Proceedings of DETC'02
ASME 2002 Design Engineering Technical Conferences
and Computer and Information in Engineering Conference
Montreal, Canada, September 29-October 2, 2002

DETC2002/CIE-34390

IMPLEMENTING SPEECH RECOGNITION IN VIRTUAL REALITY

Denis V. Dorozhkin
Judy M. Vance[1]
Department of Mechanical Engineering
Virtual Reality Applications Center
Iowa State University
Ames, Iowa 50011
dorodv@vrac.iastate.edu
jmvance@vrac.iastate.edu

ABSTRACT

Virtual Reality (VR) is becoming an important tool in the engineering product development process. The virtual environment provides the user with the ability to interact with three-dimensional digital representations of products using natural head and hand motions. While interacting with digital objects in VR seems natural, the use of traditional two-dimensional menu systems does not always provide a convenient interface to controlling task specifications in the three-dimensional space. New human-computer-interfaces are needed for this emerging VR design tool. This paper will present the details of implementing a speaker-independent, command and control, speech recognition menuing system for a virtual reality application. The menuing system will be described as it is incorporated into a virtual environment for the design of spatial mechanisms. Design and technical issues involved in the interface creation process are discussed and the resulting interaction system is described.

Keywords: speech interface, virtual reality, immersive environments, spatial mechanisms.

BACKGROUND

With advances in computer technology, Virtual Reality is becoming an important tool in the engineering product development process. It is used in a wide variety of engineering disciplines, encompassing the whole range of the product development cycle from modeling and evaluation of the first product prototypes, to providing training opportunities for end-product users [1-3]. VR technology provides a human-computer interface (HCI) that allows users to interact with digital objects as they appear in a simulated three-dimensional environment.

Spatial Mechanism Design

Application areas which are candidates for the use of VR technology are areas where understanding spatial relationships is key to completing a task. One such application area is motion synthesis of spatial mechanisms. The design objectives of spatial mechanism design are specified in three-dimensional space and the final mechanism is evaluated by viewing the mechanism motion in three-dimensional space. The designer's ability to correctly specify desired locations of an object and to effectively evaluate the motion of the resulting mechanism is essential for creation of a successful mechanism design. Currently, spatial mechanisms are designed using the traditional two-dimensional HCI of a computer monitor, keyboard and mouse. VR systems provide the ability to operate in third dimension, which allows users to manipulate objects in a more intuitive way with a variety of instrumented gloves and wands using natural hand motions. Also, the overall spatial layout of the problem as well as the correct dimensions of the objects are easily perceived in the VR environments, improving the efficiency of the design process.

Kihonge, et al. [4] developed VRSpatial as the first VR application for the design of 4C spatial mechanisms. Spatial 4C mechanisms are two degree-of-freedom closed chain linkages consisting of four rigid links connected by cylindrical, C, joints, which provide both translational and rotational movement. The mechanism synthesis algorithms that underpin the VRSpatial application are based on work detailed in Larochelle, et al. [5]. In VRSpatial, users define four design

[1] Corresponding Author

positions using geometrical models of objects by placing them in the VR environment. These objects are used by the application to generate a set of solutions for the spatial motion generation task. After one solution is selected from the solution set, the mechanism can be animated to verify the mechanism motion. During the entire process, the designer can move around in the VR environment in order to observe the mechanism from any arbitrary viewpoint. The primary operating environment for this application is the C6 virtual environment at Iowa State University (Fig. 1). The C6 environment consists of six rear-projection screens which form the sides and walls of a 10-foot by 10-foot room. The C6 is powered by a Silicon Graphics® Onyx2® computer with six InfiniteReality2™ graphics display generators. Stereo computer images are projected on each screen and magnetic position trackers are used to provide location information to the computer system. Multiple users can interact in the environment. Users wear active stereo glasses and one user has position trackers to track head and hand positions.

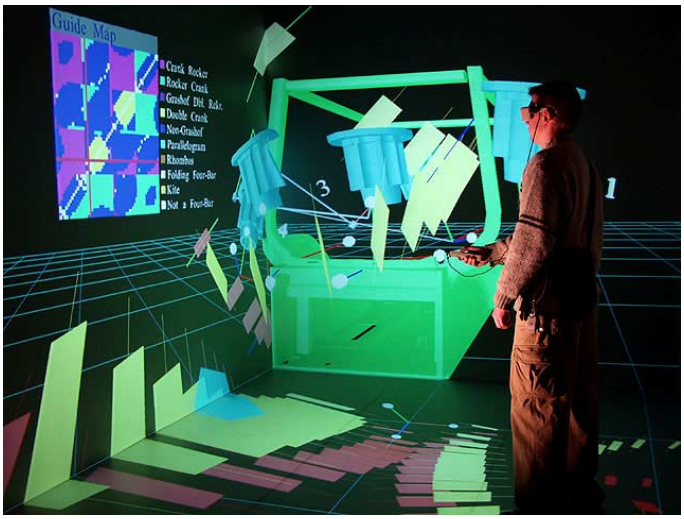

Figure 1. VRSpatial in C6 facility.

Users interact with VRSpatial using a wireless wand and a set of virtual menus that can be dynamically reconfigured during the program's execution. The menus are accessed using the wand buttons. While the menu system provides full control over all of the program's functions, the user has to make several selections from consecutive menus in order to access some options (i.e. "Animation"→"Adjust Speed"→"Go Faster"). Decreasing the number of consecutive selections by changing the overall structure of the menu system would lead to unnecessarily large and cluttered menus and ultimately decrease the effectiveness of the design process. As an alternative way to access some of the most frequently used features of the application, the menuing system functionality was extended to include speech recognition technology.

Speech Recognition

Weinschenk and Barker [6] define speech recognition as "the technologies that enable computers or other electronic systems to identify the sound of a human voice, separate that sound from noise in the environment, and accept the messages from the voice as input for controlling the system". The combination of a speech interface with the ability to interact directly with objects in the virtual environment results in a multimodal interface where users can interact with the VR environment by issuing either physical (i.e. wand motion) or speech commands [7].

There are two different approaches to implementing a speech interface to a virtual environment: fully interactive speech and 'command and control' speech. In addition, there are two methods of speech recognition: speaker-dependent and speaker-independent. Fully interactive speech provides the ability to recognize a wide variety of words and phrases, but results in the need to provide a speaker-dependent system. These systems require each user to provide samples of his/her voice that are used to train the system. This process is called enrollment [8]. Speaker-dependent software results in high recognition accuracy and extensive vocabulary, but it lacks flexibility, since it cannot be easily shared among users. Command and control systems require only a small set of predefined commands, which allows for the use of speaker-independent methods. Speaker-independent systems are intended to be used by multiple users and do not rely on the enrollment process to tailor the application to a specific individual. These systems are suitable for applications, which require only a relatively small vocabulary of approximately 40 words or less, plus the digits 0-9 [6].

The approach presented here requires only a small vocabulary so the command and control approach combined with speaker-independence has been chosen. This paper will present the details of implementing a speaker-independent command and control speech recognition menuing system for a virtual reality application.

VRSPATIAL SPEECH INTERFACE COMPONENTS

Silicon Graphics, Inc. (SGI®) computers running the IRIX® operating system are used to control the C6 virtual environment. Most of the commercially available speech-recognition software is to be used either on Microsoft® Windows® or Linux operating systems. Some versions of IRIX®-based speech recognition applications are available, but in general they represent experimental work and lack reliability. Due to these considerations, two main tasks were identified: creation of a 'command and control' speech-recognition application on a Windows® computer and development of a communication method between the Windows® computer and an SGI® system that executes the main VR application. The former was implemented with IBM ViaVoice™ for Windows® Release 8 Professional Edition and Speech for Java from IBM alphaWorks. The omniORB2 version 2.8 of the CORBA standard was used for communication purposes.

IBM Speech for Java

Speech for Java is a Java programming tool for incorporating IBM's ViaVoice™ speech technology into custom user interfaces. ViaVoice™ is a commercial speech recognition and synthesis program that can be used to control standard Windows® programs such as Microsoft® Word and Excel. In order to use ViaVoice™ as a speech engine for another program, an API such as Speech for Java™ is needed. Speech for Java is an implementation of version 1.0 of the Java™ Speech Application Programming Interface (JSAPI) developed by Sun Microsystems, Inc. in collaboration with leading speech technology companies. JSAPI specifies a cross-platform interface to support command and control recognizers, dictation systems and speech synthesizers [9]. Figure 2 shows the main components of a speech application developed with ViaVoice™ and Speech for Java.

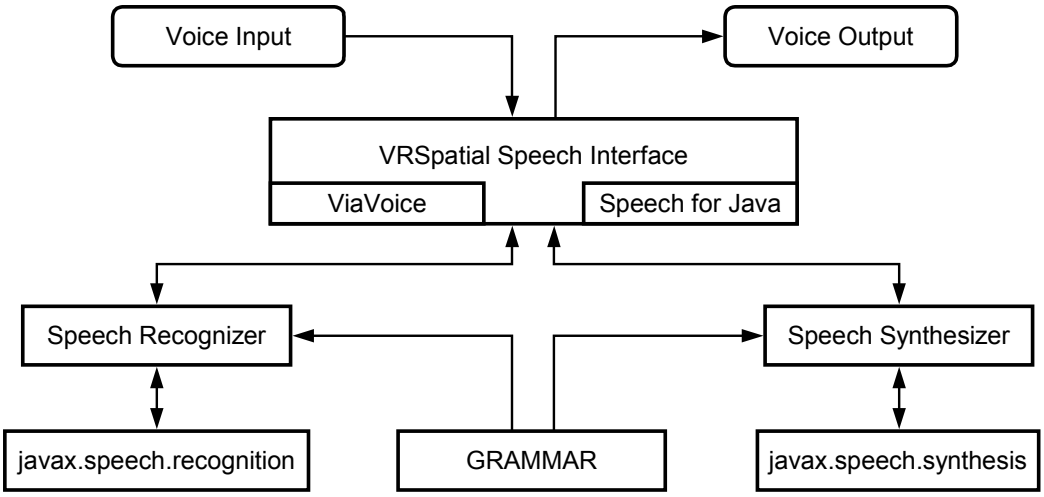

Figure 2. Speech application structure.

The speech synthesizer and the speech recognizer are instances of the `javax.speech.synthesis` and `javax.speech.recognition` packages. An essential part of a speech application is the grammar object. Within the grammar object are definitions of speech patterns and organization of speech that will be used in the application. Speech for Java relies on the Java™ Speech Grammar Format (JSGF) which is a platform-independent, vendor-independent textual representation of one type of grammar, a *rule grammar* (also known as a *command and control* grammar or *regular grammar*), for use in speech recognition. If speech synthesis is required in order to provide feedback to the user to confirm commands, the computer response is defined with the Java™ Speech API Markup Language (JSML). JSML is a text format used to annotate text input to speech synthesizers. It provides detailed information on how to speak text through definition of elements that control important speech parameters, such as pronunciations of words, emphasis and speaking rate [9]. Using the JSML, the quality, naturalness and understandability of synthesized speech output can be controlled.

OmniORB2

OmniORB2 is an Object Request Broker (ORB) that implements the 2.3 specifications of the Common Object Request Broker Architecture or CORBA [10]. It uses Remote Procedure Calls (RPC) technology that allows an application to make a remote procedure call with the same amount of effort as making a local function call. The calling application is designated as a client and the called application is designated as a server. The remote operations are grouped in interfaces, similar to C++ classes and are called CORBA objects, thus making it an object-oriented technology. One of the most important benefits of CORBA is the location transparency that it provides. It means that the operations on the CORBA objects are always invoked using the same syntax, no matter where the CORBA object is. CORBA also offers programming language neutrality – both the client and the server code can be written in any of the supported programming languages (C, C++, Java, etc.). The interface to a CORBA object is defined using the Object Management Group (OMG) interface definition language (IDL). IDL is a declarative language that is passed through an IDL compiler to map the IDL file to a specific language for the client and the server sides [11].

VRSPATIAL SPEECH INTERFACE IMPLEMENTATION

The main purpose of the speech interface implementation in the VRSpatial application was simplification of the interaction method. The ability to control the application using the existing set of menus was preserved, but each menu selection was evaluated for possible speech interface command.

Most of the menu system functionality could be effectively supplied with the speech interface. In fact, speech control gave the application users the option to access almost any point in the menu selection sequence with a single sentence. Figure 3 demonstrates two possible menu selection sequences along with the associated speech commands. For example, in order to reduce the animation speed the user has to access the "Animation" menu, select the "Adj. Speed" option and then select the "Slower" option. The same task can be accomplished by issuing the "Go slower" voice command. This eliminates the need to navigate the menu system and streamlines the interaction process.

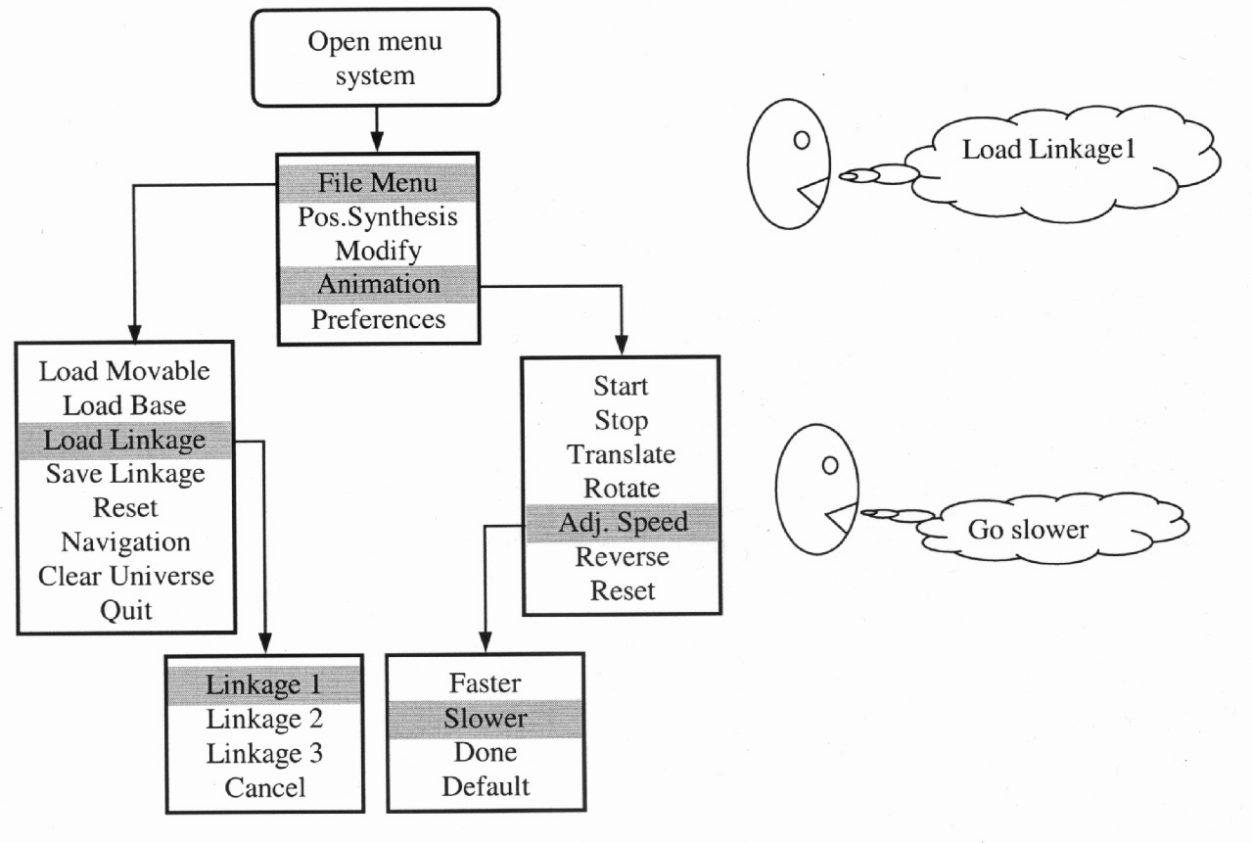

Figure 3. Menu commands and voice commands.

Hardware

A dedicated Windows® 2000 computer with access to the local network is used to handle the operation of the speech recognition interface. Network access is necessary in order to communicate with the SGI® computers that run the main VR application. Normally a wired headset or a stationary desktop microphone is used with ViaVoice™ software, but a wireless unidirectional microphone was selected for this application to allow users maximum mobility in the C6 virtual environment. Currently, only one-way communication is available, but eventually the computer audio output will be provided to the users through the sound system of the C6.

Software

The speech recognition interface is written entirely in Java. It uses Speech for Java routines in order to access ViaVoice's speech recognizer and speech synthesizer engines. The program continuously monitors the audio input by comparing it to the valid command patterns defined in the grammar file. The grammar file contains all of the valid action references ("Open", "Run", etc.) as well as many miscellaneous references that a user could possibly use while addressing the application ("Computer", "Please", etc.). As a way to provide maximum flexibility in the interaction process and to increase the range of supported user responses, several wording alternatives are also provided for a certain task (i.e. "Open Linkage", "Open Linkage File", "Load Linkage", etc.). A valid command must

contain at least one action reference and an arbitrary number of miscellaneous references.

If a valid command pattern is recognized, it is scanned for the presences of tags, which are associated with each action reference. Tags were used in order to simplify the processing of recognition results, since a single tag can be assigned to several incarnations of the same action reference. For example, "Load" and "Open" action references are assigned the {open} tag, and similarly, "Start" and "Begin" are assigned the {start} tag. The usage of tags is not required by JSGF specifications, but it was found to be convenient in this particular case. The combination of tags is analyzed and a decision is made regarding the particular command that the user has issued. That command is then relayed to the VR application via the network. If the interface determines that the user has made an effort to issue a command, but no matching command patterns were found then an appropriate message is generated by the speech synthesizer. The message, along with other possible verbal computer responses, is defined in a dedicated file that follows the JSML specifications.

To implement the networking capabilities, OMG IDL was used to define CORBA objects. An IDL compiler for Java was used to produce Java stub code. The stub code is used by the client (the speech interface in this case) to make invocations on the interface defined in the IDL file (VRSpatial in this case). Similarly, an IDL compiler for C++ was used to produce the C++ skeleton code, which is used by the server (VRSpatial) for definition of the CORBA object implementation.

When VRSpatial receives a command from the speech interface, the appropriate actions are undertaken in order to fulfill the user's request. In some cases these actions are the same that take place when a menu selection is made, while in others a different combination of actions has to be executed.

RESULTS AND CONCLUSIONS

The speech-enabled VRSpatial application was tested in the C6 environment. The speech control was found to be an extremely effective way of controlling the application. The performance accuracy of the speech interface was found to be quite satisfactory. No training of ViaVoice™ software for a specific individual was performed. Misunderstandings of valid command combinations were rare and can be contributed to the communication issues between the wireless microphone and the speech recognition computer, which is currently located at a significant distance from the C6 environment. This accuracy problem decreased drastically when the wired headset was used in a desktop environment during the programming and debugging stages, which also suggests the wireless communication as the source of the problem. The resistance of the speech interface to the ambient noise, such as sound effects generated by a VR application during its operation, was not thoroughly tested due to the absence of sound output in VRSpatial at this time. The normal conversation between the users was found to have little influence on the operation of the speech interface, as long as users directed their voices away from the microphone.

ACKNOWLEDGMENTS

The authors would like to acknowledge the support of the National Science Foundation through grants DMI-9872604. The collaboration of Dr. Pierre Larochelle and his students at the Florida State University greatly contributed to the development of the VRSpatial software.

REFERENCES

[1] Bullinger, H.G., Breining, R., Bauer, W., 1999, "Virtual Prototyping – State of the Art in Product Design," *Proc., 26th International Conference of Computers & Industrial Engineering,* Melbourn, December 15-17, 1999, pp. 103-107.

[2] Deisinger, J., Breining R., Rößler, A., Höfle, J., Rückert, D., 2000, "Immersive Ergonomic Analyses of Console Elements in a Tractor Cabin," *Proc., 4th Immersive Projection Technologies Workshop*, June 19-20, 2000, Ames, Iowa.

[3] Oliveira, J.C., Shirmohammadi, S., Hosseini, M., Cordea, M., Georganas, N.D., Petriu, E., Petriu, D.C., 2000, "VIRTUAL THEATER for Industrial Training: A Collaborative Virtual Environment," *Proc., 4th World Multiconference on Circuits, Systems, Communications & Computers*, (CSCC 2000), Greece, July 2000.

[4] Kihonge J.N., Vance J.M., Larochelle P.M., 2001, "Spatial Mechanism Design in Virtual Reality with Networking," *Proc., DETC'01: 2001 ASME Design Engineering Technical Conference*, DETC2001/21136, Pittsburgh, PA, September 9-12, 2001.

[5] Larochelle, P.M., "SPADES: Software for Synthesizing Spatial 4C Mechanisms," *Proc., DETC'98: 1998 ASME Design Engineering Technical Conference*, DETC98/MECH-5889, Atlanta, GA, September 1998.

[6] Weinschenk, S., Barker, D.T., 2000, *Designing Effective Speech Interfaces*, John Wiley and Sons, New York, NY, pp. 98-103.

[7] McGlashan, S., Axling, T., 1996, "A Speech Interface to Virtual Environments," *Proc., International Workshop on Speech and Computers*, St. Petersburg, Russia.

[8] Noyes J., 1993, "Speech Technology in the Future," In C. Baber & J. M. Noyes (eds) *Interactive Speech Technology: Human factors issues in the application of speech input/output to computers*, Taylor & Francis Ltd., London, pp. 189-208.

[9] http://java.sun.com/products/java-media/speech

[10] Bolton, F., 2002, *Pure CORBA*, Sams Publishing, Indianapolis, Indiana, pp. 6-13.

[11] Brose G., Vogel A., Duddy K., 2001, Java Programming with CORBA. Advanced Techniques for Building Distributed Applications, Wiley and Sons, New York, NY, pp. 17-43