INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

Bell & Howell Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

Human interface and interaction in the WITS training system

Atallah Mourad

B.Eng., (McGill University at Montréal, Canada), 1995

Electrical Engineering Department
McGill University
Montréal

July 1998

A thesis submitted to the Faculty of Studies and Research
in partial fulfillment of the requirements of the degree of
Master of Engineering

Abstract

This thesis research addresses the challenges involved in the design of WITS, the Welding Intelligent Tutoring System. The project was motivated by the desire of providing a realistic training environment for maintenance workers involved in the repair of grounding fixtures on electrical power station equipment. The thesis elaborates on each of WITS' three subcomponents: the user interface and peripherals, the expert system, and the 3D virtual environment. The research focuses on the creation of a 3D representation of the operator and his tools. This virtual operator is built to have basic interaction capabilities namely touch and grasp and is made capable of communicating with the expert system which contains the task expertise and thus provides the necessary coaching. Keyboard and voice command interfaces were also developed so that users with no access to a graphical virtual environment can still take full advantage of the expert system.

Résumé

La recherche de cette thèse s'attaque aux défis impliqués dans ls conception du système d'enseignement de la soudure appelé WITS. Le projet a été motivé par le désir de fournir un environnement réaliste d'enseignement aux ouvriers qui s'occupent de la réparation des appareils de mise à la terre sur les unités de production électrique. La thèse décrit chacune des composantes de WITS: l'interface usager et ses périphériques, le systeème expert et l'environnement virtuel 3D. La recherche aboutit à la création d'une représentation en 3D de l'opérateur et de ses outils. Cet opérateur virtuel possède les capacités d'interaction essentielles: toucher et saisir. Il est aussi capable de communiquer avec le système expert qui contient l'expertise de la tâche. Des interfaces de commande par clavier et la parole ont été développées pour que les usagers qui n'ont pas accès à un environnement virtuel graphique puissent se servir tout aussi bien du système expert de WITS.

Acknowledgments

My first thanks go to my thesis supervisor, Professor A. S. Malowany, who has given me guidance and encouragement throughout the research, and who has provided me with the rich advice and opportunities that have made these graduate studies the very rewarding experience they have proven to be.

I would next like to thank my colleagues in the VR lab, especially Khaled Kaddoura and Evgeny Slavkoff for their close collaboration in this research.

I would also like to thank M. R. Laliberté of IREQ, M. J. Gagnon of Hydro-Québec, and M. M. Lemay of ERICO for their collaboration, and FCAR for their financial support.

Finally, special thanks go to my parents who through their insistence succeeded in making this thesis work materialize.

Liste of Acronyms and Abbreviations

CLI	Command Line Interface
CLIPS	C-Language Integrated Production System
DAG	Directed Acyclic Graph
DHM	Dextrous HandMaster
DOF	Degrees-of-Freedom
HCI	Human Computer Interaction
HMD	Head-Mounted-Display
ITS	Intelligent Tutoring System
LCD	Liquid Crystal Display
LED	Light-Emitting Diode
MRI	Magnetic Resonance Imaging
SEDA	Expert System for Diagnosis of Apparatus
UMDH	Utah/MIT Dextrous Hand robot manipulator
VE	Virtual Environment
VO	Virtual Operator
VR	Virtual Reality
WITS	Welding Intelligent Tutoring System
WTK	WorldToolKit

Table of Contents

List of Tables

List of Figures

Chapter 1

Virtual Reality and Human Interface Review

The research described in this thesis was carried out at the McGill Virtual Reality Lab where work is being done on 3D representation and interaction of environments, tools, and human operators in virtual reality. This thesis presents the design and implementation of a prototype virtual reality training system developed to facilitate and enhance manual interactions addressed in the current Hydro-Québec welding course. The system permits trainees to take part in the welding procedure in a realistic environment where they can interact with realistic representations of the different welding object components. The WITS tutoring system was developed in collaboration with École Polythechnique de Montréal, the IREQ power division of Hydro-Québec, and ERICO, a supplier of electrical products.

An intelligent tutoring system (ITS) is usually described in terms of its requisite capabilities [Beck and al. 1996]. An ITS should model the student being taught and the experts in the task domain so that the tutoring process can adapt progressively to the performance and needs of the student. Therefore, it is imperative that the system correctly models both the procedures and the appearance and behavior of the learning environment; that is, the system must support both the procedure-related expertise and the simulation.

This chapter presents a survey covering the three major components in achieving such an ITS: the expert system which encapsulates the student and expert models by providing the procedure-related expertise, the virtual reality interface which allows for high fidelity hands-on realistic simulation, and finally the human interaction interface which allows the trainee to take part of this simulation in a learn-by-doing approach.

1.1 Expert Systems and Virtual Reality applications

An expert system can best be described as a tool to assist human decision-making in knowledge-intensive tasks requiring expertise. A view of the field today shows that its range of application spans diverse disciplines, with the greatest number of developed systems in business, manufacturing and medicine [Durkin.1996]. Although Durkin shows that the greatest application area of expert systems has been in solving diagnosis tasks (30% of diagnosis applications rely on expert systems), Stefik explains that intelligent tutoring applications have an intimate history with expert systems [Stefik 1995] and it is clear that they will be more effectively used in conjuction with virtual reality systems and human interaction interfaces which are both rapidly advancing fields.

SMITHTOWH, an expert system designed to teach basic microeconomics, has been shown by Shute, in a large-scale evaluation, to produce equal results as those achieved though traditional economics courses and took students only half the amount of time traditionally needed to cover the whole material [Shute and al. 1990]. SHERLOCK, an expert system used to train technicians for electronic fault diagnosis of F15 fighter aircraft, has been described by Katz as a learn-by-doing system that allows trainees to achieve in 20 hours a level of technical expertise almost equal to technicians having three to four years of on-the-job experience [Katz and al. 1996]. Studies by Shute and Regian show the growing recognition of the value of training using expert systems in both the military and industry fields. They also argue that the use of such systems has shown a positive trend in efficacy where learning is accelerated without any degradation in the final outcome [Shute and al. 1997] [Regian and al. 1997].

The success of SHERLOCK, one of the most acclaimed expert systems, did not come

by itself. Other intelligent tutoring systems such as WEST and SOPHIE [Stefik 1995] provided the basis for the "coached practice environment" of SHERLOCK. WEST provided the Coaching tutoring approach where the system "looks over the shoulder" of the student and intervenes with criticism or suggestions only when necessary. SOPHIE introduced the "reactive learning environment" where the student is free to experiment with different ideas while the system would only provide criticism where appropriate.

With the increase of popularity of virtual reality (VR), expert systems started taking advantage of this new form of interaction to produce more realistic hands-on training systems. The use of such systems is slowly gaining terrain in application domains such as military, industry, commerce and medicine. The rest of this section outlines some current applications of virtual reality in each of these domains where high-tech diagnosis and training systems are quickly becoming a necessity.

1.1.1 VR in Medicine

Surgeons are beginning to use computer graphics to simulate surgical procedures for training, visual assistance in diagnosis, and prediction of surgical results [Delp 1990] [Pieper 1992]. These simulations need to be as realistic as possible. For the simulation to be useful to the surgeon as a training tool or surgical assistant, the surgeon needs to be able to manipulate the graphical representation as if it were the real object.

Paintanida describes the use of semi-transparent displays to superimpose remotely sensed images such as magnetic resonance imaging (MRI) onto a real image of the patient which gives the stereoscopic impression of looking into the patient [Piantanida 1992]. Stereoscopic images are in fact used in neurosurgery more often than we think. Maybe not as proposed by Paintanida, however the demand for such advanced display technology is very visible [Peters 1995].

The creation of a surgical virtual environment to both aid surgeons during operations and provide simulations for training, opens up many exciting possibilities. As stated by Sagar et al., the surgical procedure will be able to be viewed in new ways providing a new level of surgical experience [Sagar et al 1994]. Hunter et al. show the need for force

feedback along with stereo vision. They present a teleoperated microsurgical robot for eye surgery and discuss the technical implementation of visual feedback and how signal propagation delays present a problem affecting the use of such a tool [Hunter 1993].

Virtual reality is not only interesting for surgery, other fields in medicine have taken advantage of the technology. Some hospitals in Virginia and the Bronx used VR for the rehabilitation of physically challenged people [Coull 1992] [Kleinfeld 1995]. Coull describes how modified wheelchairs are used as a navigation tool combined with a Data-Glove to interact with objects in a virtual environment thus making an exhausting exercise enjoyable. Several other applications using VR to work with disabled people are mentioned by Nemire and Vanderheiden [Nemire 1994] [Vanderheiden 1994].

1.1.2 VR in Industry

The Boeing 777 is the first jetliner to be 100 percent digitally designed using three-dimensional solids technology [Adam 1993] [Esposito 1993] [Eberl 1994]. Throughout the design process, the airplane was "preassembled" on the computer, eliminating the need for a costly, full-scale mock-up. The 777 Division uses CATIA (computer-aided, three-dimensional interactive application) and ELFINI (Finite Element Analysis System), both developed by Dassault Systems of France and licensed in the United States through IBM. Designers also use EPIC (Electronic Preassembly Integration on CATIA) and other digital preassembly applications developed by Boeing. Using the three-dimensional solid images generated on the computer, the 777 airplane can be preassembled to position parts correctly, ensuring a proper fit. The benefits of this system are innumerable: improved quality of work; reduced changes and errors, and less rework, resulting in lower costs etc.

Working at Ames on the world's most powerful supercomputers, scientists predict the flight behavior of high-speed aircraft and spacecraft. State-of-the-art computers are used to solve complex aerodynamic equations so that various aircraft configurations can be tested by "flying the aircraft in the computer." This reduces both the time and cost of developing new aircrafts [Bryson et al. 1994]. Other VR applications at NASA Ames Research Center include the Virtual Planetary Exploration Testbed where very large

terrain data sets from the Viking mission of the 1970s are used to create a virtual world in which the user can travel over certain sections of Mars [Hitchner 1992].

The European Space Agency (ESA) developed a training tool for astronauts working in the Columbus space station. Correct cues are provided through the high-definition graphics, sound and force feedback devices [Bagiana 1993]. Also at ESA, the EUROSIM simulator permits the training on dangerous landings such as an unpowered winged vehicle [Buc 1994].

The need for VR applications in industry is not limited to flight simulators. Other areas such as construction are now interested in VR technology. Hughes et al. describe the problems one equipment manufacturer had in developing a control interface for a pipe-lifter. The original design had eight levers each controlling one of eight degrees of freedom. This proved intractable for operators because the linear arrangement of the levers, combined with the nonlinearities of the system resulted in an unintuitive mapping into the control space. Hughes et al. solved the problem with a more intuitive double-joystick and processors to linearize the control task [Hughes et al. 1989]. Sturman, however, argues that micro-processors are being used to minimize the nonlinear effects of the actuators, and that what is needed is the development of a more intuitive user interface [Sturman 1992]. Thus construction can take advantage of the advances in VR user interface and device technology. A more appropriate example of the use of VR in the construction industry is a VR application used by Caterpillar Inc. to test drive new construction vehicles. They were able to test various aspects, such as vehicle design and visibility, from the operator's position in the cab [Jones 1995]. Finally, demand for VR is starting to appear in the power industry. One of the first VR training environments for substation operators, ESOPE-VR, has been developed at McGill University for Hydro-Québec. The simulator permits trainees to carry out all the switching operations necessary for their work in absolute safety, while staying in a realistic environment [Okapuu-von Veh 96] [Shaikh 1995].

1.1.3 VR in the Military

"If necessity is the mother of invention, it must be added that the Defense Department is the father of technology" [Rheingold 1992]. The military has always been the prime contractor for the most significant innovations in computer technology. The Army produced the first electronic digital computer in the 1940s and was the first to do research on head-mounted displays in the 1980s. This is no surprise since the military budget can afford to spend on high-tech research and development.

VR technology is very important for the military mainly because of its eventual cost-reduction benefits. Training soldiers on a real physical terrain with real equipment and ammunition is very costly; a good VR system can replace these training sessions and thus decrease cost. For instance, training a soldier to use an Apache helicopter can cost about $335K per one and a half hour. On the other hand simulating the exercise costs just $143 [Roos 1995]. Holzer states that the use of simulation enabled the U.S. Navy to fire only thirty Sidewinder missiles in order to test the current version, as opposed to over 300 live firings for the previous one [Holzer 1994]. Kozak reports similar simulators used by the US Navy, the Air Force and other services. The simulators were used to train pilots on a battlefield developed using terrain data for the Golf war [Kozak 1993].

At the Naval Ocean Systems Center (NOSC) in Hawaii, researchers are developing an interface to steer a robot submarine down to hook a cable onto thermonuclear-armed missiles. They want the interface to give good feedback to the operator which would teleoperate the robot arm to grip a wrench and dismantle the detonator of the weapon, and that in deep ocean!

SIMNET is a project funded by the Defense Advanced Research Projects Agency (DARPA) that includes over two hundred tank simulators, located in Germany, Washington, D.C., Fort Knox, Kentucky, and a few other places. Although they are geographically dispersed around the planet, these telecommunication-linked simulators interact with the same virtual battlefield in real time. Using the highest-speed communication lines of MILNET, these four-person simulators make it possible to fight an entire war game in cyberspace [Thorpe 1987]. Considering the high cost of training soldiers in phys-

ical war games, and the declining cost of computing power, SIMNET is an economical option. Note also that war games in cyberspace certainly use less petroleum and create fewer conditions for fatal accidents. Nonetheless, Card found an evil use of such a game in his science-fiction novel "Ender's Game", where a crack crew is trained in a SIMNET-like setup to use state-of-the-art virtual weaponry to destroy an entire civilization and that in one of the training exercises the virtual weapons are switched to teleoperated real weapons.

1.2 Human-Computer Interaction

The field of human computer interaction (HCI) can be broken down into three main levels of study: the theoretical or psychological level, the device interface level, and the psychophysical level. Extensive research has been done on each of these levels. Unfortunately, the complexity of human behavior makes HCI a difficult area in which to validate theories. Results tend to be context dependent (although, less so at the psychophysical level).

In an interesting essay on this topic, [Carroll and Campbell 1988] argues that the artifacts developed in HCI—the devices, techniques, and systems—embody theories, but the theories they embody are not powerful enough to guarantee success in other applications. They claim that many artifacts may not be reducible to explicit theory and may be incomprehensible apart from the situations in which they are used. Thus, they term HCI a *design science.* This makes reliance on prior work tenuous because, by its paradigmatic nature, each theoretical work is uniquely inseparable from the specific application generating it.

Good HCI models can be appropriated from previous work [Foley et al. 1996], but it is difficult to use theories about why the models are successful. This is not to say that prior work is irrelevant, but that where scientists habitually seek universality through theories, universal application is difficult to abstract from HCI theories. Thus, one must be aware of the fragile nature of theoretical work. Caution must be taken in translating results to other contexts.

In a related report, [Carroll and Campbell 1988] describes how scientifically rigorous psychological approaches to HCI, involving testing of low-level phenomena, have had little practical impact when expanded to general application. On the other hand, he says, attempting only to formulate models of the user's mind and actions ignores the important human-factors aspect of HCI. The conclusion to be reached is that environment, task, device, and human factors must be integrated for practical HCI development.

In the field of ecological psychology, researchers believe that the psychology of human computer interaction must be studied in terms of the human-task environment in which the actions occur. [Flach 1990] provides a good introduction to the HCI implications of ecological psychology theories. [Vicente and Rasmussen 1990] describe *ecological interface design* as a process of extracting features of tasks at various levels so as to make the geometry of the interface reflect the nature of the task in a way that exploits direct perception. One of the goals of ecological interface design is to allow operators to act directly with the task, making the intermediary sensor displays as functionally transparent as possible.

Another important concept is that of *direct manipulation* [Shneiderman 1983]. This is where the user experiences interaction as being directly with the objects of interest rather than through an intermediary system. The Apple Macintosh operating system uses direct manipulation for most of its operations. To move files from one directory or folder to another, a person clicks the mouse cursor on the files to be moved and drags them to the new folder. In a traditional operating system a command is typed to the operating system and (conceptually) *it* does the operation. [Hutchins, Hollan and Norman 1986] talks about *directness* as an impression or feeling about an interface resulting from the commitment of fewer cognitive resources. The more a person has to think about an interface, the more a person feels removed from the task. They describe *distance* as "the gulf between the user's goals and the way they must be specified to the system."

Expanding on the theme, [Laurel 1986] describes the "computer-as-a-tool" mode of interface as an artifact of the evolution of interface design. When people use a computer, she says, they are interested in the application, not the use of a computer. Therefore, the computer should become transparent. The interface should take on the aspects of

the task and the user should become an operator, or agent, in the domain of the task, rather than a distanced observer working through an intermediary operating system and command structure. [Wixon and Good 1987] also supports the notion of transparency and argue that "transparency" and "support for breakdown" should be used as measures for the usability of computer systems. They claim that schemes of hierarchal categories of user interface are misleading, and that most computer systems have a continuum of use and modality that crosses category boundaries. They conclude their essay with the hope that designers and researchers will think along continuous dimensions of usability rather than rigid categorizations.

At the device interface level, the most important general developments have to do with systems that describe virtual input devices and taxonomies of input devices. [Foley and Wallace 1974] described input tasks so as to be independent of device. Their purpose was to allow the discussion of input models without the dependency of hardware technologies. They classified four virtual devices, the *pick*, the *button*, the *locator*, and the *valuator*. This since has been refined and integrated into the GKS system [Enderle, Kansy and Pfaff 1984] as *pick*, *choice*, *locator*, *valuator*, *stroke*, and *string*. These categories can be used to provide a device-independent input library, but do not take into account the properties of specific devices that make them suited for a particular task. A trackball, tablet, or mouse can be used as a locator device, but provide different levels of performance depending on the task.

In an effort to address the human factors of devices, Buxton's *Taxonomy of Input Devices* [Buxton 1990] categorizes input devices in terms of properties sensed (position, motion, or pressure) and degrees of freedom. [Card 1991] has improved upon this taxonomy by including both the continuous and discrete properties of input devices. Buxton has taken the next step and proposed a model which accounts for the hybrid discrete/continuous properties of devices. This model uses state changes to describe input sequences and relates tasks and devices using these state descriptions.

There have been many studies of specific devices and human performance. However, as has been stated earlier, the majority of them are too context dependent to be generally useful. One early experiment stands out. Most researchers agree on the validity of *Fitts*

Law [Fitts 1954] or variations thereof. Fitts tested the time it took people to accurately move small objects from one point to another. He found that target acquisition times have a logarithmic relationship to the size and distance of the target. This is expressed formally as $MT = a + b \log_2 \frac{2A}{W}$ where MT is movement time, A is movement amplitude (distance between start and finish), W is target width (size), and a and b are application and device dependent constants. [Card , English and Burr 1979] confirmed Fitts Law for two-dimensional computer interaction (selecting text on CRT displays), and found values for a and b for different devices and tasks. When similar tests are brought to three-dimensional computer input, results are less conclusive. Researchers have found that display styles (such as stereo versus perspective views) and input metaphors affect the results [Beaten et al. 1987] [Ware 1990] [Stanney 1995]. This may have to do with the increased cognitive load of correlating the two-dimensional screen image (or synthetic stereoscopic image) with the subject's mental model of the three-dimensional space.

1.2.1 Interaction Devices and Techniques

Camera-based devices

Most camera-based interaction systems rely on LEDs strategicaly placed on the users body to aid in detecting the different body parts. However, some systems go one step further and eliminate the need to wear such cumbersome equipment. They use pure image processing to determine the components of the images and thus locate the different body parts.

LED-based systems For many years, biomechanics labs across the country used LED-based systems, such as Selspot by Selcom or OptoTrak by Northern Digital [Selcom], to track the motion of the body and limbs [Mann and Antonsson 1983]. These systems use multiple infrared cameras focused on the subject wearing LEDs activated in sequence. A computer system analyzes the position of each LED in each camera's visual field and calculates the world-space position of the LED. These systems are limited by the computer time needed to calculate the world-space position, occlusions of the LEDs by the body,

lengthy calibration procedures, and positional accuracy. [Mann and Antonsson 1983] were able to get the positional accuracy of the Selspot system to 0.1 percent of the visual field; sufficient for limb movement, but not for fine finger motions. Nevertheless, LED systems have been used successfully as tools for clinical analysis of body movement.

In the early 1980's researchers at the MIT Architecture Machine Group and then the MIT Media Lab used a camera-based LED system to track body and limb position for real-time computer graphic animation [Ginsberg and Maxwell 1983] [Purcell 1985]. The LED position data was sent to a computer graphic rendering system which drew a representation of the user's body, mimicking the user's motions. This work included a glove studded with LEDs to track finger motion. [Hall 1985] mentions using the LED glove in an experimental system that performed table-lookup on finger postures to allow input by finger spelling. This simple system leads into the use of the hand for signed language, but other than this, no attempt was made by them to interpret finger or hand motions.

Poizner and other researchers at the Salk Institute in La Jolla, California, also used a camera-based LED system to analyze signed language. In 1983 they reported on their research to analyze hand motions of American Sign Language (ASL) using point light displays [Poizner et al. 1983]. They placed the LEDs on the hand and arm so as to minimize occlusion during signing. Analysis was done in non-real-time after the motion data had been collected. This avoided some of the computational speed problems usually associated with moving point light displays. They proposed various analytical techniques, including feature analysis and frequency analysis, from which to qualify the linguistically relevant features of signed language. Although their interest was in understanding the phenomena of signed languages, their work can be adapted to computer understanding of a gestural lexicon or gestural control. Of special relevance are their methods of motion analysis to derive useful metrics of signing and their mapping of signs into various dimensions of a visual-articulatory space. They contend that humans can articulate and interpret hand motion along these dimensions. By using these same dimensions in gestural control, perhaps complex (i.e., powerful) yet manageable methods for gestural control can be developed.

There has been comparatively little other work in capturing hand motion using camera-based systems. The main problems with image-based visual tracking of the hands are that the resolution of conventional video cameras is too low to both resolve the fingers easily and cover the field of view encompassed by natural hand motions; the 30 (or 60) frames per second conventional video technology is insufficient to capture rapid hand motion; fingers are difficult to track as they occlude each other and are occluded by the hand (a common occurrence); and computer vision techniques are not developed enough to sufficiently interpret visual fields in real-time. For these reasons, researchers have turned to mechanical systems for practical monitoring of hand motion [Sowizral 1995]. There is reason to believe that when the problems of camera-based systems are overcome, there will be a return to this method of capturing hand motions. It provides the user with the convenience of not wearing devices or special clothing, or otherwise being distracted by the monitoring equipment.

Clothing-Free Systems Two camera-based "clothing-free" systems have survived. One has been developed by Myron Krueger and the other by Vivid Effects in Toronto, Canada. Both systems use silhouette images of the user. Neither deals with the problems of occlusion and image-merging of fingers close together.

Myron Krueger's systems are constructed to allow person interaction with computers without the need of encumbering equipment [Krueger 1990]. By using custom hardware to process the silhouette images, he overcomes some of the usual image processing speed problems. His techniques are successful at recognizing parts of the body—head, legs, arms, fingers—if they can be seen in the silhouette. In one application, participants can draw figures with their fingers. When the computer sees that the thumb and index are outstretched on both hands, it draws a curve that inscribes the region between the two hands The size and shape of the curve can be changed by moving the hands or fingers. A rapid pull away from the curve fixes it in place on the screen.

One of Krueger's goals is to develop an entire computer-based workspace that requires a minimum of mechanical devices, instead relying on vision techniques to interpret the user's hand and body motions. He sees the main limitations of his system as spatial and

temporal video resolution, and separation of foreground from background in cluttered environments.

Mechanical devices

Master-slave controllers Master-slave controllers that connect hand and manipulator motion through mechanical, hydraulic, and/or pneumatic linkages have been used for decades for handling hazardous materials [Minsky 1980] [Sheridan 1989]. These first manipulators afforded rudimentary dexterity, but served their function. As technology advanced, more sophisticated tasks were considered for teleoperation and dexterity requirements increased.

Developments of a dextrous robot hand [Mason and Salisbury, Jr. 1985] [Jacobsen et al. 1986] have attempted to raise the potential dexterity in telemanipulation to that of the human hand. These robotic dextrous hands are kinematically similar to human hands, and attempts have been made to control them with whole-hand masters [Hong and Tan 1989] [Pao and Speeter 1989] [Speeter 1989]. Others have concentrated on improving control through the use of kinesthetic feedback from the robot [Bejczy and Salisbury, Jr. 1983]. Providing this feedback is a difficult problem [Chin and Sheridan 1989] [Durlach 1989] and require large devices to accurately reflect the forces felt at the manipulator. A disadvantage is that these devices are too bulky to be used in many general applications.

In related work, [Kilpatrick 1976] used the master side of a large force-reflecting master-slave manipulator arm to demonstrate the use of force-feedback in computer-aided task interaction. Later, [Ouh-young 1990] used the arm to successfully assist biochemists in analyzing dockings for drug molecules. Both researchers found force to be a useful feedback tool.

Various companies and research laboratories developed smaller force and tactile feedback devices. Some are mounted on a small base appropriate for desktop use and provide force-feedback to the position of the hand and fingers within a limited space (e.g., Iwata, 1990). Others incorporate force producing elements into gloves, providing tactile sensations while still allowing the arm free range of motion (e.g., Stone, 1991). Many of the

small force-feedback devices are proprietary developments slated for commercial release and detailed information has not been published.

"EZglove" Here at the McGill VR Lab, an inexpensive light-weight glove has been developed to monitor the hand movements [Slavkoff 1997]. Based on an idea from Rich Sayre of the University of Chicago, flexible tubes (not fiber optics) are used with a light source at one end and a photocell at the other. Tubes are mounted along each of the fingers of the glove. As each tube is bent, the amount of light hitting its photocell decreases evenly. Voltage from each photocell could then be correlated with finger flexion.

Digital Data Entry Glove In 1983, Gary Grimes of Bell Telephone Laboratories received a patent for a glove interface for the entering of ASCII data [Grimes 1983]. The patent covers the use of a special electronic glove whose sole purpose is to interpret a manual alphabet for digital data entry—a keyboard replacement. The glove itself is made of cloth on which is sewn numerous touch, bend, and inertial sensors, specifically positioned so as to recognize the Single Hand Manual Alphabet for the American Deaf. The circuitry of the glove is designed so that unique combinations of sensor readings cause the output of 80 of the 96 printable ASCII characters (a superset of the Single Hand Manual Alphabet for the American Deaf).

VPL DataGlove^{TM} [Zimmerman et al. 1987] developed a glove that monitored ten finger joints and the six degrees of freedom of the position and orientation of the hand. The DataGlove (as it was called) was an improvement over existing camera-based hand-monitoring techniques because it operated faster and did not rely on line-of-sight observation. It was better than previous master-slave manipulators because it was light-weight, comfortable to wear, unobtrusive to the user, and general purpose.

Commercialization of the DataGlove by VPL Research, Inc. at a reasonable cost to research institutions has lead to its widespread use around the world.

The DataGlove consists of a lightweight lycra glove fitted with specially treated optical fibers along backs of the fingers. Finger flexion bends the fibers, attenuating the light

they transmit. The signal strength for each of the fibers is sent to a processor which determines joint angles based on precalibration for each user. Most Data Gloves have ten flex sensors, one for each of the lower two knuckles of the digits, but some have been made with abduction sensors that measure the angle between adjacent fingers. Position and orientation of the palm is determined by a Polhemus sensor attached to the back of the hand, registering distance and orientation to a companion transmitter fixed in place nearby. The three-space sensor, made by Polhemus, uses low-frequency pulsed magnetic fields to sense the six degrees of freedom (three-space position and orientation) of a small sensor relative to a source transmitter [Raab et al. 1979]. The finger-flex accuracy is rated at 1° joint rotation but formal testing and personal observations have shown the actual flex accuracy to be closer to 5° [Wise et al. 1990]. The DataGlove can collect finger data at approximately 60 samples per second.

Most of the DataGlove research has used the hand as a replacement of the more conventional input devices but adding little or no new functionality. This is not to say that the DataGlove has no advantages over conventional input devices. It can provide a much more natural interface than a mouse or joystick. However, when viewed in terms of functionality, few have used it as more than a glorified three-dimensional mouse.

[Takahashi and Kishino 1990] of the ATR Research Labs in Kyoto, Japan developed a coding scheme to allow computer recognition of the Japanese KANA manual alphabet. They used the DataGlove to capture hand posture and recognized signs through a combination of principal component analysis to determine the contributions of each finger joint to the differences between signs, and cluster analysis to group hand configurations. Because of the difficulty of accurately measuring the lower thumb joint with the DataGlove and because some of the signs have similar finger positions they were able to discriminate only 30 of the 46 kana signs.

Neural Nets Slightly more complicated is the work by [Fels 1990] using a DataGlove to interpret hand motion to drive a speech synthesizer. His particular approach used a three-stage back-propagation neural network trained to recognize gestural "words." He divided hand motions between finger positions and hand motion. Finger positions

defined the root word while hand motions modified the meaning and provided expression. No finger motions were monitored, and hand motions consisted only of variable speeds of back and forth motion in the six three-space cardinal directions. His "language" was based loosely on conventional gestural languages and his study had more to do with using neural nets to interpret a lexicon of hand signs than with the process of communicating with gestures. Nevertheless, Fels reported a 92% success rate on the recognition of 203 signs based on 66 hand shapes combined with 6 gestures.

A drawback of using neural nets is that they require extensive training that must repeated from the start each time a new hand motion is introduced. Thus, this technique would be best used with fairly established lexicons.

In his report, Fels included an interesting analysis of hand-to-language mapping at various levels of granularity, from using hand motions for the control of parameters of an artificial vocal tract, to interpreting whole hand motions as words and concepts. The trade-offs, as Fels put it, are between extent of vocabulary—unlimited at the most granular level—versus ease of learning and speed of communication—highest at the word and concept level.

Although Fels demonstrated the viability of connectionist techniques for interpreting finger position and hand motion, it is uncertain if his techniques will hold up under the added complexity of finger motions. This will be necessary to interpret the full expression of signed languages. However, as a control structure for computer input, Fels's methods may be adequate.

[Brooks 1989] also used a neural net to interpret DataGlove motion; in this case for robot control. Unlike Fels, Brooks incorporated dynamic gestures into the control language. He used Kohonen nets to recognize paths traced by finger motion in the n-dimensional space of the degrees of freedom of the digits. Since he had no Polhemus or other three-space tracking method, Brooks ignored three-space hand motion. Each Kohonen net (typically small—on the order of 20 cells) was trained to recognize a single gesture. By operating several concurrently on the DataGlove input, several gestures could be recognized. He achieved moderate success at simple gesture recognition, such as closing all the fingers, leading with the index; opening the thumb and first two fin-

gers simultaneously; and moving from a neutral hand posture to a "pen" grasp posture. However, in his conclusion, Brooks stated that he has yet to show that his methods are sufficient for practical dynamic gesture recognition or that the DataGlove is an appropriate interface for robot control.

The three methods of hand shape and motion recognition described above (and the method used by Kramer, below) are conceptually similar. Basically, they analyze the hand-space-degrees-of-freedom vector for each posture or gesture, and match it to a landmark hand-space vector representing the target posture or gesture. The match must occur within error tolerances (usually Euclidean distance) weighted by the significance of each degree of freedom. In the Takahashi-Kishino method, the principal component analysis determines the weighting of the degrees of freedom. In Fels's neural nets this process is hidden in the coefficients for each node. Brooks's Kohonen net has few nodes, each with an n-space vector of coefficients. These coefficients contain the weightings, with the interaction between the nodes of the net determining the identity of a dynamic gesture. (Kramer's implementation, described below, uses a method similar to the one used by Takahashi and Kishino.)

Exos Dextrous HandMaster^{TM} In 1987 Arthur D. Little, Inc. (ADL) in conjunction with Sarcos, Inc. developed a master controller for the Utah/MIT Dextrous Hand, a four-digit robot-hand [Marcus and Curchill 1988]. The controller was an exoskeleton-like device worn on the fingers and hand. Using Hall-effect sensors as potentiometers at the joints, it accurately measured the flexion of the three joints of each finger as well as the adduction of each finger and the complex motion of the thumb. Since the Utah/MIT Dextrous Hand has only four digits, the exoskeleton had no pinkie. After shipping several of the Dextrous HandMasters, Dr. Beth Marcus, the leader of the project at ADL, licensed the technology and formed her own company, Exos, Inc. After redesigning some of the mechanics and all of the electronics, Exos brought to market a five digit exoskeleton—the Dextrous HandMaster, Series 2 (DHM).

The DHM measures 20 degrees of freedom—four for each finger, and four for the thumb. Based on initial experience, the accuracy of the device is well within 1° of flexion.

A formal study found similar results with a 92 to 98 percent correlation between finger position and DHM readout, depending on the joint [Makower, Parnianpour and Nordin 1990]. The DHM does not measure palm position or orientation, but a three-space sensor can be attached for that purpose.

The DHM was used for clinical analysis of hand impairment as well as for experimental purposes in several research institutions. [Speeter 1989] extended his work with the Utah/MIT Dextrous Hand and DataGlove to the DHM. Since the DHM is kinematically similar to the UMDH, the transformation matrix scheme used for the DataGlove is not necessary. Instead, Speeter transforms the raw sensor data into strings of 7-bit characters. Lexical recognition routines match string patterns to autonomous manipulation functions for the UMDH (similar to the poses used for the DataGlove).

The DHM has also been used by Tod Machover at the MIT Media Lab for controlling acoustic parameters in live musical performance [Machover 1990].

Power Glove^{TM} Inspired by the success of the VPL DataGlove, the Mattel toy company manufactured in 1989 a low-cost glove for use as a controller for Nintendo games. The Power Glove, as it is called, uses flexible molded plastic on the back of the hand and fingers and lycra on the palmer side. Embedded in the plastic on the fingers are resistive-ink bend sensors that register overall flex of the thumb, index, middle, and ring fingers with two bits of precision each. Mounted on the back of the hand are sonar range finders (similar to those used in automatically focusing cameras) to locate the Glove in space accurately to 1/4-inch. The range finders also provide four bits of roll orientation for the hand (rotation of the wrist).

Although the least accurate of the whole-hand input devices, the Power Glove is also the cheapest by a factor of 100. It works with several pre-Glove Nintendo games, such as *Mike Tyson's Punch-Out* where punching motions control the swing of an on-screen boxer. Some games have been especially designed for the Power Glove. *Glove Ball* is one that allows the player to "hit" or "grab and throw" a ball against tiles in a handball-like court imaged on the screen. These games are fun to play and make good use of the whole-hand interface. In addition, many researchers are experimenting with the Power

Glove as a low cost alternative to the DataGlove for initial research into whole-hand input. Although a general purpose computer interface is not publicly available for the Power Glove, people have reverse engineered the electronics necessary for connecting the Power Glove to a computer's serial port [Eglowstein 1990]. At the McGill VR Lab, an HC11 microprocessor is used to interface the PowerGlove with the serial port of the SGI machines running the VR simulations [Shaikh 1998].

Virtex CyberGlove^{TM} James Kramer has developed a glove-based system at Stanford University to translate ASL into spoken English [Kramer and Leifer 1989]. A custom-made cloth glove has sewn into the fabric strain gauges to sense 16 degrees of freedom of finger and wrist flexion. Pattern recognition software maps the finger position into a "hand-state vector." When the instantaneous hand-state lies close enough to a recognizable state, the corresponding ASL letter or symbol is put on an output buffer. When a phrase is complete, a special sign causes the result to be spoken by a voice synthesizer. Hearing-able participants in conversations type answers back on a hand-held keyboard. The first implementation of the system only interpreted finger spelling, where each hand sign is a letter in the English alphabet. Further work resulted in other sign-language gestures to be recognized. Kramer marketed the glove as the *CyberGlove* along with a CAD virtual environment through Virtex.

Space Glove^{TM} W Industries is a British company marketing a virtual reality arcade game. In 1991 they released a glove dubbed the Space Glove^{TM} for use with their Virtuality^{TM} system. The glove is made of soft molded plastic that fits over the back of the hand. The fingers are placed through rings that sit between the fingers joints. The four fingers flexes are measured, as well as two flex angles of the thumb, all using sensors with 12-bit analog-to-digital converters. A three-space magnetic tracker is incorporated into the back of the glove. The glove responsiveness to fingers flexion and hand movement is quite good when used for a short period of time, but the glove is somewhat uncomfortable as the plastic rings around the fingers have little stretch and constrict the fingers. The stiffness of the rings also makes it hard to put on and to take off the glove.

1.2.2 Hand Manipulation

The developers of the VPL DataGlove have been primarily interested in simulated environments or *virtual realities*, and have used the hand as the user's manipulative extension into those environments [Kelly, Heilbrun and Stacks 1989]. Users wearing the DataGlove in the VPL system saw a graphic hand which followed the motions of their hand in the simulated environment. By pantomiming reaches and grabs, the user caused the graphic hand to reach and grab objects in the simulated environment. The viewpoint could be moved by pointing in the desired direction and "flying" to the destination.

The implementations of the grab and flight behaviors were based on software that triggers events on recognized finger postures. Thus the entire hand interface can be reduced to a set of abstracted input devices. The hand location is a *locator*, grabbing is achieved through posture recognition—a *button*, and motion through the environment by pointing your finger in the direction of travel is a *locator* and *button* combination. Functionally, the DataGlove could be substituted with a *bat* in VPL's application. A *bat* is a six-degree-of-freedom locator with one or more buttons [Ware and Jessome 1988]. The buttons are functionally equivalent to the posture recognition of the DataGlove software. It is perhaps a less natural interface to the environment, but it is functionally equivalent.

The Aerospace Human Factors Research Division of the NASA Ames Research Center studied the VPL DataGlove in its initial stages of development and used it for interaction with their Virtual Environment Display System [Fisher et al. 1986] [Fisher 1989]. Like VPL, they used the DataGlove as a tool for grasping and moving objects, indicating direction of motion, picking from menus, and invoking system commands (by postures). They also have used the location of the hand as a trigger for various events such as drum beats in a virtual drum machine. Like VPL, their use of the DataGlove provides functionality equivalent to a bat, but profits from the naturalness of using the hand.

In much of the literature, the DataGlove is used similarly to its application at VPL and NASA. The hand's graphic image is displayed in an interactive computer environment and used as a tool for "point, reach, and grab" interaction. At the MIT Media Lab,

the DataGlove was used as a master for a graphical hand in a virtual environment. The user could grab, move, and throw objects with the graphical hand, as well as use finger postures and motions to select from on-screen menus [Zeltzer, Pieper and Sturman 1989] [Kaufman and Yagel 1989] used the DataGlove similarly in a modeling environment. The user could grab and manipulate objects on the computer screen. [Feiner and Beshers 1990], and [Takemura, Tomono and Kobayashi 1988] also used the DataGlove to allow users to "touch," grab, and manipulate on-screen objects and recognized finger postures as event triggers (buttons).

The advantage of this model of interaction is naturalness—users' actions are closely correlated with those that might be performed on physical objects. However, in each of these applications, the DataGlove functioned little more than a bat. In fact, in the MIT implementation, the function of the DataGlove could be substituted by a Spaceball^{TM}—a six-degree-of-freedom force input device with eight buttons. The interface to the Spaceball was similar to the interface to the DataGlove with button events substituting for posture recognition.

Although the Polhemus is a position-control device, while the Spaceball is a rate-control device (and thus affects the input task differently), the functionality of the two was the same, i.e., manipulating objects. The Spaceball does not allow the same level of coordinated three-space motion as the Polhemus (mounted on the DataGlove), but does perform better for tasks requiring precision location or steady motion. This is partly due to the difference between rate-control and position-control, and partly due to the inherent jitter of freehand motion and the susceptibility of the Polhemus to electromagnetic disturbances.

AT&T Bell Laboratories, [Weimer and Ganapathy 1989] used a DataGlove in the same way as the systems described above, except they implemented two thumb-based gesture controls called *clutch* and *throttle*. Clutching was used for incremental transforms, such as rotation. When the thumb was brought towards the index finger, the screen object followed the rotation of the hand. When the thumb was pulled back, the screen object did not rotate. With this clutch mechanism, object manipulations could be ratcheted, avoiding uncomfortable contortions of the hand and arm. Throttling was a variation of

the clutch mechanism in which the abduction angle of the thumb was used to scale the effect of a hand motion. Their scheme can be described in terms of virtual devices by calling the clutch a button based on thumb posture, and the throttle a valuator based on the angle of the thumb.

Two research projects have used the DataGlove to control a Utah/MIT Dextrous Hand robot manipulator (UMDH). At AT&T, [Pao and Speeter 1989] constructed algebraic transformation matrices to map human hand poses to robot hand poses. The transformation matrix was necessary to overcome the kinematic differences between the hand (as transduced by the DataGlove) and the UMDH. The user manipulated the UMDH by mimicking the desired poses. At NYU's Courant Institute, [Hong and Tan 1989] resolved the kinematic differences between the human master hand and the robotic slave hand by determining the position of the fingertips of the user's hand and then driving the robot hand fingertip positions to match.

1.3 Thesis outline

After presenting a survey of current VR applications and an overview of VR technology, the thesis describes the overall system architecture of the Welding Intelligent Tutoring System (WITS) in chapter two. It then elaborates in chapter three each of WITS' three sub-components: the user interface and peripherals, the expert system, and the 3D virtual environment. Chapter four deals with the implementation details of the system along with the evaluation of test results obtained and some future recommendations. Finally, chapter five concludes the thesis with a summary of the research achievements.

Chapter 2

Overview of WITS

Low grounding quality found at certain Hydro-Québec sites in the past signaled a need to review the process used in training workers involved in the deployment or repair of grounding fixtures, particularly in the field of alumino-thermal welding, a welding technique dependent on the chemical reaction between powdered aluminum and copper oxide. Since then, the training procedure has consisted of a welding course given by the manufacturer ERICO, followed by evaluation of the trainee by inspectors according to the *Installation and Inspection Guide* [Laliberté et al. 95]. However, the need for practical, hands-on training not constrained by instructor or resource availability, nor by time on the part of all involved, has prompted this research project for a virtual reality (VR) form of instruction on alumino-thermal welding.

The VR system described in this thesis was designed as a prototype for this purpose of training personnel in the field of alumino-thermal welding of grounding fixtures used in electrical station equipment. The VR training system, which got to be known informally under the name WITS, the *Welding Intelligent Tutoring System*, contains the fundamental knowledge required to effectively train the user and to provide detailed, step-by-step instruction on how to successfully perform a variety of alumino-thermal welds, as well as producing an evaluation of the trainee's performance.

2.1 System Functionality

The trainee using the WITS system should be able to perform a complete and realistic welding job that follows the procedure described in the course manual. WITS should be able to assess and improve upon the user's acquired knowledge and skills. In a typical welding scenario, the human operator faces different situations depending on different factors such as weather conditions, the states of the wires and their types. Oversimplified, the scenario involves the following:

- digging a trench at the required site;
- cleaning the site wires with the proper tools;
- choosing the proper mold and ensuring that it is the right size and that it is both clean and dry;
- positioning the mold and adding the correct amount of powder for the chemical reaction;
- igniting the powder;
- removing the mold when it has cooled enough.

The operator has to choose a correct mold that satisfies the sizes of the wires involved in the welding as well as the connection type related to the wires' positions and orientations as can be seen in Table 2.1. Currently, there are 48 mold types provided by ERICO.

The operator will also need to determine the states of the wires which include cable corrosion, tension and moisture. Based on that, he should perform the necessary cleaning, loosening and drying. The welding reaction then involves choosing the correct aluminum powder cartridges to be used. This choice is directly related to the amount of cleaning performed. Table 2.1 provides two values for the "Aluminum powder cartridge" to be used for each mold type described in the table. The first determines the powder cartridge to be used when the wires are lightly cleaned while the second is for situations where the wires had to be trimmed in a V shape.

Mold ID	Description	Aluminum powder cartridge
326a	T connection between a 4/0awg and a 2/0awg wires	115 (65 65)
326b	T connection between a 350mcm and a 2/0awg wires	150 250
326c	T connection between a 500mcm and a 2/0awg wires	150 (200 150)
326d	T connection between a 4/0awg and a 4/0awg wires	150 (90 90)
327a	X connection between a 4/0awg and a 2/0awg wires	200 (150 150)
327b	X connection between a 350mcm and a 2/0awg wires	500 (500 65)
327c	X connection between a 500mcm and a 2/0awg wires	(250 250 250) (250 250 250)
327d	X connection between a 2/0awg and a 2/0awg wires	200 (200 65)
330a	V connection between a 4/0awg wire and a surface	200 200
330b	V connection between a 350mcm wire and a surface	(150 150) (150 150)
330c	V connection between a 500mcm wire and a surface	(200 200) (200 200)
330d	V connection between a 2/0awg wire and a surface	150 150
333a	S connection between a 4/0awg and a 4/0awg wires	90 90
333b	S connection between a 350mcm and a 350mcm wires	150 150
333c	S connection between a 500mcm and a 500mcm wires	200 200
333d	S connection between a 2/0awg and a 2/0awg wires	65 65

Table 2.1: Some mold IDs and their specifications as they appear in the *Cours De Soudage* of Hydro-Québec

Certain procedures are also crucial to the safety of personnel as well as equipment, and may be dictated by prevailing weather conditions. The operator will need to take the necessary measures to counter the effect of the different weather situations that can interfere with the welding job such as wind, rain and snow. The operator should build a shelter when faced with heavy rain or snow. While he could use an umbrella when faced with light rain or snow. He should always make sure to stand with his back to the wind to protect himself from the fumes.

To allow the trainee to do all of the above, the system must present him with a realistic world that contains all the objects usually available to him in the real world and that should interact in a natural way to permit the different choices present in real life such as opening an umbrella or cleaning a wire with a brush. This is achieved in WITS by introducing a 3D representation of the human operator, welding tools and the physical environment.

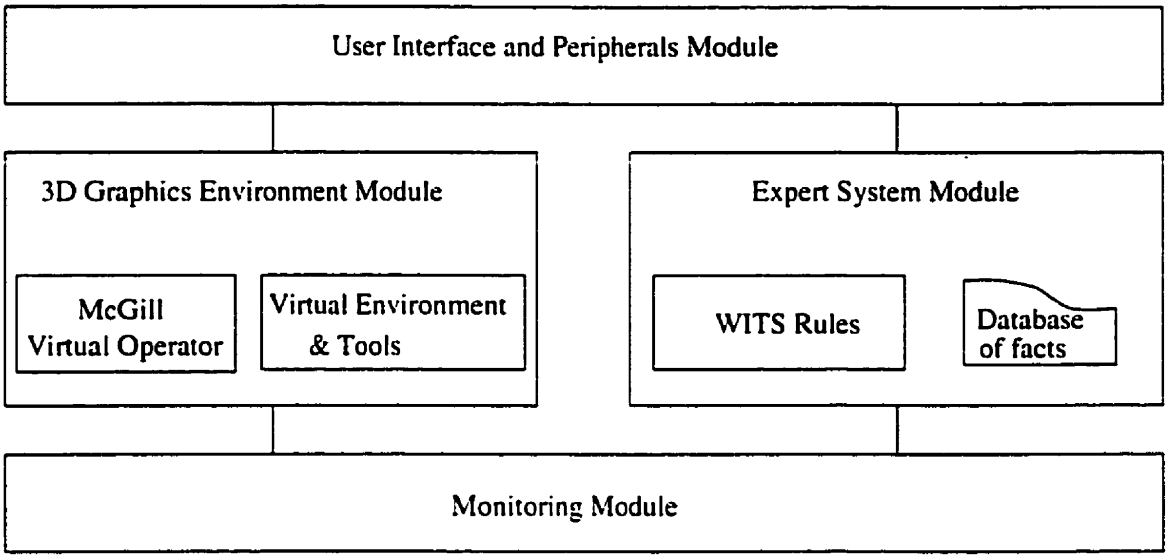


Figure 2.1: System Architecture

2.2 System Architecture

WITS is constructed through the integration of an expert system that holds all of the knowledge expertise and logic pertaining to the task to be performed and the 3D user interface that presents a realistic 3D virtual environment with which the trainee can and should interact in order to obtain the knowledge stored in the expert system.

Figure 2.1 shows the WITS over-all architecture with its four modules: the user interface and peripherals module, the expert system module, the 3D graphical environment, and the monitoring module. The expert system module interacts with the 3D graphical environment module through the monitoring module. In addition they interact with the user through the user interface and peripherals module.

Each of the system modules will be introduced briefly in the remainder of this chapter. The complete design and implementation of each will subsequently be elaborated in chapter three, except for the monitoring module which is discussed in a different study [Kaddoura 1998].

2.3 User interface and peripherals

Since there are presently no standards for hardware and software when applying VR technology to training, there are nearly as many different configurations as there are VR systems operating. The peripherals involved in the WITS training system study include the following commercially available devices: Logitech Red Baron mouse and tracker, CrystalEyes (LCD) shutter glasses with head tracking sensors, Verbex voice recognition system, Mattel PowerGlove. In addition, we have included two "in house" built devices namely pedals for navigating in the VE and gloves for measuring the flex of the hand fingers. Of course we also use the standard display monitor, keyboard and 2D mouse devices.

Most of these devices have built-in drivers supporting them in the WorldToolKit and the CLIPS software, except for the Verbex voice system, the Mattel PowerGlove, the "in house" pedals and gloves for which special drivers had to be built. All of these devices along with their drivers can be interchanged with other similar ones.

Interaction methods are defined to relate the usage of each device with the desired tasks affecting the objects in the virtual world. The basic interaction tasks are object positioning, object rotating, object selection, and quantify. The quantification task determines how the user adjusts some quantifiable aspect of the virtual world such wire tension or tool pressure. Combinations of these four tasks permit the user to address the two more general interaction tasks of navigation and manipulation in the VR world.

2.4 Expert System

The expert system is implemented using CLIPS, an expert system tool developed by the Software Technology Branch of NASA. CLIPS provides object-oriented mechanisms within a subset of the CLIPS language, known as COOL: CLIPS' Object-Oriented Language. The expert system is created using both the rule mechanisms of CLIPS for managing the heuristics of welding, and the object mechanisms of COOL for modeling the world of welding with objects that can be directly rendered into graphical objects by VR application-building software.

All knowledge of the expert system pertaining to the training steps derived from the *Cours de Soudage* is stored in a separate database file. This enables easy modification and updating of the training course via a text editor or spreadsheet.

The basic object classes of the expert system are STEP, TOOL and PERSON. The behavior of these objects can be summarized as follows:

- The STEP object stores the individual, possibly order-dependent steps required to perform a proper weld within any given scenario, all derived from the *Cours de Soudage*. STEP objects are related by step prerequisites that are also derivable from the training procedure, and the relationship amongst these steps can be viewed in the form of a branching tree or, more specifically, a graph. Note that the derived information is all stored in a separate ASCII file according to a predefined format which can be easily updated as a result of any modifications to the *Cours* by the client or another user via a text editor or spreadsheet.

- Within each welding step typically is specified a set of appropriate actions, and a required set of TOOL objects to be used to successfully execute the step.

- Each PERSON object stores all interaction and performance data of each individual user, so as to permit multi-user enhancement of WITS in a future implementation.

2.5 3D Graphical Environment

The Virtual Environment is implemented using WorldToolKit (WTK), a software development system, (available on multiple computer operating system platforms), for building high-performance, real-time, integrated 3D applications for scientific and commercial use. WTK is essentially a collection of C libraries that we used in conjunction with object oriented C++ code developed to model the different interaction methods used to interface the input devices with the 3D world and between the 3D world objects themselves.

The object classes Object2Manipulate and WTBody are the basic components of the Virtual Environment. The behavior of these objects is summarized as follows:

- the TOOL objects in the expert system are mapped in the VR world under the Object2Manipulate objects. In the VR world the tools have the distinctive feature, compared to static objects, of being manipulated by other 3D objects and thus of being dynamic.

- Each PERSON object in the expert system has to be represented in the VR world. The WTBody object makes this possible. In the VR world there is more than the keyboard interface, the WTBody object has the responsibility of interfacing all the input devices with the corresponding body objects in the VR world. The most important of these body objects are the WTHand objects. The WTHand objects permit direct object manipulation.

The monitoring systems were developed using C and C++ in conjunction with WorldToolKit's libraries. They are the agents that permit the different user's actions to take place in the 3D world and to be validated with the knowledge based expert system.

The object classes ActiveAgent, ManipulationAgent, PhysicsAgent can be considered as the basic agents of the WITS system. The behavior of these objects is summarized as follows:

- The ActiveAgent object keeps track of the user actions in the VR world which are being compared with those that are required by the expert sys-

tem. It can thus be directly related to the STEP object in the knowledge based expert system.

- The ManipulationAgent object keeps track of and executes all of the virtual operator's manipulation actions on any of the Objects2Manipulate objects.
- The PhysicsAgent object adds a more realistic feeling to the virtual environment by applying the principles of the real world physics laws such as gravity onto the virtual world objects.

Chapter 3

WITS Design

3.1 User interface and peripherals

The User Interface is the part of any complex system that allows a user to control and monitor the system. For software programs, it can consist of graphs, menus and prompts. For a product such as a radio, it can be scales, dials and knobs. For a VR system the human interface includes both software generated graphics as well as hardware devices used mostly to collect external data but sometimes also to deliver feedback to the user.

The human interface is defined as the layer through which the user controls the VR system. The user must be able to affect the virtual environment and get the right feedback from the interface. The quality of the interface often determines whether users enjoy or despise a system. In putting together the building blocks into a complete user-interface design, the emphasis must be on a top-down design approach where first the design objectives are identified and then, the design is developed through a stepwise refinement process.

The basic elements of the user interface are the 1) input/output devices, 2) the interaction techniques that determine how to use the devices to enter or receive information, and finally 3) the interaction tasks that determine the types of information transmitted.

With this in mind, first the input/output devices that are currently used in our laboratory will be introduced along with their appropriate interaction techniques. Then

the principal interaction tasks will be discussed.

3.1.1 Output devices

The purpose of the output devices is to immerse the user into the virtual simulation so as to enjoy the virtual experience and to develop a realistic expectation of the real world scenario. Output devices can be classified according to the senses they stimulate: visual, hearing, and touch. Presently, there are no devices that stimulate the smell or the taste sensory systems.

Visual devices are the most important feedback tool in virtual simulation systems. This is not a surprise since it is known that the largest specialized area of the brain is the visual cortex, dedicated to processing visual stimuli.

The current implementation uses a 21" display monitor to view the virtual simulation. The CrystalEyes LCD, *Liquid Crystal Display*, shutter glasses are combined with the display monitor to provide a stereoscopic experience. Basically, these glasses are synchronized with the screen to display slightly offset images to each eye. This has the effect of fooling the user's brain into seeing 3D scenes rather than a flat display [Hodges 1992].

Audio feedback can enhance the virtual experience by taking the simulation one step closer to the real world where background noises are present. It is also used to provide direct feedback about actions taken in the virtual world such as touching were a beep sound is generated upon collision between the user hand and virtual objects. This helps the user in determining when to close his hand to grab the touched object. It can also be considered as an extra depth cue especially when the simulation is run without the stereo-scopic vision.

Force feedback is not being used in the current implementation. We will see later in chapter four how this has influenced the implementation of some manipulation tasks such as the brushing task. In such actions, the lack of force feedback results in a miss-coordination between the real hand movement and its virtual counterpart, i.e. the virtual hand does not follow the user's and the immersion is lost.

3.1.2 Input devices

It is commonly known that the perfect interaction device for 2D interfaces is the mouse. Such a device does not yet exist for 3D interfaces, most VR systems developed up to now use different input devices that are most suitable for their specific project requirements along with budget and availability constraints. Similarly we have chosen to use input devices that are available to us at low cost. These devices will now be discussed under more general device categories.

Hand devices

It is important to note that most devices used for positioning are relative devices. Relative devices have no absolute origin and report only changes from their former position. A relative device can be used to specify an arbitrarily large change in position. Another advantage of relative devices is that the application program can reposition the reference point of the device anywhere in the VR world.

Some devices that are built for 2D interaction are extended to support 3D actions. The most important of these is the 2D mouse device which is supported by all interfaces today. The 2D positioning system of the mouse combined with the two or three buttons that it incorporates can permit six degrees of freedom interaction. Joysticks can be used in a similar way.

The well-known QWERTY keyboard is used for inputting text strings as well as for 2D navigation and to support function keys. Although the Dvorak keyboard is somewhat faster due to its better key distribution, the QWERTY is more widely used, most users know its letter positioning by heart and do not need to look at the keyboard to type commands. VR applications do not rely much on text strings, however, the keyboard can still be used for shortcut function keys.

The 3D devices used in this project are all sonar devices that can track the 3D position of the device as well as its orientation. These devices are the Logitech Red Baron mouse and simple-tracker as well as the Mattel PowerGlove. The Red Baron mouse is by far the most accurate and easy to use.

In running the ESOPE-VR demo, we noticed that 3D devices are not very helpful for "walking" the human operator around. These devices actually have to be constrained to generate horizontal movement since humans are not used to "flying" around. These 3D devices are more useful for tracking hand position. The human hands are the body parts that naturally extend in all directions and that make full use of the 6 DOF. It is true that the hand has constraints in movement, however, a device that is connected to the hand will automatically inherit these constraints. Any further refinement must be provided at the interaction task level and will hence be discussed later under that subject.

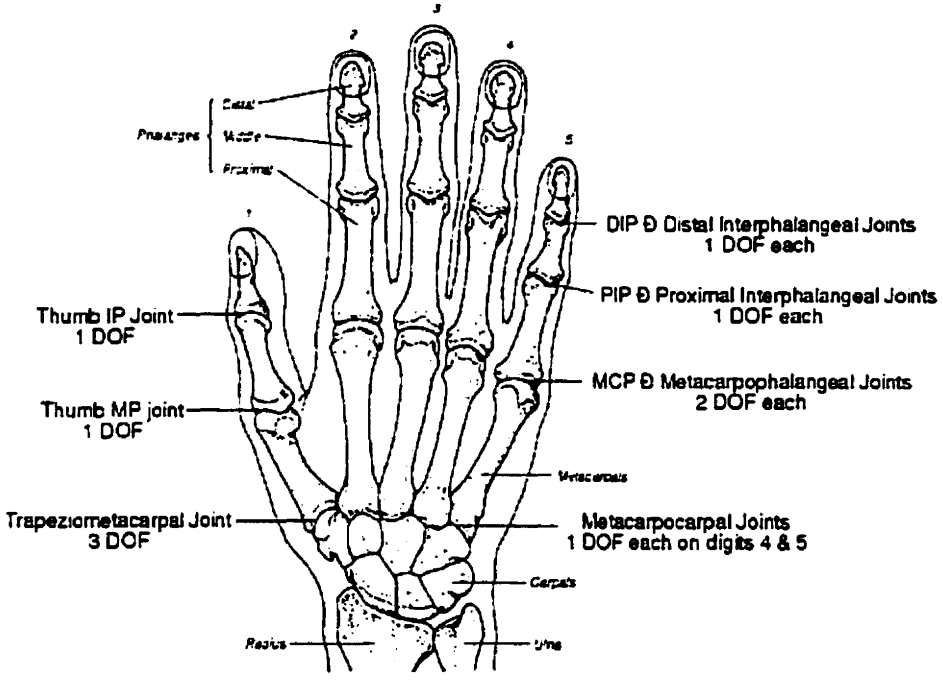


Figure 3.1: The hand joints *(Diagram adapted from Sturman, 1992)*

The human's hands are his basic tool for manipulation in the real world. They permit 29 degrees of freedom per hand: 6 degrees of freedom in the free motion of the palm, and 23 degrees of freedom in the movement of the joints as shown in figure 3.1. Restraining the hand to manipulate a 2D mouse, a joystick or even a 6D mouse is a waste of resources. On the other hand, trying to use the full potential of these 26 Degrees of freedom is impossible and would lead to an uncomfortable cumbersome interface. That is why we decided to use devices that simply track finger movements as well as hand position.

The PowerGlove had been used in the ESOPE-VR project. The most attractive thing about it is the price. When the ESOPE-VR interface with the PowerGlove was tried, its lack of accuracy and comfort was felt immediately. The PowerGlove incorporates the sonar sensor, the finger bend-resistor sensors, and a collection of pushbuttons. The sonar and bend-resistor sensors generate very noisy signals. The buttons seem to be advantageous for augmented functionality, but they also tie-up the second hand and thus make the interaction cumbersome. The Red Baron mouse can really do more, better, with only one hand, but is expensive.

Since the Baron mouse is much more accurate than the PowerGlove, we decided to augment it with a glove that can monitor the fingers movement. The data glove was built in-house and was called *the EZglove*. It uses light-tube flex sensors to measure finger rotation about the palm. The sonar 6D tracker is attached on top of the palm that wears the glove. This combination allows the user to convey both the position of his hand and its posture with good accuracy.

The EZglove is intended to be used for hand manipulations in the VR world. The experience from the ESOPE-VR project has shown that combining both manipulation and navigation tasks into the glove device decreases immersion and increases confusion. Also the use of the hand position for navigation purpose is not comfortable at all. Again, pushbutton switching from one mode to another ties up the second hand.

Head device

The Logitech head-tracker is used to monitor head movement. Two configurations are available. In the first configuration the transmitter can be screwed on an HMD with the receiver fixed on the ceiling. In the second configuration the transmitter is fixed on the CrystalEyes shutter glasses with the receiver placed on the desk (on top of the monitor).

The Logitech head-tracker works well with the HMD, however with the CrystalEyes shutter glasses display such as the one we are using, it was not found to be beneficial since the user has to stay focused on the display area which is easy when the head stays still but becomes harder once the head is used to control the viewpoint. Furthermore, there is a limitation on how much the head can be rotated about its vertical axis. For

this reason, the current design of WITS does not incorporate head tracking.

Foot device

Pedals have been created to behave as a modified joystick resulting in a 2D interaction device that is operated by the feet, thus freeing the hands to perform other functions. The pedals are intended to be used for navigation in the 3D world. Humans are familiar with walking around in the real world on their feet. This shows that there is room for simple devices in a VR interface, however, the choice of the devices is very important. It has been noticed that in addition to freeing the hands, the pedals are very natural to use for navigation. However, choosing the pedals to control the cursor positioning in a windows interface is not such a good idea. In a windows interface the whole desktop is visible to the user and the mouse can efficiently cover the space quickly and with good accuracy, a joystick device is better than the mouse for interfaces where the display only shows a portion of the system's world. The user associates the joystick movement with real world movement (forward, backward, turn left, and turn right). This association can not be easily made with the mouse. When using a mouse the user usually thinks of up, down, left, and right positions on the screen.

Voice device

The Verbex 7000, a speaker dependent voice recognition device, accepts short segments of continuous speech of approximately 20 words in length. This can be very useful as an alternative for keyboard commands [Okapuu-von Veh 96]. Once correctly trained the Verbex 7000 is a reliable input device that can free the user hands from the keyboard. In addition to freeing the hands, the Verbex 7000 provides a natural way of communication between the user and the computer especially for people with little or no knowledge about computers and their peripherals. In the case of the ESOPE-VR training system, the Verbex 7000 was used to permit the trainee to interact with the simulation using voice input to perform tasks that are not part of the training task such as transporting (beaming) the trainee to a specific location. Another very effective feature implemented

using speech recognition is the integration of the help function permitting online consultation on the different possible actions. A voice interface option was developed to replace the keyboard interactions with the WITS expert system.

3.1.3 Interaction Tasks

Interaction tasks are independent of the logical input devices being used. Interaction tasks are defined by what the user accomplishes. Traditional 2D interfaces have identified four basic interaction tasks: positioning, text entry, object selection, and quantification. The unit of information supplied in a positioning interaction task is of course a position. Similarly, the text entry task yields a text string; the selecting task yields an object identification; and the quantification task yields a numeric value. It is important to mention that many different interaction techniques can be used for a given interaction task. Thus, interaction tasks can be described as being device-independent. For 3D interfaces, we need to add the rotation task as well as modifying the positioning and selecting tasks so that they would take into account the difficulty of perceiving 3D depth relationships of objects relative to each other. The navigation and manipulation tasks are built-up from these basic tasks listed above.

The Navigation and Manipulation Tasks

The training system frequently requires the user to situate himself at a variety of sites in the VR world. This repositioning of the user or observer constitutes the navigation task which can be decomposed into the following two subtasks: 1) the positioning interaction task, 2) the rotation interaction task.

The user is also required to perform actions that involve manipulating other objects in the world such as the welding tools. In general, this manipulation task consists of selecting and then interacting with objects and can be decomposed into the four basic subtasks: 1) the positioning interaction task, 2) the rotation interaction task, 3) the selection interaction task, 4) the quantification interaction task.

The Positioning Interaction Task The positioning interaction task consists of moving a selected object to a desired location. This can be achieved either by assigning the object's new position (x, y, z) relatively to its previous one or by providing an absolute position coordinates. The absolute position's coordinates can, if desired, be entered as text on either a real or a simulated keyboard; however, this is definitely not desirable in a 3D VR world since it will complicate the user's life, making the interface less immersive. Absolute positioning is used in WITS by saving a few critical positions to which the user can be tele-ported (beamed). To tele-port to such specific positions, voice or the keyboard is used.

The Rotation Interaction Task The rotation interaction task consists of changing the orientation of a selected object. This can be achieved either by specifying the object's new orientation (ϕ, θ, α) using relative or absolute orientation. As in the positioning task, entering absolute orientation's angles is also too cumbersome for the user.

The Selection Interaction Task The selection task is that of choosing an element from a choice set. Typical sets are commands, attribute values, object classes, and object instances. The first three types of selection tasks have relatively limited choices and thus are of fixed-size. These can be controlled through speech, keyboard function keys, or hand gestures (posture). The object instances selection on the other hand is a varying-sized choice set since instances can be created and destroyed during the simulation and hence it is difficult to assign a separate function key to each object instance in the VR world. Two techniques are particularly well suited to such varying-sized choice sets. These are selecting objects by pointing and naming.

Selection using the virtual operator hand as the pointing element is the most natural technique. It involve positioning the hand on the object to be selected in such a way as to touch it. Once the object is touched a flag is set to indicate that the object has been selected by the user and that further interaction with the object can now take place. When the user stops touching the object the flag is reset and the object is unselected.

Selection can also be done by naming the specific object. Although this technique

may sound simple, it is hard to believe that the user will be able to remember the name of every object's instance in the VR world. However this technique can be very helpful when trying to select part of a complex object or even a simple object lying too close to other objects. In such cases the user can point at the complex scenery and then select the desired object to be selected by naming its object class or if needed its specific name. The interface can also help the user by listing the possible choices. Speech input is very helpful here since the user's hands are not free for typing.

The Quantification Interaction Task The quantify task involves selecting a numeric value from a specific range. Typical interaction techniques are typing the value, setting a dial to the value, and using an up-down counter to select the value. For a VR system the second technique is the most common since it is heavily used in the real world itself. A few examples are the radio, oscilloscope, gauges etc... As in a radio the up-down counter can be used for fine tuning. Typing is good when the user knows the exact value to be entered and where the set of possible desired values is of a fixed size.

As in real life we can use rotary or linear potentiometers as a means for entering values. The user points at the potentiometer object, grabs the current-value indicator, and drags it to rotate or slide it to the desired value. One should note that rotary potentiometers are easy to adjust while linear potentiometers current value can be easier to read.

Other Interaction issues

In this section, some other issues related to interaction techniques will be discussed.

Coordinate systems When interacting with objects in the VR world it is important to define which coordinate system is referenced. Objects can be manipulated about the world coordinate system, the viewer coordinate system, or the object's own coordinate system. This is illustrated in figure 3.2.

Most of the time the viewer coordinate is the best choice because of the fact that the user can identify with it. When choosing the coordinate system one should keep in mind

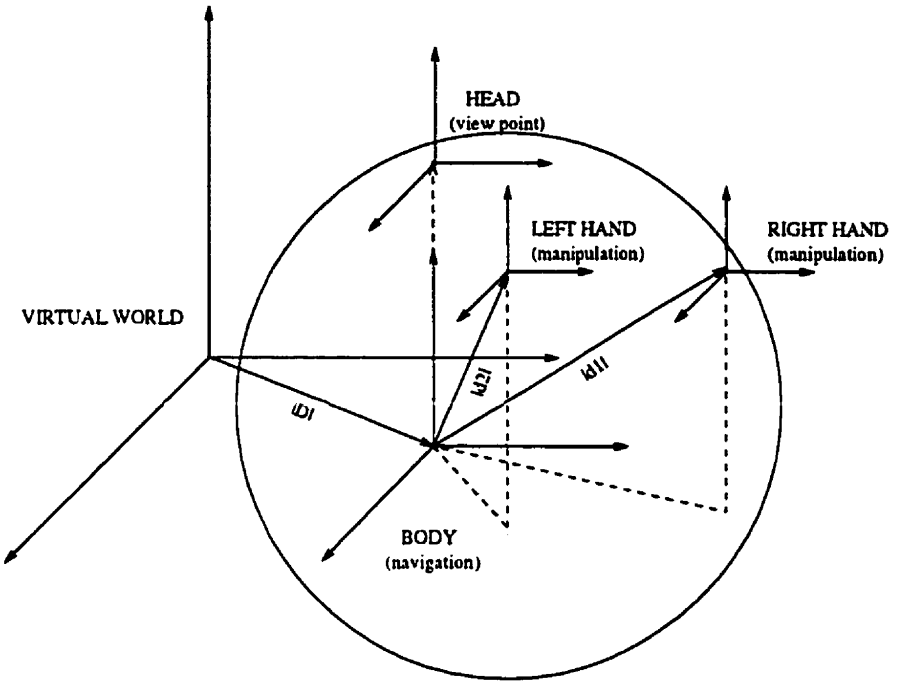


Figure 3.2: The Coordinate systems

the human-factors principle of stimulus-response compatibility, which states that system responses to user actions must be in the same direction or same orientation, and that the magnitude of the responses should be proportional to the actions.

Scale factor There exists a scale difference between the virtual objects displayed to the user and the real life objects. Thus a proper scaling factor must be applied consistently to all objects in the VR world. However, the scaling factor applied to user input data, from the different devices, must be treated separately since they affect the control-to-display ratio which is the ratio between input device movement and the displayed virtual object movement attached to the device. A large ratio is good for accurate manipulation, but makes rapid movements tedious. On the other hand, a small ratio is good for speed but not for accuracy. The scaling factor in such cases should not be constant, but should be changed adaptively as a function of control-movement speed. Rapid movements indicate that the user is making a gross hand movement, so a small ratio can be used; as the speed decreases, the control-to-display ratio is increased. Until now and similarly to the

ESOPE-VR demo, this is being done manually, but more study should be invested in a good algorithm.

Feedback There are five possible ways of conveying feedback to the human user and they correspond to the five senses: sight, hearing, smell, taste, or touch. It is not yet obvious how to convey smell and taste feedbacks. Lots of literature covers force feedback, but an effective mechanism has not yet been agreed on. Visual and auditory feedback on the other hand, have existed for many years and are the standard to use in any VR application. Both can provide spatial as well as linguistic feedback. Visual relations between objects' position and size, will convey feedback about position and orientation in the VR world. Similarly, auditory three-dimensional sound could be used to construct an appropriate sound "image" that would convey the necessary 3D cues. Linguistic feedback can also be used with both visual text and auditory speech, however, it does not associate with real life situations.

Learning time Most input devices used are indirect devices where the user moves a graphical object on the screen using a device that is not on the screen. Such devices require the user to develop eye-hand (or eye-foot, eye-head,...etc.) coordination. For 2D devices this is of minor importance since most user are already familiar with such methods of interaction. Three-dimensional devices, on the other hand, offer a bigger challenge. They are new to the user and are not yet perfected for maximum comfort. Eventually, however, these devices will become easier to learn due to their compatibility with real life interaction methods. The glove is such a device that is gaining popularity.

Direction preference It is sometimes desirable to constrain or impede the user from movement in specific directions. At its simplest form we can use a 2D device in the 3D environment without any mapping so that the user will only be able to use the device to control two axes instead of three. The use of the pedals is exactly for this purpose where the user is restrained to horizontal navigation.

3.2 Expert System

A Knowledge Based Expert System was developed to take into account all the conditions outlined in chapter two and all the welding procedure's instructions. The expert system allows two basic modes, tutorial and demonstration modes, along with help and evaluation mechanisms included for better self-instruction. The expert system is interfaced with the Virtual Environment which is developed to allow performing the welding procedure in a realistic VR environment which will be presented in section 3.3 .

The WITS combination of the expert system with the human 3D user interface was designed to fulfill all the roles of a course instructor and more, by providing practical hands-on training that is cost-effective and unconstrained by instructor or resource availability.

Although inspired directly from our collaboration with Hydro-Québec and ERICO, the VR welding course, with both training and evaluation self-contained, would enable universal instruction to groups both within and without that organization, such as:

- employees involved with site installations;
- inspectors who would like to refresh their knowledge on the subject of alumino-thermal welding;
- external contractors;
- other manufacturers and suppliers.

This chapter describes the specifications of the WITS knowledge based expert system introduced above, and details the design and its current implementation.

All information in alumino-thermal welding is taken from the *Hydro-Québec Course on Alumino-Thermal Welding, March 1995 Edition* [Laliberté et al. 95].

3.2.1 Environment

Hardware and Operating System

Although the expert system is developed on a Sun-4 UNIX workstation, the expert system can be run on any platform that runs CLIPS. The portability of the CLIPS program was shown by successfully running the expert system on an IBM-PC version of CLIPS with Microsoft Windows. By recompiling the CLIPS source program, many production platforms of the expert system are possible. This was the main reason for choosing CLIPS over other inference engines. Another reason was its availability at a low cost for educational purposes.

User Environment

The WITS expert system provides a user-friendly, text-based user environment with on-line help. User help occurs in several capacities, from command syntax and usage to instructions to actions and steps required within the training tutorial, all in a concise yet informative manner, for an audience with perhaps no prior experience in alumino-thermal welding nor in use of the expert system interface. The vocabulary used in the instruction text and also that expected of the user is simple with a straightforward grammar structure, with a flexibility in key-phrase order as well as in keyword choice. Evidently, the choice of terms and concepts used are those as described in the *Cours de Soudage* [Laliberté et al. 95]. Appropriate feedback is provided to the user, as a result of erroneous input and to acknowledge user input, as well as to prompt the user where necessary for the next action to perform.

Although it is still present, the text-based interface is invisible to the user once running under a VR platform. The voice interface takes full advantage of the syntax developed for the text-based interface by following it closely. By design, the syntax developed was made simple and clear in a manner that facilitates the translation of VR actions into such text and vice-versa.

The final version of the WITS system is intended to support a multiple number of users collaborating simultaneously, in a setup in which each user has complete use of a

fully equipped VR workstation. Even though the current version serves only one user at a time, the system was designed with the above perspective in mind and can thus undergo a straightforward upgrade to a multi-user environment. The current expert system therefore keeps track of the status and progress of individual users, including current scores and level of advancement in the training course, and, most importantly, the user's performance history.

3.2.2 System Architecture

To cater to the scenarios previously outlined in chapter two, and also ensure an adaptability to changes and updates to the welding procedures when required, a rule-based or non-procedural approach was adopted in conjunction with object-oriented methods. The rule-based method is used for propagating through the required steps of the training procedure, of which there are a myriad of possibilities and combinations and yet in some cases only a single permissible sequence of execution. Meanwhile, objects were designed to model both real-world physical objects as well as to store some abstract data concepts such as the *step*. The use of objects in this way facilitates manipulation of data and provides an easy adaptability to changes in the data to be stored by making use of encapsulation, message-handling, inheritance and polymorphism.

The design allows for multi-user operation of the system, if a layer of multi-user functions are added on a multi-user platform such as UNIX, by separately storing data and information pertaining to each user thereby tracking users and their progress individually.

User Input

Text Interface The operation of the expert system involves text entry via the keyboard in the Command Line Interface (CLI) or through a socket connection in the 3D virtual interface. The Text interface operation can be divided into two phases, executing a welding step, and interacting with the expert system itself via command keywords. The execution of a step can in turn be divided into the two stages of obtaining a tool and performing an action. Because of the diversity of actions available to be performed within

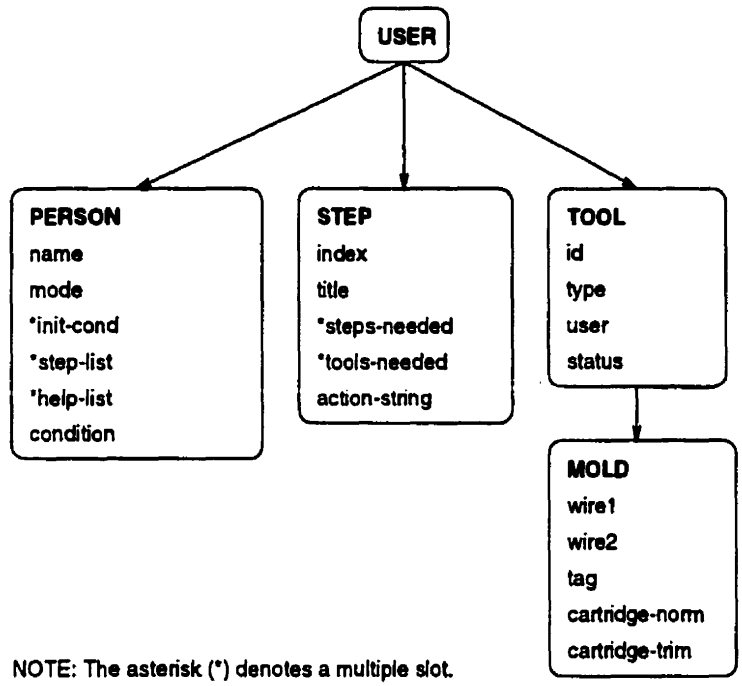


Figure 3.3: Object class hierarchy of the expert system

the expert system, a syntax language was developed to simplify user input.

Database All knowledge of the expert system pertaining to the training steps derived from the *Cours de Soudage* are stored in a separate database file. This enables easy modification and update to the training course via a text editor or spreadsheet.

Object Classes

The object class hierarchy for the present design of the expert system is given in Figure 3.3.

Class STEP As the concept of training *steps* forms such a fundamental element of the training course structure, it is no surprise that one of the COOL classes defined for the expert system is that of the **STEP**. The **STEP** class consists of the following slots:

- slot **index**, which is used to uniquely label a particular step to be performed by the user as found in the step-by-step procedures in the *Cours*

	How?	Tools required
2- Clean	a- clean the cables with a non-oxydizing brush b- clean metal surfaces with a rough file c- clean moulds with a paintbrush and/or rod d- clean oil using a solvent	1 non-oxydizing brush 1 rough file 1 paintbrush, 1 rod 1 bottle of manufacturer's solvent

Figure 3.4: Steps **2-a**, **2-b**, **2-c** and **2-d** in the training procedure

de Soudage;

- slot `title`, a single keyword identifier for a step or group of steps in the *Cours de Soudage*;
- multislot `steps-needed`, which stores all the immediate prerequisite steps that must have been performed by the trainee before the current step can take place;
- multislot `tools-needed`, which stores all the tools required to perform the current step;
- slot `action-string`, which stores the action to be entered by the user to correctly specify that the step is to be carried out. The expert system employs a parser to understand all the combinations of action phrases that the user may enter.

These slots will now be explained in detail. Figure 3.4 reproduces part of the welding procedures presented in the *Cours de Soudage*. These four steps, indexed **2-a**, **2-b**, **2-c** and **2-d**, all grouped under the step title **Clean**, describe how to properly clean equipment and materials to perform a good-quality alumino-thermal weld. From this figure it can be seen how slots `index` and `title` can be defined directly from the *Cours*. The information for multiple slot `tools-needed` can be extracted from the rightmost column of the table.

The slot `action` stores the action to be performed in the current step. In order to break down the action sentence entered by the user into meaningful components, a parser

is developed which uses field delimiters, including some prepositions and conjunctions of the English language, as reserved words in WITS.

The multiple slot **steps-needed** requires more explanation. Because many of the steps in the welding procedure are ordered, in that certain steps must occur before others, the slot **steps-needed** stores the steps that must have been performed by the user before the current step can be carried out. Two issues should be noted when specifying prerequisite steps for this multiple slot:

The slot **condition** dictates whether or not the particular instance of the step needs to be performed, based on the initial conditions of the welding scenario such as the prevailing weather conditions and the site preparation.

First, only *immediate* dependencies need be specified. For example, if a step **A** must be performed before a step **B**, and step **B** must be performed before a step **C**, then the **steps-needed** slot for step **C** need only contain **B** since the **steps-needed** for step **B** already specifies step **A**. The requirement that step **A** be performed as a prerequisite for step **C** is therefore implicitly specified. This is pointed out in order to avoid storage of unnecessary data in the slot, which can become quite cumbersome when a vast number of steps are specified.

Second, one must be careful not to unwittingly create step dependencies that may seem to exist by virtue of the sequential format of the training procedure. This is in the interest of preserving realism, an issue that becomes more apparent when VR characteristics are built onto the system. For example, if a step **G** consists of the measuring of two points with an ohmmeter, it must be realized that such an action can, in fact, be performed at any time and should not have any prerequisite steps except, perhaps the acquisition of the ohmmeter itself.

Class TOOL Each step is typically associated with one or more tools, without which the step cannot be successfully performed. Tools are assigned their own object class, with the following slots:

- slot **id**, which stores the tool identification code if applicable,

- slot **type**, which stores the functional name of the tool,
- slot **user**, which stores the name of the user in possession of the tool, and
- slot **status**, which stores the tool's condition or status of **new** or **used**.

The tool **id** is the unique identification code for the tool. The **id** should ideally be the same as the name of the tool object instance. The apparent redundancy here is due to the fact that the manipulation of instance names is not always convenient.

The *functional* name of the tool, which is stored in slot tool **type**, is distinct from the tool's *instance* name. This becomes relevant in the real and practical situation where multiple instances of particular objects exist. For example, it is realistic to expect more than one brush to exist in the wide array of welding tools, especially in a situation in which more than one welder is present. In the implementation, each instance of an object must be assigned a unique name, so each instance of a brush object must nonetheless be given a unique instance name such as *Brush1*, *Brush2*, and so on. However, so as not to oblige the user (or welder) to keep track of the various instance names of all the brushes, the functional name *brush* is stored as the tool's **type**. This way, a welder attempting to acquire a brush can simply refer to the type *brush* rather than to its instance name which is more obscure. If a brush is available, the welder is enabled to acquire it regardless of its instance name. This approach attempts to better model the real world.

The **user** slot stores the name of the user possessing or using the tool. This enables easy cross referencing of user names with tool types when referring to the *type* of a tool instance that has been previously associated with a particular user name.

Class MOLD Class **MOLD** is a subclass of tools, taking on all the attributes of the **TOOL** superclass. Other attributes are added to this subclass to store related information. Slots are as follows:

- slot **id**, which is inherited from class **TOOL**;
- slot **type**, which is inherited from class **TOOL**;
- slot **user**, which is inherited from class **TOOL**;

- slot **status**, which is inherited from class TOOL;
- slot **wire1**, which stores the first connection supported by the mold;
- slot **wire2**, which stores the second connection supported by the mold;
- slot **tag**, which stores a mold-related parameter;
- multislot **cartridge-norm**, which stores a mold-related parameter, and
- multislot **cartridge-trim**, which stores a mold-related parameter.

The usefulness of linking the TOOL instance to a user-name is even more useful in the context of the MOLD, when the **user** slot is used to enforce the rule that specific instances of MOLD objects cannot be modified by other users. This consideration is important in the multi-user world.

Slots **wire1** and **wire2** in class MOLD store the two connections that the mold object supports. The connections may be surfaces, and not necessarily wires.

Class PERSON The PERSON class is used to identify the various welders or personnel that may be currently undergoing the training course, of which there may be more than one at a given time in a realistic situation. Slots of this class are:

- slot **name**, which stores the name of the PERSON object;
- slot **mode**, which describes the help level permissible;
- multislot **init-cond**, which stores the initial environment conditions generated by the system specifically for the user;
- multislot **step-list**, which stores all the steps performed by the user up to the current one;
- multislot **help-list**, which stores the indices of all the steps at which the user has requested help;
- multislot **error-list**, which stores the indices of all the steps at which the user erred.

The use of the slot **name** is explained in the same way as the **id** slot for the TOOL class, the PERSON instance name is stored in the **name** slot. The permissible help levels in the **mode** slot are determined by the current user mode:

- **demo** indicates that a demonstration is presented, which is essentially the generation of help at each automatically-determined step of operation;
- **learn** indicates that help is available upon request;
- **eval** indicates that little or no help is available.

Object Manipulation and Behavior

Because the TOOL class represents the class of inanimate tool objects that changes little and the STEP class stores information that is unchanged after initialization, it is the PERSON class object that exhibits the most behavior, as one might expect. The mechanisms required to manipulate and simulate the objects in the expert system are discussed below, divided into subsections from the point of view of the operation of the system as a whole. Within each subsection the object behavior, including message handlers, are described.

Initialization

A. Defining the training procedure: Making STEP instances

- The training procedure is completely specified by the information stored in the instances of class **STEP**. Because such a procedure does not alter during the training process, the set of **STEP** instances are created upon startup of the expert system. The step data to be used are found in figure 3.5 and the appropriate information is used to fill the **steps-needed** and **tools-needed** slots of each **STEP** object instance. For example, before performing step **2d** the trainee is required to have the solvent tool and to perform steps **2a, 2b** and **2c**.
- The **STEP** object instances will be defined during implementation, using a database stored in a single text file. This enables any future updates to

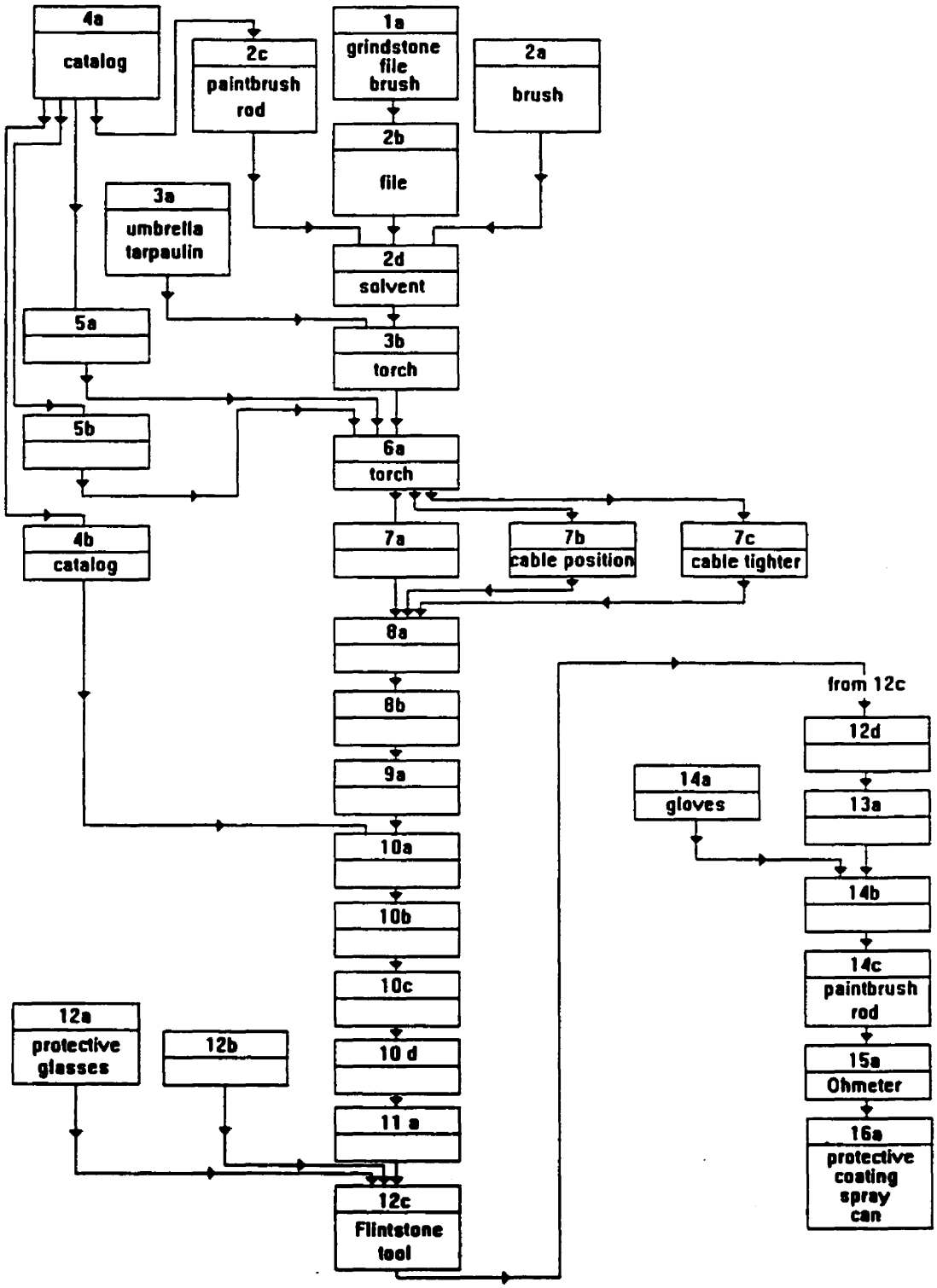


Figure 3.5: The welding steps

the training procedure to be performed by merely modifying the text file and starting the expert system again.

B. Defining the tool objects: Making TOOL instances

The definition of possible tools, which includes molds, in the training course world occurs in much the same way as does the definition of the training steps. The primary difference is that while the steps are all initialized at the start of the execution of the expert system, all tools are created "on-the-fly", or as they are needed by the expert system (or by the user).

- Like the training procedure, all *possible* TOOL and MOLD objects and their parameters are defined in separate text files according to a pre-defined format that enables straightforward updating of the information when necessary.
- Instances for TOOL and MOLD classes are created when required via CLIPS defrules that compare the user request against the tool and mold data files, or catalogues, as applicable, and confirm that the request is valid.

C. Defining the user object: Making the PERSON instance

- The expert system is currently defined for the single user, although design provisions are made for the multi-user scenario. Upon running the expert system, the user is prompted for a user-name. A PERSON class instance is then created for the user, with the user-name used both in the name slot and as the instance name. The training environment is user-specific. The training mode slot is defined by the user via the **mode** command (the **mode** command is discussed below), while the initial training parameters are generated as described in item **D** which follows, and stored in the PERSON instance multiple slot `init-cond`.

D. Generating the training environment: Defining training parameters

- Training parameters are defined to set up the training environment. Parameters consist of the welding task to be performed by the user, and the atmospheric conditions that may affect proper welding procedure. These parameters are stored in the PERSON instance multiple slot `init-cond`, and

determine the course of the training procedure based on the user's performance. The last item in this list describes how new training parameters are determined.

- To generate a welding task for the user/trainee, a mold ID is randomly selected from the mold catalogue (the data file of molds available to the expert system). The connection parameters of the mold thus selected, that is, the `wire1` and `wire2` slot data, are presented to the user together with the environmental conditions as the welding task to be performed.

- If certain steps are inapplicable due to the initial parameters, the step indices of these steps are entered into the multiple slot `step-list` of the user concerned. This forces the expert system to overlook the execution of these steps yet also satisfies any prerequisite role that these steps may have in other steps. For example, if the weather is sunny, the user should not be obliged to erect a shelter.

- To properly train a user, different welding tasks should be presented in succession, and tasks in which the user encounters difficulty should be performed more than once. It is upon this basis that the learning and evaluation modes of the expert system are designed. The `help-list`, `error-list`, and `init-cond` information of the user's `PERSON` instance are taken together for analysis, with weights of importance applied to each item that are to be implementation-defined, to determine the next set of initial training parameters for the user.

Interaction with the user The text-based user interface is a layer over CLIPS that accepts the trainee's input. The user, or trainee, has the option of entering an *action*, which consists of a valid action key-phrase, or entering one of the valid command keywords below, at a screen prompt: **help, mold catalog, tool catalog, task, tools, demo, next, next more, score** and finally **quit.**

Each entry option, including the step action, will now be discussed.

A. Performing an action

Valid user actions can be divided into two categories, those that execute a valid training step action that is defined in the multiple slot `action-strings` in the list of `STEP` class instances, and those that involve obtaining and releasing tool objects.

i. Obtaining a tool

Only valid tools defined in the text file list of available tools can be obtained by the user via the keyword **get**. At the screen prompt, a tool is obtained by the user by typing:

get <tool> [*<tool>]

For example,

```
> get brush umbrella
you now have the umbrella
you now have the brush
>
```

where *<tool>* is any one of the valid tool functional names or `types`, or a valid mold `id`. This results in the following actions by the program:

a. The *<tool-type>* specified is verified for being listed in the tool catalogue. If it is listed, then

b. a new `TOOL` or `MOLD` instance, depending on the request, is made using `make-instance`;

c. the name of the user making the request is stored into the `user` slot of the new `TOOL` or `MOLD` instance via a message-handler.

ii. Executing a step

The user executes a step by entering a key-phrase that specifies the desired action. This results in the following actions by the program:

a. The action specified is compared against all possible actions stored in the multiple slots `action-strings` of all the `STEP` instances, using the `find-all-instances` operation. The `STEP` instances created upon initialization of the expert system form the library of permissible steps.

b. If a match is found among the possible steps, a comparison is performed: The data in the **steps-needed** multiple slot of the matching STEP instance is compared against the **step-list** multiple slot of the user's PERSON object. If not all of the prerequisite steps are present in the **step-list**, the user cannot proceed with this step action, and proper feedback is given to this effect. Otherwise:

c. The **tools-needed** multiple slot of the matching STEP instance is then checked by the program. If it is empty, then the user has correctly performed the step. If it is not empty, then the user is prompted with the question, **with what?**, and user input is awaited by the program; if the user correctly enters one of the tool types listed in the **tools-needed** slot of the applicable STEP instance, then the user has correctly performed the step.

d. If the user has correctly performed the step, the **index** of the step is appended to the user's PERSON object **step-list**.

e. If the user has incorrectly performed the step, the **index** of the step is appended to the user's PERSON object **error-list**, and error counter of the individual user is incremented.

f. If the user has requested **help** to perform the step, the **index** of the step is appended to the user's PERSON object **help-list**, and the help counter of the individual user is incremented.

Appropriate feedback is generated to the user in any of the above cases. A simple interaction consisting of a valid cable-cleaning step using a brush (see figure 3.4) is as follows:

```
> clean cable
with what?
> brush
```

B. Using the score command

The **score** command generates information about the user's performance. It not applicable in the **demo** mode of operation of the expert system. The following is displayed as a result of the command:

- the number of help requests made by the user, and
- the number of errors made by the user, together with the step index of each error.

C. Using the next **command**

The **next** command displays a possible subsequent action that can be performed by the user, while the addition of the **more** parameter displays *all* possible actions that can be performed at the current stage of the training. This command is not available to the user in the evaluation mode.

D. Using the tool catalog **command**

The **tool catalog** command displays the possible tools that are available for use in the expert system. The information is reprinted from the tool data files.

E. Using the mold catalog **command**

The **mold catalog** command displays the possible molds that are available for use in the expert system. The information is reprinted from the mold data files.

F. Using the help **command**

The **help** command, displays the list of commands that the user can use to interact with the interface.

G. Using the tools **command**

The **tools** command displays an inventory of tools currently in the user's possession, obtained via the **get** command.

H. Using the demo **command**

The **demo** command presents to the user a complete demonstration of a particular welding scenario. The demonstration algorithm is explained in detail later in this section. This command is not available to the user in the evaluation mode.

I. Using the quit command

The **quit** command exits the expert system, restoring the regular CLIPS prompt.

Demonstration mode The demonstration mode of the expert system runs automatically after the initial conditions for the welding task have been set. This provides a demonstration on welding for the new user. The demonstration is presented upon entering the **demo** command. It is implemented by following the flowchart of step dependencies shown in figure 3.5, in this way:

a. The initial step to be performed in the flowchart must be predefined. This initial step index is taken to be the current step index. The indices of all the steps performed are stored in the `step-list` multiple slot of the user's `PERSON` class, just as in the `learn` and `eval` modes of operation.

b. The longest string among the `action-strings` of the `STEP` instance having as the current step index as its `index` (found by a `find-instance` operation) is printed out.

c. The next step to be performed is determined by selecting the `STEP` instance that specifies the current step index as a prerequisite (in the multiple slot `steps-needed`). If all the prerequisites of the next step thus selected have been satisfied (by checking the `step-list`), perform step *b.* as before, until it is impossible to continue or there are no more steps.

d. If it is impossible to continue because the prerequisites for the next step have not all been performed, use one of the *prerequisite steps* of the impossible next step as the next step to be performed. Proceed to step *c.* above.

This process can be viewed as a 'pseudo-depth-first' algorithm that navigates through the tree or graph structure resulting from the interdependencies of steps. This is illustrated in figure 3.6.

In this way, a demonstration of how to properly perform a weld under the given initial conditions can be presented. Once all objects have been interfaced to the VR setting, a visual (and possibly auditory) demonstration can be created.

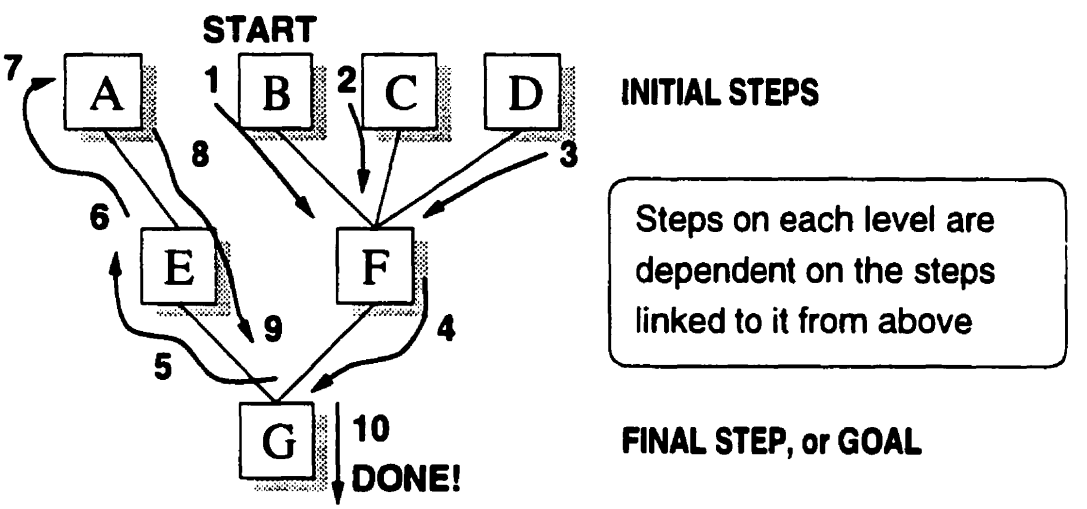


Figure 3.6: An oversimplified illustration of the step graph traversal

3.3 3D Virtual Environment

The developments in virtual reality have permitted training systems to take advantage of this new technology and create full training environments where users can practice procedures that incorporate body postures, object manipulation, and team collaboration. As already mentioned, one of the main goals of this thesis is to incorporate a "Human Operator" into the Virtual Training Environment that would facilitate and make more natural the user interaction with the computer generated VR world.

As its main task, the virtual operator, (VO), must be capable of carrying out the welding procedures described by the WITS expert system. However, our experience from the previous project ESOPE-VR and its lack of re-usability have shown that it is not enough to satisfy the requirement of the specific welding project and environment. We need to adopt a system design which would allow us to have greater freedom and flexibility in addressing future requirements. This led to the concept of the McGill Virtual Operator: a VO that would not only perform the grounding procedure, but will eventually be able to represent the human user in any virtual environment and thus embody the user's interactions between the user and any VR world.

The McGill VO must also allow natural and comprehensive human-computer interactions. Thus the devices accessible to the user, such as the 2D mouse, must be easy to control in such a way that they will result in implicit/predictable actions in the VE.

The following sections will present the objects and physics of the virtual environment, the McGill VO, and the interactions between them.

3.3.1 Objects in the VE

Building blocks

The 3D objects are the basic building blocks of the virtual environment. They consist of a graphical representation that can be either simple or complex.

- Simple Objects are geometric graphical entities that consist of one structure, i.e. they have no moving sub-parts. The structure can be built of

one or many basic geometric primitives, such as box, sphere, cylinder, etc. These primitives are all "glued together" and manipulated as one object having the same center of gravity, bounding box, and physical properties.

- Complex Objects are composed of two or more simple objects, where each has distinct physical and kinematic properties and can thus be manipulated separately. For example, the McGill VO, described in section 3.3.3, is constructed as a complex object, where each joint is controlled separately.

Static and Dynamic objects

The 3D objects can be divided into two major groups: Static and Dynamic. Static objects are graphical entities that stay the same once loaded in the virtual universe. The attributes they hold: midpoint, radius, orientation, local coordinate frame axes, extents, bounding box, and pivot point keep their initial values until the end of the simulation.

A dynamic object's attributes, on the other hand, can be modified during the simulation. They can thus be interfaced with external input devices, through specialized drivers. The user can then affect these objects' attributes and hence interact with the virtual world. The McGill VO is such an object. It gives the user complete control over its different elements, as will be seen below. The VO's torso is controlled through the pedals to permit the VO to travel around the virtual universe. The hands also, are controlled through the Logitech Baron tracker (for palm position) and the EZglove (for finger flex). Alternatively, the PowerGlove can be used to control both the hand position and the finger postures (all open or all closed).

Intelligent Objects

Dynamic objects can be categorized into two kinds: Dumb objects and intelligent objects. Intelligent objects are objects that hold some notion about the virtual environment that surrounds them. They can react to events in the world, can be indirectly manipulated, and can permit physics to take part in the virtual environment. Dumb objects, on the other hand, do not contain such information and only react through predefined actions

such as animations or in response to external input devices such as the glove. The McGill VO graphical elements can be considered, to a certain extent, a dumb object: it is completely controlled by the user otherwise it behaves like a static object. A clock on the wall is another example of a dumb object which reacts according to a predefined task.

Intelligent objects are the second most important part of the user interface after the McGill VO. They can be manipulated indirectly due to their understanding of the environment around them. To achieve this they hold specific knowledge and constraint features allowing some interaction with the VE. The word "allowing" is very critical since it indicates the essential difference between these objects and the other categories mentioned previously. Non-intelligent objects when manipulated are under the complete control of the manipulator object. They do not allow or forbid as do intelligent objects.

Intelligent objects hold informations such as: free, touched or grabbed. This information is important and helps determine the status of the object and type of manipulation that can take place. This is very important when we talk about training systems where the user is not allowed to do whatever pleases him but rather is asked to perform a specific task. Thus eventually achieving the training objective.

With intelligent objects, fitting the right wire in the right mold becomes easy. The mold knows what exact wire fits and thus would only accept an object instance of this wire object class. Also, the brush can wear out since it knows about its usage status and this aspect can be displayed graphically on the object to give quantifiable feedback to the trainee. These objects also know about physics laws. They thus react to external virtual forces such as gravity and friction. An object that is dropped in free space, would fall down until it collides with another surface.

3.3.2 Physics of the VE

To achieve a realistic training environment, the user must be presented with an environment that reacts in the same way as in the real world. This cannot be done successfully without the introduction of the real world physical laws. That is exactly what was done

by introducing the *PhysicsAgent* object class in conjunction with the intelligent objects presented above. Thus giving these virtual objects the ability to react autonomously to external virtual forces. A physical model should take into account collision, gravity, friction, momentum, etc. These effects can be viewed as constraints on objects.

Collision Detection

Collision detection is the most basic physics principle applied in the VR world. It transforms the VR experience from a surreal fly-through to a more realistic travel-in experience. Collision detection permits objects to be treated as physical entities that can be manipulated. This is true due to the fact that with collision detection objects can not inter-penetrate. This allow objects to be touched, lifted, pushed, grasped, stacked etc. which is exactly how physical objects react in the real world. WTK offers some integrated functionality to handle collision detection.

Gravity and Attraction forces

In this research, we wanted to go one step further and add more physical laws into the virtual environment to increase its realism. The gravity law is the most important of these since, like the collision task, it provides a realistic visual and functional feedback. When objects are dropped in space they normally fall down due to the gravity force.

Furthermore, the principles of gravity field and attraction can be used in the simulation to facilitate some simple tasks such as stacking objects on top of each other, or more complex tasks such as magnetic attraction by introducing exaggerated attraction forces between two objects. The concept of attraction forces can be used here to facilitate the task of fitting the wire into the mold's hole: if the correct wire falls within the field, it is snapped into the hole.

Others Concepts

Other physics concepts, namely mass and force, have to be added to give a more realistic virtual experience. Currently, constraints are used to overcome the lack of such physics

concepts in the VE. For example, when the hand collides with another object it is constrained to stop so that it would not penetrate into the object. Unfortunately, with such constraints the push, bend, and break actions can not take place. However, once force and mass handling is added, to the virtual world, the decision of what happens if the hand collides with another object would be easy to determine.

3.3.3 McGill Virtual Operator

The Human Geometric Model

The design of the operator's geometric or graphical model was done using a hierarchical structure following a bottom-up construction process. The basic components were designed first specifying their spatial layout, shape, and other attributes affecting their appearance such as color. These basic components have been then used as building blocks to create higher-level entities, which in turn serve as building blocks for yet higher-level entities. In building the higher-level entities connectivity of the basic components had to be specified in order to get a robust configuration or topology.

For the alumino-thermal welding application what was needed is an interface capable of transporting the operator to the assigned ditch and allowing him to perform the welding task at hand. To achieve this, the VO was designed to consist of a body and two hands. The body is used as the reference point for locating and transporting the VO while the two hands are used to interact with the VE and to perform the welding task.

The Hierarchical geometric model of the VO is shown in figure 3.7 symbolized by a directed acyclic graph (DAG). The McGill VO is composed of two hands constructed from a palm and five fingers including the thumb. The palm and the fingers have "sensitive skin" attached to them. It is important to note that this model is an highly simplified model. A full human model would include elements such as arms, legs, and head.

For the purpose of this research, this simple model works best since it allows most of the basic functionality to be supported by today's device technology. The addition of the other parts of the body would make the training experience more realistic if running in a collaborative multi-user scenario where the trainees would be able to see each other's

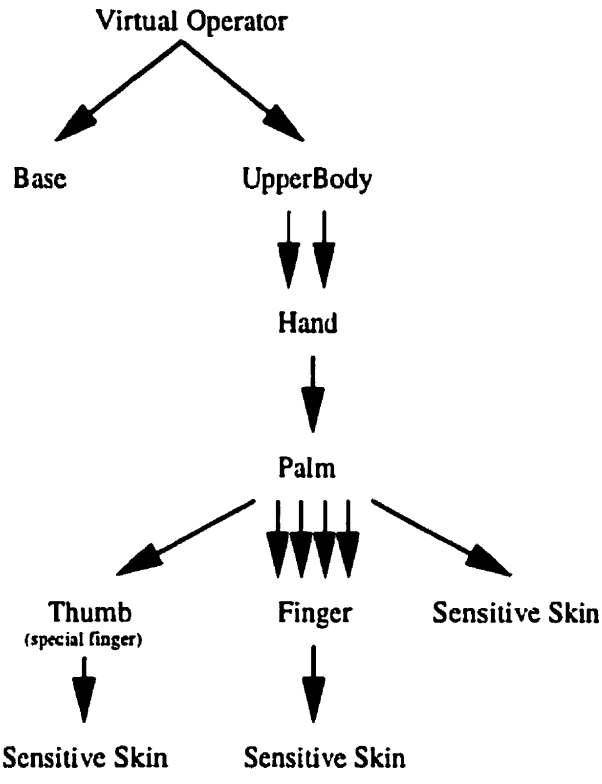


Figure 3.7: Hierarchical geometric model of the McGill VO

representation in the VR world. In the present stage of the VR training system, the trainee works alone and does not actually view his virtual body representation unless we introduce some virtual mirrors.

A complete body representation can be achieved by simply adding the different geometric graphical parts and the appropriate attachments to the existing body objects. This is mainly due to the McGill VO's hierarchical model presented above resulting in the following two advantages:

- Enables the construction of complex objects, such as the McGill VO, in a modular fashion.
- Enables update propagation where change in the definition of a building-

block object is automatically propagated to all higher-level objects that use that object.

Basic building blocks

In building the McGill VO geometric model we used simple atomic components which are defined in WTK in terms of lower-level geometric primitives, such as vertices and lines. The atomic components used are

- The 3D basic shapes such as cylinders, spheres, and parallelepipeds to create the body, hand palms and fingers.
- The 2D polygons to create the hand's sensitive skin.

A less abstract basic component of the VO is the joint object. All the joints are attached together in a serial chain linkage mechanism, thus forming sub complex objects, such as the fingers and palm objects, that are themselves linked together to form the hand (see figure 3.8). Additionally, each of the joint link objects has its own parameters associated with it containing information about its behavior as well as its graphical representation. Elements of the graphical representation are the shape, color, position and orientation. Behavioral information encompasses anatomical constraints and articulated functionality.

The body center The torso element represents the reference location for the VO in the virtual environment. Thus it permits the VO to navigate in the virtual world. The user can use the 2D mouse sensor or the pedals to control the torso element. Note that any other available sensor could be used to control the body; this holds for any element of the VO interface. However, the pedals were the perfect choice, since as explained earlier, they limit the VO to navigate in a 2D plane. This is perfectly acceptable acknowledging the fact that humans normally do not fly, although, in VR, everything is possible.

The graphical hand model The hands elements are the visible part of the VO which is consistent with real life where the hands are the only parts visible or "in-focus" while

Figure 3.8: The graphical representation of the McGill VO hand

performing most tasks. The hands represent the primary manipulation tool available to the VO and they permit the user to affect the virtual world. The EZglove combined with the Logitech Red Baron mouse are used to control the VO hands. Previously the Mattel PowerGlove was used for the same purpose, unfortunately it lacks the resolution needed to perform precise grasping and picking manipulations.

The graphical hand model presently used (see figure 3.8) is constructed from eleven boxes: two for each finger and one for the palm. This model is complex enough to map almost the full hand-joint motion. In conjunction with the above joint boxes there are eleven polygons representing the sensitive skin areas of each joint.

Sensitive skin is displayed with lighter coloring so as to clarify to the operator which side is the front/back of the hand. This is important because the hand is constructed using simple boxes as explained above, thus there is no other visible difference between the front and back views of the hand.

Mapping the hand-joint motion The device used to sense the hand-joint motion is the EZglove. This device uses four light-tubes placed at strategical points on the glove to permit tracking two degrees of freedom per finger and mapping it into one degree of freedom. The assumption made is that each finger bends uniformly and around one axis. i.e., the angles between the joints of each finger are equal so that knowing the total flex value of the finger is sufficient to determine the flex value of all its joints.

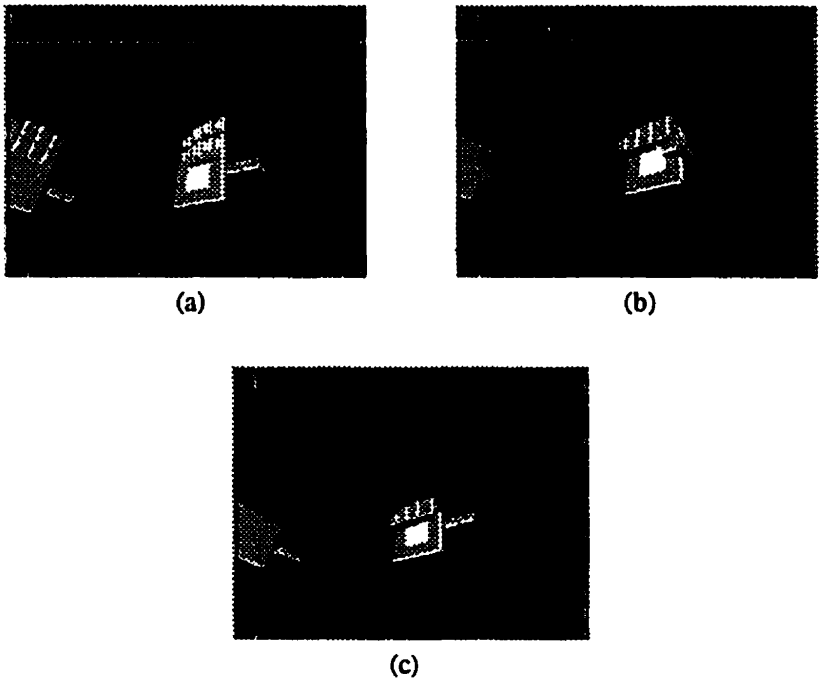


Figure 3.9: The three hand calibration postures: (a) Opened hand, (b) Closed hand, and (c) Open thumb posture

For calibration purpose, three hand positions are important: Open hand, Closed hand, and the Open thumb hand position as shown in figure 3.9. The close and open positions determine the range of flex motion to be considered. Any sensor flex value outside this range is ignored thus providing an accurate mapping of the hand motion. These bounds can also permit rescaling the angular readings to increase the resolution of the finger rotation. However, this was not needed since the 8 bit readings from the EZglove offered more accuracy than needed for this application.

With the EZglove the flex value of the thumb is determined using a sensor placed

between the index and the thumb to capture the full rotation of the thumb which goes against the other fingers. The drawback of this setup is that the thumb rotation readings is affected by the rotation of the index. The Open thumb hand position is used to filter out this effect.

The glove calibration is done manually for the time being. A semi-automatic calibration, where the simulation would ask the user to perform the different gestures presented above, would be preferable to correct any hardware miss-calibration.

Although this model of the human is very simple, it is sufficient for basic interaction with any virtual environment since it provides a way of locomotion to transport the VO to any part of the virtual world and it is equipped with two hands that permit the VO and thus the user to interact with objects in the virtual world.

3.3.4 Hand Manipulation

Touch action

The touch action is determined by the collision detection function. Once we can determine that two objects are colliding we can infer that they are touching. Since the touch action is a prerequisite for the grasping action, it should be considered as equivalent to a select interaction task. It permits the user to select the object on which manipulation is to take place. The manipulation actions are not restricted to grasping but include other actions such as pushing or deforming. Grasping is the first that comes to mind since it is the prerequisite for many other manipulation actions in the virtual world.

In order to emphasize the touch action, feedback has to be presented to the user. One way of doing this is to change the color of the object to visually show that it has been selected. This technique however is not suitable for VR world where realism is an important issue. In real life, objects do not change color when touched. Another way to deliver feedback is through the use of audible sounds. Every time the user touches an object he/she would hear a touching sound. The draw back of this technique is that of the annoyance of the sound when the object is touched for a long time. Faced with no other solution we picked the latter and found that the user adapts positively to the

interface.

Grasping

Grasping is a complex process that must check for collision detection, hand and object status. As mentioned previously, only intelligent objects can be grasped and thus static objects are ignored in this process.

Whenever the hand makes contact with a dynamic object, there is a possibility of a grasping situation. How do we determine if the operator really wants to grasp the object? The answer to this question is to set certain rules or assumptions that determine when the operator is manipulating an object and when he is not. Referring to the real world, grasping is usually done by holding the object using at least two fingers where most of the time one of the fingers is the thumb. Thus an easy rule would be to associate grasping with the event of a thumb and any other finger both colliding with the desired object to be manipulated. However this is not enough, since an open hand touching the object with both the thumb and another finger would be considered as grasping. Thus, more restrictions were introduced such as grasping should involve contact between the object and the internal surface of the hand.

Manipulating grasped objects

With the presence of the McGill Virtual Operator, the trainee can associate himself directly with the virtual interface. The trainee controls the McGill VO hands through the gloves. The direct association with the virtual hands removes the obstacle found when using a different input device such as the mouse or the joystick where the user has to do an indirect mapping of the different interaction tasks and their device dependent techniques. Thus the McGill VO interface permits direct manipulation in the VR wold. The trainee grasps objects as if they were real. This favors a total immersion experience as long as the virtual world understands his actions.

Once the object is grasped, it is attached to the VO's hand and follows its movement. The user can thus, move the grasped object and perform the required training tasks by

interacting with other objects. In the case of the alumino-thermal welding task, the user will grab an object such as the brush and perform a task such as cleaning a wire.

Chapter 4

Implementation and Test Results

4.1 Building the virtual world

When the virtual simulation starts it first initializes the universe which in WorldToolKit is done by initializing the default viewpoint. Then the terrain along with the static objects that enrich it are created. "Manipulated" objects are then added to the world followed by the Virtual Operator(s). Next, the sensor devices drivers for the 2D mouse, Red-Baron mouse, pedals and the gloves are started and are attached to the proper objects. The 2D mouse and the pedals are attached to the viewpoint while the Red-Baron and the gloves are attached to the hands.

Light objects are next defined and loaded into the world. Then the keyboard handler is set up to process key events. The most important of such key events are the 'd' keystroke which detaches any grabbed object from the hand and the 'q' key stroke which exits the simulation and frees any remaining used system memory resources.

The WorldToolKit rendering engine then initializes the graphical representation of all objects and prepares the universe for rendering. Finally a connection is established with the expert system before WorldToolKit enters the main simulation.

From this point, the simulation will keep running in the infinite *Action Loop* until the user quits where the simulation halts. The following events are scheduled to run in the following order at each turn of the *Action Loop*:

- objectsensor event: polls data from the input sensors. Data from non-WorldToolKit supported sensors (glove and pedals) is processed through specialized functions.
- action event: calls the action function which contains the core of the simulation and enables user interaction with the simulation.
- tasks event: WTK permits specific animation tasks to assigned to objects in the simulation.

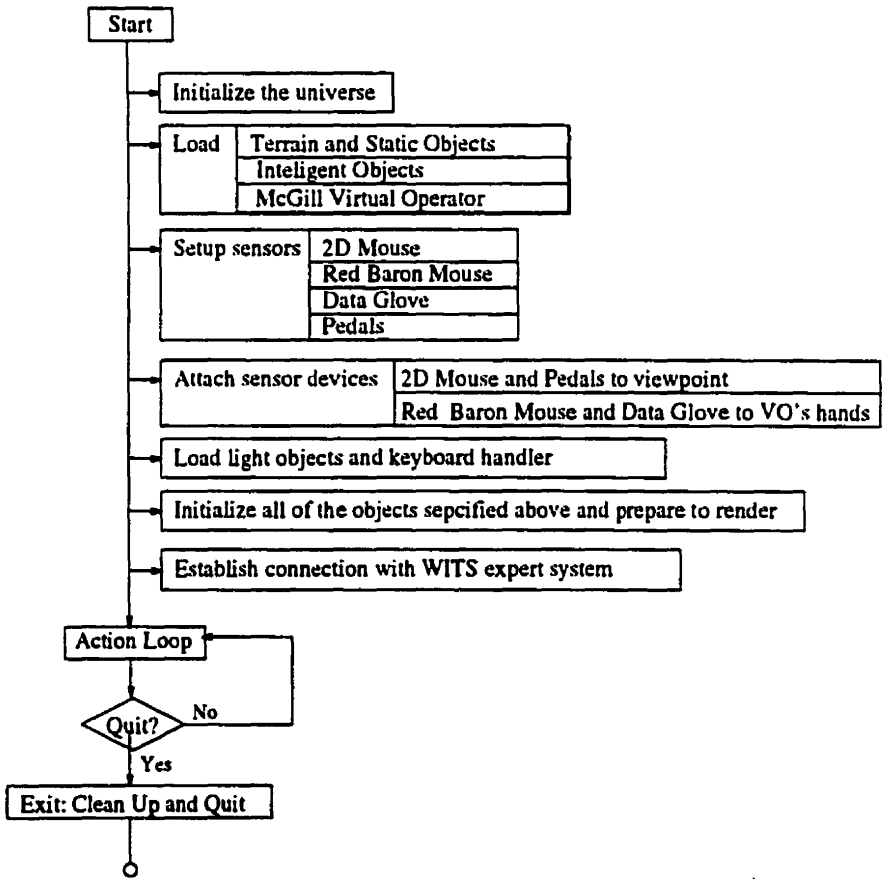


Figure 4.1: The Overall program flowchart of the WITS training system

4.2 Creating an instance of the operator

The operator body is assembled from the different elements that constitute it, in our design: two hands and a torso. These elements are positioned correctly relative to each other with the proper attachments, pivots and constraints.

One would expect that the virtual operator would be positioned at the viewpoint. However, at that position the VO would be behind the viewable area mapped on the screen. The VO is actually placed in front of the viewpoint (camera) so as to have its hands visible to the user as shown in figure 4.2.

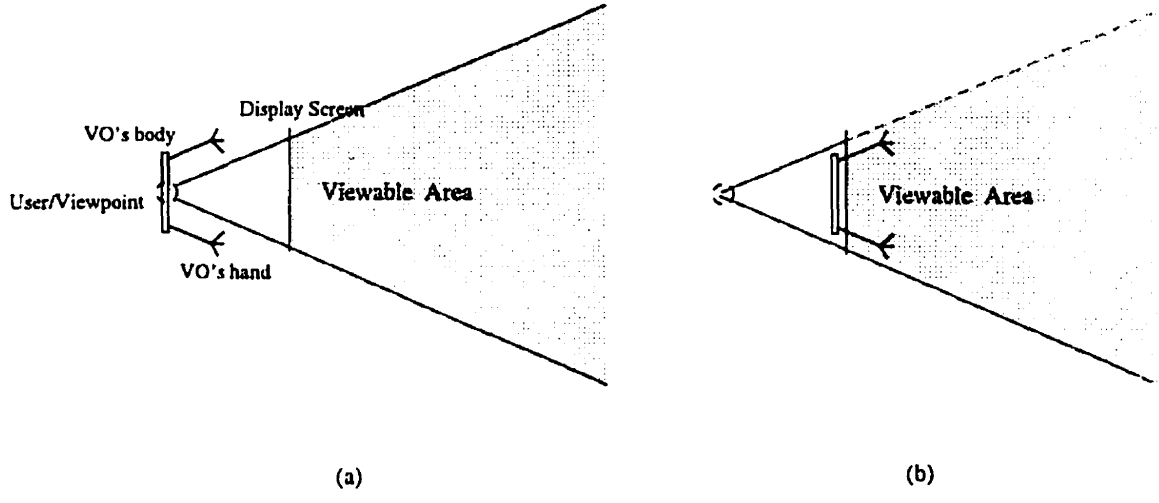


Figure 4.2: (a) VO placed at the viewpoint, (b) VO placed in front of the viewpoint

In our simple implementation the VO has no forearm and elbow parts. As a result, a mechanism had to be implemented to keep the hands from wandering away from the body. Normally this is not necessary since the real hand is itself limited by the articulations of the elbow and arm structures).

The mechanism used makes sure that the hands are always inside a specific boundary. Whenever the hands move, the movement is validated against the boundary so that they would not cross the boundary limits. We have chosen the boundary to be a sphere with a radius equal to the body radius. This choice is close to reality and allows customization to different users and/or environments by simply changing the scale factor.

This validation mechanism was not trivial to implement since, we needed to work in two coordinate systems: the world coordinate system and the body local coordinate system, as can be seen in the code segment below. For details on the coordinate systems refer to figure 3.2.

```
/*************************************************************/
/* Here the position of the palms are tracked and sent to    */
/* the glove_bound function to be verified and reevaluated   */
/* if necessary                                              */
/*************************************************************/

WTpq *posq;

posq = (WTpq *)malloc(sizeof(WTpq));
WTpq_init(posq);

WTobject_getposition(hand[LEFT]->palm, posq->p);
WTobject_getorientation(hand[LEFT]->palm, posq->q);
glove_bound(posq->p);
WTobject_moveto(hand[LEFT]->palm, posq);
WTobject_moveto(hand[LEFT]->bpalm, posq);


WTobject_getposition(hand[RIGHT]->palm, posq->p);
WTobject_getorientation(hand[RIGHT]->palm, posq->q);
glove_bound(posq->p);
WTobject_moveto(hand[RIGHT]->palm, posq);
WTobject_moveto(hand[RIGHT]->bpalm, posq);

free(posq);

/*************************************************************/
/* The glove_bound function accepts a 3D position vector.    */
/* It then compares that position vector with the body_radius */
/* vector to make sure that the position sent is in the sphere */
/* defined by the body_radius vector. If this is not true, it */
/* updates the sent position vector by replacing its value   */
/* with the value of the body_radius vector.                 */
/* For the comparison to make sense, the coordinate system   */
/* frames must match. Thus the sent position vector is first */
/* translated from the world coordinate frame to the         */
/* body frame. At the end it gets retranslated into the world */
/* coordinate frame.                                         */
/*************************************************************/

void glove_bound(WTp3 &pos)
{
    WTp3 pos_body;
    WTobject_world2local(body->torso, pos, pos_body);
    if (labs( pos_body[X])> body_radius)
if(pos_body[X]>0.0)
    pos_body[X] = body_radius;
else
    pos_body[X] = -body_radius;
    if (labs( pos_body[Y])> body_radius)
if(pos_body[Y]>0.0)
    pos_body[Y] = body_radius;
else
    pos_body[Y] = -body_radius;
    if (labs( pos_body[Z])> body_radius)
if(pos_body[Z]>0.0)
    pos_body[Z] = body_radius;
```

```
else
    pos_body[Z] = -body_radius;
    if ( pos_body[Z]<0.0)
pos_body[Z] = 0.0;
    WTobject_local2world(body->torso, pos_body, pos);
}
```

4.3 Creating an instance of the hand object class

To construct a new hand the following parameters are needed: hand dimension, color, texture and hand type (left or right). These basic parameters can then be used to shape the hand elements: palm, fingers, sensitive-skin, and bounding objects. This includes determining the: spacing, width, length, thickness and displacement of these elements.

The hand initialization process involves allocating memory, setting flags and assigning labels to each element of the hand so it can be distinguished and referenced correctly.

The process of hand creation involves building each element of the hand independently and then assembling all the parts with defined relationships between them. Thus, when the elements are created they are placed correctly relative to each other. Pivots and attachments are then defined and added between the elements. Figure 3.8 shows this very clearly.

4.4 Hand Functionality

One important functionality of the hand is finger rotation which is present in most actions involving hand manipulation such as grabbing.

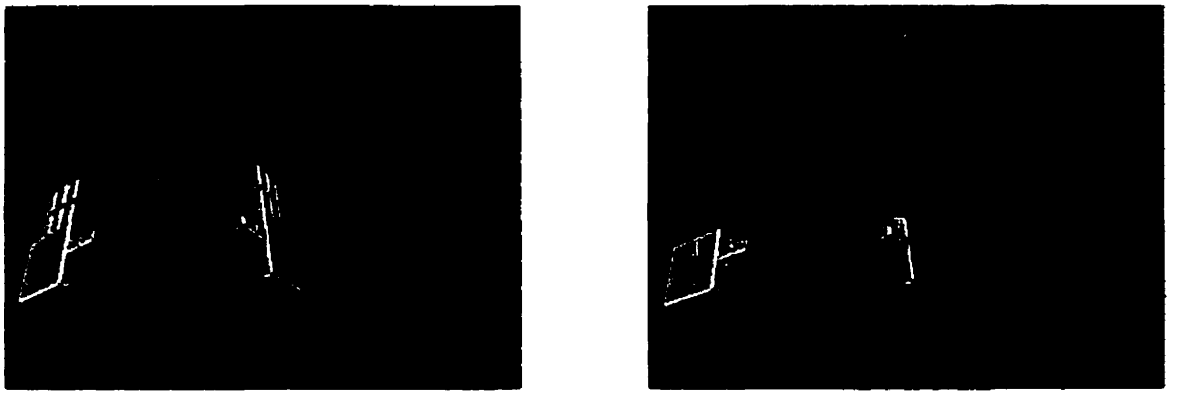
The parameters needed for the finger rotation process are the hand, the target finger index, and the amount of flex detected by the sensor. The finger flex value detected is used in conjunction with a table containing a set of minimum and maximum flex values. The values provided in the table help limit the range in which finger flex values are processed and thus help eliminate noise problems. If the flex value is less than the minimum indicated in the table the finger is set at zero angle which correspond to the open finger posture. If the flex value is higher than the maximum indicated in the table

then the finger is set to maximum angle which correspond to the closed finger posture. In any other case the finger is rotated according to the flex value.

The process of rotating the fingers is not a trivial process. This is due to the fact that the hand changes orientation continuously. In order to perform such a rotation correctly the following is done:

- Store the present hand orientation;
- Reinitialize the hand orientation to its value at creation time (zero rotation);
- Rotate the fingers according to the sensors flex values
- Rotate the hand using the orientation parameters stored.

Once the rotation is done the process tests for any collisions. If the fingers or the palm collide with an object in the VR world, the object is warned of the collision and flagged as touched. The process then checks to see if the collision results in a grab posture. Figure 4.3 illustrates both situations, i.e. the touch and grasp positions.



(a) (b)

Figure 4.3: The hand grasping the brush: (a) the touch position, (b) the grasp position

Another process ensures that none of the hand parts penetrate through the objects by restoring the finger flex value and palm position to their previous values once collision is detected. This was added in response to users' feedback where they were frustrated when

the object grasped penetrated through their virtual hand. In the present implementation of this process, the hand can rotate freely around the touched object. Although, the users did not consider this to be a real drawback, a more sophisticated algorithm is needed to constrain the hand orientation in a more realistic manner.

As mentioned in chapter 3, audible sound feedback is delivered to the user every time he touches an intelligent object. The effect of audio feedback and stereo vision on distance evaluation and object manipulation in the 3D environment is depicted in Table 4.1 where subjects were asked to describe their impressions using one of the following descriptors: Excellent, Good, OK, Bad, or Frustrating. The subjects referred to in this table are volunteers who tested the system. With the exception of "subject #1", the volunteers had no real experience with VR systems. The volunteers were asked to reach and grasp five different objects: a *brush*, a *grindstone*, a *rough-file*, a *box*, and a *sphere*. They repeated the task four times, one for each combination of audio and visual feedback presence as shown in Table 4.1. The users feedback shows how they appreciated the sound feedback feature and found that it really removes the frustration of selecting (touching) the objects to be manipulated.

Most volunteers, with the exception of "Subject #3", showed some improvement when using stereo vision. "Subject #3" simply did not succeed in seeing the 3D depth provided by the shutter glasses. The others found this depth cue helpful in evaluating distance as well as in locating and touching objects.

With both audio and visual feedback the users showed great satisfaction. Users, as expected, were able to cope with the lack of tactile feedback. They adapted to the task presented using the available resources. However, when asked whether they would appreciate the addition of tactile feedback they responded positively.

4.5 Manipulated objects

In order to improve the video frame rate performance of the VR simulation, objects have been divided into two categories: *Static* and *Dynamic* objects. The *static* objects are objects that do not move and that can not be manipulated. On the other hand, *dynamic*

Experimental conditions:	Subject #1	Subject #2	Subject #3	Subject #4
With no audio feedback or stereo vision, how well is:				
Distance evaluation	OK	Bad	OK	Bad
Object location and touching	Bad	Frustrating	Bad	Bad
Object grasping	Bad	Frustrating	Frustrating	Frustrating
With stereo vision, how well is:				
Distance evaluation	Good	Good	OK	Good
Object location and touching	Good	OK	Bad	OK
Object grasping	Good	Bad	Frustrating	OK
With audio feedback, how well is:				
Distance evaluation	OK	Bad	OK	Bad
Object location and touching	Good	OK	Good	Good
Object grasping	Excellent	Good	Excellent	Good
With both audio feedback and stereo vision				
Distance evaluation	Good	Good	OK	Good
Object location and touching	Excellent	Good	Good	Excellent
Object grasping	Excellent	Good	Excellent	Excellent

Table 4.1: User ratings for object grasping with audio feedback and stereo vision

objects are objects that can move and be manipulated by the VO. This classification reduces overhead associated with checking all the objects in VR world for manipulation tasks.

The manipulated object class from which the object instances are created, contains more useful information than what is available with the basic static objects. This additional information gives the manipulated objects the advantage of understanding the world around them and thus simplifies most of the tasks involving manipulation of such objects.

Basic constructors were created to instantiate simple objects such as a box or a sphere as well as more complex objects that are loaded from a 3D representation file with *.nff* of *.obj* formats.

The grasp function checks against a predefined acceptable hand posture which includes the palm and a fingers position and orientation when touching the object. If such a posture is found the object gets "attached" to the associated hand. The intelligent objects know enough about the external world so as to take the appropriate decisions and actions. Thus different objects can demand different hand grasping postures. Furthermore, special object constructors were made available that would accept a default

grasp position and orientation as part of their parameters. These parameters would then be stored in the corresponding object's default grasp position and orientation attributes. The default position and orientation attributes can then be used to automatically position the object in the user hand when the "precise grasp" flag, which is part one of the object's class parameters, is enabled at initializion time. Its job is to facilitate the use of such tools since they would be correctly held.

Our volunteers were asked to grab two similar brushes distinguished by their colors, a green brush with the "precise grasp" feature on, and a brown brush with the feature disabled. They were then asked to perform the clean wire action using each brush in turn. All users but #1, felt more comfortable when working with the green brush. They had no problem in grasping the brown brush. However, using the brown brush, they needed to adjust their grasp more than once before they could clean the wire comfortably. User #1 wanted to be more in control and did not like the concept of being constrained to one grasping posture. From the comments received, we can say that the automatic grasp feature, although un-natural, is very helpful by showing that the system is responsive to the trainees actions and understands their intentions.

4.6 Physics laws

The purpose of these algorithms is to build a basis for the virtual environment that is similar to the real world. To do so we chose to start by incorporating the concept of gravity which is one of the essential forces that affects objects in the real world. Once all the physical laws are adapted to the VR world it would become more realistic and the user would thus feel more immersed.

The gravity task exercises a downward pulling force on all the objects in the VR world. Thus, all objects that are not constrained should drop vertically until they reach the "terrain" or until they become constrained. The falling movement must be accompanied with a simulation of acceleration. This introduces a problem since acceleration in our VR world means bigger steps which means that the object might cross another object without even detecting it due to the nature of the collision detection functions. Thus

an algorithm was developed so that the bounding boxes expand and shrink dynamically according to the velocity at which the object is falling. When the object is moving at a higher speed its awareness of the external world increases so that it can take the appropriate actions in time.

In addition to free falling vertically, once an object hits the floor, it is supposed to reach a stable posture. Thus proper rotations are performed so as to align the object with the "terrain".

All four volunteers appreciated the presence of gravity. They felt more immersed and discovered new ways of handling the objects around them, taking full advantage of the gravity feature. With the gravity feature, the user can lift and drop objects, stack them on top of each other, or on his hand. Essentially the environment became alive!

4.7 Peripherals

Navigation in the 3D environment is achieved on two levels: navigation of the VO's body and navigation of the VO's hands. The Pedals are used for navigating the body. They are connected to the game port of a PC running the WIN 3.11 operating system from which a TCP/IP socket connection transmits the input values to the SGI. Drivers were created for the SGI and PC platforms. A server runs on the SGI that permits data to be transmitted between the two clients: (1) The PC/pedals client polls the data from the Joystick port and sends it to the SGI through the established socket. (2) The SGI/WTK client polls the data from the established socket and uses it to position the VO's body.

The Logitech Red Baron mouse is used to control the position of the hands. The driver was already supplied in the WTK software. As can be seen from table 4.2, the trainees found the Baron mouse very comfortable for navigating the hands and prefered it to all other devices, namely the 2D mouse and the PowerGlove. As expected the Pedals device was the best choice for body navigation. The trainees had difficulties at the beginning to understand the mapping between the foot movement and the VO's movement. However, this difficulty was overcome in a matter of minutes. The users liked the concept of navigation using their feet and that is exactly why the pedals rated better

than the joystick.

Ratings of devices	Subject #1	Subject #2	Subject #3	Subject #4
For body navigation				
Pedals	1	2	1	1
2D mouse	4	5	3	3
Red-Baron	3	3	4	4
Joystick	2	1	2	2
PowerGlove	5	4	5	5
For hand navigation				
Red-Baron	1	1	1	1
PowerGlove	2	2	2	2
For hand manipulation				
EZglove	1	1	1	1
PowerGlove	2	2	2	2
For Ease of adaptability				
Pedals	4	3	3	4
2D mouse	6	5	4	6
Red-Baron	3	4	5	3
Joystick	2	2	2	2
EZglove	1	1	1	1
PowerGlove	5	6	6	5

Table 4.2: Performance ratings of the different peripheral devices

Along with the Baron mouse attached to the user hand, a data glove is used to permit hand manipulation of 3D objects. The glove's data is pooled by a mini-controller: HC11 in the case of the PowerGlove, and a Basic Stamp [Slavkoff 1997] in the case of the EZglove. The mini-controller communicates the data to the SGI through a simple program which establishes a serial connection between the two machines thus permitting a special built-in driver [Shaikh 1995] in WTK to poll the data from the mini-controller.

Table 4.2 depicts the subjects ratings of the effect that the different peripheral devices have on the navigation and manipulation tasks. The subjects were asked to rate the devices by order of performance accuracy, where '1' indicates the prefered device. Most subjects were surprised to find the EZglove, in-house developed device, very accurate. The EZglove mapped their fingers movement with no noticeable latency. On the other hand, the PowerGlove failed to deliver such performance. This latency is due to the interpolation algorithm added in the case of the PowerGloce sensor driver. This algorithm was necessary in delivering a realistic finger flexion. Voice control was implemented to work directly with the expert system. The Verbex 7000 communicates through a serial

port with the expert system. This was extremely useful for the help function integrated in the expert system. The user can receive the replies as voice feedback which made the interaction very natural and had an important effect on the immersion level of the trainee.

Users found the Verbex 7000 hard to train but very useful once it operates properly. Although the full syntax grammar was available to the users, they mostly used the short command functions, mainly the *help* and *next* commands.

4.8 Expert System

The expert system described in chapter three was implemented in a server/client architecture. The server runs in the background on one of the Unix servers. Every time a training session is needed the server forks a copy of the expert system to service that particular session. The client session can be a normal text-based session along with the voice command interface or it can be called from the virtual reality environment itself where it communicates with the different monitoring systems. Here too, the voice interface is still supported. The text version can currently be run as a Telnet session to minerva.ece.mcgill.ca on port 3001. A web interface has also been developed allowing WITS to be operated from a web browser such as netscape at http://eddie.ece.mcgill.ca/~wits .

Although WITS was designed for the purpose of training and monitoring users in the performance of welding tasks, it can be easily reused for other training systems. The system design is such that the welding task can be represented and modeled through a step graph in which each node has other nodes as prerequisites. The step graph is defined in a database file. Hence, the design is applicable to any training system that uses a similar step structure. This was demonstrated by developing a French version of the WITS expert system.

4.8.1 WITS in French

WITS was developed in an English environment at McGill University. However, the system parser does not really understand English, it actually breaks down sentences into meaningful fields. The only important field separators are the preposition (*with, under, around,* etc.) which are defined as reserved words. Thus if we specify the steps and the reserved words in French, the system should be capable of understanding French actions. This capability was demonstrated by modifying the code so that the reserved keywords can be defined from the database file directly. Then a French version of the database was loaded into the system and run. A flavor of the resulting running system is shown in the following excerpt:

```
ata@WITS>>prochaine etape ?
Essayer l'action suivante:

 - METTRE les GANTS

ata@WITS>>mettre les gants
Il vous manque un outil !

ata@WITS>>obtenir les gants
ata: vous avez obtenu les gants

ata@WITS>>mettre les gants
Etape completee avec succes

ata@WITS>>se placer le dos-au-vent
Etape completee avec succes

ata@WITS>>prochaines etapes ?
Essayer une des actions suivantes:

 - METTRE les VERRES-PROTECTEURS

 - VERIFIER MOULE pour OUVERTURES-ELARGIES et AJUSTEMENT-IMPRECIS

 - VERIFIER le MOULE pour des EGRATIGNURES et CRAQUES

 - PROTECGER les CONDUCTEURS et les OUTILS de la PLUIE avec une TOILE en cas de forte pluie

 - NETTOYER le MOULE avec un PINCEAU

 - ENLEVER la CORROSION et OXYDATION des SURFACES avec la MEULE

ata@WITS>>
```

Both the English and French versions of the system were demonstrated to our collaborators from Hydro-Québec and ERICO, using both the keyboard and voice command interfaces. They found the demonstration to be impressive and accurate and were enthusiastic about the ongoing development of the VR interface.

4.9 Monitoring Systems

The training program needs to determine whenever one of the welding actions is performed and whether it has been performed correctly. This can be achieved through the introduction of monitoring agents specific to the different steps of the welding task.

For instance, when the user needs to clean a wire, he simply grabs the appropriate tool (brush, file, grindstone). Then he brings it to the wire and starts cleaning. The cleaning agent knows the following: the trainee holds a brush, he is touching the wire. The agent determines whether the trainee is really trying to brush the wire. It monitors if the brush is going along the length of the wire while continuously touching it. Once the user completes one of the steps of the welding task, the corresponding monitoring agent informs the expert system. The monitoring agent also gives appropriate feedback to the trainee so as to assist and encourage him to complete his task successfully. The monitoring systems are being developed by other members of the McGill VR-Lab and are described in [Kaddoura 1998].

4.10 Performance Evaluation and Future Recommendations

The WITS expert system was designed to satisfy all its requirements, and its implementation closely followed the design approach used. The response times obtained on a SUN Spark ELC were excellent and effectively instantaneous for keyboard and voice command operations. This prototype version of the system still awaits a comprehensive review to confirm its effectiveness as a training environment for welding procedures and standards. However any updates or modifications that may be necessary can easily be implemented at the database file level. The next version of the expert system should support full multi-user team capabilities that would allow collaborative work in performing the training task. This extension should also allow the introduction of a Virtual Teacher in the VE that would be controlled by a real teacher who can then interact on-line with the Trainees. The current design of the expert system took into account these future

enhancements as mentioned in chapter 3.

The VR interface tests were based on users appreciation of the navigation and interaction of the VO for WITS. As seen previously, Tables 4.2 and 4.1, users had preferences for specific devices such as the pedals for the navigation task and the EZglove for the manipulation task. Also users appreciated the presence of gravity in the environment as well as the use of audible feedback. Along with user rating, it is appropriate to present the performance of the VR system in terms of frame updates per second relative to the complexity of the environment, the number of dynamic objects and number of polygons. Tables 4.3 and 4.4 show some test results.

Run No.	No. of Polygons	No. of Objects	Performance (Frames/Second)	Comments
1	6	2	68.5	Terrain
2	24	4	48.5	Terrain + Cube
3	108	4	42.5	Terrain + Screw
4	114	6	28.1	Terrain + Cube + Brush
5	126	6	35	Terrain + Cube + Screw
6	164	6	35.6	Terrain + Cube + Sphere
7	216	8	23.5	Terrain + Cube + Brush + Screw
8	266	8	25	4 tools
9	356	10	20.3	5 tools
10	457	12	18.9	6 tools
11	584	14	14.7	7 tools
12	710	16	13.9	8 tools
13	784	18	11.9	9 tools
14	834	20	11.2	10 tools
15	1048	22	9.9	11 tools
16	1088	24	9.1	12 tools
17	1253	26	8.7	13 tools
18	1440	28	7.9	14 tools
19	1466	30	7.7	15 tools
20	2102	110	5	15 tools + Body
21	2593	139	3.1	All tools + station 003
22	4019	175	1.4	All tools + station 001

Table 4.3: Performance ratings of the graphical simulation

Table 4.3 compares the frames per second performance of the system to the number of objects present in the virtual environment. Special functions were created that permited the addition of objects to the simulation environment on the fly. Thus the tests started with the terrain object which itself is formed of two objects. Later on, tools were added to the terrain such as the screw and brush. The operator virtual body consisting of the hands and torso was then added. Finally, the electrical station with varying complexity

Run No.	Gravity function	Manipulation function	No. of Polygons	No. of Objects	Performance (Frames/Second)	Comments Comments
1	no	no	520	71	13.3	Terrain + Brush
2	yes	yes	520	71	13.3	Terrain + Brush
3	no	no	1258	87	11	Terrain + 8 tools
4	no	yes	1258	87	10	Terrain + 8 tools
5	yes	no	1258	87	7	Terrain + 8 tools
6	no	no	3175	132	1.7	Station001 added
7	no	yes	3175	132	1.7	Station001 added
8	yes	no	3175	132	1.4	Station001 added
9	yes	yes	3175	132	1.4	Station001 added

Table 4.4: Effect of the gravity and manipulation functions on performance

was added to the simulation environment, (station 003 being having much less details than station001). Note that the "No. of Objects" field is always double the number of element present under the "Comments" field. This is due to the fact that each element added has a bounding box object attached to it.

Table 4.4 is similar to Table 4.3 except that it emphasises on the gravity and manipulation functions and shows how they affect the frame per second performance.

The conclusion from the tests results in tables 4.3 and 4.4 is that the bottleneck of the performance is the number of polygons rendered not the procedures running in the simulation loop. If we want to have a real-time responsive system for the more complex environments we definitely need better computer hardware to increase the CPU processing power and the graphical card performance. Apart from the number of polygons we can see from Table 4.4 that some gain could be achieved by improving the gravity algorithm. Looking closely at Table 4.4 we can see that the performance is really not affected when only one tool object is present in runs 1 and 2. However with 8 tools are present, the effect is more noticeable in runs 3 to 9. This observation can be explained by the fact that the gravity and manipulation functions are run for each object tool present in the environment. Thus it would be very advantageous to have a single "agent" monitoring these tools and calling the appropriate task functions only when needed.

Other than the requirement mentioned above, the WITS virtual interface could greatly be improved by including the following features:

- Improved algorithms to control the mapping of the different input devices' data. For example the auto-calibration of the EZglove to track the user's fingers movements with increased resolution. Also, auto-adjustment of the pedals sensitivity should take into account the intention of the user optimize his traveling time. When the user is continuously moving with maximum displacement, the system should increase the steps sizes until the user slows down.

- Additional laws of physics laws other than gravity such as friction, wind and inertia, should be incorporated to add more realism to the VE.

- More sensors should be incorporated to allow the addition of the different human parts. This could also include the addition of force-feedback devices that would permit more precise and realistic manipulations to take place in the VE.

- Monitoring systems are needed to track the user actions and to provide him with feedback and help. These are currently under development.

Chapter 5

Conclusion

This thesis has presented WITS, a Welding Intelligent Tutoring System for training operators to carry out electrical station grounding installations using alumino-thermal welds. The modular design combines a rule based expert system and a 3D virtual reality simulation environment.

The expert system developed allows two basic modes, tutorial and demonstration modes, along with help and evaluation mechanisms included for better self-instruction. The expert system is interfaced with the Virtual Environment which is developed to allow performing the welding procedures in a virtual reality environment.

The expert system uses a rule-based or non-procedural approach in conjunction with object-oriented methods. This allows easy adaptability to changes and updates to the welding procedures when required. The design allows the system to be easily adapted to other similar training courses.

Another achievement of this research was the design of the McGill VO as a stand alone interface that can be introduced in any virtual environment. The McGill VO permits the user to interact and affect the synthetic world in real-time VR applications. This research concentrated on the VO's hands since these are the most dextrous parts of the human body. The hands are necessary for the performance of precise and delicate manipulations such as the welding procedures and other complex tasks. The innate naturalness, adaptability, and dexterity of the hand interface improves the feeling of

immersion in the VR world by adding an important dimension of realism to it.

In addition to the McGill VO, the VE contains intelligent objects that hold some notion about the environment that surrounds them. They can react to events in the world, can be indirectly manipulated, and can permit physics to take part in the virtual environment. In other words, they understand the world that is around them which simplifies most of the tasks involving their manipulation.

The system architecture uses object oriented modular approach which ensures reusability and adaptability to other training problems. Each module was implemented with this approach and thus can be ported to and reused in other projects such as the SEDA-Transform [Tam 1998] which also uses the WITS expert system design methodology.

The overall results are very encouraging. Preliminary evaluations have found that VR interface interactions are very promising and definitely offer a more exciting learning environment compared to a textbook based or a 2D click and drag based training course. The research is continuing in the development of a more comprehensive virtual operator [Kaddoura 1998], [Badra 1998].

References

[Adam 1993] J. A. Adam, *Virtual reality is for real*, IEEE Spectrum Magazine, Vol 30, No 10, October 1993, pp. 22-29.

[Beaten et al. 1987] T. J. Beaten, R. J. DeHoff, N. Welman and P. H. Hildebrandt, *An evaluation of input devices for 3-d computer display workstations*, Proceedings of SPIE - The International Society for Optical Engineering, Vol 761, pp. 94-101.

[Badra 1998] F. Badra, *Real-time immersive articulation of the human body in the WITS virtual training environment*, M.Eng. Thesis, Electrical and Computer Engineering Departement, McGill University, to be submitted July 1998.

[Bagiana 1993] F. Bagiana, *Tomorrow's space: Journey to the virtual worlds*, Computers and Graphics, 1993, Vol 17, No 6, pp. 683-690.

[Beck and al. 1996] J. Beck, M. Stern, E. Haugsjaa, *Applications of AI in education*, ACM Crossroads, Issue 3.1, Fall 1996, ACM Press, New York, NY, pp. 11-15.

[Bejczy and Salisbury, Jr. 1983] A. K. Bejczy, and J. k. Salisbury, Jr., *Controlling remote manipulators through kinesthetic coupling*, Computers in Mechanical Engineering, July 1983, pp. 48-60.

[Brooks 1989] Martin Brooks, *The DataGlove as a man-machine interface for robotics*, The Second IARP Workshop on Medical and Healthcare Robotics, Newcastle upon Tyne, UK September 5-7, 1989, pp. 213-225.

[Bryson et al. 1994] S. Bryson, S. Feiner *Research frontiers in virtual reality*, Proceedings of SIGGRAPH 94, Orlando, Florida, July 24-29, 1994, Computer Graphics Proceedings, Annual Conference Series, 1994, ACM SIGGRAPH, pp. 473-474.

[Buc 1994] N. Buc, D. Paris and T. Izumi, *Simulation of the landing of a re-entry vehicle using eurosim*, ESA Bulletin August 1994, No 79, pp. 27-33.

[Buxton 1990] W. Buxton, *The prgmatics of haptic input*, April 1990, ACM CHI'90 Tutorial Notes 26, Seattle, WA.

[Card 1991] Card, Orson Scott, *Ender's game*, New York, N.Y., T. Doherty Associates, 1991.

[Card , English and Burr 1979] S. K. Card, W. K. English and B. J. Burr, *Evaluation of mouse, rate-controlled isometric joystick, step keys, and text keys for text selection on a CRT*, Ergonomics, 1979, Vol 21, No 8, pp. 601-613.

[Carroll and Campbell 1988] J. M. Carroll, and R. L. Campbell, *Artifacts as psychological theories: The case of human-computer interaction*, RC 13454, #60225, 1988, IBM Research Division, T. J. Watson Research Center, Yorktown Heights, NY.

[Chin and Sheridan 1989] K. P. Chin, and T. B. Sheridan, *The effect of force feedback on teleoperation*, Work with computers: Organizational, management, stress and health aspects, Elsevier Science Publishers B.V., Amsterdam 1989, pp. 505-511.

[Coull 1992] T. Coull, *VR applications: From wall street to rehabilitation*, Wescon Conference Record, 1992, Los Angeles, CA, Vol 36, pp. 399-402.

[DeFanti and Sandin 1977] T. A DeFanti, and D. J. Sandin, *Final report to the national endowment of the arts*, US NEA R60-34-163, University of Illinois at Chicago Circle 1977.

[Delp 1990] Delp *An interactive graphics-based model of the lower extremity to study orthopaedic surgical procedures*, IEEE Transactions on Biomedical Engineering, 37(8), August, 1990, Special issue on interaction with and visualization of biomedical data.

[Durkin 1996] J. Durkin, *Expert systems: A view of the field*, IEEE Expert, Vol 11, No 2, Apr. 1996, IEEE CS Press, Los Alamitos, CA.

[IBM 1998] IBM, *Advances in human language technologies*, An IBM White Paper (11/97).

[Durlach 1989] N. I. Durlach, *Research on reduced-capability human hands*, Proposal to office of Navel Research, M.I.T. Research Laboratory of Electronics, Cambridge, MA 1989.

[Eberl 1994] U. Eberl, *Erstflug im computer*, Dailmer-Benz High Tech Report 1/95, Stuttgart, Germany 1994, pp. 40-47.

[Eglowstein 1990] H. Eglowstein, *Reach out and touch your data*, Byte, July 1990, pp. 283-290.

[Enderle, Kansy and Pfaff 1984] G. Enderle, K. Kansy and G. Pfaff, *Computer graphics programming: GKS - the graphics standard*, Springer-Verlag, 1984, New York.

[Esposito 1993] C. Esposito, *Virtual reality research at BOEING*, Wescon Conference Record, 1992, Los Angeles, Vol 36, pp. 17-22.

[Feiner and Beshers 1990] S. Feiner, and C. Beshers, *Visualizing n-dimensional virtual worlds with n-vision*, Computer Graphics, Proceedings 1990 Symposium on Interactive Graphics, March 1990, pp. 37-38.

[Fels 1990] S. S. Fels, *Building adaptive interfaces with neural networks: The glove-talk pilot study*, Technical Report CRG-TR-90-1, Department of Computer Science, University of Toronto February 1990.

[Fisher et al. 1986] S. S. Fisher, M. McGreevy, J. Humphries and W. Tobinett, *Virtual environment display system*, Proc. 1986 ACM Workshop on Interactive Graphics, Chapel Hill, NC October 23-24, 1986, pp. 77-87.

[Fisher 1989] S. S. Fisher, *Virtual environments, personal simulation & telepresence*, ACM SIGGRAPH '89 Course Notes #29, Implementing and interacting with real-time microworlds, July 31, 1989.

[Fitts 1954] Fitts, P. M., *The information capacity of the human motor system in controlling amplitude of movement*, Journal of Experimental Psychology, 1954, Vol 47, No 6, pp. 381-391.

[Flach 1990] J. M. Flach, *The ecology of human-machine systems I: Introduction*, Ecological Psychology, 1990, Vol 2, No 3, pp. 191-205.

[Foley et al. 1996] J. D. Foley, A. Van Dam, S. K. Feiner, and J. F. Hughes, *Computer Graphics: Principles and Practice*, Addison-Wesley Publishing Company, Inc., 1996.

[Foley and Wallace 1974] J. D. Foley, and V. L. Wallace, *The art of natural graphic man-machine conversation*, Proceedings of the IEEE, April 1974, Vol 62, No 4, pp. 462-471.

[Laliberté et al. 95] X. Troussaut, A. Di Vincenzo, and R. Laliberté, *Cours d'Hydro-Québec de soudage alumino-thermique*, Hydro-Québec, édition mars 1995.

[Ginsberg and Maxwell 1983] C. M. Ginsberg, and D. Maxwell, *Graphical marionette*, Proc. ACM SIGGRAPH/SIGART Workshop on Motion Toronto, Canada, April 1983, pp. 172-179.

[Grimes 1983] G. J. Grimes, *Digital data entry glove interface device*, United States Patent 4,414,537, Bell Telephone Laboratories, Murray Hill, NJ November 8, 1983.

[Hall 1985] J. A. Hall, *The human interface in three dimensional computer art space*, Unpublished MSVS Thesis, Media Lab, Massachusetts Institute of Technology, Cambridge, October 1985.

[Hitchner 1992] L. Hitchner, *The NASA ames virtual planetary exploration testbed*, Wescon Conference Record, 1992, Los Angeles, CA, Vol 36, pp. 376-381.

[Hodges 1992] L.F. Hodges, *Tutorial: Time-multiplexed stereoscopic computer graphics*, IEEE Computer Graphics and Applications, 1992, Vol 12, No 3, pp. 20-30.

[Holzer 1994] R. Holzer, *U.S. debates live fire vs. simulation*, Defense News, U.S., August 1-7, 1994, pp. 22-23.

[Hong and Tan 1989] J. Hong, and X. Tan, *Calibrating a VPL DataGlove for teleoperating the Utah/MIT hand*, Proc. IEEE International Conference on Robotics and Automation, April 1989, San Francisco, CA, pp. 1752-1757.

[Hughes et al. 1989] P. J. Hjughes, D. G. Alciatore, J. T. O'Connor and A. E. Traver, *Construction manipulator operation with "ergosticks"*, Work with computers: Organizational, management, stress and health aspects, Elsevier Science Publishers B.V., Amsterdam 1989, Proceedings of HCI'89, Boston, Mass., pp. 571-578.

[Hunter 1993] I. Hunter, *A teleoperated microsurgical robot and associated virtual environment for eye surgery*, Presence, 1993, Vol 2, No 4, pp 265-280.

[Jacobsen et al. 1986] S. C. Jacobsen, E. k. Iversen, D. F. Knutti, R. T. Johnson and K. B. Biggers, *Design of the Utah/MIT dexterous hand*, Proc. IEEE International Conference on Robotics and Automation, April 1986, San Francisco, CA, pp. 1520-1532.

[Hutchins, Hollan and Norman 1986] E. L. Hutchins, J. D. Hollan and D. A. Norman, *Direct manipulation interfaces*, User centered system design, 1986, Lawrence Erlbaum Associates, Inc., Hillside, NJ, pp. 87-124.

[Jones 1995] C. Jones, *Virtual reality moves mountains with WorldToolKit*, Silicon Graphics World, June 1995, Vol 5, No 6, pp. 7-8.

[Kaddoura 1998] M. K. Kaddoura, *Monitoring human interaction in the WITS virtual reality training environment*, M.Eng Thesis, Electrical and Computer Engineering Departement, McGill University, to be submitted July 98.

[Katz and al. 1996] S. Katz, A. Lesgold, *Towards the design of more effective advisors for learning-by-doing systems*, Proc. Intelligent Tutoring Systems, ITS 96, Montreal, Canada, June 1996, Springer-Verlag, Berlin.

[Kaufman and Yagel 1989] A. Kaufman, and R. Yagel, *Tools for interaction in three dimensions*, Work with computers: Organizational, management, stress and health aspects, Elsevier Science Publishers B.V., Amsterdam 1989, Proceedings of HCI '89, Boston, pp. 469-475.

[Kilpatrick 1976] P. J. Kilpatrick, *The use of kinesthetic supplement in an interactive system*, Unpublished doctoral dissertation, Computer Science Departement, University of North Carolina et Chapel Hill 1976.

[Kleinfeld 1995] R. Kleinfeld, *Stepping through a computer screen, disabled veterans savor freedom*, New York Times, March 12 1995.

[Kelly, Heilbrun and Stacks 1989] K. Kelly, A. Heilbrun and B. Stacks, *An interview with Jaron Lanier: Virtual reality*, Whole Earth Review, Fall 1989, pp. 108-119.

[Kohonen 1984] T. Kohonen, *Self-organization and associative memory*, Springer-Verlag 1984.

[Kozak 1993] J. J. Kozak, P. A. Hancock, E. J. Arthur and S. T. Chrysler, *Transfer of training from virtual reality*, Ergonomics, 1993, vol 36, No 7, pp. 777-784.

[Kramer and Leifer 1989] J. Kramer, and L. Leifer, *The talking glove: An expressive and receptive "verbal" communication aid for the deaf, deaf-blind, and nonvocal*, Department of Electrical Engineering, Stanford University, 1989.

[Krueger 1990] M. W. Krugger, *Artificial reality (2nd ed.)*, Addison-Wesley Reading, MA 1990.

[Laurel 1986] B. K. Laurel, *Interface as mimesis*, User Centered System Design, 1986, Lawrence Erlbaum Associates, Inc., Hillside, NJ, pp. 67-85.

[Machover 1990] T. Machover, *Flora*, Bridge Record Inc, 1990.

[Makower, Parnianpour and Nordin 1990] J. Makower, M. Parnianpour and M. Nordin, *The validity assessment of the dexterous hand master: A linkage system for the measurement of joints in the hand*, Abstracts of the First World Congress of Biomechanics (Volume II), La Jolla, California, September 1990, pp. 338-339.

[Mann and Antonsson 1983] R. W. Mann, and E. K. Antonsson, *Gait analysis-precise, rapid, automatic, 3-d position and orientation kinematics and dynamics*, BULLETIN of the Hospital for Joint Diseases Orthopaedic Institute, XLIII (2), 1983, pp. 137-146.

[Marcus and Curchill 1988] B. A. Marcus, and P. J. Churchill, *Sensing human hand motions for controlling dexterous robots*, The Second Annual Space Operations Automation and Robotics Workshop, held at Wright State University, Sponsored by NASA and USAF, July 20-23, 1988.

[Mason and Salisbury, Jr. 1985] M. T. Mason, and J. D. Salisbury, Jr., *Robot hands and the mechanics of manipulation*, M.I.T. Press, Cambridge, MA 1985.

[Minsky 1980] M. Minsky, *Telepresence*, Omni, June 1980, pp. 45-50.

[Nemire 1994] K. Nemire, A. Burke and R. Jacoby, *Human factors engineering of a virtual laboratory for students with physical disabilities*, Presence, 1994, Vol 3, No 3, pp. 216-226.

[Okapuu-von Veh 96] A. Okapuu-von Veh, A. Shaikh, E. Garant, A. S. Malowany *Design and operation of a virtual reality operator-training system*, Paper 96WM PWRS, IEEE Winter Power Meeting, Baltimore, MD, Jan. 1995, pp. 157-163 and IEEE Transactions on Power Systems, Aug. 1996, Vol 11, No 3, pp. 1585-1591.

[Okapuu-von Veh 96] A. Okapuu-von Veh, *Sound and vision: Audiovisual aspects of a virtual reality personnel training system*, M. Eng. Thesis, McGill University, Montreal, QC, July 1996.

[Ouh-young 1990] M. Ouh-young, *Force display in molecular docking*, Unpublished doctoral dissertation (TR90-004), Department of Computer Science, University of North Carolina at Chapel Hill, February 1990.

[Pao and Speeter 1989] L. Pao, and Thomas H. Speeter, *Transformation of human hand positions for robotic hand control*, Proc. IEEE International Conference on Robotics and Automation, April 1989, San Francisco, CA, pp. 1758-1763.

[Peters 1995] T. M. Peters, *Recent developments in medical imaging*, IEEE Seminar Series, McGill University, Department of Electrical Engineering, March 9, 1995.

[Piantanida 1992] T. Piantanida, *Practical applications of virtual reality*, Wescon Conference Record, 1992, Welson, Los Angeles, CA, Vol 36, pp. 388-396.

[Pieper 1992] S. D. Pieper, *CAPS: Computer-Aided Plastic Surgery*, PhD Thesis, February 1992, Media Lab, Massachusetts Institute of Technology, Cambridge.

[Poizner et al. 1983] H. Poizner, E. S. Klima, U. Bellugi and R. B. Livingston, , Proc. ACM SIGGRAPH/SIGART Workshop on Motion, Toronto, Canada, April 1983, pp. 148-171.

[Purcell 1985] P. Purcell, *Gestural input to interactive systems*, Computer Bulletin, September 1985, pp 3-7.

[Raab et al. 1979] F. H. Raab, E. B. Blood, T. O. Steiner and H. R. Jones, *Magnetic position and orientation tracking system*, IEEE Transactions on Aerospace and Electronic Systems, AES-15, September 1979, pp. 709-718.

[Regian and al. 1997] W. Regian, *Point paper on ICAI*, Armstrong Laboratory, Brooks Air Force Base, April 1997,
"http://www.brooks.af.mil/AL/HR/ICAI/icaitap/icaitap.htm".

[Roos 1995] J. G. Roos, *Is it for real? The rush to trainig simulation*, Armed Forces Journal International, January 1995, pp. 24-26.

[Rheingold 1992] H. Rheingold, *Virtual reality: The revolutionary technology of computer-generated artificial worlds - and how it promises to transform society*, A Touchstone Book, Published by Simon and Schuster, New York 1992.

[Sagar et al 1994] M. A. Sagar, D. Bullivant, G. D. Mallinson, P. Hunter and I. W. Hunter, *A virtual environment and model of the eye for surgical simulation*, Proceedings of SIGGRAPH 94, Orlando, Florida, July 24029, 1994, Computer Graphics Proceedings, Annual Conference Series, 1994, ACM SIGGRAPH, pp. 205-212.

[Selcom] Selcom Inc., Southfield, MI, USA; a division of Selspot, AB, P.O. Box 250, 433 25 Partille, Sweden.

[Shaikh 1998] A. Shaikh, *Alternative manipulation devices and stratefies in a virtual reality operator-training system*, M. Eng. Thesis, McGill University, Montreal, QC, March 1997.

[Shaikh 1995] A. Shaikh, A. Okapuu-von Veh, E. Garant, A. S. Malowany *Alternative manipulation strategies in a virtual reality training system*, Canadian Conference on Electrical and Computer Engineering, Le Centre Sheraton, Montreal, QC, September 1995, pp. 788-792.

[Sheridan 1989] T. B. Sheridan, *Merging mind and machine*, Technology Review, October 1989, pp. 33-40.

[Shneiderman 1983] B. Shneiderman, *Direct manipulation: A step beyond programming languages*, IEEE Computer, August 1983, Vol 16, No 8, pp. 57-69.

[Shute and al. 1990] V. J. Shute, and R. Glaser, *A large-scale evaluation of an intelligent discovery world: Smithtown*, Interactive Learning Environments, Vol 1, No 1, 1990, pp. 51-77.

[Shute and al. 1997] V. J. Shute, and J. Psotka, *Intelligent tutoring systems: past, present, and future*, Handbook of Research on Educational Communications and Technology, Scholastic Publications, 1995,
"http://www.brooks.af.mil/AL/HR/ICAI/its/its.htm".

[Sowizral 1995] H. Sowizral, *Tutorial: An Introduction to Virtual Reality*, Virtual Reality Annual International Symposium, 1995.

[Stanney 1995] K. Stanney, *Realizing the Full Potential of Virtual Reality: Human Factors Issues Than Could Stand in the Way*, IEEE Proceedings of the Virtual Reality Annual International symposium in Research Triangle Park, NC, March 11-15, 1995, pp. 28-34.

[Slavkoff 1997] E. Slavkoff, *Articulating human hands and manipulating objects in virtual environments*, M. Eng. Thesis, McGill University, Montreal, QC, July 1997.

[Stefik 1995] M. Stefik, *Introduction to knowledge systems*, Morgan Kaufmann, San Francisco, CA, 1995.

[Speeter 1989] T. H. Speeter, *Transforming human hand motion for telemanipulation*, Technical Memorandum submitted to IEEE-SMC 10/89, AT&T Bell Laboratories, Holmdel, NJ Sept. 19, 1989.

[Sturman 1994] D.J. Sturman, and D. Zeltzer *A survey of glove-based input*, IEEE Computer Graphics and Applications, Vol 14, Jan. 1994, pp. 30-39.

[Sturman 1992] D.J. Sturman *Whole-hand input*, Doctoral Thesis, Massachesetts Institute of Technology, February 1992.

[Takahashi and Kishino 1990] T. Takahashi, and F. Kishino, *Hand gesture coding based on experiments using a hand gesture interface device*, Technical Report, ATR Communication System Research Laboratories, Kyoto, Japan 1990.

[Takemura, Tomono and Kobayashi 1988] H. Takemura, A. Tomono and Y. Kobayashi, *An evaluation of 3-d object pointing using a fields sequential stereoscopic display*, Proceedings Graphics Interface '88, Edmonton, 1988, pp. 157-163.

[Tam 1996] E. Tam, P. Allard, M. K. Kaddoura, M. Faraj, A. Mourad, and A. S. Malowany *WITS: A reusable architecture for a VR-based ITS*, Workshop Proc., Intelligent Tutoring Systems, ITS 96, Montreal, Canada, June 10-12 1996.

[Tam 1998] E. Tam, *A web-based virtual environment for operator training*, M. Eng. Thesis, McGill University, Montreal, QC, November 1997.

[Thorpe 1987] J. A. Thorpe, *The new technology of large scale simulator networking: Implications for mastering the art of warfighting*, Ninth Interservice Industry Training Systems Conference, 1987.

[Vanderheiden 1994] G. C. Vanderheiden, and J. Mendenhall, *Use of a two-class model to analyze applications and barriers to the use of virtual reality by people with disabilities*, Presence, 1994, Vol 3, No 3, pp. 193-200.

[Vicente and Rasmussen 1990] K. J. Vicente, and J. Rasmussen, *The ecology of human-machine systems II: Mediating "direct perception" in complex work domains*, Ecological Psychology, 1990, Vol 2, No 3, pp. 207-249.

[Ware 1990] C. Ware, *Using the hand position for virtual object placement*, The Visual Computer, 1990, Vol 6, pp. 245-253.

[Ware and Jessome 1988] C. Ware, and D. R. Jessome, *Using the Bat: A six-dimensional mouse for object placement*, IEEE Computer Graphics and Applications 1988, Vol 8, pp. 65-70.

[Weimer and Ganapathy 1989] D. Weimer, and S. K. Ganapathy, *A synthetic visual environment with hand gesturing and voice input*, Proceedings CHI'89 May, Seattle, WA, pp. 235-240.

[Wise et al. 1990] S. Wise, W. Gardner, E. Sabelman, E. Valainis, Y. Wong, K. Glass, J. Drace and J. Rosen, *Evaluation of a fiber optic glove for semi-automated goniometric measurements*, Journal of Rehabilitation research and Development, 1990, pp. 411-424.

[Wixon and Good 1987] D. Wixon, and M. Good, *Interface style and eclecticism: Moving beyond categorical approaches*, Proceedings of the Human Factors Society—31st Annual Meeting, 1987, pp. 571-575.

[Zeltzer, Pieper and Sturman 1989] D. Zeltzer, S. Pieper and D. Sturman, *An integrated graphical simulation platform*, Proceedings Graphics Interface '89, London, Ontario June 1989, pp. 266-274.

[Zimmerman et al. 1987] T. G. Zimmerman, J. Lanier, C. Blanchard, S. Bryson and Y. Harvill, *A hand gesture interface device*, Proc. Human Factors in Computing Systems and Graphics Interface (CHI+GI'87), Toronto, Canada April 1987, pp. 189-192.