World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:2, No:11, 2008

A Support System Applicable to Multiple APIs for Haptic VR Application Designers

Masaharu Isshiki, Kenji Murakami, and Shun Ido

***Abstract*—**This paper describes a proposed support system which enables applications designers to effectively create VR applications using multiple haptic APIs. When the VR designers create applications, it is often difficult to handle and understand many parameters and functions that have to be set in the application program using documentation manuals only. This complication may disrupt creative imagination and result in inefficient coding. So, we proposed the support application which improved the efficiency of VR applications development and provided the interactive components of confirmation of operations with haptic sense previously.

In this paper, we describe improvements of our former proposed support application, which was applicable to multiple APIs and haptic devices, and evaluate the new application by having participants complete VR program. Results from a preliminary experiment suggest that our application facilitates creation of VR applications.

***Keywords*—**VR application, Support system, Haptic devices, Haptic APIs.

I. Introduction

RECENTLY, development of virtual reality technology and improvements of a computer performance enable construction of VR space with an actual tactile feeling [1]. Furthermore, in order to construct VR space which has an actual feeling more, research for showing a haptic sense and the sense of smell, in addition to vision and hearing, is proceeding rapidly. Since the user can interact with objects in VR space, the haptic sense is very important. Devices that can present a haptic sense have been developed such as PHANTOM, SPIDAR [2]– [5]. In order to develop applications that use these haptic devices, APIs (Application Programming Interface), such as OpenHaptics Toolkit [6], Spatial GUI System [7], and e-Touch [8], are indispensable. Various APIs, including those used to build VR space, to perform a physical simulation [9], those that control two or more pieces of apparatus [10], have been developed. However, when the VR application designers create applications, it is often difficult to handle and understand many parameters and functions that have to be set in the application program, with reference only to document manuals. In the conventional programming, the designers consult document manuals in order to program the users' application and edit the text file of program. They cannot by sure that the program correctly reflects the selected parameters and functions until they compile and execute the program.

Authors are with Graduate School of Science and Engineering, Ehime University, Japan (e-mail: masaharu.isshiki@ic.cs.ehime-u.ac.jp).

Consequently, we proposed a support application based on the Spatial GUI System that permit user application development to be performed efficiently. We showed that understanding the parameters and functions of haptic APIs can improve working efficiency in application development. However, although our support application has been revealed properties of the parameters and functions of the API, it was difficult to learn the flow of the whole application [11]. Moreover, since our application supported only the Spatial GUI System, it was difficult to expand for wider use.

In this paper, we present a support system for VR application designers of haptic devices. With our method, designers can use the interactive manual of different software, namely the support application, and haptic APIs. Our method makes it possible for designers to learn the meaning of parameters and functions by their own senses of sight and force using the support application.

In the proposal system, we include a parameter study component common to all the APIs, and the check component uniquely tailored to each particular API. We incorporate function definition used by XML in the parameter study component. We construct the check component in the plug-in model. Our system is extensible to two or more APIs. Moreover, our method can output source codes. This component enables study of the flow of a program. The source code can be used as a model of the application which the designer creates. This application enables for the designers to learn the roles of parameters by perceiving the consequence of actual operation with their own sense of sight and force. They can produce VR applications efficiently using this support system.

II. Development of Support System

The important parameters for VR application with haptics are the followings:

- Attributes of VR space (viewpoint, light source)
- Attributes of virtual objects (position, shape, material, physical attributes)
- States of haptic device (position, haptic properties)

Thus, there are many important parameters needed for constructing VR space. Moreover, the setting methods of these parameters differ for each API. It is inefficient to develop applications specialized in each API. For instance, it is possible to change the viewpoint parameters both in the Spatial GUI System and OpenHaptics Toolkit. These parameters can be set either through the API in the Spatial GUI System or by editing OpenGL functions directly in OpenHaptics Toolkit.

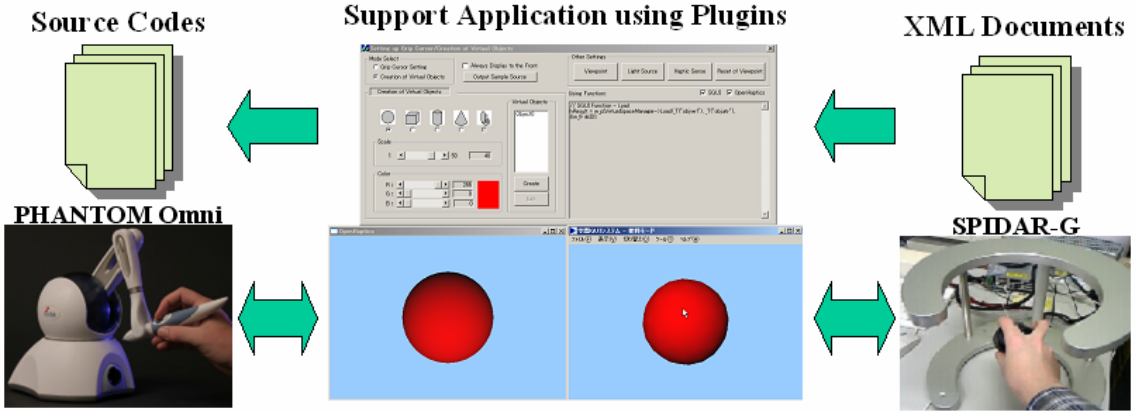

Fig. 1 The configuration of a proposal system

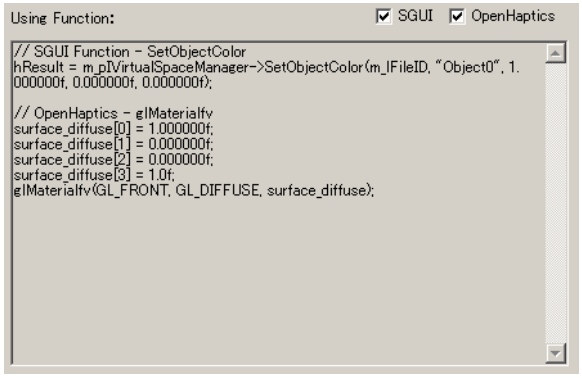

Fig. 2 Display using functions

When VR application designers create applications, it is often difficult to handle and understand many parameters and functions that have to be set in the application program using only the document manuals. Therefore, the system for supporting designers is required.

In order to develop of the VR applications using the APIs efficiently, the following two points are very important.

- Learn about API's parameters and functions.
- Learn the flow of the whole program using the API.

We propose a support system which has the following three components for supporting application development.

- Display API's function using XML.
- Confirm of system operation using a plug-in.
- Output source codes.

Using the component which displays functions, and the component which confirms details of operation using the plug-in, designers can learn the parameters and functions of the APIs. Using the source code output component, the designers can learn about the flow of application development. The following paragraph explains details of each our system components.

The configuration of a proposed system is shown in Fig. 1. The proposed system is composed of a support application, plug-in for each API, and haptic devices. The support application can change parameters important for VR space construction easily. Moreover, the function then used can be displayed from XML files. Changes resulting from re-setting parameters can be confirmed by operators with their own sense of sight and force. The changes of VR space can be outputted as source codes. The designers can use the source code as a model of application. The proposal system corresponds to the Spatial GUI System and the OpenHaptics Toolkit currently. Plug-ins which are used each API perform control of SPIDAR and PHANTOM, and VR space construction.

The SPIDAR is a haptic device which is invented By Prof. Makoto Sato at Tokyo Institute of Technology. It is controlled by Spatial GUI System consisting of a spatial GUI, device driver, and interface (firmware). The Spatial GUI System makes it possible to move, rotate, and grasp of virtual objects and so on.

The PHANTOM haptic interface, originally developed by Thomas Massie at MIT, is a convenient, active device that senses the position of a user's fingertip, and can exert a precisely controlled force back upon the fingertip. The PHANTOM is controlled by OpenHaptics Toolkit.

A. Display API's Function using XML

When parameters are changed or the functions of API are used, our proposal system displays the used function in order to understand the APIs and support creating an application. We call this component "Display using functions." In order to display the functions, the proposal system reads the XML files. API's functions, which are used in changing parameters, are correlated with character strings of the used functions. Moreover, we defined common function name for each API's functions.

Fig. 2 shows an example of "Display using functions" which is related to Spatial GUI system and OpenHaptics Toolkit for a certain common function. It is possible to select the display/nondisplay for each API by means of check box at the upper right. The API's functions for each API are displayed for certain common functions.

The character string to display is read from XML files. XML (eXtensible Markup Language) is a markup language for describing data which are embedded as tags. It is possible to make unique tags under XML. So, we adopt unique tags such as <SGUI></SGUI> for Spatial GUI System. Table I and Table II

show examples of XML documents.

TABLE I
THE EXAMPLE OF A XML DOCUMENT OF SPATIAL GUI SYSTEM

```
.<?XML VERSION="1.0" ENCODING="SHIFT_JIS" ?>
<SGUI>
  <SetMaterialDynamicFriction>
    <comment>
      // SGUI Function – SetDynamicFriction ConstantProperty
    </comment>
    <function>
      SetDynamicFrictionConstantProperty(%d, %s, %ff);
    </function>
  </SetMaterialDynamicFriction>
</SGUI>
```

TABLE II
THE EXAMPLE OF A XML DOCUMENT OF OPENHAPTICS TOOLKIT

```
<?xml version="1.0" encoding="Shift_JIS" ?>
<OpenHaptics>
  <SetMaterialDynamicFriction>
    <comment>
      // OpenHaptics - Dynamic Friction
    </comment>
    <function>
      hlMaterialf(HL_FRONT_AND_BACK,
      HL_DYNAMIC_FRICTION, %ff);
    </function>
  </SetMaterialDynamicFriction>
</OpenHaptics>
```

XML language defines a concise and strict grammar, it is simple to check whether the grammar is correct or not, and it is manageable to analyze the syntax because of libraries preparation for analyzing XML. We adopted XML as a data file standard as described above.

XML documents are composed of common function names, API's functions name and their explanations. Both of Table I and Table II are examples of dynamic friction of virtual objects. The function "SetDynamicFrictionConstantProperty" is executed in Spatial GUI System, The function "hlMaterialf" received the argument "HL_DYNAMIC_FRICTION" in OpenHaptics Toolkit. Even if the features of functions are same, the names of functions and the range of parameters are different.

So, we had to define the tags of common function names for each API's function such as "SetMaterialDynamicFriction" in Table I and Table II. And we adopted the description specification of XML documents using the tag <function> for each function's name in each APIs. This led to ease of applying for any other haptic device and its API in our system. To adopt new device to our system, only creation of an XML document is needed.

The UI dialogs box in this system and changes the color of a virtual object, the function "SetObjectColor" is displayed according to the XML file. When the designer develops an application with haptic device API under a development environment such as VC++, this system can support copying and pasting the displayed function and this is effective to shorten the program development time.

TABLE III
EXAMPLES OF COMMON FUNCTION NAMES AND FUNCTION NAMES OF EACH API

Common function name	Spacial GUI System OpenHaptics Toolkit
Attributes of VR space	
SetLightPosition	SetLightPosition glLightfv
Attributes of virtual objects	
SetMaterial–DynamicFriction	SetDynamicFriction–ConstantProperty hlMaterialf
States of Haptic Device	
SetForce	SetForce hlTriggerEffect

B. The Feature of Confirmation of System Operation using Plug-In

In order to develop VR applications, it is important for designers to learn important parameters and setting ranges. However, since various haptic devices and its API are developed, it is inefficient to develop applications which are specialized for each API. Moreover, it is important for designers to confirm important parameters and functions by the common interface. Almost all APIs provide important parameters and functions needed to develop haptic VR applications. Therefore, it is important for designers to learn those parameters and functions by the same interface. Therefore, we developed a plug-in (DLL: Dynamic Link Library) for checking operation of each API.

In this system, if each plug-in has each function name and when users change parameters of the proposed system, the search system is needed to determine for which function should be called. This causes slowdowns of execution speed. So, we adopted the common function names to each API's function. Table III shows example of common function names and functions for each API.

Fig. 3 shows operation of proposed system in changing parameter.

1) The user changes the parameters in dialog.
2) The common functions name is called by API plug-in.
3) Each API plug-in renews display of confirmation of system operations.
4) Common function and each API's functions which are called from plug-in are written in XML files in advance. (Table I and Table II)
5) Each XML file is read and the information for "Display using functions" is acquired.
6) The system displays the using functions reflected on the changed parameters.

This process makes it possible for proposed system to control haptic device and create virtual environment through the plug-in. To adapt a new device to our system, only creation of a plug-in and a XML document is needed.

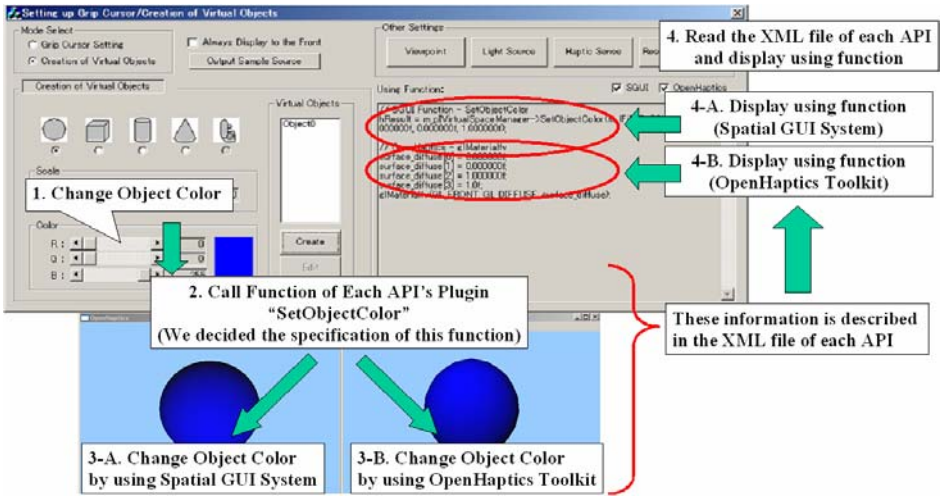

Fig. 3 Semantics of our proposed system when parameters are changed

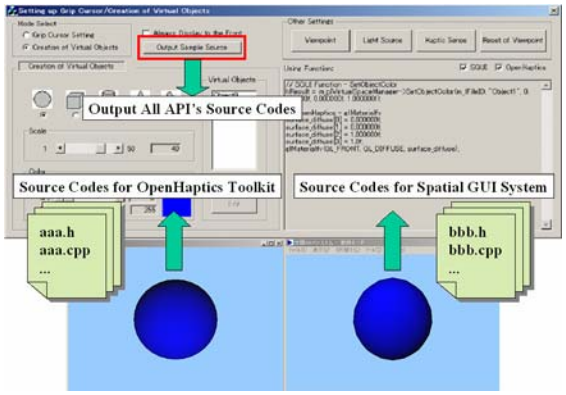

Fig. 4 The feature of output source codes

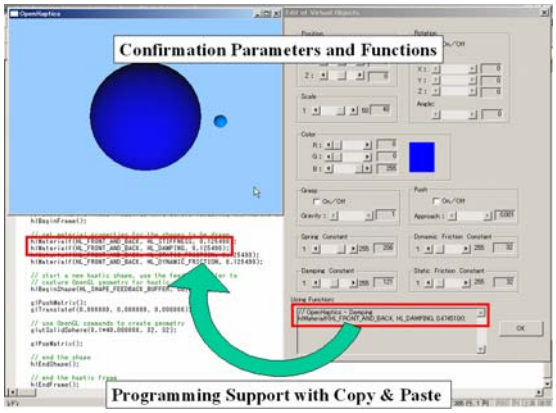

Fig. 5 The role of output source codes component

C. *The Feature of Output Source Codes*

Designers need to learn the flow of the whole program which used API. For example, which parts are initializations and end processing, and which parts are preformed in order to change the status of virtual environment. Designers usually learn these by referring document manuals and sample programs. It takes a lot of time to understand flow of the whole program.

So, this proposed system provides the feature of the output source codes as C++ which is confirmed by plug-in as described above. The parallel usage of display of the used functions and confirmation of the system operation makes it possible for designers to understand the program efficiently for virtual environment on the source code level (Fig. 4).

The designer opens source codes under the VC++ development environment together with proposed system. He or she can recognize which codes are corresponding to the API functions or parameters (Fig. 5).

The output source codes can be used as a whole program model of the applications for beginner designers.

III. The Features of the Proposed Support Application

The following important parameters for establishment of VR spatial can be changed and confirmed freely in this support application with haptic devices.

- Set the grip cursors
- Create and edit virtual objects
- Viewpoints and light sources
- Haptic senses

Our support application system deals with the parameters of the physical properties such as spring constant, dynamic friction, etc. as well as of the visual properties. Moreover, the system applies for multiple haptic devices. So, the designer can consider the suitable parameter values for each device and the adequate haptic device comparing with other devices.

A. *Studying of Haptic Information*

Fig. 6 shows a dialog example for set the grip cursor and the object. The position, rotation, spring constant, dynamic friction, mass, grasp status and push status as well as shape, size and color are can be changed. The haptic sense can be controlled for each X, Y, and Z axis direction by a grip cursor. The grip cursor is a special virtual object which roles the hand in virtual space. When the grip cursor contacts another virtual object, the haptic sense is generated. Setting the strengths of haptic signal along each axis generates an arbitrary haptic signal (Fig. 7). The designer can feel the haptic signal that he or she wants in developing application by setting and studying parameters.

B. Confirmation of Operations with Senses of Sight and Force

The confirmation of the operation takes place under each parameter value with senses of sight and force. The color and shape of created virtual objects can be confirmed in executed dialogs. The gravity and friction of them can be confirmed by manipulating haptic devices such as SPIDAR-G and PHANTOM Omni for designer feeling the force sense with his or her hand (Fig. 8).

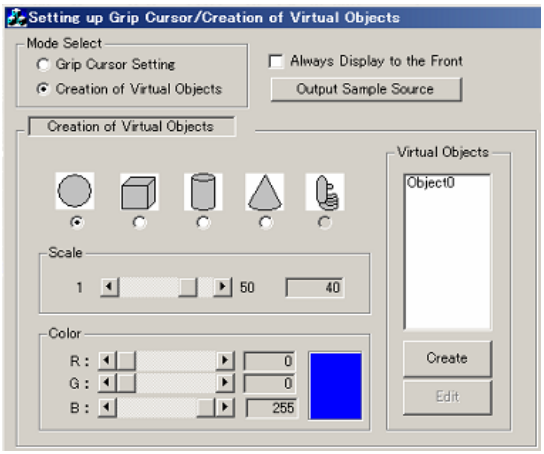

Fig. 6 The dialog for set the grip cursor and the object

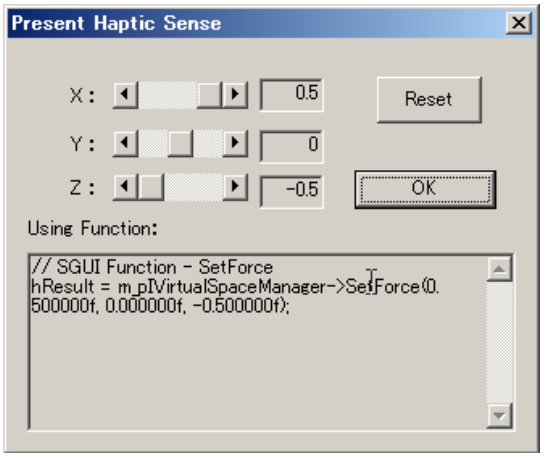

Fig. 7 The dialog for simulating haptic senses

Fig. 8 The confirmation of system operation with senses of sight and force

The multiple APIs can be operated at the same time and it is possible to compare the differences of parameter ranges or haptic sense. The present virtual space can be output as source codes such as VC++ and it is possible to compile and execute them promptly.

IV. Evaluation

We have evaluated the usefulness of proposal application. The purpose of proposal application is to support a haptic VR application development which used haptic APIs. Therefore, we compared the following two cases.

- The case which used proposed application
- The case which does not use proposal application

We provide 2 tasks for six participants unfamiliar with the Spatial GUI System and OpenHaptics Toolkit. We have evaluated proposal application by the following two methods.

- Completion time for a task (objective evaluation)
- Questionnaire survey (subjective evaluation)

A. Procedure for the Testing

The procedure is as follows:

1) Explain haptic devices and it's APIs,
2) Confirmation of the execution result from the sample program,
3) Consider the solutions of the task,
4) Work on the task (programming),
5) Confirmation of the program after compiling.

The sample program consisted of output source codes from this support system. It indicates existence of one red ball and one blue ball in virtual space. This experiment was performed under Windows 2000 OS and Visual C++ 6.0. To confirm the execution result from the sample program requires opening the sample program from Visual C++ 6.0, compiling and executing the sample program, and checking the execution screen and experiencing performance of VR with the haptic device.

Two tasks have almost same difficulty and should be accomplished Task 1 and Task 2 in that order. These are concerned with adding functions to a sample program. These tasks are,

Task1: Modify the color and the position of a virtual object,
Task2: Modify the spring constant and friction of a virtual object.

The items 4 and 5 are regarded as a completion time for the task. Evaluation based on the questionnaire after finishing the tasks is on a subjective scale of one to five.

B. Results

Fig. 9 shows the completion time of each task.

The completion time was shortened by using proposal application in both Task1 and Task2. These results indicate our application manual provided designers with the high efficiency programming and good comprehension of parameters and functions.

Fig. 10 shows the questionnaire survey. The performance using the proposed application was better than that using

document manual in all questions: (a) Intelligibility of parameters, (b) Intelligibility of functions, and (c) Search Facility of function. The question concerning the search facility of functions showed the greatest difference. This corroborated that the search facility of functions affected task completion time as shown in Fig. 9.

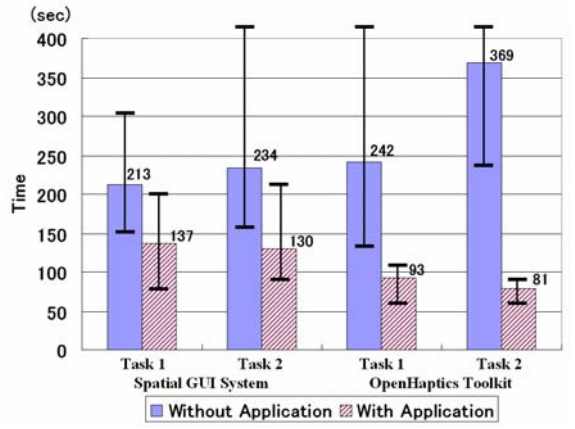

Fig. 9 The completion time of tasks

Participants remarked that it took much time to program with only document manual but it was efficiency for designer with proposed application to program with copying and pasting the displayed functions.

Thus, the evaluation of the new proposed application was the same as for the previous application with the Spatial GUI System. Moreover, it was the same evaluation also to the OpenHaptics Toolkit. Therefore, we conclude that proposal application was useful to two or more APIs.

V. Conclusion

We have developed a support system for VR application designers dealing with the haptic devices. They can more easily understand the parameters and functions of haptic APIs which are most important for constructing VR space using the VR application software. Proposed system enables designers to learn the roles of the parameters by perceiving the consequence of actual operation with their own sense of sight and force. It makes it possible to have them vary the system parameters without any difficulties and confirm the roles of parameters in order to construct a program. Moreover, it can output a source code and enables study of the flow of a program. An experiment confirmed the usefulness of this support system. Our application revealed the greater efficiency in programming and good comprehension of the properties of a certain VR system with the use of the proposed application compared to conventional methods. Moreover, proposed application was useful to two or more APIs.

In the future work, we will experiment our system with other haptic devices except SPIDAR-G and PHANTOM Omni and validate the generality its system. This system will be extended to any other APIs such as GHOST and Springhead. Further evaluation of understanding for APIs by participants will be made.

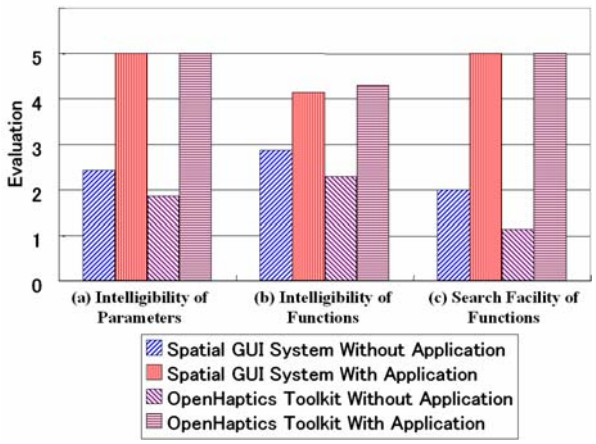

Fig. 10 Subjectivity evaluation by the questionnaire

References

[1] A. Kimura, F. Shibata, T. Tsuruta, T. Sakai, M. Oniyanagi, and H. Tamura, "Design and Implementation of Minority-Report-Style Gesture Interaction with Wide-view Electronic Working Space," Information Processing Society of Japan, vol. 47, no. 4, 2006.

[2] T. H. Massie and J. K. Salisbury, "The PHANTOM Haptic Interface: A Device for Probing Virtual Objects," in Proceedings of the ASME Winter Annual Meeting, Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems, Nov. 1994.

[3] S. Grange, F. Conti, P. Helmer, P. Rouiller, and C. Baur, "Overview of the Delta Haptic Device," in *Proc. Eurohaptics'01*, July 2001.

[4] M. Sato, Y. Hirata, and H. Kawarada, "Space Interface Device for Artificial Reality - SPIDAR -," *IEICE Trans.*, vol. J74-D-II, no. 7, 1991.

[5] J. Murayama, Y. Luo, K. Akahane, S. Hasegawa, and M. Sato, "A haptic interface for two-handed 6DOF manipulation -SPIDAR- G&G system," *IEICE Trans. on Infomation and Systems*, vol. E87-D, no. 6, June 2004.

[6] B. Itkowitz, J. Handley, and W. Zhu, "The OpenHaptics Toolkit: A Library for Adding 3D Touch Navigation and Haptics to Graphics Applications," in *Proc. WHC'05*, Mar. 2005.

[7] K. Sotome, N. Takahashi, S. Sasada, and M. Sato, "A primary study of novel digital archive with interactive art," *The Journal of The Society for Art and Science*, vol. 1, no. 3, 2000.

[8] A. Breckenridge, B. Hamlet, D. Mehlhorn, and K. Oishi, "A Dynamic Design Strategy for Visual and Haptic Development," in *Proceedings of the 6th PHANTOM Users Group Workshop*, Oct. 2001.

[9] S. Hasegawa, N. Fujii, and M. Sato, "Real-time Rigid Body Simulation for Haptic Interactions Based on Contact Volume of Polygonal Objects," *Computer Graphics Forum*, vol. 23, no. 3, Sept. 2004.

[10] M. Hirose, H. Iwata, Y. Ikei, T. O. adn Kouichi Hirota, H. Yano, and N. Kakehi, "Development of Haptic Interface Platform (HIP)," *Virtual Reality Society of Japan*, vol. 3, no. 3, 1998.

[11] M. Isshiki, S. Ido, and K. Murakami, "A Support Application for VR Design Process Based on Spatial GUI System," *The Journal of the Institute of Image Electronics Engineers of Japan*, vol. 34, no. 5, pp. 522–528, Sept. 2005.