

# An interrogative visualization environment for large-scale engineering simulations

Hatem M. Wasfy\*, Tamer M. Wasfy, Ahmed K. Noor

*Center for Advanced Engineering Environments, Old Dominion University, Hampton, VA USA*

Received 10 April 2004; revised 18 May 2004; accepted 24 June 2004

Available online 24 August 2004

---

## Abstract

An interrogative visualization environment is described for the interactive display and querying of large datasets. The environment combines a web-based intelligent agent facility with a visualization engine. The intelligent agent facility (IAF) incorporates a rule-based expert system for natural-language understanding, voice and text input facilities, a hierarchical clickable command list, an interface for multimodal devices such as menu-based wireless handheld devices and gesture recognition devices, and human-like avatars acting as virtual assistants. The IAF interacts with, and controls, the visualization engine through a TCP/IP network socket interface. The environment enables multiple users using a variety of interaction modes and devices to effectively browse through large amounts of data, focus on and query interesting features, and more easily comprehend and make use of the data. Application of the environment to the visualization of engineering simulations is described.

© 2004 Elsevier Ltd. All rights reserved.

*Keywords:* Intelligent software agent; Expert system; Multimodal; Interface; Natural language; Visualization

---

## 1. Introduction

Although the WIMP (Windows, Icons, Menus, and Pointing devices) paradigm has provided a stable global interface, it will not scale to match the myriad of form factors and uses of platforms in future collaborative distributed simulation environments. New technologies have been developed which create more natural and intuitive interfaces, and provide human-like interactions that enable broad uses of computers as virtual assistants or agents. These interfaces will allow the user to interact with the computer in the same way as one interacts with other humans using voice commands. The user will also be able to touch and manipulate objects in 3D immersive virtual environments using haptic technology. Multimodal interfaces enable the user to select the most appropriate modality for the task at hand, with the different modalities providing similar or complementary functionalities [1]. This can be particularly useful for multidisciplinary teams working on complex problems [2] (Fig. 1). Intelligent interfaces include

ones with software agents that act as virtual technical assistants or advisors, offer guidance to the user, explain the different aspects of the problem being investigated, or act as a session moderator for a collaborative session. Software agents including human-like avatars can also be embedded in intelligent virtual environments.

The present paper describes an ongoing research project on the application of intelligent interfaces to controlling visualization software. In a previous publication, a rule-based natural language interface was used to control a virtual simulation environment that was generated using the IVRESS toolkit<sup>1</sup> [3].

## 2. The basic components of the environment

Fig. 2 shows a diagram of how the various sub-components of the interrogative visualization environment fit

---

<sup>1</sup> IVRESS (Integrated Virtual Environment for Synthesis and Simulation) toolkit from Advanced Science and Automation Corp. is an Engineering visualization package that can display immersive virtual environments.

\* Corresponding author. Tel.: +1-757-766-5216.

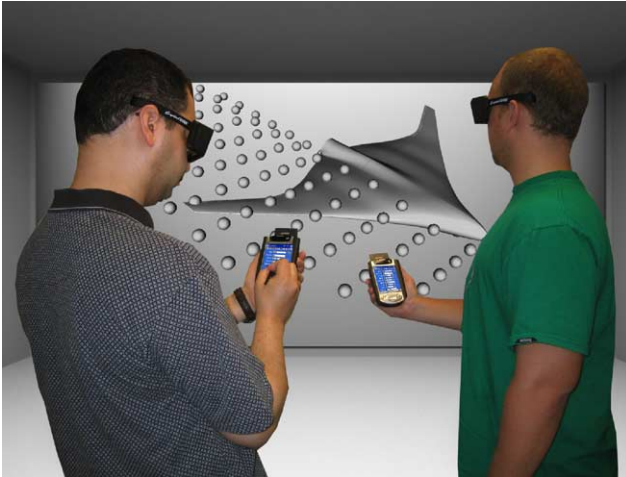


Fig. 1. Two users are interacting with the results of a CFD simulation in a power wall immersive virtual environment using wireless handheld devices.

together. The environment consists of two major components, a web-based intelligent agent facility (IAF) and a visualization engine. In the present study, the Enight visualization package developed by Computational Engineering International (CEI) [4] is the visualization engine used. The web-based IAF enables multiple users to interact, using different modalities, with the visualization engine. One of the web facilities acts as a server facility that connects directly to the visualization engine, while other client web facilities connect to the visualization engine through the server facility. The users send commands using natural sub-language<sup>2</sup> voice or text that gets translated to Enight script using a hierarchical rule-based expert system.<sup>3</sup> The expert system resides on the server web-based facility and hence a single logic interprets multiple modes of user input [7].

All commands and queries are routed through the server web-based facility. Another role of the server facility is to echo the audio response of the system to the users' commands to all the client facilities. Robustness of the system is enhanced by the fact that each of the client facilities can be turned into a server if the current server is no longer available. The web-based facilities take as input natural sub-language commands from a microphone, a text box, or a clickable hierarchical command list. Multiple interface multimodalities (such as handheld computers or gesture recognition systems) can also send commands as natural language text to the visualization package through the web-based facilities.

<sup>2</sup> A sub-language is a specialized subset of a language that is usually used by a group of specialists in a given field [5]. In the present study the sub-language in question is the one used by Computational Fluid Dynamics and Engineering Structures experts.

<sup>3</sup> An expert system is a knowledge-based system that has expertise in a specialized domain [6]. In the present study, the expert system's field of expertise is the inner workings of the Enight visualization package.

In a distributed collaborative session, the display output of the visualization package and the voice output of all intelligent agents connected to the package is routed to each computer connected to the session. This allows multiple geographically-dispersed users to interact with the visualization in a seamless manner. A supervisor agent running on the server web-based facility alerts the users if they issue commands too quickly or too slowly.

### 3. The web-based intelligent agent facility

The web-based IAF encompasses a hierarchical rule-based expert system, a supervisor agent, human-like avatars representing the user agent and supervisor agent, and multimodal interfaces. The IAF is based on the LEA<sup>4</sup> intelligent agent engine. Each of these components is described subsequently. A snapshot of the IAF web-interface is shown in Fig. 3.

#### 3.1. Hierarchical rule-based expert system

A hierarchical rule-based expert system is used to interpret and execute the natural sub-language commands issued by the users. A list of the commands in the rule-based expert system along with a brief description of each command is given in Appendix A. Rules are arranged into rule groups that can connect to each other. Within each rule group, the system searches for the rule that has the highest total score and executes it. All rules start with a default score of zero. The words in each user command are divided into three categories: keywords, ignored words and other words. Keywords are further divided into two subcategories: words that are required for the command's execution and words that render the command ambiguous or incomprehensible in relation to the current rule. The score of any given rule is the sum of the scores of all the words recognized by that rule. Keywords that are required for the rule's execution are assigned a positive score, while keywords that would render the command ambiguous or incomprehensible in relation to the current rule, or that are associated with other rules within the same rule group, are assigned a negative score. Ignored words are assigned a score of zero. Words that are neither keywords nor ignored words are assigned a small negative score to ensure that the system understands most of the words spoken by the user within the context of the current rule. If no rule with a score higher than a given preset value is found, the system returns the output "I did not recognize your command". While a command is being interpreted by the hierarchical expert

<sup>4</sup> LEA (Learning Environments Agent) is a web-based intelligent agent engine from Advanced Science and Automation Corp. It encompasses a hierarchical rule-based expert system engine, structured and unstructured knowledge engines, voice synthesis/recognition interfaces, and a client/server network interface.

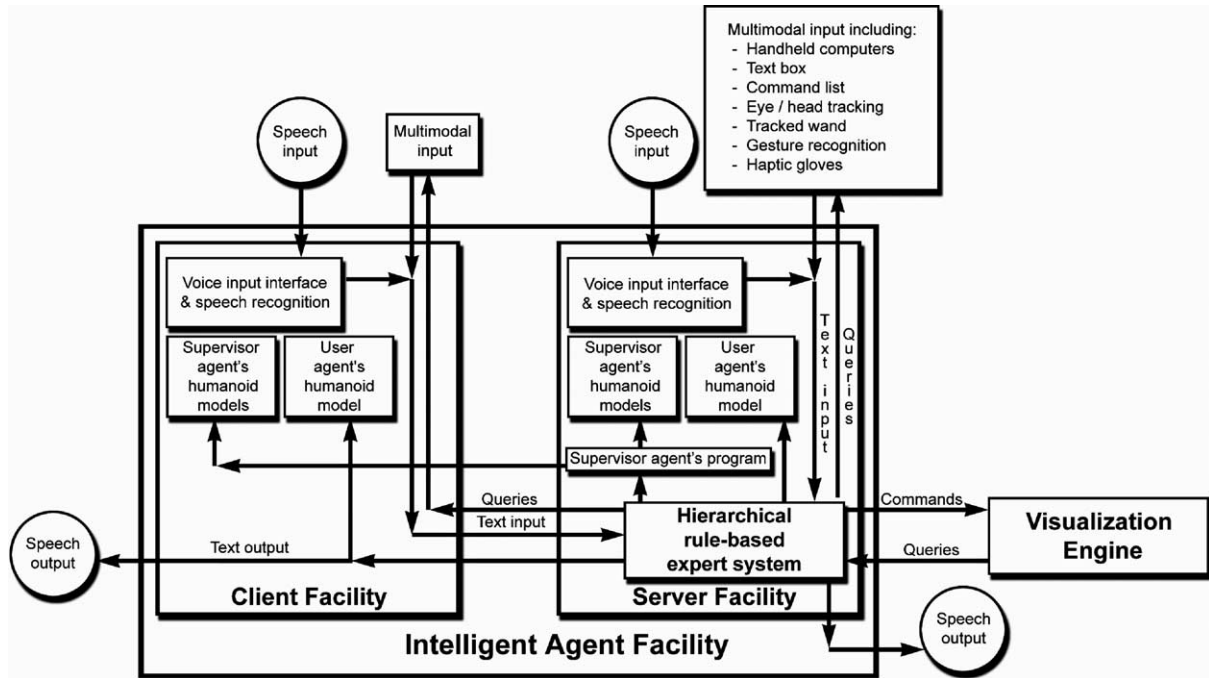


Fig. 2. Components of the web-based intelligent agent facility.

system, the Keywords and ignored words that a rule finds are removed from the input phase.

State variables are set by the expert system's rules and are used to keep track of the context of the last command issued. This is done by storing the names of, say, the model and part that the current command affected.

Then, when the user issues the next natural-language command, if the system is unable to find a rule within a given group with a score higher than the minimum allowable score for rule execution, then the system checks all the rules in that group a second time while including the state variables as if they were part of the user's input phrase.

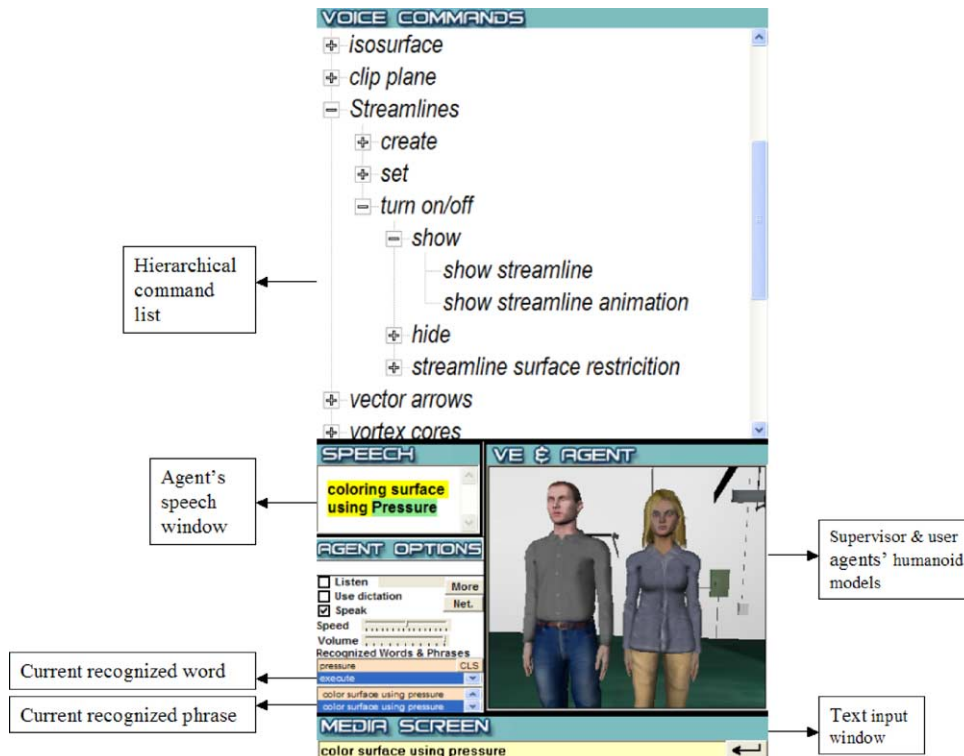


Fig. 3. Snapshot of the web interface of the IAF.

For example, if in the first command the user says “color the airplane wing using pressure,” the system will store “airplane” and “wing” in the state variables. The next command, the user can say “color by temperature.” The system will first try to execute the command as is and fail, then it will append the state variables and execute the command “color airplane wing by temperature.”

Rules can also set ‘general-purpose’ global variables. General-purpose global variables do not affect the total rule score but they do affect the rule’s execution by affecting the agent’s reply to the user, or the script that is sent to the visualization engine. Global variables can also affect the path that the rule takes through the rule hierarchy by specifying the name of the rule group that any given rule connects to.

The system can interpret commands with fuzzy words, such as “a lot” or “a little.” For variables, whose range is known, this is done by obtaining the maximum and minimum values of the variable in question from the visualization package and then mapping a pre-selected percentage of the range to each fuzzy word. For quantities that have no maximum or minimum (such as the number of contour lines), the system maps a percentage of the current value to each fuzzy word. An example of a natural sub-language command along with the manner in which it triggers different rules to formulate the system’s response is given in Appendix B.

In the case where the user issues an ambiguous command, such as omitting the type of a clip plane in its creation command, the expert system can request further information to allow the correct interpretation of the user’s intentions. In addition, the expert system can provide information about the properties and states of the various simulation objects and the values of all the simulation variables (e.g. answer the question “what is the maximum pressure?”) Additional roles that the system can perform are to introduce the background information concerning a particular simulation, to answer questions related to the visualization tools,<sup>5</sup> to answer questions concerning the simulation being viewed, or to identify important features of the simulation for the user.

### 3.2. Supervisor agent

The role of the supervisor agent is to monitor the multi-user collaboration session and alert the users when the interval between the commands issued by them is too small. In this case, the supervisor agent states the names of the users who are issuing commands simultaneously and instructs them to issue commands one at a time. When the system remains idle for a pre-selected length of time,

the supervisor agent also alerts the users by stating that it is waiting to receive new commands.

### 3.3. Virtual characters

To add a social dimension to the interface [8], each of the server and client web-based facilities has near-photo-realistic full-body virtual characters (also known as “user interface agents” or “avatars”) that are used in the roles of user agent (virtual assistant) and supervisor agent (Fig. 3). Every web-based facility in the visualization session has a user agent. The role of the user agent includes confirming the last issued command by speaking the user’s command as it is being executed. A text version of this audible output also appears on the screen.

The avatars’ lip movements are synchronized with the speech. The avatars can express emotions through facial expressions and voice tone, and can also gesture using the head, body and hands. Special tags inserted into the output text that is spoken by the avatar can be used to display pre-defined facial expressions and gestures. Multiple gestures can be run simultaneously, thus enabling a very large number of combinations of body language. For example, the agent can display a certain emotion by combining a facial expression, a bodily posture, and a gesture. The facial expressions and gestures along with the synchronization of the voice output with the agent’s lip movements all act to give the agent human-like qualities. The agent’s gestures can also be used to point at a certain feature of the model.

### 3.4. Multimodal interfaces

The user can speak or type his/her command in natural sub-language. The user can also select natural sub-language commands from a hierarchical clickable list in the web user interface (Fig. 3) or on a hand-held PDA. The last issued command can be stored in that list by issuing the command “store.” Multiple wireless handheld devices running hierarchical command menus and gesture recognition devices can be connected to the web-based facility, and enable multiple users to issue commands to the visualization package. Having a wide variety of interaction modalities available is important since the user can choose the modality that is best for a given task, and can easily switch to a different modality [1].

Several advantages can be cited for issuing commands in natural sub-language via the web-based facility by the handheld computers and gesture recognition systems, rather than connecting directly to the visualization package. Instead of sending lengthy Ensign scripts, a command can be issued by sending only a few words. Not only can the interface program running on the device be made smaller, but also the person(s) programming the device no longer need to have the specialized knowledge of the scripting language of the visualization package. The handheld and gesture recognition

<sup>5</sup> Examples of such questions are: “what is an iso surface?”, “how are vortex cores computed?”, or “how can I visualize the flow around the model?”

devices can also benefit from the capabilities of the intelligent user agent. The name of the user issuing the current command is added to the voice and text output of the intelligent user agent, hence allowing all the users in the session to know who is currently issuing commands to the environment and what commands are being issued. This is especially useful for distributed collaborative sessions. A session report can also subsequently be produced that identifies all the commands issued by each user, and the response given by the intelligent agent. A brief description of some of the input interfaces that can be used with the system is given subsequently.

#### 3.4.1. Voice input interface

Natural-language is one of the many interaction modalities with virtual environments. The accuracy of speech recognition has significantly increased in the last few years. The voice input interface used in the present study employs Microsoft Speech API 5.1 and the built-in Microsoft speech recognition engine for voice recognition. The vocabulary file within which Microsoft Speech API searches for possible matches contains around 1000 words and short phrases. When speaking, the user needs to pause briefly (0.2 s) between words in order to get good recognition accuracy.

The voice input interface (Fig. 3) displays the current recognized word as well as the entire current sentence. When the user is done giving a voice input, he/she can either say “execute” to execute a command or “answer” to request an answer to a question. To cancel the entire current input sentence, the user says “cancel”, while to erase just the last spoken word, the user says “backspace”. The command “repeat” is used to issue the last command again and the command “undo” is used to cancel the last change.

One of the problems encountered sometimes with the system is the difficulty of recognizing certain words as spoken by a given individual. Even after proper training of the voice recognition software, recognizing certain words can require several tries or is not possible at all. Errors in recognition can occur when the system sometimes consistently recognizes a word that is different from the intended word when spoken by a given individual. To alleviate this problem, the vocabulary file, which contains a list of all possible words that are recognized by the system, can be dynamically changed. In this case, when the user says “backspace” to erase a word that was recognized in error, the last recognized word is temporarily removed from the vocabulary file. This prevents the system from repeating the same mistake. Moving to the next word or saying “cancel” resets the vocabulary file to its initial state.

Although voice recognition has taken big strides in recent years, still the technology requires a lot of improvement to enhance the recognition rate specially when the user is speaking at a normal pace or in a noisy environment. The technology can be improved by combining natural speech understanding with other interaction styles. It is possible to

capitalize on the additional cues for disambiguation provided by other modalities (such as lip movement recognition) to improve the voice recognition accuracy [9,10].

#### 3.4.2. Handheld interfaces

Using wireless handheld devices, any number of users can interact with the Ensign collaborative session. The handheld devices use hierarchical menus to issue text commands using natural sub-language to Ensign via the web-based facilities. The wireless handheld devices connect to the web-based facility using a TCP/IP network socket interface. Fig. 4 shows a snapshot of one of the Ensign menus on a handheld computer.

#### 3.4.3. Gesture interfaces

Working in an immersive virtual environment allows the use of several modalities to interact with the system. Some modalities are best suited for view manipulation functions such as rotation, zooming or panning, including hand gesture, head movement and eye tracking. A variety of hand gesture recognition devices are available which can interpret gestures. These include gesture gloves, special sensors and image processing [11,12]. Voice commands can be associated with different hand gestures.

#### 3.4.4. Tracking and haptic interfaces

Head, hand and eye tracking (via image processing, ultrasound, or electro-magnetic tracking) are used for navigation and object selection and manipulation. Haptic devices (such as gloves and touch pens) allow users to interact with the virtual objects by touching them. Tracking and haptic interfaces require information from



Fig. 4. Snapshot of a typical Ensign menu on a wireless handheld device.

the visualization engine. Therefore the IAF interfaces with those devices through the visualization engine.

#### 4. The visualization engine

The graphics post-processing package Ensign (from CEI) [4] is used in the present study to demonstrate multimodal interaction between the IAF and commercial visualization packages. Ensign is an engineering visualization package with many advanced visualization features. These features include isosurfaces, clip planes, streamlines, contours, elevated surfaces, 2D plots, separation and attachment lines, vortex cores, and shock waves. The package can also produce flipbook and key-frame type animations and provides extensive tools for manipulating large engineering datasets. The graphics output can be displayed to multiple users using heterogeneous platforms including high-end virtual reality facilities. Ensign can interface with a variety of input devices including head and hand tracking.

#### 5. Application to visualization of engineering simulations

Applications of the foregoing environment to computational fluid dynamics (CFD) and computational structural mechanics (CSM) visualizations are described herein. All the commands for engineering visualization applications can be broken down into an action word, a part identifier and a variable identifier. Examples of action words are “color”, “contour”, “show”, “hide”, “set”, and “delete.” Parts can be identified by their name such as “isosurface”, “vortex cores”, or by their number<sup>6</sup> such as “part 6.” In case multiples of a given part type exist, an additional numerical identifier is needed when selecting a part of that type by name such as “clip plane one” or “third contour lines.” A parameter in the form of a number or an attribute can also be used to set a specific property such as color, opacity, size, and shape of the different visualization objects. Rules for identifying the natural-language names of seventy variables used in engineering Computational Fluid Dynamics (CFD) and computational structures were added to the expert system to allow it to interpret the variable identifier part of the user’s command. The natural-language names of the first and second derivatives of these variables with respect to around 100 independent variables can also be recognized by the system. Recognizing complex derivative variable names is enabled by the hierarchical structure of the rules with each level of rules interpreting a part of the variable name. An example of the steps involved in the interpretation

and execution of a command using the expert system is given in Appendix B.

#### 6. Concluding remarks

An interrogative visualization environment is presented for the interactive display and querying of large datasets. The environment encompasses two major components: A visualization engine for displaying the simulation and a web-based IAF. The IAF includes a hierarchical rule-based expert system for natural sub-language communication between the user and the visualization engine; humanoid models of a user agent and a supervisor agent that add a social dimension to the interaction while assisting the user in performing his/her tasks; and multimodal interfaces for capturing the user’s input that include voice, wireless handheld devices, and various gesture recognition devices.

To enable multiple users in a distributed collaboration session to interact with the visualization, the IAF consists of multiple facilities. One of the facilities is a server facility while the rest are client facilities. All the facilities can receive multimodal command input in the form of natural sub-language text or voice that is subsequently processed by the rule-based expert system residing on the server facility.

Future work will include making the interface adapt and optimize the flow of information to the users through the inclusion of other devices that can sense and recognize the affective state of the user, and identify their state of knowledge. The rate of information flow is reduced when the user is tired, also more explanation can be provided when dealing with a non-expert user.

#### Acknowledgements

The authors acknowledge the help of the staff of Computational Engineering International in the use of the Ensign visualization package. The human-like avatars display engine was provided by Haptik Inc. Natural male and female text-to-speech voices were provided by NeoSpeech. The LEA and IVRESS software systems were provided by Advanced Science and Automation Corp. The present work is supported by a NASA cooperative agreement NNL-0-4A-A05A.

#### Appendix A

The commands in the hierarchical rule-based expert system used in the present study can be grouped into five categories. The list of the commands in the five categories is shown in the Table A1.

<sup>6</sup>The part number is automatically generated by the visualization package when a part is created.

Table A1

Category	Syntax	Explanation
Declaration statements	<b>DEF</b> ruleName <b>RuleType1</b> {...}	Declares a rule with the name 'ruleName.' All of the commands of this rule are within the curly brackets.
	<b>DEF</b> groupName <b>Group</b> {children [...]}	Declares a group of rules with the name 'groupName.' Any number of rules can be placed between the square brackets.
	<b>DEF</b> variableName <b>Variable</b> "variableValue"	Declares a variable with name 'variableName' having a value 'variableValue.' This command is used to initialize variables when the system starts.
Rule score calculation	<b>startingScore</b> score	The default initial score of all rules is zero. This command assigns an initial score to a rule equal to score. It is usually set to 100 to execute the rule by default if none of the rules within a certain rule category are triggered.
	<b>require</b> PlusScore MinusScore ["keyword1" ... "keywordn"]	Defines a required word along with all its possible synonyms. If one of the 'keywords' is found in the command, then PlusScore is added to the total rule score. If none of the 'keywords' is found in the command, then MinusScore is subtracted from the rule's score.
	<b>ignored</b> ["keyword1" ... "keywordn"]	Defines a set of ignored words. The ignored words do not affect the total rule score.
	<b>scoreOther</b> score	This numeric value is added to the total rule score for every word which is neither a required nor an ignored word. The value of 'score' should be negative.
	<b>readNumber</b> variableName	Reads a real number from the input command and stores it in variableName.
Variable manipulation	<b>readInteger</b> variableName	Reads an integer number from the input command and stores it in variableName.
	<b>setVar</b> variable Name "variable Value"	Defines a variable 'variableName' and sets its value to 'variableValue.'
	<b>getVar</b> variable Name [script]	Defines a variable 'variableName' and gets its value from Ensign using the script that is between the square brackets.
	<b>strcat</b> variable Name "variable Value"	Appends the string 'variableValue' to the variable variableName.
	<b>incVarPercent</b> variableName value	Increases a real number variable value by a desired percentage. The command <b>decVarPercent</b> does the opposite.
	<b>incVarVal</b> variableName value	Increases a real number variable value by a desired value. The command <b>decVarVal</b> does the opposite.

Table A1 (continued)

Category	Syntax	Explanation
	<b>incMeanPercent</b> variableName1 variableName2 value	Increases the mean value of two real number variables by a desired percentage. The command <b>decMeanPercent</b> does the opposite.
	<b>incMeanVal</b> variableName1 variableName2 value	Increases the mean value of two real number variables by a desired value. The command <b>decMeanPercent</b> does the opposite.
	<b>incRangePercent</b> variableName1 variableName2 value	Increases the range value of two real number variables by a desired percentage. The command <b>decRangePercent</b> does the opposite.
	<b>incRangeVal</b> variableName1 variableName2 value	Increases the range value of two real number variables by a desired value. The command <b>decRangePercent</b> does the opposite.
	<b>statei</b> stateValue	Sets the state variable <i>i</i> (where <i>i</i> goes from 1 to n) to the string 'stateValue.'
IO Com- mands	<b>output</b> "message"	Instructs the agent to output (speak and display) the given message. The message is scanned for variables and before it is sent, the variable names are replaced by the variable values.
	<b>script</b> [...]	Sends the given script to Ensign. The script is scanned for variables and before it is sent, the variable names are replaced by the variable values.
Hierarchical properties	<b>connect</b> rule GroupName	Instructs the agent to connect to another rule with the name 'ruleGroupName'.
	<b>connectName</b> variableName	Instructs the agent to connect to another rule whose name is in the variable 'variableName'.

## Appendix B

An example of a voice command is presented herein, along with an illustration of how the rule-based expert system interprets and executes this command. In this example, five steps are followed for command interpretation. The command "Color surface using temperature" is used as an example.

(1) The first words that are recognized by the expert system is the action word "color" or "shade" combined with one of the words in the second line of the following rule. The action words "set" and "change" receive a score of -100 since they are associated with a different action command that colors the object using a single color such as "red" or "blue". The ignored words take a score of zero. The variable "action\_name" is used in the system's audio

response in step 5. The variable “action\_script\_part” determines the rule group that will be searched after the part name selection group (step 3). The variable “action\_script\_part\_noselect” contains the name of the group that the part name selection group links to if it is not able to find a match for a part name. After executing the rule, the expert system is then directed to the “part\_name\_select” rule group to search for the object on which the coloring action will be performed.

```
DEF color_using RuleType1 {
  require 50 0 ["color" "shade"]
  require 50 0 ["with" "using" "by" "use"]
  require -100 0 ["set" "change"]
  ignore ["I" "want" "to" "the" "please"]
  setVar action_name "coloring"
  setVar action_script_part "derivative_select"
  setVar action_script_part_noselect "derivative_select"
  setVar derivative_script "variable_select"
  setVar action_script "actions_colorby"
  setVar firstd_script "actions_colorby"
  setVar secondd_script "actions_colorby"
  connect USE part_name_select
}
```

(2) The expert system then searches the rule group “part\_name\_select” containing all the possible object names for a match. In this case the rule “part\_name\_select\_surface” is the one that is executed. Note that the second line of the rule contain words having a score of -100 that are associated with other rules of the “part\_name\_select” group that also have “surface” as a keyword. The state variable “state1” allows the expert system to recognize the context within which the next user command will be issued. The system then connects to the rule group given by the variable “action\_script\_part” which is “derivative\_select.”

```
DEF part_name_select_surface RuleType1 {
  require 100 -100 ["model" "surface"]
  require -100 0 ["iso" "clip" "elevated" "hidden"]
  ignore ["I" "want" "at" "to" "the"]
  state1 surface_current
  setVar selected_part ["(CASE:Caseselectd_case)surface"]
  setVar text_name "surface"
  setVar select_by "name"
  setVar part_select_begin "select_partname_begin"
  setVar part_select_end "select_partname_end"
  connectName action_script_part
}
```

(3) Since there is no derivative in the requested variable, the system skips the rules for first and second derivatives. The rule for no derivative will then be triggered by default. The variable “variable\_name” is set to blank to remove any value stored therein from previous commands. Note that had the variable in question been “the sensitivity of temperature with respect to the thermal conductivity” for example; the rule for first derivative

would have been triggered in this step and the variable “variable\_name” would have been set to “the first derivative of.” The variable “action\_script” would have also changed to refer to the group used for first derivative variable selection. The system then connects to the rule group given by the variable “derivative\_script” which is “variable\_select.”

```
DEF derivative_none RuleType1 {
  startingScore 100
  setVar variable_name ""
  connectName derivative_script
}
```

(4) The rule for temperature from the rule group “variable\_select” is then triggered, and the word “temperature” is appended to the variable “variable\_name.”

```
DEF variable_temperature RuleType1 {
  require 100 -100 ["temperature"]
  require -100 0 ["normalized" "log" "stagnation"]
  require -100 0 ["range" "mean" "average"]
  ignore ["I" "want" "at" "to" "the" "by" "using" "magnitude"]
  setVar activate_command "variables: activate"
  strcat variable_name "temperature"
  setVar short_name "Temperature"
  connectName action_script
}
```

(5) Finally, the rule group “actions\_colorby” given by the variable “action\_script” is executed. Note that the system automatically replaces any variable name it recognizes within the “output” or “script” commands by its preset definition:

```
DEF actions_colorby_variable RuleType1 {
  startingScore 100
  output "action_name text_name using variable_name"
  script [command: part_selection_by select_by
  part: part_select_begin
  selected_part
  part: part_select_end
  activate_command short_name
  part: modify_begin
  part: colorby_palette short_name
  part: modify_end
  ]
}
```

The system executes the script given between the square brackets in the previous rule and speaks the output: “coloring surface using temperature.” Note that the three slightly different voice commands: “use temperature to color model”, “shade surface with temperature”, and “color model by temperature”, will trigger the same set of rules described above and generate the same system output.

Alternatively, the aforementioned command can be issued as two successive commands: “select surface”, and “color using temperature.”

## References

- [1] Oviatt S, Bers J, Cohen P, Holzman T, Wu L, Winograd T, Vergo J, Landay J, Duncan L, Larson J, Suhm B, Ferro D. Designing the user interface for multimodal speech and pen-based gesture applications: state-of-the-art systems and future research directions. In: Carroll John M, editor. *Human-computer interaction in the new millennium*. New York: ACM Press; 2002.
- [2] Arias E, Eden H, Fischer G, Gorman A, Scharff E. Transcending the individual human mind: creating shared understanding through collaborative design. In: Carroll John M, editor. *Human-computer interaction in the new millennium*. New York: ACM Press; 2002.
- [3] Wasfy T, Noor A. Rule-Based natural-language interface for virtual environments. *Adv Eng Software* 2002;33:155–68.
- [4] *Ensign User Manual: for Version 7.6*. CEI: Apex NC, 2003.
- [5] Grishman R, Kittredge R. *Analyzing language in restricted domains: sublanguage description and processing*. New York: Earlbaum Associates; 1986.
- [6] Hopgood A. *Intelligent systems for engineers and scientists*. Boca Raton, FL: CRC Press; 2001.
- [7] Ball T, Colby C, Danielsen P, Jagadeesan L, Jagadeesan R, Läufer K, Mataga P, Rehor K. Sisl: several interfaces, single logic. *Int J Speech Technol* 2000;3:93–108.
- [8] Müller W, Spierling U, Alexa M, Rieger Th. Face-to-Face with your assistant. Realization issues of animated user interface agents for home appliances. *Comput Graph* 2001;25:593–600.
- [9] Deng L, Huang X. Challenges in adopting speech recognition. *Communications of the ACM* 2004;47(1):69–75.
- [10] Roush W. Digital Lip Reader. *Technol Rev* 2003;106(7):26.
- [11] Rekimoto J. GestureWrist and gesturepad, unobtrusive wearable interaction devices. *Proceedings of IEEE International Symposium on Wearable Computer (ISWC 2001)* 2001.
- [12] Yamauchi Y. Gesture-based ping-pong game using realtime depth-image input device. *SIGGRAPH*, New Orleans, LA, July 2000.