languages. However, an issue arises due to a characteristic of the language JavaScript, namely the difficulty of synchronization that is triggered by the end of an input acceptance or an output presentation. This is because JavaScript does not provide a "`wait`" function. The developer still has to write a somewhat complicated program to execute this type of synchronization. We would like to resolve this issue in the future.

6 Conclusions

In this paper, we have proposed a multimodal interaction API for JavaScript, and have also provided a library that can be executed on general web browsers. Although the API is very simple one which contains only four types of input functions and three types of output functions, it has a strong descriptive power in the arguments given to these functions. Through the development process of a web-based English pronunciation training application for Japanese students, we confirmed that the API and the library provide more detailed control of a complicated interaction, including control of the timing of inputs/outputs and coordination with Flash programs. This feature enables developers to construct complicated interaction scenarios more easily than they would use some other languages.

The remaining studies are to resolve the problem of output synchronization, which was mentioned in Section 5, and to implement various non-implemented functions of the library (such as the nesting of inputs/outputs, and some gestures of the dialogue agent, among others). In the near future, we intend to publish the API and the library on the web.

References

1. XHTML+Voice: http://www.w3.org/TR/xhtml+voice/
2. Katsurada, K., Nakamura, Y., Yamada, H. and Nitta, T.: XISL: A Language for Describing Multimodal Interaction Scenarios. Proc. of ICMI'03 (2003) 281-284
3. Wang, K.: SALT: A spoken language interface for web-based multimodal dialog systems. Proc. of InterSpeech2002 (2002) 2241-2244
4. SMIL: http://www.w3.org/AudioVideo/
5. Tsutsui, T., Saeyor, S. and Ishizuka, M.: MPML: A Multimodal Presentation Markup Language with Character Agent Control Functions. Proc. WebNet 2000 World Conf. on the WWW and Internet (2000)

6. Hayashi, Ueda, and Kurihara: TVML (TV program Making Language) - Automatic TV Program Generation from Text-based Script -. ACM Multimedia'97 State of the Art Demos (1997)
7. http://www.microsoft.com/products/msagent/main.aspx
8. Nishimura, Y., Minotsu, S., Dohi, H., Ishizuka, M., Nakano, M., Funakoshi, K., Takeuchi, J., Hasegawa, Y. and Tsujino, H.: A markup language for describing interactive humanoid robot presentations. Proc. of IUI07 (2007) 333-336
9. JSON: http://www.json.org/index.html
10. Kawahara, T., Kobayashi, T., Takeda, K., Minematsu, N., Itou, K., Yamamoto, M., Yamada, A., Utsuro, T., Shikano, K.: Sharable software repository for Japanese large vocabulary continuous speech recognition. Proc. ICSLP'98 (1998) 3257-3260
11. Aques Talk : http://www.a-quest.com/aquestalk/
12. Mori, T., Iribe, Y., Katsurada, K. and Nitta, T.: Real-time Visualization of English Pronunciation on an IPA Vowel-Chart Based on Articulatory Feature Extraction. IPSJ SIG Technical Report 89-15 (2011) (In Japanese)