

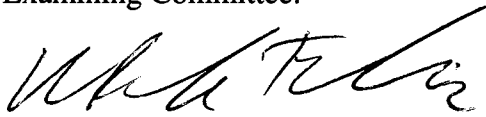
**HAPTIC AND MULTI-MODAL INTERACTION
FOR TEACHING AND DESIGNING BASIC CONTROLS**

by

Linda L. Lim

A Thesis Submitted to the Graduate
Faculty of Rensselaer Polytechnic Institute
in Partial Fulfillment of the
Requirements for the Degree of
DOCTOR OF PHILOSOPHY
Major Subject: Computer Systems Engineering

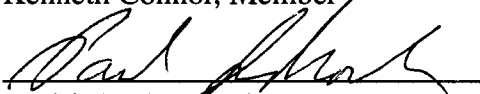
Approved by the
Examining Committee:



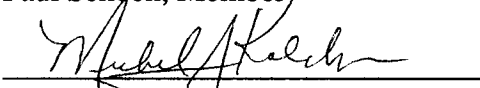
W. Randolph Franklin, Thesis Advisor



Kenneth Connor, Member



Paul Schoch, Member



Michael Kalsher, Member

Rensselaer Polytechnic Institute
Troy, New York

July 2004
(For Graduation August 2004)

UMI Number: 3140953

Copyright 2004 by
Lim, Linda L.

All rights reserved.

INFORMATION TO USERS

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 bleed-through, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send 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.

UMI[®]

UMI Microform 3140953

Copyright 2004 by ProQuest Information and Learning Company.

All rights reserved. This microform edition is protected against
unauthorized copying under Title 17, United States Code.

ProQuest Information and Learning Company
300 North Zeeb Road
P.O. Box 1346
Ann Arbor, MI 48106-1346

Copyright 2004
by
Linda L. Lim
All Rights Reserved

ii

TABLE OF CONTENTS

LIST OF FIGURES	v
Acknowledgements	vii
Abstract	viii
CHAPTER 1 LITERATURE REVIEW	1
1.0 Developments in User Interfaces	2
1.1 History of Haptic Devices	3
1.2 Applications of Haptic Devices	7
1.3 Haptic Devices in Education – Four Examples	11
1.4 Haptics Research	14
1.4.1 Haptic-augmented traditional interfaces	14
1.4.2 Original types of feedback and devices	15
1.4.3 Haptic rendering	16
1.4.4 Haptics for data perceptualization	17
1.4.5 Effective use of haptics	18
CHAPTER 2 HAPTICS	21
2.0 The Sense of Touch	21
2.1 Terminology	22
2.2 Types of Haptic Displays	23
CHAPTER 3 MULTI-MODAL INTERACTION	28
3.0 Human-Computer Interfaces	28
3.1 Research in Human-Computer Interfaces	28
3.2 Multi-modal Interaction	31
3.3 Designing a Multi-modal Interaction System	34
CHAPTER 4 SYSTEM DESIGN AND IMPLEMENTATION	37
4.0 General Description	37
4.1 Hardware Configuration	39
4.2 Haptic Rendering	40
4.3 Designing the haptic feedback:	41
4.4 Software Architecture	44
4.5 Multithreading	45
4.6 Control Algorithm Implementation	48
4.7 Simulation specifics	50
CHAPTER 5 RESEARCH METHODOLOGY	51

5.0	Teaching Control Algorithms	51
5.1	Experimental Method	53
5.1.1	Procedure	54
5.1.2	Design	56
5.1.3	Participants	57
CHAPTER 6 RESULTS AND DISCUSSION		59
6.0	Pre- and Post-assessment Survey Results	59
6.0.1	Demographic Data	59
6.0.2	Quantitative Evaluation	62
6.0.3	Qualitative Evaluation	70
6.1	Discussion	72
6.1.1	Quantitative results	73
6.1.2	Qualitative results	77
6.1.3	Cautions about the internal validity of the experiment:	78
6.2	General observations	79
6.3	Participant Feedback and Suggestions	81
6.4	Fundamental Contributions	82
6.5	Future Research Possibilities	83
6.5.1	Classroom use	83
6.5.2	Research ideas	85
REFERENCES		89
APPENDIX A: Pre-assessment Survey for Final Sample		95
APPENDIX B: Post-assessment Survey for Final Sample Non-Haptic Users		100
APPENDIX C: Post-assessment Survey for Final Sample Haptic Users		105
APPENDIX D: Worksheet for Final Sample Non-Haptic Users		111
APPENDIX E: Worksheet for Final Sample Haptic Users		115
APPENDIX F: Simulation System Code		120

LIST OF FIGURES

FIGURE 1.1: “Hardiman” Exoskeleton	4
FIGURE 1.2: CyberGrasp Force Feedback Glove and Virtual Reality Bodysuit	5
FIGURE 1.3: PHANToM Haptic Display and MouseCAT Force Feedback Display	6
FIGURE 1.4: Laparoscopic surgery force feedback display and sample graphical image.	9
FIGURE 1.5: “Virtual Explorer” Theater	11
FIGURE 2.1: Pin array Tactile Display	24
FIGURE 2.2: Body-based Net Force Displays - Rutgers Master and PERCRO Exoskeleton	25
FIGURE 2.3: Ground-based Net Force Displays - MPB Technologies Freedom Haptic Device, Cybernet Systems 6DoF Hand Controller, and SensAble PHANToM Omni	26
FIGURE 3.1: Handheld Haptic Media Controller	33
FIGURE 3.2: geOrb 3D Controller	33
FIGURE 3.3: User Interaction Model	34
FIGURE 4.1: Screen Shot of the Simulation	37
FIGURE 4.2: Help Graphic of Joystick Controls	38
FIGURE 4.3: Graph of Scaling Function	43
FIGURE 4.4: Graph of Scaled and Unscaled Haptic Feedback Magnitude	44
FIGURE 4.5: Multi-threaded Software Architecture for Multi-modal Display	47
FIGURE 5.1: Testing station	55
FIGURE 6.1: Final Test Group: Participant Ranking of their Education in Controls	60
FIGURE 6.2: Final Sample - Mean Times for Completing Surveys and Worksheet	61

FIGURE 6.3: Pilot Group - Average Change in Score	62
FIGURE 6.4: Pilot Group Pre- and Post-Survey Scores (%)	63
FIGURE 6.5: Final Test Group: Ave Change in Scores	64
FIGURE 6.6: Final Test Group: Individual Pre- and Post-survey Scores (%)	64
FIGURE 6.7: Final Test Group - Ave Change in Concept Question Scores	67
FIGURE 6.8: Final Test Group - Ave Change in Objective Question Scores	68

Acknowledgements

First and foremost, I would like to extend my grateful thanks to my advisor, Professor W. Randolph Franklin, who has been unfailingly supportive and encouraging during this process - without his guidance, patience, and occasional prodding, this research and its completion would not have been possible. My thanks also go to the members of my committee, Professor Ken Connor, Professor Paul Schoch, and Professor Mike Kalsher, whose help, input and encouragement have been invaluable. Thank you to Dr. Grant Deffenbaugh, and soon-to-be doctors Abhijeet Golwelkar and Dean Lewis, who kindly reviewed portions of this research and offered their suggestions and feedback, and to Dr. Marcus Hotaling, whose encouragement and suggestions helped provide structure for completing the work.

My friends and colleagues have been a continuing source of encouragement throughout my years at RPI; thank you to Paul Kulp, Mike Gile, and Professor Frank DiCesare, who first gave me the opportunity to work for LITEC, to Bill Mnich, Steve Dombrowski, Professor Paul Schoch, and Professor Jim Kokernak, whose dedication and vision have continued to make LITEC an innovative and challenging course to be involved in, and to the many LITEC administrators, instructors, and staff with whom I have had the privilege to work during the course of my graduate school career.

Thanks also to the staff and faculty of the Electrical, Computer and Systems Engineering department and of Core Engineering for their instruction, encouragement, financial support, and help in countless large and small ways.

My thanks also go to the many friends I have been privileged to find at RPI and in the Troy area, and especially to the community of Brunswick Church - the continuing love, encouragement and prayers of these friends have been beyond measure.

To my parents, Dr. and Mrs. Eleuterio and Muriel Lim, and my brothers and sisters and family, thank you for your love, your generosity, and your continuing support.

Finally, I give thanks to God, who brought me to this place and these people, and who I firmly believe has been guiding each step along the way.

Abstract

The user interface is the primary means of interaction between a user and the computer. Presently, human-computer interfaces in general use are primarily visual; however, as applications become more complex, the limitations of these kinds of interfaces is becoming more obvious as users become more frustrated with the amount and complexity and difficulty in understanding the visual information being presented. Recent advances in the speed and availability of suitable technology have facilitated the development of multi-modal interfaces - interfaces which deliver and/or receive information to or from the user through more than one sensory mode. Devices which present tactile information are known as haptic displays.

The goal of this research is to demonstrate that a haptic (force-feedback) device can be effective in increasing the depth and quality of students' learning, especially when used in conjunction with other learning tools such as computer simulations. Specifically, we have shown that a learning system consisting of a graphical computer simulation and haptic feedback joystick can be used to allow students to explore different control algorithms and gains, and is more effective than a purely visual simulation.

In the course of this research, we designed and developed a multi-modal computer interface and simulated control system. The simulation is of a car steering automatically to follow a white line on a dark track. The accuracy of the steering is controlled by the selected control algorithm and the settings of the applicable gains. Two versions of the system were developed - one is purely visual, with user controls and choices being made through a menu and mouse system; the other includes the haptic display component which provided the user with augmented information about the steering by delivering force effects in the direction of and proportional to the direction and magnitude of the steering of the simulated car. Visually, the two versions are identical.

Two groups of subjects used the software and took pre- and post-assessment surveys to evaluate their general understanding and learning about basic controls. After evaluating the data provided by the surveys, it was determined that the results support the

hypothesis that a multi-modal interface system, and specifically one with a haptic component, can play a significant role in aiding in the learning process. Moreover, in developing this system, we have demonstrated the feasibility of using easily-available, low-priced components for such a system, including a relatively low-speed computer and an off-the-shelf haptic device. By having the simulation and haptic processing drive the system, and taking advantage of a multi-threading software architecture and embedded microprocessor hardware architecture, we were able to produce the target system with available equipment.

This work opens up possibilities for developing more effective educational tools and methods which will engage the students through multiple senses and is not limited to those with the means to use the latest and fastest technology. Moreover, the opportunity to use this system to design controllers that will not only be used in a computer simulation, but also can be used with a real-world system opens up possibilities for uses beyond academia.

CHAPTER 1: LITERATURE REVIEW

The user interface is the primary means of interaction between a computer and a user. As such, the design of an interface must take into account the goal of delivering information as simply, efficiently, and naturally as possible. In recent history, the “standard” for user interfaces has become the Graphical User Interface (GUI), one which is heavily dependent on visual information delivery. As the limitations of this approach have become more apparent, limitations caused by the human inability to efficiently process information from a single sensory channel beyond a certain capacity, researchers have started looking to the other senses as possible receptors of information to augment or replace information now being delivered visually (Sanders & McCormick, 1993; Wickens, 1992). One possibility being explored is to deliver information using the sense of touch.

Studies have shown that delivering information in more than one sensory mode will often increase retention and understanding. For example, a graph or picture (visual mode) can be used to clarify or illustrate a lecture (audio mode). In general, these studies have focused on the visual and auditory senses (e.g., Liu, 2001; Lee & Bowers, 1997; Davis et al., 1999). However, in the last two decades or so, technological advances have opened up the possibility of delivering information more easily using other sensory modes such as touch.

Devices which present tactile information are known as haptic displays. These displays can be used in varied fields and applications, including training, product development, and education. Except for the fields of medicine and vehicle operation (like flight training), the development of haptic-enhanced tools for education is still in its early stages. We propose that a haptic device can be used in conjunction with a computer-generated simulation to aid and enhance undergraduate student learning about different basic control algorithms, specifically, proportional, proportional + integral (PI), and proportional + integral + derivative (PID). By providing information in a tactile mode as well as visual

mode, the user is provided with reinforced information about the simulation system and the performance of its controller.

1.0 Developments in User Interfaces

In the field of user interface design, one of the major goals has always been to deliver greater amounts of information more quickly and efficiently. Starting from punch cards and moving to text-based interfaces, user interfaces evolved over time into Graphical User Interfaces (GUIs), where information is delivered not one line at a time, but in documents, images, menus, and other constructs designed to convey the maximum usable information to the user. The “WIMP” (Windows, Icons, Mouse, Pointer) type GUI has now become standard for those working with computers, and many experts now believe that the WIMP interface has just about reached the limits of its usefulness (Gentner & Nielson, 1996; Scali et al., 2003; Schwartz, 2001; Van Dam, 1997).

With this conclusion has come the goal to develop new forms of interaction and information delivery. Among the approaches being tried are such innovations as 3-dimensional interfaces (where the user navigates spatially through the applications and files on his or her computer) (Leach et al., 1997), time-organized file systems, gestural interfaces (Van Dam, 1997), and multi-modal interfaces.

“Multi-modal” is a general term used to describe user interfaces that interact with the user using different modes to engage multiple senses. The standard WIMP interface is primarily visual in its information delivery, with some aural component, usually clicks from the mouse, beeps to indicate errors, and other simple interactions. As has been noted by numerous researchers, human beings interact with the natural world using multiple senses in combination with each other, and in fact, it is possible for a person to become overloaded by too much data on one particular “channel” or sense. Many cognitive psychologists have noted how, as more input is presented on the same sensory channel, so the possibility of distraction, confusion, and decreased performance on one or more tasks is increased. This can be observed with many of the software applications available today - with the present emphasis on visual delivery of information, it is not uncommon for a new

user to be overwhelmed and confused just by the initial presentation, let alone a complex dataset or complicated process. By distributing information to the other senses, the information load to a single sense is reduced. Wickens and other proponents of multiple resource theory have demonstrated that “multitask processing can be supported more efficiently by cross-modal presentation than by within-modal presentation” (p.361, Wickens et al., 2003). Moreover, there is potential for users with different sensory strengths to benefit more from this type of interface.

At this time, the most ambitious (and complex) form of multi-modal interface is the virtual reality or immersive interface. This type of interface attempts to allow the user to experience the computer application or setting in the same way that he or she experiences the real world, through multi-sensory input, movement, and manipulation. Because of the technical requirements for providing such an experience, the applications and settings where these types of interfaces are used are extremely limited, usually requiring large, complex, and expensive graphical display systems, specialized haptic hardware, and dedicated space. In addition, problems such as VR-sickness, limited user mobility, and sensory latencies have yet to be solved (Fowlkes et al., 2002).

1.1 History of Haptic Devices

Haptic displays were developed to provide tactile information that would otherwise be unavailable because of physical danger, distance, expense, problems of scale, and other difficulties.

The origin of haptic devices actually dates back to the master arms used in the 1950s for remote handling of radioactive materials. Initially just passive replicas of the slave arm, master arms were manipulated by the operator to command the slave arm performing the task. Later on, master arms were motorized so that they could impart feedback forces on the operator that were proportional to the forces being felt by the slave arm operating in the remote environment...(Carignan & Cleary, 2000).

From those origins in the 50s came a community of researchers and companies that, through the following decades, developed remote handling devices for undersea,

space, and hazardous environment use. Mechanical linkages were replaced by cable linkages and servos, allowing for more flexible and compact remote handling. Force feedback became adjustable, allowing the user some measure of control over his or her experience with the device. It was these types of devices that gave rise to the ideas of “teleoperation - the extension of a person’s sensing and manipulation capability to a remote location” and “telepresence - the ideal of sensing sufficient information about the teleoperator and task environment, and communicating this to the human operator in a sufficiently natural way, that the operator feels physically present at the remote site” (p.1, Stone, 2000).

The 60s saw the development of the first exoskeleton devices - like the master-slave arms of the 50s, the slave portion of the device replicates (and sometimes amplifies) the master manipulations; however, with exoskeletons, the master is attached directly to the human user, and the replicated motion is a more exact copy of the user’s motions. Examples of these include the “Handyman” controller, a forearm-and-hand exoskeletal device, and the “Hardiman” by General Electric, which was a full-body exoskeleton.



FIGURE 1.1: “Hardiman” Exoskeleton

While the weight and movement limitations of exoskeletal devices severely limited their usefulness, recent years have seen a resurgence of interest in exoskeletons, especially for virtual reality-type interfaces, due to the development of smaller, light-weight components and faster computers.

As computers became more powerful, complex kinematics computations

could be performed in real-time enabling the development of master arms which bore little resemblance[sic] to the slave arms they were controlling... It was only a matter of time before researchers began to realize that this new breed of hand controllers could be used for simulating virtual environments as well as reproducing the forces sensed in a “real” environment (Carignan & Cleary, 2000).

Through the 80s and 90s, the development of numerous and varied haptic devices continued, with varying degrees of success. These have included flexible gloves with pneumatic feedback like the Teletact series, rigid hand exoskeletons like the CyberGrasp and the Rutgers Master, larger and full-body interface devices like the FREFLEX and the Virtual Reality Bodysuit, point-based systems like the PHANToM, and variations on standard haptic devices such as haptic mice and joysticks (Nahvi & Hollerbach, 2000).

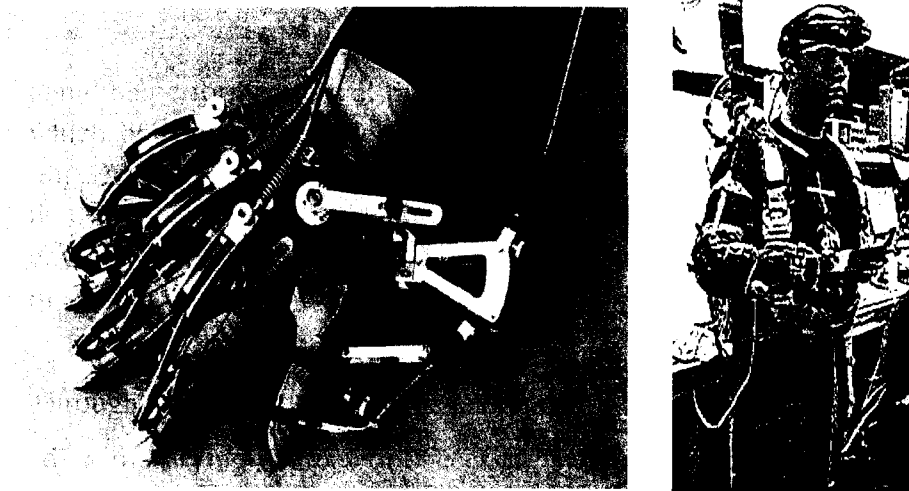


FIGURE 1.2: CyberGrasp Force Feedback Glove and Virtual Reality Bodysuit

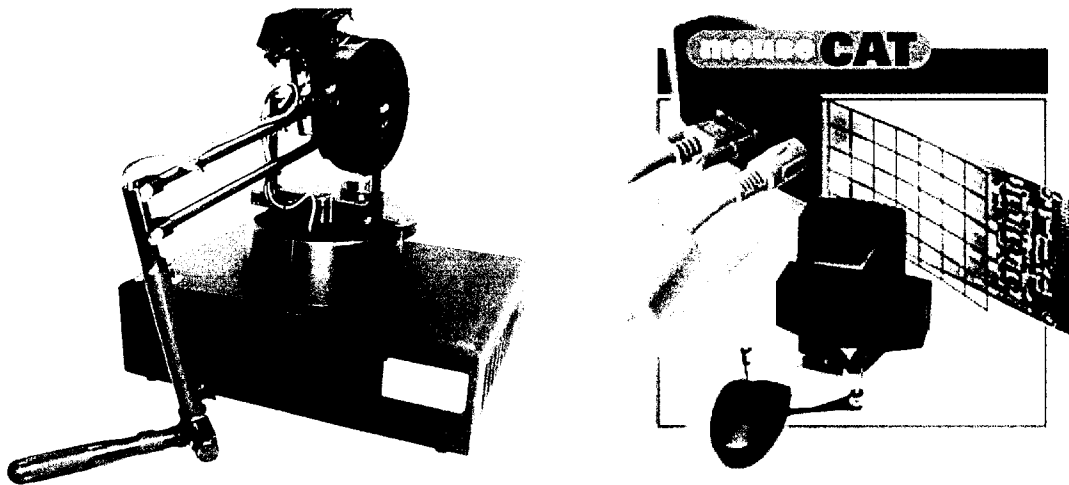


FIGURE 1.3: PHANTOM Haptic Display and MouseCAT Force Feedback Display

As computers have become increasingly present in our society, so too has interest in using them for tasks that have traditionally involved hands-on manipulations, human-to-human interactions, travel to remote sites, or other “real world” experiences. This interest has driven the development of various “virtual” components - virtual tools, virtual desktops, virtual environments, and of course, virtual reality. However, the potential for the virtual experience to be as “real” as possible is hampered if all the senses are not engaged by the experience. Thus, a flight simulator where the control stick does not exert force against the pilot’s hand as it would in a real plane will leave the student unprepared for the actual experience. Similarly, a computer simulation of a surgical procedure must give the user a realistic experience of cutting into flesh, suturing, etc. if it is to be more than just a game.

Once the use of tactile feedback to enhance the realism of a computer-generated experience or device had been established, it was only a small step to start considering what other uses might be made of the technology, uses that might not necessarily directly represent a real-world task or situation, but which could be used to increase a user’s knowledge or experience in useful ways. The question became, are there ways in which

haptic feedback could be used to increase users' understanding of complex information or procedures?

1.2 Applications of Haptic Devices

In the last few years, the spectrum of uses of haptic feedback technology has increased dramatically. One European initiative is focusing on developing a system which trains its users in aircraft maintenance - a haptic device is used to control the actions of a virtual worker as it performs preparation procedures and landing gear testing. A system developed for the French army uses a PHANTOM haptic display to train soldiers in land-mine clearance, where the user must probe the "ground" to determine the configuration of the virtual land mine. This system also provides expert help by displaying possible types of mines based on the data generated by the soldier (Stone, 2000).

Several companies are using haptic technology for product development – artisans at ceramics companies such as Wedgwood have been able to use haptic tools for virtual prototype development, and rapid output of actual prototypes (Stone, 2000). Researchers have developed virtual hand tools that can be used to carve shapes from blocks of virtual material (Balakrishnan et al., 1994); Sanjay Sarma at MIT is working on a haptic feedback device for use with computerized milling machines (Thilmany, 2000).

George Fitzmaurice of the University of Toronto Computer Science Department has done doctoral research working with "graspable user interfaces" (Fitzmaurice, 1996) – interface components which provide a more intuitive method for manipulating virtual objects than a mouse or keyboard. Among other items, he uses small blocks to move and control graphical shapes on the computer, and has determined that these interactions are more efficient than the multiple mouse-clicks and dragging that usually are required when using graphics applications.

At the University of North Carolina, researchers are attempting to use haptic devices to recreate the feel of painting using different size and texture brushes (Mahoney, 2001). Engineers for BMW have developed a mouse for use in the company's cars – in an attempt to keep the interactions simple and non-distracting visually, they are relying on

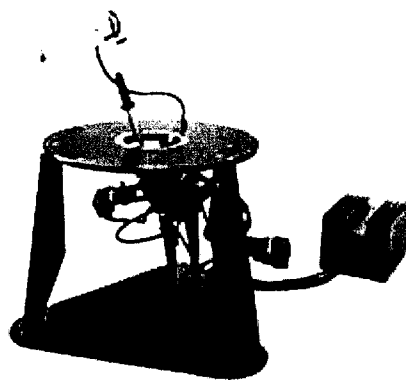
tactile cues to inform the user of his or her location or interaction with components such as menus and lists (Sharke, 2001).

One haptics application that could have implications for a large percentage of the population is its use in “drive by wire” cars. Car companies such as DaimlerChrysler and General Motors are developing cars where mechanical systems such as the throttle control or braking are replaced with sensors and electronic signals to activate the desired operation or component (Gizmo, 2004; Harris, 2004). As the mechanical linkages are no longer present to provide the drive with the “feel” of the driving process, one of the challenges in developing these cars is to provide the drive with useful tactile information about the vehicle, the operation being performed, and the driving conditions. Thus, researchers at Clemson University (Setlur, 2002) and at Aalborg University in Denmark (Izadi-Zamanabadi) are experimenting with haptic devices for drive by wire applications, attempting to design feedback that is both informative and comfortable for the user, and also to make the feedback system more fault-tolerant. In addition, the Alps Electric Co., Ltd. has developed a line of haptic components for use in drive by wire systems, including a haptic steering wheel, shift stick, and pedal (Alps, 2004).

In the field of education, the primary uses of haptic technology have been flight training and medical simulation. Haptic feedback has long been an important aspect of flight simulators used for teaching pilots. Within the last decade or so, as graphics and haptic technology have become smaller, faster, and less expensive, medical simulations have begun making use of tactile feedback used in conjunction with anatomy simulations, which allow medical students and doctors to practice operations and procedures without using cadavers and test animals, and before risking the health of a live patient (Sharke, 2001). Developments in minimally invasive and endoscopic surgery have particularly encouraged the development of surgical simulations. These procedures are generally performed without the surgeon being able to directly observe the surgical site of his or her work, but rather using a tiny camera mounted on a probe (the endoscope) inserted through a small incision into the body, working with instruments through similar incisions, and watching the work on a monitor. Because the surgeon is receiving his or her feedback

through tools and watching the procedure on a screen, these types of surgeries already bear a similarity to a computer-generated simulation; what is needed is for the simulation to *feel* like the actual procedure.

The Immersion Corporation's Medical group began marketing simulators for needle and catheter insertion, and endoscopic procedures in the late 1990's - these included tactile feedback devices combined with realistic computer images to train students in these techniques (Immersion Medical, 2002).



**Laparoscopic Impulse Engine
Force Feedback Surgical
Simulation Tool**



FIGURE 1.4: Laparoscopic surgery force feedback display and sample graphical image.

Haptic displays have also been finding use in allowing the sight-impaired greater access to computers. As computer interfaces have become more graphically (and therefore visually) oriented, blind and low-vision users have been less able to make use of these interfaces. The addition of tactile information can give these users greater ability to interact with their computers, by giving a haptic cue when the mouse pointer moves over the edge of a window or is over a button or other object. Moreover, research has been done in using haptic displays to aid visually-impaired users in experiencing data and simulations for educational purposes. Researchers at Oregon State University, working with colleagues at Immersion Corporation have explored the use of a force-feedback mouse combined with a computer simulation to help blind students understand and experience electric fields. This system allowed students to feel the effects of an electric charge on the surface of a sphere, gather and chart data about the field, and feel the mouse move from

data point to data point on the chart, to gain a better understanding of the phenomenon (Wies et al., 2000).

More recently, haptic devices have started finding their way into less-specialized classrooms. Researchers at Ohio University, working in conjunction with NASA's Langley Research Center, have developed a series of physics tutorials for high school physics students – these tutorials allow the students to experience different levels of friction, spring forces, magnetic forces, and other simple demonstrations using a force-feedback joystick (Williams et al., 2000). Project GROPE at the University of North Carolina is an application of haptic technology to scientific visualization – users are able to feel force fields through the haptic device, and perform complex (virtual) molecular dockings (Brooks et al., 1990). At Stanford University, instructors in a dynamic systems course are using simple haptic devices to demonstrate pendulum motion, damping forces, second order systems, and other course topics (Richard et al., 2000).

A research group at the University of California, San Diego has developed the “Virtual Explorer”, a multi-sensory virtual environment for teaching about immunology (Figure 1.3). Conceptually similar to the movie “Fantastic Voyage”, the users pilot a virtual craft through the human body, observing the workings of the immune system, and making decisions that will affect those workings. A force-feedback flightstick is used for controlling the craft, and environmental conditions and events such as viscosity, speed, and collisions are reflected to the user through the joystick (Dean et al., 2000). The developers note that, “Many topics in science education involve processes that occur simultaneously on multiple time and length scales that are difficult to accurately represent, perceive, and visualize with traditional static media.... we are convinced that properly implemented virtual environments can serve as valuable supplemental teaching and learning resources to augment and reinforce traditional methods” (p.1, Dean et al., 2000).

The potential for haptic devices to contribute to the educational process is still in the early stages of exploration for most subject areas. As the technology continues to develop, and as educators become more comfortable and familiar with the tools available to them, there is little doubt that haptic feedback will be shown to be increasingly useful in



FIGURE 1.5: “Virtual Explorer” Theater

delivering information to students and aiding in the learning process. As one study participant at Oregon State University noted, “I can’t even begin to enumerate the possible applications, but I can see this technology being valuable across a wide range of disciplines and to students and professionals with a range of learning styles and capacities... The possibilities seem almost endless...” (p.110, Wies et al., 2000).

1.3 Haptic Devices in Education – Four Examples

1) Demonstrating electrical fields using a force feedback mouse

Goal: Web-based touch display for visually-impaired students

Overview: Using a simulation of an electrically-charged sphere and a test-charge, students can experience the attracting or repelling forces through the force feedback mouse. The mouse can also be used to help the students experience test data – it can be set to be attracted to data points on a graph, or to move from point to point.

Test users: 4 visually-impaired students and experts

Results: “All evaluators were quite enthusiastic about the force feedback aspects of the curriculum.” (p.110)

Comments:

- Had little or no visual component
- Had no quantitative testing of effectiveness
- Shows the viability of using a commercially available and low-cost haptic device
- Shows that a haptic device can be useful even without a visual component

Researchers: Evan F. Wies, John A. Gardner, M. Sile O'Modhrain, Christopher J. Hasser, Vladimir L. Bulatov, Immersion Corporation and Oregon State University

Reference: Wies, Evan F., Gardner, John A., O'Modhrain, M. Sile, Hasser, Christopher J., Bulatov, Vladimir L., "Web-Based Touch Display for Accessible Science Education", *Workshop on Haptic Human-Computer Interaction*, pp108-112, September 2000.

2) Haptics-Augmented Physics Simulations

Goal: Demonstrate certain physics principles to high school students

Overview: Using a force-reflecting joystick and computer-generated simulations, students can experience spring forces, particle motion, and other physics demonstrations. The haptics software activities are used in addition to a series of HTML tutorials.

Test users: 34 high school physics students from 3 classes, 26 responded to the survey

Results: All students rated the combined tutorials and haptics activities as "effective" or "somewhat effective". Students were not asked to rate the haptics activities alone.

Comments:

- The haptics were used primarily for simple experiential information (i.e., feeling different types of friction) or for interesting effects (i.e., gun recoil for the particle motion demo)
- Had no quantitative testing of effectiveness
- Demonstrates the usability of a commercially available joystick used in conjunction with graphical computer simulations

Researchers: Robert L. Williams and Meng-Yun Chen, Ohio University; Jeffrey M. Seaton, NASA Langley Research Center

Reference: <http://www.ent.ohiou.edu/~bobw/html/NASAHap/HapticsWeb/indexf.htm> (Retrieved from the Internet on April 4, 2002)

3) Using a force-feedback joystick to teach dynamic systems

Goal: To allow students to feel and observe the effects of damping, inertia, and other dynamic systems topics

Overview: Students build a 1 degree-of-freedom haptic device and use it, along

with control software, to experience and observe topics such as pendulum motion, inertia, and feedback control

Test users: ~60 undergraduate Mechanical Engineering students in a dynamic systems course, working in groups of 3-4

Results: Generally positive responses from the students – scores for the different labs ranged from 2.9-4.4 on a scale of 1-5. Researchers observed that the haptic devices, “helped students grasp concepts that had previously been inaccessible” (p.348, Okamura et al., 2002).

Comments:

- Most exercises had no visual component beyond the movement of the paddle (haptic device)
- Had no quantitative testing of effectiveness

Researchers: C. Richard, A. Okamura, M. Cutkosky – Center for Design Research, Stanford

Reference: Richard, C., Okamura, A., and Cutkosky, M., “Feeling is Believing: Using a Force-Feedback Joystick to Teach Dynamic Systems”, *Proceedings of the 2000 ASEE Annual Conference and Exposition*, Session 3668, pp1-15.

4) Virtual Explorer – an immersive virtual environment for education

Goal: To demonstrate the applicability of virtual reality to education

Overview: Using a small theater setup, students navigate a virtual “nanobot” through a simulation of a human body which has been infected by an unknown pathogen. The students learn about immunology while attempting to complete the mission of generating a successful immune response. A force feedback joystick is used for navigation and throttle adjustment, as well as giving information on environmental viscosity and craft speed and acceleration.

Test users: 7000+ visitors have experienced the demonstration

Results: “Very positive” responses

Comments:

- Very expensive and space intensive – uses a 4-processor SGI Power Onyx, 3 52” rear-projection television screens, “small theatre” seating, etc.
- Very complex system for users
- No quantitative evaluation of effectiveness

Researchers: Kevin L. Dean, Xylar S. Asay-Davis, Evan M. Finn, Time Foley, Jeremy A. Friesner, Yo Imai, Bret J. Naylor, Sarah R. Wustner, Scott S. Fisher, Kent R. Wilson – Senses Bureau, University of California, San Diego

Reference: Dean, Kevin L., Asay-Davis, Xylar S., Finn, Evan M., Foley, Tim, Friesner, Jeremy A., Imai, Yo, Naylor, Bret J., Wustner, Sarah R., Fisher, Scott S., Wilson, Kent R., “Virtual Explorer: Interactive Virtual Environment for

1.4 Haptics Research

As has already been stated, the immediate goal of haptics research is often not so much to replace the traditional visually-weighted interfaces, but to improve them, enhance their usability, and make them more intuitive and effective for the users. Most researchers in the field share the belief that, "the haptic modality complements the visual channel" and "output to the hand and fingers will reduce cognitive load" (Munch & Dillmann, 1997). As Lawrence et al. have noted, "such an interface works best when the haptic component is treated as a synergistic companion to visual display. That is, not to replace the visual display of a data set, but to augment this display with reinforcing or disambiguating information through the user's hands" (Lawrence et al., 2000).

1.4.1 Haptic-augmented traditional interfaces

In the attempt to make haptic information a cohesive part of the user interface, many researchers and developers have started with the traditional WIMP (Windows, Icon, Mouse, Pointer) interface and augmented or added to it in subtle or large ways. Hughes and Forrest (1996) have modified traditional computer mice to create several specialized haptic mice which provided vibrotactile feedback using speaker coils and magnets. They have experimented with using that feedback to aid users in finding a particular object in a field of similar objects, and also designed a Vibro Map application, which would indicate the amount of vegetation in an area by changing the amount of vibration as the user moved the cursor around on a computer-displayed map.

Stefan Munch and Rudiger Dillmann at the University of Karlsruhe also modified a traditional mouse to build a specialized ForceMouse haptic interface - they attached two electromagnets to the base and inserted a movable pin in the left mouse button. The electromagnets are used to aid the user in navigating quickly to and stopping on the expected target, and the pin gives feedback about structure and texture, reinforcing the graphic components of the interface. Their application also gathers information about the user and

develops an intelligent model to predict the next target for the mouse (Munch & Dillmann, 1997).

Campbell and his colleagues at IBM's Almaden Research Center are attempting to add vibrotactile feedback to the user interface by way of a keyboard's Trackpoint, an IBM in-keyboard pointing device (Campbell et al., 1999). The modified Trackpoint (called the Tractile) can be made to vibrate at up to 30 Hz, and has been used to try to simulate textures in a virtual environment. Their work has also supported the importance of coordinating the haptic and visual feedback.

Miller and Zeleznik at Brown University are working on possibilities for adding haptic feedback to the desktop aspects of an interface. Thus, they have added virtual ridges around menu items and icons and other enhancements to a windows-type interface. The haptic display used for this device is the SensAble Technologies PHANToM (Miller & Zeleznik, 1998).

1.4.2 Original types of feedback and devices

Until relatively recently, the commercial options for haptic devices were extremely limited, and even now, a large number of those working in haptics research prefer to design and build their own specialized devices. By doing so, the developer has greater control over the use and usability of the device, and he or she is able to be much more creative and original in designing the interaction.

Vincent Hayward at McGill University is studying ways to use haptics as part of a car's instrument panel. His work involves using small pins to stretch the skin and produce sensation. He hopes to eventually be able to simulate different textures using this technology (Graham-Rowe, 2001).

Sklar and Sarter have obtained encouraging results using haptic notification of unexpected changes in a flight simulator. A wristband provided vibration to the inside or outside of the user's wrist, depending on the type of transition being signalled. They found that reaction times and accuracy were significantly improved when the haptic notification was used, as opposed to a purely visual notification (Sklar & Sarter, 1999).

Verplank et al. (2002) at Stanford University have built a haptic device they call “The Plank” from old disk drive motors and controlled by an embedded microcontroller. They have experimented with using the Plank to manipulate wave shapes to produce varied audio output. This capability also allows the user to experience a virtual terrain. Their eventual goals include using the interface to communicate with a synthesizer to produce music, and reproducing the feel of traditional musical instruments.

Hikiji and Hashimoto (2000) at Waseda University are doing experiments to develop haptically-based human-robotic interfaces. They have developed an autonomous robot that can lead or follow a human using hand-to-hand force interactions, that is, holding hands. When following the human, the robot has force sensors to measure the amount and direction of force being exerted; when leading, the robot is programmed to exert low-level force, and also to stop if the human exerts a significant amount of force to stop or in a different direction.

Snibbe and MacLean and their colleagues have done extensive work in developing unique prototype haptic devices to be used in media control, trying to create intuitive and effective methodologies for scrolling through, playing, and otherwise using and manipulating various media, such as video, audio and graphics. They have experimented with such ideas as a “haptic clutch”, which allows the user to feel the slipping and movement of a virtual wheel through a physical wheel, to represent the progression of a media stream, and a “haptic fisheye”, which allows the user to change the haptic resolution by increasing or decreasing the pressure on a haptic device control (MacLean et al., 2002; Snibbe et al., 2001).

1.4.3 Haptic rendering

Research in haptics and haptic interfaces is not limited to building new devices and/ or finding new applications for them. Along with these developments comes the problem of how best to model and implement the haptic display, both in and of itself, and in conjunction with visual and other displays. The process of determining, calculating, and delivering haptic information is known as “haptic rendering” (see Section 4.2). Like graphic rendering, the challenge is to deliver the effect in a timely manner, in conjunction

with the rest of the displays, and utilizing the computing resources efficiently. Also like graphic rendering, those doing research in the field are experimenting with various models, simplifications, algorithms, and architectures to achieve these goals.

Zilles and Salisbury at MIT have developed an algorithm to aid in haptic rendering called the “god-object” method, which simplifies the process of producing realistic haptic interactions with virtual rigid objects modeled as polyhedra. The god-object is a virtual representation of the haptic interface which is part of the virtual environment, and by being forced to conform to the physical laws of that environment, helps in generating convincing haptic feedback. This work uses a point-based haptic device, much like the PHANTOM (Zilles & Salisbury, 1995).

Munch and Dillmann have structured their system as a multi-agent system, and use a Parallel Virtual Machine (PVM) for interprocess communication. Their eventual goal is to make their system “plug-and-play” for applications using the same operating system and graphics software (Munch & Dillmann, 1997).

Mark et al. (1996) at the University of North Carolina at Chapel Hill have created a software library to support force feedback devices over ethernet and other TCP/IP networks - it supports several of the PHANTOM series of haptic devices, and the Sarcos Dexterous Master. For their work, they have decoupled the haptic processing from the graphics processing, using two different machines specialized for their particular tasks.

Balaniuk and Laugier (2000) have also taken the approach of uncoupling the haptic and graphics processing, using a software architecture that includes a “buffer model” between the haptic device and the virtual environment. The buffer model is a simplified model of the virtual environment, and the haptic device uses this in calculating contact forces, collisions, and other haptic interactions.

1.4.4 Haptics for data perceptualization

While the idea of using haptic displays for applications such as virtual reality or immersive experiences is a fairly obvious one, several researchers are exploring the use of haptic technology to display information that does not necessarily have a definite physical form. Much like graphic displays can aid the user in understanding complex data, so

researchers are looking for ways that haptic displays can be used for data visualization, or more accurately, perceptualization. In some of these systems, the haptic display provides redundant or reinforcing information coordinated with that of the visual display; in others, the goal is for some portion or aspect of the data to be displayed only in haptic form.

Lawrence et al. (2000) at the University of Colorado have developed a system for working with fluid dynamics data, specifically shock surfaces and vortex cores. Their system includes a graphic display and a 5 DOF stylus-based haptic device which can be used as a 3D mouse. When the desired mode is activated, the stylus will also be moved automatically to indicate a chosen vector quantity, with the magnitude of the force increasing as the user gets closer to a vortex. Because of the size and complexity of the data sets being used and the graphic and haptic displays, two separate machines are used for the graphics and haptic processing and display, with a communication link to coordinate the two displays.

Nesbitt et al. (2001) have investigated the use of a virtual environment, including haptic interactions through a PHANTOM device, for multi-sensory exploration of data consisting of a “multivariate mathematical model of fluid flow and temperature within a blast furnace” (p.1, Nesbitt et al., 2001). The haptic feedback was developed to allow the user to interact with vector flow fields while the visual display showed temperatures, flow magnitudes, and other components. While their work demonstrated that such data exploration was feasible, their preliminary results indicated that the multi-sensory experience did not dramatically improve the user’s understanding or interpretation of the data.

1.4.5 Effective use of haptics

As haptic technology has improved, and as possibilities for working with different applications and interfaces have increased, more research can be performed addressing the questions of how best to use this modality, whether in fact it is useful and effective, and under what conditions or for what applications.

Fowlkes et al. (2002) have been conducting experiments for the U.S. Army to determine the effectiveness of haptic feedback, particularly in terms of training systems. Their study investigated the level of detail necessary in representing a virtual hand graphi-

cally, whether haptic or audio feedback were equally effective in using the application, and how haptic feedback might effect the perception of immersion.

In Campbell and his colleagues' work with the Tractile, 16 users were asked to navigate a cursor through a virtual tunnel using the Tractile, under four different conditions: Visual Only, Visual + Tactile, Botts Dots, and Unconcerted Visual + Tactile. The contours of the tunnel were indicated by bumps, either visual or tactile or both. Pairwise t-tests showed that the mean completion time was significantly shorter with the Visual + Tactile condition, with no significant difference between the other three conditions. However, the error rate was significantly lower under the Botts condition, with the other three conditions giving no significant difference (Campbell et al., 1999).

Nyarko et al. (2002) at Morgan State University performed experimental testing with their Network Intrusion Visualization software. They tested with ten subjects, five of whom used a visual+haptic version of the system, and five who used the visual-only version. Their performance measures were the time to detect specific attacks, and the accuracy of detection. Their results show a small improvement in detection speed with the haptic feedback, and a larger improvement in accuracy. No figures were given to address the statistical significance of these findings.

Hughes and Forrest at the University of East Anglia have conducted experiments addressing the use of touch to deliver information by way of a vibrotactile mouse. Twenty-two subjects used both a regular mouse and the haptic mouse to try to identify one different object from a field of similar and otherwise identical objects (needle in the haystack problem). Their findings show that on average the time to find the desired object using the vibrotactile device was 30% of the time needed when using a conventional mouse. Their experiments with a "Vibro-Map", where a haptic effect was used to display data such as the amount of vegetation, had mixed success, and required a tactile scale and training for the users to be able to use it effectively. No statistical analysis on the results of this experiment were included.

In their experiments using a wrist-mounted vibrotactile device to notify airplane pilots of unexpected changes, Sklar and Sarter performed a between-subjects experiment

with 21 pilots randomly assigned to one of three groups that received visual, tactile, or visual+tactile feedback. After a training session including practice flights in the simulator, participants flew a simulated flight during which they had to monitor the usual aircraft changes and displays as well as the unexpected transitions. Reaction time and accuracy of transition identification were recorded, and statistical analysis of the data showed a significant effect for the feedback which incorporated haptic elements.

As can be seen from this overview of the research and uses to which haptic tools are being applied, this field of study both shows great potential, and presents great challenges to those desiring to use it effectively. Like the early computer interfaces, some of the present day devices and applications are unwieldy and nonintuitive, slow and confusing. Yet, as the technology continues to improve and become more available, we can expect to see haptic displays play an increasing role in improving the quality and usability of human-computer interfaces.

CHAPTER 2: HAPTICS

That the sense of touch is an important and useful source of information is inarguable. From the carpenter who checks his sanding job by running his fingers over the wood, to the driver who can feel the pulling of an under-inflated tire, to the small aircraft pilot who flies, almost literally, by the seat of his pants, we depend on our sense of touch to easily and quickly give us information we could otherwise acquire only with great difficulty, if at all. It is one of the primary ways in which human beings interact with and make sense of the physical world.

Research has shown that subjects can judge different thicknesses of paper more finely and accurately with their fingers than with their sight. Subjects differentiated between papers whose thickness varied only by .02 - .04mm, where visual differentiation was difficult for differences of less than .2mm. Similarly, people's sense of touch is significantly better at detecting certain stimuli, such as vibration, than their eyes (Sections 31 and 41, Katz, 1989). As many have noted, even our language subtly hints at the importance of tactile experience – we speak of “getting a feel for” a subject, “grasping” an idea, and of course, every employer prefers employees with “hands-on” experience (p.xi, Burdea, 1996).

2.0 The Sense of Touch

The tactile sense includes the following characteristics:

1. *Bidirectional* – The sense of touch is used both to deliver and to receive information. It is “physically and neurally co-located and coordinated with motor functions” (p.1, MacLean, 2000), both voluntary and reflex. Tactile exploration or communication usually involves physical movement, as well as a much higher degree of two-way information exchange than with the other four senses.

2. *Intentional* – Touching generally requires motivation and deliberate action; culturally, it often requires permission, and can be declined if it is perceived as possibly uncomfortable.
3. *Multi-parametered* – Tactile perception includes many components, including texture, temperature, moisture, force, and pressure. Many tactile descriptors encompass many components – for example, “fuzzy”, “slimy”, or “prickly”.
4. *Low absolute association* – The sense of touch is very useful for detecting subtle differences, but does not work well for identifying specific categories or classes except by comparison.

Tactile information is combined with input from the other senses to form an overall picture of the system, environment, or situation. The sense of touch may be used for the following types of tasks:

1. *Assessment* – Touch is used to evaluate an object’s properties, such as weight, bulk, and texture.
2. *Continuous monitoring of activity* – For example, the vibration of a motor running can be felt even in the absence of other sensory information.
3. *Building mental models of unseen processes and activities* – For example, feeling the door of a closed room if one suspects there may be a fire on the other side.
4. *Verification of start or completion* – For example, the feel of a deadbolt sliding into place.

There are also social implications of touching, both interpersonal and person-to-object, but we will not explore that area here.

2.1 Terminology

Tactile feedback: Sensation applied to the skin, typically in response to

contact or other actions in a virtual world...

Force feedback: The sensation of weight or resistance in a virtual world. Force feedback requires a device which produces a force on the body equivalent (or scaled) to that of a real object. It allows a person in cyberspace to feel the weight of virtual objects, or the resistance to motion that they create. [CyberEdge Journal, 1993]...

Haptic feedback: From the Greek *haptesthai*, meaning to touch, is synonymous with tactile feedback [Webster, 1985]. This author and others extend its meaning to that of force feedback.

Kinesthetic feedback: Synonymous with proprioception, it refers to kinaesthesia, a sense mediated by end organs located in muscles, tendons, and joints and stimulated by bodily movements and tensions [Webster, 1985]...

Proprioceptive feedback: Relates to stimuli arising within the organism. It provides information related to body posture and is based on receptors located at the skeletal joints, in the inner ear, and on impulses from the central nervous system...

(pp.3-4, Burdea, 1996)

2.2 Types of Haptic Displays

Most haptic display devices can be grouped into two categories: tactile displays and net force displays. A tactile display attempts to recreate the feel of an object or sensation as it contacts the user's skin; such sensations may include texture, temperature, slipperiness, etc. Tactile displays are still in the early stages of development – most are only capable of reproducing one type of sensation, rather than anything approaching the full range of sensations that human skin is able to experience.

Surface texture and limited shape information can be delivered by devices using pneumatic stimulation, vibrotactile stimulation, micro-pins, or electrotactile stimulation. Pneumatic stimulation uses small jets of air to supply stimulus directly to the skin, or may use small inflatable devices to produce pressure. Vibrotactile stimulation produces vibratory stimuli using devices such as very small speakers or micro-pin actuators. Micro-pins can also be used either singly or in arrays to produce a sensation of texture or shape. Electrotactile stimulation involves sending very small electrical currents through electrodes on

the skin to produce sensation. As might be expected, each of these methodologies has benefits and disadvantages in terms of cost, weight, comfort, and invasiveness.

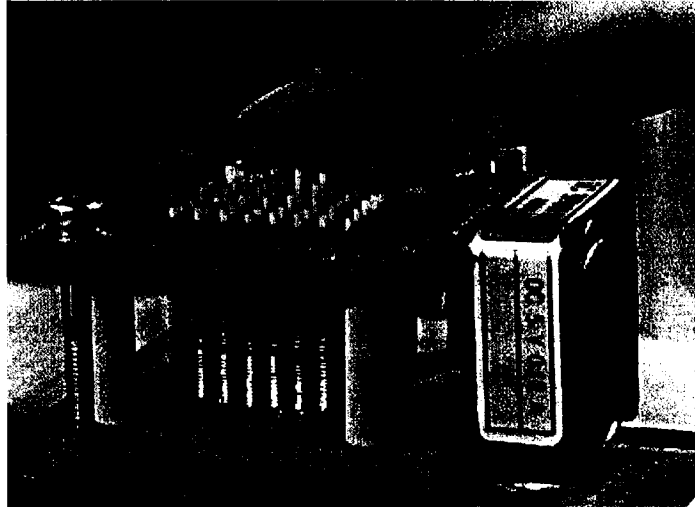


FIGURE 2.1: Pin array Tactile Display

Another class of tactile displays is used to deliver slip feedback. When a person grasps an object, he or she will adjust their grip depending on how much slip is felt. The goal with these devices is to reproduce that feeling when the user grasps a virtual object. Generally, this sensation is recreated by mechanically moving a surface against the fingertips. One group of researchers have used two small plates and moved them while in contact with the thumb and index fingers. Others have developed a device that uses a small motorized cylinder to provide slip feedback (Burdea, 1996).

The third main type of sensation delivered by tactile displays is surface temperature. Certain type of surfaces, such as aluminum and wood, have distinctive “temperature signatures” which allow them to be more easily identified. Thus, temperature feedback adds to the sense of realism when touching certain virtual surfaces. In order to change the user’s skin temperature, researchers use a device called a Peltier heat pump combined with feedback control to adjust the temperature to the desired level.

It should be noted that, while the term “tactile display” technically refers to a device that attempts to recreate skin sensations, most of the literature uses the term synon-

ymously with “haptic display” and includes net force displays under that term. We will follow this practice in the course of this research.

Net force displays allow the user to feel the direction and magnitude of forces, but as if through a tool or a glove - skin sensation is not recreated. Most of the research being performed with haptic displays uses net force displays, of which there are numerous kinds. We will not attempt to identify all of them, but will discuss some of the different types and identify some of the more commonly used devices.

Net force displays may be classified as ground-based or body-based – a ground-based device is mechanically grounded to an immobile object or surface, such as a desk or floor. A body-based device is grounded on the user’s body – most exoskeleton-type feedback devices are body-based. Some body-based exoskeletons include only the hand, others may deliver information to part or all of the arm as well. The Rutgers Master works only with the hand, while the University of Salford Arm Master extends nearly to the shoulder.

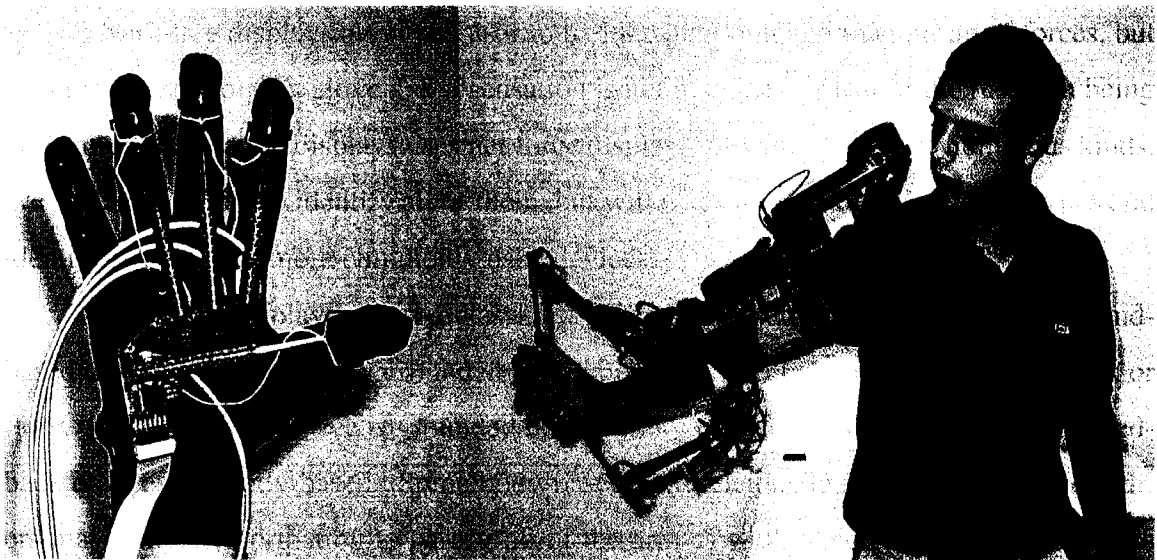


FIGURE 2.2: Body-based Net Force Displays - Rutgers Master and PERCRO Exoskeleton

Of the ground-based net force devices, the joystick is probably the best known to the general public. Most joysticks have at least two degrees of freedom, and, thanks to the gaming industry, there are several which are able to provide force feedback at a relatively

low cost. Researchers at various institutions have also developed more complex joysticks, including Cartesian joysticks, which allow the base to move along two to three axes, and six degree of freedom joysticks.

Another type of ground-based force device is what Burdea (Burdea, 1996) refers to as “pen-based masters” – with these devices, the users experience virtual objects using a pen-shaped tool or pointing-type tool; the best-known of these is the SensAble Devices Personal Haptic Interface Mechanism (PHANToM).

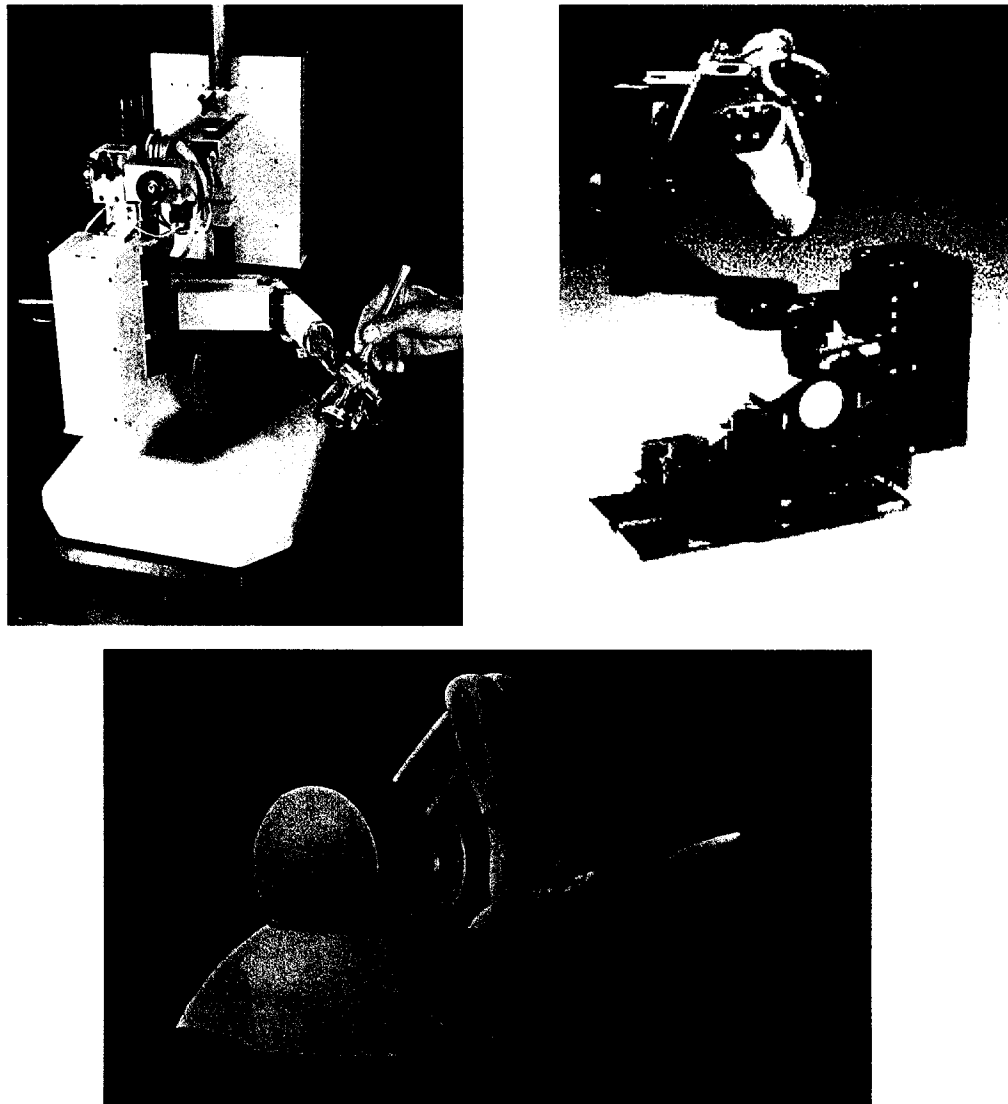


FIGURE 2.3: Ground-based Net Force Displays - MPB Technologies Freedom Haptic Device, Cybernet Systems 6DoF Hand Controller, and SensAble PHANToM Omni

The most common and best-developed haptic displays are the ground-based net force displays, such as joysticks, pen-based masters, and master control arms, and it is these types of devices which have played the largest role in haptics research and development. In some ways, haptic displays could be said to have a long history; in other ways, one could say they are still in the relatively early stages of development. The complexity, multi-dimensionality, and sensitivity of the human sense of touch make even the most sophisticated haptic devices seem crude and limited in comparison. Even so, and with all their limitations, haptic devices have already demonstrated their usefulness in many applications, and it is only a matter of time before their potential is more fully realized.

CHAPTER 3: MULTI-MODAL INTERACTION

3.0 Human-Computer Interfaces

In the field of human-computer interfaces, the practice of discussing, speculating on, and mentally designing the “next big interface” is a common one. The progression from punch cards to command line to WIMP (Windows, Icon, Mouse, Pointer) interface is well known, and it is commonly assumed that there will be an equally significant leap to the next interface paradigm. Along the way, research in the field continues to expand and take advantage of the many new technologies being developed which provide new or augmented ways for people to interact with their computers.

As the cost and fabrication time has decreased on key components such as CPU speed, memory, and specialized hardware, interfaces have no longer had to be limited to what can be easily obtained and built, and design and research in this field is being taken up by more than just engineers and computer scientists - the fields of ergonomics and cognitive science and human physiology are also affecting the way human-computer interactions are being designed, and many of the researchers in the field are taking a multi-disciplinary approach, drawing from many different fields in order to design interactions that will be more natural and intuitive, and make better use of the human senses and capabilities.

From the work in “calm technology” and “ubiquitous computing” pioneered by Mark Weiser and John Seely Brown at Xerox PARC (Weiser & Brown, 1996) to the Flow-Field CAVE immersive installation (Chen et al., 2002), the trend is to engage the user with more than just the visual sense and even, where possible, to offload some of the visual information into another form. With these new types of interface, the senses of hearing, touch, and even smell become part of the human-computer interaction.

3.1 Research in Human-Computer Interfaces

As in any field that is experiencing rapid change, the HCI research being conducted covers a wide spectrum in terms of devices and technology used, supporting orga-

nizations, and background and education of the researchers. In the subset of interfaces that use, consist of, or coordinate with haptic technology, this observation holds true. In addition, the tension between obtaining quantitative data and testing what are often qualitative experiences and interactions makes for a very diverse number of approaches to testing and evaluating these interactions.

When using a haptic device as a coordinating or auxiliary part of an interface, it is often convenient to categorize its use in an application as “showing” something that cannot be seen or is not being shown visually vs. showing something that is also being displayed visually, and therefore acting as more of a redundant and reinforcing information delivery. Thus, we have the developments in augmented desktop interfaces, which use haptic effects to reinforce the user’s experience of moving into and out of windows, over buttons, and interacting with other interface features (Miller & Zeleznik, 1998; Munch & Dillmann, 1997). The approach of using haptic effects to reinforce visual information is also demonstrated by the haptics-augmented physics demonstrations developed by Williams and his colleagues (Williams et al., 2000).

In the area of visualization, a haptic device may be used, either alone or in conjunction with a visual component, to represent something that has no actual seeable (or at least not easily seeable) form. For example, the NIVA (Network Intrusion Visualization Application) system developed at Morgan State University uses haptic feedback coordinated with the software’s visual displays to deliver information about hostile intrusion attempts on a computer network. In this system, attacks on a particular computer system are visualized as nodes in 3-D space, and the attractive force which the user experiences haptically near a particular node indicates the frequency of attacks. Haptically, the nodes are modelled as particles with electric charges, thus producing electric fields which are expressed by the haptic device. Using 10 subjects, half of whom worked with a purely visual system, and half of whom had the additional haptic interface, the researchers found that the accuracy with which the subjects located certain specific attacks was significantly greater using the visual+haptic interface. A PHANToM device was used to provide the haptic feedback (Nyarko et al., 2002).

Similarly, researchers at the University of Utah have incorporated a PHANToM haptic interface into visualization software used to allow the user to interact with vector fields. “Users of [this] system can simultaneously see and feel a vector field” (p.1, Durbeck, 1998). Using combined visual and haptic feedback, the user can trace the path of flow lines within the vector field. Like many researchers in the field, these have used very high-end interface devices including a SensAble PHANToM haptic interface and an SGI Octane for the graphics (Durbeck et al., 1998).

Another research project in the use of visual/haptic interfaces is being conducted at the University of Colorado, where a specialized 5 DoF haptic device is being used in coordination with visualization software to demonstrate shock surfaces and vortex cores. They note that, “scientific visualization is more than the graphical display of data. It is the process of understanding a physical system’s behavior by developing mental models of that behavior” (p.131, Lawrence et al., 2000). While it appears that there are as yet no formal studies to examine the effectiveness of this approach, the authors feel that the response to the initial system has been encouraging (Lawrence et al., 2000).

Similarly, McLaughlin and Orenstein (1997) at CSIRO Mathematical and Information Sciences have used a PHANToM haptic device in conjunction with graphical displays to represent data from offshore seismic surveys. The user can both feel and view the data in various modes.

Some experiments have attempted to compare the different uses of haptics either as reinforcing information or a sole-source display. Researchers at IDEO Product Development and OSU have performed tests with airline pilots using haptic notification of uncommanded changes. They tested users with three kinds of feedback - visual only, tactile only, and visual+tactile, and discovered that users responded more quickly and consistently to the notification that used tactile elements. This work supports the idea of multiple resource theory, which proposes that different sensory “resources” provide additional channels for information presentation and reception. This suggests a potential solution to the problems presented by visual overload, and the possibility of increased effectiveness by using redundant information delivery (Sklar & Sarter, 1999). Campbell

et al. (1999) tested a visual-only, visual+haptic, and unconcerted visual+haptic display (where some features could only be seen and others could only be felt) to try to determine the effectiveness of the different display technologies for a steering task. Fowlkes et al. (2002) also tested users with various display options including augmenting the visual display with haptic or audio feedback.

3.2 Multi-modal Interaction

As has been noted, “multi-modal” is a general term used to describe user interfaces that allow the user to interact using different modes to engage multiple senses. Multi-modal interfaces may cover a broad range of interfaces and interaction types, from a WIMP interface augmented with additional auditory cues to a full virtual reality experience. The defining characteristic is that they attempt to engage more of the user’s senses more fully. The specific goals of the different interfaces and research projects may vary, but in general, they all attempt to give the user a more complete and positive experience - more complex, more instructive, more accurate, more enjoyable, more intuitive, relaxing, educational, etc. and in doing so, to enhance the usefulness of the computer system as a whole. Thus, the researchers in Japan with their “SmartFinger” fingernail-mounted interface (Ando et al., 2002) share the goals of the developers of the CAVE immersive virtual environment (Pape, 2001) and other researchers all across the spectrum.

Many user interface developers are working on ways to deliver information using the other senses, either as an augmentation or a replacement for some of the primarily visual information delivery that is commonly used. The two senses that naturally present themselves as possibilities for use in multi-modal interfaces are hearing and touch. As was noted above, auditory cues, albeit simple ones, are already an integral part of most user interfaces. However, because of physical, technological, and monetary constraints, the use of tactile cues and feedback has only recently become feasible for most interface developers. As a team of researchers doing work for the U.S. Army observed, “while visual and audio stimuli have long been effectively incorporated into immersive training environments, haptics stimulation has lagged behind” (p.1, Fowlkes et al., 2002). More-

over, this observation holds true not just for immersive and virtual reality-type interfaces, but for human computer interfaces as a whole.

In the realm of HCI, researchers and developers have only recently had access to both the computing power and the haptic technology to really start experimenting with interface designs that take advantage of tactile and other modalities.

Karon MacLean, from the University of British Columbia, has observed that, “Haptic feedback is often most effective when associated with other sensory modalities and must be designed in conjunction with them” (p.2, Maclean, 2000) and, “As with vision and audition, touch has many qualitatively distinct components; this information is integrated with input from other senses to form a complex impression” (p.1, MacLean, 2000).

Immersive, or virtual reality interfaces have already been mentioned, and in general, these types of applications have tended to over-shadow other types of interface research. However, a more practical, and potentially more useful form of interface research and development is in the area of what might be called “augmented” interfaces and “alternative device” interfaces. An alternative device interface generally involves the development and building of an original interface device, usually one that replaces the mouse and/or keyboard. Some examples of these include the handheld haptic devices designed by MacLean, Shaver, and Pai at the University of British Columbia (MacLean et al., 2002), and the “geOrb” spherical 3D pointing device developed by Global Haptics (Global, 2003). Augmented interfaces generally build off a traditional WIMP interface and add an additional sensory component either with a specialized form of a common interface device (like a haptic mouse), or with an additional component, such as a joystick.

Another recent development in interface development, and haptic interfaces in particular, has been the development and availability of low-cost haptic devices of reasonable quality. While the vast majority of multi-modal and haptic research is performed using either custom-built interface devices or high-priced research-quality devices, an increasing number of developers have started experimenting with off-the-shelf, or easily-assem-

bled low-cost devices. Thus, Sklar and Sarter used a simple wristband with attached transducers to provide haptic notification in the form of vibrations. Similarly, Hughes and

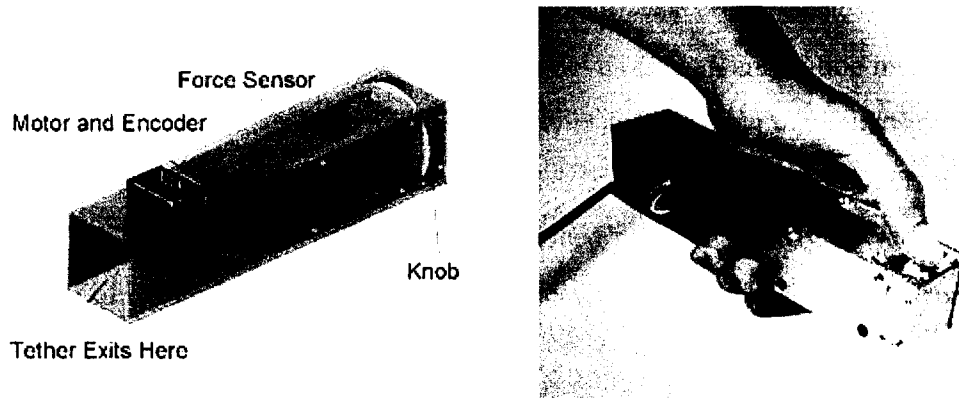


FIGURE 3.1: Handheld Haptic Media Controller

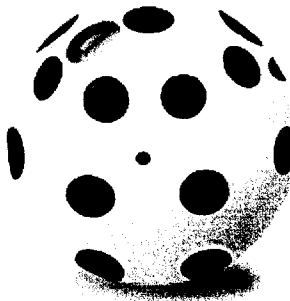


FIGURE 3.2: geOrb 3D Controller

Forrest at the University of East Anglia modified ordinary computer mice by attaching a small speaker coil and magnet to it, with vibrotactile pads positioned where a user's fingers would naturally rest (Hughes & Forrest, 1996).

Research being performed using off-the-shelf haptic devices include that being done by Johansson and Linde, who have experimented with the MicroSoft Sidewinder Force Feedback Pro for visualizing objects. Their application involved having the user explore a virtual maze partially by feeling their way, with the walls being simulated using force feedback. They concluded that using an off-the-shelf device is viable with limitations, such as being limited to 2D, and the limited maximum force available making the walls penetrable and feel somewhat "soft" (Johansson & Linde, 1999). The Microsoft

Sidewinder was also used by Williams and his colleagues (Williams et al., 2000) at Ohio University to demonstrated physics concepts to high school students.

One of the challenges faced with developing new types of interfaces, particularly haptic interfaces, is the lack of history and common language for such devices. Much of the software available today, particularly for Windows/PC-type computers, make use of a set of common interface conventions, i.e., Ctrl-C for copying text, Ctrl-V for pasting, etc. These conventions are not so much intuitive or easy to guess as they have been in use for so long and across so many different software packages, that they have become a de facto standard. Multi-modal interfaces are still too new to have developed such a standard, and thus may seem very foreign to new users. It is assumed that, over time, conventions will develop for these interfaces as well, thus giving users another (or a broader) “language” with which to interact with their computers.

3.3 Designing a Multi-modal Interaction System

The following general model can be used as a structure for examining and designing the different aspects of computer-user interaction (MacLean, 2000).

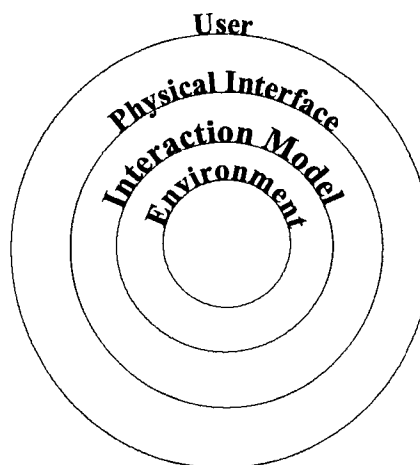


FIGURE 3.3: User Interaction Model

At the outermost layer is the user; a well-designed computer-human interface system will take into account what is known or assumed about the target user, including the user’s goals, his or her experience with similar systems, and any physical limitations.

The physical interface layer refers to the actual displays and input devices, such as the keyboard, mouse, monitor, and force-feedback joystick. For the most commonly used interface devices (monitor, mouse, keyboard, speakers), it is generally not necessary for the designer to take steps to control these devices directly. However, for less-commonly-used devices, the demands or limitations of the system resources may require the designer to include direct control as he or she develops the interaction system. Because of the processing demands for producing a smooth and tightly-coupled interaction experience, many researchers program their systems to directly control the haptic devices being used, a practice that generally requires either building one's own haptic device or using one of the more expensive "research-level" haptic devices. However, because the goals of this research include using commonly-available system components, we instead chose to use an off-the-shelf haptic unit, which included some built-in low-level control capabilities.

The next layer is the interaction model – this "defines the relation between user and environment" (MacLean, 2000). It is necessary to determine the form that the interactions between user and environment will take, as well as how the user input will be interpreted and how it will affect the environment. The interaction model may include sub-models which define the interactions for each sensory modality for both input and output. Thus, Snibbe et al. (2001) used a model of a braking system in developing a haptic media controller, and Johansson and Linde used spring forces to model contact with walls in their maze. Mark et al. (1996) have noted that the interaction model (which they call the intermediate representation) can vary significantly, from using spheres of contact to model molecular interactions to simulating force fields using equations to describe the fields. Their own work has involved extensive use of the probe-to-plane model, and point-to-point springs model.

The innermost layer represents the environment that is being observed and manipulated, in this case, the computer-generated world or model or representation. The environment may be a direct representation of a real-world mechanism or system, such as a virtual cockpit, or it may be a more abstract construct as visualized by the developer. Thus, the Virtual Explorer tries to create a realistic-feeling excursion into the human body,

including visual and haptic displays of fluid flow, encounters with bacteria, and human immune responses. At a more abstract level, McLaughlin and Orenstein (1997) visualized seismic data as volumes of varying colors, with faults indicated by identifying colors and a barrier haptic effect.

For this research, the virtual environment is a representation of the real-world “Smart Car” system. This consists of a car which will attempt to automatically follow a white line on a dark track using input from two optical sensors near the front of the car. The car’s speed may be adjusted, and its steering performance will change depending on which control algorithm has been selected, and the gain settings. More details on the design of the virtual environment will be found in the next chapter.

The interaction model for this system is based on standard Windows interface components, including drop-down menus and point-and-click buttons. The joystick interface is loosely based on controllers from joystick-controlled video games such as flight simulators where the movement of the vehicle is reflected in the movement of the controller. For the graphic+haptic simulation, the input is both menu-driven and device-driven, where the joystick controls have been mapped to the various menu functions to create a custom interface device. The output is both visual and haptic, with the haptic component providing reinforcing information to the aspects of the graphic display that communicate the direction and magnitude of the car’s steering.

CHAPTER 4: SYSTEM DESIGN AND IMPLEMENTATION

4.0 General Description

The application software creates a window with an overhead view of a car running on a track - the car follows the white line in the middle of the black track by simulating the action of optical sensors which sense the reflective light from the white versus the black part of the track. The car can steer in the range of -30 to 30 degrees from the car's direction of travel. The magnitude of the steering is dependent on the difference in data from the two optical sensors, and which control algorithm has been selected for use. The user can choose one of three control algorithms: Proportional, Proportional + Integral (PI), or Proportional + Integral + Derivative (PID); he or she can also adjust the appropriate gains.

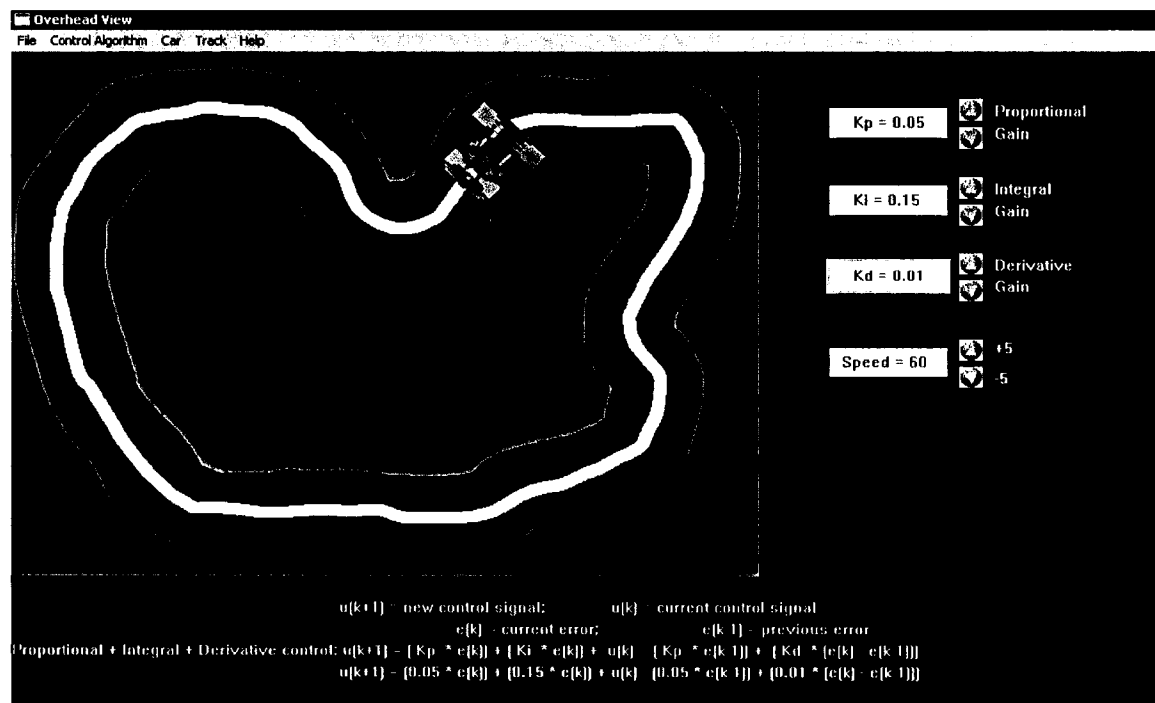


FIGURE 4.1: Screen Shot of the Simulation

The control algorithm being used is displayed at the bottom of the window, including the selected gains. A drop-down menu system allows the user to choose the desired

control algorithm, choose one of three tracks of different difficulty levels, and also to send the car back to the starting point or to reverse its direction. For the joystick interface, we were able to map all the menu and simulation controls onto the various joystick buttons, except for the action of choosing different tracks. An additional slider control on the joystick would have served well for that - unfortunately, the joystick we used only had one slider, and that worked very effectively for speed control. A “help” graphic consisting of a picture of the joystick with the various controls identified was provided to help the users become accustomed to the controls.

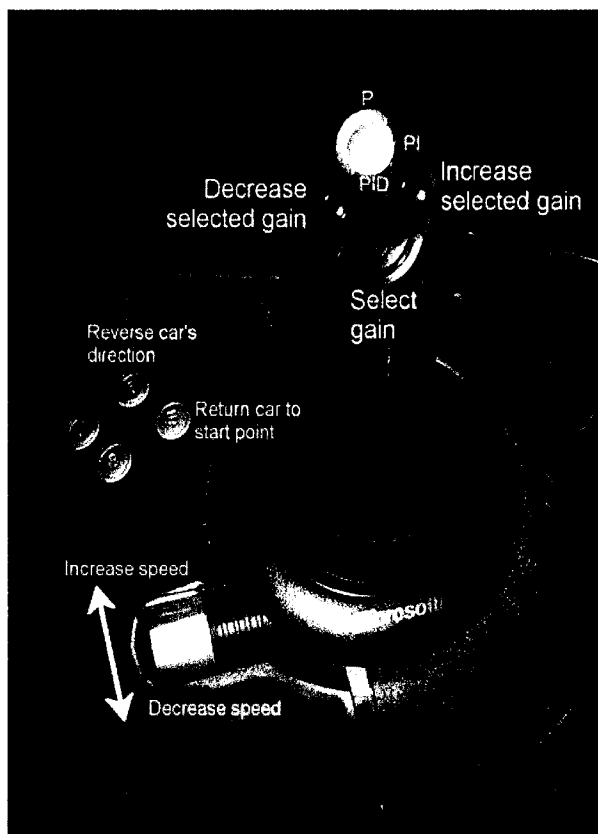


FIGURE 4.2: Help Graphic of Joystick Controls

The haptic joystick provides feedback by reacting to the car's steering with a force in the same direction as the steering, and with a magnitude proportional to the magnitude of the steering. The maximum force feedback magnitude was set at 5000, with a maximum possible value of 10,000 - 5000 seemed quite sufficiently strong enough for the users, although the option of changing the maximum force feedback could be added fairly

easily. An interesting experiment might be to give the users that option as well as the option of turning the force feedback off completely, and then observing how often and in what ways the users made use of the force feedback.

4.1 Hardware Configuration

Broadly speaking, there are three possible hardware configurations for use in a multi-modal interaction system. The simplest is to use a single CPU for all processing, including updating the virtual model, computing the haptic and other media displays, controlling and driving the displays, and keeping track of user input. The advantages of this configuration is that it is conceptually simple and will be minimally impacted by interprocess communication lag times. There are limitations, however, in that the demands of computing for multiple displays often pushes or exceeds the capabilities of the CPU, causing slowdowns for other processes, and even within the interactive system.

Many researchers have chosen to use multiple networked CPUs for their systems, often using significantly different configurations in order to optimize each CPU for its assigned function. This approach takes advantage of the ability to optimize different systems for different applications. However, this approach can cause difficulties with synchronization of the different components, as well as requiring the developer to account for network communication time lag and problems. Moreover, this approach can easily become very expensive.

A variant on the use of multiple networked CPUs is to use a low-cost embedded controller in the haptic device in conjunction with a single main CPU. The main advantages of this approach are the efficient compartmentalization of the low-level haptic display processing, and relatively low cost. There are limitations, namely, the developer is limited to an extent by the capabilities of the embedded controller. Also, as with the multiple networked CPUs, there is the possibility that the communication speed and bandwidth may be limited. Nonetheless, the low-cost embedded controller used in conjunction with a main CPU seems to afford the most possibilities for developing a low-cost and effective multi-modal display system. The joystick used for this research was a Microsoft

Sidewinder Force Feedback 2, which includes an on-board 16-bit microprocessor which runs at 25 MHz (Johansson & Linde, 1999).

4.2 Haptic Rendering

Haptic rendering is the process of determining, calculating, and delivering tactile information to the user of a haptic interface. For most haptic displays, it is the process of determining which and how the forces representing some aspect of the virtual environment will be applied to the user through the force-feedback device. Thus, two questions are addressed:

What should a particular action/activity/object/event feel like?

How can the forces which will provide that feeling be calculated and delivered?

Clearly, different actions and events should feel different – for example, contacting a beaker of radioactive material with a remote arm should feel differently than shooting a machine gun in a video game. What the developer must determine is how to use the capabilities of the haptic display to deliver forces in ways that will most clearly provide the desired information to the user.

In general, the forces representing a virtual environment are calculated using mechanical systems to represent the virtual actions – these mechanical systems may or may not be actual representations of the virtual actions. Thus, the behavior of a haptic display representing a virtual flightstick may be based on a model of an actual flightstick, while a haptic device displaying surface friction may use a model of a smooth surface with a number of small grooves in it (Mark et al., 1996).

One factor that might need to be considered in designing haptic effects is Weber's Law, which states that the magnitude of a change in intensity which is necessary for the change to be noticeable is not constant. Rather, the ratio of the change in intensity to the original intensity is a constant. From this law, a high-magnitude effect will require a larger change in magnitude in order for the change to be perceived than would a low-magnitude effect. For this research, Weber's Law was not taken into account in designing the haptic feedback as described below; nonetheless, that and other principles learned from

psychophysics research can provide important input in the development of haptic and multi-modal interfaces.

Like graphic rendering, there are many different approaches, algorithms, and shortcuts that can be used to make the haptic rendering process more effective and efficient. The methodology chosen may depend on the environment being observed, the interaction model, or the preferences of the developer.

4.3 Designing the haptic feedback:

For this research, the purpose of the haptic feedback was to give the user an overall sense of how the simulated car was steering, specifically trying to convey the magnitude and direction. We began by designing a basic constant force haptic response - this type of haptic feedback will exert a continuing force in a designated direction until it is explicitly stopped, or a parameter is changed. This type of force display worked reasonably well; however, we also wanted to try some other force responses to see if they would be more effective. In particular, we had theorized that a spring force would make effective feedback. Unfortunately, using the pre-defined spring force proved to be infeasible because it is dependent on the user moving the joystick, which is not part of the interaction design. A number of other force displays were tried, including a wave force, ramp force, and square periodic wave. None of these forces were as effective as the constant force, so the decision was made to refine and work with that. We also attempted to make the haptic feedback more interesting by layering in a low-level rapid waveform to give the feel of “engine noise”. Unfortunately, this served to weaken the main haptic feedback so that the lower magnitude constant force responses were harder to feel; therefore, the engine noise was eliminated.

We designed the constant force feedback effect to vary its magnitude proportional to the amount of steering that the simulated car would do, and in the same direction. Because we found that the lower magnitude force effects were difficult to perceive, it was decided to attempt to use a scaling function that would increase the magnitude of the lower-magnitude effects, while having a lesser effect as the original calculated magnitude

increased. Durbeck et al. suggest the possibility of using nonmonotonic functions in a vector visualization application in order to amplify the effect of small vectors, or filter out a specific range of vectors. The suggested amplification function shown in (Durbeck et al., 1998) resembles a square root function $y(x) = \sqrt{2x}$ or error function

$$erf\ x = \frac{2}{\sqrt{\pi}} \left(\int_0^x e^{-t^2} dt \right).$$

Because neither Durbeck nor any of the other research that we found went into detail about whether or how their haptic effects were scaled, we attempted to identify a suitable function that would meet the requirements of increasing the magnitude of the input non-linearly with a fairly rapid increase at lower input values, and less of an increase as the magnitude of the input became larger. One function that looked appropriate was that of a simple single degree of freedom non-linear spring (FEMur 1996; Purdue, 2003; Unisa, 2004). The formula for this is:

$$F_{spring} = (kx + bx^3) \quad (4.1)$$

where k and b are spring constants and x is the magnitude of the spring's displacement. If the spring constant b is positive, the spring force will become stronger non-linearly as the spring displacement increases; if b is negative, then the spring force will weaken as the spring is displaced.

However, after performing several test calculations with this formula, it became clear that this would reduce the magnitude of the haptic feedback too much too quickly, so the formula:

$$F = (kx + bx^2) \quad (4.2)$$

was used, with $k = 11$ and $b = -0.01$ (Figure 4.3). At the point where the function goes

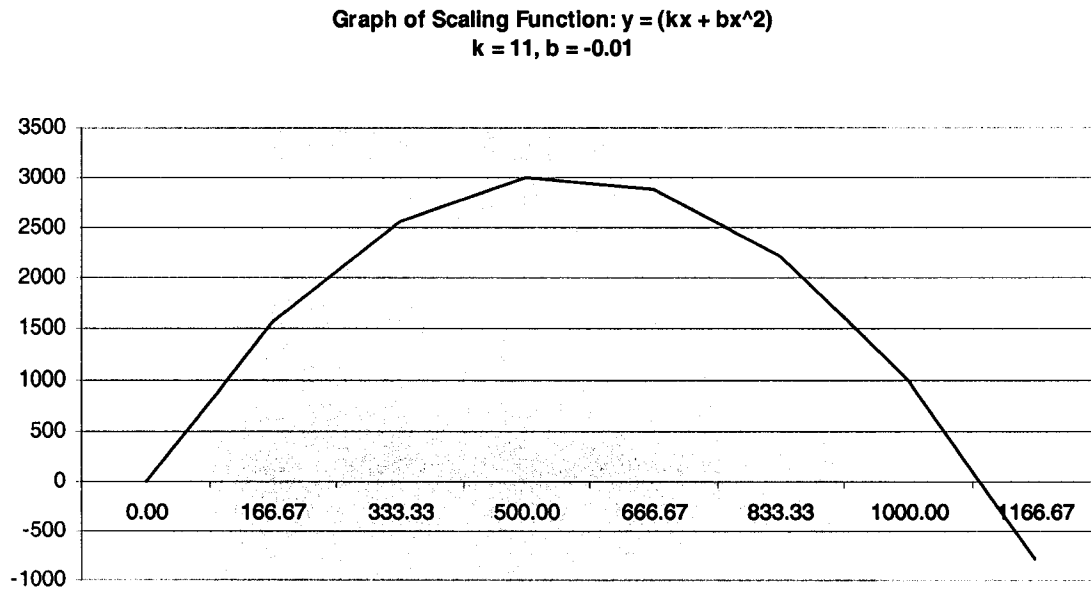


FIGURE 4.3: Graph of Scaling Function

negative, the function is no longer applied to the magnitude of the force feedback. This gives the graph for the haptic feedback magnitude in Figure 4.4.

We chose to use this function, which gives the needed additional force to smaller haptic effect values to give the user a more definite effect. However, as Figure 4.4 clearly shows, there is a drop in the scaled force magnitude after the unscaled force has reached 500, and only when the unscaled force magnitude is at around 1100 does the scaling equation stop affecting the force magnitude, which then increases linearly to the designated maximum of 5000.

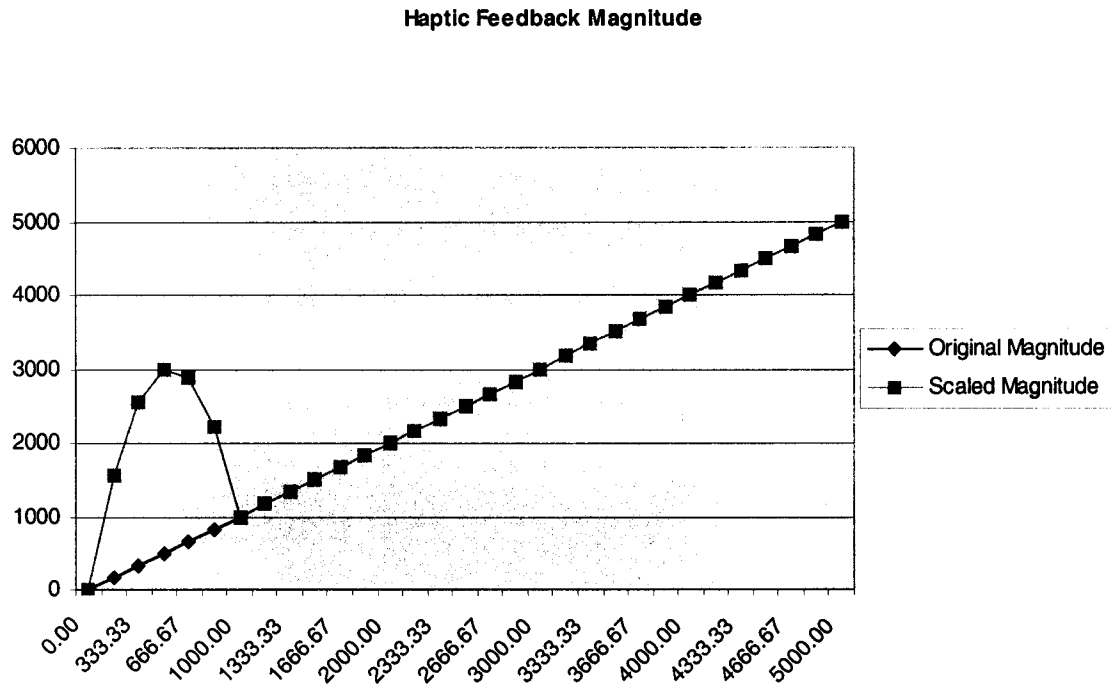


FIGURE 4.4: Graph of Scaled and Unscaled Haptic Feedback Magnitude

A force envelope was also added - this structure allows the designer more control over the haptic feedback, by designating the level at which the force will start, the amount of time it will take to reach the full designated strength (attack time), the level at which the force will end, and the amount of time it will take to go from full strength to the ending level (fade time). After some experimentation, we settled on an attack time and fade time of 0.1 seconds. The starting and ending levels were left at 0, and the full designated strength was the calculated and scaled magnitude of the force effect as described previously.

4.4 Software Architecture

There are two major processes that need to take place concurrently – computing and updating the graphic display, and computing and updating the haptic display. The update rates for the two displays are very different, with 1000 Hz being the standard for haptic devices, and 60 Hz being appropriate for graphic devices. Achieving the appropri-

ate update rate for a haptic display can be a significant challenge for the researcher, and one that for many applications has only been solved by investing in two different (and usually high-powered) machines to process and update the displays. However, the system design for this research did not involve complex graphics or VR-type haptic manipulations, and it was therefore determined that the update requirements for the haptic display could be less stringent. This opened up the possibility that an effective multi-modal display could be developed by focusing on an appropriate software architecture.

Much of the research that has been done using multi-modal displays was based on graphical display work, and thus the graphics processing took priority. More recently, researchers have taken the approach of giving priority to the process with the more demanding requirements. We chose the latter approach, and gave priority to the haptic display processing. The software architecture is multi-threaded and driven by the virtual environment and haptic display processing.

4.5 Multithreading

A thread is a piece of a process that is created and runs within the address space of the larger process that created it. When using multithreaded programming, a thread can perform a task in parallel with other process tasks, thus making the process as a whole execute more quickly. A multithreaded architecture allows the developer to prioritize different tasks by placing more important or processing-intensive tasks in the main thread and less demanding or important tasks in one or more secondary threads. For the development of this simulation, it was decided that the virtual environment processing and updating, and the haptic processing should take place in the primary thread. We experimented with having all the visual display processing take place in the secondary thread; however, this resulted in obvious discontinuities in the visual display. Therefore, some pieces of the graphics processing were placed in the main thread.

The following processes and calculations took place in the main thread:

1. Virtual Model calculations and updating
 - Update car position

- Check joystick controls for input, update parameters
 - Get “optical sensor” data
 - Calculate new steering angle
 - Calculate haptic feedback magnitude
 - Update car and sensor angle
2. Graphics processing
- Draw car
 - Draw control formulas

The secondary thread performed the remaining graphics processing:

- Draw the track
- Draw the increment and decrement arrows for the gains and speed
- Draw the gain and speed displays and labels, including high-lighting the selected gain

It should be noted that this research was performed using single processor computers - therefore, the multi-threading was virtual rather than actual. However, the multi-threading is part of the system that was developed and uses the multithreading libraries; some minor modifications should be all that is needed to run the system on a multiple-processor computer.

The virtual multithreading provides each thread of execution with processor time in a round-robin fashion, with all threads with equal priorities receiving equal time. After experimenting with different priority settings for the secondary thread, it was discovered that a lower priority caused visual disruptions because re-drawing was not happening quickly enough. Therefore, the secondary and primary threads were assigned equal priorities. Having the graphics run as a lower priority thread did not appear to affect the performance of the primary thread, which included the haptic feedback. We believe that, with a

faster computer and especially one with multiprocessing capabilities, the ability to prioritize the threads will be much more useful and effective.

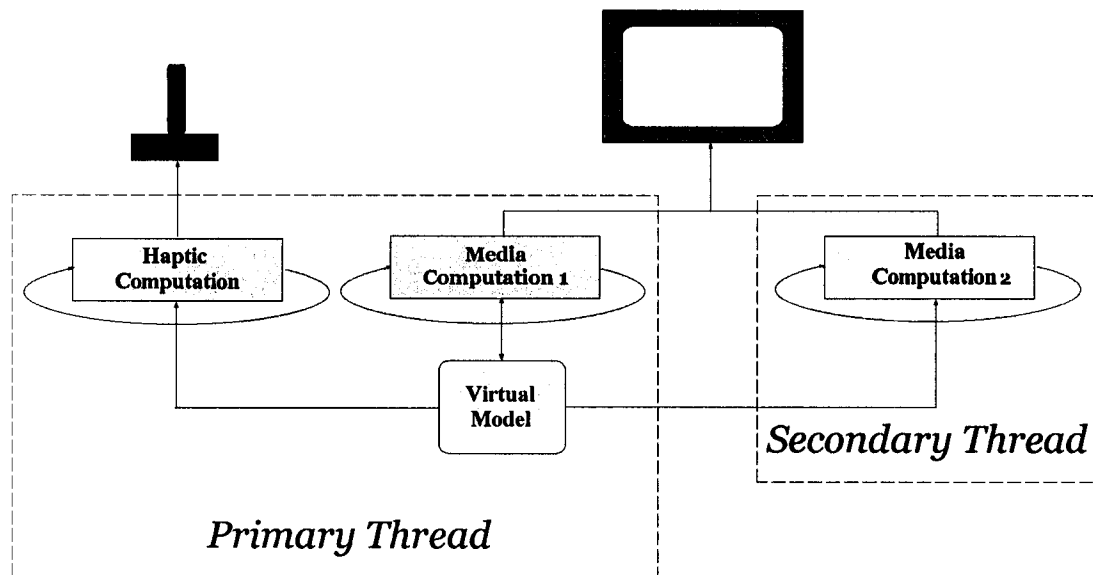


FIGURE 4.5: Multi-threaded Software Architecture for Multi-modal Display

While this is by no means a complete test of the use of a multi-threaded architecture to produce a multi-modal display, this work demonstrates the feasibility and usefulness of multi-threading capabilities in developing and running software that needs to perform graphics and other functions at a high speed.

It should be noted that, with a faster and/or an actual multiprocessor machine, timing will become much more of an issue. When using multi-threading, it is important to synchronize threads so that a secondary thread completes its task before the primary thread completes its parallel task and closes the secondary thread. For this simulation, the secondary thread tasks were few enough that there was no need to worry about checking for its completion. However, given time to develop the multithreading capabilities for more optimal operation, it would become important to make sure the threads were synchronized so as to be sure the essential processing is completed before those values are needed.

4.6 Control Algorithm Implementation

The three types of control algorithms available to users of the software are: Proportional, Proportional + Integral (PI), and Proportional + Integral + Derivative (PID). These were used because they are generally the earliest control methodologies that students learn, and they also have wide applicability.

Proportional controllers have the control law:

$$u(t) = K_p e(t) \quad (4.3)$$

where $u(t)$ is the control signal at time t , $e(t)$ is the error measurement at time t , and K_p is the proportional gain. In a discrete time system, such as the one being simulated, t can be replaced by k , representing the time-step. This will give the following formula:

$$u(k) = K_p e(k) \quad (4.4)$$

This formula was implemented to give the control signal using proportional control.

For the research simulation, the error is the difference between the values given by the right and left optical sensors.

The control law for a PI controller is:

$$u(t) = K_p e(t) + K_I \int e(t) dt \quad (4.5)$$

where $u(t)$ is the control signal at time t , $e(t)$ is the error measurement at time t , K_p is the proportional gain, and K_I is the integral gain. Once again, working with a discrete time system, t is replaced by k , and the integral is approximated by a sum as follows:

$$u(k+1) = K_p e(k) + K_I \sum_{j=0}^k e(j) \quad (4.6)$$

This equation can then take the form:

$$u(k+1) = K_P e(k) + K_I e(k) + u(k) - K_P e(k-1) \quad (4.7)$$

where the error measurement from the previous time-step as well as the present time-step is used to calculate the next control signal. This equation was implemented in the simulation program to determine the appropriate control signal when using Proportional+Integral control.

The control law for a PID controller is:

$$u(t) = K_P e(t) + K_I \int e(t) dt + K_D \frac{d}{dt} e(t) \quad (4.8)$$

where $u(t)$ is the control signal at time t , $e(t)$ is the error measurement at time t , K_P is the proportional gain, K_I is the integral gain, and K_D is the derivative gain. Once again, working with a discrete time system, t is replaced by k , the integral is approximated by a sum, and the derivative is approximated as follows:

$$u(k+1) = K_P (e(k) - e(k-1)) + K_I e(k) + u(k) + K_D (e(k) - 2e(k-1) + e(k-2)) \quad (4.9)$$

where the error measurement from the previous time-step as well as the present time-step is used to calculate the next control signal. This equation was implemented in the simulation program to determine the appropriate control signal when using Proportional+Integral+Derivative control.

We also implemented a “bang-bang” controller option to give the user an additional control possibility, but it was found that the car would immediately go off the track if it steered the maximum amount on every iteration, so the option was removed.

Once the new control signal was determined, it was scaled to be within the -30 to 30 degree range, which was the maximum left and right turning parameter for the simulated car.

4.7 Simulation specifics

The speed of the simulation car is given in a range of 10 to 70. The actual speed is controlled by increasing or decreasing a delay variable at the end of each iteration through the main program. A delay of 0 gives the maximum speed and a delay of 450 gives the slowest speed - it was decided to scale these values to be displayed in a range more familiar to the students - the embedded control cars have a maximum speed of ~70, and moreover, actual cars going 70 are moving relatively quickly.

The maximum turn of the wheels is set at ± 30 degrees - this is also close to the specifications of the embedded control Smart Car.

The optical sensor data is simulated by taking the actual pixel values at the locations of the simulated car's two optical sensors - the pixel and each pixel surrounding it (9 pixels total) are checked for their color values, and the average is used as the sensor input. The difference between the two sensors gives the error value, which is then used in the control calculations.

This simulation was developed on a Dell Precision Workstation 410 MT desktop PC running Windows 2000 Professional on an Intel Pentium II CPU running at 600MHz, with a 19" color display. The simulation graphics were developed in 8-bit color and drawn using Jasc Paint Shop Pro 7. The car graphic was modified from a public-domain clipart image. The software development was done in Microsoft Visual Studio 6.0 using C, C++, and DirectX 8.0. The code makes use of C++ libraries developed by Andre LaMothe. The haptic interface is a Microsoft Sidewinder Force Feedback 2 joystick, controlled through the USB port using USB 1.1.

CHAPTER 5: RESEARCH METHODOLOGY

5.0 Teaching Control Algorithms

At Rensselaer Polytechnic Institute, the Embedded Control Laboratory course is an important part of the Core Engineering curriculum. As a Core Engineering course, it is taken by nearly all engineering students, usually during their sophomore or junior years. The goals of the course include: teaching students the basic principles and use of embedded control, giving students hands-on experience in developing and analyzing an embedded control system, and teaching students basic supporting concepts, such as simple programming, basic circuits, and control algorithms. In accomplishing these goals, the students build and program a “Smart Car”, a small car chassis with sensors to acquire data, controllers to perform steering and driving actions, and an on-board microprocessor that is programmed to control the functioning of the car based on the data received. The basic Smart Car system is able to use optical tracking units to follow a white line on a dark surface.

As noted above, control algorithms is one of the supporting concepts that is taught in the class – the topic is addressed, and the students use the information, but it is not covered exhaustively. While it is not the goal of the Embedded Control Laboratory course to give students the same amount and depth of knowledge as a controls course would, the instructors have often observed that some of the students attain only a cursory understanding of the subject. This has been particularly noticeable on the exams, when students are asked to apply the knowledge to an unfamiliar problem, and when the students are attempting to get their cars to run on a difficult track – under these circumstances, it becomes clear that many of the students do not fully understand the control algorithm being used, and the effects of changes to the algorithm and the gains. (The control algorithm used for both the steering and the drive motor is a PI controller.)

Even when the students are able to observe the behavior of their car, they are often not experienced enough to know what changes to try based upon what they have observed. They often draw incomplete or incorrect conclusions about the car’s performance and

problems. Moreover, many of them are much more intent on getting “checked off” than on understanding their car’s control characteristics. If, however, a relatively simple demonstration of different control algorithms could be made available to the students, one which would allow them to experience the differences in behavior on multiple levels, then there is the possibility of increasing the depth of their understanding, as well as allowing them to experiment with different algorithm/gain combinations with minimal extra effort.

It was hoped that, by using the simulation, students would learn some of the basic concepts relating to Proportional, Proportional+Integral (PI), and Proportional+Integral+Derivative (PID) controllers. At the very least, we hoped it would be clear that PI and PID controllers provide more effective control than a proportional controller, particularly with more difficult tracks and/or higher speeds. In addition, we wanted the users to note the faster response as the proportional and integral gains were increased, and also how too great an increase would result in system instability. Similarly, decreasing the proportional and integral gains would demonstrate how to make the controller less reactive, possibly to the point of being unable to follow the track.

While the system was not intended to teach terminology such as “underdamped” and “overdamped” systems, we wanted the users to gain an experience of those kinds of systems, by observing how an overdamped controller would not provide enough or rapid enough of a response to keep the car on a tight curve, or an underdamped controller would result in a rough ride by numerous over-corrections in the steering. In addition, it was hoped that the students would start to get a sense of the way other factors can affect a controller’s effectiveness; for example, noting that a car moving at top speed on even a moderate difficulty track may need a more reactive controller than when the car is travelling at a lower speed on the same track. Moreover, we wanted the users to start thinking about the kinds of trade-offs that are sometimes necessary in controller design, such as sacrificing a very smooth ride in order to be able to complete a difficult track at a high speed.

It was also hoped that, by displaying the control formulas being used, the more interested students might be able to get a sense of how the different components interact with each other.

5.1 Experimental Method

The chosen methodology for this research is based in part on that of researchers at the University of North Carolina at Chapel Hill and Iowa State University which was used in a study of student use of the NanoManipulator system. The NanoManipulator uses an Atomic Force Microscope and PHANToM force feedback device to allow students to explore the properties of adenoviruses. The researchers noted that, “We are interested in understanding how the addition of haptic feedback to visual feedback and active manipulation of viruses affects students’ learning. Researchers in virtual reality have recently shown that the addition of haptic feedback to virtual environments enhances the performance of people when learning a navigational task” (p.2, Jones et al., 2002). The participants in this study were 209 junior high and high school biology students. The study included a pre-assessment process (including a Knowledge Test, Beliefs Questionnaire, and Pre-Opinion Questionnaire) about two weeks prior to any instruction, five class periods of instructional activities, three days of student team exercises, a one-class writing project, and a post-assessment process (with Post-Knowledge Test, Post-Experience Questionnaire, and Post-Opinion Questionnaire) about one week after instruction. There were also individual interviews with students before and after the main instruction process. Half the students were provided with visual and haptic feedback, and the other half worked with a purely visual system.

The assessment tests and questionnaires included both qualitative and quantitative questions as researchers wanted to determine not only if students would develop more accurate knowledge about viruses, but also if the addition of haptic feedback would affect their attitudes toward the learning system and toward scientists and science in general. Overall, the study showed that student attitudes were more positive when they had the haptic feedback as part of their experience; and that clay models of viruses made by students at the post-assessment were slightly more likely to resemble the virus models they had seen during instruction if the students had haptic feedback. However, the paper did not address the quantitative scores that the students received (Jones et al., 2002).

For this research, we have utilized a pre- and post-assessment process, with assessment questions which addressed both quantitative and conceptual knowledge measurement. The testing process did not extend over the same length of time as the above study, nor was there extensive instruction or other learning projects to augment the students' learning. Moreover, the subjects in this research did not use a device of nearly the complexity or sophistication of the PHANToM. By using a relatively inexpensive "off-the-shelf" haptic device, we planned to show that the benefits of haptic interfaces need not be limited to schools or departments with large budgets.

5.1.1 Procedure

The user testing for this research was performed in a laboratory classroom on the RPI campus. Each test subject worked on one computer, with no more than one other subject using the system on another computer in the same vicinity. Test subjects could ask for help or offer comments at any time. They were also allowed to take as long as they wished to complete any aspect of the experiment.

The first step of the experiment was the administration of the pre-assessment survey - this involved a combined knowledge assessment and background questionnaire. The background questionnaire asked the user about his or her background in controls - whether he or she has had one or more controls courses, or if controls was one of several topics in a course, etc. Other questions had the participant rank his or her knowledge or use of simulations, computer games, and basic controls. It also gathered demographic data such as age and gender. The knowledge assessment attempted to determine the user's overall knowledge about basic controls, including algorithms, the effects of changing the different gains, and the correct way to use control algorithms in problem-solving. The pre-assessment was taken before any actual use of the simulation system. It should be noted that no participant was told which version of the system he or she would be using until after the pre-assessment survey was finished.

After the pre-assessment surveys were completed, the test subjects were provided with a one-on-one orientation, lasting two to three minutes, to the software, describing the operation of the simulated car and pointing out the menu options and/or joystick control

options. They were then encouraged to work with the software until they felt comfortable with it, and then to begin work on the worksheet.

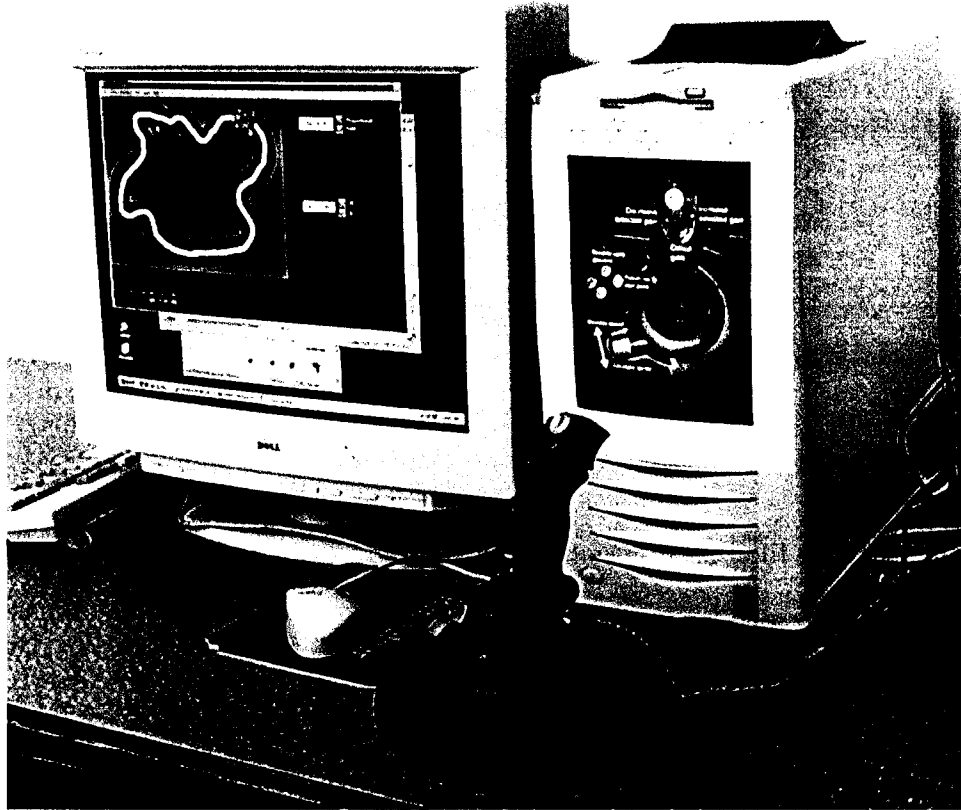


FIGURE 5.1: Testing station

The worksheet guided the user through a series of exercises using the simulation, asking them to try different tracks and algorithms, and write down the settings they used and their observations. The end of the worksheet included a request for comments or feedback from the users. In addition, once the worksheet was completed, a few minutes were spent wherein the users were individually asked for feedback about the system and its usability, and suggestions for possible improvements.

The post-assessment survey was given at least a day after the first part of the study; the goal of this requirement was to lessen any short-term learning effects caused by taking the pre-assessment survey. The amount of time from pre-survey to post-survey varied from one to seven days, with the average being about three days. The post-survey

included questions of similar difficulty to the pre-survey, with the goal of evaluating the user's learning about basic controls. The questions included objective and subjectively scored questions. At the end of the post-survey was a more detailed feedback and comment section which asked the subject to evaluate and score a series of statements about the simulation, to indicate any additional features or help they would have liked to have, and to express their opinions, feelings, and suggestions for the simulation system.

The knowledge assessment portions of the surveys were evaluated by two graduate students with extensive experience in controls and the embedded controls course - they provided feedback and suggested appropriate answers. The possibility that the pre-test might motivate information-seeking about the subject, or that additional learning about the subject might take place after the pre-test and use of the software, but before taking the post-test, was not controlled for.

5.1.2 Design

The experiment used a pre-test post-test between-groups design. The goal was to determine the probability of a relationship between the type of software the participant used and his or her learning. The independent variable examined was the type of learning software, of which there were two types: a visual-only system, and a multi-modal (visual+haptic) system. The dependent variable was learning as measured by the difference in scores between a knowledge assessment test given before using the software, and one given after.

The pre- and post-assessment surveys were prepared by the researcher; colleagues and the pilot group participants offered feedback and suggestions for the surveys, some of which were incorporated into the surveys for the final test group. All final test group participants were given identical pre- and post-assessment surveys, except for the correction of minor typographical errors. The knowledge assessment questions included a mix of multiple choice and short answer questions - some questions were based on test questions that had been used in the embedded controls course. The worksheet which was used with the software was also developed by the researcher.

The user goal of the system is to choose a control algorithm and set the gains in such a way that the simulated car follows the white line as smoothly as possible on the selected track. Participants could choose from six tracks - an easy, medium difficulty, and difficult track, each with a variation of a wider and narrower white line. The performance of the car's steering is demonstrated visually by the movement and rotations of the simulated car. As mentioned previously, the haptic display delivers steering information in the form of scaled force effects in the direction of the car's steering. It should be noted that the haptic display is designed such that it will only deliver force effects when the user is holding it; if the user lets go, the joystick will not move. The rest of the system is not affected by whether or not the joystick is being held.

Participants could choose between Proportional, Proportional + Integral (PI), and Proportional + Integral + Derivative (PID) control algorithms using the menu system, or, for the haptic users, the hat button on the joystick. Depending on which control algorithm had been selected, the appropriate gains would be displayed, with buttons for the user to increment or decrement the gain values. The car speed was also displayed on a scale of 10 to 70. Below the main screen, the appropriate control equation was displayed. System controls also offer the option of sending the car back to the start position (at the lower center of the track screen), and reversing the car's direction.

5.1.3 Participants

Participants for the study were recruited using advertising posters displayed on campus, and in the summer session of the embedded controls course. A total of 20 users were tested for this research - 10 used the simulation software with the haptic feedback, and 10 used the software without. The first six test subjects were classed as the pilot study group - based on their suggestions and comments, certain changes were made to the software and to the pre- and post-assessment surveys. The primary changes consisted of:

1. Users observed that changing the speed by ± 1 had very little effect, and it was time-consuming to change the car's speed. Therefore, the speed increment and decrement buttons were changed to "+5" and "-5" to allow speed changes to take place more quickly.

2. Users felt that the car would be more interesting to control if it could go faster. Therefore, the maximum car speed was increased to the maximum that the computer could generate (i.e., no delay between iterations).
3. Users with little background in controls observed that the use of unfamiliar terminology made it difficult to answer questions on the pre- and post-assessment surveys. Questions were added to the pre- and post-assessment surveys that used the terms “under-correcting” and “over-correcting” in place of “over-damped” and “underdamped”.

The remaining 14 test subjects were classed as the final study group. Once the testing entered that phase, no changes were made to the software or the surveys, except for correcting minor typographical errors.

Users were assigned to the two versions of the simulation in alternating order of when they were scheduled to be tested - thus, the first subject used the non-haptic simulation, the second subject used the haptic simulation, the third used the non-haptic, and so on. The exception to this was when two test subjects came in together - then they were allowed to determine which one would use which version. This occurred once with the pilot cohort, and twice with the final cohort.

The pilot cohort consisted of four men and two women, ages ranging from 18 to 30. The final group consisted of eight men and six women, aged from 18 to 29. Four women and three men used the non-haptic system, and two women and five men used the haptic system.

The results and data analysis of the pre- and post-assessment surveys, including the feedback, are summarized in the following chapter.

CHAPTER 6: RESULTS AND DISCUSSION

6.0 Pre- and Post-assessment Survey Results

6.0.1 Demographic Data

As was noted in the previous chapter, the pilot cohort consisted of four men and two women, ages ranging from 18 to 30 ($M = 21.67$, $SD = 4.59$), and the final group consisted of eight men and six women, aged from 18 to 29 ($M = 22.21$, $SD = 3.62$). In the final test group, four women and three men used the non-haptic system, and two women and five men used the haptic system. Most of the participants were students - in the final group, there were 10 undergraduates (three sophomores, one junior, six seniors), three graduate students, and one participant who was no longer in school.

As part of the pre-assessment survey, participants were asked to rank their knowledge and use of certain items, including computer simulations, computer and/or video games, and basic control algorithms. In addition, they were asked to indicate the amount of learning they had in controls by choosing one of the statements in Table 6.1.

TABLE 6.1: Pre-assessment Survey - Amount of Controls Education

Amount of Learning in Basic Controls
I have no background knowledge in basic controls and algorithms.
I have had no classes which taught about basic control algorithms.
I have had no college-level classes which taught about basic control algorithms.
I have had one class which, among several other topics, taught about basic control algorithms.
I have had one class which taught mostly or exclusively about controls.
I have had more than one class which, among several other topics, taught about basic control algorithms.
I have had more than one class which taught mostly or exclusively about controls.

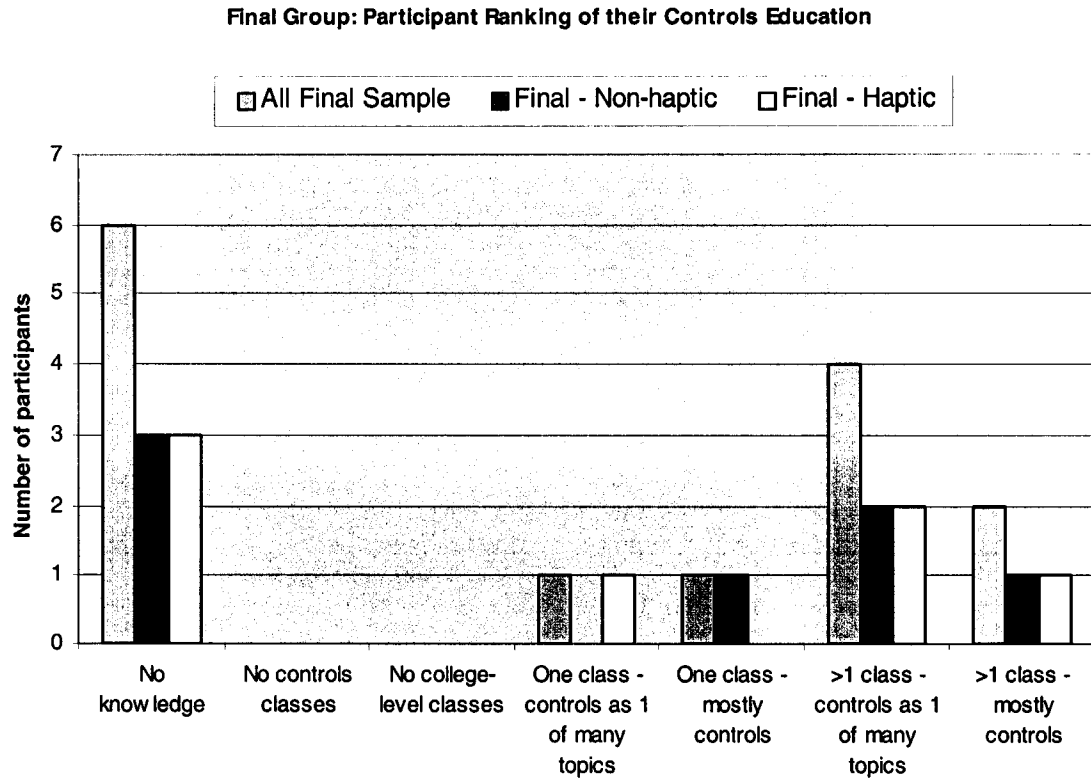


FIGURE 6.1: Final Test Group: Participant Ranking of their Education in Controls

TABLE 6.2: Final Test Group - Ranking of Background Knowledge/Experience (0=No Background, 5=Extensive Use)

	Mean	SD	Non-haptic Mean	Non-haptic SD	Haptic Mean	Haptic SD
Knowledge and use of computer simulations:	2.79	1.42	2.86	1.57	2.71	1.38
Knowledge and use of computer/video games:	3.71	0.73	3.57	0.53	3.86	0.90
Knowledge of basic control algorithms:	1.75	1.70	1.57	1.72	1.93	1.79

As can be seen in Figure 6.1, 6 of the 14 (~43%) participants in the final test group had no education or background knowledge in basic controls. In addition, the amount of controls education between the haptic-using sample and the non-haptic sample is very

similar. From Table 6.2, we can see that the final test group overall had a mean of 1.75 on a scale of 0 to 5 ($SD = 1.70$) for their knowledge of basic control algorithms at the start of the experiment. Independent t-tests showed no significant difference (all p 's > .05) between the haptic and non-haptic user groups in terms of their knowledge and use of computer simulations, computer or video games, or basic control algorithms.

As was mentioned previously, participants were allowed to take as much time as they wished to complete the different parts of the experiment. Figure 6.2 gives an overview of the mean amount of time that the final test group took in completing the pre- and post-assessment surveys and the worksheet.

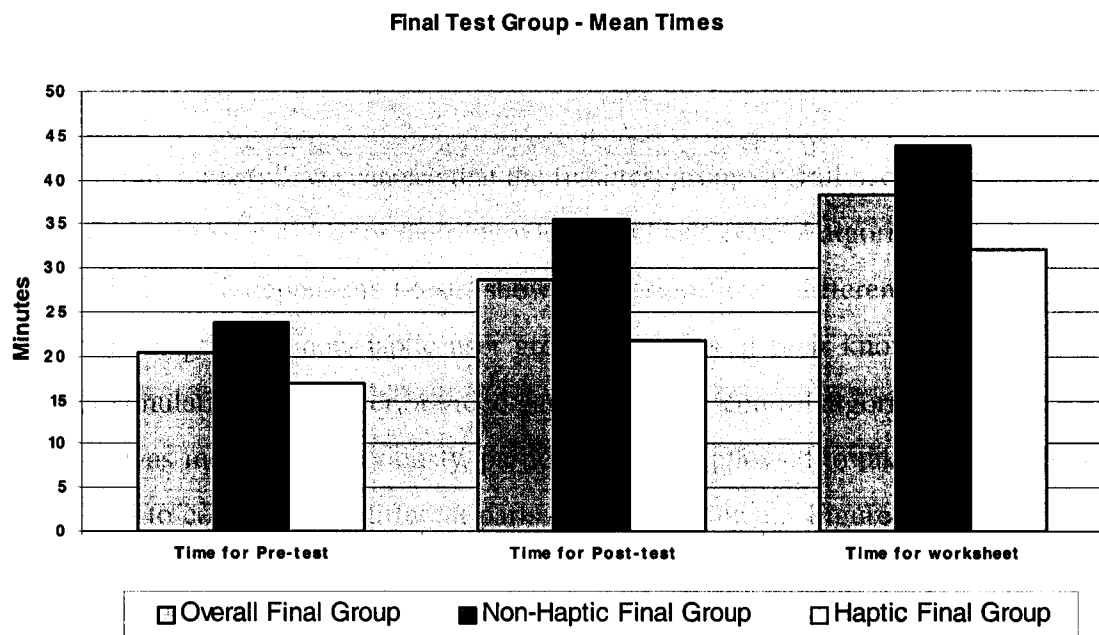


FIGURE 6.2: Final Sample - Mean Times for Completing Surveys and Worksheet

It should be noted that participants recorded their own times, and therefore strict accuracy is not assured, especially when recording the time that was taken to complete the worksheet, where it is likely that some participants wrote down the time when they began working with the learning software as their start time, and others wrote down the time when they actually began filling in the worksheet. It is interesting to note that the haptic users took noticeably less time for all these tasks.

6.0.2 Quantitative Evaluation

As was expected, for most test subjects, their knowledge assessment scores increased from the pre-assessment survey to the post-assessment survey for both the haptic and non-haptic sample groups. Interestingly, the average score increase for the pilot non-haptic group was somewhat larger than that for the pilot haptic group (Figure 6.3). Side by side comparison of the scores for the six pilot group subjects (Figure 6.4) shows that, while five of the six pilot subjects had increased survey scores, one in the haptic group actually had a lower score on the post-survey.

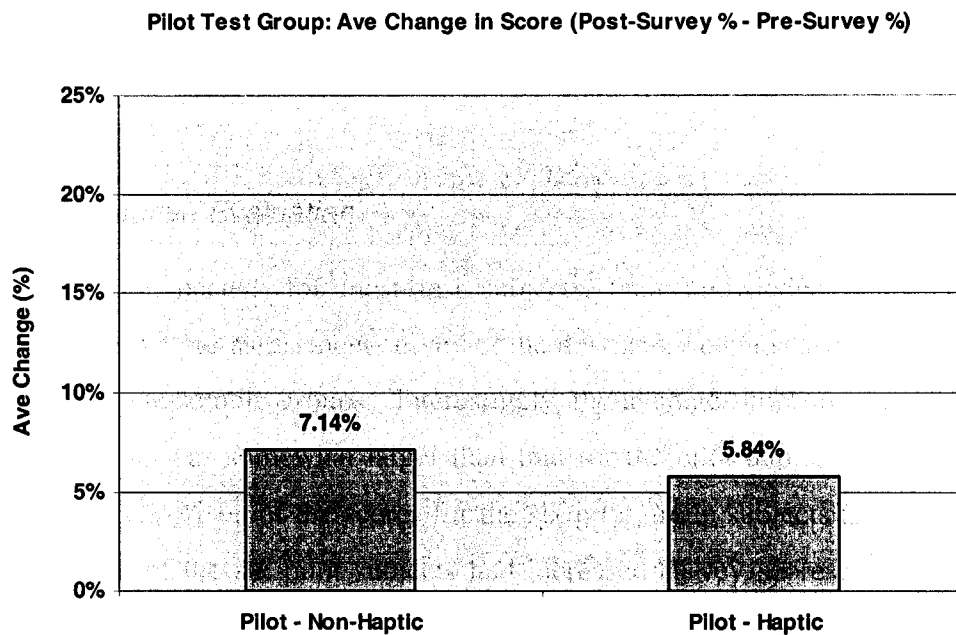


FIGURE 6.3: Pilot Group - Average Change in Score

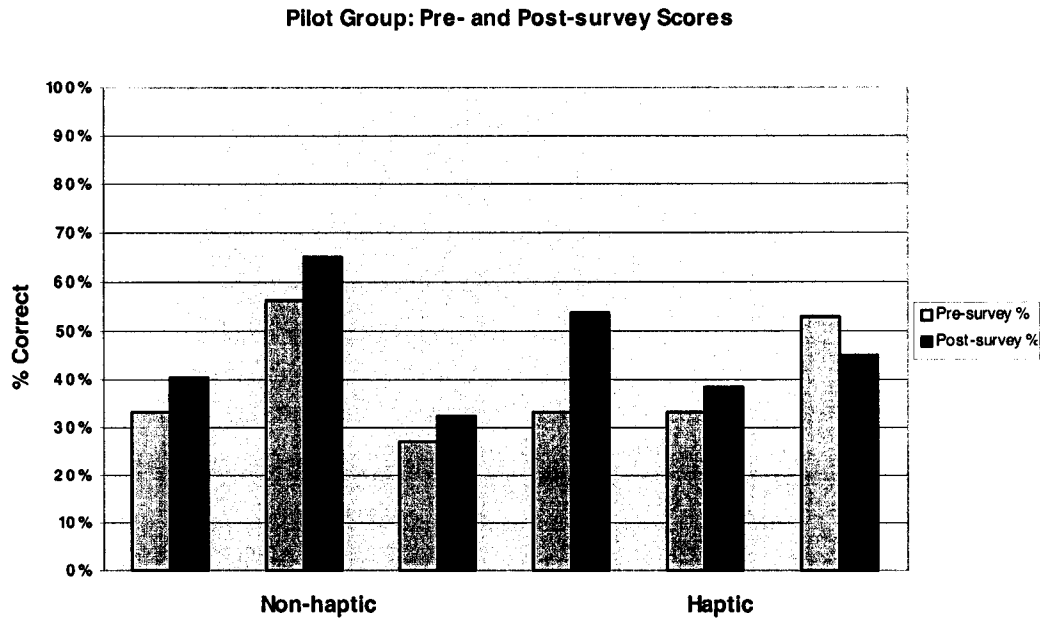


FIGURE 6.4: Pilot Group Pre- and Post-Survey Scores (%)

An independent groups two-sample t-test was used to compare the pre-survey knowledge assessment scores for the pilot haptic group with that of the pilot non-haptic group. This test was not found to be statistically significant ($p > .05$). This indicates that the difference between the mean pre-survey knowledge assessment scores for both the haptic and non-haptic samples was not statistically significant; i.e., it is likely that neither group had a greater average level of knowledge about basic controls at the start of the experiment. A t-test of the pilot sample's haptic and non-haptic groups' post-survey scores also did not indicate a statistically significant difference ($p > .05$).

It should be noted that the surveys and worksheet, as well as the software itself, were undergoing changes during the course of the pilot group testing; that and the small sample size make it inappropriate to attribute too much importance to the analysis of these scores.

Figure 6.5 shows the average change (in % correct) between the pre- and post-survey knowledge assessment scores for the final haptic and non-haptic test groups. An independent groups two-sample t-test was used to compare the pre-survey knowledge

assessment scores for the final haptic and non-haptic samples - this test was not found to be significant ($p > .05$). This indicates a strong probability that both groups had an equivalent level of basic controls knowledge at the start of the experiment.

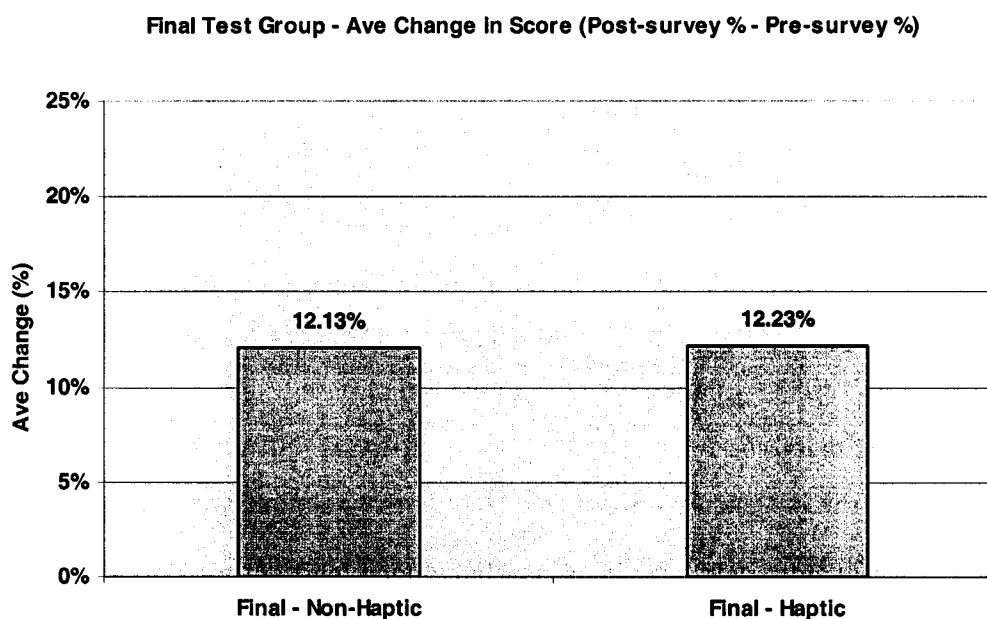


FIGURE 6.5: Final Test Group: Ave Change in Scores

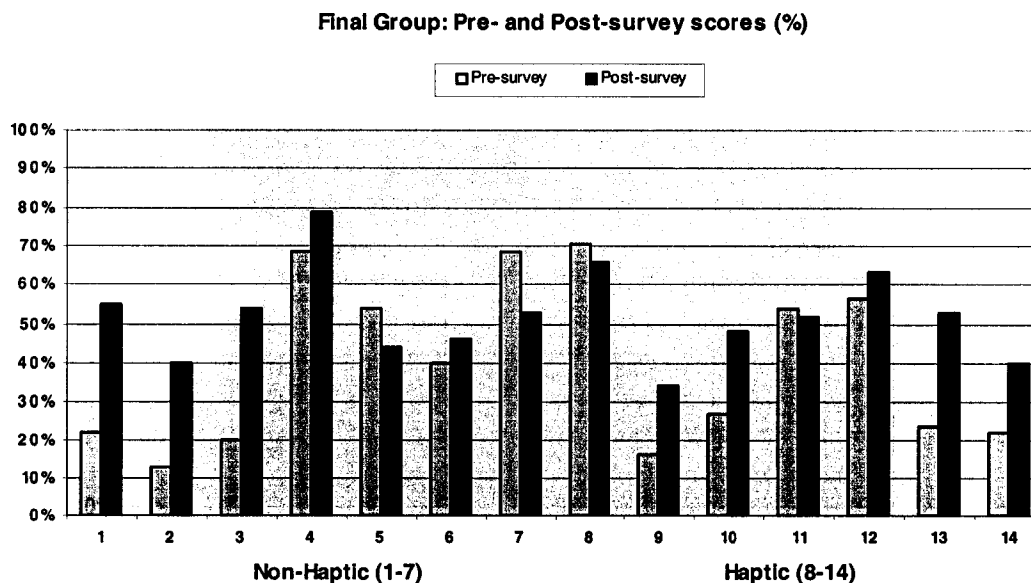


FIGURE 6.6: Final Test Group: Individual Pre- and Post-survey Scores (%)

A correlated groups t-test compared the mean pre-survey score with the mean post-survey score for all the final test sample participants. This test was found to be statistically significant, $t(13) = 2.16, p < .05$, suggesting that the knowledge assessment scores after using the simulation ($M = 0.519, SD = 0.117$) were higher for the population than the scores previous to using the simulation ($M = 0.398, SD = 0.213$). The strength of the relationship between simulation system use and assessment score was 0.382 as indexed by eta-squared, indicating that the percentage of variability in test scores that is associated with the software after the influence of individual differences has been removed is 38%. However, this omnibus test did not allow a comparison of the individual contributions of the different systems tested.

In order to examine the effectiveness of the visual-only versus the visual+haptic software systems, correlated groups t-tests were performed on each of the final non-haptic and haptic system user samples.

The correlated groups t-test for the non-haptic system users compared the mean pre-survey score with the mean post-survey score for those participants. This test was not found to be statistically significant ($t(6) = 1.59, p > .05$), which indicates that the increase in mean knowledge assessment scores may not be related to use of the learning software.

A correlated groups t-test comparing the mean pre-survey score with the mean post-survey score for the haptic system users was found to be statistically significant at an alpha level of .05, $t(6) = 2.60$, suggesting that the control knowledge assessment scores after using the multi-modal version of the simulation ($M = 0.509, SD = 0.115$) are reliably different than the scores prior to using the simulation ($M = 0.386, SD = 0.210$). The strength of the relationship between haptic system use and assessment score was 0.53 as indexed by eta-squared.

In order to examine the results more closely, the questions on the pre- and post-surveys were categorized as “Conceptual” and “Objective”, and these categories of questions were scored, and the results analyzed. The conceptual questions were those that attempted to evaluate the subject’s ideas about control algorithms and how they work - several of the questions asked the subject to describe what algorithm they would use and to explain why;

others had the subject try to evaluate which algorithm or type of control would work best for different applications. It should be noted that, because of the nature of these questions, several of them required subjective scoring by the researcher. The objective questions were multiple choice items.

Figure 6.7 shows that the average increase in the concept scores for the haptic group was greater than that for the non-haptic group. A correlated groups t-test was used to compare the mean pre-survey conceptual questions score ($M = 0.413$, $SD = 0.279$) with the mean post-survey conceptual questions score ($M = 0.606$, $SD = 0.101$). This test was found to be statistically significant at an alpha level of .05, $t(13) = 2.160$. This indicates that the participants' knowledge as measured by the knowledge assessment conceptual questions was significantly higher after using the simulation software. The strength of the relationship between simulation system use and conceptual question score was 0.397 as indexed by eta-squared - this indicates that the percentage of variability in test scores that is associated with the software after the influence of individual differences has been removed is about 40%.

A correlated t-test was used to compare the change in conceptual question scores for the non-haptic and haptic test groups, respectively. Neither t-test was statistically significant ($p > .05$). This indicates that we cannot reject the null hypothesis for either group. The eta-squared is 0.44, which would indicate a strong relationship if the t-value was significant. Because the t-value was non-significant but the eta-squared value was high, there is a possibility that the two variables are related and the sample size was just too small to give a significant result (Jaccard & Becker, 2002).

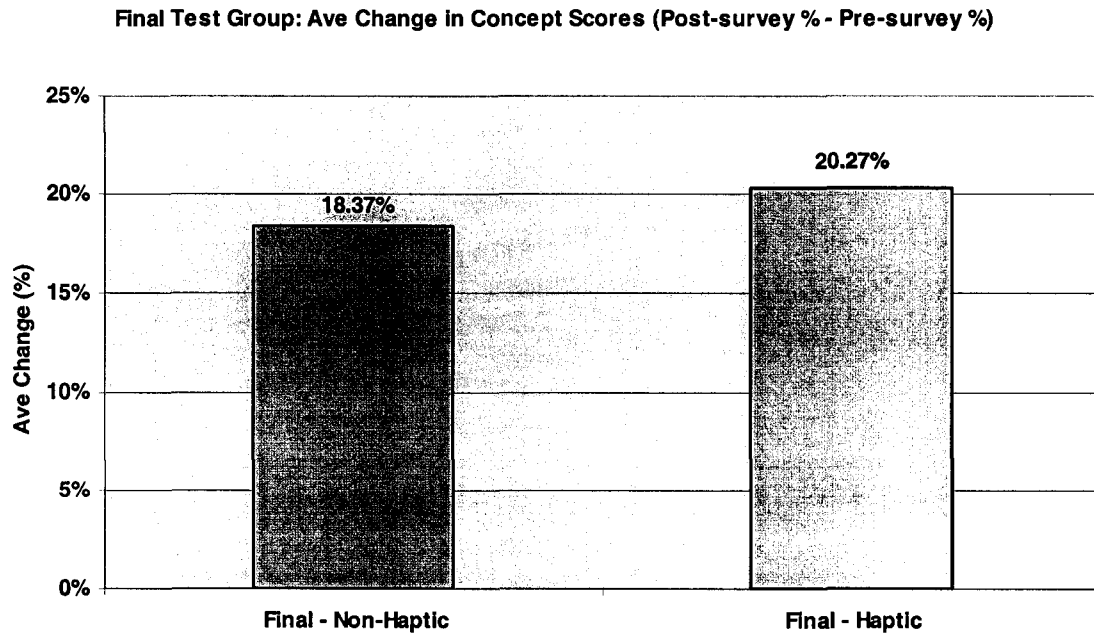


FIGURE 6.7: Final Test Group - Ave Change in Concept Question Scores

Interestingly, the objective question scores gave a different picture than the overall and conceptual question scores. As shown in Figure 6.8, the average change in score for the objective questions for the haptic group was not only lower than that of the non-haptic group, but actually negative. A correlated t-test of post-survey objective question scores for the final sample group indicates that the test is not significant at an alpha level of 0.05, and therefore the null hypothesis cannot be rejected. Correlated t-tests for the haptic and non-haptic groups' objective question scores, respectively, were not significant ($p > .05$). The eta-squared values were small for the change in objective scores for the haptic and non-haptic user samples, reinforcing the statistical decision not to reject the null hypothesis.

Final Test Group: Ave Change in Objective Question Scores

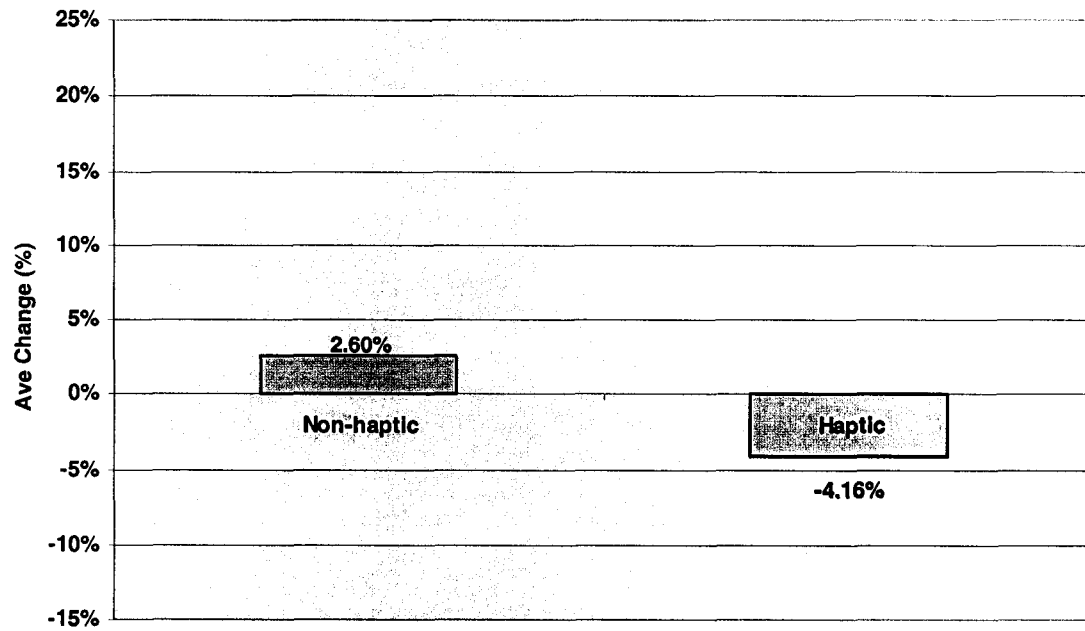


FIGURE 6.8: Final Test Group - Ave Change in Objective Question Scores

TABLE 6.3: Summary of Correlated t-test Results for Final Test Group

	Mean Diff	\hat{s}_D	t	critical t value	eta ²	Significant? (p < .05)
Overall Score - Final Group	0.122	0.043	2.837	+/-2.160	0.382	Y
Overall Score - Non-Haptic Group	0.121	0.076	1.592	+/-2.447	0.297	N
Overall Score - Haptic Group	0.122	0.047	2.596	+/-2.447	0.529	Y
Concept Score - Final Group	0.193	0.066	2.924	+/-2.160	0.397	Y
Concept Score - Non-Haptic Group	0.184	0.101	1.813	+/-2.447	0.354	N
Concept Score - Haptic Group	0.203	0.093	2.183	+/-2.447	0.443	N
Objective Questions - Final Group	-0.008	0.044	-0.182	+/-2.160	0.002	N

TABLE 6.3: Summary of Correlated t-test Results for Final Test Group

	Mean Diff	$\hat{s}_{\bar{D}}$	t	critical t value	eta ²	Significant? (p < .05)
Objective Questions - Non-Haptic Group	0.026	0.054	0.481	+/-2.447	0.037	N
Objective Questions - Haptic Group	-0.042	0.071	-0.591	+/-2.447	0.055	N

A summary of the t-test findings is shown in Table 6.3. The first column contains the mean difference between the pre- and post-survey knowledge assessment scores in percent for the designated sample. $\hat{s}_{\bar{D}}$ is an estimate of the standard deviation of the sampling distribution of the mean of difference scores, also called the standard error of the mean of difference scores. “The existence of a relationship between an independent variable and a dependent variable in the correlated groups case is tested by converting the observed mean difference in the sample to a t value and comparing this with the critical t values that define the rejection region” (p.305, Jaccard & Becker, 2002). In other words, if the t value for the sample results is less than the critical t value (or greater than the negative of the critical t value), we conclude that the difference is not statistically significant. The strength of the relationship is given by eta², and the final column in the table specifies whether the t-test results indicate a significant relationship.

The change in the overall knowledge assessment scores for the final test group was found to be statistically significant, as was the change in the conceptual knowledge assessment scores for that sample. When the overall knowledge assessment scores for the users of the haptic system were evaluated separately from the scores for the users of the visual-only system, we can see that the difference for the haptic users is significant at an alpha level of 0.05, while the difference for the non-haptic users does not meet the criteria for significance at that level.

However, on examining the difference in conceptual knowledge scores for the non-haptic users and haptic users as separate groups, we do not find a significant relation-

ship for the change for either group. Moreover, similar results are obtained when examining the difference in objective question scores, both in comparing within the non-haptic and haptic sample groups, and in examining the results for the final test group as a whole.

6.0.3 Qualitative Evaluation

In addition to the pre- and post-survey questions to evaluate the test subjects' knowledge and learning about basic control algorithms, the subjects were also asked to evaluate the software and give suggestions and feedback. This was done for both the haptic and non-haptic groups. Table 6.4 shows that, of the 20 total test subjects, 14 chose the descriptors "Useful" and "Enjoyable", and 15 chose "Interesting" to describe the software. It should also be noted that some of those who described the system as "Simplistic" wrote comments indicating that their interpretation of the word included the idea of "not overly complex". A side-by-side comparison between the haptic and non-haptic groups (Table 6.5) shows that, overall, fewer haptic users chose the positive descriptors except for "Interesting"; however, they didn't choose the negative or neutral descriptors either.

Table 6.6 shows how users of the visual+haptic and the visual-only interface ranked the software in terms of enjoyability, ease of use, usefulness in a classroom, and other evaluative statements. Independent group t-tests show no significant difference between the two groups for these evaluation scores.

TABLE 6.4: All Participants - Software Descriptors Selected

	Total # of Users
Useful	14
Boring	0
Simplistic	7
Enjoyable	14
Confusing	1
Okay	3
Interesting	15
Helpful	8

TABLE 6.5: Final Test Group - Software Descriptors Selected

	Total # of Users	# of Non-haptic Users	# of Haptic Users
Useful	11	6	5
Boring	0	0	0
Simplistic	6	4	2
Enjoyable	9	5	4
Confusing	1	1	0
Okay	3	2	1
Interesting	10	4	6
Helpful	5	3	2

TABLE 6.6: Final Test Group - Software Evaluation (1 = Not True, 5 = Very True)

Evaluation Statement	Non-haptic Mean	Non-haptic SD	Haptic Mean	Haptic SD
The software was useful for learning about basic control algorithms.	3.86	0.69	3.71	0.76
I feel that I now have a better understanding of the basic control algorithms and the effects of the gains.	3.43	0.79	4.00	1.15
The software was enjoyable to use.	4.57	0.53	4.86	0.38
I would find this software useful as a classroom tool for learning about controls.	4.00	0.58	4.43	0.79
The interface was easy to use.	4.29	0.76	4.43	0.53
The interface took some practice to get used to.	3.14	1.35	2.00	0.82
The interface was complicated, but useful after some practice.	2.29	1.11	1.71	0.76
The interface was difficult to learn and use.	1.43	0.53	1.14	0.38

TABLE 6.7: Joystick and Haptic Feedback Evaluation Scores (1 = Not True, 5 = Very True)

Evaluation Statement	Mean	SD
The joystick controls were easy to use.	4.43	0.79
The joystick controls were confusing.	2.14	1.21
In general, I used the mouse and menus more than the joystick controls.	1.29	0.76
The touch feedback was interesting to use and experience.	4.71	0.49

TABLE 6.7: Joystick and Haptic Feedback Evaluation Scores (1 = Not True, 5 = Very True)

Evaluation Statement	Mean	SD
The touch feedback made the simulation more enjoyable.	4.43	0.79
The touch feedback really didn't affect the experience.	1.71	1.11

TABLE 6.8: Final Test Group - Haptic Feedback Descriptors Selected

Descriptor	# of Users
Useful	5
Too strong	0
Too weak	0
Enjoyable	6
Distracting	0
Neutral	0
Interesting	7

6.1 Discussion

This research examined how the use of a multi-modal software learning tool would affect a user's knowledge and understanding of basic control algorithms. After taking a pre-assessment survey to give an indication of the participant's level of knowledge at the start of the experiment, the participant made use of either a visual-only or a multi-modal version of a computer-simulated control system while completing a worksheet. After a period of at least a day, participants took a post-assessment survey which was intended to evaluate the user's level of knowledge of basic controls and algorithms.

Statistical analyses were performed on the data from the pre- and post-surveys. In performing these analyses the following assumptions were made:

1. *The test subject sample is representative of the target population, i.e., undergraduate engineering students.* Actually, only 10 of the final sample were undergraduates, and only 9 were in engineering.
2. *The pre-survey knowledge assessment and post-survey knowledge assessment were equal or close to equal in difficulty.* While the researcher attempted to

make the assessments equally difficult by including parallel and similar questions on both, this assumption is unproved.

3. *Aside from use of the software, no learning in the subject area took place for any of the participants between the time they took the pre-test and the time they took the post-test.* The possibility exists that, after the first part of the experiment, some participants may have been motivated to check some of their answers or otherwise acquire more information about controls.

At the start of the experiment, the following results were predicted:

1. After use of the simulation software, participant performance on the post-survey knowledge assessment will be higher than that on the pre-survey knowledge assessment.
2. Users of the multi-modal (visual+haptic) system will show a greater increase in performance than users of the visual-only system.

6.1.1 Quantitative results

As the quantitative analysis shows, a significant relationship was found between the use of the simulation system and learning as measured by the change in overall and conceptual question scores between the pre-assessment and the post-assessment surveys. The strength of the relationship as indicated by eta-squared is suggestive of a significant effect; however, because of the small sample size, no large conclusions should be drawn. As was predicted, users of the multi-modal system showed a greater increase in their overall scores on the post-survey knowledge assessment than the users of the visual-only system. Interestingly, on examining the change in overall scores for the haptic group and the non-haptic group considered as separate sample groups, the change for the haptic group was found to be significant whereas the change for the non-haptic group was not. These results differ from those of the omnibus test and may in part demonstrate one of the weaknesses of using such a small sample size. The results do indicate that use of the multi-modal simulation system has a significant probability of increasing learning for the target

population, though the magnitude of the increase when compared to that of the users of the visual-only system, is quite small.

When examining the conceptual question scores, a significant relationship was observed between use of the software and increased conceptual learning; however, no significant relationship was observed for the specific type of system (visual-only or visual + haptic). We can posit that there is a significant probability of increased conceptual learning with use of the simulation, but which version is used may not affect the outcome. The eta-squared value for the final sample as a whole suggests a significant effect, but once again, the sample size prevents us from drawing strong conclusions. The eta-squared values for the haptic and non-haptic users suggest the possibility that a relationship may exist but that the sample size was too small to indicate a significant relationship (Jaccard & Becker, 2002). However, further research is needed to test for this possibility.

Testing the objective question scores did not show a significant change, either when examining the scores for the final test group as a whole, or when examining the scores for the non-haptic and haptic user groups. There are several possible explanations for these results, including the following:

- Confusion with terminology - As has been mentioned, participants with little or no background in controls were confused by terms such as “over-damped”, “under-damped”, and “critically damped”. Moreover, by introducing terms such as “over-correcting” and “under-correcting” in the hopes that these terms would be less confusing for the those participants, we suspect it may have increased the confusion for users with some knowledge of controls. At the very least, if the participant was reading quickly, it would be fairly easy to make mistakes such as reading the word “under-correcting” as “under-damped” and therefore choosing the wrong answer.
- Poorly-worded questions - Even if the participant understood the terminology, the mental picture of the system being presented in the question could be considerably different from that intended by the questioner. Without interviewing at

least a few of the participants, it should not be assumed that most or all of them understood what was being asked.

- Participant haste - If a participant was trying to finish quickly, he or she might not read or think through the questions carefully, or might pick the first reasonable answer without reading through all the possible answers.

One item of interest is the amount of time that the participants took to complete the different parts of the experiment (i.e., the pre-test, post-test, and the worksheet) (see Figure 6.2), particularly the observation that the users of the haptic system on average took less time to complete all the tasks. While we do not know why this occurred, we can speculate that possibly more of the haptic users had taken or been exposed to the embedded controls course and/or the Smart Car system on which the simulation was based. This familiarity with the goals of the simulated car could then aid those users in learning to use the software and allow them to complete the worksheet more quickly. Moreover, because the pre- and post-surveys were prepared by someone with extensive experience with the course, it is possible that the structure and terminology used in the questions might be more understandable to a user who had been involved with the course. Participants were not explicitly asked if they had taken the embedded controls course or been involved with it.

Another possibility is that, while both groups had equivalent scores in their knowledge and use of computer or video games, the haptic group might have had members with more experience using haptic devices with these games, or with other applications. None of the participants were asked about their experience or knowledge of haptic devices.

To date, the vast majority of research in incorporating haptics as part of the human-computer interface has been heavily focused on the “how” aspect. The challenges implicit in including the tactile sense as part of the interaction has caused many developers to devote their time and efforts to implementing and testing different hardware configurations and software architectures, haptic rendering algorithms, and other aspects of using this type of technology. As is often the case in a new field of inquiry, the emphasis is on

showing that something can be done with less attention paid as to whether it ought to be or how effective it is. Thus, most of the available research is of the “this is what we wanted to do, this is why it was difficult, this is how we did it (or did part of it, or made a start at doing it)” variety. Evaluation of these efforts has usually been in the form of qualitative measures or feedback, seeking to identify increased interest or better attitudes or perceived understanding.

There are, of course, exceptions to this, especially in the use of haptic technology for identification or notification tasks (Hughes & Forrest, 1996; McLaughlin & Orenstein, 1997; Nyarko et al., 2002); several researchers have demonstrated quantitative improvement in performance measures when including a haptic component to aid in identifying a particular feature or aspect, usually in a visually complex or cluttered virtual environment. Similarly, Sklar and Sarter (1999) have shown statistically significant results when using a haptic device for notification of important events.

Another notable exception is the research in immersive and virtual-reality type applications, where experiments have generally supported the proposition that haptic feedback is helpful and possibly even vital for an effective experience.

In the area of educational applications, however, quantitative evaluations of multi-modal interfaces including haptic components are relatively few, and have produced mixed results. While Jones et al.’s (2002) study using the NanoManipulator demonstrated significantly better attitudes from the students using the haptic-enhanced interface, their results also only showed a “slight difference” in the students’ conceptions of the subject matter (viruses) as demonstrated by clay models that they made. The experiments by Lawrence et al. (2000) using a haptic display to help users experience fluid flow data gave “encouraging” initial test results, but more definite results have not yet been demonstrated. Nesbitt et al.’s (2001) study using haptic technology to display blast furnace data produced results that were inconclusive as to the value and effectiveness of the haptic display.

Certain results of this research support the hypothesis that a multi-modal interaction which includes a haptic component will be useful and effective for a learning applica-

tion with quantitatively measurable and statistically significant results, though the magnitude of the greater benefits for the multi-modal system users were quite small. Like Hughes, Nesbitt, and Lawrence, the haptic display is used for experiential data perceptualization. However, unlike those studies, the data being displayed is relatively simple and the tactile effects are reinforcements of the visual display, a use conceptually similar to the NanoManipulator, and the haptic-enhanced physics demonstrations by Williams and his colleagues (Williams et al., 2000).

Other of our results are inconclusive in regards to the effectiveness of the multi-modal system. However, as this is one of a fairly small number of studies that have produced significant quantitative results, we are convinced of the benefits of continuing and expanding on this research.

6.1.2 Qualitative results

Survey questions which attempted to gauge the users' reactions to the system and other qualitative measures show an overall positive reaction from the experiment participants. The majority of participants in the final test group evaluated the system at either "4" or "5" (where 5=Very True) for the following evaluation statements:

- The software was useful for learning about basic control algorithms.
- I feel that I now have a better understanding of the basic control algorithms and the effects of the gains.
- I would find this software useful as a classroom tool for learning about basic controls.
- The interface was easy to use.

and all the final group participants gave a "4" or "5" ranking to the statement:

- The software was enjoyable to use.

In addition, most of the users of the haptic system agreed that the joystick was easy to use, and that the haptic feedback was interesting and enjoyable. The results of independent t-tests on the mean scores for the positive reaction statements gave no indication that the haptic group had a significantly better experience or reaction than the non-haptic

group. It should be noted that no strong conclusions can be drawn from these results, as none of the test subjects had the opportunity to use both the multi-modal and the visual-only systems.

There are several possible reasons for the results:

- **Small sample size:** With only 14 participants in the final test group, and 20 overall, it is difficult to draw strong conclusions from the results of the experiment. Future researchers will most likely wish to experiment with a much larger sample.
- **No basis for comparison:** Because each participant only used one version of the simulation, there is no way to evaluate whether the visual+haptic version was preferred over the visual-only system, or vice versa.
- **Insufficient background knowledge:** A significant portion of the participants had little or no background in the subject area which the simulation was intended to demonstrate. It is possible that a user group with more understanding of the subject would find the augmented display more helpful.
- **Ineffective haptic design:** The art of designing useful and informative haptic effects is still a very new field, and one for which there are few guidelines. It is quite possible that a different, or differently implemented, haptic effect would be more effective in communicating information to the users.

Nonetheless, the overall positive reactions of all the users to the software, and the positive reactions to the haptic system in particular, give us strong motivation to continue developing and experimenting with multi-modal display systems.

6.1.3 *Cautions about the internal validity of the experiment:*

1. There was no control sample for the experiment; i.e., there were no users who took the pre- and post-assessment surveys without using the simulation system.

The possibility exists that just taking the surveys would result in increased learning.

2. The small number of participants in the experiment makes it difficult to achieve statistically significant results.
3. From observation, some users will make the general assumption that a more complex algorithm will work better with more difficult or complex systems. Thus, what appears to be learning may just be an assumption based on little data or a preconception.
4. Test subjects were told at the start of the study that they should feel free to leave questions blank or indicate where they didn't know the answer to questions on the surveys. Nonetheless, the subjects clearly knew that they were supposed to have learned something, and therefore might have made more of an attempt to answer the questions on the post-survey, which could account for some of the score increases.

6.2 General observations

As the research was conducted, certain ideas emerged with great regularity, regardless of which version of the simulation the subjects were using. The data analysis only confirmed what observation and conversation with the test subjects had strongly indicated were important aspects which need to be considered not only in examining this research data, but also for planning future work.

- *The need for context:* While the overall opinion of the students was that the simulation overall was useful and informative, it was also commonly noted that it would have been more helpful to have some background and/or preliminary instruction in the subject area. This was especially evident with survey questions that used controls terminology, such as “overdamped”, “underdamped”, “critically damped”, etc. One subject apparently did not understand the purpose of the study, although it was specified in the preliminary contact e-mails, and the forms the subjects filled out, and he assumed that the purpose was the development of an automatically-driving car for highway use. Another subject con-

cluded that joystick movement was overall a negative thing, but could not figure out whether it was better to have many small adjustments, or fewer and larger adjustments.

Thus, while the subjects still felt that they were gaining an understanding of basic controls, nearly all expressed the opinion that it would be more beneficial to them if they had some sort of additional or background instruction that would provide them with more understanding of the way in which the control algorithms work, and the terminology involved.

As was noted in Section 5.1.3, some of this was alleviated by modifying the surveys to include questions using more easily understood terminology, such as “over-correcting” rather than “underdamped”. Some questions with controls terminology were left in the surveys, since some of the subjects did have some background in controls. In addition, the formulas for the different control algorithms were displayed at the bottom of the window, along with a key defining the different terms.

Nonetheless, it seems like a strong possibility that the most effective use of this simulation would be in a classroom setting, or as part of a learning module, which would allow the user to gain some understanding of the subject area before using the system, and therefore hopefully increase the user’s learning after using the software.

- *The need for information:* Several suggestions were made requesting additional informational options for the user as he or she is making use of the software. These included more detailed visual displays of the steering, graphing of various data, and “Help” pages with additional information about controls. Moreover, some of the test subjects appeared to become frustrated when they could not tune the controllers so that the car could follow the track (this especially occurred with the difficult track). Possibly some additional display options (either visual or otherwise) could be used to provide clearer indications of why the system was not performing as desired.

- *The need for speed:* While the movement of the simulation car with a speed setting of approximately 55-60 most closely approximates that of the embedded control car which is the basis of the simulation, most of the subjects quickly set the speed to maximum and attempted to have the car following the different tracks at that speed. One of the

earliest feedback suggestions was that the speed control increments be increased (they started out at ± 1 , and then were changed to ± 5), and that the maximum speed be increased. Possibly the increased speed made the simulation more “game-like” and therefore more enjoyable for the test subjects.

It should be noted that, even though the simulation was modified to run as fast as possible on the given computer (that is, with no delay between iterations), it could be observed that many users would have preferred an even faster car. Given more time, this could possibly have been accomplished by optimizing the code; or just by using a faster computer.

6.3 Participant Feedback and Suggestions

Overall, the participant reactions to the system, both haptic and non-haptic, were very positive. In their comments, participants often called the software “useful”, “enjoyable”, “good”, “fun”, “interesting”, and “helpful”. Several participants noted that it was easy to use, and felt that it would be a good learning tool.

The major suggestion, made by nearly all participants, was a desire for more knowledge or instruction in basic controls. Representative comments include, “I’ve never been introduced to basic controls algorithms. Terms and phrases like P, PI, PID controllers were new,” “I wish I knew more about control algorithms going into this,” and “it would have been useful to know what they [the control algorithms] were supposed to do, then be able to see this demonstrated by the software.” As was noted in Section 6.0.3, a large number of the participants felt that the system would be useful in a classroom setting, which would, of course, provide the desired background knowledge.

Another common recommendation was for additional information about the car’s performance as the system was being used. Suggestions included adding graphs of the gains, a visual display of the steering wheel, a display of the error, and block diagrams and/or other indicators for whether the system is underdamped, overdamped, or critically damped.

The participants were also eager for additional options to make the system more interesting, useful, or challenging. Several users suggested the possibility of designing their own tracks for the car to run on. Others wanted longer tracks, changes in terrain such as ramps, and the ability to enter their own control algorithms. Another user wanted to be able to “undo” a portion of the track, change the control setting, and then retrace that portion. And, as has been mentioned before, many participants would have liked a significantly faster maximum speed.

Not all the comments were positive - one user felt that having just a “picture” lacked the interest of a physical car. Several users were confused by the controls terminology (although this only appeared on the surveys, not in the simulation system itself), and some were unsure how to balance the goals of speed and smooth steering.

In terms of the haptic feedback, the comments were largely positive. One user called it, “definitely the best part”, and another noted that, “one gets [a] really good feel about how the control system responds depending on the value of the constants!” In general, participants seemed to feel that it made the system more fun and interesting to use. One user felt that it was tiring on the wrist after a while, and another would have liked the option of controlling the strength of the effect.

As can be seen even in just this brief summary of user feedback, the participants were interested and excited about the possibilities of the system, and its usefulness for learning about a largely unfamiliar topic. Their main suggestions involved additional information, either about the subject area in general, or about the performance of the simulation system, or both. In addition, many of the users would have liked additional options and challenges in the system. The effect of the haptic display on the qualitative experience is inconclusive, and one might hypothesize that the effectiveness of the haptic information may vary depending on the user’s preferences and learning style.

6.4 Fundamental Contributions

- The development of a computer-based control system simulation with the option for visual-only or visual+haptic display. This system can be used to effectively

demonstrate control system behavior, algorithm performance, and the effect of gain adjustment, and could also be used as the basis for other system demonstrations.

- Developed and tested a multi-threaded, virtual environment and haptic display-driven system architecture for use with the simulation, and which can run on commonly available and relatively inexpensive computer and display components.
- Quantitatively tested and supported the hypothesis that multi-modal and multi-sensory information delivery is effective for information delivery in a learning application. In addition, some results supported the hypothesis that multi-modal delivery is more effective than single-mode; however, other results were inconclusive, and therefore further study is needed.
- Qualitative evaluation of test subject reactions to the simulation system supported the hypothesis that a multi-modal system will be positively received by users, but more research is needed to prove or disprove the general assumption that such a system will be preferred over a purely visual system.

6.5 Future Research Possibilities

6.5.1 Classroom use

As part of the motivation for this research came from experience with an embedded controls course, it is only fitting that consideration should be given to ways in which the simulation system could be used as a learning tool within or auxiliary to the course. As has been mentioned, Proportional and Proportional+Integral (PI) controllers are important supporting concepts that the students are introduced to while building and developing their Smart Cars.

The first six or so weeks of the course are spent on an introductory lab project (usually a simple game), and topics such as number systems, analog to digital conversion,

and basic programming. By week 7, the students have been given an overview of the Smart Car target system and its subsystems, and in week 8, they are introduced to proportional feedback control. Over approximately 6 of the subsequent class sessions, the students, through several lectures, a worksheet, and their hands-on work in the lab, learn to use proportional and PI control algorithms to control their car's steering and drive speed.

The simulation system created for this research could easily be integrated into the structure of the course, and could be used both for illustrative purposes by the instructor, and as an interactive learning tool by the students. For example, after the students have been introduced to the proportional control algorithm in a lecture, they could be given a short exercise in which they use the simulation to experiment with different proportional gains and observe their effectiveness on different tracks and with the simulated car traveling at different speeds. Even though the real-world Smart Cars would not be able to be run on a track for several weeks yet, the simulation could help the students start to think in terms of how their controller might have to be tuned differently depending on how difficult the track is and how fast they want their car to go.

Within the course, PI control is used for drive motor (speed) control, which differs from the simulated Smart Car, where the control algorithms are only used for steering. Nonetheless, the software could still serve as an introduction to PI control, as well as giving the students an opportunity to apply that knowledge to a different system when they go to develop the drive motor subsystem.

While the embedded control course does not really include instruction on PID control, it is introduced in the lab manual. Depending on the goals of the instructor, a brief simulation exercise which includes use of the PID controller could be offered for the sake of completeness or to impress upon the students that control options other than the ones they are using are available. Even if PID control is not included in the curriculum, the option would still be available to interested students, possibly even encouraging some of them to add a derivative term to their controller as part of an enhancement of the basic car.

6.5.2 *Research ideas*

Clearly it would be beneficial to run the experiment again with more participants, in order to produce more significant results. In addition, a redesign of the experiment could eliminate some of the weaknesses of the present design, thus strengthening the validity of the results. Such a redesign might include adding a control group, providing participants with a base level of instruction, and reducing or eliminating the confusing terminology.

If an experiment could be designed that could take advantage of the fact that this simulation could have applicability to several of the undergraduate engineering courses (most notably the Embedded Control Laboratory course, but other courses address basic controls as well), that could provide both a larger pool of test subjects and one more in line with the target population. Moreover, if simulation use could be incorporated as part of a larger learning sequence, this would help to reduce the confusion that some of the participants experienced as the result of unfamiliar terminology as well as responding to the stated desire for more background knowledge about the subject.

While it would be tempting to assign a course section to be one of each of the test groups (visual-only system, multi-modal system, control group), this would require the researchers to take into account the differences between instructors, not to mention the possibility of perceived coercion and fairness issues mentioned below. A better procedure might be to advertise for volunteers within all sections of the course, providing a financial incentive (as was done for this experiment), and randomly assigning interested students to the different test groups.

It should be noted that performing an experiment where different versions of the software are only available to certain students in a course raises issues of fairness, since the test subjects who are using one version may have an advantage in terms of learning over those using a different version and/or over a control group. One way to address this might be to time the experiment so that it is completed before any exam that will cover the material, and leaving enough time before the exam so that any test subjects who wish to can then make use of other versions of the system.

Including some sort of instruction component as part of the experiment could also provide the opportunity for testing whether the students are making connections between that instruction and what they are observing and learning from the software. One way of examining the learning progression of the test subjects would be to administer a sequence of knowledge assessments during the course of the experiment.

Improvements to the pre- and post-assessment surveys could be made by having them evaluated by both experienced instructors and students, and students with little or no controls background. In addition, the researchers could determine more specifically what controls concepts they want the participants to learn, and use different types of questions addressing the same concept to better determine whether and how well the concept was learned. More useful qualitative assessment results could be obtained by having a group of experiment participants who make use of both versions of the simulation.

Useful information could also be obtained by observing how the participants make use of the system; for example, noting how long a user experiments with the software before starting on the worksheet, or whether and in what ways the user takes advantage of simulation capabilities that are not possible (or at least are very difficult) for the real world Smart Car. One example of this is when the user changes the type of control algorithm “on the fly” depending on whether the simulated car is on an easier or more difficult portion of the track.

The possibility also exists to expand the simulation (including the haptic component) to be a general-use tool for demonstrating control algorithms as applied to various problems or devices. The effects of different control strategies could be demonstrated and experienced for any number of applications, such as a walking robot, an automatic tracking device, a balancing mechanism, etc. Moreover, additional research could be performed in incorporating other modalities, such as a more informative audio component.

Research possibilities also exist for applying the multi-modal display system to other educational topics. For example, perhaps students could navigate their way through a virtual electrical circuit, experiencing voltage differences, current flow, and resistance.

A more immersive learning environment is another avenue for exploration, such as a small 180-degree projection display.

Another application would be to use the simulation to give students an experience similar to an automotive drive by wire system. As was described in Section 1.2, drive by wire cars replace mechanical linkages with sensors and electronically activated steering, braking, and/or acceleration. Thus, while the car is improved in terms of reliability and efficiency, the driver is left without the tactile and kinesthetic feedback that is often a useful and important part of driving. Haptic devices can be used to deliver this feedback to the user. While the research system in its present form only provides feedback on steering performance, the potential exists to add additional haptic effects to allow the user to experience the simulated car's acceleration or slowing down. Moreover, with a communication link such as an RF transmitter/receiver, such a system could give the user a drive by wire-type experience not only with the simulated car, but also with a real world Smart Car, and in doing so, encourage the students to think in terms of other real world systems, and other applications of haptic technology.

As several users suggested, additional information and visualization tools could be added to the system to aid in the learning process and give the users more information to work with – these might include recording and showing the path taken by the virtual vehicle, auditory cues, and allowing the user to specify transfer functions to be applied to the data to help them isolate interesting characteristics or behaviors.

Additional complexity and virtual components could also be developed for use with the simulation and learning system, allowing the students to add more sensors to their virtual car, to set up walls or obstacles on or around the path, and to feel collisions and other actions taking place in the virtual world. Similarly, the users could be given customization options, allowing them to change system parameters such as maximum and minimum force displayed, and sensitivity.

Experiments with other types of haptic devices and feedback, especially other commercial products such as haptic mice, could prove useful in determining what sorts of feedback are most effective, and which low-end devices can prove to be cost-effective and

useful for certain educational purposes. The possibility of using more than one type of haptic feedback at the same time is also worth exploring, either for reinforcing the information being delivered, or to increase the number of information channels and therefore the amount of data being experienced by the user.

Another question to be addressed concerns if and how much students with different learning styles may benefit from the use of multi-modal display technology (Carver et al., 1999). Because this application displays the control system's behavior in an experiential manner, one would think that students whose learning styles are "active" and "sensing" would benefit most from the display. However, we have no direct proof of that, and testing could show that, for example, the learning experience does not provide a satisfying level of realism for a sensing learner. Moreover, it is possible that the experiential aspect of the haptic display may be distracting to students with other learning styles.

REFERENCES

1. Alps Electric Co., Ltd., "ALPS Now Developing Proprietary Drive-by-Wire System", http://www.alps.co.jp/press/new2004/0426_e.htm, 2004.
2. Ando, Hideyuki, Miki, Takeshi, Inami, Masahiko, and Maeda, Taro, "The Nail-Mounted Tactile Display for the behavior modeling", *SIGGRAPH Conference Abstracts and Applications, Computer Graphics Annual Conference Series*, p264, 2002.
3. Balakrishnan, Ravin, Ware, Colin, and Smith, Tim, "Virtual Hand Tool with Force Feedback", *Conference Companion – CHI 94*, pp83-84, Boston, 1994.
4. Balaniuk, Remis, and Laugier, Christian, "Haptic Interfaces in Generic Virtual Reality Systems", *Proceedings of the 2000 IEEE/RSI International Conference on Intelligent Robots and Systems*, pp1310-1315, 2000.
5. Brauer, Karl, "Why Drive-by-Wire?", <http://www.edmunds.com/news/innovations/articles/43033/article.html>, 11/29/2000.
6. Brooks Jr., Frederick P., Ouh-Young, Ming, Batter, James J., Kilpatrick, P. Jerome, "Project GROPE – Haptic Displays for Scientific Visualization", *Computer Graphics*, Vol. 24, Number 4, August 1990.
7. Burdea, Grigore C., *Force and Touch Feedback for Virtual Reality*, John Wiley & Sons, Inc., New York, 1996.
8. Campbell, Christopher S., Zhai, Shumin, May, Kim W., Maglio, Paul P., "What You Feel Must Be What You See: Adding Tactile Feedback to the Trackpoint", *Human-Computer Interaction - Proceedings of INTERACT '99*, IOS Press, pp383-390, 1999.
9. Carignan, Craig R., and Cleary, Kevin R., "Closed-Loop Force Control for Haptic Simulation of Virtual Environments", *Haptics-e*, Vol. 1, Number 2, February 2000.
10. Carver, Curtis A., Howard, Richard A., and Lane, William D., "Enhancing Student Learning Through Hypermedia Courseware and Incorporation of Student Learning Styles", *IEEE Transactions on Education*, Vol. 42, Number 1, February 1999.
11. Chen, Timothy, Fels, Sidney, Schiphorst, Thecla, "FlowField: Investigating the Semantics of Caress", *SIGGRAPH Conference Abstracts and Applications, Computer Graphics Annual Conference Series*, p185, 2002.
12. Clark, Jason, "May the Force Feedback Be with You: Grappling with DirectX and DirectInput", <http://www.microsoft.com/msj/0298/force.aspx>, February 1998.
13. Davis, Elizabeth T., Scott, Kevin, Pair, Jarrell, Hodges, Larry F., Oliverio, James, "Can Audio Enhance Visual Perception and Performance in a Virtual Environment?", Lec-

- ture, *Virtualnomics - Virtual Environments Technical Group of the HFES*, September 1999.
14. Dean, Kevin L., Asay-Davis, Xylar S., Finn, Evan M., Foley, Tim, Friesner, Jeremy A., Imai, Yo, Naylor, Bret J., Wustner, Sarah R., Fisher, Scott S., Wilson, Kent R., "Virtual Explorer: Interactive Virtual Environment for Education", *Presence: Teleoperators & Virtual Environments*, Vol. 9, Issue 6, pp505-520, Dec. 2000.
 15. Durbeck, Lisa J. K., Macias, Nicholas J., Weinstein, David M., Johnson, Chris R., Hollerbach, John M., "SCIRun Haptic Display for Scientific Visualization", *PHAN-ToM User's Group*, 1998.
 16. Elmes, David G., Kantowitz, Barry H., Roediger III, Henry L., *Research Methods in Psychology - 7th Edition*, Wadsworth Thomson Learning, Belmont, CA, 2003.
 17. FEMur (Author unknown), "Force-Displacement Relationship for Single Degree of Freedom (SDOF) Nonlinear Spring", <http://femur.wpi.edu/Learning-Modules/The-Basic-Finite-Element-Equation/sdof2.html>, WPI, 1996.
 18. Fitzmaurice, George W., "Graspable User Interfaces" A thesis submitted in conformity with the requirements of the Degree of Doctor of Philosophy, Graduate Department of Computer Science, University of Toronto, 1996.
 19. Fowlkes, Jennifer, Durlach, Paula J., Drexler, Julie, Daly, Jason, Alberdeston, Roberto, and Metevier, Chris, "Optimizing Haptics Perceptions for Advanced Army Training Systems: Impacts on Performance", <http://www.asc2002.com/manuscripts/M/MO-01.PDF>, 2002.
 20. Gentner, Don, and Nielson, Jakob, "The Anti-Mac Interface", *Communications of the ACM*, Vol. 39, Number 8, August 1996.
 21. Gizmo Highway, "Drive by Wire cars", http://www.gizmohighway.com/autos/drive_by_wire.htm, 2004.
 22. Global Haptics, Inc., <http://www.globalhaptics.com>, 2003.
 23. Graham-Rose, Duncan, "Feel it in your fingers", *New Scientist*, No. 2281, March 10, 2001.
 24. Harris, Tom, "How GM's Hy-wire Works", <http://auto.howstuffworks.com/hy-wire.htm/printable>, 2004.
 25. Hikiji, Riku, and Hashimoto, Shuji, "Hand-Shaped Force Interface for Human-Cooperative Mobile Robot", *Proceedings of First International Workshop on Haptic Human-Computer Interaction*, pp113-118, 2000.
 26. Hughes, R.G., and Forrest, A.R., "Perceptualisation using a Tactile Mouse", *IEEE Visualization - Proceedings of the 7th Conference on Visualization '96*, pp181ff, 1996.

27. Immersion Medical, <http://www.immersion.com/products/medical/overview.shtml>, 2002.
28. Izadi-Zamanabadi, Roozbeh, Supervisor, "Fault-tolerant drive-by-wire system - A hybrid approach", <http://www.control.auc.dk/~riz/fault-tolerant-dist-steering-system.pdf>, Aalborg University, Denmark.
29. Jaccard, James, and Becker, Michael A., *Statistics for the Behavioral Sciences - 4th Edition*, Wadsworth Thomson Learning, Belmont, CA, 2002.
30. Johansson, Anders J., and Linde, Joakim, "Using Simple Force Feedback Mechanisms as Haptic Visualization Tools", <http://www.tde.lth.se/home/ajn/publications/IMTC99.pdf>, 1999.
31. Jones, M. Gail, Bokinsky, Alexandra, Andre, Thomas, Kubasco, Dennis, Negishi, Atsuko, Taylor, Russell, and Superfine, Richard, "NanoManipulator Applications in Education: The Impact of Haptic Experiences on Students' Attitudes and Concepts", *Proceedings of the 10th Symposium On Haptic Interfaces For Virtual Environments and Teleoperator Systems*, IEEE Computer Society, 2002.
32. Katz, David (Edited and trans. by Lester E. Krueger), *The World of Touch*, Lawrence Erlbaum Associates, Publishers, Hillsdale, New Jersey, 1989.
33. LaMothe, Andre, *Tricks of the Windows Game Programming Gurus*, Sams Publishing, Indianapolis, Indiana, 2002.
34. Lawrence, Dale A., Lee, Christopher D., Pao, Lucy Y., Novoselov, Roman Y., "Shock and Vortex Visualization Using a Combined Visual/Haptic Interface", *Proceedings of the IEEE Conference on Visualization and Computer Graphics*, pp131-137, 548, 2000.
35. Leach, Geoff, Al-Qaimari, Ghassan, Grieve, Mark, Jinks, Noel, McKay, Cameron, "Elements of a Three-dimensional Graphical User Interface", <http://goanna.cs.rmit.edu.au/~gl/research/HCC/interact97.html> , 1997.
36. Lee, Adrienne Y., and Bowers, Alexia N., "The Effect of Multimedia Components on Learning", *Proceedings of the Human Factors and Ergonomics Society 41st Annual Meeting*, pp340-344, 1997.
37. Liu, Yung-Ching, "Comparative study of the effect of auditory, visual and multimodality displays on drivers' performance in advanced traveler information systems", *Ergonomics*, Vol. 44, Number 4, pp425-442, 2001.
38. McLaughlin, J. P., Orenstein, B. J., "Haptic Rendering of 3D Seismic Data", *Proceedings of the Second PHANTOM Users Group Workshop*, SensAble Technologies, Inc., October 1997.
39. MacLean, Karon E., Snibbe, Scott S., Shaw, Robert S., "An Architecture for Haptic Control of Media", *Proceeding of the ASME Dynamic Systems and Control Division: 1999 International Mechanical Engineering Congress and Exposition, Eighth Annual*

Symposium on Haptic Interfaces for Virtual Environments and Teleoperator Systems, 1999.

40. MacLean, Karon E., "Designing with Haptic Feedback", *Symposium on Haptic Feedback in the Proceedings of IEEE Robotics and Automation (ICRA 2000)*, 2000.
41. MacLean, Karon E., Shaver, Michael J., Pai, Dinesh K., "Handheld Haptics: A USB Media Controller with Force Sensing", *Proceedings of the 10th Symposium On Haptic Interfaces For Virtual Environment & Teleoperator Systems*, pp311-318, 2002.
42. Mahoney, Diana Phillips, "Painting with Feeling", *Computer Graphics World*, Vol. 24, Issue 8, August 2001.
43. Mark, William R., Randolph, Scott C., Finch, Mark, Van Verth, James M., Taylor II, Russell M., "Adding Force Feedback to Graphics Systems: Issues and Solutions", *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pp447-452, 1996.
44. Miller, Timothy, and Zeleznik, Robert, "An Insidious Haptic Invasion: Adding Force Feedback to the X Desktop", *Proceedings of the 11th Annual ACM Symposium on User Interface Software and Technology*, pp59-64, 1998.
45. Munch, Stefan, and Dillmann, Rudiger, "Haptic Output in Multimodal User Interfaces", *Proceedings of IUI'97*, ACM Press, 1997.
46. Nahvi, Ali, and Hollerbach, John M., "Haptic Interfaces", <http://www.cs.utah.edu/classes/cs6360-jmg/Nahvi/haptic.html#intro>, University of Utah, 2000 .
47. Nesbitt, Keith V., Gallimore, Randall J., and Orentstein, Bernard J., "Useing Force Feedback for Multi-sensory Display", *Proceedings of the Second Australasian user Interface Conference (AUIC '01)*, IEEE Computer Society, 2001.
48. Nyarko, Kofi, Capers, Tanya, Scott, Craig, and Ladeji-Osias, Kemi, "Network Intrusion Visualization with NIVA, an Intrusion Detection Visual Analyzer with Haptic Integration", *Proceedings of the 10th Symposium on Haptic Interfaces For Virtual Environment & Teleoperator Systems*, IEEE Computer Society, 2002.
49. Okamura, Allison M., "Literature Survey of Haptic Rendering, Collision Detection, and Object Modeling", *Dextrous Manipulation Lab Technical Report*, August 27, 1998.
50. Okamura, Allison M., Richard, Christopher, Cutkosky, Mark R., "An Educational Brief – Feeling is Believing: Using a Force-Feedback Joystick to Teach Dynamic Systems", *Journal of Engineering Education*, pp345-349, July 2002.
51. Pape, Dave, "The CAVE Virtual Reality System", <http://www.evl.uic.edu/pape/CAVE>, 2001.
52. Purdue (Author unknown), "Computer Project #1: Nonlinear Springs", <http://www.math.purdue.edu/courses/MA266/project1.pdf>, Purdue University, 2003.

53. Richard, Christopher, Okamura, Allison M., Cutkosky, Mark R., "Feeling is Believing: Using a Force-Feedback Joystick to Teach Dynamic Systems", *Proceedings of the 2000 ASEE Annual Conference and Exposition, Session 3668*, 2000.
54. Sanders, Mark S. and McCormick, Ernest J., *Human Factors in Engineering and Design*, McGraw-Hill, Inc., New York, 1993.
55. Scali, Silvia, Wright, Mark, Shillito, Ann Marie, "3D Modelling Is Not for WIMPs", http://www.edvec.ed.ac.uk/library/papers/HCI2003/HCI2003_camready2.pdf, HCI International, Crete 2003.
56. Schwartz, Michael, "The Interface Revolutionary", *Computerworld*, February 5, 2001.
57. Setlur, P., Dawson, D., Chen, J., Wagner, J., "A Nonlinear Tracking Controller for a Haptic Interface Steer-by-Wire Systems", *Proceedings of the IEEE Conference on Decision and Control*, Las Vegas, NV, December 2002.
58. Sharke, Paul, "Making sense", *Mechanical Engineering*, Vol. 123, Issue 1, pp44-46, New York, Jan 2001.
59. Sklar, Aaron E., and Sarter, Nadine B., "Good Vibrations: Tactile Feedback in Support of Attention Allocation and Human-Automation Coordination in Event-Driven Domains", *Human Factors*, Vol. 41, No. 4, pp543-552, 1999.
60. Snibbe, Scott S., MacLean, Karon E., Shaw, Rob, Roderick, Jayne, Verplank, William L., Scheeff, Mark, "Haptic Techniques for Media Control", *Proceedings of the 14th Annual ACM Symposium on User Interface Software and Technology (UIST 2001)*, 2001.
61. Srinivasan, Mandayam A., and Basdogan, Cagatay, "Haptics in Virtual Environments: Taxonomy, Research Status, and Challenges", *Computers & Graphics*, Vol. 21, No. 4, pp393-404, 1997.
62. Stone, Robert J., "Haptic Feedback: A Potted History, From Telepresence to Virtual Reality", <http://www.dcs.gla.ac.uk/~stephen/workshops/haptic/papers/stone.pdf>, Aug/Sep 2000.
63. Thilmany, Jean, "Envisioning the outcome", *Mechanical Engineering*, Vol. 122, Issue 6, pp58-61, New York, June 2000.
64. Unisa (Author unknown), "A 'soft' nonlinear spring", http://www.unisa-net.unisa.edu.au/10374/soft_spring.html, University of South Australia, 2004.
65. Van Dam, Andries, "Post-WIMP User Interfaces", *Communications of the ACM*, Vol. 40, Number 2, February 1997.
66. Verplank, Bill, Gurevich, Michael, and Mathews, Max, "THE PLANK: Designing a simple haptic controller", *Proceedings of the Conference on New Instruments for Musical Expression (NIME-02)*, May 2002.

67. Weiser, Mark, and Brown, John Seely, "The Coming Age of Calm Technology", <http://www.ubiq.com/hypertext/weiser/acmfuture2endnote.htm>, October 1996.
68. Wickens, Christopher D., *Engineering Psychology and Human Performance - Second Edition*, HarperCollins Publishers Inc., New York, 1992.
69. Wickens, Christopher D., Goh, Juliana, Helleberg, John, Horrey, William J., Talleur, Donald A., "Attentional Models of Multitask Pilot Performance Using Advanced Display Technology", *Human Factors*, Vol. 45, Number 3, pp360-380, Fall 2003.
70. Wies, Evan F., Gardner, John A., O'Modhrain, M. Sile, Hasser, Christopher J., and Bulatov, Vladimir L., "Web-Based Touch Display for Accessible Science Education", *Workshop on Haptic Human-Computer Interaction*, pp108-112, September 2000.
71. Williams II, Robert L., Chen, Meng-Yun, Seaton, Jeffrey M., "Haptics-Augmented High School Physics Tutorials", *International Journal of Virtual Reality*, October 2000.
72. Zilles, C.B., and Salisbury, J.K., "A Constraint-based God-object Method For Haptic Display", *Proceedings of the International Conference on Intelligent Robots and Systems*, Vol. 3, p3146, 1995.

APPENDIX A: Pre-assessment Survey for Final Sample

Pre-assessment Survey
for
Basic Control Algorithms Simulation Study

Subject number: _____

Subject name: _____

RIN Number: _____

Date: _____

Start time: _____

End time: _____

General Information and Background Knowledge:

Contact information:

Name: _____

E-mail: _____

Phone: _____

General information:

Class year: _____

Major: _____

Age: _____ Sex: _____

- 1) On a scale of 0 to 5, where 0 = *no background* and 5 = *extensive use*, rate your knowledge and use of computer simulations: _____
- 2) On a scale of 0 to 5, where 0 = *no background* and 5 = *extensive use*, rate your knowledge and use of computer and/or video games: _____
- 3) On a scale of 1 to 5, where 0 = *no background* and 5 = *very proficient*, how would you rank your knowledge of basic control algorithms? _____
- 4) Mark the statement which best describes your learning experience in controls. If you need to clarify your answer, use the lines below.
 _____ I have no background knowledge about basic controls and algorithms.
 _____ I have had no classes which taught about basic control algorithms.
 _____ I have had no college-level classes which taught about basic control algorithms.
 _____ I have had one class which, among several other topics, taught about basic control algorithms.
 _____ I have had one class which taught mostly or exclusively about controls.
 _____ I have had more than one class which, among several other topics, taught about basic control algorithms.
 _____ I have had more than one class which taught mostly or exclusively about controls.

Clarify: _____

If you wish to add to or clarify any of the general information or background knowledge, use the lines below:

97

This portion of the survey will be used to gather information about your knowledge of basic control algorithms, that is, Proportional Control, Proportional + Integral Control (PI), and Proportional + Integral + Derivative Control (PID).

For the multiple choice, circle **all** that apply:

- 5) In a system controlled by a Proportional + Integral control algorithm, the time to reach the desired output (rise time) may be increased by...
- i) decreasing the proportional gain constant, K_P
 - ii) increasing the proportional gain constant, K_P
 - iii) decreasing the integral gain constant, K_I
 - iv) increasing the integral gain constant, K_I

Why might you want to increase the time to reach the desired output? _____

- 6) Closed-loop control...
- i) requires a feedback signal.
 - ii) requires the use of microprocessors.
 - iii) will always maintain the desired output.
 - iv) can involve human interaction in the control loop.
- 7) A car whose speed is controlled with an under-correcting control system response reaches its desired speed on level ground, and then begins to travel uphill on a ramp. The car will have the following response:
- i) The actual speed will be less than the desired speed.
 - ii) The actual speed will be 0.
 - iii) The actual speed will be half of maximum speed.
 - iv) The actual speed will be greater than the desired speed.

- 8) While riding in an elevator, you notice that it "bounces" (lightly) a couple of times as it comes to a stop at the desired floor. Knowing that the elevator is controlled with a PID controller, how might you try to eliminate the bounce?

- 9) Give a brief explanation of feedback control. _____

- 10) In a feedback control system, "error" can be defined as: _____

- 11) Describe the differences between Proportional Control, Proportional + Integral Control (PI), and Proportional + Integral + Derivative Control (PID).

- 12) Describe (a couple sentences each) 2 real-life examples of open-loop control systems and 2 examples of closed loop control systems.

- 13) Why might you want to use a PI (Proportional + Integral) controller, as opposed to a Proportional control?

- 14) Why might you use a Proportional + Integral + Derivative Control (PID) controller?

For the multiple choice, circle **all** that apply:

- 15) If a system using PID control is over-correcting, the gains should be adjusted as follows:

- i) decrease the proportional gain constant, K_P
- ii) increase the proportional gain constant, K_P
- iii) decrease the integral gain constant, K_I
- iv) increase the integral gain constant, K_I
- v) decrease the derivative gain constant, K_D
- vi) increase the derivative gain constant, K_D

16) An overdamped control system would be appropriate for which of the following systems:

- i) passenger elevator
- ii) ship steering
- iii) thermostat
- iv) CD player motor control
- v) Videotape rewinder

17) An underdamped system will:

- i) over-correct for the amount of error
- ii) under-correct for the amount of error
- iii) correct appropriately for the error in the system
- iv) approach the desired output quickly
- v) approach the desired output slowly

18) You are designing an automatic lighting control that will brighten or dim the lights in a room depending on the amount of light coming through the window. What control algorithm (Proportional, Proportional + Integral (PI), or Proportional + Integral + Derivative (PID)) would you use and why?

APPENDIX B: Post-assessment Survey for Final Sample Non-Haptic Users

Post-assessment Survey
for
Basic Control Algorithms Simulation Study

Subject number: _____

Student name: _____

RIN Number: _____

Date: _____

Start time: _____

End time: _____

This survey will be used to gather information about your knowledge and learning in basic controls, especially in respect to the software simulation that you have been using.

- 1) What are some observations or things you learned about control algorithms by using the software?

- 2) Describe (a couple sentences each) 2 examples of open-loop control systems and 2 examples of closed loop control systems.

3) What is the purpose of using PI controller as opposed to a Proportional controller?

4) What is the purpose of using a PID (Proportional+Integral+Derivative) controller as opposed to a Proportional or PI (Proportional+Integral) controller?

5) How is the error defined in a feedback control system?

6) You are designing an automatic volume control which will increase or decrease the stereo volume as the noise level in the room increases or decreases. With your initial design, you have noticed that there is a long delay between the time when the room gets louder and the time that the stereo volume increases. What might you do in terms of the control to shorten the delay?

Your stereo volume control has been purchased by a library to play background music during library hours. They would like you to modify the design of the controller so that the music volume is never greater than the desired volume. What adjustments could you make to the controller to accomplish this?

For the multiple choice, circle **all** that apply:

7) If a system using PI control is under-correcting, the gains should be adjusted as follows:

- i) decrease the proportional gain constant, K_P
- ii) increase the proportional gain constant, K_P
- iii) decrease the integral gain constant, K_I
- iv) increase the integral gain constant, K_I
- v) decrease the derivative gain constant, K_D
- vi) increase the derivative gain constant, K_D

8) An overdamped system will:

- i) over-correct for the amount of error
- ii) under-correct for the amount of error
- iii) correct appropriately for the error in the system
- iv) approach the desired output quickly
- v) approach the desired output slowly

9) What are some characteristics of a critically damped system?

- i) approaches the desired output but never quite reaches it
- ii) initially over-corrects for the amount of error
- iii) the actual output fluctuates above and below the desired output but doesn't settle on the desired output
- iv) the actual output fluctuates above and below the desired output and then settles on the desired output
- v) continuously under-corrects for the amount of error

10) A centrifuge must reach a desired velocity of rotation and maintain it exactly - which control algorithm would you use to accomplish this and why?

- 11) In the list of control systems below, mark which ones would work best with, respectively, an Overdamped, Underdamped, or Critically Damped controller:

	Overdamped	Underdamped	Critically Damped
i) Spin motor on a DVD player	_____	_____	_____
ii) Automatic volume control on stereo	_____	_____	_____
iii) Oven temperature control	_____	_____	_____
iv) Humidifier (moisture level control)	_____	_____	_____
v) Auto cruise control	_____	_____	_____
vi) Blood glucose regulator	_____	_____	_____
vii) Die press pressure control	_____	_____	_____

- 14) If the actual output from a system controlled with PID control keeps fluctuating significantly above and below the desired output value, it may be because:

- i) K_p is too low.
- ii) K_p is too high.
- iii) K_i is too low.
- iv) K_i is too high.
- v) K_d is too low.
- vi) K_d is too high.

For the multiple choice, circle **all** that apply:

- 12) A plane's automatic altitude control is controlled by an over-correcting control system. When it flies over an area with numerous and intermittent strong updrafts, the plane will:

- i) Stay fairly level.
- ii) Ascend slowly.
- iii) Ascend rapidly.
- iv) Descend slowly.
- v) Descend rapidly.
- vi) Ascend and descend in rapid succession.

- 13) Closed-loop control:

- i) *might* use the difference between the desired output and the actual output to calculate a new control signal.
- ii) *must* use the difference between the desired output and the actual output to calculate a new control signal.
- iii) will always eventually produce the desired output.
- iv) often works better if previous error values as well as present error values are used to calculate the control signal

General Reactions to the Software

On a scale of 1 to 5, where 1 = not true and 5 = very true, answer the following questions:

- 15) _____ The software was useful for learning about basic control algorithms.
- 16) _____ I feel that I now have a better understanding of the basic control algorithms and the effects of the gains.
- 17) _____ The software was enjoyable to use.
- 18) _____ I would find this software useful as a classroom tool for learning about controls.
- 19) _____ The interface was easy to use.
- 20) _____ The interface took some practice to get used to.
- 21) _____ The interface was complicated, but useful after some practice.
- 22) _____ The interface was difficult to learn and use.
- 23) _____ The worksheet was helpful as part of learning to use the software.
- 24) _____ The worksheet was helpful for learning about basic controls, in conjunction with the software.

Mark the words which best describe your experience with the control simulation. If you need to clarify your answer, use the lines below.

Software:

_____ Useful _____ Boring _____ Simplistic _____ Enjoyable _____ Confusing

_____ Okay _____ Interesting _____ Helpful _____

Clarify: _____

25) I would have liked more:

_____ Instruction in using the software

Explain:

_____ Instruction about basic controls and algorithms

Explain:

_____ Tracks to run the car on

Explain:

_____ Control over the car

Explain:

Other comments or suggestions about the simulation, the study, etc.:

APPENDIX C: Post-assessment Survey for Final Sample Haptic Users

Post-assessment Survey
for
Basic Control Algorithms Simulation Study

Subject number: _____

Student name: _____

RIN Number: _____

Date: _____

Start time: _____

End time: _____

This survey will be used to gather information about your knowledge and learning in basic controls, especially in respect to the software simulation that you have been using.

- 1) What are some observations or things you learned about control algorithms by using the software?

- 2) Describe (a couple sentences each) 2 examples of open-loop control systems and 2 examples of closed loop control systems.

3) What is the purpose of using PI controller as opposed to a Proportional controller?

4) What is the purpose of using a PID (Proportional+Integral+Derivative) controller as opposed to a Proportional or PI (Proportional+Integral) controller?

5) How is the error defined in a feedback control system?

6) You are designing an automatic volume control which will increase or decrease the stereo volume as the noise level in the room increases or decreases. With your initial design, you have noticed that there is a long delay between the time when the room gets louder and the time that the stereo volume increases. What might you do in terms of the control to shorten the delay?

Your stereo volume control has been purchased by a library to play background music during library hours. They would like you to modify the design of the controller so that the music volume is never greater than the desired volume. What adjustments could you make to the controller to accomplish this?

For the multiple choice, circle **all** that apply:

7) If a system using PI control is under-correcting, the gains should be adjusted as follows:

- i) decrease the proportional gain constant, K_P
- ii) increase the proportional gain constant, K_P
- iii) decrease the integral gain constant, K_I .
- iv) increase the integral gain constant, K_I .
- v) decrease the derivative gain constant, K_D .
- vi) increase the derivative gain constant, K_D .

8) An overdamped system will:

- i) over-correct for the amount of error
- ii) under-correct for the amount of error
- iii) correct appropriately for the error in the system
- iv) approach the desired output quickly
- v) approach the desired output slowly

9) What are some characteristics of a critically damped system?

- i) approaches the desired output but never quite reaches it
- ii) initially over-corrects for the amount of error
- iii) the actual output fluctuates above and below the desired output but doesn't settle on the desired output
- iv) the actual output fluctuates above and below the desired output and then settles on the desired output
- v) continuously under-corrects for the amount of error

10) A centrifuge must reach a desired velocity of rotation and maintain it exactly - which control algorithm would you use to accomplish this and why?

- 11) In the list of control systems below, mark which ones would work best with, respectively, an Overdamped, Underdamped, or Critically Damped controller:
- 14) If the actual output from a system controlled with PID control keeps fluctuating significantly above and below the desired output value, it may be because:

	Overdamped	Underdamped	Critically Damped
i) Spin motor on a DVD player	_____	_____	_____
ii) Automatic volume control on stereo	_____	_____	_____
iii) Oven temperature control	_____	_____	_____
iv) Humidifier (moisture level control)	_____	_____	_____
v) Auto cruise control	_____	_____	_____
vi) Blood glucose regulator	_____	_____	_____
vii) Tire pressure control	_____	_____	_____

i) K_p is too low.
 ii) K_p is too high.
 iii) K_i is too low.
 iv) K_i is too high.
 v) K_d is too low.
 vi) K_d is too high.

For the multiple choice, circle all that apply:

- 12) A plane's automatic altitude control is controlled by an over-correcting control system. When it flies over an area with numerous and intermittent strong updrafts, the plane will:
- i) Stay fairly level.
 - ii) Ascend slowly.
 - iii) Ascend rapidly.
 - iv) Descend slowly.
 - v) Descend rapidly.
 - vi) Ascend and descend in rapid succession.
- 13) Closed-loop control:
- i) *might* use the difference between the desired output and the actual output to calculate a new control signal.
 - ii) *must* use the difference between the desired output and the actual output to calculate a new control signal.
 - iii) will always eventually produce the desired output.
 - iv) often works better if previous error values as well as present error values are used to calculate the control signal

General Reactions to the Software

On a scale of 1 to 5, where 1 = not true and 5 = very true, answer the following questions:

- 15) _____ The software was useful for learning about basic control algorithms.
- 16) _____ I feel that I now have a better understanding of the basic control algorithms and the effects of the gains.
- 17) _____ The software was enjoyable to use.
- 18) _____ I would find this software useful as a classroom tool for learning about controls.
- 19) _____ The interface was easy to use.
- 20) _____ The interface took some practice to get used to.
- 21) _____ The interface was complicated, but useful after some practice.
- 22) _____ The interface was difficult to learn and use.
- 23) _____ The joystick controls were easy to use.
- 24) _____ The joystick controls were confusing.
- 25) _____ In general, I used the mouse and menus more than the joystick controls.
- 26) _____ The touch feedback was interesting to use and experience.
- 27) _____ The touch feedback made the simulation more enjoyable.
- 28) _____ The touch feedback really didn't affect the experience.
- 29) _____ The worksheet was helpful as part of learning to use the software.
- 30) _____ The worksheet was helpful for learning about basic controls, in conjunction with the software.

Mark the words which best describe your experience with the control simulation. If you need to clarify your answer, use the lines below.

Software:

____ Useful ____ Boring ____ Simplistic ____ Enjoyable ____ Confusing
____ Okay ____ Interesting ____ Helpful ____

Clarify: _____

Force feedback:

____ Useful ____ Too strong ____ Too weak ____ Enjoyable ____ Distracting
____ Neutral ____ Interesting

Clarify: _____

31) I would have liked more:

____ Instruction in using the software

Explain:

____ Instruction about basic controls and algorithms

Explain: _____

_____ Tracks to run the car on

Explain:

_____ Control over the car

Explain:

_____ Options for the force feedback

Explain:

110

Other comments or suggestions about the simulation, the study, etc.:

APPENDIX D: Worksheet for Final Sample Non-Haptic Users

Worksheet
for
Basic Control Algorithms Simulation Study

Subject number: _____

Student name: _____

RJN Number: _____

Date: _____

Start time: _____

End time: _____

Introduction:

The purpose of this simulation is to enable the user to learn about basic control algorithms: Proportional control, Proportional + Integral (PI) control, and Proportional + Integral + Derivative (PID) control. The car in the simulation is steering automatically to follow a white line on a black track using sensor data from optical sensors on the front of the car (shown in light blue). **The amount and direction that the car steers in is determined by the chosen control algorithm and the gains.**

Choosing different control algorithms, and adjusting the gains, will affect the car's steering - it may over-correct, under-correct, not steer at all, etc. The goal is for the car to follow the track as smoothly as possible. There are 3 different types of tracks to experiment on: an easy (oval) track, medium difficulty, and difficult track. Each of these tracks has a variation where the white line is slightly narrower - this may also affect the car's steering performance.

If you have any questions or comments, feel free to let me know.

Thanks for taking part in this study!

Worksheet:

1. Take a few minutes to familiarize yourself with the simulation including the different menu options.

2. Set the simulation so that the car is running on an oval track using a Proportional control algorithm for steering.

a. Write down the proportional gain (K_p): _____ and speed setting: _____

b. Describe the steering behavior of the car using the gain from a: _____

c. Would you describe the steering control as *Over-correcting*, *Under-correcting*, *Close to Ideal*, or *None of the above* (circle one)?

d. Increase the gain until you see a significant difference in the car's steering. Write down the new gain value: _____. If you changed the speed, record the new speed setting: _____.

e. Describe the steering behavior of the car using the gain from d.: _____

f. Increase the gain more or decrease the gain until you see a significant difference in the car's steering. Write down the new gain value: _____

g. Describe the steering behavior of the car using the gain from e.: _____

h. Adjust the gain until the car is steering in a way that you would consider to be close to ideal. Write down that gain value: _____

i. Why did you choose the gain setting in h? _____

j. Increase the car's speed by 10 or more and write down the new speed setting: _____ Does this change the car's steering behavior? If so, how is it different?

k. If necessary, adjust the gain so the steering is more stable. Did you increase, decrease, or not change the proportional gain (K_p)? _____

3. Change to a different track - if the car is off the track, return it to the start position using the menu option under the heading "Car". Which track are you using (remember to specify if it's one of the narrow tracks)? _____

a. Does the gain you used in Question 1.h work equally well for this track? _____

b. If the answer to a. is "No", describe how the car's steering is different. _____

c. Try to find an appropriate gain setting for this track. Write down the gain value: _____

d. Change to a different control algorithm (PI or PID) and circle the algorithm you are using. Write down the gain values being used: K_p = _____; K_i = _____; K_d = _____. (Write "N/A" for gains not being used.)

e. Describe the car's steering behavior. _____

f. Slow down the car if necessary and adjust the gain values until the car is following the track well. Write down the gain values being used: K_p = _____; K_i = _____; K_d = _____ and the speed value: _____.

g. Describe the car's steering on the more straight portions of the track: _____

h. Describe the car's steering on the curved portions of the track: _____

i. Change to a different control algorithm (P, PI or PID) and circle the algorithm you are using. Write down the gain values being used: K_p = _____; K_i = _____; K_d = _____.

j. Increase the speed to maximum (70) and adjust the gain values until the car is following the track well. Write down the gain values being used: K_p = _____; K_i = _____; K_d = _____.

<p>k. Describe the car's steering on the more straight portions of the track: _____</p> <p>_____</p> <p>_____</p> <p>l. Describe the car's steering on the curved portions of the track: _____</p> <p>_____</p> <p>_____</p> <p>4. Change to a different track - if the car is off the track, return it to the start position using the menu option under the heading "Car". Which track are you using? _____</p> <p>Adjust the speed as desired, and write down the setting: _____</p> <p>a. Circle the control algorithm you are using (P or PI or PID). Write down the gain values being used: K_p = _____; K_i = _____; K_d = _____.</p> <p>b. Describe the car's steering behavior. _____</p> <p>_____</p> <p>_____</p> <p>c. Adjust the gain values until the car is following the track well. Write down the gain values being used: K_p = _____; K_i = _____; K_d = _____.</p> <p>d. Describe the car's steering on the more straight portions of the track: _____</p> <p>_____</p> <p>_____</p> <p>e. Describe the car's steering on the curved portions of the track: _____</p> <p>_____</p> <p>_____</p>	<p>5. What are some conclusions you can draw about the effect of increasing or decreasing the gains in a feedback control system?</p> <p>_____</p> <p>_____</p> <p>6. Under what conditions (speed, type of track, etc.) did the P controller seem to work best?</p> <p>_____</p> <p>_____</p> <p>7. Under what conditions (speed, type of track, etc.) did the PI controller seem to work best?</p> <p>_____</p> <p>_____</p> <p>8. Under what conditions (speed, type of track, etc.) did the PID controller seem to work best?</p> <p>_____</p> <p>_____</p> <p>9. Other notes, comments, conclusions:</p> <p>_____</p> <p>_____</p> <p>_____</p> <p>_____</p> <p>_____</p>
---	--

APPENDIX E: Worksheet for Final Sample Haptic Users

Worksheet
for
Basic Control Algorithms Simulation Study

Subject number: _____

Student name: _____

RIN Number: _____

Date: _____

Start time: _____

End time: _____

Introduction:

The purpose of this simulation is to enable the user to learn about basic control algorithms: Proportional control, Proportional + Integral (PI) control, and Proportional + Integral + Derivative (PID) control. The car in the simulation is steering automatically to follow a white line on a black track using sensor data from optical sensors on the front of the car (shown in light blue). **The amount and direction that the car steers in is determined by the chosen control algorithm and the gains.**

Choosing different control algorithms, and adjusting the gains, will affect the car's steering - it may over-correct, under-correct, not steer at all, etc. The goal is for the car to follow the track as smoothly as possible. There are 3 different types of tracks to experiment on: an easy (oval) track, medium difficulty, and difficult track. Each of these tracks has a variation where the white line is slightly narrower - this may also affect the car's steering performance.

Most of the settings can be changed using the joystick controls (refer to the help graphic), or the window menus. Note that changing to a different track can only be done using the menu. The joystick will also give feedback reflecting how much and in what direction the car is steering. Note that forcing the joystick to move will have no effect.

If you have any questions or comments, feel free to let me know.

Thanks for taking part in this study!

Worksheet:

1. Take a few minutes to familiarize yourself with the simulation including the joystick controls and menu options.

2. Set the simulation so that the car is running on an oval track using a Proportional control algorithm for steering.

a. Write down the proportional gain (K_p): _____ and speed setting: _____

b. Describe the steering behavior of the car using the gain from a.: _____

c. Would you describe the steering control as *Over-correcting*, *Under-correcting*, *Close to Ideal*, or *None of the above* (circle one)?

d. Increase the gain until you see a significant difference in the car's steering. Write down the new gain value: _____. If you changed the speed, record the new speed setting: _____.

e. Describe the steering behavior of the car using the gain from c.: _____

f. Increase the gain more or decrease the gain until you see a significant difference in the car's steering. Write down the new gain value: _____

g. Describe the steering behavior of the car using the gain from e.: _____

h. Adjust the gain until the car is steering in a way that you would consider to be close to ideal. Write down that gain value: _____

i. Why did you choose the gain setting in h? _____

j. Increase the car's speed by 10 or more and write down the new speed setting: _____. Does this change the car's steering behavior? If so, how is it different?

k. If necessary, adjust the gain so the steering is more stable. Did you increase, decrease, or not change the proportional gain (K_p)? _____

3. Change to a different track - if the car is off the track, return it to the start position using the menu option under the heading "Car" or button 5 on the joystick base. Which track are you using (remember to specify if it's one of the narrow tracks)? _____

a. Does the gain you used in Question 1.h work equally well for this track? _____

b. If the answer to a. is "No", describe how the car's steering is different. _____

c. Try to find an appropriate gain setting for this track. Write down the gain value: _____

d. Change to a different control algorithm (PI or PID) and circle the algorithm you are using. Write down the gain values being used: K_p = _____; K_i = _____; K_d = _____. (Write "N/A" for gains not being used.)

e. Describe the car's steering behavior. _____

f. Slow down the car if necessary and adjust the gain values until the car is following the track well. Write down the gain values being used: K_p = _____; K_i = _____; K_d = _____ and the speed value: _____.

g. Describe the car's steering on the more straight portions of the track: _____

h. Describe the car's steering on the curved portions of the track: _____

i. Change to a different control algorithm (P, PI or PID) and circle the algorithm you are using. Write down the gain values being used: $K_p =$ _____; $K_i =$ _____; $K_d =$ _____.

j. Increase the speed to maximum (70) and adjust the gain values until the car is following the track well. Write down the gain values being used: $K_p =$ _____; $K_i =$ _____; $K_d =$ _____.

k. Describe the car's steering on the more straight portions of the track: _____

l. Describe the car's steering on the curved portions of the track: _____

4. Change to a different track - if the car is off the track, return it to the start position using the menu option under the heading "Car" or button 5 on the joystick base. Which track are you using? _____ Adjust the speed as desired, and write down the setting: _____

a. Circle the control algorithm you are using (P or PI or PID). Write down the gain values being used: $K_p =$ _____; $K_i =$ _____; $K_d =$ _____.

b. Describe the car's steering behavior. _____

c. Adjust the gain values until the car is following the track well. Write down the gain values being used: $K_p =$ _____; $K_i =$ _____; $K_d =$ _____.

d. Describe the car's steering on the more straight portions of the track: _____

e. Describe the car's steering on the curved portions of the track: _____

5. What are some conclusions you can draw about the effect of increasing or decreasing the gains in a feedback control system?

6. Under what conditions (speed, type of track, etc.) did the P controller seem to work best?

7. Under what conditions (speed, type of track, etc.) did the PI controller seem to work best?

8. Under what conditions (speed, type of track, etc.) did the PID controller seem to work best?

9. Other notes, comments, conclusions:

--	--	--	--	--

APPENDIX F: Simulation System Code

```

// OneView.CPP
// Created 03/23/04 - Based on TwoViews.CPP 03/22/04 version (or closest date)
//
// Creates a window including a single (overhead) view of the track, with car
// travelling on the track. Includes force feedback, and menu options for choosing
// different control algorithms and adjusting gains.
//
// Modified 03/23/04 - Code clean-up, including removing ref. to "Redcar3", etc.
// Modified 03/24/04 - Modified force feedback for greater sensitivity - using
// constant force w/ max gain and magnitude dependant on size
// of rotate_angle
// Modified 03/25/04 - Added menu options to choose between 3 different tracks,
// created new track files
// Modified 03/26/04 - Added text display to show control algorithm formula including
// gain
// - Changed PID algorithm code slightly
// Modified 04/12/04 - Added car speed control
// Modified 04/20/04 - Mapped menu controls to joystick buttons/controls
// Modified 05/17/04 - Added multi-threading functionality to perform certain
// graphics/drawing operations

// INCLUDES //////////////////////////////////////

#define WIN32_LEAN_AND_MEAN // just say no to MFC

#define INITGUID

#include <windows.h> // include important windows stuff
#include <windowsx.h>
#include <mmsystem.h>
#include <iostream.h> // include important C/C++ stuff
#include <conio.h>
#include <stdlib.h>
#include <malloc.h>
#include <memory.h>
#include <string.h>
#include <stdarg.h>
#include <stdio.h>
#include <math.h>
#include <io.h>
#include <fcntl.h>

#include <ddraw.h> // DirectX includes
#include <dinput.h>
#include "T3DLIB1.H"
#include "T3DLIB2A.H"
#include "TWOVIEWS_RES.H"

// DEFINES //////////////////////////////////////

// defines for windows
#define WINDOW_CLASS_NAME "WINCLASS1"

// default screen size
#define SCREEN_WIDTH 640 // size of screen
#define SCREEN_HEIGHT 480
#define SCREEN_BPP 8 // bits per pixel
// #define WINDOW_WIDTH 700 // size of window
#define WINDOW_WIDTH 1000 // size of window
// #define WINDOW_HEIGHT 1000
#define WINDOW_HEIGHT 600

```

```

#define WINDOW1_WIDTH      640 // size of window
#define WINDOW1_HEIGHT     480
#define WINDOW2_WIDTH      640 // size of window
#define WINDOW2_HEIGHT     480
#define CONTROL_WINDOW_WIDTH 250
// #define CONTROL_WINDOW_HEIGHT 500
#define CONTROL_WINDOW_HEIGHT 700

#define WINDOW_BPP          8 // bitdepth of window (8,16,24 etc.)
// note: if windowed and not
// fullscreen then bitdepth must
// be same as system bitdepth
// also if 8-bit the a palette
// is created and attached

#define WINDOWED_APP        1 // 0 not windowed, 1 windowed
#define WINDOW1_TITLE "Overhead View"
#define WINDOW2_TITLE "First Person View"

#define BITMAP_ID            0x4D42 // universal id for a bitmap
#define MAX_COLORS_PALETTE  256

// TYPES //////////////////////////////////////

// this will hold the car
typedef struct CAR_OBJ_TYP
{
    LPDIRECTDRAWSURFACE7 frames[1]; // no animation right now
    int x_pos,y_pos; // position of car
    int rsensor_x, rsensor_y, lsensor_x, lsensor_y; // positions of Right & Left
sensors wrt car frame
    int x_velocity, y_velocity; // x-velocity
    int angle_of_rotation;
} CAR_OBJ, *CAR_OBJ_PTR;

typedef struct CAR_SENSOR_TYP
{
    int rsensor_x, rsensor_y, lsensor_x, lsensor_y; // positions of Right & Left
sensors wrt car frame
    int x_velocity, y_velocity; // x-velocity
} CAR_SENSOR, *CAR_SENSOR_PTR;

typedef struct VEL_TABLE_ENTRY_TYP
{
    int frame_num;
    float x_velocity, y_velocity;
    int angle_of_rotation;
    CAR_SENSOR car_otus;
} VEL_TABLE_ENTRY, *VEL_TABLE_ENTRY_PTR;

// PROTOTYPES //////////////////////////////////////

int Init_CF_Effect(DWORD rgdwAxes[2], LONG rglDirection[2], DICONSTANTFORCE cf);

void delay(long int delay_time);

DWORD WINAPI Sleep_Thread(LPVOID data);

// MACROS //////////////////////////////////////

```

```

// tests if a key is up or down
#define KEYDOWN(vk_code) ((GetAsyncKeyState(vk_code) & 0x8000) ? 1 : 0)
#define KEYUP(vk_code)   ((GetAsyncKeyState(vk_code) & 0x8000) ? 0 : 1)

// initializes a direct draw struct
#define DDRAW_INIT_STRUCT(ddstruct) { memset(&ddstruct,0,sizeof(ddstruct)); }
ddstruct.dwSize=sizeof(ddstruct); }

// GLOBALS //////////////////////////////////////

HWND      main_window_handle = NULL; // globally track main window
int        window_closed     = 0;    // tracks if window is closed
//HINSTANCE hinstance_app     = NULL; // globally track hinstance
HINSTANCE main_instance      = NULL; // globally track hinstance

PALETTEENTRY Win_Palette[256];

// directdraw stuff
BITMAP_FILE      bitmap1;           // holds the bitmap
BITMAP_FILE      bitmap2;           // holds the bitmap
BITMAP_FILE      bitmap3;           // holds the bitmap
BITMAP_FILE      bitmap4;
BITMAP_FILE      bitmap5;
BITMAP_FILE      bitmap6;
//BITMAP_FILE      bitmap_temp;
BITMAP_FILE      u_arrow_bitmap;
BITMAP_FILE      d_arrow_bitmap;

BITMAP_IMAGE     track;
BITMAP_IMAGE     FP_track;
BITMAP_IMAGE     Kp_up_arrow;
BITMAP_IMAGE     Kp_down_arrow;
BITMAP_IMAGE     Ki_up_arrow;
BITMAP_IMAGE     Ki_down_arrow;
BITMAP_IMAGE     Kd_up_arrow;
BITMAP_IMAGE     Kd_down_arrow;
BITMAP_IMAGE     Speed_up_arrow;
BITMAP_IMAGE     Speed_down_arrow;
//CAR_OBJ         car;
BOB              red_car;
BOB              FP_car;
BOB              Wide_track;
CAR_SENSOR       car, orig_car;
int              Car_Start_X = 239, Car_Start_Y = 384;

char buffer[120];           // general printing buffer
char buffer2[120];         // general printing buffer
char buffer3[200];
char buffer4[200];

int gwidth  = -1;
int gheight = -1;

// directinput stuff

LPDIRECTINPUTEFFECTlpdiEffectCF = NULL;
LPDIRECTINPUTEFFECTlpdiEffectSqPer = NULL;
LPDIRECTINPUTEFFECTlpdiEffectWavy = NULL;
LPDIRECTINPUTEFFECTlpdiEffectBumpRight = NULL;

```

```

LPDIRECTINPUTEFFECTlpdiEffectSpring = NULL;
LPDIRECTINPUTEFFECTlpdiEffectRamp = NULL;

//DIJOYSTATE2          joy_state; // this holds the joystick data - declared in
T3DLIB2A.h

DWORD                  g_dwNumForceFeedbackAxis = 0;

// Globals for car behavior
int FF_flag            = 1;
int Max_Desired_FF= 5000;
float Max_Wheel_Turn = 30.0;
int Max_Error          = 255;
int Max_Speed          = 0; // Note that speed is really an indicator of how long to pause
int Min_Speed          = 400; // between iterations of the Game_Main fn
int Desired_Speed= 100;
int Min_Display_Speed = 10;
int Max_Display_Speed = 70;

// control algorithm globals

int          Control_Type = 1;
int          rotate_angle, present_angle, new_angle;
int          error, prev_error, prev_prev_error;
float Kp, Ki, Kd;
float new_steer, pres_steer;

char Kp_buffer[16];          // general printing buffer
char Ki_buffer[16];          // general printing buffer
char Kd_buffer[16];          // general printing buffer
char Speed_buffer[16];       // general printing buffer
char Kp_label_buffer1[15];   // general printing buffer
char Kp_label_buffer2[15];   // general printing buffer
char Ki_label_buffer1[15];   // general printing buffer
char Ki_label_buffer2[15];   // general printing buffer
char Kd_label_buffer1[15];   // general printing buffer
char Kd_label_buffer2[15];   // general printing buffer
char Speed_label_buffer1[4]; // general printing buffer
char Speed_label_buffer2[4]; // general printing buffer

// Globals for controlling interactivity
int          Cmd_Focus = 1;
LONG         Old_Slider_Setting = 0;
int          Display_Speed = 0;

// FUNCTIONS //////////////////////////////////////

////////////////////////////////////

int Rotate_Frame_BOB(BOB_PTR bob, BITMAP_FILE_PTR bitmap, int frame,
                    int rot_angle, int cx, int cy, int mode)
{
    DDSURFACEDESC2 ddsd;
    int i,j;
    int bs, cs;
    float as, gs;
    float cos_val, sin_val;
    float half_width, half_height;
    UCHAR *source_ptr,
          *dest_ptr;

```

```

    UCHAR *new_source_ptr,
          *new_dest_ptr;

    if (!bob)
        return(0);

    if (mode == BITMAP_EXTRACT_MODE_CELL)
    {
        // re-compute cx, cy
        cx = cx * (bob->width + 1) + 1;
        cy = cy * (bob->height + 1) + 1;
    }

    cos_val = cos_look[rot_angle];
    sin_val = sin_look[rot_angle];

    half_width = bob->width >> 1;
    half_height = bob->height >> 1;

    source_ptr = bitmap->buffer + (cy * bitmap->bitmapinfoheader.biWidth) + cx;

    ddsd.dwSize = sizeof(dds);

    (bob->images[frame])->Lock(NULL,
                               &dds,
                               DDLOCK_WAIT | DDLOCK_SURFACEMEMORYPTR,
                               NULL);

    dest_ptr = (UCHAR *)dds.lpSurface;

    new_dest_ptr = dest_ptr;
    new_source_ptr = source_ptr;

    for (j=0; j<bob->height; j++)
    {
        for (i=0; i<bob->width; i++)
        {
            memcpy(new_dest_ptr, new_source_ptr, 1);

            bs = i - half_width;
            cs = j - half_height;

            as = (bs * sin_val) + (cs * cos_val) + half_width;
            gs = (bs * cos_val) - (cs * sin_val) + half_height;

            int as_int = (int) (as + 0.5);
            int gs_int = (int) (gs + 0.5);

            new_dest_ptr = dest_ptr + (j*dds.lPitch + i);
            new_source_ptr = source_ptr + (as_int*bitmap->bitmapinfoheader.biWidth +
gs_int);
        }
    }

    (bob->images[frame])->Unlock(NULL);
    bob->attr |= BOB_ATTR_LOADED;

    return(1);
}

////////////////////////////////////

```

```

int Rotate_Sensors(BOB_PTR bob, int rot_angle)
    /* Note that the rotation of the sensors is done in a clockwise direction,
       whereas the rotation of the car frame is counter-clockwise. */
{
    int bs, cs;
    float as, gs;
    float cos_val, sin_val;
    float half_width, half_height;

    if (!bob)
        return(0);

    cos_val = cos_look[rot_angle];
    sin_val = sin_look[rot_angle];

    half_width = bob->width >> 1;
    half_height = bob->height >> 1;

    // Calculate new OTU sensor positions

    if (rot_angle < 0)
        rot_angle = abs(rot_angle);
    else if (rot_angle < 360)
        rot_angle = 360 - rot_angle;

    cos_val = cos_look[rot_angle];
    sin_val = sin_look[rot_angle];

    bs = car.rsensor_x - half_width;
    cs = car.rsensor_y - half_height;

    as = (cs * cos_val) + (bs * sin_val) + half_height; // y'
    gs = (bs * cos_val) - (cs * sin_val) + half_width;  // x'

    car.rsensor_x = (int) (gs + 0.5);
    car.rsensor_y = (int) (as + 0.5);

    bs = car.lsensor_x - half_width;
    cs = car.lsensor_y - half_height;

    as = (cs * cos_val) + (bs * sin_val) + half_height;
    gs = (bs * cos_val) - (cs * sin_val) + half_width;

    car.lsensor_x = (int) (gs + 0.5);
    car.lsensor_y = (int) (as + 0.5);

    // Calculate new x and y velocities
    car.x_velocity = (int) (cos_val + 0.5);
    car.y_velocity = (int) (sin_val + 0.5);

    return (1);
}
/////////////////////////////////////////////////////////////////

void Extract_Image_Segment(BOB_PTR bob, BITMAP_FILE_PTR bitmap,
                           int rot_angle, int height, int width)
    /* This function extracts a rectangular segment from the overhead view track
       image. */

```



```

{
    int i,j;
    int bs, cs;
    float as, gs;
    float cos_val, sin_val;
    float cos_val2, sin_val2;
    float half_width, half_height;
    int far_width, near_width;

    float mid_x_fl, mid_y_fl;
    int mid_x, mid_y;
    float x_mod, y_mod;
    float x_mod2, y_mod2;
    int NL_x, NL_y, NR_x, NR_y;
    int FL_x, FL_y, FR_x, FR_y;

    int view_angle;
    int rot_ctr_x, rot_ctr_y;

    UCHAR *source_ptr, *new_source_ptr, *new_dest_ptr;
    UCHAR *row_ptr;
    // UCHAR dest_array[150][100] = {0};
    UCHAR OH_segment_array[175][350] = {0};

    float x_scale, y_scale;
    float old_x_fl, old_y_fl;
    int old_x, old_y;
    UCHAR scale_array[320][640];

    UCHAR *dest_ptr;
    DDSURFACEDESC2 ddsd;
    int frame=0;

    int x_add2, y_add2, x_add, y_add;

    far_width = width;
    near_width = width - 200;

    mid_x_fl = (float)(car.rsensor_x + car.lsensor_x)/2.0;
    mid_y_fl = (float)(car.rsensor_y + car.lsensor_y)/2.0;

    mid_x = (int)(mid_x_fl + 0.5);
    mid_y = (int)(mid_y_fl + 0.5);

    // get cos & sin
    cos_val = cos_look[new_angle];
    sin_val = sin_look[new_angle];

    half_width = width>>1;
    half_height = height>>1;

    y_mod = cos_val * (float)(half_width);
    x_mod = sin_val * (float)(half_width);

    y_mod2 = cos_val * (float)height;
    x_mod2 = sin_val * (float)height;

    NL_x = red_car.x + mid_x - (int)(x_mod + 0.5);
    NL_y = red_car.y + mid_y - (int)(y_mod + 0.5);

```

```

NR_x = red_car.x + mid_x - (int)(-1.0 * x_mod + 0.5);
NR_y = red_car.y + mid_y - (int)(-1.0 * y_mod + 0.5);

FL_x = NL_x + (int)(y_mod2 + 0.5);
FL_y = NL_y - (int)(x_mod2 + 0.5);

FR_x = NR_x + (int)(y_mod2 + 0.5);
FR_y = NR_y - (int)(x_mod2 + 0.5);

// These 2 lines are useful for checking the field of view for the FP window
// Draw_Line(NL_x, NL_y, NR_x, NR_y, 250, back_buffer, back_lpitch);
// Draw_Line(FL_x, FL_y, FR_x, FR_y, 250, back_buffer, back_lpitch);

// source_ptr = bitmap->buffer + (NL_y * bitmap->bitmapinfoheader.biWidth) + NL_x;
source_ptr = back_buffer + (NL_y * back_lpitch) + NL_x;

ddsd.dwSize = sizeof(ddsd);

(bob->images[frame])>Lock(NULL, &ddsd,
                          DDLOCK_WAIT | DDLOCK_SURFACEMEMORYPTR,
                          NULL);

dest_ptr = (UCHAR *)ddsd.lpSurface;

for (j=0; j<height; j++)
{
    float x_add2_fl = j * cos_val;
    float y_add2_fl = j * sin_val;

    if (x_add2_fl > 0)
        x_add2 = (int) (x_add2_fl + 0.5);
    else
        x_add2 = (int) (x_add2_fl - 0.5);

    if (y_add2_fl > 0)
        y_add2 = (int) (y_add2_fl + 0.5);
    else
        y_add2 = (int) (y_add2_fl - 0.5);

    row_ptr = source_ptr - (y_add2 * back_lpitch) + x_add2;

    int row_x = NL_x + x_add2;
    int row_y = NL_y - y_add2;
    {
        for (i=0; i<width; i++)
        {
            float x_add_fl = i * sin_val;
            float y_add_fl = i * cos_val;

            if (x_add_fl > 0)
                x_add = (int)(x_add_fl + 0.5);
            else
                x_add = (int)(x_add_fl - 0.5);
            if (y_add_fl > 0)
                y_add = (int)(y_add_fl + 0.5);
            else

```

```

        y_add = (int)(y_add_fl - 0.5);

        new_dest_ptr = (OH_segment_array[j] + i);
        new_source_ptr = row_ptr + (y_add * back_lpitch) + x_add;

        memcpy(new_dest_ptr, new_source_ptr, 1);
    }
}

// Cleans up OH_segment_array - replaces non-black or white values w/ blue
//    Need to find correct color for background blue
for (j=0; j<height; j++)
{
    for (i=0; i<width; i++)
    {
        if (OH_segment_array[j][i] != 0 && OH_segment_array[j][i] != 255)
        {
            OH_segment_array[j][i] = 254;
        }
    }
}

// Scale image fragment to fit into Bob dimensions

x_scale = (float)bob->width/(float)width;
y_scale = (float)bob->height/(float)height;

for (j=0; j<bob->height; j++)
{
    for (i=0; i<bob->width; i++)
    {
        old_x_fl = (float)i / x_scale;
        old_y_fl = (float)j / y_scale;

        old_x = (int)(old_x_fl + 0.5);
        old_y = (int)(old_y_fl + 0.5);

        scale_array[j][i] = OH_segment_array[old_y][old_x];
        // scale_array[j][i] = persp_array[old_y][old_x];
    }
}

// Adjust the scaled image to give the appearance of perspective

int x_val;
int y_val = 10;
int z_val;
float x_val_fl, y_val_fl;
int view_dist = 150;
int half_w = bob->width >> 1;
int half_h = bob->height >> 1;
float x2d_fl, y2d_fl;
int x2d, y2d;
UCHAR persp_array[320][640];

for (j=0; j<bob->height; j++)

```

```

        {
            for (i=0; i<bob->width; i++)
            {
                persp_array[j][i] = 248;
            }
        }

    for (j=0; j<bob->height; j++)
    {
        //      z_val = j+1;
        z_val = j+1;

        for (i=0; i<bob->width; i++)
        {
            /*      x2d_fl = (float)(i * view_dist)/(float)z_val;
            //      x2d = half_w + (int)(x2d_fl + 0.5);
            x2d = (int)(x2d_fl + 0.5);
            y2d_fl = (float)(y_val * view_dist)/(float)z_val;
            //      y2d = half_h + (int)(y2d_fl + 0.5);
            y2d = (int)(y2d_fl + 0.5);

            */

            x_val_fl = ((i - half_w) * z_val)/view_dist;
            //      x_val_fl = ((i) * z_val)/view_dist;
            x_val = (int)(x_val_fl + 0.5);

            //      y_val_fl = ((j - half_h) * z_val)/view_dist;
            y_val_fl = (j * z_val)/view_dist;
            y_val = (int)(y_val_fl + 0.5);

            // correct to center image in front of camera(?)
            x_val = x_val + half_w;
            //      y_val = y_val; //+ (half_h/4);

            /*      if ((x2d >= 0 && x2d < bob->width) && (y2d >= 0 && y2d < bob-
            >height))
            {
                persp_array[y2d][x2d] = scale_array[j][i];
            }

            */

            if ((x_val >= 0 && x_val < bob->width) && (y_val >= 0 && y_val <
            bob->height))
                persp_array[j][i] = scale_array[y_val][x_val];
        }
    }

    // Copy the adjusted image into the bob frame

    //      new_source_ptr = (scale_array[bob->height-1] + 0);
    new_source_ptr = (persp_array[bob->height-1] + 0);
    new_dest_ptr = dest_ptr;

    //      for (j=0; j<bob->height; j++)

    for (j=(bob->height-1); j>=0; j--)
    {
        //      for (i=0; i<width; i++)

        for (i=0; i<bob->width; i++)
        {
            memcpy(new_dest_ptr, new_source_ptr, 1);

```

```

        int k = (bob->height-1) - j;
//      new_source_ptr = (scale_array[j] + i);
//      new_source_ptr = (persp_array[j] + i);
//      new_dest_ptr = dest_ptr + (j*ddsd.lPitch + i);
//      new_dest_ptr = dest_ptr + (k*ddsd.lPitch + i);
    }

    (bob->images[frame])->Unlock(NULL);
    bob->attr |= BOB_ATTR_LOADED;

}
////////////////////////////////////

LRESULT CALLBACK WindowProc(HWND hwnd,
                            UINT msg,
                            WPARAM wparam,
                            LPARAM lparam)
{
    // this is the main message handler of the system
    PAINTSTRUCT ps;          // used in WM_PAINT
    HDC         hdc;         // handle to a device context
    char buffer[120];        // used to print strings

    // what is the message
    switch(msg)
    {
        case WM_CREATE:
        {
            // do initialization stuff here
            // return success
            return(0);
        } break;

        case WM_COMMAND:
        {
            switch(LOWORD(wparam))
            {
                // handle the FILE menu
                case MENU_FILE_ID_OPEN:
                {
                    // do work here
                } break;

                case MENU_FILE_ID_CLOSE:
                {
                    // do work here
                } break;

                case MENU_FILE_ID_SAVE:
                {
                    // do work here
                } break;

                case MENU_FILE_ID_EXIT:
                {
                    // do work here
                    PostQuitMessage(0);
                }
            }
        }
    }
}

```

```

        } break;

// handle the Control Algorithm menu
case MENU_CONTROL_ID_P:
{
    // do work here
    Control_Type = 1;
} break;
case MENU_CONTROL_ID_PI:
{
    // do work here
    Control_Type = 2;
} break;
case MENU_CONTROL_ID_PID:
{
    // do work here
    Control_Type = 3;
} break;
case MENU_CONTROL_ID_BANG:
{
    // do work here
    Control_Type = 4;
} break;

// handle the CAR menu
case MENU_CAR_ID_START:
{
    orig_car.rsensor_x    = 86;
    orig_car.rsensor_y    = 54;
    orig_car.lsensor_x    = 86;
    orig_car.lsensor_y    = 45;

    car.rsensor_x         = 86;
    car.rsensor_y         = 54;
    car.lsensor_x         = 86;
    car.lsensor_y         = 45;

    present_angle         = 0;
    rotate_angle          = 0;
    new_angle              = 0;

    // initialize the control algorithm parameters
    pres_steer             = 0.0;
    new_steer              = 0.0;
    prev_error             = 0;
    prev_prev_error        = 0;
    error                  = 0;

    Rotate_Frame_BOB(&red_car, &bitmap3, 0, new_angle,
0,0,BITMAP_EXTRACT_MODE_ABS);

    Set_Pos_BOB(&red_car,Car_Start_X,Car_Start_Y);

    Sleep(10);
} break;
case MENU_CAR_ID_REVERSE:
{
    new_angle = present_angle + 180;
    if (new_angle<0)
        new_angle = 360 - abs(new_angle);
    else if (new_angle > 360)

```

```

        new_angle = new_angle - 360;
        Rotate_Sensors(&red_car, new_angle);
        Rotate_Frame_BOB(&red_car, &bitmap3, 0, new_angle,
0, 0, BITMAP_EXTRACT_MODE_ABS);
    } break;
    case MENU_CAR_ID_PAUSE:
    {
        // don't use while(1)
        while(msg==MENU_CAR_ID_PAUSE);
    } break;

    // handle the TRACK menu
    case MENU_TRACK_ID_OVAL:
    {
        // use oval track
        if (!Load_Bitmap_File(&bitmap1, "OH_oval_track1.bmp"))
            return(0);

        Create_Bitmap(&track, 0, 0, WINDOW1_WIDTH, WINDOW1_HEIGHT,
WINDOW_BPP);

        track.attr |= BITMAP_ATTR_LOADED;

        Load_Image_Bitmap(&track, &bitmap1, 0, 0,
BITMAP_EXTRACT_MODE_ABS);

    } break;
    case MENU_TRACK_ID_OVAL_NARROW:
    {
        // use oval track
        if (!Load_Bitmap_File(&bitmap1, "OH_oval_track1_narrow.bmp"))
            return(0);

        Create_Bitmap(&track, 0, 0, WINDOW1_WIDTH, WINDOW1_HEIGHT,
WINDOW_BPP);

        track.attr |= BITMAP_ATTR_LOADED;

        Load_Image_Bitmap(&track, &bitmap1, 0, 0, BITMAP_EXTRACT_MODE_ABS);

    } break;
    case MENU_TRACK_ID_MEDIUM:
    {
        // use medium track
        if (!Load_Bitmap_File(&bitmap1, "OH_Track2.bmp"))
            return(0);

        Create_Bitmap(&track, 0, 0, WINDOW1_WIDTH, WINDOW1_HEIGHT,
WINDOW_BPP);

        track.attr |= BITMAP_ATTR_LOADED;

        Load_Image_Bitmap(&track, &bitmap1, 0, 0, BITMAP_EXTRACT_MODE_ABS);

    } break;
    case MENU_TRACK_ID_MEDIUM_NARROW:
    {
        // use medium track
        if (!Load_Bitmap_File(&bitmap1, "OH_Track2_narrow.bmp"))
            return(0);

```

```

                                Create_Bitmap(&track,0,0,WINDOW1_WIDTH, WINDOW1_HEIGHT,
WINDOW_BPP);
                                track.attr |= BITMAP_ATTR_LOADED;

                                Load_Image_Bitmap(&track,&bitmap1,0,0,
BITMAP_EXTRACT_MODE_ABS);

                                } break;
                                case MENU_TRACK_ID_DIFFICULT:
                                {
                                    // use difficult track
                                    if (!Load_Bitmap_File(&bitmap1,"OH_Track3.bmp"))
                                        return(0);

                                    Create_Bitmap(&track,0,0,WINDOW1_WIDTH, WINDOW1_HEIGHT,
WINDOW_BPP);
                                    track.attr |= BITMAP_ATTR_LOADED;

                                    Load_Image_Bitmap(&track,&bitmap1,0,0,
BITMAP_EXTRACT_MODE_ABS);

                                    } break;
                                case MENU_TRACK_ID_DIFFICULT_NARROW:
                                {
                                    // use difficult track
                                    if (!Load_Bitmap_File(&bitmap1,"OH_Track3_narrow.bmp"))
                                        return(0);

                                    Create_Bitmap(&track,0,0,WINDOW1_WIDTH, WINDOW1_HEIGHT,
WINDOW_BPP);
                                    track.attr |= BITMAP_ATTR_LOADED;

                                    Load_Image_Bitmap(&track,&bitmap1,0,0,
BITMAP_EXTRACT_MODE_ABS);

                                    } break;

                                // handle the HELP menu
                                case MENU_HELP_ABOUT:
                                {
                                    MessageBox(hwnd, "Basic Control Simulation",
                                        "Created by L. Lim - 2004",
                                        MB_OK | MB_ICONEXCLAMATION);

                                    } break;
                                default: break;

                                } // end switch wparam

                                } break; // end WM_COMMAND

                                case WM_LBUTTONDOWN:
                                {
                                    // get the position of the mouse
                                    int mouse_x = (int)LOWORD(lparam);
                                    int mouse_y = (int)HIWORD(lparam);

                                    // get the button state
                                    int buttons = (int)wparam;

```



```

62))          if ((mouse_x > 810 && mouse_x < 830) && (mouse_y > 42 && mouse_y <
//          {
//              Kp += (double)0.1;
//              Kp += (double)0.05;
//          }
//          else if ((mouse_x > 810 && mouse_x < 830) && (mouse_y > 67 &&
mouse_y < 87))
//          {
//              Kp -= (double)0.1;
//              Kp -= (double)0.05;
//          }
//          if ((mouse_x > 810 && mouse_x < 830) && (mouse_y > 112 && mouse_y <
132))
//          {
//              Ki += (double)0.1;
//              Ki += (double)0.05;
//          }
//          else if ((mouse_x > 810 && mouse_x < 830) && (mouse_y > 137 &&
mouse_y < 157))
//          {
//              Ki -= (double)0.1;
//              Ki -= (double)0.05;
//          }
//          if ((mouse_x > 810 && mouse_x < 830) && (mouse_y > 182 && mouse_y <
202))
//          {
//              Kd += (double)0.01;
//          }
//          else if ((mouse_x > 810 && mouse_x < 830) && (mouse_y > 207 &&
mouse_y < 227))
//          {
//              Kd -= (double)0.01;
//          }
//          // Change the speed
//          if ((mouse_x > 810 && mouse_x < 830) && (mouse_y > 262 && mouse_y <
282))
//          {
//              float speed_incr = (float)(Min_Speed-Max_Speed)/
(float)(Max_Display_Speed-Min_Display_Speed);
//              float desired_speed_fl = (float)Desired_Speed -
(5.0*speed_incr) - 0.5;
//              Desired_Speed = (int)desired_speed_fl;
//          }
//          else if ((mouse_x > 810 && mouse_x < 830) && (mouse_y > 287 &&
mouse_y < 307))
//          {
//              float speed_incr = (float)(Min_Speed-Max_Speed)/
(float)(Max_Display_Speed-Min_Display_Speed);
//              float desired_speed_fl = (float)Desired_Speed +
(5.0*speed_incr) + 0.5;
//              Desired_Speed = (int)desired_speed_fl;
//          }
//          /*
//          hdc = GetDC(hwnd);

```

```

        sprintf(buffer, "Mouse (x,y) = (%d,%d)", mouse_x, mouse_y);
        TextOut(hdc, 0, 0, buffer, strlen(buffer));

        ReleaseDC(hwnd, hdc);
    */
        } break;

    case WM_PAINT:
    {
        // simply validate the window
        hdc = BeginPaint(hwnd, &ps);

        // end painting
        EndPaint(hwnd, &ps);

        // return success
        return(0);
    } break;

    case WM_DESTROY:
    {
        // kill the application, this sends a WM_QUIT message
        PostQuitMessage(0);

        // return success
        return(0);
    } break;

    default: break;

} // end switch

// process any messages that we didn't take care of
return (DefWindowProc(hwnd, msg, wparam, lparam));

} // end WinProc

////////////////////////////////////
//Name: EnumFFDevicesCallback()
//Desc: Called once for each enumerated force feedback device. If we find
//      one, create a device interface on it so we can play with it.
//
BOOL CALLBACK EnumFFDevicesCallback(const DIDEVICEINSTANCE* pInst,
                                    VOID* pContext)
{
    LPDIRECTINPUTDEVICE8 pDevice;
    HRESULT hr;

    // Obtain an interface to the enumerated force feedback device
    hr = lpdi->CreateDevice(pInst->guidInstance, &pDevice, NULL);

    // If it failed, we can't use this device for some reason. So continue
    enumerating
    if (FAILED(hr) )
        return DIENUM_CONTINUE;

    // We successfully created an IDirectInputDevice8. Stop looking for another.
    lpdijoy = pDevice;

```

```

        return DIENUM_STOP;
    }

    //////////////////////////////////////
    //Name: EnumAxesDevicesCallback()
    //Desc: Callback function for enumerating the axes on a joystick and counting
    //       each force feedback enabled axis
    //
    BOOL CALLBACK EnumAxesCallback(const DIDEVICEOBJECTINSTANCE* pdidoi, VOID* pContext)
    {
        DWORD* pdwNumForceFeedbackAxis = (DWORD*) pContext;

        if ( (pdidoi->dwFlags & DIDOI_FFACTUATOR) != 0)
            (*pdwNumForceFeedbackAxis)++;

        return DIENUM_CONTINUE;
    }

    //////////////////////////////////////
    //Name: Init_CF_Effect()
    //Desc: Initializes a Constant Force feedback effect
    //
    int Init_CF_Effect(DWORD rgdwAxes[2], LONG rglDirection[2], DICONSTANTFORCE cf)
    {
        DIEFFECT eff;
        DIENVELOPE diEnvelope;      // envelope

        cf.lMagnitude = 0;
        // cf.lMagnitude = DI_FFNOMINALMAX;

        // set the modulation envelope
        /*
        diEnvelope.dwSize = sizeof(DIENVELOPE);
        diEnvelope.dwAttackLevel = 1;
        diEnvelope.dwAttackTime = (DWORD) (0.01 * DI_SECONDS);
        diEnvelope.dwFadeLevel = 0;
        diEnvelope.dwFadeTime = (DWORD) (1.0 * DI_SECONDS);
        */

        ZeroMemory(&eff, sizeof(eff));
        eff.dwSize = sizeof(DIEFFECT);
        eff.dwFlags = DIEFF_CARTESIAN | DIEFF_OBJECTOFFSETS;
        eff.dwDuration = INFINITE;
        eff.dwSamplePeriod = 0;
        eff.dwGain = DI_FFNOMINALMAX;
        // eff.dwGain = 1000;
        eff.dwTriggerButton = DIEB_NOTRIGGER;
        eff.dwTriggerRepeatInterval = 0;
        eff.cAxes = g_dwNumForceFeedbackAxis;
        eff.rgdwAxes = rgdwAxes;
        eff.rglDirection = rglDirection;
        // eff.lpEnvelope = &diEnvelope;
        eff.lpEnvelope = NULL;
        eff.cbTypeSpecificParams = sizeof(DICONSTANTFORCE);
        eff.lpvTypeSpecificParams = &cf;

        //eff.dwStartDelay = 0;
    }

```

```

        if (FAILED(lpdijoy->CreateEffect(GUID_ConstantForce,
                                         &eff, &lpdiEffectCF, NULL))
    )
        return(0);
    else
        return(1);
}

////////////////////////////////////
//Name: Init_Ramp_Effect()
//Desc: Initializes a Ramp Force feedback effect
//

int Init_Ramp_Effect(DWORD rgdwAxes[2], LONG rglDirection[2])
{
    DIEFFECT eff;
    DIRAMPFORCE ramp;

    DWORD dwAxes[1] = {DIJOFS_Y};
    // DWORD dwAxes[1] = {DIJOFS_X};
    LONG lDirection[1] = {1000};

    ramp.lStart = 0;
    ramp.lEnd = -10000;

    ZeroMemory(&eff, sizeof(eff));

    eff.dwSize = sizeof(DIEFFECT);
    eff.dwFlags = DIEFF_CARTESIAN | DIEFF_OBJECTOFFSETS;

    // How long? Forever
    // eff.dwDuration = INFINITE;
    eff.dwDuration = 2 * DI_SECONDS;

    // But it takes 2 seconds to actually reach full tension
    // eff.dwSamplePeriod= 1 * DI_SECONDS;
    eff.dwSamplePeriod= 0;

    // Max gain
    eff.dwGain = DI_FFNOINALMAX;
    eff.dwTriggerButton= DIEB_NOTRIGGER;
    // eff.dwTriggerButton= DIJOFS_BUTTON0;

    // It is infinite, so it will never repeat
    eff.dwTriggerRepeatInterval = 0;

    // 1 axes
    // eff.cAxes = g_dwNumForceFeedbackAxis;
    eff.cAxes = 1;
    // eff.rgdwAxes = rgdwAxes;
    eff.rgdwAxes = dwAxes;
    // eff.rglDirection= rglDirection;
    eff.rglDirection= &lDirection[0];
    // eff.lpEnvelope = &diEnvelope;
    eff.lpEnvelope = NULL;
    eff.cbTypeSpecificParams = sizeof(DIRAMPFORCE);
    eff.lpvTypeSpecificParams = &ramp;

    if (FAILED(lpdijoy->CreateEffect(GUID_RampForce,

```

```

NULL)) )
    return(0);
else
    return(1);
}

////////////////////////////////////
//Name: Init_Spring_Effect()
//Desc: Initializes a Spring feedback effect
//

int Init_Spring_Effect(DWORD rgdAxes[2], LONG rglDirection[2], DICONDITION sp)
{
    DIEFFECT eff;

    sp.lOffset = 0;
    sp.lPositiveCoefficient = 10000;
    sp.lNegativeCoefficient = 10000;
    sp.dwPositiveSaturation = 10000;
    sp.dwNegativeSaturation = 10000;
    sp.lDeadBand = 0;

    ZeroMemory(&eff, sizeof(eff));
    eff.dwSize = sizeof(DIEFFECT);
    eff.cAxes = g_dwNumForceFeedbackAxis;
    eff.rgdwAxes = rgdAxes;
    eff.dwFlags = DIEFF_CARTESIAN | DIEFF_OBJECTOFFSETS;
    eff.rglDirection = rglDirection;
    eff.lpEnvelope = NULL;
    eff.lpvTypeSpecificParams = &sp;

    eff.dwDuration = INFINITE;
    eff.dwSamplePeriod = 0;
    eff.dwGain = DI_FF_NOMINAL_MAX;
    // eff.dwGain = 1000;
    eff.dwTriggerButton = DIEB_NOTRIGGER;
    eff.dwTriggerRepeatInterval = 0;
    // eff.lpEnvelope = &diEnvelope;
    eff.cbTypeSpecificParams = sizeof(DICONDITION);

    //eff.dwStartDelay = 0;

    if (FAILED(lpdiJoy->CreateEffect(GUID_Spring, &eff, &lpdiEffectSpring, NULL)) )
        return(0);
    else
        return(1);
}

////////////////////////////////////
//Name: Init_SquarePeriodic_Effect()
//Desc: Initializes a Square Periodic Force feedback effect
//

```

```

int Init_SquarePeriodic_Effect(DWORD rgdAxes[2], LONG rglDirection[2])
{
    // force feedback setup
    //DWORD      dwAxes[2] = { DIJOFS_X, DIJOFS_Y };
    LONG        lDirection[2] = { 0, 0 };

    DIPERIODIC diPeriodic;      // type-specific parameters
    DIENVELOPE diEnvelope;      // envelope
    DIEFFECT    diEffect;        // general parameters

    // setup the periodic structure
    //diPeriodic.dwMagnitude = DI_FFNO MINALMAX;
    diPeriodic.dwMagnitude = 2000;
    diPeriodic.lOffset = 0;
    diPeriodic.dwPhase = 0;
    diPeriodic.dwPeriod = (DWORD) (1.0 * DI_SECONDS);
    //diPeriodic.dwPeriod = (DWORD) (2.0 * DI_SECONDS);

    // set the modulation envelope

    diEnvelope.dwSize = sizeof(DIENVELOPE);
    diEnvelope.dwAttackLevel = 0;
    diEnvelope.dwAttackTime = (DWORD) (0.5 * DI_SECONDS);
    //diEnvelope.dwAttackTime = (DWORD) (0.1 * DI_SECONDS);
    diEnvelope.dwFadeLevel = 0;
    diEnvelope.dwFadeTime = (DWORD) (1.0 * DI_SECONDS);
    //diEnvelope.dwFadeTime = (DWORD) (0.5 * DI_SECONDS);

    // set up the effect structure itself
    ZeroMemory(&diEffect, sizeof(diEffect));
    diEffect.dwSize = sizeof(DIEFFECT);
    diEffect.dwFlags = DIEFF_CARTESIAN | DIEFF_OBJECTOFFSETS;
    //diEffect.dwFlags = DIEFF_POLAR | DIEFF_OBJECTOFFSETS;
    diEffect.dwDuration = INFINITE; // (DWORD) (1 * DI_SECONDS);

    // set up details of effect
    diEffect.dwSamplePeriod = 0;           // = default
    diEffect.dwGain = DI_FFNO MINALMAX;    // no scaling
    //diEffect.dwTriggerButton = DIEB_NOTRIGGER; // no trigger button
    diEffect.dwTriggerButton = DIJOFS_BUTTON0; // connect effect to trigger button
    diEffect.dwTriggerRepeatInterval = 0;
    diEffect.cAxes = 2;
    diEffect.rgdwAxes = rgdAxes;
    //diEffect.rgdwAxes = dwAxes;
    //diEffect.rglDirection = &rglDirection[0];
    diEffect.rglDirection = &lDirection[0];

    diEffect.lpEnvelope = &diEnvelope;
    diEffect.cbTypeSpecificParams = sizeof(diPeriodic);
    diEffect.lpvTypeSpecificParams = &diPeriodic;

    // create the effect and get the interface to it
    if (FAILED (lpdijoy->CreateEffect(GUID_Square, // standard GUID
                                     &diEffect, // where the data is
                                     &lpdiEffectSqPer, // where to put interface pointer
                                     NULL)) ) // no aggregation
        return(0);
    else
        return(1);
}

```

```

}
/////////////////////////////////////////////////////////////////
BOOL InitWavyEffect(void)
{
    // What axes we are mapping the effect to
    DWORD dwaxes[1] = {DIJOFS_Y};
    LONG lDirection[1] = {200};

    HRESULT hr;

    DIPERIODICdiPeriodic;
    DIEFFECTdiEffect;

    // Set up the "wavy" effect
    // This effect is periodic, so we will use DIPERIODIC

    // Set the magnitude (1/2 of max = 5000)
    // diPeriodic.dwMagnitude = 1000;
    // diPeriodic.dwMagnitude = 900;

    // No offset or phase defined - we want it cycling around center
    // and we don't much care where it starts in the wave
    diPeriodic.lOffset = 0;
    diPeriodic.dwPhase = 0;

    // Effect takes 1 sec to complete
    diPeriodic.dwPeriod = (DWORD)(0.05 * DI_SECONDS);

    // DIEFFECT structure
    ZeroMemory(&diEffect, sizeof(diEffect));
    diEffect.dwSize = sizeof(DIEFFECT);

    // Coordinate types for direction; Cartesian = x,y
    diEffect.dwFlags = DIEFF_CARTESIAN | DIEFF_OBJECTOFFSETS;

    // Do this effect as long as button is pressed
    diEffect.dwDuration = INFINITE;

    // Use our default
    diEffect.dwSamplePeriod = 0;

    // Max gain
    diEffect.dwGain = DI_FFNOMINALMAX;

    // Map this effect to be active when button 1 is pressed
    // diEffect.dwTriggerButton = DIJOFS_BUTTON1;
    // diEffect.dwTriggerButton = DIEB_NOTRIGGER;

    // Since this effect is INFINITE, there will never be a repeat
    diEffect.dwTriggerRepeatInterval = 0;

    // Effect only runs on X axis
    diEffect.cAxes = 1;
    diEffect.rgdwAxes = dwaxes;
    diEffect.rglDirection = &lDirection[0];
    diEffect.lpEnvelope = NULL;
    diEffect.cbTypeSpecificParams = sizeof(diPeriodic);
    diEffect.lpvTypeSpecificParams = &diPeriodic;
}

```

```

// Create the effect and get an interface back in our pointer
hr = lpdijoy->CreateEffect(GUID_Sine, &diEffect, &lpdiEffectWavy, NULL);

if (FAILED(hr))
{
    lpdiEffectWavy = NULL;
    return FALSE;
}
else
    return TRUE;
}

////////////////////////////////////

BOOL InitBumpEffect(void)
{
    // What axes we are mapping the effect to
    DWORD dwaxes[2] = {DIJOFS_X, DIJOFS_Y};
    LONG lDirection[2] = {200, 00};

    // Constant force, full strength
    DICONSTANTFORCE diCF = {-10000 };
    HRESULT hr;

    DIEFFECTdiEffect;

    // Set up the bump right
    diEffect.dwSize = sizeof(DIEFFECT);

    // Using x/y coords
    diEffect.dwFlags = DIEFF_CARTESIAN | DIEFF_OBJECTOFFSETS;

    // Duration here is a 10th of a second
    diEffect.dwDuration = DI_SECONDS/10;

    // No period
    diEffect.dwSamplePeriod = 0;

    // Max gain
    diEffect.dwGain = DI_FFNOINALMAX;

    // Effect not mapped to a trigger, will have to be played explicitly
    // using the Start() function
    diEffect.dwTriggerButton = DIEB_NOTRIGGER;
    diEffect.dwTriggerRepeatInterval = 0;

    // both Axes
    diEffect.cAxes = 2;
    diEffect.rgdwAxes = dwaxes;
    diEffect.rglDirection = lDirection;

    // No envelope
    diEffect.lpEnvelope = NULL;
    diEffect.cbTypeSpecificParams = sizeof(DICONSTANTFORCE);
    diEffect.lpvTypeSpecificParams = &diCF;

    // Get an interface to the new effect
    hr = lpdijoy->CreateEffect(GUID_ConstantForce, &diEffect,
                                &lpdiEffectBumpRight, NULL);

```



```

        if (FAILED(hr))
        {
            lpdiEffectBumpRight = NULL;
            return FALSE;
        }

        return TRUE;
    }

    //////////////////////////////////////

void Set_CF_Effect(LONG rglDir[2], int adjustAmt)
{
    //    LONG rglDirection[2] = {0,0};
    //    DICONSTANTFORCE cf;

    DIENVELOPE diEnvelope;
    DIEFFECT eff;

    cf.lMagnitude = adjustAmt;
    //    cf.lMagnitude = abs(adjustAmt) + 1400;
    //    cf.lMagnitude = abs(adjustAmt) * 1000;
    //    cf.lMagnitude = DI_FFNOINALMAX;
    if (cf.lMagnitude > DI_FFNOINALMAX)
        cf.lMagnitude = DI_FFNOINALMAX;
    //    else if (cf.lMagnitude < (-1 * DI_FFNOINALMAX))
    //        cf.lMagnitude = (-1 * DI_FFNOINALMAX);

    // set the modulation envelope
    diEnvelope.dwSize = sizeof(DIENVELOPE);
    diEnvelope.dwAttackLevel = 2;
    diEnvelope.dwAttackTime = (DWORD) (0.1 * DI_SECONDS);
    //    diEnvelope.dwAttackTime = 0;
    diEnvelope.dwFadeLevel = 0;
    diEnvelope.dwFadeTime = (DWORD) (0.1 * DI_SECONDS);

    lpdijoy->Acquire();

    // Unnecessary
    /*    if (lpdieffect)
        lpdieffect->Start(1,0); // Start the effect (<# of
iterations>,<other flags>)
    */

    ZeroMemory(&eff, sizeof(eff));
    eff.dwSize = sizeof(DIEFFECT);
    eff.dwFlags = DIEFF_CARTESIAN | DIEFF_OBJECTOFFSETS;
    eff.cAxes = g_dwNumForceFeedbackAxis;
    eff.rglDirection = rglDir;
    //    eff.dwGain = DI_FFNOINALMAX;
    //    eff.dwGain = 1000;
    //    eff.dwDuration = (DWORD) (0.01 * DI_SECONDS);
    //    eff.dwDuration = INFINITE;
    eff.lpEnvelope = &diEnvelope;
    //    eff.lpEnvelope = NULL;
    eff.cbTypeSpecificParams = sizeof(DICONSTANTFORCE);
    eff.lpvTypeSpecificParams = &cf;

    lpdiEffectCF->SetParameters(&eff, DIEP_DIRECTION |

```

```

DIEP_TYPESPECIFICPARAMS |
                                                                    DIEP_START);

}

////////////////////////////////////
//Name: Set_Ramp_Effect()
//Desc: Changes a Ramp Force feedback effect
//

void Set_Ramp_Effect(LONG rglDir[2], int adjustAmt)
{
    DIEFFECT eff;
    DIRAMPFORCE ramp;

    DWORD dwAxes[1] = {DIJOFS_Y};
    // DWORD dwAxes[1] = {DIJOFS_X};
    LONG lDirection[1] = {1000};

    // ramp.lStart = 0;
    // ramp.lEnd = -10000;

    lpdijoy->Acquire();

    ZeroMemory(&eff, sizeof(eff));
    eff.dwSize = sizeof(DIEFFECT);
    eff.dwFlags = DIEFF_CARTESIAN | DIEFF_OBJECTOFFSETS;

    // 1 axes
    // eff.cAxes = g_dwNumForceFeedbackAxis;
    eff.cAxes = 1;
    // eff.rgdwAxes = rgdwAxes;
    eff.rgdwAxes = dwAxes;
    // eff.rglDirection= rglDir;
    eff.rglDirection= &lDirection[0];
    // eff.lpEnvelope = &diEnvelope;
    eff.lpEnvelope = NULL;
    eff.cbTypeSpecificParams = sizeof(DIRAMPFORCE);
    eff.lpvTypeSpecificParams = &ramp;

    lpdiEffectRamp->SetParameters(&eff, DIEP_DIRECTION |
DIEP_TYPESPECIFICPARAMS |
                                                                    DIEP_START);

}

////////////////////////////////////

void Set_SquarePeriodic_Effect(LONG rglDir[2], int adjustAmt)
{
    DIPERIODIC diPeriodic; // type-specific parameters
    // DIENVELOPE diEnvelope; // envelope
    // DIEFFECT diEffect; // general parameters

    DIEFFECT eff;
    LONG lDirection[2] = { 0, 0 };

```

```

        // setup the periodic structure
// diPeriodic.dwMagnitude = DI_FFNO MINALMAX;
/* diPeriodic.dwMagnitude = adjustAmt * 1000;
diPeriodic.lOffset = 0;
diPeriodic.dwPhase = 0;
diPeriodic.dwPeriod = (DWORD) (0.5 * DI_SECONDS);
*/

// acquire the joystick
lpdijoy->Acquire();
/* if (lpdijoy->Acquire() != DI_OK)
    return(0);
*/

if (lpdiEffectSqPer)
    lpdiEffectSqPer->Start(5,0); // Start the effect (<# of iterations>,<other
flags>)

/* ZeroMemory(&eff, sizeof(eff));
eff.dwSize = sizeof(DIEFFECT);
eff.dwFlags = DIEFF_CARTESIAN | DIEFF_OBJECTOFFSETS;
// eff.dwFlags = DIEFF_POLAR | DIEFF_OBJECTOFFSETS;
eff.cAxes = g_dwNumForceFeedbackAxis;
// eff.rglDirection = &lDirection[0];
eff.rglDirection = rglDir;
// eff.dwDuration = (DWORD) (1.0 * DI_SECONDS);
eff.dwDuration = (DWORD) INFINITE;
// eff.lpEnvelope = 0;
eff.cbTypeSpecificParams = sizeof(DIPERIODIC);
eff.lpvTypeSpecificParams = &diPeriodic;

lpdiEffectSqPer->SetParameters(&eff, DIEP_DIRECTION |
DIEP_TYPESPECIFICPARAMS |
DIEP_START);
*/
}

////////////////////////////////////

void Set_Wavy_Effect(LONG rglDir[2], int adjustAmt)
{
    DIPERIODIC diPeriodic; // type-specific parameters
    DIEFFECT diEffect;

    DWORD dwaxes[1] = {DIJOFS_X};
    LONG lDirection[1] = {200};
// rglDir[0] = 200;
// Set up the "wavy" effect
// This effect is periodic, so we will use DIPERIODIC

// Set the magnitude to 1/2 of max
diPeriodic.dwMagnitude = 5000;

// No offset or phase defined - we want it cycling around center
// and we don't much care where it starts in the wave
diPeriodic.lOffset = 0;
diPeriodic.dwPhase = 0;

// Effect takes 1 sec to complete
diPeriodic.dwPeriod = DI_SECONDS;

```

```

ZeroMemory(&diEffect, sizeof(diEffect));

// DIEFFECT structure
diEffect.dwSize = sizeof(DIEFFECT);

// Coordinate types for direction; Cartesian = x,y
diEffect.dwFlags = DIEFF_CARTESIAN | DIEFF_OBJECTOFFSETS;

// Do this effect as long as button is pressed
diEffect.dwDuration = INFINITE;

// Use our default
diEffect.dwSamplePeriod = 0;

// Max gain
diEffect.dwGain = DI_FFNOMINALMAX;

// Map this effect to be active when button 1 is pressed
// diEffect.dwTriggerButton = DIJOFS_BUTTON1;
diEffect.dwTriggerButton = DIEB_NOTRIGGER;

// Since this effect is INFINITE, there will never be a repeat
diEffect.dwTriggerRepeatInterval = 0;

// Effect only runs on X axis
diEffect.cAxes = 1;
diEffect.rgdwAxes = dwaxes;
diEffect.rglDirection = &rglDir[0];
diEffect.rglDirection = &lDirection[0];
// diEffect.lpEnvelope = NULL;
diEffect.cbTypeSpecificParams = sizeof(diPeriodic);
diEffect.lpvTypeSpecificParams = &diPeriodic;

/*      if (lpdiEffectWavy)
        lpdiEffectWavy->Start(1,0); // Start the effect (<# of iterations>,<other
flags>)
*/
        lpdiEffectWavy->SetParameters(&diEffect,
                                        DIEP_DIRECTION |
                                        DIEP_TYPESPECIFICPARAMS |
                                        DIEP_START);
}

////////////////////////////////////
// Function: Get JoystickInput
//          This function is used to get joystick state information from the
//          IDirectInputDevice object for the enumerated joystick.
//
// Parameters:
//          nX -- used to pass back a horizontal direction: -100 thru 100
//          nY -- used to pass back a vertical direction: -100 thru 100
//          bButton -- used to pass back whether the "button" is presed
//
// Returns:
//          Success or Failure

BOOL GetJoystickInput(int *nX, int *nY, BOOL *bButton)
{
    HRESULT          hr;

```

```

        DIJOYSTATE diJoyState;

        // If no device object, give up
        if (!lpdijoy)
            return FALSE;

        // Initializing
        *nX = 0; *nY = 0; *bButton = FALSE;

        // Poll (retrieve) information from the device
        hr = lpdijoy->Poll();

        if (hr == DIERR_INPUTLOST || hr == DIERR_NOTACQUIRED)
        {
            // If the poll failed, try to acquire the device
            lpdijoy->Acquire();

            // When we lose control of the device we may need to re-download the
            // effects that are mapped to the buttons; other effects will auto-
download
            if (lpdiEffectWavy)
                lpdiEffectWavy->Download();

            // Let's try the poll again
            hr = lpdijoy->Poll();
        }

        if (FAILED(hr))
            return FALSE;

        // Get the device state and populate the DIJOYSTATE structure
        if (FAILED(lpdijoy->GetDeviceState(sizeof(DIJOYSTATE), &diJoyState)) )
            return FALSE;

        // Map information here to the return variables. We can easily retrieve other
        // info (like rudder, hat, throttle) from the structure
        *nX = diJoyState.lX;
        *nY = diJoyState.lY;
        *bButton = diJoyState.rgbButtons[0];

        return TRUE;
    }

    //////////////////////////////////////
    DWORD WINAPI Sleep_Thread(LPVOID data)
    {
        // this thread function sleeps

        // Try some timing
        DWORD start_time = GetTickCount();

        // clear the drawing surface
        DDRAW_Fill_Surface(lpddsback, 0);

        // lock the back buffer
        DDRAW_Lock_Back_Surface();

        // draw the background image
        Draw_Bitmap(&track, back_buffer, back_lpitch, 0);
    }

```

```

Draw_Bitmap(&Kp_up_arrow, back_buffer, back_lpitch, 0);
Draw_Bitmap(&Kp_down_arrow, back_buffer, back_lpitch, 0);

if (Control_Type > 1)
{
    Draw_Bitmap(&Ki_up_arrow, back_buffer, back_lpitch, 0);
    Draw_Bitmap(&Ki_down_arrow, back_buffer, back_lpitch, 0);
}
if (Control_Type > 2)
{
    Draw_Bitmap(&Kd_up_arrow, back_buffer, back_lpitch, 0);
    Draw_Bitmap(&Kd_down_arrow, back_buffer, back_lpitch, 0);
}

Draw_Bitmap(&Speed_up_arrow, back_buffer, back_lpitch, 0);
Draw_Bitmap(&Speed_down_arrow, back_buffer, back_lpitch, 0);

// unlock the back buffer
DDraw_Unlock_Back_Surface();
/*

Rotate_Frame_BOB(&red_car, &bitmap3, 0, new_angle, 0, 0, BITMAP_EXTRACT_MODE_ABS);

Draw_BOB(&red_car, lpddsback);

//*****

// Shows where the optical sensors are located by coloring them
// (254=light blue, 251=yellow, 250=lt green)
Draw_Rectangle(red_car.x + car.rsensor_x-1, red_car.y + car.rsensor_y-1,
               red_car.x + car.rsensor_x+1, red_car.y + car.rsensor_y+1, 254,
               lpddsback);

Draw_Rectangle(red_car.x + car.lsensor_x-1, red_car.y + car.lsensor_y-1,
               red_car.x + car.lsensor_x+1, red_car.y + car.lsensor_y+1, 254,
               lpddsback);
*/
/*
    sprintf(buffer, "Control_type = %d, Kp = %.2f, error = %d, new_steer = %.3f,
pres_steer = %.3f",
            Control_Type, Kp, error, new_steer, pres_steer);

    sprintf(buffer2, "Present_angle = %d, New_angle = %d, Rotate_angle = %d, Error =
%d; FF_magnitude = %d",
            present_angle, new_angle, rotate_angle, error, FF_magnitude);
*/
/*
    int line_indent = 0;

    if (Control_Type == 1)
    {
        sprintf(buffer, "u(k) = current control signal; e(k) = current error");
        sprintf(buffer2, " ");
        sprintf(buffer3, "Proportional control: u(k) = Kp * e(k)");
        sprintf(buffer4, "u(k) = %.2f * e(k)", Kp);
        line_indent = 135;
    }
    else if (Control_Type == 2)
    {
        sprintf(buffer, "u(k+1) = new control signal; u(k) = current
control signal");

```

```

        sprintf(buffer2, "e(k) = current error;                e(k-1) =
previous error");
        sprintf(buffer3, "Proportional + Integral control: u(k+1) = ( Kp * e(k))
+ ( Ki * e(k)) + u(k) - ( Kp * e(k-1))");
        sprintf(buffer4, "u(k+1) = (%.2f * e(k)) + (%.2f * e(k)) + u(k) - (%.2f *
e(k-1))",
            Kp, Ki, Kp);
        line_indent = 200;
    }
    else if (Control_Type == 3)
    {
        sprintf(buffer, "u(k+1) = new control signal;                u(k) = current
control signal");
        sprintf(buffer2, "e(k) = current error;                e(k-1) =
previous error");
        sprintf(buffer3, "Proportional + Integral + Derivative control: u(k+1) = (
Kp * e(k)) + ( Ki * e(k)) + u(k) - ( Kp * e(k-1)) + ( Kd * (e(k) - e(k-1)))");
        sprintf(buffer4, "u(k+1) = (%.2f * e(k)) + (%.2f * e(k)) + u(k) - (%.2f *
e(k-1)) + (%.2f * (e(k) - e(k-1)))",
            Kp, Ki, Kp, Kd);
        line_indent = 280;
    }

    //sprintf(buffer, "kx = %d, e_xsq = %.3f, ff_mag_temp = %.3f, FF_magnitude = %d", kx,
e_xsq, FF_mag_temp, FF_magnitude);
    /*
    /*      sprintf(buffer2, "ERror = %d; Rotate_angle = %d, New_angle = %d, x_vel = %d, y_vel
= %d",
        error, rotate_angle, new_angle, x_velocity, y_velocity);
    /*
    /*
    Draw_Text_GDI(buffer,line_indent,WINDOW_HEIGHT-100,RGB(100,100,50),lpddsback);
    Draw_Text_GDI(buffer2,line_indent+100,WINDOW_HEIGHT-80,RGB(100,100,50),lpddsback);
    Draw_Text_GDI(buffer3,0,WINDOW_HEIGHT-60,RGB(60,102,204),lpddsback);
    Draw_Text_GDI(buffer4,line_indent,WINDOW_HEIGHT-40,RGB(60,202,104),lpddsback);
    /*

    // Used for ~30 fps timing
    //      while((GetTickCount() - start_time) < 33);

    //      Sleep(Desired_Speed);

    if (Control_Type > 1)
    {
        if (Cmd_Focus == 2)
        {
            Draw_Rectangle(697,117,803,150, 250, lpddsback);
        }

        Draw_Rectangle(700,120,800,147, 251, lpddsback);
        sprintf(Ki_buffer,"Ki = %.2f",Ki);
        Draw_Text_GDI(Ki_buffer,720,125,RGB(100,100,50),lpddsback);
        sprintf(Ki_label_buffer1, "Integral");
        sprintf(Ki_label_buffer2, "Gain");
        Draw_Text_GDI(Ki_label_buffer1,840,115,RGB(50,200,50),lpddsback);
        Draw_Text_GDI(Ki_label_buffer2,840,135,RGB(50,200,50),lpddsback);
    }

    if (Control_Type > 2)
    {

```

```

        if (Cmd_Focus == 3)
        {
            Draw_Rectangle(697,187,803,220, 250, lpddsback);
        }

        Draw_Rectangle(700,190,800,217, 251, lpddsback);
        sprintf(Kd_buffer, "Kd = %.2f", Kd);
        Draw_Text_GDI(Kd_buffer, 720, 195, RGB(100, 100, 50), lpddsback);
        sprintf(Kd_label_buffer1, "Derivative");
        sprintf(Kd_label_buffer2, "Gain");
        Draw_Text_GDI(Kd_label_buffer1, 840, 185, RGB(50, 200, 50), lpddsback);
        Draw_Text_GDI(Kd_label_buffer2, 840, 205, RGB(50, 200, 50), lpddsback);
    }

    // Show speed
    Draw_Rectangle(700,270,800,297, 251, lpddsback);
    sprintf(Speed_buffer, "Speed = %d", Display_Speed);
    Draw_Text_GDI(Speed_buffer, 710, 275, RGB(100, 100, 50), lpddsback);
    sprintf(Speed_label_buffer1, "+5");
    sprintf(Speed_label_buffer2, "-5");
    Draw_Text_GDI(Speed_label_buffer1, 840, 262, RGB(50, 200, 50), lpddsback);
    Draw_Text_GDI(Speed_label_buffer2, 840, 290, RGB(50, 200, 50), lpddsback);

    return((DWORD)data);
}

////////////////////////////////////
int Game_Main(void *parms = NULL, int num_parms = 0)
{
    // this is the main loop of the game, do all your processing
    // here

    int index; // general looping variable
    int car_dir = 1; // 1=>right, 2=> left

    LONG rgldirection[2] = {0,0}; // will define direction force comes from

    // make sure this isn't executed again
    if (window_closed)
        return(0);

    // for now test if user is hitting ESC and send WM_CLOSE
    if (KEYDOWN(VK_ESCAPE))
    {
        PostMessage(main_window_handle, WM_CLOSE, 0, 0);
        window_closed = 1;
    } // end if

    /****
    // clear the drawing surface
    DDRAW_Fill_Surface(lpddsback, 0);

    // lock the back buffer
    DDRAW_Lock_Back_Surface();

```



```

// draw the background image
Draw_Bitmap(&track, back_buffer, back_lpitch, 0);
//Draw_Bitmap(&FP_track, back_buffer, back_lpitch, 0);

//      Extract_Image_Segment(&red_car, &bitmap2, new_angle, 100, 150);

Draw_Bitmap(&Kp_up_arrow, back_buffer, back_lpitch, 0);
Draw_Bitmap(&Kp_down_arrow, back_buffer, back_lpitch, 0);

if (Control_Type > 1)
{
    Draw_Bitmap(&Ki_up_arrow, back_buffer, back_lpitch, 0);
    Draw_Bitmap(&Ki_down_arrow, back_buffer, back_lpitch, 0);
}
if (Control_Type > 2)
{
    Draw_Bitmap(&Kd_up_arrow, back_buffer, back_lpitch, 0);
    Draw_Bitmap(&Kd_down_arrow, back_buffer, back_lpitch, 0);
}

Draw_Bitmap(&Speed_up_arrow, back_buffer, back_lpitch, 0);
Draw_Bitmap(&Speed_down_arrow, back_buffer, back_lpitch, 0);

// unlock the back buffer
DDraw_Unlock_Back_Surface();
****/

    present_angle = new_angle;
    float cos_val = cos_look[present_angle];
    float sin_val = sin_look[present_angle];

    float x_vel_fl = cos_val * 2.0;
    float y_vel_fl = sin_val * 2.0;

    int x_velocity, y_velocity;

    if (cos_val > 0.0)
    {
        red_car.x += (int)(x_vel_fl + 0.5);
        x_velocity = (int)(x_vel_fl + 0.5);
    }
    else
    {
        red_car.x += (int)(x_vel_fl - 0.5);
        x_velocity = (int)(x_vel_fl - 0.5);
    }

    if (sin_val > 0.0)
    {
        red_car.y += ((int)(y_vel_fl + 0.5)) * -1;
        y_velocity = ((int)(y_vel_fl + 0.5)) * -1;
    }
    else
    {
        red_car.y += ((int)(y_vel_fl - 0.5)) * -1;
        y_velocity = ((int)(y_vel_fl + 0.5)) * -1;
    }

```

```

    }

    // test if off screen edge, and wrap around
    if (red_car.x > WINDOW1_WIDTH)
        red_car.x = 0;
    else if ((red_car.x+100) < 0)
        red_car.x = WINDOW1_WIDTH;

    // if (red_car.y > (WINDOW1_HEIGHT + red_car.height))
    if ((red_car.y+80) > WINDOW1_HEIGHT)
        red_car.y = 0;
    else if (red_car.y < 0)
        red_car.y = WINDOW1_HEIGHT - 100;

    rglDirection[0] = 0;
    rglDirection[1] = 0;

    // Set_SquarePeriodic_Effect(rglDirection, 1);

    // Check state of joystick
    if (!DInput_Read_Joystick())
    {
        // error
    }

    if (joy_state.rgbButtons[1]) // if button 2 on the joystick is pressed
    {
        while (joy_state.rgbButtons[1])
            DInput_Read_Joystick();

        Cmd_Focus++;
        if (Cmd_Focus > Control_Type)
            Cmd_Focus = 1;

        mouse_state.lX = joy_state.lX;
        mouse_state.lY = joy_state.lY;
    }

    if (joy_state.rgbButtons[2]) // if button 3 on the joystick is pressed
    {
        while (joy_state.rgbButtons[2])
            DInput_Read_Joystick();

        if (Cmd_Focus == 1)
        {
            // Kp -= (double)0.1;
            Kp -= (double)0.05;
        }
        else if (Cmd_Focus == 2)
        {
            // Ki -= (double)0.1;
            Ki -= (double)0.05;
        }
        else if (Cmd_Focus == 3)
        {
            Kd -= (double)0.01;
        }
    }
}

```

```

if (joy_state.rgbButtons[3]) // if button 4 on the joystick is pressed
{
    while (joy_state.rgbButtons[3])
        DInput_Read_Joystick();

    if (Cmd_Focus == 1)
    {
        //      Kp += (double)0.1;
        //      Kp += (double)0.05;
    }
    else if (Cmd_Focus == 2)
    {
        //      Ki += (double)0.1;
        //      Ki += (double)0.05;
    }
    else if (Cmd_Focus == 3)
    {
        Kd += (double)0.01;
    }
}

if (joy_state.rgdwPOV[0] == 0)
{
    Control_Type = 1;
}
else if (joy_state.rgdwPOV[0] == (90 * DI_DEGREES))
{
    Control_Type = 2;
}
else if (joy_state.rgdwPOV[0] == (180 * DI_DEGREES))
{
    Control_Type = 3;
}

if (joy_state.rgbButtons[4]) // if button 5 on the joystick is pressed
{
    while (joy_state.rgbButtons[4])
        DInput_Read_Joystick();

    PostMessage(main_window_handle, WM_COMMAND, MENU_CAR_ID_START, 0);
}

if (joy_state.rgbButtons[5]) // if button 6 on the joystick is pressed
{
    while (joy_state.rgbButtons[5])
        DInput_Read_Joystick();

    PostMessage(main_window_handle, WM_COMMAND, MENU_CAR_ID_REVERSE, 0);
}

// Speed control using slider on joystick
int speed_range = Min_Speed - Max_Speed;

if (Old_Slider_Setting != joy_state.rglSlider[0])
{
    float speed_scale = 65535.0/(float)speed_range;

    LONG raw_desired_speed = joy_state.rglSlider[0];
    float desired_speed_fl = (float)raw_desired_speed/(float)speed_scale;
}

```

```

// Desired_Speed = (int)(desired_speed_fl + 0.5) + 50;
Desired_Speed = (int)(desired_speed_fl + 0.5);
Old_Slider_Setting = joy_state.rglSlider[0];
}

if (Desired_Speed < Max_Speed)
    Desired_Speed = Max_Speed;
else if (Desired_Speed > Min_Speed)
    Desired_Speed = Min_Speed;

// Calculate display speed in range 10..70
int display_speed_range = Max_Display_Speed - Min_Display_Speed;
float display_speed_factor = (float)(Min_Speed - Desired_Speed)/
(float)speed_range;
float display_speed_fl = display_speed_factor * (float)display_speed_range;
Display_Speed = (int)(display_speed_fl + 0.5) + 10;

int condition = 0;

int curr_frame_num = 0;

int i,j;
int Rsensor_sum = 0, Lsensor_sum = 0;
float Rsensor_val_fl = 0.0, Lsensor_val_fl = 0.0;
int Rsensor_val = 0, Lsensor_val = 0;

for (i=-1; i<=1; i++)
{
    for (j=-1; j<=1; j++)
    {
        Rsensor_sum += (track.buffer[(640 * ( (INT) red_car.y+
(car.rsensor_y + i))) + ( (INT) red_car.x+ (car.rsensor_x + j))]) );
        Lsensor_sum += (track.buffer[(640 * ( (INT) red_car.y+
(car.lsensor_y + i))) + ( (INT) red_car.x+ (car.lsensor_x + j))]) );
    }
    Rsensor_val_fl = (float) Rsensor_sum / 9.0;
    Lsensor_val_fl = (float) Lsensor_sum / 9.0;

    Rsensor_val = (int) (Rsensor_val_fl + 0.5);
    Lsensor_val = (int) (Lsensor_val_fl + 0.5);

// error = Rsensor_val - Lsensor_val;
error = Lsensor_val - Rsensor_val;

new_steer = 0.0;

// Control Algorithm code

if (Control_Type == 1)
{
    new_steer = Kp * (float)error;
}
else if (Control_Type == 2)
{
    new_steer = (Kp*(float)error) + (Ki*(float)error) + pres_steer -
(Kp*(float)prev_error);
    pres_steer = new_steer;
    prev_error = error;
}

```

```

    }
    else if (Control_Type == 3)
    {
// Original PID algorithm code
/*      new_steer = (Kp*(float)error) + (Ki*(float)error) + pres_steer -
(Kp*(float)prev_error);
      int diff_error = error - prev_error;
      new_steer = new_steer + (Kd * (float)diff_error);
      pres_steer = new_steer;
      prev_error = error;
*/
// Modified PID algorithm code using prev_prev_error, based on Grant's notes
      int diff_error = error - prev_error;
      new_steer = (Kp*(float)diff_error) + (Ki*(float)error) + pres_steer + (Kd *
(float)(error - 2*prev_error + prev_prev_error));
      pres_steer = new_steer;
      prev_error = error;
      prev_prev_error = prev_error;
    }
    else if (Control_Type == 4)
    {
      new_steer = (float)Max_Error;
    }

float scale_factor = Max_Wheel_Turn/255.0;
float rotate_angle_fl = scale_factor * (float)new_steer;

if (rotate_angle_fl >= 0.0)
    rotate_angle = (int)(rotate_angle_fl + 0.5);
else
    rotate_angle = (int)(rotate_angle_fl - 0.5);

if (rotate_angle > Max_Wheel_Turn)
    rotate_angle = Max_Wheel_Turn;
else if (rotate_angle < (-1 * Max_Wheel_Turn))
    rotate_angle = (-1 * Max_Wheel_Turn);

new_angle = present_angle + rotate_angle;
if (new_angle < 0)
    new_angle = 360 - abs(new_angle);
else if (new_angle > 360)
    new_angle = new_angle - 360;

// Old way of figuring FF magnitude
/*      float FF_scale_fl = 0.0;
      int FF_scale = 0;

      if (rotate_angle != 0)
      {
          car = orig_car;
          Rotate_Sensors(&red_car, new_angle);
//          Rotate_Sensors(&red_car, rotate_angle);
//          Rotate_Sensors(&red_car, -10);
//          FF_scale_fl = (float)rotate_angle/6.0;
          FF_scale_fl = (float)rotate_angle;
          if (FF_scale_fl > 0.0)
              FF_scale = (int)(FF_scale_fl + 0.5);
          else
              FF_scale = (int)(FF_scale_fl - 0.5);
      }

```

```

//          if (abs(FF_scale) < 1)
//              FF_scale = 1;
//      }
*/

// New way of figuring FF magnitude
float FF_mag_fl = 0.0;
int FF_magnitude = 0;

FF_mag_fl = (float)Max_Desired_FF * ((float)rotate_angle/(float)Max_Wheel_Turn);
// FF_mag_fl = (float)10000 * ((float)rotate_angle/Max_Wheel_Turn);
if (FF_mag_fl > 0.0)
    FF_magnitude = (int)(FF_mag_fl + 0.5);
else
    FF_magnitude = (int)(FF_mag_fl - 0.5);

int neg_mag = 0; //Flag for negative FF_magnitude
if (FF_magnitude < 0)
    neg_mag = 1;

// This is an attempt to scale the smaller FF_magnitude values so they are
// easier to feel
int k_spring = 11;
float e_spring = -0.01;

FF_magnitude = abs(FF_magnitude);
LONG FF_mag_sq = FF_magnitude * FF_magnitude;
LONG kx = k_spring * FF_magnitude;
DOUBLE e_xsq = (DOUBLE)e_spring * (DOUBLE)FF_mag_sq;
DOUBLE FF_mag_temp = (DOUBLE)kx + e_xsq;

if (FF_mag_temp > 0.0)
    FF_magnitude = (LONG)(FF_mag_temp + 0.5);

if (neg_mag == 1)
    FF_magnitude = -1 * FF_magnitude;

// Old way of calculating FF_magnitude
// FF_magnitude = abs(FF_magnitude) + 1400;
// FF_magnitude = (int)(FF_magnitude/500);
/* if (FF_magnitude < 0) // Add a scaling factor so that the magnitude of the
    FF_magnitude -= 1400; //force feedback is at least abs(1400)
else if (FF_magnitude > 0)
    FF_magnitude += 1400;
// FF_magnitude += 2400;
else if (FF_magnitude == 0)
    FF_magnitude = 0;
*/

car = orig_car;
Rotate_Sensors(&red_car, new_angle);

if (FF_flag == 1)
{
    // if Rsensor is over black and Lsensor is over white, then turn car Left
    if (error > 0)
    {

```

```

        rglDirection[0] = 1;
        rglDirection[1] = 0;

        Set_CF_Effect(rglDirection, FF_magnitude); // give force feedback response
    }

    // if Rsensor is over white and Lsensor is over black, then turn car Right
    else if (error < 0)
    {
        rglDirection[0] = 1;
        rglDirection[1] = 0;
        Set_CF_Effect(rglDirection, FF_magnitude);
    }

    // if Rsensor is over white and Lsensor is over white, then continue in same
direction
    else if (error == 0)
    {
        rglDirection[0] = 0;
        rglDirection[1] = 0;
        Set_CF_Effect(rglDirection, FF_magnitude);
    }

    // if neither sensor is over black
    else
    {
        rglDirection[0] = -1;
        rglDirection[1] = 0;
    }
} // end if (FF_flag == 1)

//Sleep(Desired_Speed);

Rotate_Frame_BOB(&red_car, &bitmap3, 0,
                 new_angle, 0, 0, BITMAP_EXTRACT_MODE_ABS);

Draw_BOB(&red_car,lpddsback);

//*****

// Shows where the optical sensors are located by coloring them
// (254=light blue, 251=yellow, 250=lt green)
Draw_Rectangle(red_car.x + car.rsensor_x-1, red_car.y + car.rsensor_y-1,
               red_car.x + car.rsensor_x+1, red_car.y + car.rsensor_y+1, 254, lpddsback);

Draw_Rectangle(red_car.x + car.lsensor_x-1, red_car.y + car.lsensor_y-1,
               red_car.x + car.lsensor_x+1, red_car.y + car.lsensor_y+1, 254, lpddsback);

/*
    sprintf(buffer,"Control_type = %d, Kp = %.2f, error = %d, new_steer = %.3f,
pres_steer =%.3f", Control_Type, Kp, error, new_steer, pres_steer);

    sprintf(buffer2, "Present_angle = %d, New_angle = %d, Rotate_angle = %d, Error =
%d; FF_magnitude = %d",present_angle, new_angle, rotate_angle, error, FF_magnitude);
*/
    int line_indent = 0;

```

```

        if (Control_Type == 1)
        {
            sprintf(buffer, "u(k) = current control signal; e(k) = current error");
            sprintf(buffer2, " ");
            sprintf(buffer3, "Proportional control: u(k) = Kp * e(k)");
            sprintf(buffer4, "u(k) = %.2f * e(k)", Kp);
            line_indent = 135;
        }
        else if (Control_Type == 2)
        {
            sprintf(buffer, "u(k+1) = new control signal; u(k) = current
control signal");
            sprintf(buffer2, "e(k) = current error; e(k-1) =
previous error");
            sprintf(buffer3, "Proportional + Integral control: u(k+1) = ( Kp * e(k))
+ ( Ki * e(k)) + u(k) - ( Kp * e(k-1))");
            sprintf(buffer4, "u(k+1) = (%.2f * e(k)) + (%.2f * e(k)) + u(k) - (%.2f *
e(k-1))",
                Kp, Ki, Kp);
            line_indent = 200;
        }
        else if (Control_Type == 3)
        {
            sprintf(buffer, "u(k+1) = new control signal; u(k) = current
control signal");
            sprintf(buffer2, "e(k) = current error; e(k-1) =
previous error");
            sprintf(buffer3, "Proportional + Integral + Derivative control: u(k+1) = (
Kp * e(k)) + ( Ki * e(k)) + u(k) - ( Kp * e(k-1)) + ( Kd * (e(k) - e(k-1)))");
            sprintf(buffer4, "u(k+1) = (%.2f * e(k)) + (%.2f * e(k)) + u(k) - (%.2f *
e(k-1)) + (%.2f * (e(k) - e(k-1)))",
                Kp, Ki, Kp, Kd);
            line_indent = 280;
        }

//sprintf(buffer, "kx = %d, e_xsq = %.3f, ff_mag_temp = %.3f, FF_magnitude = %d", kx,
e_xsq, FF_mag_temp, FF_magnitude);

//      sprintf(buffer2, "Error = %d; Rotate_angle = %d, New_angle = %d, x_vel = %d, y_vel
= %d",
//          error, rotate_angle, new_angle, x_velocity, y_velocity);

Draw_Text_GDI(buffer, line_indent, WINDOW_HEIGHT-100, RGB(100,100,50), lpddsback);
Draw_Text_GDI(buffer2, line_indent+100, WINDOW_HEIGHT-80, RGB(100,100,50), lpddsback);
Draw_Text_GDI(buffer3, 0, WINDOW_HEIGHT-60, RGB(60,102,204), lpddsback);
Draw_Text_GDI(buffer4, line_indent, WINDOW_HEIGHT-40, RGB(60,202,104), lpddsback);

// Show control gains

if (Cmd_Focus == 1)
{
    Draw_Rectangle(697,47,803,80, 250, lpddsback);
}
Draw_Rectangle(700,50,800,77, 251, lpddsback);
sprintf(Kp_buffer, "Kp = %.2f", Kp);
Draw_Text_GDI(Kp_buffer, 720, 55, RGB(100,100,50), lpddsback);
sprintf(Kp_label_buffer1, "Proportional");
sprintf(Kp_label_buffer2, "Gain");
Draw_Text_GDI(Kp_label_buffer1, 840, 45, RGB(50,200,50), lpddsback);

```



```

Draw_Text_GDI(Kp_label_buffer2,840,65,RGB(50,200,50),lpddsback);

/*
if (Control_Type > 1)
{
    if (Cmd_Focus == 2)
    {
        Draw_Rectangle(697,117,803,150, 250, lpddsback);

        Draw_Rectangle(700,120,800,147, 251, lpddsback);
        sprintf(Ki_buffer, "Ki = %.2f",Ki);
        Draw_Text_GDI(Ki_buffer,720,125,RGB(100,100,50),lpddsback);
        sprintf(Ki_label_buffer1, "Integral");
        sprintf(Ki_label_buffer2, "Gain");
        Draw_Text_GDI(Ki_label_buffer1,840,115,RGB(50,200,50),lpddsback);
        Draw_Text_GDI(Ki_label_buffer2,840,135,RGB(50,200,50),lpddsback);
    }

    if (Control_Type > 2)
    {
        if (Cmd_Focus == 3)
        {
            Draw_Rectangle(697,187,803,220, 250, lpddsback);

            Draw_Rectangle(700,190,800,217, 251, lpddsback);
            sprintf(Kd_buffer, "Kd = %.2f",Kd);
            Draw_Text_GDI(Kd_buffer,720,195,RGB(100,100,50),lpddsback);
            sprintf(Kd_label_buffer1, "Derivative");
            sprintf(Kd_label_buffer2, "Gain");
            Draw_Text_GDI(Kd_label_buffer1,840,185,RGB(50,200,50),lpddsback);
            Draw_Text_GDI(Kd_label_buffer2,840,205,RGB(50,200,50),lpddsback);
        }

        // Show speed
        Draw_Rectangle(700,270,800,297, 251, lpddsback);
        sprintf(Speed_buffer, "Speed = %d",display_speed);
        Draw_Text_GDI(Speed_buffer,710,275,RGB(100,100,50),lpddsback);
        sprintf(Speed_label_buffer1, "+5");
        sprintf(Speed_label_buffer2, "-5");
        Draw_Text_GDI(Speed_label_buffer1,840,262,RGB(50,200,50),lpddsback);
        Draw_Text_GDI(Speed_label_buffer2,840,290,RGB(50,200,50),lpddsback);
    }
*/

// move objects around

// flip the surfaces
DDraw_Flip();

// Multi-threading attempt
HANDLE thread_handle;
DWORD thread_id;

// create the thread, IRL we would check for errors
thread_handle = CreateThread(NULL,
0,
Sleep_Thread,

```

```

        (LPVOID)1,
        0,
        &thread_id);

/*thread_handle = CreateThread(NULL,
        THREAD_SET_INFORMATION,
        Sleep_Thread,
        (LPVOID)1,
        0,
        &thread_id);

*/
/*
if (SetThreadPriority(thread_handle, THREAD_PRIORITY_BELOW_NORMAL));
//if (SetThreadPriority(thread_handle, THREAD_PRIORITY_LOWEST));
else
{
    // error
}
*/

// wait a sec
// Sleep(300);
Sleep(Desired_Speed);

CloseHandle(thread_handle);

//lpdiEffectCF->Stop();

// return success or failure or your own return code here
return(1);

} // end Game_Main

////////////////////////////////////

int Game_Init(void *parms = NULL, int num_parms = 0)
{
    // this is called once after the initial window is created and
    // before the main event loop is entered, do all your initialization
    // here

    //      DIPROPDWORD dipdw;

    // Load palette
    // Load_Palette_From_File("Win256_4.pal",&Win_Palette);

    /*      Save_Palette(Win_Palette);

    if (FAILED(lpdd->CreatePalette(DDPCAPS_8BIT|
                                DDPCAPS_ALLOW256|
                                DDPCAPS_INITIALIZE,
                                Win_Palette,
                                &lpddpal,
                                NULL)))
    {
        //error
    }
    */

```

```

        lpddsprimary->SetPalette(lpddpal);
*/
        // initialize DirectDraw
// DDraw_Init(SCREEN_WIDTH, SCREEN_HEIGHT, SCREEN_BPP);
DDraw_Init(WINDOW_WIDTH, WINDOW_HEIGHT, WINDOW_BPP, WINDOWED_APP);

        // start up DirectInput
        if (!DInput_Init())
            return(0);

        // initialize the joystick
// if (!DInput_Init_FFJoystick(-24,24,-24,24))
if (!DInput_Init_FFJoystick(-1024,1024,-1024,1024,0))
    return(0);

//////////

if (FAILED(lpdijoy->EnumObjects(EnumAxesCallback,
(VOID*)&g_dwNumForceFeedbackAxis, DIDFT_AXIS)) )
    return(0);

// Right now we'll only support one or 2-axis joysticks
if (g_dwNumForceFeedbackAxis > 2)
    g_dwNumForceFeedbackAxis = 2;

// Only one effect right now - raw forces

DWORD          rgdwAxes[2] = {DIJOFS_X, DIJOFS_Y};
LONG           rglDirection[2] = {0,0};
DICONSTANTFORCE cf = {0};
DICONDITION sp = {0};

if (Init_CF_Effect(rgdwAxes, rglDirection, cf) == 0)
    return(0);

/*if (Init_Spring_Effect(rgdwAxes, rglDirection, sp) == 0)
    return(0);
*/
if (Init_Ramp_Effect(rgdwAxes, rglDirection) == 0)
    return(0);

/*if (Init_SquarePeriodic_Effect(rgdwAxes, rglDirection) == 0)
    return(0);
*/
if (InitWavyEffect() == 0)
    return (0);

// Check state of joystick
/*
        if (!DInput_Read_Joystick())
        {
            // error
        }

        rglDirection[0] = -1;
        rglDirection[1] = 0;

```

```

while (!(joy_state.rgbButtons[0]))
    DInput_Read_Joystick(); // wait until trigger is pressed
if (joy_state.rgbButtons[0]) // if button 1(trigger) on the joystick is pressed
{
    while (joy_state.rgbButtons[0])
    {
        DInput_Read_Joystick();
        Set_CF_Effect(rglDirection, Max_Desired_FF); // give force feedback
response
    }
}
while (!(joy_state.rgbButtons[0]))
    DInput_Read_Joystick(); // wait until trigger is pressed

lpdiEffectCF->Stop();

while (!(joy_state.rgbButtons[0]))
    DInput_Read_Joystick(); // wait until trigger is pressed

rglDirection[0] = 1;
rglDirection[1] = 0;

if (joy_state.rgbButtons[0]) // if button 1(trigger) on the joystick is pressed
{
    while (joy_state.rgbButtons[0])
    {
        DInput_Read_Joystick();
        Set_CF_Effect(rglDirection, Max_Desired_FF); // give force feedback
response
    }
}
while (!(joy_state.rgbButtons[0]))
    DInput_Read_Joystick(); // wait until trigger is pressed
*/

//lpdiEffectSqPer->Start(1,0); // Start the effect (<# of iterations>,<other flags>)
//lpdiEffectCF->Start(1,0);
//lpdiEffectRamp->Start(1,0);
//lpdiEffectWavy->Start(1,0);
//lpdiEffectSpring->Start(1,0);

Build_Sin_Cos_Tables();

////////////////////////////////////

// load the 8-bit track image
//if (!Load_Bitmap_File(&bitmap1,"Oval_track_seg2.bmp"))
//if (!Load_Bitmap_File(&bitmap1,"Oval_track_seg_narrow.bmp"))
if (!Load_Bitmap_File(&bitmap1,"OH_oval_track1.bmp"))
    return(0);

Create_Bitmap(&track,0,0,WINDOW1_WIDTH, WINDOW1_HEIGHT, WINDOW_BPP);
track.attr |= BITMAP_ATTR_LOADED;

Load_Image_Bitmap(&track,&bitmap1,0,0,BITMAP_EXTRACT_MODE_ABS);

// we don't unload the bitmap file, so it can be used to produce a first person view
//Unload_Bitmap_File(&bitmap1);

```

```

// load the 8-bit track (First Person view) image
/*
if (!Load_Bitmap_File(&bitmap2,"Track1.bmp"))
    return(0);

Create_Bitmap(&FP_track,0,485,WINDOW2_WIDTH, WINDOW2_HEIGHT, WINDOW_BPP);
FP_track.attr |= BITMAP_ATTR_LOADED;

Load_Image_Bitmap(&FP_track,&bitmap2,0,0,BITMAP_EXTRACT_MODE_ABS);

// unload the bitmap file, we no longer need it
Unload_Bitmap_File(&bitmap2);
*/

RECT window1_rect = {0,0,WINDOW1_WIDTH,WINDOW1_HEIGHT};
//RECT window2_rect = {0,485,WINDOW2_WIDTH,(485+WINDOW2_HEIGHT)};
RECT window2_rect = {0,485,WINDOW_WIDTH,WINDOW_HEIGHT};
RECT window3_rect = {690,0,(700+CONTROL_WINDOW_WIDTH),CONTROL_WINDOW_HEIGHT};

RECT win_list[3];
win_list[0] = window1_rect;
win_list[1] = window2_rect;
win_list[2] = window3_rect;

//lpddclipper = DDraw_Attach_Clipper(lpddsback,1,&window1_rect);
lpddclipper = DDraw_Attach_Clipper(lpddsback,3,win_list);

// load the car image

/*if (!Load_Bitmap_File(&bitmap3,"RedCar_seq3.bmp"))
    return(0);
*/if (!Load_Bitmap_File(&bitmap3,"RedCar2.bmp"))
    return(0);

//    if (!Load_Bitmap_File(&bitmap_temp,"RedCar3.bmp"))
//        return(0);

Create_BOB(&red_car,10,190,100,100,1,
            BOB_ATTR_VISIBLE | BOB_ATTR_SINGLE_FRAME,
            DDSCAPS_SYSTEMMEMORY, 255);

/*Create_BOB(&red_car,10,190,100,100,36,
            BOB_ATTR_VISIBLE | BOB_ATTR_MULTI_FRAME,
            DDSCAPS_SYSTEMMEMORY, 255);
*/

Load_Frame_BOB(&red_car,&bitmap3,0,0,0,BITMAP_EXTRACT_MODE_ABS);

// unload the bitmap file
//Unload_Bitmap_File(&bitmap3);

red_car.curr_frame = 0;
//Set_Pos_BOB(&red_car,170,330); // for oval track
Set_Pos_BOB(&red_car,Car_Start_X,Car_Start_Y);

// load the first person car image

```

```

/*
if (!Load_Bitmap_File(&bitmap4,"Car_Nose1.bmp"))
    return(0);

// now create the car bob - note that the color key(transparency) = 255 (white)
if (!(Create_BOB(&FP_car,250,880,170,120,1,
                BOB_ATTR_VISIBLE | BOB_ATTR_SINGLE_FRAME,
                DDSCAPS_SYSTEMMEMORY, 255)) )
    return(0);

if (!(Load_Frame_BOB(&FP_car,&bitmap4,0,0,0,BITMAP_EXTRACT_MODE_ABS)) )
    return(0);

// unload the bitmap file
Unload_Bitmap_File(&bitmap4);

Set_Pos_BOB(&FP_car,250,880);
*/

// load the track image as a BOB
/*
if (!Load_Bitmap_File(&bitmap5,"FP_straight.bmp"))
    return(0);

if (!(Create_BOB(&Wide_track,0,650,640,320,5,
                BOB_ATTR_VISIBLE | BOB_ATTR_MULTI_ANIM,
                DDSCAPS_SYSTEMMEMORY, 215)) )
    return(0);

if (!(Load_Frame_BOB(&Wide_track,&bitmap5,0,0,0,BITMAP_EXTRACT_MODE_ABS)) )
    return(0);

// unload the bitmap file
Unload_Bitmap_File(&bitmap5);

//Unload_Bitmap_File(&bitmap6);

Set_Pos_BOB(&Wide_track,0,650);
*/

// seed random number generator
srand(GetTickCount());

// initialize the car sensors
/*
car.rsensor_x      = 82;
car.rsensor_y      = 62;
car.lsensor_x      = 82;
car.lsensor_y      = 30;
*/
/*car.rsensor_x      = 85;
car.rsensor_y      = 48;
car.lsensor_x      = 85;
car.lsensor_y      = 42;
*/
orig_car.rsensor_x  = 86;
orig_car.rsensor_y  = 54;
orig_car.lsensor_x  = 86;

```

```

orig_car.lsensor_y      = 45;

car.rsensor_x           = 86;
car.rsensor_y           = 54;
car.lsensor_x           = 86;
car.lsensor_y           = 45;

present_angle           = 0;
rotate_angle            = 0;
new_angle               = 0;

// initialize the control algorithm parameters
pres_steer              = 0.0;
new_steer               = 0.0;
prev_prev_error         = 0;
prev_error              = 0;
error                   = 0;
Kp                      = 0.1;
Ki                      = 0.1;
Kd                      = 0.01;

// load the 8-bit arrow images
if (!Load_Bitmap_File(&u_arrow_bitmap, "up_arrow.bmp"))
    return(0);

Create_Bitmap(&Kp_up_arrow, 810, 42, 20, 20, WINDOW_BPP);
Kp_up_arrow.attr |= BITMAP_ATTR_LOADED;

Load_Image_Bitmap(&Kp_up_arrow, &u_arrow_bitmap, 0, 0, BITMAP_EXTRACT_MODE_ABS);

Create_Bitmap(&Ki_up_arrow, 810, 112, 20, 20, WINDOW_BPP);
Ki_up_arrow.attr |= BITMAP_ATTR_LOADED;

Load_Image_Bitmap(&Ki_up_arrow, &u_arrow_bitmap, 0, 0, BITMAP_EXTRACT_MODE_ABS);

Create_Bitmap(&Kd_up_arrow, 810, 182, 20, 20, WINDOW_BPP);
Kd_up_arrow.attr |= BITMAP_ATTR_LOADED;

Load_Image_Bitmap(&Kd_up_arrow, &u_arrow_bitmap, 0, 0, BITMAP_EXTRACT_MODE_ABS);

Create_Bitmap(&Speed_up_arrow, 810, 262, 20, 20, WINDOW_BPP);
Speed_up_arrow.attr |= BITMAP_ATTR_LOADED;

Load_Image_Bitmap(&Speed_up_arrow, &u_arrow_bitmap, 0, 0, BITMAP_EXTRACT_MODE_ABS);

// unload the bitmap file, we no longer need it
Unload_Bitmap_File(&u_arrow_bitmap);

if (!Load_Bitmap_File(&d_arrow_bitmap, "down_arrow.bmp"))
    return(0);

Create_Bitmap(&Kp_down_arrow, 810, 67, 20, 20, WINDOW_BPP);
Kp_down_arrow.attr |= BITMAP_ATTR_LOADED;

Load_Image_Bitmap(&Kp_down_arrow, &d_arrow_bitmap, 0, 0, BITMAP_EXTRACT_MODE_ABS);

```

```

Create_Bitmap(&Ki_down_arrow,810,137,20,20, WINDOW_BPP);
Ki_down_arrow.attr |= BITMAP_ATTR_LOADED;

Load_Image_Bitmap(&Ki_down_arrow,&d_arrow_bitmap,0,0,BITMAP_EXTRACT_MODE_ABS);

Create_Bitmap(&Kd_down_arrow,810,207,20,20, WINDOW_BPP);
Kd_down_arrow.attr |= BITMAP_ATTR_LOADED;

Load_Image_Bitmap(&Kd_down_arrow,&d_arrow_bitmap,0,0,BITMAP_EXTRACT_MODE_ABS);

Create_Bitmap(&Speed_down_arrow,810,287,20,20, WINDOW_BPP);
Speed_down_arrow.attr |= BITMAP_ATTR_LOADED;

Load_Image_Bitmap(&Speed_down_arrow,&d_arrow_bitmap,0,0,BITMAP_EXTRACT_MODE_ABS);

// unload the bitmap file, we no longer need it
Unload_Bitmap_File(&d_arrow_bitmap);

// return success or failure or your own return code here
return(1);

} // end Game_Init

////////////////////////////////////////

int Game_Shutdown(void *parms = NULL, int num_parms = 0)
{
// this is called after the game is exited and the main event
// loop while is exited, do all you cleanup and shutdown here

// kill all the surfaces

Destroy_Bitmap(&track);
//Destroy_Bitmap(&FP_track);

Destroy_BOB(&red_car);
//Destroy_BOB(&FP_car);

// Unacquire the DI device
DInput_Release_Joystick();

// Release the DI objects
lpdiEffectCF->Release();
//lpdiEffectSqPer->Release();
//lpdiEffectSpring->Release();
lpdiEffectRamp->Release();
lpdiEffectWavy->Release();
//lpdiEffectBumpRight->Release();

DInput_Shutdown();

DDraw_Shutdown();

```



```

// return success or failure or your own return code here
return(1);

} // end Game_Shutdown

////////////////////////////////////
void delay(long int delay_time)
{
    int i;

    for (i=0; i<delay_time; i++);
}

// WINMAIN //////////////////////////////////////

int WINAPI WinMain(HINSTANCE hinstance,
                  HINSTANCE hprevinstance,
                  LPSTR lpcmdline,
                  int ncmdshow)
{
    WNDCLASSEX winclass; // this will hold the class we create
    HWND        hwnd;    // generic window handle
    MSG          msg;     // generic
    HDC          hdc;     // graphics device context

    // first fill in the window class structure
    winclass.cbSize      = sizeof(WNDCLASSEX);
    winclass.style       = CS_DBLCLKS | CS_OWNDC |
                          CS_HREDRAW | CS_VREDRAW;
    winclass.lpfnWndProc= WindowProc;
    winclass.cbClsExtra= 0;
    winclass.cbWndExtra= 0;
    winclass.hInstance= hinstance;
    winclass.hIcon       = LoadIcon(NULL, IDI_APPLICATION);
    winclass.hCursor= LoadCursor(NULL, IDC_ARROW);
    winclass.hbrBackground= (HBRUSH)GetStockObject(BLACK_BRUSH);
    winclass.lpszMenuName= NULL;
    winclass.lpszClassName= WINDOW_CLASS_NAME;
    winclass.hIconSm     = LoadIcon(NULL, IDI_APPLICATION);

    // save hinstance in global
    main_instance = hinstance;

    // register the window class
    if (!RegisterClassEx(&winclass))
        return(0);

    // create the window
    if (!(hwnd = CreateWindowEx(NULL, // extended style
                              WINDOW_CLASS_NAME, // class
                              WINDOW1_TITLE, // title
                              (WINDOWED_APP ? (WS_OVERLAPPED | WS_SYSMENU | WS_CAPTION) :
                              (WS_POPUP | WS_VISIBLE)),
                              0,0, // initial x,y
                              WINDOW_WIDTH,WINDOW_HEIGHT, // initial width, height
                              NULL, // handle to parent
                              NULL, // handle to menu
                              hinstance, // instance of this application
                              NULL))) // extra creation parms

```

```

return(0);

/*
CreateWindowEx(NULL,          // extended style
               "button", // class
               "Push Me", // title
               WS_VISIBLE | WS_CHILD | BS_PUSHBUTTON,
               700,300, // initial x,y
               100,24, // initial width, height
               main_window_handle, // handle to parent
               (HMENU)(100), // handle to menu
               hinstance, // instance of this application
               NULL); // extra creation parms
*/

// save main window handle
main_window_handle = hwnd;

// load the menu resource
HMENU hmenuhandle = LoadMenu(hinstance, "MainMenu");

// attach the menu to the window
SetMenu(hwnd, hmenuhandle);

//////////
// resize the window so that client is really width x height
if (WINDOWED_APP)
{
    // now resize the window, so the client area is the actual size requested
    // since there may be borders and controls if this is going to be a windowed app
    // if the app is not windowed then it won't matter
    RECT window_rect = {0,0,WINDOW_WIDTH-1,WINDOW_HEIGHT-1};

    // make the call to adjust window_rect
    AdjustWindowRectEx(&window_rect,
        GetWindowStyle(main_window_handle),
        GetMenu(main_window_handle) != NULL,
        GetWindowExStyle(main_window_handle));

    // save the global client offsets, they are needed in DDraw_Flip()
    window_client_x0 = -window_rect.left;
    window_client_y0 = -window_rect.top;

    // now resize the window with a call to MoveWindow()
    MoveWindow(main_window_handle,
        0, // x position
        0, // y position
        window_rect.right - window_rect.left, // width
        window_rect.bottom - window_rect.top, // height
        FALSE);

    // show the window, so there's no garbage on first render
    ShowWindow(main_window_handle, SW_SHOW);
} // end if windowed

//////////

```

```

// initialize game here
Game_Init();

// enter main event loop
while(TRUE)
{
    // Try some timing
    DWORD start_time = GetTickCount();

    // test if there is a message in queue, if so get it
    if (PeekMessage(&msg,NULL,0,0,PM_REMOVE))
    {
        // test if this is a quit
        if (msg.message == WM_QUIT)
            break;

        // translate any accelerator keys
        TranslateMessage(&msg);

        // send the message to the window proc
        DispatchMessage(&msg);
    } // end if

    // main game processing goes here
    Game_Main();

    // Used for ~30 fps timing
    // while((GetTickCount() - start_time) < 33);

    } // end while

// closedown game here
Game_Shutdown();

// return to Windows like this
return(msg.wParam);

} // end WinMain

////////////////////////////////////

```

```

// TwoViews.rc
//
// Resource file - used to create window menu system

#include "TwoViews_res.h"

CURSOR_CROSSHAIR CURSOR crosshair.cur

MAINMENU MENU DISCARDABLE
BEGIN
    POPUP "File"
    BEGIN
        // MENUITEM "Open",          MENU_FILE_ID_OPEN
        // MENUITEM "Close",         MENU_FILE_ID_CLOSE
        // MENUITEM "Save",          MENU_FILE_ID_SAVE
        MENUITEM "Exit",            MENU_FILE_ID_EXIT
    END
    POPUP "Control Algorithm"
    BEGIN
        MENUITEM "Proportional",    MENU_CONTROL_ID_P
        MENUITEM "Proportional + Integral", MENU_CONTROL_ID_PI
        MENUITEM "PID",              MENU_CONTROL_ID_PID
        // MENUITEM "Bang Bang",    MENU_CONTROL_ID_BANG
    END
    POPUP "Car"
    BEGIN
        MENUITEM "Return to Start",  MENU_CAR_ID_START
        MENUITEM "Reverse direction", MENU_CAR_ID_REVERSE
        // MENUITEM "Pause",         MENU_CAR_ID_PAUSE
    END
    POPUP "Track"
    BEGIN
        MENUITEM "Oval Track",       MENU_TRACK_ID_OVAL
        MENUITEM "Oval Track - Narrow", MENU_TRACK_ID_OVAL_NARROW
        MENUITEM "Medium Difficulty", MENU_TRACK_ID_MEDIUM
        MENUITEM "Medium Difficulty - Narrow", MENU_TRACK_ID_MEDIUM_NARROW
        MENUITEM "Difficult",        MENU_TRACK_ID_DIFFICULT
        MENUITEM "Difficult - Narrow", MENU_TRACK_ID_DIFFICULT_NARROW
    END
    POPUP "Help"
    BEGIN
        MENUITEM "About",            MENU_HELP_ABOUT
    END
END

```

```
//TwoViews_res.h

#define MENU_FILE_ID_OPEN1000
#define MENU_FILE_ID_CLOSE1001
#define MENU_FILE_ID_SAVE1002
#define MENU_FILE_ID_EXIT1003

#define MENU_CONTROL_ID_P2000
#define MENU_CONTROL_ID_PI2001
#define MENU_CONTROL_ID_PID2002
#define MENU_CONTROL_ID_BANG2003

#define MENU_CAR_ID_START3000
#define MENU_CAR_ID_REVERSE3001
#define MENU_CAR_ID_PAUSE3002

#define MENU_TRACK_ID_OVAL4000
#define MENU_TRACK_ID_OVAL_NARROW4001
#define MENU_TRACK_ID_MEDIUM4002
#define MENU_TRACK_ID_MEDIUM_NARROW4003
#define MENU_TRACK_ID_DIFFICULT4004
#define MENU_TRACK_ID_DIFFICULT_NARROW4005

#define MENU_HELP_ABOUT      5000

#define CURSOR_CROSSHAIR200
```

```

// Functions from T3DLIB2a.CPP
//
//      T3DLIB2.CPP - Game Engine Part II : Written by Andre LaMothe (LaMothe, 2002)
//      The following functions were added and/or modified for use with this simulation.
//

////////////////////////////////////

int DInput_Init_FFJoystick(int min_x, int max_x, int min_y, int max_y, int dead_zone)
{
    // this function initializes the joystick, it allows you to set
    // the minimum and maximum x-y ranges

    // first find the GUID of your particular joystick
    lpdi->EnumDevices(DI8DEVCLASS_GAMECTRL,
        EnumFFDevicesCallback,
        DInput_Enum_Joysticks,
        &joystickGUID,
        DIEDFL_ATTACHEDONLY | DIEDFL_FORCEFEEDBACK);

    // create a temporary IDirectInputDevice (1.0) interface, so we query for 2
    LPDIRECTINPUTDEVICE lpdijoy_temp = NULL;

    //if (lpdi->CreateDevice(joystickGUID, &lpdijoy, NULL)!=DI_OK)
    if (FAILED(lpdi->CreateDevice(joystickGUID, &lpdijoy, NULL)) )
        return(0);

    // set cooperation level
    //if (lpdijoy->SetCooperativeLevel(main_window_handle,
    //                                DISCL_NONEXCLUSIVE | DISCL_BACKGROUND)!=DI_OK)
    if (FAILED(lpdijoy->SetCooperativeLevel(main_window_handle,
        DISCL_EXCLUSIVE | DISCL_BACKGROUND)) )
        return(0);

    // set data format
    //if (lpdijoy->SetDataFormat(&c_dfDIJoystick)!=DI_OK)
    if (FAILED(lpdijoy->SetDataFormat(&c_dfDIJoystick2)) )
        return(0);

    // set the range of the joystick
    DIPROPRANGE joy_axis_range;

    // first x axis
    joy_axis_range.lMin = min_x;
    joy_axis_range.lMax = max_x;

    joy_axis_range.diph.dwSize      = sizeof(DIPROPRANGE);
    joy_axis_range.diph.dwHeaderSize = sizeof(DIPROPHEADER);
    joy_axis_range.diph.dwObj      = DIJOFS_X;
    joy_axis_range.diph.dwHow      = DIPH_BYOFFSET;

    lpdijoy->SetProperty(DIPROP_RANGE,&joy_axis_range.diph);

    // now y-axis
    joy_axis_range.lMin = min_y;
    joy_axis_range.lMax = max_y;

    joy_axis_range.diph.dwSize      = sizeof(DIPROPRANGE);
    joy_axis_range.diph.dwHeaderSize = sizeof(DIPROPHEADER);

```

```

joy_axis_range.diph.dwObj          = DIJOFS_Y;
joy_axis_range.diph.dwHow          = DIPH_BYOFFSET;

lpdijoy->SetProperty(DIPROP_RANGE,&joy_axis_range.diph);

// and now the dead band
DIPROPDWORD dead_band; // here's our property word

// scale dead zone by 100
//dead_zone*=100;
//dead_zone = 1000;
dead_zone = 0;

dead_band.diph.dwSize              = sizeof(dead_band);
dead_band.diph.dwHeaderSize        = sizeof(dead_band.diph);
dead_band.diph.dwObj               = DIJOFS_X;
dead_band.diph.dwHow               = DIPH_BYOFFSET;

// deadband will be used on both sides of the range +/-
dead_band.dwData                   = dead_zone;

// finally set the property
lpdijoy->SetProperty(DIPROP_DEADZONE,&dead_band.diph);

dead_band.diph.dwSize              = sizeof(dead_band);
dead_band.diph.dwHeaderSize        = sizeof(dead_band.diph);
dead_band.diph.dwObj               = DIJOFS_Y;
dead_band.diph.dwHow               = DIPH_BYOFFSET;

// deadband will be used on both sides of the range +/-
dead_band.dwData                   = dead_zone;

// finally set the property
lpdijoy->SetProperty(DIPROP_DEADZONE,&dead_band.diph);

/***** Turn off joystick's autocenter feature *****/
/*
DIPROPDWORD DIPropAutoCenter;

DIPropAutoCenter.diph.dwSize       = sizeof(DIPROPDWORD);
DIPropAutoCenter.diph.dwHeaderSize= sizeof(DIPROPHEADER);
DIPropAutoCenter.diph.dwObj        = 0;
DIPropAutoCenter.diph.dwHow        = DIPH_DEVICE;
DIPropAutoCenter.dwData             = FALSE;

lpdijoy->SetProperty(DIPROP_AUTOCENTER, &DIPropAutoCenter.diph);
*/
/*****

// acquire the joystick
//if (lpdijoy->Acquire()!=DI_OK)
if (FAILED(lpdijoy->Acquire()))
    return(0);

// set found flag
joystick_found = 1;

// return success
return(1);

```

```

} // end DInput_Init_FFJoystick

////////////////////////////////////////

int DInput_Read_Joystick(void)
{
    // this function reads the joystick state

    // make sure the joystick was initialized
    if (!joystick_found)
        return(0);

    if (lpdijoy)
    {
        // this is needed for joysticks only
        // if (lpdijoy->Poll() != DI_OK)
        if (FAILED(lpdijoy->Poll()) )
            return(0);

        // if (lpdijoy->GetDeviceState(sizeof(DIJOYSTATE), (LPVOID)&joy_state) != DI_OK)
        if (FAILED(lpdijoy->GetDeviceState(sizeof(DIJOYSTATE2), (LPVOID)&joy_state)) )
            return(0);
    }
    else
    {
        // joystick isn't plugged in, zero out state
        memset(&joy_state, 0, sizeof(joy_state));

        // return error
        return(0);
    } // end else

    // return success
    return(1);

} // end DInput_Read_Joystick

```



```

// T3DLIB2a.H - Header file for T3DLIB2a.CPP game engine library

// Modified from T3DLIB2.H, which was written by Andre LaMothe (LaMothe, 2002)

// watch for multiple inclusions
#ifndef T3DLIB2
#define T3DLIB2

// DEFINES ////////////////////////////////////////

// MACROS ////////////////////////////////////////

// TYPES ////////////////////////////////////////

// PROTOTYPES ////////////////////////////////////////

// input
int DInput_Init(void);
void DInput_Shutdown(void);

int DInput_Init_Joystick(int min_x=-256, int max_x=256,
                        int min_y=-256, int max_y=256, int dead_band = 10);
int DInput_Init_FFJoystick(int min_x=-256, int max_x=256,
                          int min_y=-256, int max_y=256, int dead_band = 10);

int DInput_Init_Mouse(void);
int DInput_Init_Keyboard(void);
int DInput_Read_Joystick(void);
int DInput_Read_Mouse(void);
int DInput_Read_Keyboard(void);
void DInput_Release_Joystick(void);
void DInput_Release_Mouse(void);
void DInput_Release_Keyboard(void);

// GLOBALS ////////////////////////////////////////

// EXTERNALS ////////////////////////////////////////

extern HWND main_window_handle; // save the window handle
extern HINSTANCE main_instance; // save the instance

// directinput globals
extern LPDIRECTINPUT8 lpdi; // dinput object
extern LPDIRECTINPUTDEVICE8 lpdikey; // dinput keyboard
extern LPDIRECTINPUTDEVICE8 lpdimouse; // dinput mouse
extern LPDIRECTINPUTDEVICE8 lpdijoy; // dinput joystick
extern GUID joystickGUID; // guid for main joystick
extern char joyname[80]; // name of joystick

// these contain the target records for all di input packets
extern UCHAR keyboard_state[256]; // contains keyboard state table
extern DIMOUSESTATE mouse_state; // contains state of mouse
//extern DIJOYSTATE joy_state; // contains state of joystick
extern DIJOYSTATE2 joy_state; // contains state of joystick
extern int joystick_found; // tracks if stick is plugged in

#endif

```