Effect of Scene Polarity and Head Orientation on Roll Illusions in a Virtual Environment

by

Adam Skwersky

B.S. Mechanical Engineering
Massachusetts Institute of Technology, 1994

SUBMITTED TO THE DEPARTMENT OF MECHANICAL ENGINEERING IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE IN MECHANICAL ENGINEERING
AT THE
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

SEPTEMBER 1996

Signature of Author:.....

Department of Mechanical Engineering
August 26, 1996

Certified by:.....

Charles M. Oman, Ph.D.
Senior Research Engineer and Senior Lecturer
Department of Aeronautics and Astronautics

Accepted by:.....

Ain Ants Sonin
Professor of Mechanical Engineering

The Effect of Scene Polarity and Head Orientation on Roll Illusions in a Virtual Environment

by
Adam Skwersky

Submitted to the Department of Mechanical Engineering on August 19, 1996 in Partial Fulfillment of the Requirements for the Degree of Master of Science in Mechanical Engineering

ABSTRACT

Three virtual scenes of decreasing polarity and increasing symmetry -- a furnished room (FR), an unfurnished symmetric windowed room (SR), and an unfurnished windowless dotted room (DR) -- were rotated CW or CCW at 30± 1.5°/sec about the subject's anterior-posterior axis using a Virtual Environment System (VES) prototypical of equipment that is planned for use in an experiment on human visual orientation on the 1998 "Neurolab" Space Shuttle mission. The system consisted of a DOS based PC (100 MHz Pentium), dual SPEA I860 Fireboards, and an Eyephone I Head Mounted Display (HMD). Experiment control software was developed for this project in C, using World Tool Kit (Sense8 Corp.).

Sixteen subjects viewed the scenes while in erect and supine postures and reported on their illusions, such as vection (self-motion), self-tilt and visual reorientation illusions (sudden exchanges in the subjective identity of walls, ceilings, and floors). A subject's vection latency (time until vection onset) and saturation (percentage of roll velocity perceived as self-motion) both demonstrated a sensitivity to scene content but only a slight dependence on head orientation. On average, subjects reported vection latency 2.1 seconds sooner for the FR than the DR, but only 0.2 seconds sooner when erect than supine. Head orientation played a more significant role in whether the subjects felt full 360° vection (full-tumbling) and the axis of this perceived tumbling. These experiments did confirm DeSouza's [1995] observation of more subjects reporting of vertical full tumbling sensation (VFT) when viewing a rotating furnished room from the erect as opposed to supine posture, but the effect of posture was not significant for individual subjects.

Subjects frequently experienced visual reorientation illusions as the rooms rotated. Subjects reoriented to all four surfaces, but the furnished and symmetric rooms had two preferred orientations (FR's floor and SR's floor or ceiling in lower visual field) which for 9 (57%) subjects triggered reorientations 20-25° earlier, and for the remaining subjects triggered reorientations at least twice as often. Reorientation behavior was interpreted in terms of a heuristic model. The frequency of full-tumbling illusions in a gravitationally vertical plane presumably mainly influenced by the strength of visual cues, and the frequency of horizontal full-tumbling illusions by the strength of vestibular cues.

Thesis Supervisor: Charles M. Oman, Senior Lecturer, Director of Man Vehicle Lab.
Supported by Grant NAGW 3958 from the National Aeronautics and Space Administration (MIT OSP 62760).

Acknowledgements

I'd like to start about by tremendously thanking Prof. Chuck Oman and Dr. Alan Natapoff. Without Prof. Oman's insight into the problems of visual-vestibular interaction you, the reader, wouldn't be holding this huge thing in your hand called a "Masters Thesis."

Of course, without Dr. Natapoff's help, I would not have finished a **huge** thesis. Dr. Natapoff was able to explain complicated and powerful statistical tests so that they could fit into a small portion of my head, while leaving enough room in my kepach to come up with creative explanations for what I saw in my data.

A special thanks goes to Christine Duffy. Without her support, I probably would not have finished this thesis. Not only did she keep encouraging me to finish already so we could take a long vacation, but she loaned me her powerbook which made it tremendously easier to get work done in the lab.

Of course, my parents and family deserve a lot of credit, too. As much as being excellent role-models, Mr. Skwersky and Dr. Skwersky were always there encouraging me and cheering me on. In the few weeks before I finished, my mom and dad actually called and left "sis-boom-bah" messages on my answering machine. They also spent a lot of time trying to decipher my earlier drafts, thanks! Dawn and Tina, many thanks to you too for being excellent sisters!

I can't forget to mention Tom Westcott. While I was performing experiments on subjects, he blocked out two weeks of his time so he could interpret for me at a moments notice. Leslie Regan and Joan Kravitz kept encouraging me to keep at it. Everytime I left their office, I was wearing a smile.

Finally, I'd like to thank the members of the Man Vehicle Lab: You've put up with my computer hogging long enough! Maybe soon I'll get my own powermac?

Table of Contents

ABSTRACT 2
ACKNOWLEDGEMENTS 3
TABLE OF CONTENTS 4
LIST OF FIGURES 6
LIST OF TABLES 8
LIST OF ABBREVIATIONS 9
1. INTRODUCTION 10
2. APPARATUS 16
2.1 Why Virtual Environments? 16
2.2 Proposed Virtual Environment System 18
3. INDEPENDENT VARIABLES 23
3.1 Scenes 23
3.1.1 Furnished Room 23
3.1.2 Symmetric Room 24
3.1.3 Dotted Room 25
3.2 Head orientation with respect to gravity 27
3.2.1 Erect 27
3.2.2 Supine 28
3.3 Direction 29
3.4 Experiment Schedule 29
3.5 Subjects 30
4. DEPENDENT VARIABLES 32
4.1 Objective versus subjective measurements 32
4.2 Reporting of event data 35
4.2.1 Vection and latency 35
4.2.2 Reorientations 36
4.2.3 Single and mixed presentation block design 38
4.3 Categorizing of sensations 39
4.3.1 An improved system 40
4.3.2 Subjects' Responses 42
4.3.3 Predicted effects of scene content on orientation 44
4.3.4 Predicted effects of posture on orientation 47
5. RESULTS 51
5.1 Vection 51
5.1.1 Full-Tumbling 52
5.1.2 Latency of Vection 60
5.1.3 Saturation of Vection 62
5.2 Reorientations 65
5.2.1 Polarization effects: Early and On-Time Classification 66
5.2.2 Posture and Perceived Frontal Plane Effects 82
5.3 Tilt Illusions 82
5.4 Perceived Frontal Plane Orientation 84

6. CONCLUSIONS 88
6.1 Vection 88
6.2 Reorientations 91
6.3 Future Research 93
7. REFERENCES CITED 95
8. APPENDIX: ANOVAS 97
8.1 Latency 97
8.2 Saturation 97
8.3 Reorientation Angle 97
8.3.1 All Subjects 97
8.3.2 Early Subjects 98
8.3.3 On-Time Subjects 98
8.3.4 Frequency of Reorientations 99
9. APPENDIX: RECRUITMENT AND TRAINING 100
10. APPENDIX : EXPERIMENTAL SOFTWARE 111
10.1 Software Design 111
10.1.1 Background and Requirements 111
10.1.2 Implementation 112
10.1.3 Modules 114
10.2 Using the Software: Instructions 115
10.3 Source Code 115
10.3.1 central.mak 116
10.3.2 central.lnk 117
10.3.3 subjdb.h 117
10.3.4 expmod.h 118
10.3.5 runexp.h 119
10.3.6 fplan.h 119
10.3.7 move.h 120
10.3.8 datarec.h 120
10.3.9 dataread.h 121
10.3.10 joysense.h 121
10.3.11 statmod.h 121
10.3.12 dirutils.h 122
10.3.13 myprint.h 122
10.3.14 subjdb.c 122
10.3.15 expmod.c 131
10.3.16 runexp.c 134
10.3.17 spin.c 138
10.3.18 fplan.c 141
10.3.19 move.c 143
10.3.20 datarec.c 146
10.3.21 dataread.c 148
10.3.22 joysens.c 150
10.3.23 joyutil.c 150
10.3.24 statmod.c 152
10.3.25 myprint.c 165
10.3.26 dirutils.c 165

List of Figures

FIGURE 2-1: THE VIRTUAL ENVIRONMENT SYSTEM 19

FIGURE 3-1: FURNISHED ROOM (FR). 24

FIGURE 3-2: SYMMETRIC ROOM (SR). 25

FIGURE 3-3: DOTTED ROOM (DR). 26

FIGURE 3-4: SUBJECT IN THE ERECT POSTURE. 28

FIGURE 3-5: SUBJECT IN THE SUPINE POSTURE 29

FIGURE 4-1: OBJECTIVE AND SUBJECTIVE MEASUREMENT BLOCK DIAGRAM. 33

FIGURE 4-2: DETERMINING REORIENTATION SURFACE FROM REORIENTATION ANGLE 38

FIGURE 4-3: INTERNAL STATE REPRESENTATION MODEL OF SENSORY CONFLICT WITH PAST EXPERIENCE 43

FIGURE 4-4: VISUAL, GRAVITATIONAL, IDIOTROPIC VECTORS 48

FIGURE 5-1: NUMBER OF SUBJECTS FEELING VFT "N" TIMES (ERECT). 54

FIGURE 5-2: NUMBER OF SUBJECTS FEELING VFT "N" TIMES (SUPINE) 54

FIGURE 5-3: NUMBER OF SUBJECTS FEELING HFT "N" TIMES (ERECT). 55

FIGURE 5-4: NUMBER OF SUBJECTS FEELING HFT "N" TIMES (SUPINE) 57

FIGURE 5-5: MEAN LATENCIES FOR EACH SUBJECT WITH STANDARD ERROR BARS (Σ_{MI}) OF THE MEAN. 62

FIGURE 5-6: MEAN SATURATION FOR EACH SUBJECT WITH STANDARD ERROR BARS (Σ_{MI}) OF THE MEAN. 64

FIGURE 5-7: CODING OF SURFACES IN A ROOM DEPENDS ON ROTATION DIRECTION 65

FIGURE 5-8: REORIENTATION ANGLE (Θ) AND SURFACE OF 65

FIGURE 5-9: DETERMINATION OF EARLY OR ON-TIME BEHAVIOR 67

FIGURE 5-10: FURNISHED ROOM: EARLY AND 68

FIGURE 5-11: SYMMETRIC ROOM: EARLY AND 69

FIGURE 5-12: DOTTED ROOM: EARLY AND 70

FIGURE 5-13: EARLY SUBJECTS' MEAN REORIENTATION ANGLES FOR THE FLOORS AND CEILINGS OF FR AND SR. 75

FIGURE 5-14: EARLY SUBJECTS' MEAN REORIENTATION ANGLES FOR THE WALLS OF THE FR AND SR. 76
FIGURE 5-15: ON-TIME SUBJECTS' MEAN REORIENTATION ANGLES FOR THE FLOORS AND CEILINGS OF FR AND SR. 77
FIGURE 5-16: ON-TIME SUBJECTS' MEAN REORIENTATION ANGLES FOR THE WALLS OF FR AND SR. 78
FIGURE 5-17: SUBJECT MAKING POSTURAL ADJUSTMENTS DURING THE PASSAGE OF A SURFACE 83
FIGURE 5-18: FREQUENCY OF VFP PER SUBJECT (ERECT) 86
FIGURE 5-19: FREQUENCY OF VFP PER SUBJECT (SUPINE). 86
FIGURE 9-1: DOTTED WALL TRAINING STIMULUS. 100
FIGURE 9-2: FURNISHED ROOM TRAINING STIMULUS 101
FIGURE 10-1: MODULAR EXPERIMENT DESCRIPTION. 113
FIGURE 10-2: EXPERIMENTAL SOFTWARE USER INTERFACE 116

List of Tables

TABLE 3.1: SCHEDULE OF STIMULUS 31

TABLE 4.1: A NEW PARADIGM FOR REPORTING SENSATIONS 42

TABLE 5.2: FULL-TUMBLING IN THE FIRST AND LAST RUNS OF THE FR AND DR WITH SUBJECT IN THE ERECT POSTURE ($X^2_{(3)} = 0.881$; P=0.83). 59

TABLE 5.3 FITTED LSM VECTION LATENCY AVERAGED ACROSS SUBJECTS 60

TABLE 5.4: REPORTED VECTION SATURATION (%) BY ROOM AND POSTURE 63

TABLE 5.5: VECTION SATURATION BY BLOCK 64

TABLE 5.6: PATTERNS WITHIN EARLY AND ON-TIME REORIENTATION 71

TABLE 5.7: PATTERNS BETWEEN EARLY AND ON-TIME REORIENTATION 71

TABLE 5.8: RANGE OF MEAN REORIENTATION ANGLE FOR EARLY AND ON-TIME SUBJECTS. 74

TABLE 5.9: FITTED LSM OF FREQUENCY REORIENTATIONS PER REFERENCE SURFACE 80

TABLE 5.10: FITTED LSM REORIENTATION ANGLE BY SUBGROUP AND SCENE (ALL SCENES). 80

TABLE 5.11: FITTED LSM REORIENT ANGLE FOR FURNISHED ROOM SURFACES 81

TABLE 5.12: FITTED LSM REORIENT ANGLE FOR SYMMETRIC ROOM SURFACES 81

TABLE 5.13: FITTED LSM REORIENT ANGLE FOR DOTTED ROOM SURFACES. 81

TABLE 5.14: NUMBER OF SUBJECTS FEELING VERTICAL MORE OFTEN IN ONE POSTURE 87

List of Abbreviations

(F,S,D)R.....(FURNISHED, SYMMETRIC, DOTTED) ROOM

(H)CNS.....(HUMAN) CENTRAL NERVOUS SYSTEM

(V,H)FP(I).....(VERTICAL, HORIZONTAL) FRONTAL PLANE (ILLUSIONS)

(V,H)FT.....(VERTICAL, HORIZONTAL) FRONTAL-PLANE FULL TUMBLING

ANOVA.....ANALYSIS OF VARIANCE

API .....APPLICATION PROGRAMMERS INTERFACE

COUHES.....COMMITTEE ON USE OF HUMANS AS EXPERIMENTAL SUBJECTS

CW, CCW.....CLOCKWISE, COUNTERCLOCKWISE

DOS .....(MICROSOFT) DISK OPERATING SYSTEM

FOV .....FIELD OF VIEW

FPS .....FRAMES PER SECOND

GLM .....GENERAL LINEAR MODEL

HMD.....HEAD MOUNTED DISPLAY

IHAF .....I HOPE IT'S ALMOST FINISHED!

IHTFP.....I HAVE THE FINEST PROFESSORS, I HOPE (IT'S) THE FINAL PRODUCT

LSM .....LEAST SQUARED MEANS

MIT .....MASSACHUSETTS INSTITUTE OF TECHNOLOGY

NASA.....NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

RGB .....RED BLUE GREEN

V, R .....VECTION, REORIENTATION BLOCK

VES .....VIRTUAL ENVIRONMENT SYSTEM

VRI .....VISUAL REORIENTATION ILLUSION

VRV, RVR.....MIXED BLOCKS

WSRT.....WILCOXON SIGNED RANKS TEST

WTK.....WORLD TOOL KIT™

1. Introduction

The objective of this thesis was to continue the development of a flexible virtual environment system (VES) for orientation research, and to use it to study the interaction of visual, vestibular, and haptic cues to roll orientation. The experimental approach was to provide conflicting sensory cues to subjects, who were trained to report on their tilt, vection and reorientation illusions. In order to rapidly design and perform this experiment on a large number of subjects, versatile and flexible VES experiment control software was developed. The software has already been adapted and reused in other VES experiments [11], and is being considered by NASA for use in their space-based VES.

Humans determine their exocentric orientation by combining visual, vestibular, haptic, and non-vestibular gravireceptic information -- all of which are egocentrically based. Usually, all these sensory modalities work synergetically to provide a consistent exocentric position. With the advance of technology, however, humans are being thrust into environments which require them to make unaccustomed egocentric-to-exocentric transformations.

One such environment is the microgravity of space, where gravitational (e.g. vestibular or gravireceptic) cues are nearly absent. Previous research [Oman, 1987] has shown that astronauts will compensate the lack of vestibular and other gravireceptic input by becoming more dependent on visual and haptic input. This illustrates the adaptability -- in spite of sensory conflict -- of human self-orientation determination.

In microgravity, astronauts are able to move around freely, and thus they view their environment from a variety of different orientations. Astronauts have reported a variety of striking and labile Visual Reorientation Illusions (VRI) in which "floors",

"walls", and "ceilings" exchanged identities [Oman 1986], [Oman, Lichtenberg, Money, 1986]. These illusions have been known to cause a number of human-factors problems -- including disorientation and space-sickness. As space agencies increase mission lengths, they will need to expand their study of these illusions. The research herein serves as a prelude to a 1998 Spacelab (Neurolab) experiment [Oman, Howard, Carpenter-Smith, 1995] proposed to better understand how well the human egocentric-to-exocentric transformation system will adapt to microgravity.

One does not need to be in microgravity to experience orientation illusions. On a plane accelerating for take-off, passengers are exposed to approximately 0.3g, and this acceleration is sensed in the passengers via otolith and/or haptic stimulation. When passengers looks straight along the fuselage of the plane, their senses may be "telling" them that they are looking diagonally upward, even though the plane is still horizontal on the ground. The passengers might look out the window and see that the plane is still flat on the ground and believe instead that the ground itself is tilted upward. This is an example of how the human central-nervous-system (HCNS) may misinterpret or be confused by certain combinations of sensory stimuli.

Researchers have constructed devices whose sole purpose is to confuse the part of the HCNS that determines orientation. Witkin and Asch [1948] demonstrated that illusory self-tilt can be induced in erect subjects by tilting a luminous 1 m square 28° to the left or right in their frontal-plane. Based on their observations of subjects' adjusting a rod to the perceived vertical, they concluded that the tilted horizontals and verticals (i.e., the frame) induced subjects to feel 6° of illusory self-tilt in the same direction of tilt as the frame. This suggested that frames were used by the CNS to determine exocentric orientation.

Held [1975] and Howard [1988] showed that up to 20° of constant self tilt can be induced in subjects by exposing them to large displays of dots rotating in their

frontal-planes. These displays contained only randomly placed dots, therefore the induced constant tilt must have been due to the movement of purely textural cues. Howard [1988] explored this further by building a large 9 ft diameter sphere that rotated around the subjects roll axis. The increase in fraction of the visual field taken up by dotted textures helped to induce an average self-tilt of 25°, with some subjects reporting up to 90° of self-tilt.

In building tilted as much as 8° by an earthquake, residents still believed that the (furnished) rooms inside were upright [Kitaharo and Uno 1967]. This suggested that familiar objects that tend to have constant positions or orientations with respect to gravity are used as reference points to determine self-orientation. Concisely stated, a fully furnished room is visually polarized by the corresponding placement of objects that provide up-down references.

One of the first ever devices built that demonstrated the combination of frame, motion, and polarity was a fairground attraction built even before all these papers were written. The "Haunted Swing" [Wood, 1895] was a large room that rotated about an axle through the walls of the room. Thrill-seekers sat in a gondola, which swung back and forth from the axle, and watched as the room rotated around them. Those in the gondola felt as if the room was stationary while they were rotating. Yet this observation did not conclusively show that the vestibular system was being overcome by visual cues, because, even if the observer was moving, his vestibular inputs would not register major changes in orientation. The centrifugal force acting on an observer would keep his otolith hair cells aligned with the resultant force acting along the body axis, thus the inversion of the body would not be registered by the vestibular organs. During the acceleration phase of the "Haunted Swing" ride, there would be some conflicting signals from the vestibular system, but it was still not strong enough to overcome the visual signals. A fuller

explanation was given by Howard and Childerson [1994]. They postulated that the only way to investigate extent to which visual stimuli could override a full-range of vestibular input would be to rotate the visual stimuli on or near the axis of visual rotation.

Experiments by Kleint [1937] used a furnished room that rolled 360° about the subject's visual axis. Some subjects experienced a feeling of complete 360° rotation, and others felt only an angular oscillation about the visual axis. Since the visual stimuli was rotating about the visual axis in those experiments, the otolith cues conflicted strongly with the visual cues. The vestibular system was sensing gravity directed along the body axis, but the visual system was perceiving a rotation about the subjects posterior-anterior axis. That subjects felt full-tumbling demonstrated the power that the combination of the three visual aspects (frame, movement and polarity) have over the non-visual gravireceptic senses. Unfortunately, Kleint did not present quantitative or comparative data.

Howard and Childerson [1994] and Allison, Zacher and Howard [1995], conducted experiments with real rotating rooms and classified subjects' sensations into four categories: (1) constant tilt, (2) alternating tilt, (3) full (360°) tumbling and (4) supine response. The new category, supine response, was the illusion of lying supine or in some plane other than the vertical, and was usually accompanied by a sensation of vection or tilt depending whether the subject felt completely supine. If subjects felt no illusions at all, they were classified as type (1), constant 0° tilt. Howard and Childerson used three different rotating rooms: furnished, dotted walled, and dotted sphere. They found that more subjects experienced full tumbling in the furnished room than in the other rooms. The combination of frame, polarity and motion was strong enough to overcome the conflicting otolith cues in 60% of the subjects. Compare this to the 13% who felt full-tumbling in the dotted-sphere, which lacked frame and polarity.

DeSouza [1995] extended Howard and Childerson's [1994] study by using, instead of a real room, a virtual furnished room generated by a virtual environment system (VES). He also performed tests on subjects with different head-orientations. He found that the previous categories were insufficient to classify his subjects' responses. Subjects were retrospectively classified with a new set of categories based on descriptions of their illusions. In the cases where subjects felt no vection or tilt, DeSouza further specified whether the person felt gravitationally erect or supine. Still, there were several illusions that did not fit into any of his or his predecessors' categories. One subject reported constant self tilt and two reported alternating tilt about the earth vertical axis. DeSouza's surprising result was that 75% (9) of his subjects felt full tumbling in an earth vertical plane when they were erect, but only 33.3% (4) when supine. DeSouza believed that this was because the conflict between mono-oriented objects (aligned horizontally with respect to gravity) and the vestibular input (aligned vertically) prevented subjects from feeling full-tumbling in the supine posture. DeSouza also reported that when subjects were following an object at the periphery of the display they reported similar sensations as when they were gazing at the center of the display.

This study takes DeSouza's research a few steps further. In order to contrast within-subject to between-subject effects, a new experimental design was developed that increased the number of trials and repetitions of conditions. Not only were more runs-per-subject performed, but two more scenes were added to provide a richer set of independent variables. In this experiment, a different set of categories was used by subjects to describe their sensations. In addition, the phase and frequency of occurrence of visual reorientation illusions were systematically examined. The experiment design also let us look for effects of order and method of reporting. Also, DeSouza used a head-tracker in part of his experiment, but found it detrimental for sustaining vection in subjects. Since the goal of

this thesis was to study effect of head-orientation and scene content, not head-tracking, on roll illusions, the head tracker was not used in this study. Finally, in this experiment, more time was invested in training subjects.

The next chapter describes the apparatus, and the following two chapters present the variables (dependent and independent) used in this experiment. In these two chapters, the choice of variables is explained and a new categorization scheme which was used by subjects to describe their dominant sensations is outlined. Results and discussion are provided in Chapter 5, followed by conclusions and suggestions for future research in Chapter 6. Analyses of Variance (ANOVAs) on the collected data are listed in Appendix: ANOVAs, while Appendix : Experimental Software describes the software used to run the experiments.

2. Apparatus

The most important part of the apparatus was undoubtedly the virtual environment generator. Presented first is the rationale for using a virtual as opposed to real environment, and second, a description of the VES used in this experiment.

2.1 Why Virtual Environments?

Visual-vestibular interaction experiments have required environments realistic enough to "fool" subjects. One common technique has been to build a mock-environment and move it around the subject, but this equipment was generally expensive, required a large amount of lab space, took a long time to operate, and demanded regular maintenance. Moreover, large rotating rooms cannot be flown on a space shuttle or space station. This is a niche ideally suited for a virtual-reality (VR) stimulus device.

A VES provides the researcher with a simulated reality in which to explore interaction of visual and vestibular perception. It takes up only a single desktop (or less if its a portable design), and is more adaptable to a variety of experiments. Flexible experiment control software and rapid prototyping of virtual scenes reduce the time needed to design complex perceptual experiments. Consider, for example, an experimental design in which 3 different scenes were cycled for 24 runs, as was done in this research. A VES can switch between the scenes instantaneously, whereas a physical rotating room, even a miniature model of one, would take some time to alter. Experiments can be designed and carried out sooner with the VES, and less time means less cost.

While money has been a big issue in research, especially with government down-sizing and reductions in research spending, space has also increasingly become an issue. While it has often been said that "smaller is better", this is especially true on the frontier of space exploration. Compactness has been a primary design requirement for most space-related experiments. Previous visual-vestibular research in space, such as the rotating dome experiment, has used relatively small mechanical contraptions. The rotating dotted dome itself was unalterable, presenting only a spinning dotted surface [Young, et al, 1993]. Proposals for future experiments in space have included deeper studies on the effects of various visual cues on motion perception, and require a wider range of neural-stimulus. It has been expected that the Neurolab mission ('98) will launch with a sophisticated VES that will provide the capability to explore the effects of more complex visual stimuli.

While the engineering world continues to improve the finer aspects of VR, such as haptic stimulation, it has already achieved (and is still achieving) remarkable success with the visual aspects. Displays are rapidly improving, gaining in clarity, resolution and field-of-view. Despite the multitude of advances in VR technology, however, there has not been a significant amount of visual-vestibular research using virtual environments. While previous experimenters have used large displays [Held, 1975] or rooms [Kleint, 1937] [Howard and Childerson, 1994] [Allison, Zacher, and Howard, 1995] to study various rolling perceptions, DeSouza [1995] was one of the first to use virtual rotating rooms to study orientation perception in a conflicting visual-gravitational field.

One drawback of using computer-generated environment has been the limited field-of-view of the available head-mounted displays (HMD's). Size of field-of-view (FOV) has already been linked to the strength and type of roll-illusions reported by

subjects [Zacher Allison and Howard, 1995]. While the HMD (VPL Eyephone I) used in DeSouza's experiments had a horizontal FOV of 90° (60° overlap), which was above average on the market at the time of its purchase, its vertical FOV was only 58°. When reporting self-tilt, subjects have traditionally been asked to align a rod with their perceived angle of tilt. The HMD's rectangular-apertures, however, make questionable the use of virtual "rods" for subjects to report on tilt. Subjects might be biased toward aligning rods with the vertical boundary of the visual field. This made it necessary for subjects to estimate angles verbally. Finally, limited resolutions have hindered the realism of virtual environments. Since screen-resolution has been important in object-recognition, which has been associated with roll-illusions [Howard and Childerson, 1994], this technological constraint effects the ultimate ability of a virtual-environment system to induce roll-illusions.

2.2 Proposed Virtual Environment System

The virtual environment system (VES) used in this experiment, shown in Figure 2-1 was the same one used by DeSouza [1995], with the exception of completely rewritten experiment control-software (see Appendix : Experimental Software on p. 93). The hardware consisted of a 90 MHz Pentium-based microcomputer with various enhancements and add-ons. A stereoscopic head-mounted-display (HMD), the VPL Eyephone, was driven by dual SPEA i860 Fireboard graphics cards. The Fireboards were used by Sense8™'s WorldToolKit™ (WTK) version 2.0 for DOS. WTK provided an application programmer's interface (API) in C for developing virtual-reality applications. Code was compiled using Metaware's High C\C++ v3.31. WTK for DOS required a 32 bit DOS-Extender, hence PharLAP's TNT DOS-Extender was used. The Fireboards provided two channels of RBG output, which were converted -- using two Harmonics

Research part CV121A converters -- to the NTSC format that the VPL Eyephone needed. The right eye video was also routed to a TV allowing one to monitor the experiment. DeSouza [1995] has provided a complete description of this system, including Eyephone viewing parameters. The head-tracker was not used in this experiment because it had a detrimental effect on the frequency and strength of illusions in DeSouza's subjects.

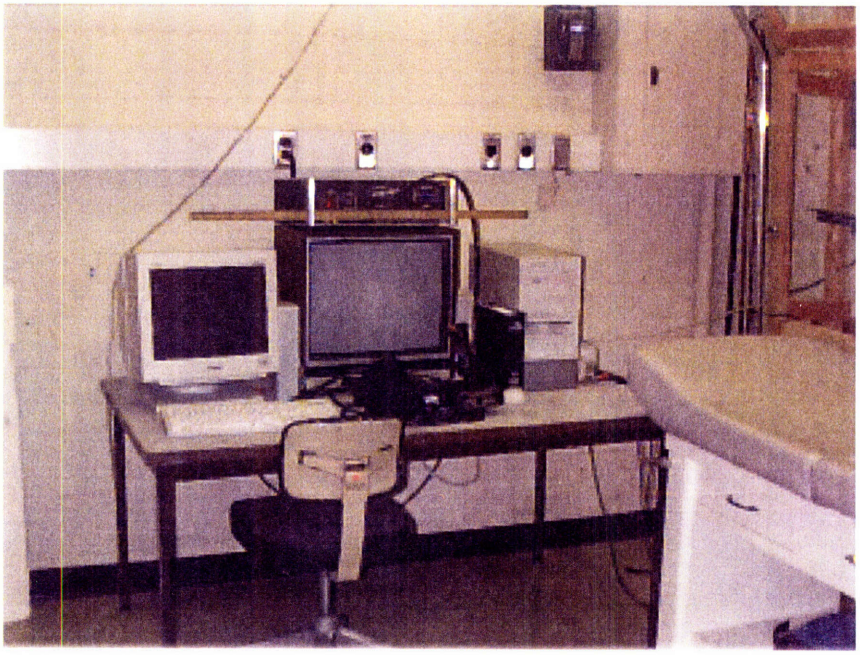

Figure 2-1: The Virtual Environment System

Sense8™ said that the speed and type of the CPU has directly effected the rate of the rendering process. This was hard to verify since only one computer was available to operate on. There was, however, a 5% increase in rendering speed when the compiler was told to make code specifically for Pentiums, as opposed to making executable that could also be run on a 486 as well. This 5% improvement was probably a result of the compiler taking advantage of the scheduling optimizations that the Pentium thrives on. These improvements, however, were limited because WTK's engine came as binary libraries, not as C source code. Speed increases from compiler optimizations were small, since most of the CPU time was taken up by WTK's rendering engine, and not the

relatively smaller portion of code that was built around the engine to create operator-interface.

The three scenes used in this experiment will be described in detail in the next chapter, but they are briefly discussed here from a simulation perspective. The furnished room, with around 110 polygons (less than half of these were textured), rendered at 19.5 frames per second (FPS). The frame rate was not fixed by the WTK engine since it was free-running. Frame rate, therefore, was very sensitive to the percentage of view that was covered by textures. The rendering engine slowed to 15 FPS when the dotted room, with 5 fully textured polygons that cover the entire scene, was simulated.

Since instantaneous frame rate could not be measured accurately,[1] its average was used to determine how much to rotate the viewpoint in a given rendering cycle. To calculate the angle-increment, the following equation was used,

Equation 2-1

$$\Delta\theta = \dot{\theta}/FPS$$

where $\dot{\theta}$ was the desired speed of rotation and FPS was the expected average frame rate in frames-per-second. Since constant velocity rolling was simulated for this experiment, the viewpoint was rotated at equal increments for each rendering cycle To obtain the average FPS for a scene, the room was spun around 4 times and the total number of rendering cycles was divided by the total amount of time the simulation took. In the next chapter, average frame-rates are presented with the description of the rooms themselves..

---

[1] WTK provided a function "`Wtuniverse_framerate()`" that averaged framerate over the last 30 frames, but it did not provide a direct way to determine the time taken up by the last frame, hence instantaneous frame rate. Furthermore, the computer's system clock (accessed via DOS system calls) jumped in ~0.05 second increments. This explained why DeSouza, when trying to average the furnished-room frame-rate (~20 FPS or ~0.05 seconds per frame) over two cycles, observed some cycles taking up half as much time has others.

When a room with unbalanced textures was simulated, however, the frame rate varied, and the scene slowed down and sped up cyclically. When simulating the virtual furnished room used in this experiment (shown in Figure 3-1), for example, the system's instantaneous frame-rate depended on the roll angle of the viewpoint. In general, WTK slowed down as the number of visible textured surfaces increased. At two orientations in the furnished room, 90° and -90° from the vertical, the room's textured ceiling and floor took up the larger part of the head-mounted display's screens (recall that the HMD was wider horizontally than vertically). At these orientations, the rendering process took longer, and these slowdowns caused artifacts of perceived acceleration that were obstacles to simulating constant rotation speed. While simulating the furnished room, for example, the WTK engine varied from 18.5 to 20.5 FPS. For a ω_0=30°/sec (π/6 rad/sec) constant rolling velocity, this translated to a ±1.5°/sec (i.e., ±5% of average rotation rate) variation in perceived velocity. If one assumes the virtual scene velocity follows the formula,

Equation 2-2
$$\omega = \omega_0 + {}^{p}\!/_{100}\,\omega_0 \sin(2\omega_0 t)$$

which is based on the frame rate varying between 100-p% and 100+p% of the average with velocity peaks occurring at 180° intervals (hence twice per virtual revolution), then differentiating this one gets the virtual scene acceleration,

Equation 2-3
$$\dot{\omega} = {}^{p}\!/_{50}\,\omega_0^2 \cos(2\omega_0 t)$$

Thus, based on a sinusoidally varying frame rate, the maximum virtual scene acceleration is $p\omega_0^2/50$. In order to prevent the HCNS from detecting a disparity in visual and vestibular cues, it would be necessary to keep p small enough so that the virtual scene acceleration is less than the threshold of perceived angular acceleration about a visual axis. When visual cues are absent, and rotations are applied about the earth vertical, the HCNS has been able to detect angular accelerations as small as 0.1°/s^2 [Howard, 1982],

presumably because of the high sensitivity of the human semicircular canals. For ω_0=30°/sec (π/6 rad/sec), therefore, one would need $p < 1/\pi$ (i.e., less than 0.32%). In other words, to prevent unwanted virtual acceleration cues, the frame rate variation would have to be less than ±0.32% of its average! This an order of magnitude smaller than the frame rate variation when simulating the furnished room. Theoretically, however, this threshold would be higher in the presence of a conflicting visual field, due to the ensuing confusion in the HCNS. If a subject's vestibular threshold is raised when contradictory visual cues are present, then she will feel vection in the furnished room, in spite of the visual acceleration cues that are unconfirmed by the vestibular system. Even if a subject's visually perceived acceleration is not confirmed by the subjects vestibular senses, she may still feel vection since the virtual velocity changes are small compared to the simulated constant velocity (<5%).

3. Independent Variables

Scene content, subject posture, and direction of scene rotation were the three independent variables chosen to characterize the way the HCNS resolves sensory conflict between the vestibular and visual senses. In a virtual furnished room, for example, conflict was generated in the subject when the room's visual down was not oriented with (1) her vestibular system's imposed down, or (2) with her body's axes when she was supine.

3.1 Scenes

This experiment used three scenes of decreasing polarity and increasing symmetry. The two characteristics were complementary, since adding polarity to a scene necessarily took away some of its symmetry. Previous research has suggested that type and strength of illusions reported by subjects were not sensitive subjects' direction of gaze [Allison, Zacher and Howard 1994] [DeSouza 1995]. Therefore, subjects were asked not to focus on any one point of the scene, but instead to take in the whole scene by gazing straight into it. Scenes were as texture-balanced as possible to minimize virtual acceleration cues. The mean and range of frame rate variation is given for each room.

3.1.1 Furnished Room

The presence of mono-oriented objects, objects that have a natural "upright" orientation, have been attributed to an increase in number of subjects feeling full 360° tumbling [Howard and Childerson, 1994]. Mono-oriented objects must have a readily identifiable "top" and "bottom", and the axis between the two must be aligned with gravity in everyday experience. Since it had many aligned mono-oriented objects, the first scene

(Figure 3-1) was a highly polarized furnished room (FR) that was identical to the one used by DeSouza[1995]. The room had a table with a computer on it, a door on the right wall, a bookcase with a leaning book on the far wall, paintings on the far and left walls, a carpeted floor and a tiled ceiling. All the objects in the room provided down cues, including the floor, since carpeting of that texture has rarely been found on a ceiling. The FR averaged 19.5 FPS and ranged between 18.5 and 20.5 FPS.

Figure 3-1: Furnished Room (FR).

3.1.2 Symmetric Room

Figure 3-2 shows a room with axi-symmetric windows and matching floor and ceiling tiling, but without any mono-oriented objects in it. (The windows were not mono-oriented because they could be rotated 90° and still be considered "upright".) This scene was the symmetric room (SR) used in this experiment. The SR was not entirely un-polarized since its symmetry gave the subject two orientations (0°, 180°) that could be interpreted as

upright. The room's residual polarity was hypothetically weaker than that of the FR, since it had no visually-aligned mono-oriented objects. Once her viewpoint started rolling, the subject would have had to follow the floor in order to keep track of it. The SR averaged 20 FPS and ranged between 19.5 and 20.5 FPS

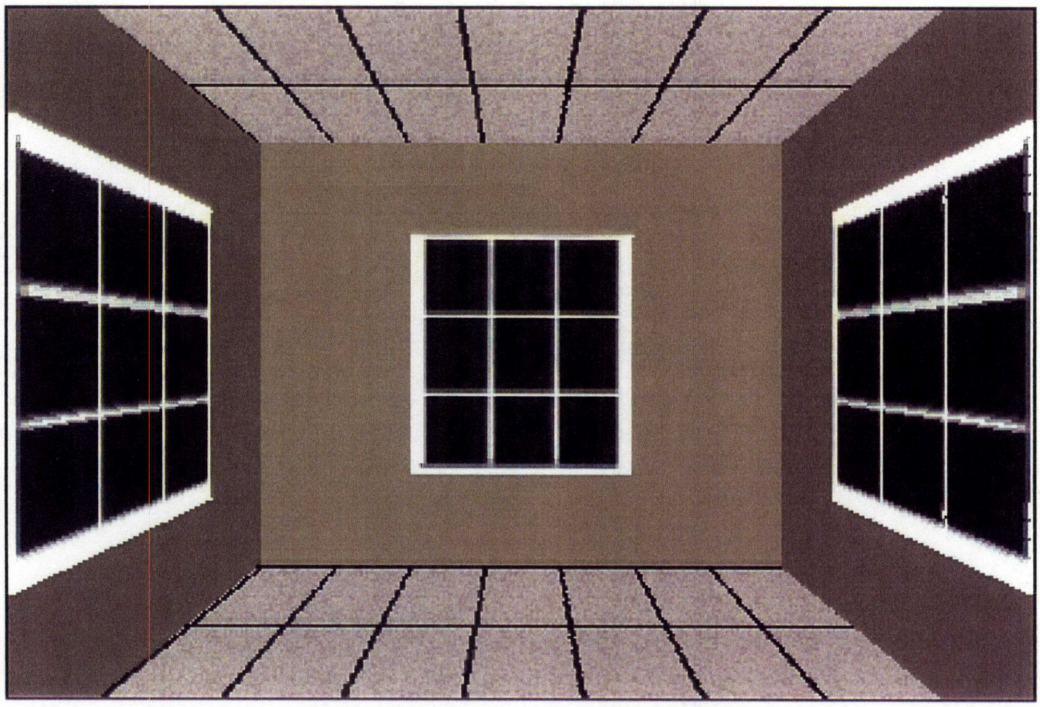

Figure 3-2: Symmetric Room (SR).

3.1.3 Dotted Room

When all the walls (including the ceiling and floor) were textured with colored dots (covering 10-15% of the surface) as shown in Figure 3-3, the room became a dotted room (DR), which had four room orientations (0°, 90°, 180°, 270°) that could be interpreted as upright. The DR had little polarity, but, in common with the previous two scenes, it had an identical frame. The DR averaged 15 FPS and did not exhibit much frame rate variation.

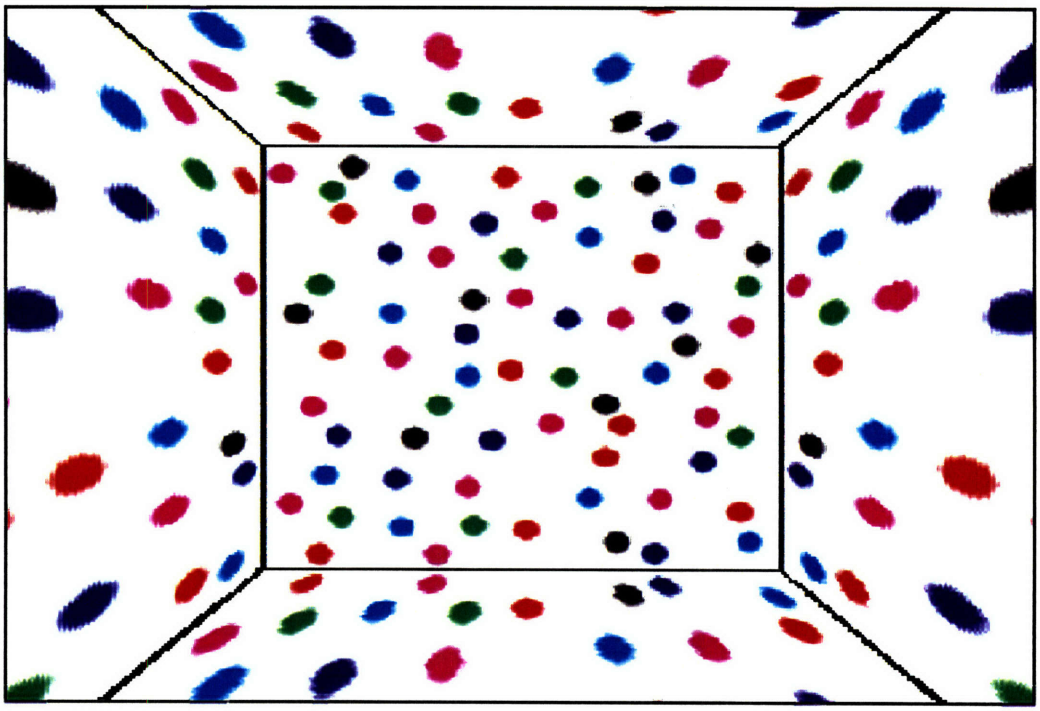

Figure 3-3: Dotted Room (DR).

The experimental research on non-framed dotted surfaces was quite extensive. In addition to previously mentioned research, Young, et al [1993] studied the effects of a rotating dotted dome on astronauts. He found that the dome induced stronger vection in astronauts in space than it did for astronauts oriented both vertically and horizontally with respect to gravity on earth. Howard and Childerson [1994] contrasted a dotted with a furnished room as part of their study on the effect of visual polarity and frame on body tilt. The furnished room used in our experiments was a virtual replica of the real rotating room used by Howard and Childerson [1994], but it had slightly different furnishings. Their dotted room was composed of smaller and variable-sized colored dots. Their dots were 1.3, 2.6 and 3.5 cm in diameter and covered 18% of the white surface. The current dotted room had dots 9 cm in diameter that covered 10-15% of the white surface. The windowed (symmetric) room was unique for this experiment. One other difference in the visual stimulus was that in this experiment, the rooms were spun at 30°/sec, as opposed to 15°/sec that Howard and Childerson spun their rooms at.

3.2 Head orientation with respect to gravity

The subject's posture, or gravitational orientation, was the second independent variable. Subjects were in an erect or supine posture with respect to gravity by sitting or lying down on a medical exam bed. These two positions provide different vestibular and haptic stimulation.

3.2.1 Erect

In the erect, or upright, posture (see Figure 3-4), subjects were free to lean in any one direction. There were several reasons for this. First, any effects the visual stimulus had on the body's closed-loop orientation control system could be observed. Posture-adjustments implied that the visual stimulus was effecting the body's perceived orientation. It was also believed that physically restraining the body's tilt would compromise the illusion of being immersed in the virtual environment. Even though sensory conflicts present in the experiment needed be controlled, unconstrained posture adjustments had be allowed. Even though those adjustments may have altered the vestibular system's orientation in gravity, it still provided observations on a closed-loop control system that included the perception and interpretation of visual cues. In any case, it would be impractical to control all the sensory conflicts. The subject, for example, still felt haptic cues on the seat of the pants, and from the weighty head-mounted display

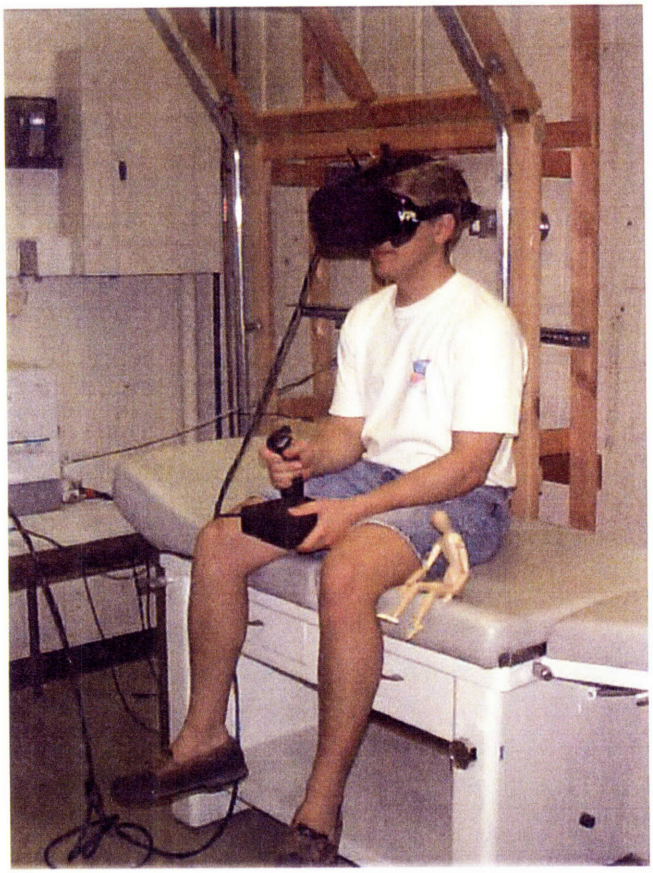

Figure 3-4: Subject in the Erect Posture.

3.2.2 Supine

In the supine posture, the subject laid flat on the bed with the viewing axis directed upward (tolerance was a ~3°) along the gravitational vector (see Figure 3-5), and was unable to make significant posture corrections. Also, the subjects felt haptic stimulation over a greater part of backs. In erect posture, subjects felt the bulk of their weight at the seat of their pants, whereas in the supine posture, their weight is spread out over their entire back. The supine posture has this advantage: the subjects' neck muscles no longer supported the full weight of the head-mounted display.

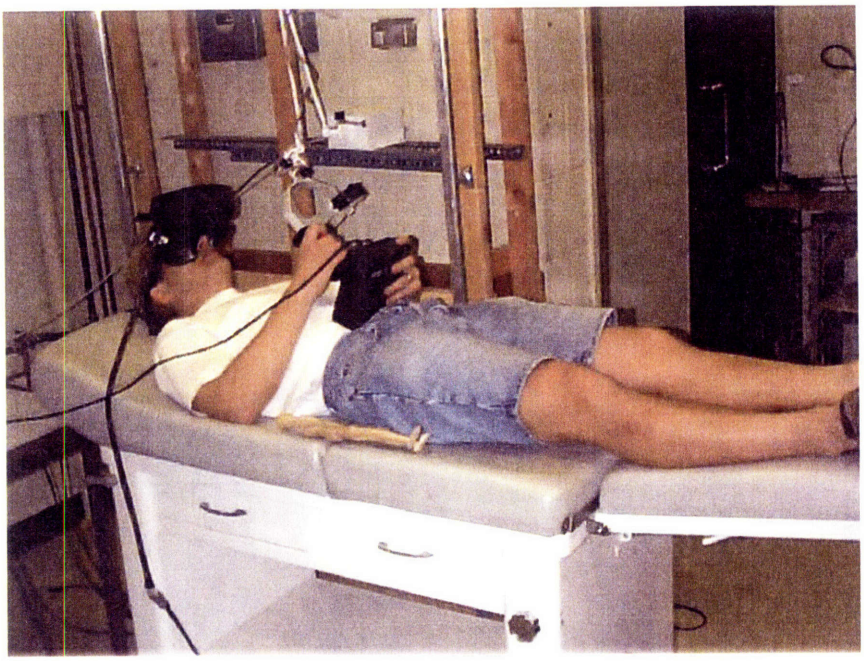

Figure 3-5: Subject in the Supine Posture.

3.3 Direction

The last independent variable was direction of rotation. Changing it reduced the subject's buildup of expectations due to adaptation or other effects in either direction. No direction was repeated for more than 2 consecutive runs. This variable should have uncovered any relationship between the dominant eye, the handedness, and the tendency to feel vection or reorientations in one direction more than the other. Previous research has shown no bias of vection for clockwise or anti-clockwise rotation [Howard and Childerson 1994] [Allison, Zacher and Howard 1995].

3.4 Experiment Schedule

Subjects underwent training on the first day, and the actual experiments were performed on the second and third days. (See Appendix 9 for detailed description of training regimen.) On each experiment day, twenty-four combinations of independent variables were

presented in an order shown in Table 3.1. The sequence was designed to balance out the effects of presentation order of the scenes, hence no scene appeared twice in a row. The posture variable, however could be similar twice (only once it appears three times) in a row. In the table, the independent variables occurred as evenly as possible throughout the experiment.

3.5 Subjects

The experimental protocol was approved by the MIT COUHES. Subjects were recruited by advertisement, and all were naive as to the purpose of the experiment. Subjects were not allowed to participate if they had serious medical conditions that could be aggravated by becoming motion sick. Subjects were given an information packet and questionnaire (see Appendix: Recruitment and Training) that they were to read and fill out. The questionnaire was used to determine the suitability of the subject and to record biographical information relevant to the study. The experiment design required 16 subjects to participate in the experiment. Twenty-one people were recruited but 5 of them developed motion sickness symptoms and chose to withdraw. Eight of the sixteen subjects were male, 3 had previously had dizzy spells, 8 had experienced motion sickness before, 9 had dominant right eyes (6 left, 1 even), 12 were right handed, 5 were "cross-wired" -- having dominant eye and hand on different sides, all but one had normal peripheral vision and depth perception. 11 subjects have used corrective lenses. The ages ranged from 18 to 54 years, with a mean of 22.9 years and standard deviation 8.7 years. All of the subjects said they had normal hearing and balance. Most people who required corrective lenses either wore contacts or did not wear glasses during the experiment. One subject very near-sighted subject wore glasses underneath the HMD.

Run #	Scene	Posture	Direction
1	FR	erect	CCW
2	SR	supine	CW
3	DR	erect	CCW
4	SR	erect	CW
5	DR	supine	CCW
6	FR	supine	CCW
7	DR	erect	CW
8	FR	supine	CW
9	SR	supine	CCW
10	FR	erect	CCW
11	SR	supine	CW
12	DR	supine	CW
13	SR	erect	CCW
14	FR	supine	CW
15	DR	erect	CCW
16	FR	erect	CW
17	DR	erect	CCW
18	SR	supine	CW
19	DR	supine	CW
20	SR	erect	CCW
21	FR	supine	CW
22	SR	erect	CCW
23	FR	erect	CCW
24	DR	supine	CW

Table 3.1: Schedule of Stimulus

4. Dependent Variables

This chapter discussed the selection of dependent variables that would best elucidate the cognitive processes that a person would undergo during this experiment. First, some principles related to subjective and objective measurements were laid out to guide the process of the variable selection. The next few sections describe a set of variables that were used in the experiment, and the last section will laid some groundwork for the interpretation of subjects' responses.

4.1 Objective versus subjective measurements

It was desirable to design the least intrusive, yet most flexible reporting system for the subjects. This involved a consideration of subjective and objective forms of measurements. When performing experiments on animals, researchers have been limited to objective forms of measurement by the inability to communicate effectively with their subjects. When performing experiments on humans, however, researchers were able to make subjective measurements (which allowed monitoring of subjects' perceptions). The objective and subjective measurement processes were diagrammed in Figure 4-1.

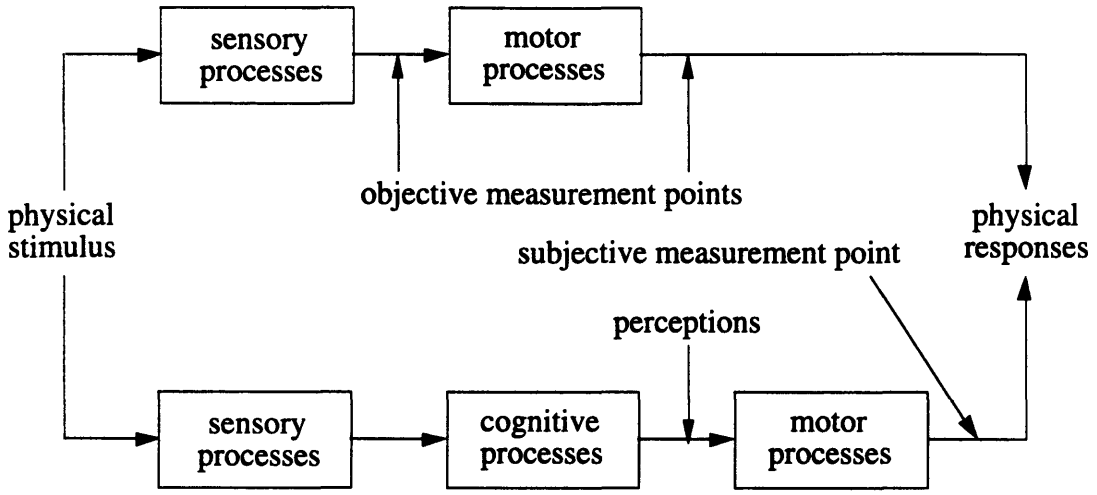

Figure 4-1: Objective and Subjective Measurement Block Diagram.

Both subjective and objective measurements have been suitable for different perception-related research-goals. In Anderson and Dyre [1989], researchers used objective measurements to study the frequency response of human sway to optical-flow stimuli. Their results showed that optical flow excites natural sway according to the optical-flow frequency. They also showed that excited sway tapers off at higher frequencies. This kind of result could only be obtained from using explicit objective measurements of sway. One could not ask the subject "what frequency and magnitude are you swaying at?" and expect accurate results. Further, the usefulness of the objective measurements depended on an unconscious transformation of optical-flow (sensory input) into muscle-activity (output) causing the sway. The sway was a by-product of the body's internal postural control system. The subjects in that experiment were not told to sway with the room, but were unconsciously responding to a change in the perceived environment.

A subject's perceptions cannot be directly measured by experimenters, thus they must be reported in some subjective and cognitive way by the subject herself. In many experiments, for example, researchers have been interested in perceptual effects or events that have no known manifestation in unconscious physical activity. For example, the onset

of vection in virtual environment experiments has not persistently produced common muscle activity in all humans. Also, the intensity, or saturation of vection cannot been measured without the subject's cooperation. They inherently must be determined through some sort subjective report. Researchers have depended on the subjects to consciously give reliable reports on their vection, and, at the same time, hoped that the act of inspection and reporting would not distract the subjects, thereby tainting the results.

One cannot always be sure that objective measurements were not in fact subjective measurements. To a person who is not familiar with natural sway in humans, the sway in Anderson and Dyre [1989] might have been interpreted as the subjects' actively trying to maintain their upright orientation in the face of visually-induced acceleration sensation. Orientation-control was something we have learned to do unconsciously. Walking on a ramp, most people have automatically slanted themselves so that they were walking upright. Were the subjects' natural sway really subconscious or were subjects completely aware of the posture corrections they were performing? Both objective and subjective measurement types involved a mapping of senses to action. They differed in that subjective measurements were more sensitive to the subjects' cognitive processes. On the other hand, a subject's reporting actions could become almost like reflexes if she was trained long enough.

In any case, the experiments occurred under the assumption that the subjects were operating in the region where their act of reporting had a minimal effect on their perceived state. Ideally, the subjects reported their senses accurately and promptly. Thus, it was desirable to have all subjects reporting with as little cognitive activity and as much reflex as possible without reducing the validity of their report. It became important, therefore, for all subjects to undergo symmetric and sufficient training. The regimen that was used can be found in Appendix 9. It was also important to design the reporting

methods so that they had the smallest impact on the subject's accuracy. The next section describes such a system in detail.

4.2 Reporting of event data

The experiment was divided into runs in which a virtual scene appeared to make 3 full rotations in 36 seconds. Data were collected during and after a run. The onset of vection, cessation of vection, and the feeling of being reoriented were events that can occur more than once in a given run. They were described by the time and angle (of the virtual image) at which they occur, and the subject pressed a joystick button that caused the data to be recorded.

After the run, the subjects described the overall sensations they had during it. Previous research on conflicting visual-vestibular stimuli asked subjects to choose among pre-defined categories to report what they felt [Howard and Childerson, 1994], [Allison, Zacher, and Howard 1995] [DeSouza 1995]. This experiment used an improved categorical approach described below.

Since subjects needed a simple but sure way to specify when an event occurred, protected against inaccuracies and human error, they required a peripheral device connected to the computer running the virtual environment. The experimental software could then accurately record the time and angle at which the event occurred.

4.2.1 Vection and latency

Subjects pressed an index-finger trigger button while they are feeling vection. The software recorded the pertinent data at the moments the button was pressed down and then released. The descriptive events for vection were the onset ("drop-in") and cessation("drop-out") of vection. This ensured that the window of vection is documented. The first drop-in was called the latency of the vection. It tells how soon after the visual stimuli (virtual rotating

scene) starts, that it had overcome the cues from the other senses. Latency of linear vection had been attributed to the absence of an acceleration cue to the body's gravireceptors at the onset of scene motion [Dichgans and Brandt, 1978; Henn, et al 1980]. A drop-out was hypothesized to indicate that the visual input is coming into strong sensory conflict with other senses or internal state representations. This could happen either because the visual cues were no long as compelling as they were previously, or because the CNS had increased its weighting of vestibular or haptic senses.

4.2.2 Reorientations

The illusion of reorientation was also reported using a single-click of the top joystick button. Reorientations were expected to occur when the scene was oriented such that the subject begun to reinterpret the surfaces of the room (e.g., wall as a ceiling). In order for this to have happened, the new interpretation of the surfaces must have been more compelling than the previous one. Even though subject's were asked to "look at the whole scene," most people are accustomed to watching where they walk (i.e., looking down), or looking straight ahead. The most common reorientation illusion, therefore, was expected to be due to the subject believing that the surface closest to their feet was a floor. Thus it may be useful to speak of a reorientation illusion as a subject's "reorienting to a new floor". Even though it was not known by the experimenter that the new floor was causing the reorientation, one can still assume, and pilot studies supported this, that the surface lowest (or approaching the lowest) in the visual field was the becoming a new floor. To analyze all reorientation events consistently among subjects, the reorientation angles can then be referenced from the floor.

It was believed that reorientations would be randomly distributed about a certain mean angle from a reference surface, and that the subjects would reorient sooner for

orientations of the scene that are easily interpreted as upright (i.e., highly polarized orientations of the scene). A scene orientation was more polarized than others when mono-oriented objects were aligned with the subject's head to foot axis. The most polarized orientation for the FR, for example, occurred when the reference surface was the floor.

Preliminary research showed that nearly all reorientations occurred between -60° and 30° from the reference surface. All reorientations between -60° and 30° from a surface, therefore, were initially assigned to that surface. In Figure 4-2, reorientations for surface n are denoted R_n. In a few cases, a subject might have pressed the button so lightly that it switched back and forth between the on and off state, recording several reorientations very quickly. A filter removed the surplus button presses by looking at how far apart they were: if two reorientation occurred within 30° of each other, and they were both on the same side of a cutoff angle at 0°, then the later one was ignored (Figure 4-2b and c). If the two reorientations were on opposite sides of the cutoff angle, then the first was counted for current surface, n, and the second for the next surface, $n+1$ (Figure 4-2d). This was for the case in which a subject reorients before the surface reaches the horizontal (0°), and again, for the next surface, just after the previous surface passes the horizontal (but is less than 30° from it). For the FR, a subject might have reoriented to the wall exactly when just before 0° but then again for the floor soon after. The software correctly interpreted the reorientations in this case. If the subject, however, had pushed the button just before 0° and, inadvertently, again just after 0°, the algorithm would incorrectly assign the accidental reorientation to the next surface. Fortunately the subjects never did this. After the few runs in which a subject reported two reorientations close to 0°, the subject was asked which surfaces she was reorienting too. The implementation of the algorithm can be found in the functions `find_wall()` and `performstats2_1()` in statmod.c on page 152.

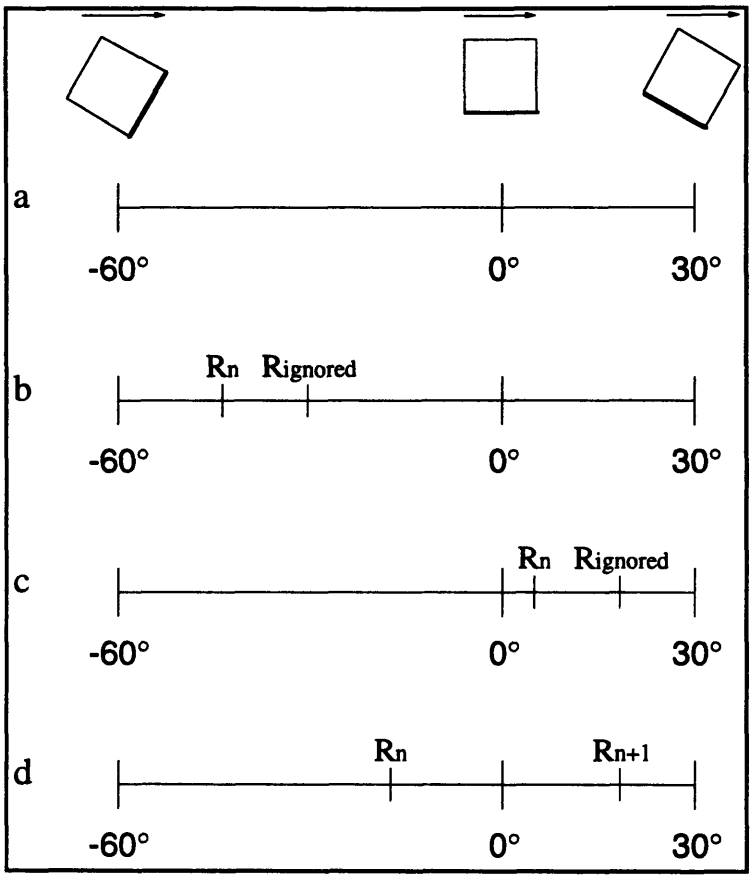

Figure 4-2: Determining Reorientation Surface from Reorientation Angle

4.2.3 Single and mixed presentation block design

It was believed to be confusing for the subject to report on both vection (V) and reorientations (R) in a single run. Therefore, the experiment included both single (V or R) and mixed (VRV or RVR) protocols . This design should have found (or ruled out) an effect of V-runs on R-responses on the same day. Then further research could focus efficiently on one or the other in a single experimental day. In each design, subjects underwent one day of training, immediately followed by two consecutive days of experiments with 24 (plus 3 warm-up) runs on each day. For both days, the order of scene, posture and direction followed the same factorial design described previously, but single

and mixed subjects reported on different events at different times. In the pure design, subjects concentrate on vection (V) events on one day and reorientation (R) events on the other day. Subjects who focused on vection events on the first and reorientations on the second day were called VR subjects (vection-reorientation). Subjects who reported on reorientation events on the first and vection on the second day, were called RV subjects.

Mixed subjects alternated RVR (or VRV) on a single day. To balance the combinations of the independent variables across the two days, the 24 runs were split into a 9-6-9 set. Subjects therefore reported on vection events for the first 9 runs, reorientation events for the next 6 runs and vection events (again) for the last 9 runs. The following day, they reported on reorientation first, then vection and then reorientation again. Hence this subject was called a VRVRVR or $(VR)^3$ subject, and her complement would be the RVRVRV or $(RV)^3$ subject. Thus there were two presentation orders for each type, single and mixed. Care was taken to ensure that the subject pool (including gender) was split equally into single and mixed designs.

4.3 Categorizing of sensations

Previous experiments have relied on the subjects' categorization of their dominant sensations [Howard and Childerson 1994], [Allison, Zacher and Howard 1995], [DeSouza 1995]. This constrained the subjects' responses to a limited pre-determined set. This situation was analogous to the projection of a 3-dimensional environment on 2-dimensional plane. While it's simpler to create a lower dimensional surface (i.e. photographs as opposed to holographs), the ability to describe accurately what was actually happening was lost. In DeSouza's experiments, subjects often described their sensations in ways that did not fit his pre-set categories. To accommodate this, DeSouza was forced to reclassify his subjects according to a new set of expanded categories. In this thesis, a new

framework was suggested for describing the subject's experience. The main purpose of the new system was to allow maximum flexibility in the subject's description of their senses without making the recording process and eventual statistical analyses unmanageable.

4.3.1 An improved system

Ideally, it was desired to have a reporting system which could classify any possible sensation, but it also had to be manageable and its data analyzable. A close inspection of the previously used categories showed that subjects are describing four aspects of their perceived orientation. Table 4.1 shows a new set of categories that are described below.

Angular vection, the sensation of moving at a angular velocity was the first mode of motion. The second mode was tilt, and consisted of constant and alternating tilt. Tilt and vection had to be reported independently, because in preliminary runs, subjects could (paradoxically) experience limited tilt and vection simultaneously. Choosing vection and tilt categories separately, subjects could report a richer variety of sensations.

The first two categories, (left-right) vection and tilt, occurred in the subjects perceived frontal-plane. In several cases subjects reported that their perceived frontal-plane was at different orientations with respect to gravity. Thus, the third and fourth modes of motion describes the inclination of the frontal-plane itself with respect to gravity. Subjects could report whether their frontal-plane seemed horizontal or vertical with respect to gravity, and they could modify this a little by specifying whether they felt slightly tilted forward to or backward from the rolling scene. Thus mode 4 was a modifier of mode 3.

In this new design, subjects had more flexibility in characterizing what they felt. Subjects can combine categories from the four groups to come up with over 36 possible descriptions. Typically, a subject would choose one category from each of the 4 groups, but they did not actually see Table 4.1; instead, they were asked 4 questions after

every run: "Did you feel vection?", "Did you feel alternating or constant tilt?", "Did you feel vertical or horizontal with respect to gravity?", and "Did you feel tilt toward or away from the scene?". In cases where subjects felt tilt, they were asked for its maximum magnitude. In order to ensure the categories and modifiers were understood by the subjects, a mannequin was used to demonstrate them.

To clarify the categorization system, some examples of subjects' sensations and how would they report them are presented. A subject who felt like she was rolling 360° around a horizontal axis (full-tumbling) would specify that she felt vection with no tilt, and that her frontal-plane was vertical. A subject who felt that she was having constant tilt, and that she was in a supine position with respect to gravity would specify that she felt no vection, constant tilt, and her frontal-plane was horizontal. A subject who felt that she was laying down watching the scene above her spin, and that her head was closer to the scene then her feet might report that she felt no vection, no tilt, and her frontal-plane was nearly horizontal with a slight forward tilt. A subject who felt like they were floating and looking down at the scene may have reported that she her frontal-plane was horizontal and that she was looking down. A subject feeling frontal-plane tilt 45° had several options. If the subject felt like she was nose-up 45° tilted back from the vertical, she might also report it as feeling horizontal with 45° tilt up. In this ambiguous case, the subject becomes the tie-breaker, deciding whether to specify tilt from the vertical or horizontal.

Rolling Motion in the perceived Frontal Plane		Inclination of perceived Frontal Plane with respect to gravity	
1	2	3	4
Vection	Alternating tilt	Vertical	Forward tilt
No vection	Constant tilt	Horizontal	Backward tilt
	No tilt		No tilt

Table 4.1: A New Paradigm for Reporting Sensations.

The improved system was easy to learn and use, but it was also powerful in its descriptive ability. There were now 36 possible categories of perceived motion, but not all categories needed to be explained individually. Subjects only had to learn the four groups describing perceived motion, each with only 2 or 3 possible categories. The simplicity was critical for this experiment, since the subjects could only recall a limited number of categories while trying to focus on their own feelings. It was also advantageous to separate subjects' sensations into various characteristics of their perception. The various groupings could then be analyzed independently of one another: how many subjects experienced vection for a given set of conditions, and how many felt horizontal in another set of conditions?

4.3.2 Subjects' Responses

Subjects were expected to interpret their visual, vestibular and idiotropic cues using an internal state representation, which had been implicated as part of a large heuristic mathematical model of the dynamics of space sickness [Oman, 1982]. (The idioptropic effect was hypothesized by Mittelstaedt [12]: when asked to indicate the vertical, subjects have a tendency to bias their subjective down so that it is more aligned with their longitudinal axis.) Subjects, however, depended not only on sensory stimulation, but also on memories of previous experiences. Encountering strange conflicting stimuli, subjects

tried to interpret them according to what they had experienced before, or they might even have discovered new ways to interpret their perceptions (and resolve conflict). In the next chapter data was presented in terms of this theory, which was depicted in Figure 4-3.

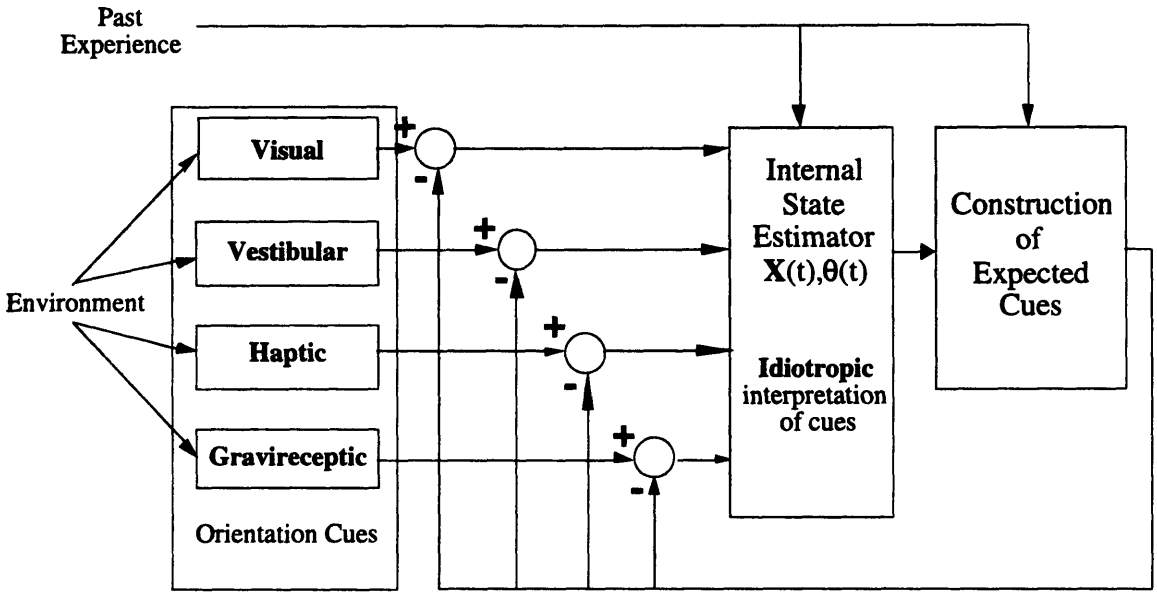

Figure 4-3: Internal State Representation Model of Sensory Conflict with Past Experience

If adaptation trends were exhibited in the results, this would support an internal-state-representation theory that included past experience. If the subjects started to feel one sensation more often than before, it could mean that their CNS' had learned a "new path" to resolve their sensory conflict. Some subjects, for example, may have discovered a new way of interpreting a scene that minimized the sensory conflict present. The SR and DR could potentially be interpreted in different ways depending on the subject's perceived orientation. This theory would suggest that a subject's CNS would continually try to minimize sensory conflict by revising its internal-state-representation of its body.

At a first glance, one may have believed that viewing the DR when supine would have elicited different results than viewing it when erect, because vestibular-visual

interaction was different for each posture. Using the model described recently, however, an idiotropic-visual interaction, which was present in a supine subject viewing the DR, could have the same effect as a vestibular-visual interaction in an erect subject. Thus, an idiotropic down might have mimicked gravitational down under some conditions, and this could have lessened the sensitivity of the independent variables to posture. If there was little difference between the erect and supine results for a given subject, it would have meant either that the subject was insensitive to gravitational cues and was relying more on visual cues (probably feeling stronger vection or tilt sensations), or that the subject was relying on her idiotropic down to determine her orientation (subject was probably feeling weak illusions).

4.3.3 Predicted effects of scene content on orientation.

The dimensions of each room were identical, therefore any difference in a subject's responses for the scenes must be due to scene content. Based on Howard and Childerson's [1994] results, this research assumed that the strength of vection will depend strong visual down cues (i.e., polarity) in the scene. In that experiment, more subjects felt complete 360° tumbling in a furnished room than in a similarly framed dotted room.

It was further believed that the room orientation at which certain illusions occur will depend on the symmetry of the room. In some strongly symmetric compartments of the space shuttle, astronauts experienced reorientation-illusions at 180° intervals. When in these symmetric environments, they sometimes reached the wrong way for an object usually found in one direction. Many astronauts have tended to carry their "down" with them, using their head-to-feet vector for orienting themselves to their tasks. When in an unfamiliar or non-polarized area of the space-shuttle, astronauts would feel that the surface nearest to their feet is a floor [Oman, 1987]

Subjects were expected to exhibit similar behavior with virtual rotating rooms. Even though the subjects were actually stationary, they would feel reoriented to various walls, ceilings and floors in the virtual environment as it moved around them. Subjects in the erect posture might believe the surface closest to their feet is the floor, the one closest to their head a ceiling, or those closest to their sides walls. A subject in the supine posture might also feel reorientations because her idiotropic vector is still imposing a head-to-foot down cue.

A reorientation in a rolling virtual room, therefore, is akin to a sudden reinterpretation of its surfaces (i.e., wall, ceiling or floor). A reinterpretation might draw on several senses (e.g., visual and gravireceptic), and thus be influenced by both scene content and head orientation. When a virtual room rotated, its surfaces spent only a finite amount of time "under" the subject's feet. A surface might be reinterpreted as a wall shortly after it moved off to the side. At that point, the surface moving to the lower part of the visual display might be reinterpreted as a floor. Before the event, the subject felt as if she was rolling off the floor to one side, but afterwards, as if she was rolling toward the "new" floor. In this example, the subject may also have felt an alternating-tilt to accompany the reorientations.

If and when a subject reorients in a scene might be related to the polarity in it. The visual polarity of a scene was a property of the scene itself. The polarities of individual objects combine with the room symmetry to produce a scene polarity which changes as a function of viewing angle. Even as the subject watches the FR floor roll around herself, she might still believe it is the real floor, but she might still feel reoriented when she detects a wall moving underneath her feet. This is because the surface below us, or lowest in the gravitational field, is usually a floor, and the CNS might interpret the wall as such. Thus, there was competition occurring the CNS between the recognition that the

surface is a wall, and the belief that the surface beneath us is usually a floor. As the subject sees the FR wall moving around their feet (the surface moving away is the floor), then she might either still believe it is the wall or she might, at some point, reinterpret it as the floor (thus feeling reoriented). The angle of the surface (from the horizontal) which the subject is reinterpreting as the floor is called the reorientation angle. The subject's reorientation to the FR floor might occur sooner than those for the walls since it is easier to believe (based on visual polarity cues) that the FR floor is actually the floor. Similarly, the subject might reorient sooner for the FR ceiling than for its walls, since it is easier for the subject to believe that the FR wall -- which was previously reinterpreted to be a floor -- is a wall again.

For the SR, both ceiling and floor (which are identical), are more easily interpreted as floors than are the SR's walls, especially since the walls have windows -- which are very rarely found on floors. Again, there is there is also the possibility that, because of the windows, the subject might not even reorient for the walls of the SR. If subjects do reorient for the SR's walls, they would probably wait until the walls are exactly "beneath their feet".

In the DR, all surfaces are almost identical. Subjects should not exhibit any difference in reorientation angle for the surfaces, since each surface is equally believable (based on visual cues) as a floor, wall or ceiling. The DR -- a room without any polarity -- should serve as a control for the reorientation data. The contrast of reorientation results between the FR, SR and DR will present, if any, a significant effect of visual polarity on the occurrence of and/or angles of reorientations.

4.3.4 Predicted effects of posture on orientation.

The posture variable provided two different types of sensory conflict. The three vectors that represent the three determinants of orientation -- gravity, vision and idiotropic perception -- and their direction for each posture were depicted in Figure 4-4. These vectors represented three "votes" that the CNS uses to determine the orientation of the subject. They could conflict, or they could reinforce one another by being "coherent" in a sensory manner. The CNS -- the vote-counter -- might even have weighed one vote more heavily than another if it felt the other senses were providing unusual, strange, or incomplete data. This theory was consistent with astronauts' increased reliance on visual cues in the absence of strong gravireceptic cues. A visual-down cue was generally stronger in the presence of greater scene polarity, such as in the FR. In this experiment the idiotropic-down cue was relatively constant. Subjects could not see the rest of their body while wearing the HMD, but the screens rectangular shape can provide a relatively weaker replacement for the head-to-feet vector. The idiotropic vote might have been weaker because the CNS was not accustomed to the absence of its body in the visual field.

As shown in the diagram, the visual vector, which represented the direction of the polarity generated by the virtual room, rotated with the room. The visual vector might or might not revolve 360°, depending on the scene, subject and her past experience. For the SR, for example, the visual vector might rotate 180° and then return to the 0° orientation since the room was visually symmetric about 180°. For the DR, the visual vector might rotate only up to 90° because of its four-way symmetry. The subject may, however, keep track of the original floor; in that case, the visual vector would revolve completely around 360°. The FR has a visual vector oriented strongly toward the floor, but when this angle gets beyond 90°, in some subjects, the visual vector might flip over and

point toward the ceiling, albeit more weakly. This is because, when viewing the FR when it is inverted, subjects may see the FR ceiling as a floor.

The visual vector rolls around a horizontal axis when the subject was erect and a vertical axis when the subject was supine. The idiotropic vector was along the body axis, so it was vertical when the subject was erect and horizontal when the subject was supine. Gravity remains fixed in the vertical direction. In the erect posture, for example, the subject's vestibular (gravitational) input was aligned with the visual input only at certain intervals; in the FR, at 0°; in the SR at 0° and 180°; and in the DR it 0°, 90°, 180° and 270".

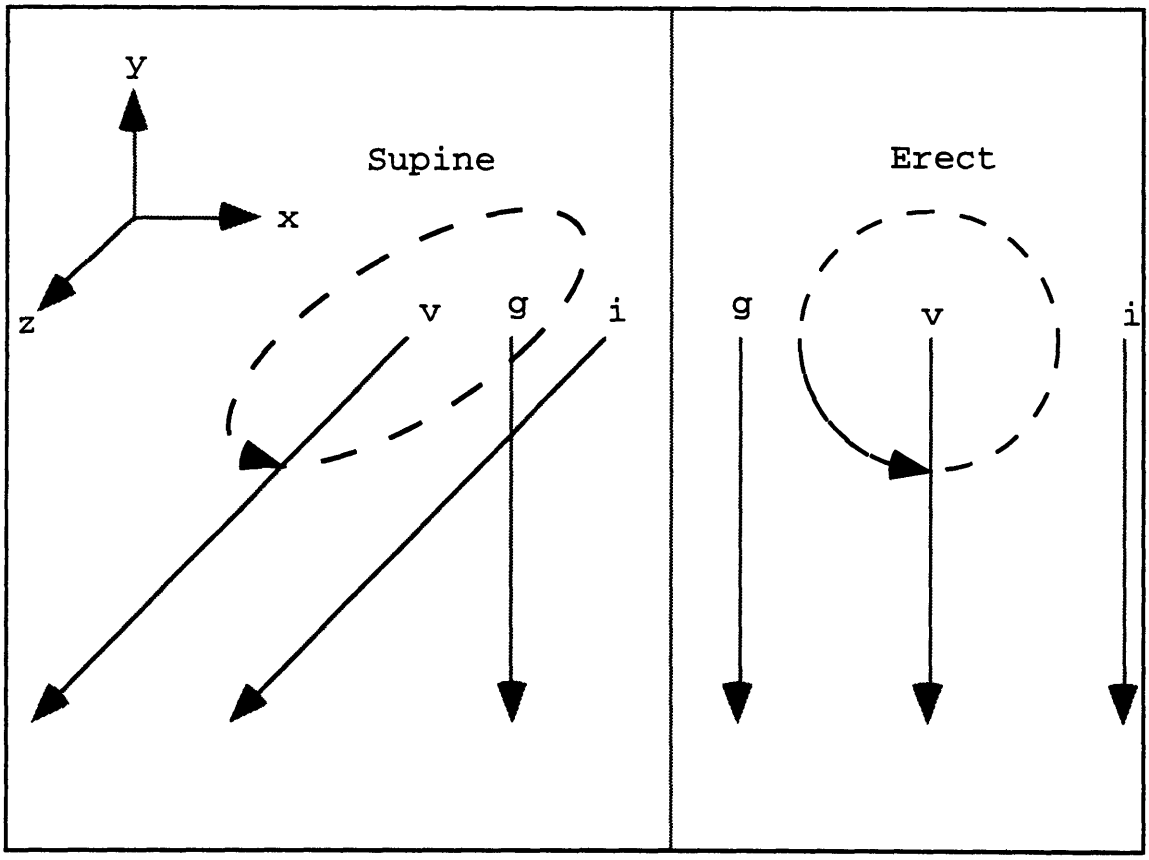

Figure 4-4: Visual, Gravitational, Idiotropic Vectors

A scene's polarity was measured by the number and potency of the mono-oriented objects in it. An object's potency varied with its size and ambiguity. The strength

of vestibular-visual alignments (coherence) in a room should be weaker if it has less polarity. By definition, a room that has more ambiguous orientations (such as the DR) has weaker polarity. Using the "voting" analogy presented previously, one could have said "the visual vote from a less polarized room was hypothesized to be weighted less than a visual-vote from a highly polarized room." The FR's single alignment, therefore, was more compelling than any alignment in the SR or DR. Nonetheless, viewing a rotating scene from an erect posture usually caused sensory conflict mixed with moments of coherence.

A supine subject--oriented perpendicular to the gravitational field-- had a different kind of visual-vestibular interaction. There were no periods of acute sensory coherence (when all three sensory votes coincided), and the sensory conflict was usually not as strong since gravity did not have any components in the plane of the visual polarity and idiotropic vector. With the symmetric scenes (DR and SR), where the visual polarity was not as strong, it was even hard to convince the subject that she was not horizontal with respect to gravity. The gravitational vector perceived by the erect-oriented vestibular system, however, actively conflicted with the rotating scenes visual polarity, except when the two were precisely aligned. To one in the supine posture, however, the physical gravity vector was along the line of sight. Since the rotation axis of the virtual scene was then parallel to gravity, the vestibular input did not actively impose its own polarity onto the visual scene.

The above discussion assumed that the subject was not imposing a strong idiotropic "down", for her doing that would have explained similar behavior in both the erect and supine postures. To an erect subject, gravity imposed a "down" that corresponds with the idiotropic vector. When supine, however, the subjects may have imposed their idiotropic down on their visual input. The rotation axis of the virtual scene was always perpendicular to the idiotropic vector, therefore the idiotropic vector corroborates the visual

polarity at least once for the FR (twice for the SR and four times for the DR). These moments of coherence might have caused a supine subject to feel instantaneously vertical instead of supine, and could explain why subjects might feel reorientations equally as often in both the erect and supine postures.

The conflicts and coherence may also have varied from room to room. In the SR, for example, a subject may feel she was looking up at a skylight, while lying down on a rotating platform. If the subject was supine while experiencing this, she would have had near continuous sensory coherence, since all her senses were confirming this. The illusion of lying down on a rotating platform, a horizontal-plane rotation, was also possible with the DR, which had no strong mono-oriented objects. Since this scenario has not commonly experienced by the CNS, it was more readily open to reinterpretation. Yet, with the FR, the subject was receiving strong visual "down" cues, making it difficult to feel a horizontal-plane rotation. Thus, various combinations of scene and posture provided a potential exploration of the way our CNS resolves sensory conflict.

5. Results

In this chapter, vection, tilt, and reorientation were in reported in separate sections. In the last section learning effects among subjects were examined. Most of the analyses were based on the general linear model (GLM) as realized in Systat 5.21 software on a 601-based Power Macintosh. Data was presented as fitted least-square means (LSM) unless otherwise noted, and "p" levels were Bonferroni-adjusted t-tested means-comparisons marked as "p*" to distinguish them from uncorrected values, "p"'s. All F-ratios were calculated from analyses of variance (ANOVAs) tabulated in Appendix B. When the Wilcoxon Signed Ranks Test (WSRT) was used, all subjects were included. This may not be apparent from the description of the results, since the WSRT by definition ignores subjects which do not exhibit any effect.

5.1 Vection

This section describes results related to the sensation of vection. The expanded category system allowed subjects to report two types of full-tumbling: full-tumbling horizontally and vertically with respect to gravity. Thus the full-tumbling analysis was broken down into to parts. Scene did not have the expected significant effect on full-tumbling. The interactions between scene and posture were more complicated than previously thought. On the other hand, scene -- but not posture -- significantly affected the reported latency of vection. Finally, scene content contributed significantly to increased saturation, while posture did not.

5.1.1 Full-Tumbling

When an erect subject experienced the illusion of "vertical full tumbling" (VFT), her visual input was dominating her vestibular cues. She believed she was rolling through 360° about a horizontal axis (i.e., gravitationally vertical plane) despite the vestibular report that she was not rotating at all. Unfortunately, the new categorization system provided subjects no direct way to report the sensation of full-tumbling. Instead, when subjects reported vection with neither constant-tilt nor alternating-tilt, it was inferred that they were feeling a continuous rolling without any angular limits, i.e., full-tumbling. In presenting the full-tumbling results, a distinction was made between horizontal-frontal-plane full-tumbling (HFT) -- the sensation of full 360° tumbling in a gravitationally horizontal plane -- and VFT. This was because, while VFT illusions indicated a dominance of the visual input (i.e., visual polarity over otolith stimulus), HFT illusions represented the HCNS weighting of the vestibular over visual input.

In DeSouza's experiment, 9/12 (75%) of the subjects reported feeling VFT in the furnished room when they were erect, but only 6/16 (33%) of them felt it when supine.[1995] In the present experiment, but not in DeSouza's, conditions were repeated. One way to compare the present full-tumbling results with DeSouza's is to look at the number of subjects in this experiment who felt VFT at least once for the FR. In the present experiment, 9/16 (56%) felt VFT at least once in the erect position, whereas (38%) did while supine. From this viewpoint, DeSouza's results were not that different from that of the current experiment. Based DeSouza's single trial experiment, however, one might hypothesize that over repeated trials, a subject will feel VFT while viewing the FR more often when erect than supine. The results of the current experiment, however, which compared each subject's response in the two different orientations, were not consistent with

this interpretation. As will be shown, while a given subject viewed the FR they did not feel VFT more often when erect than supine.

In the present experiment, subjects felt full-tumbling more often when supine than when erect. Figure 5-1 and Figure 5-2 summarize the subjects' frequency of VFT illusions. Three subjects (19%) indicated that they did not feel full-tumbling in any of their 48 runs, hence they were omitted from these charts. The charts report the number of subjects indicating full-tumbling a given number of times. Taller columns on the left mean fewer reports of full-tumbling for a given set of conditions. Taller columns on the right indicate more frequent reports of full-tumbling. Since there were 8 runs for each of the six scene and posture combinations, the subject could report up to 8 instances of full-tumbling for a given combination.

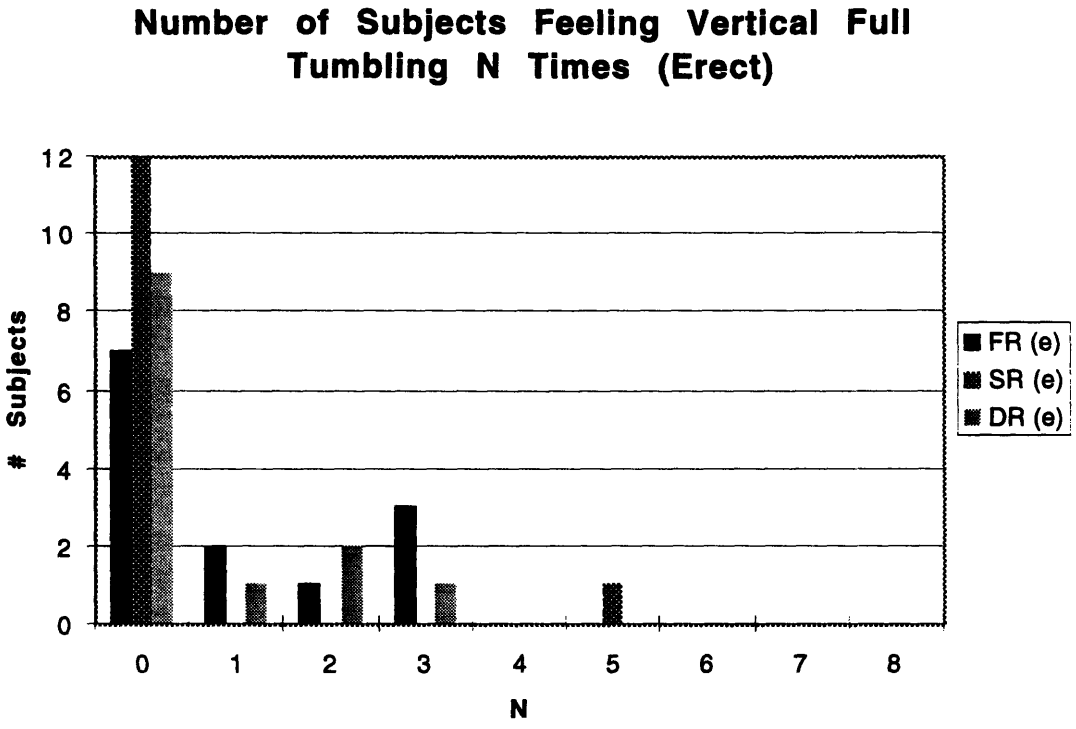

Figure 5-1: Number of Subjects Feeling VFT "N" Times (Erect).

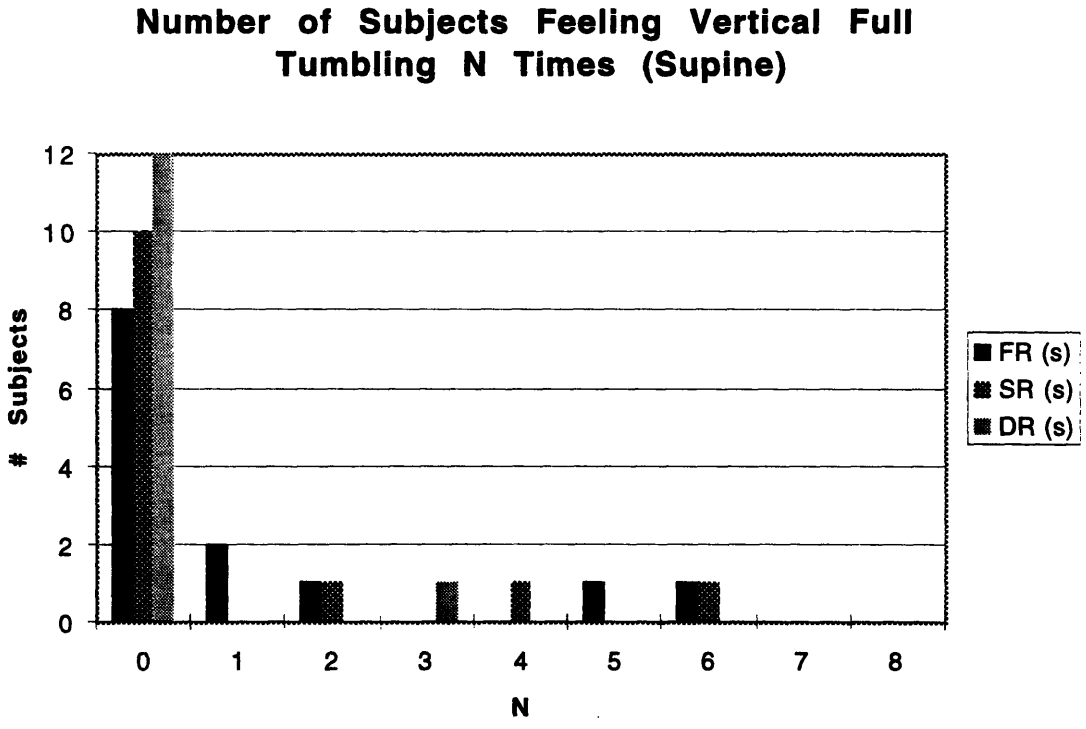

Figure 5-2: Number of Subjects Feeling VFT "N" Times (Supine).

When viewing the DR, subjects had a tendency to feel VFT more often when erect than supine. In the DR, 9 subjects felt VFT more often (only 2 less often) when erect (the others reported VFT equally often in both postures). By the Wilcoxon Signed Ranked Test, this was significant (WSRT $z = -2.28$; $p = 0.0221$). When subjects were viewing the FR and SR, their posture did not have a significant effect on their frequencies of VFT. For the FR, only 5 felt VFT more often (3 less often) when erect (WSRT $z = -0.422$; $p = 0.673$). For the SR, only 5 felt VFT more often (4 less often) when erect (WSRT $z = 0.357$; $p = 0.729$). Note, the three subjects who did not feel full-tumbling in any run are conveniently ignored by the WSRT. The FR result differed from DeSouza's. His subjects felt VFT in the FR more often when erect than supine.

The effect of posture much more dramatic on subjects viewing the DR, a

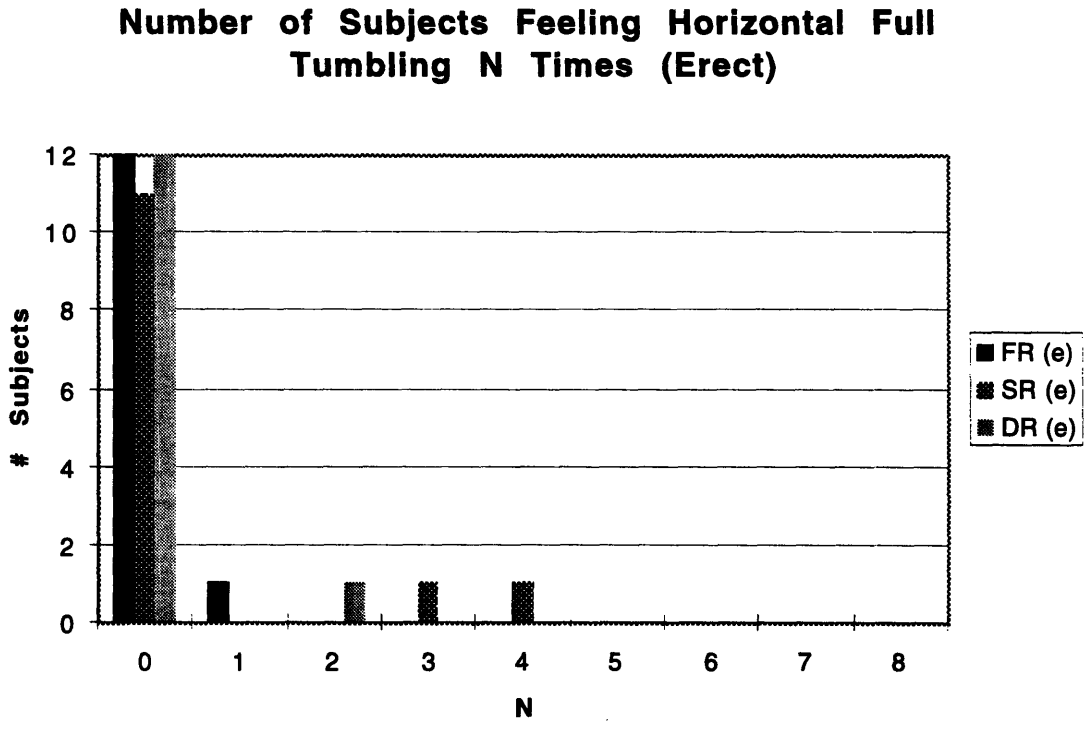

Figure 5-3: Number of Subjects Feeling HFT "N" Times (Erect).

stimuli that was devoid of polarity cues. This would be consistent with the hypothesis that subjects use mono-oriented objects as reference points to determine their own orientation. For the supine posture, 6 subjects felt VFT more often (1 less often) in the FR than DR (WSRT $z = -1.98$; $p = 0.048$). The DR did not have any visual elements that provided an "upright" cue to the subject, thus the subject became more dependent on her vestibular cues in determining her orientation. The FR, however, provided polarity directed perpendicular to the subjects' visual axis induced, hence it induced (more often than did the DR) supine subjects to feel as though their frontal-planes were vertical with respect to gravity. The FR polarity, therefore, would lessen the effect of posture on the frequency of VFT illusions. A similar argument can be made for the SR, which did not have mono-oriented objects, but did have windows, floors, and ceilings, which provided a relatively weaker, but still present, vertical cue. Thus the residual polarity in the SR masked the effect of posture on the subjects' frequency of reporting VFP illusions in that scene.

Subjects exhibited a different trend for HFT illusions. As depicted in Figure 5-3 and Figure 5-4, they reported HFT illusions less often in the erect than supine posture. This trend was highly significant for each room. For the FR, 10 subjects felt HFT more often, none less often (the rest as often), in the supine than erect posture (WSRT $z = 2.82$; $p = 0.005$); for the SR, 9 more, 1 less (WSRT $z = 2.40$; $p = 0.016$); for the DR, 11 more, 1 less (WSRT $z = 2.99$; $p = 0.0028$). This trend was expected since, for subjects in the erect posture, the combination of "vertical" otolith stimulus and visual polarity (in the case of the FR and SR) made it difficult for the subjects to feel horizontal, much less HFT. If the subject were to feel HFT when erect, she would have to discount her vestibular sense, but when supine, she would not. When the subject was supine, it was much easier for her to believe that she was feeling HFT since it would *nearly* resolve all sensory conflict. If the subject was feeling HFT when viewing the FR in the supine posture, it meant she was

discounting the visual polarity that suggested here frontal-plane was gravitationally vertical. In effect, she would believe that the FR was not a real room, but the image of one. The subject might also be discounting the reality of the SR, but another possibility was that she reinterpreted the surfaces to be something else. A few subjects mentioned that, when viewing the SR in the supine posture, it seemed like they were spinning with a platform on the floor of a room with a skylight. In this case, conflict was completely resolved. These subjects reported similar illusions while viewing the DR although less often than for the SR. Hence, HFT more often VFT illusions represented the resolving of conflict.

In their experiments, Howard and Childerson [1994] found an increase in subjects reporting VFT in passing from their dotted-room to their furnished room. Surprisingly, however, the scene content had, in this experiment, no significant effect on the frequency of reports of VFT. The number of subjects in the erect posture reporting

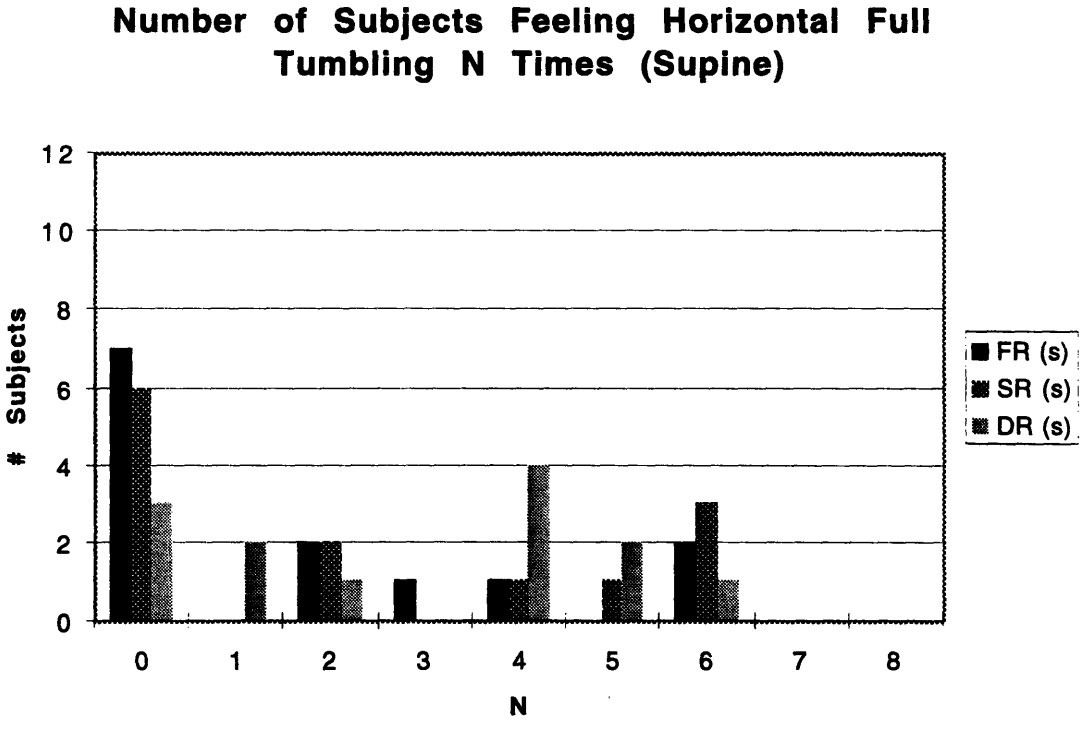

Figure 5-4: Number of Subjects Feeling HFT "N" Times (Supine).

VFT did not significantly increase or decrease from the FR to DR: 4 subjects felt VFT more often (3 less often) in the FR (WSRT $z=-0.34$; $p = 0.732$). In the supine posture, however, subjects reporting VFT did significantly increase from the DR to FR: 6 subjects felt VFT illusions more often (only 1 less often) in the FR than DR (WSRT $z = -1.98$; $p=0.048$). Again, the polarity of the FR overcame the vestibular conflict in many subjects. Between the FR and SR, however there was not a significant difference: 3 felt VFT more often (1 less often) in the FR (WSRT $z = -1.3$; $p = 0.194$). Thus, this experiment neither confirms nor disagrees with Howard and Childerson's results.

There were several possible explanations for the small effect of scene on the frequency of full-tumbling illusions. The limited FOV, for example, might have reduced the impact of the visual stimuli. Previous multiple-scene studies of full-tumbling used real rotating rooms with unconstrained field-of-view (FOV) [Kleint 1937], [Howard and Childerson 1994]. In other research, the percentage of subjects reporting full-tumbling increased 25% going from 90° FOV to an unconstrained FOV. Thus, the 90° FOV used in the present experiment could have reduced the effectiveness of the visual stimuli, causing the subjects to feel constant or alternating tilt rather than full-tumbling.

One might be tempted to somehow attribute this result to the multiple trials used in the present experiment, since single trials had been used for each scene in previous research. Howard and Childerson [1994] used only one repetition per subject for each scene. To test whether subjects were more sensitive to scene content in earlier runs, the subjects' first and last runs for the FR and DR in the erect posture were compared. According to Table 3.1, these were run #1 and #3 on the first day and, #17 and #23 on the second day. The contingency table for these runs (shown in Table 5.2), however, did not indicate any significant learning effects for subjects. Thus the result cannot be explained by the increase in number of trials.

Table 5.2: Full-Tumbling in the First and Last Runs of the FR and DR with Subject in the Erect Posture ($\chi^2_{(3)} = 0.881$; p=0.83).

Full Tumbling	First FR	Last FR	First DR	Last DR
No	11	11	12	14
Yes	5	5	4	2

Since, in this experiment, subjects indicated full-tumbling in ways that differed from previous research, that may have caused the a different result. Subjects were not given a direct method to specify full-tumbling. Instead, they decided whether or not they felt limited tilt (i.e., constant tilt or alternating tilt). That would have indicated that they did not feel full-tumbling. If, on the contrary, subjects reported no limiting tilt, accompanied by a sensation of vection, that was taken to be a report of full-tumbling. This difference might have made full-tumbling less sensitive to scene content, and more sensitive to the same variables that effected tilt. Training subjects to be aware of and report reorientation illusions may have made them more sensitive to the sensation of tilt, hence they might have reported tilt more often than subjects who would not have been trained to report on reorientation illusions. This could have greatly reduced the frequency of full-tumbling in subjects and made it difficult to see any significant effect of scene and posture on it.

To conclude the analysis, all instances of HFT and VFT illusions were combined into one "overall" full-tumbling illusion. In the furnished room, 12 subjects felt full tumbling more often (only 1, less often) in the supine than erect posture (WSRT z=2.99; p=0.0028). Similar results were observed for each scene: in the symmetric room, 12 of 12 felt full tumbling more often in the supine posture (WSRT z=3.08; p=0.0021); in the dotted room 12 of 13 (WSRT z=3.02; p=0.0025). Overall, there were nearly twice as

many reports of full-tumbling in the supine as in the erect posture. This followed a trend noticed by Young[1975] that subjects' visually induced self motion increased as their heads were tilted away from the vertical axis.

5.1.2 Latency of Vection

The apparent rolling of objects, that are usually stationary, provokes vection. A scene with more mono-oriented objects should therefore have provided more opportunities for inducing vection, and thus produce shorter average latencies. As shown in Table 5.3, the scene's content did have a significant effect on latency ($F(2,351)=149.3$ $p=0.0003$). Subjects had shorter latencies in the FR ($p^* = 0.0004$) and SR ($p^* = 0.0059$) than they did in the DR. Subjects had slightly shorter latencies for the FR than the SR, but this was not significant ($p^* = 1.0$). Supine latencies were slightly--but not significantly ($p^* = 0.59$) -- longer than erect.

Table 5.3 Fitted LSM Vection Latency Averaged Across Subjects

Posture	Furnished Room	Symmetric Room	Dotted Room	Combined
Erect	7.12 s	7.36 s	9.15 s	7.88 s
Supine	7.24 s	7.76 s	9.33 s	8.11 s
Combined	7.18 s	7.56 s	9.24 s	7.99 s

The weak effect of posture on vection latencies suggested that latency was driven by scene content, rather than the subject's posture. This result suggested, but did not conclusively prove, that the presence of mono-oriented objects induces vection sooner. Subject's CNS may have been discounting its vestibular information sooner because it conflicted with strong visual cues that advertised self-motion. Returning to the voting

analogy, one could say that, when viewing the strong visual stimuli, the CNS did not lend credence to the vestibular system's input.

It was surprising that the subjects' latencies for the DR were significantly longer than those for the SR, even though the SR did not have any mono-oriented objects. One subject noted that her vection was strongest when she was looking at the tiled surfaces of the floor and ceiling in the SR (and the ceiling in the FR). This suggested that, in addition to polarity, the CNS used frame and spatial frequency of the visual stimulus to determine when they began self-motion. That the FR latency was not much shorter than the SR latency suggested that scene polarity might not have effected latency as much as the other scene characteristics. The latency relationship could be written as

Equation 5-1 $$T_{latency} = T_0 - P(\dot{\theta}) - F(\dot{\theta}) - f(\dot{\theta})$$

T_0 is a maximum latency corresponding to the weakest possible visual stimulus. P, F, and f, are respectively the polarity, frame and spatial-frequency effects on latency. Since roll velocity might effect latency, it was included as a parameter in Equation 5-1.

There was significant amount of latency variation between subjects ($F(15,351) = 14.05$; $p<0.0001$). Figure 5-5 plots the subjects' mean latencies with error bars (using the standard deviation of the mean). The large variation between subjects suggested that each subject had a characteristic P, F, f, and T_0. Future research should explore this possible relationship.

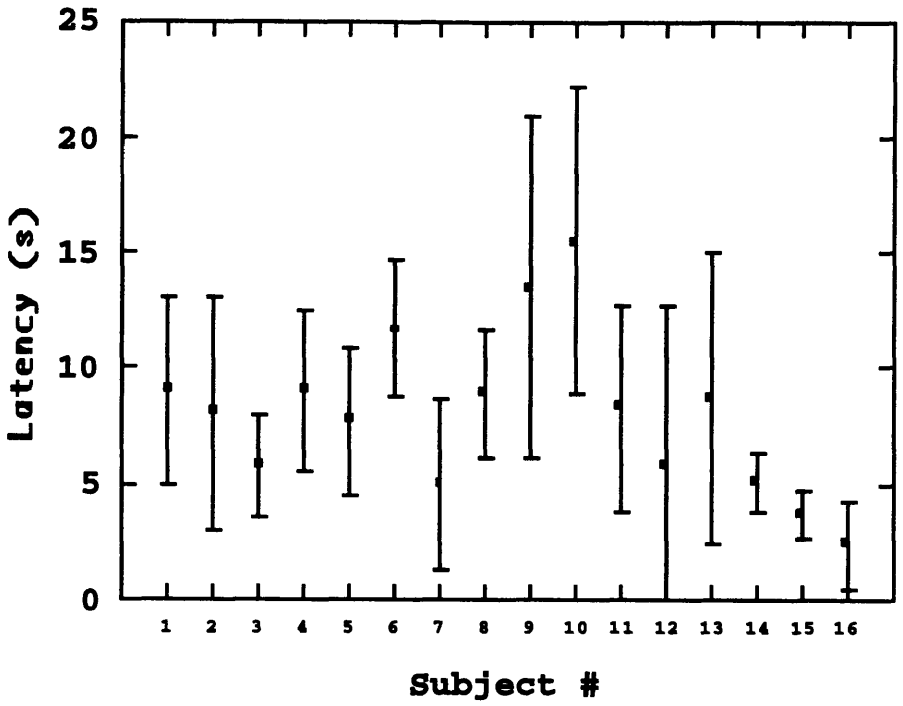

Figure 5-5: Mean Latencies for Each Subject with Standard Error Bars ($\sigma_{\mu i}$) of the Mean.

5.1.3 Saturation of Vection

Saturation was defined as the percentage of the scene's velocity of roll that was perceived as self-motion. The average saturation was expected to increase with the number of mono-oriented objects in the scene, i.e., the average vection saturation should decrease from FR to SR to DR ($SAT_{FR} > SAT_{SR} > SAT_{DR}$). This followed the assumption that the apparent rolling of (usually stationary) mono-oriented objects, helped convince the subjects that they, and not the scene, were moving.

The results (Table 5.4) supported this theory. Scene had a significant effect on saturation ($F(2,746)=35.8$; $p<0.0001$). The FR induced significantly higher average

saturation than those of both the SR ($p^* = 0.0093$) and DR ($p^*<0.0001$). The SR also induced higher saturation than did the DR ($p^*<0.0001$).

Table 5.4: Reported Vection Saturation (%) by Room and Posture

	Furnished Room	Symmetric Room	Dotted Room	Combined
Erect	45.3%	42.7%	34.5%	40.1%
Supine	49.1%	43.1%	35.6%	42.6%
Overall	47.2%	42.9%	35.1%	41.7%

While it was expected subjects would report higher saturation in the supine than in the erect position, posture was not a significant main effect ($F(1,746)=2.22$; $p=0.1365$). When exposed to contradictory sensory information, subjects may (as suggested) have given greater weight to visual cues than to their haptic and vestibular cues. This would increase the effect of the scenes' visual content, and reduce the effect of the subjects' posture.

The idiotropic down may also have reduced the effect of posture on saturation. While supine, the subjects might have been weighting their idiotropic vector more than their visual and vestibular cues. Thus the idiotropic cue might have taken the place of gravity, and maintained a similar level of conflict that was present in the erect subject. Since full-tumbling *was* effected by posture, it must have been more dependent than vection saturation on actual gravireceptor cues (as opposed to idiotropic cues).

During the vection blocks, subjects are concentrating more on it than on reorientation, and may be inducing stronger vection, and therefore, higher reported saturation. Indeed, (Table 5.5) block had a significant, although not very dramatic, effect on saturation ($F(1,746)=8.5$; $p=0.0037$).

Table 5.5: Vection Saturation by Block

V-Block	R-Block
44.2%	39.3%

As with full-tumbling, variation between subjects was large ($F(15,746) = 41.33$; $p<0.0001$). Figure 5-6 plotted the subjects' mean latencies with error bars (using the standard deviation of the mean). This plot suggested that subjects had characteristic saturation functions, just as they did for latency. Although subjects may have varied significantly in their mean saturation, they still followed the trends related to scene and posture mentioned earlier.

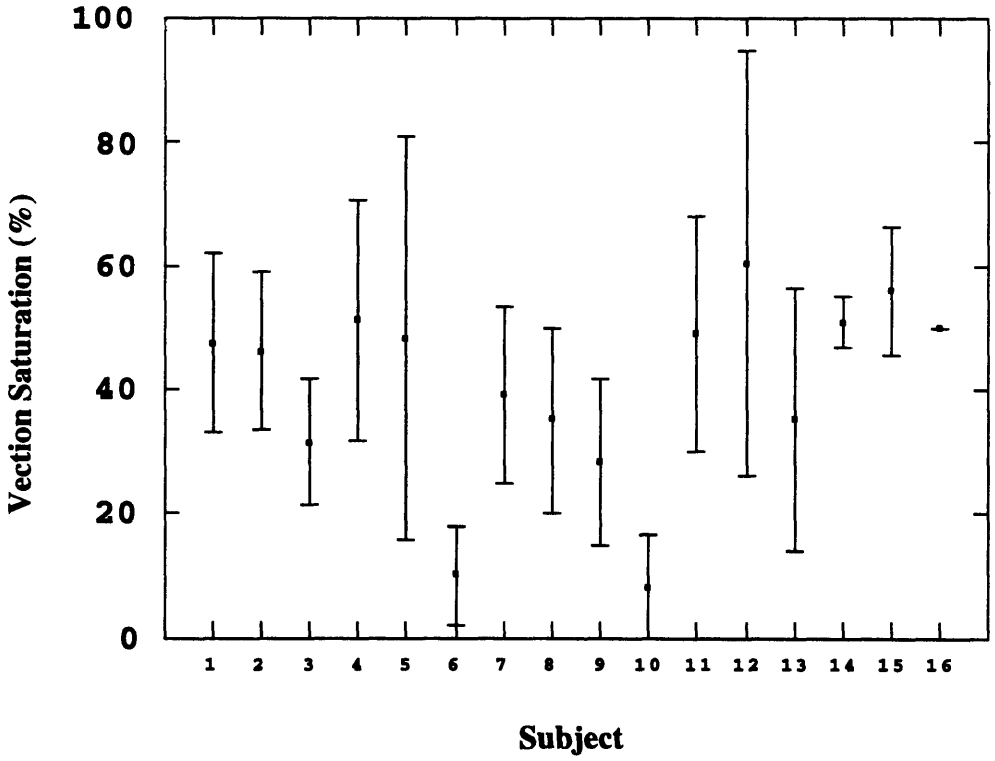

Figure 5-6: Mean Saturation for Each Subject with Standard Error Bars ($\sigma_{\mu i}$) of the Mean.

5.2 Reorientations

Reorientation angles were specified by citing the angle to the horizontal made by the relevant surface of the scene (i.e., the surface that the reorientation was assigned to using algorithm described in section 4.2.2). Each scene had 4 surfaces because they shared a common frame. The surfaces are numbered 0 to 3, with (0,1,2,3) representing (original "floor", first wall, original ceiling, second wall to appear under foot)[2] as in Figure 5-7.

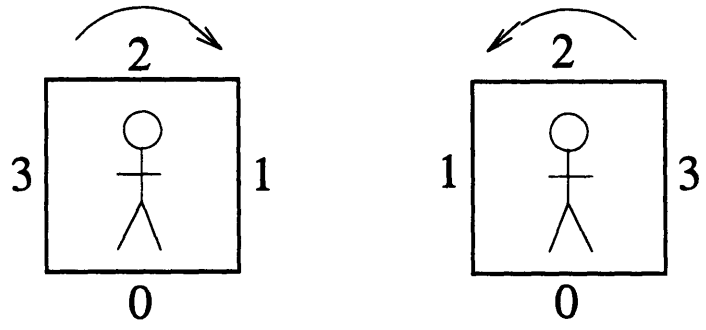

Figure 5-7: Coding of Surfaces in a Room Depends on Rotation Direction

A reorientation angle of 0° indicated that it occurred when the relevant surface was exactly horizontal and beneath the subjects feet. Negative angles indicated a reorientation before (positive angles after) the surface reached this position (Figure 5-8).

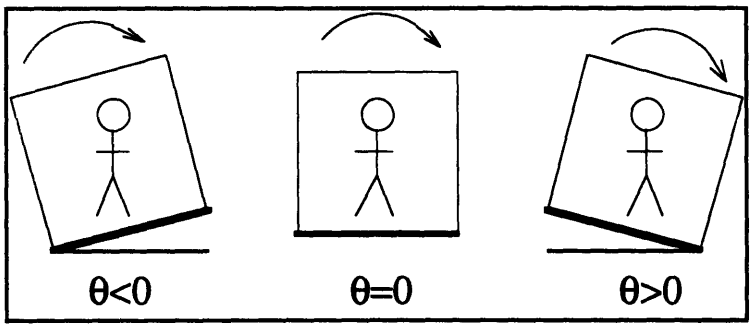

Figure 5-8: Reorientation Angle (θ) and Surface of Reorientation (darker line)

---

[2] Note that for the DR surface 0 = 1 = 2 = 3, for the SR 0 = 2, and 1=3, but for the FR, 0≠1≠2≠3. For the SR and DR, both rotation directions provide identical stimuli, but for the FR, CCW provides a different stimuli than CW, since for CW, the left wall is surface 3, but for CCW the left wall is surface 0.

5.2.1 Polarization effects: Early and On-Time Classification

Since reorientations were expected to occur earlier for highly polarized orientations of the scene (Section 4.2.2), mean reorientation angle was expected to be more negative for these surfaces. Specifically, mean reorientation angles were expected to be more negative when the FR's floor and the SR's floor and ceiling were subjective floors (reference surfaces). Only subjects who had very negative mean reorientation angles when the FR floor was a reference surface exhibited this expected behavior. The others, while not showing significant sensitivity of reorientation *angle* to the orientation of the scene, did exhibit a significant sensitivity of reorientation *frequency* to the orientation of the scene.

Two patterns of behavior in the reporting of reorientations were recognized. Subjects were classified into two subgroups, based on their average reorientation angle for the floor of the FR (Figure 5-9). The FR floor was chosen because all subjects had at least 5 reorientations for it, and also because it theoretically corresponded to the most polarized scene and orientation in the experiment. Subjects were classified as (early, on-time) if their mean angle of reorientation for the FR floor was (less than, greater than) -22.5°. This angle was chosen because the Figure 5-9 seemed to suggest two distinct response groups, and -22.5° was the value which best split the groups. Subject 1, for example, had 14 reorientations for the FR floor, averaging 8.13°, and was classified as on-time. Subject 3 had 24 reorientations for the FR floor, averaging -32.7°, and was classified as early. Nine subjects were classified as early and seven as on-time. Even though the subjects' mean reorientation angles for the FR floor fell clearly on one side of -22.5°, their standard errors of the means may have overlapped. In the ensuing discussion, subjects are called early or on-time based the criteria explained above. Subjects who have been classified as early might have exhibited on-time behavior for surfaces other than the FR floor (e.g., the FR ceiling.

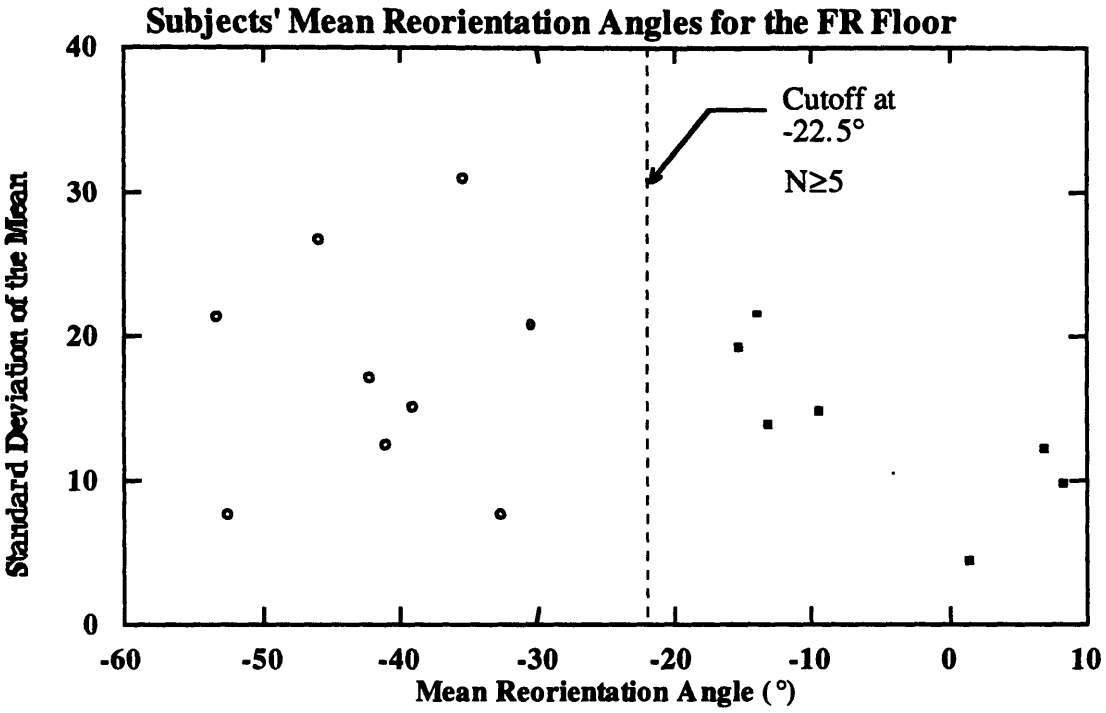

Figure 5-9: Determination of Early or On-Time Behavior.

Figure 5-10 through Figure 5-12 reveal that a subject's early or on-time behavior was consistent for the floors and ceilings of the FR and SR. The figures show that early subjects -- subjects who reoriented early (i.e. mean less than -22.5°) for the FR floor -- also reoriented early for the FR ceiling. These subjects reoriented early for the floor and ceiling of the SR as well. Despite this, early subjects tended to exhibit mixed early and on-time behavior for the walls of the FR and SR. For the walls, some subjects continued to reorient early, but others on-time. In some cases a subject had only one or no reorientation for a given surface of a room, and was excluded from the corresponding plot. The two reorientation patterns are summarized in Table 5.6 and Table 5.7.

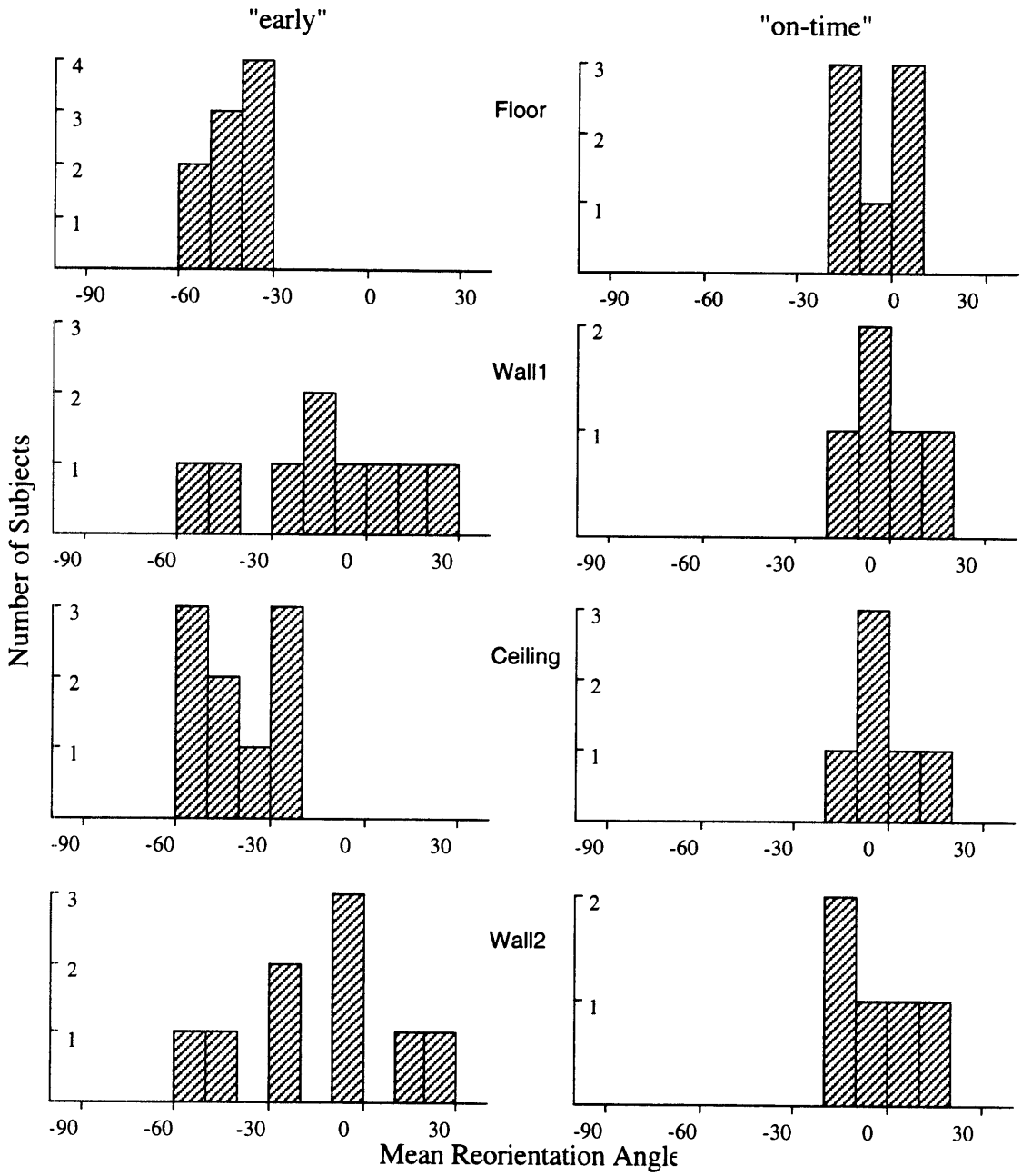

Figure 5-10: Furnished Room: Early and On-Time Mean Reorientation Angles.

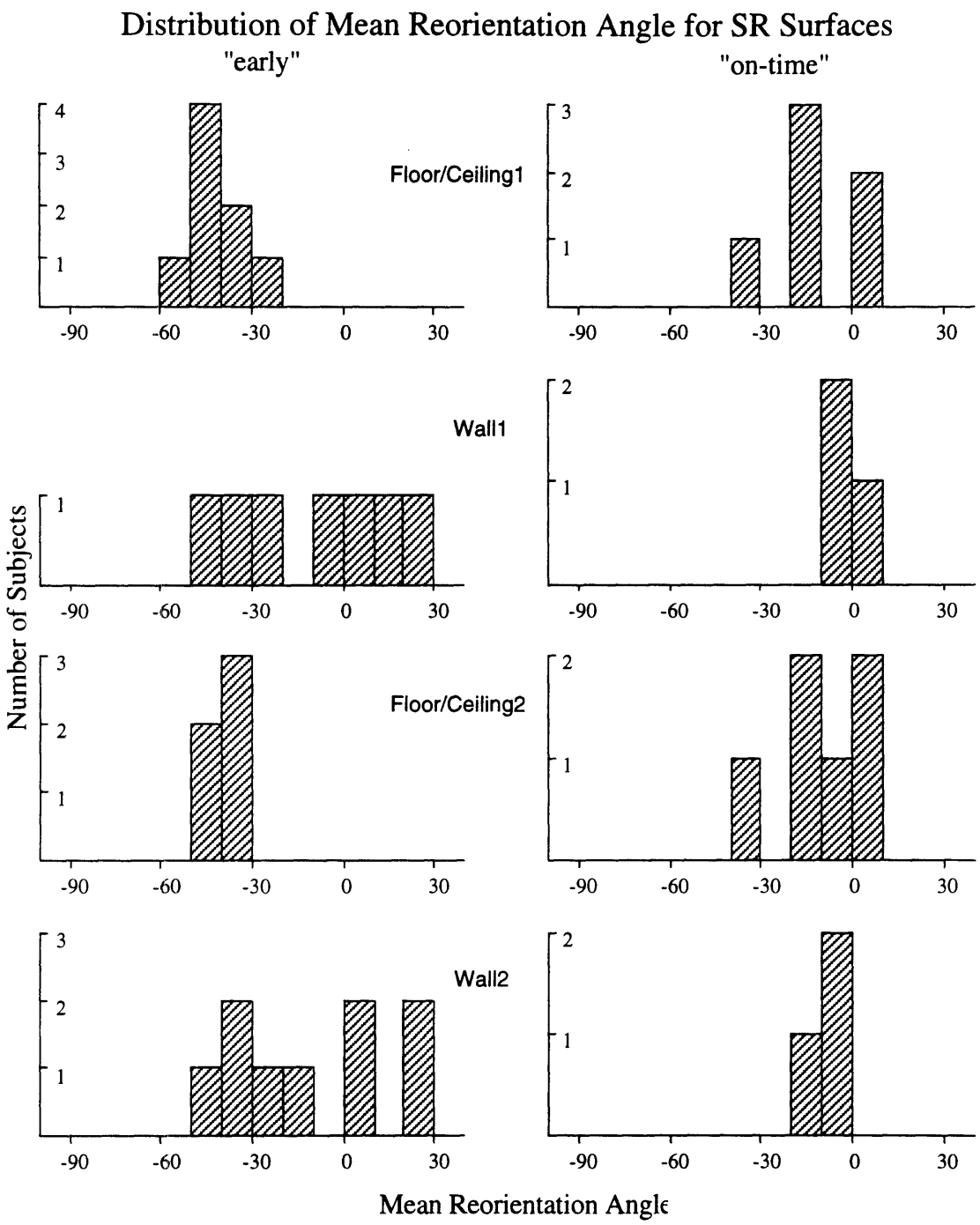

Figure 5-11: Symmetric Room: Early and On-Time Mean Reorientation Angles.

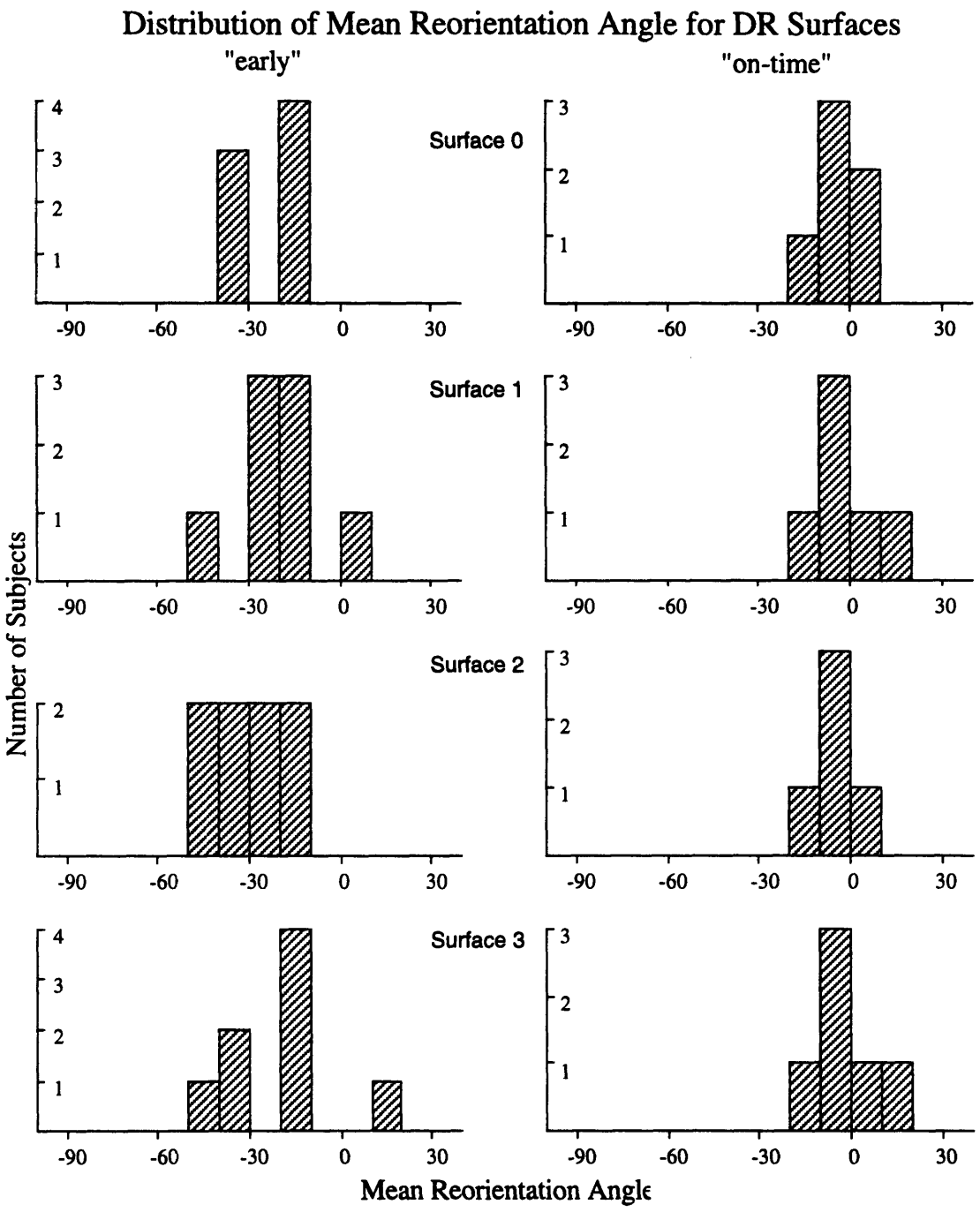

Figure 5-12: Dotted Room: Early and On-Time Mean Reorientation Angles.

Table 5.6: Patterns Within Early and On-Time Reorientation Reporting.

Scene	Early (A_i)	On-Time (B_i)
Furnished Room	$A_0 \approx A_2 \leq A_1 \approx A_3$	$B_0 \approx B_2 \approx B_1 \approx B_3$
Symmetric Room	$A_0 \approx A_2 \leq A_1 \approx A_3$	$B_0 \approx B_2 \approx B_1 \approx B_3$
Dotted Room	$A_0 \approx A_2 \approx A_1 \approx A_3$	$B_0 \approx B_2 \approx B_1 \approx B_3$
Number of Subjects	9	7

Table 5.7: Patterns Between Early and On-Time Reorientation Reporting (A_i,B_i).

Furnished Room	$A_0 \approx A_2 > B_0 \approx B_2$; $A_1 \approx A_3 \geq B_1 \approx B_3$
Symmetric Room	$A_0 \approx A_2 > B_0 \approx B_2$; $A_1 \approx A_3 \geq B_1 \approx B_3$
Dotted Room	$A_0 \approx A_2 \geq B_0 \approx B_2$; $A_1 \approx A_3 \geq B_1 \approx B_3$

Although early subjects reoriented earlier in the FR and SR when the subjective floor (reference surface) was either a floor or ceiling than when it was a wall, they reoriented at similar angles for all four DR surfaces. The SR and DR result was to be expected, but it was odd that early subjects also reoriented early when the FR ceiling was a subjective floor. The dotted room did not have one particular orientation which seemed more believably upright than the others. Two orientations of the SR were more believably upright than the others (when the identical floor and ceiling were below and above the center of the visual field), and it was these scene-orientations that the "early" subjects reoriented earlier for. Yet the FR arguably had only one "upright" orientation due to the large number of aligned mono-oriented objects in it. Despite this, early subjects reoriented almost as early when the subjective floor was the ceiling than when it was the floor. Early subjects may have become used to believing that the FR ceiling was a floor because the SR

floors were identical to it, hence they would reorient just as early when the FR was inverted than when it was upright in the visual field.

It was also possible that these subjects may have used the "walls" to reference their orientations from. A "competing-walls" model , where the two pairs of parallel surfaces compete to "become" walls, could explain the observed behavior of the early-subjects. As these subjects observed the scenes' real walls approaching their appropriate positions (on the side of the visual frame), they would reorient sooner than when the floors and ceilings were approaching the same positions. This was because the walls of the FR and SR were more "believably" walls. Note that each surface in the DR was identical, hence the competing-walls model would predict that subjects would reorient at identical angles from all reference surfaces. Yet it was still possible that reorientations are "floor" driven. A "competing-floor" model, where the two surfaces lowest in the visual frame compete to "become" a floor could explain the observed behavior of the early subjects. The competing-floor model would clearly predict that the real floor would induce reorientations sooner than a wall as it moved toward the subjects feet. Also, since they may have felt the wall was disturbing as a subjective floor, early subjects may have tried to reorient as soon as possible after the wall moved out from under the their feet. This could explain why the inverted FR (with the ceiling as a subjective floor) still induced early reorientations from early-subjects. It should be noted that we cannot know whether the subjects in this experiment were attending to the floors or the walls, and so we cannot distinguish between a "competing floors" model and a "competing walls" model on the basis of the present experimental results.

A "competing-ceiling" model might explain why subjects would reorient sooner when the real ceiling approached its appropriate position, but it would not explain why this happened for the floor as well (as it did for the FR and SR). A "competing-

frame" model -- where all the surfaces compete simultaneously to become a floor-ceiling-walls set -- might have some credibility in explaining the observed behavior in early subjects, but it was more likely that subjects followed a competing-floor or competing-walls strategy (or maybe even a mixture of the two). Consider that many people, when sitting or walking upright, pay no attention the ceiling of a room, but instead, they regard the walls and floor. They watch floor sometimes because they have to watch where they are stepping. When they are not watching their step, they are probably looking straight, where the walls are in plain sight.

On-time subjects, however, did not exhibit much difference in mean reorientation angle between any particular upright orientation of a room. This implied that the early subjects were more actively seeking out particularly strongly polarized orientations of the scene to reorient to, whereas the on-time subjects were more passively reorienting, waiting until the scene was almost exactly "upright" before feeling reoriented.

Table 5.8 summarizes the range of mean reorientation angles for each reference surface for early and on-time subjects. It lumps the mean reorientations to walls of the FR and SR into one set, and to floors and ceilings of the FR and SR in another. Mean reorientations to surfaces of the DR are lumped into a third set since they were very similar. For the FR and SR, early subjects' mean reorientation angles for the walls had a much larger range than for the floors and ceilings. For on-time subjects, the mean reorientation angles for the FR and SR floors and ceilings were offset approximately 10° earlier than for the walls`. Individual subject means for all surfaces of each scene are plotted with their standard errors in Figure 5-13 and Figure 5-14 for early subjects, and in Figure 5-15 and Figure 5-16 for on-time subjects. Each point represents one subject's mean reorientation angle for a reference surface in the respective scene, with a different symbol used for each subject. In Figure 5-13, for example, there are four pentagons, each

representing one subject's mean reorientation angle when the reference surface was the FR or SR floor or ceiling.

Table 5.8: Range of Mean Reorientation Angle for Early and On-Time Subjects.

	FR and SR Walls		FR and SR Floors and Ceilings		All surfaces in DR	
# Subjects reorienting (who didn't) .	9 of 9 (36 means)		9 of 9 (36 means)		8 of 9 (32 means) (#12 did not reorient for any surface.)	
	Minimum	Maximum	Minimum	Maximum	Minimum	Maximum
Early	-55.4°	26.6°	-61.2°	-21.3°	-44.5°	-22.8°
# Subjects reorienting (who didn't)	6 of 7 (18 means) (#4 did not reorient for any walls. #1 did not reorient to SR walls. #2, #11 reoriented for one wall in SR and FR.)		7 of 7 (26 means) (#4 did not reorient to SR floor and FR ceiling.)		6 of 7 (24 means) (#4 did not reorient for any surface.)	
	Minimum	Maximum	Minimum	Maximum	Minimum	Maximum
On Time	-19.5°	23.2°	-32.7°	12.3°	-14.9°	13.0°

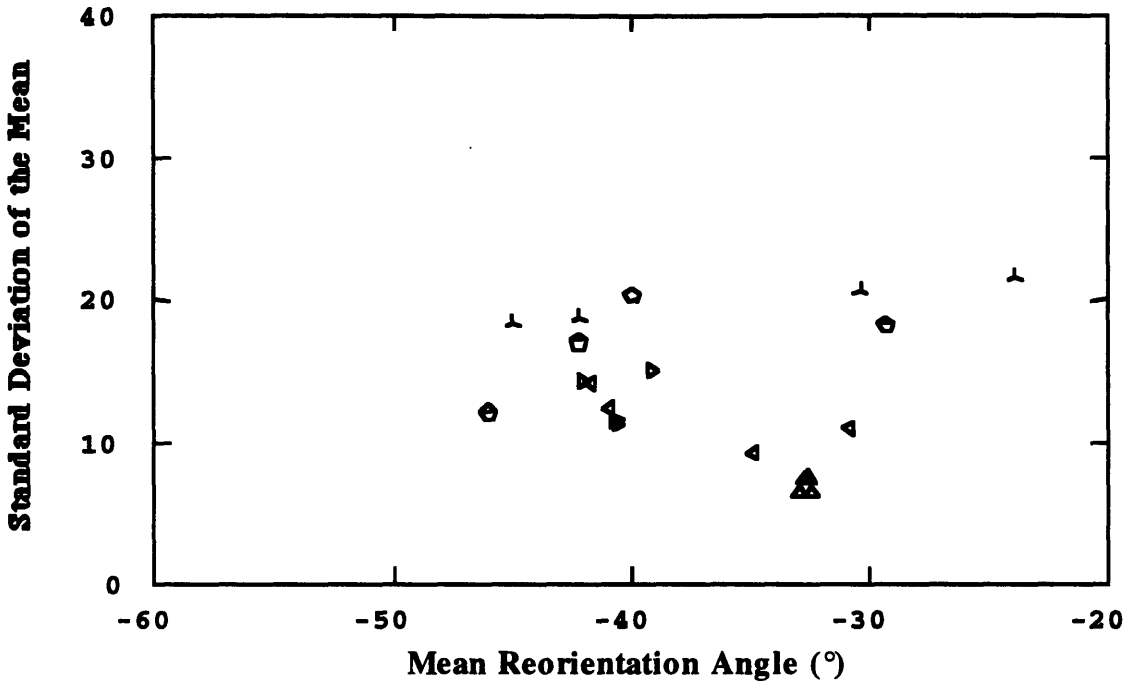

Figure 5-13: Early Subjects' Mean Reorientation Angles for the Floors and Ceilings of FR and SR.

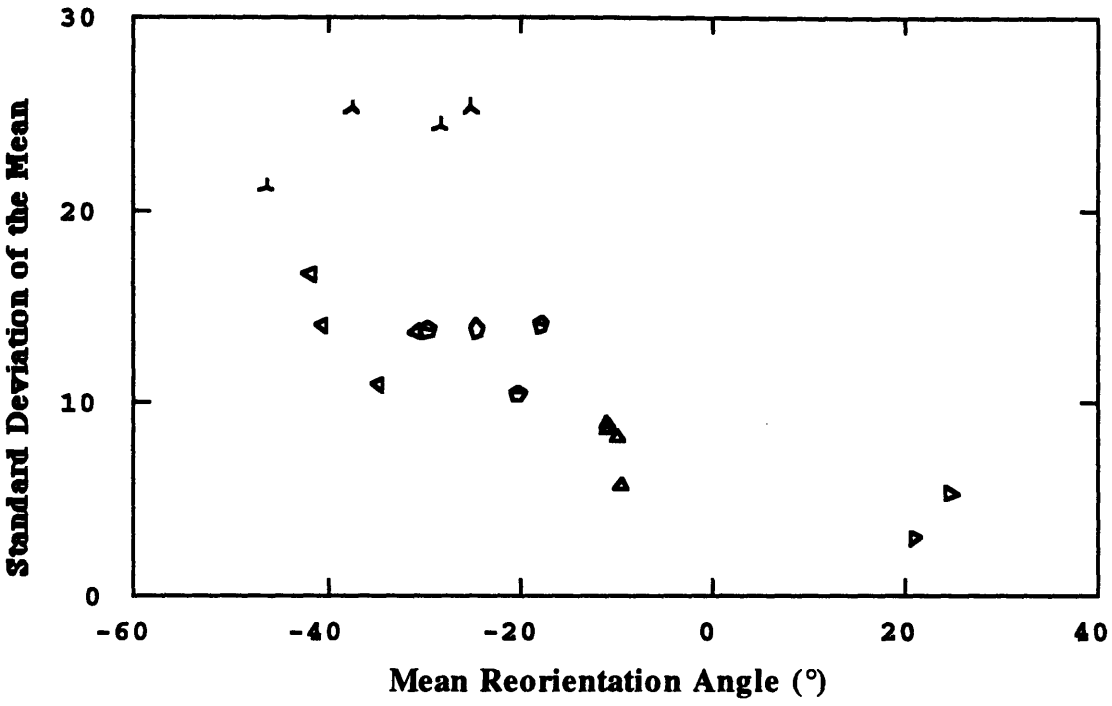

Figure 5-14: Early Subjects' Mean Reorientation Angles for the Walls of the FR and SR.

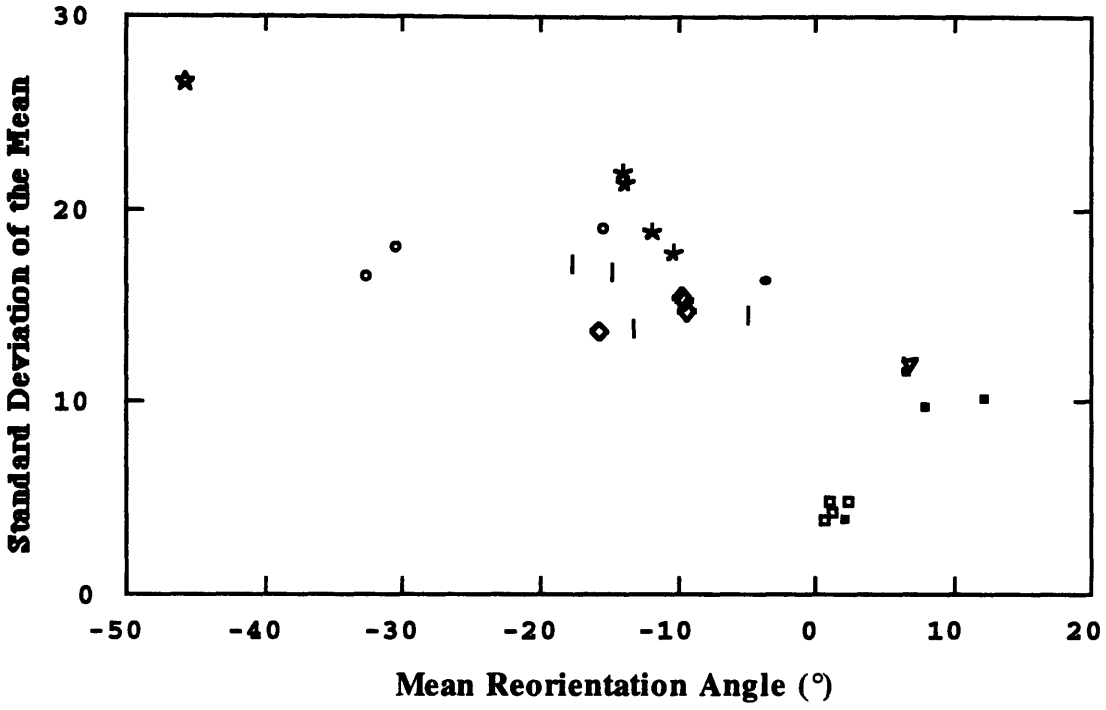

Figure 5-15: On-Time Subjects' Mean Reorientation Angles for the Floors and Ceilings of FR and SR.

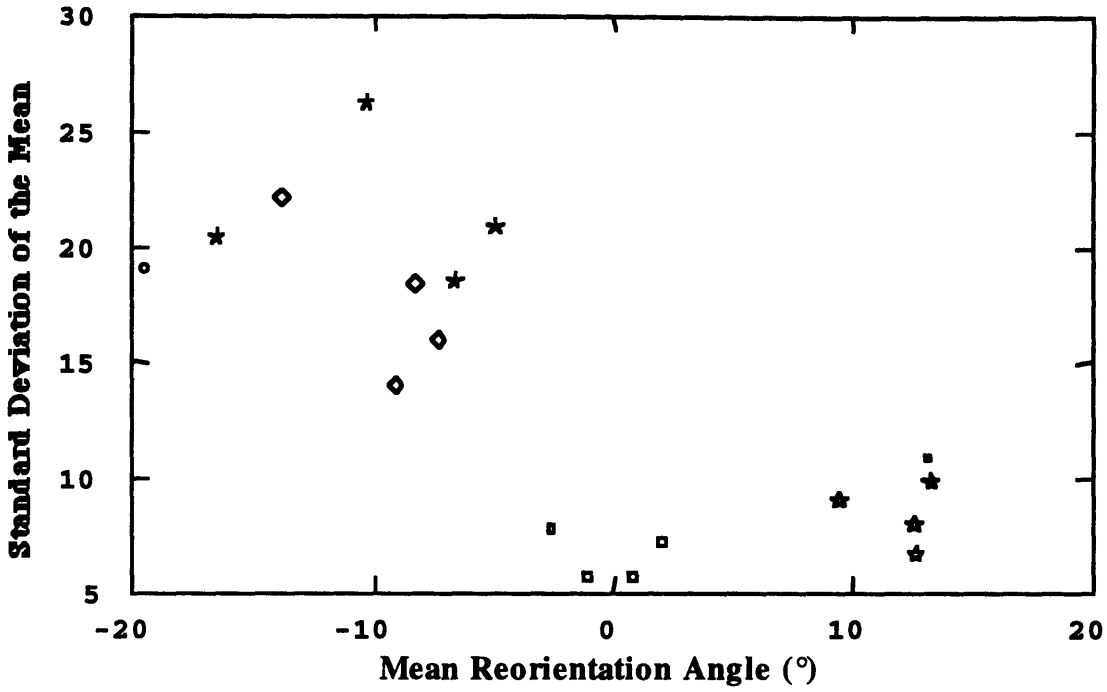

Figure 5-16: On-Time Subjects' Mean Reorientation Angles for the Walls of FR and SR.

Table 5.8 also presents an account of subjects who did not reorient for certain surfaces. One on-time subject (#4) did not reorient for any surface except for the FR floor and SR ceiling. Another on-time subject (#1), did not reorient to any walls of the SR. Two on-time subjects (#2 and #11) reoriented only a few times on various walls of the SR and FR, but many times for the surfaces of the DR. The early subjects had reorientations for every surface of the FR, SR, and DR (except for #12 who did not reorient for any surface of the DR).

ANOVAs of reorientation angle were performed separately for each subgroup. The most salient difference was that early subjects' reorientation angles ($F(3,1811) = 108.05$; $p<0.0001$) were more sensitive to surface than on-time subjects ($F(3,971) = 1.84$; $p=0.1388$). This was consistent with the previously mentioned result that

early-subjects were reorienting earlier for floors and ceilings than for walls, whereas on-time subjects were not reorienting at significantly different angles for surfaces in room.

Although on-time subjects' reorientation *angles* were not sensitive to surface, their *frequency* of reorientations was. An ANOVA done on frequency of reorientations for a given scene and surface showed that subgroup (i.e., early and on-time) was a significant main effect on reorientation frequency ($F(1,168) = 20.3$; $p<0.0001$), and that subgroup * surface was a significant cross effect ($F(3,168) = 4.12$; $p=0.0075$). Table 5.9 lists this ANOVA's fitted LSM frequencies of reorientation for each subgroup. For each scene, on-time and early LSM reorientation frequencies are given by surface. On-time subjects reoriented, on average, more than twice as often for the FR and SR's floors and ceilings than walls, but for the dotted rooms, they reoriented equally as often for all surfaces.

On-time and early subjects reacted to polarity cues in different ways. The (early-subjects, on-time subjects) were reorienting (earlier, more frequently) for polarized than non-polarized surfaces. This explained why early-subjects were not significantly sensitive to scene ($F(2,1811) = 0.6151$; $p=0.54$), but on-time subjects were ($F(2,971) = 14.5$; $p<0.0001$). As shown in Table 5.10, on-time subjects reoriented more frequently when the subjective floors were floors and ceilings of the FR and SR, hence their mean reorientation angles for the FR and SR were influenced more by the highly polarized orientations of the room than by the non-polarized. The early-subjects, on the other hand, reoriented equally as often for walls, ceilings and floors, hence their mean scene reorientation angles for the FR and SR were equally influenced by polarized and non-polarized orientations. This made on-time subjects' mean reorientation angles for a scene more sensitive to polarity.

Table 5.10 through Table 5.13 lists the by-scene-and-surface fitted LSM reorientation angles which were calculated for the ANOVAs. For on-time subjects, the mean reorientation angles (including all surfaces) for the furnished ($p^* = 0.0035$) and symmetric ($p^*<0.0001$) rooms were significantly different from the dotted room. This supported the previous paragraph's explanation that the on-time subjects' more frequent reorientations for the FR and SR's floor and ceiling were making their FR and SR mean reorientation angles more negative.

Table 5.9: Fitted LSM of Frequency Reorientations per Reference Surface (out of 24 passes).

Scene	Subgroup	Surface 0	Surface 1	Surface 2	Surface 3	All
Furnished Room	Early	18.4	17.6	16.2	17.1	17.3
	On-time	19.0	6.7	12.7	6.9	11.3
Symmetric Room	Early	14.6	16.8	14.1	17.7	15.8
	On-time	16.6	6.6	16.6	7.3	11.8
Dotted Room	Early	17.2	17.8	17.0	19.4	17.9
	On-time	13.3	11.3	12.4	12.6	12.4
All Rooms	Early	16.7	17.4	15.8	18.1	17.0
	On-Time	16.3	8.2	13.9	8.9	11.8

Table 5.10: Fitted LSM Reorientation Angle by Subgroup and Scene (All Scenes).

Subgroup	Furnished Room	Symmetric Room	Dotted Room
Early	-27.5°	-26.6°	-26.2°
On-Time	-4.3°	-7.1°	0.3°

Table 5.11: Fitted LSM Reorient Angle for Furnished Room Surfaces.

Subgroup	Floor (0)	Wall1 (1)	Ceiling (2)	Wall2 (3)
Early	-40.1°	-16.5°	-36.7°	-16.6°
On-Time	-5.7°	-3.5°	-1.1°	-7.0°
Combined	-24.4°	-7.1°	-20.7°	-8.0°

Table 5.12: Fitted LSM Reorient Angle for Symmetric Room Surfaces.

Subgroup	Floor/Ceiling (0)	Wall1 (1)	Floor/Ceiling (2)	Wall2 (3)
Early	-40.1°	-12.8°	-38.6°	-15.0°
On-Time	-10.6°	-4.0°	-10.1°	-3.8°
Combined	-26.0°	-4.3°	-25.0°	-6.0°

Table 5.13: Fitted LSM Reorient Angle for Dotted Room Surfaces.

Subgroup	Wall1 (0)	Wall2 (1)	Wall3 (2)	Wall4 (3)
Early	-27.4°	-23.6°	-29.7°	-24.1°
On-Time	--0.9°	1.2°	0.0°	1.0°
Combined	-14.7°	-11.7°	-16.0°	-12.2°

Since on-time subjects reoriented roughly 2/3 as often as the early-subjects, they might have been feeling full-tumbling more often. Indeed, the three subjects who did not feel full tumbling in any of their runs were early subjects. The average frequency of horizontal full-tumbling (HFT) among the on-time subjects was 11.6 and among the early subjects 7.4. Interestingly, the frequency of vertical full-tumbling (VFT) was slightly greater for early subjects (7.8) than for on-time subjects (6.6). Overall, however, on-time subjects experienced vection without any tilt (i.e., full-tumbling in either vertical or horizontal planes) in, on average, in 8 more runs than early subjects

5.2.2 Posture and Perceived Frontal Plane Effects

Since early and on-time subjects were behaving differently with respect to polarity, it was natural to believe they would respond differently to other independent variables. While posture was a significant main effect on early-subjects' reorientation angles ($F(1,1811) = 5.01$; $p=0.0253$), it had no effect on those of the on-time subjects' ($F(1,971) = 0.089$; $p=0.92$). Early subjects reoriented on average 2.9° sooner ($p^* = 0.0253$) when erect than supine.

While posture had no significant effect on on-time subjects' reorientation angle, it may have indirectly influenced their reorientations through the perceived frontal-plane orientation. Some on-time subjects, when they were supine and felt gravitationally horizontal, tended to feel reorientations only when a surface was exactly horizontal below them. They momentarily felt vertical but then reported a return to a horizontal plane of perceived motion. It's possible that subjects did not feel reoriented until the surface was just "beneath" them because they were not expecting a reorientation while they felt horizontal. Unfortunately, this alternating vertical-horizontal sensation was not recorded accurately, which precludes any extended analysis. It seems, however, a reorientation during perceived horizontal-plane rotation will, in at least some subjects, bring a sudden sensation of being vertical with respect to gravity.

5.3 Tilt Illusions

The model outlined at the end of the previous chapter accounts for tilt in several ways. A constant physical tilt will obviously send a "tilt" message to the CNS. When in the erect posture, a subject might actually tilt in order to correct the virtual tilt simulated visually by the virtual room. In that case, the subject would report a illusory tilt in the direction that she actually tilted. In fact, 12 (75%) subjects exhibited physical tilt from postural-adjustments

in at least 5 (21%) the runs in which they were in the erect posture. Not surprisingly, when subjects in the erect posture made postural adjustments, 90% of the time they would report some form of tilt sensation, nearly always corresponding to the posture adjustments they actually made. For runs in which subjects in the erect posture did not exhibit postural adjustments (real tilt) they reported illusory tilt, on average, only 60% of the time. Eventually, the experimenter was able to predict what kind and magnitude of tilt subjects were feeling just by watching them move in the erect position.

When subjects maintained a steady physical tilt, they tended to report constant tilt, but when they oscillated they often reported alternating tilt. Several subjects would tilt their heads slightly forward (probably due to neck fatigue), and later report they felt they were "looking down" at the scene. Often, the subjects' oscillation would coincide with the passing of a floor underneath, as if they were trying to orient themselves so that their feet were pointed to the "current floor" (see Figure 5-17). Two subjects exhibited extreme posture adjustments to counteract the rolling visual field, oscillating 10° about the vertical. It was not a surprise that these subjects tended to report alternating-tilt during these episodes.

Not all the tilt sensations could be attributed solely to automatic postural-

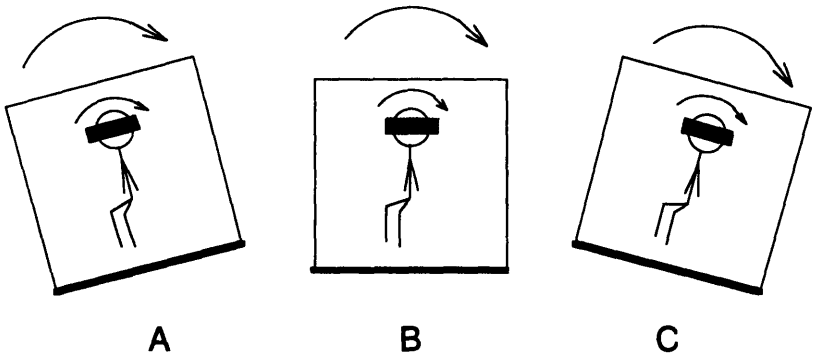

Figure 5-17: Subject Making Postural Adjustments During the Passage of a Surface.

adjustments. The supine posture constrained most of the postural adjustments, but twelve (75%) subjects still reported at least 5 instances of tilt when they were supine (out of 24 runs). Despite the fact that most of the subjects were able to feel tilt while supine, most of the instances of tilt sensation occurred in subjects in the erect posture. Subjects experienced tilt sensations while supine less frequently (45.1% of all supine runs across subjects) than when erect (74.7% of all erect runs across subjects). All subjects felt tilt more often in the erect posture, except for one, who felt tilt equally as often in both postures. These results suggest its easier to induce a tilt sensation in subjects who are erect.

While these previously mentioned thirteen subjects were supine, eight tended to feel constant tilt more frequently than alternating tilt, two felt both constant and alternating tilt equally often , and three tended to feel alternating tilt. This ratio of supine tilt tendencies (constant-tilt: both: alternating tilt = 8:2:3) was not much different from the overall (both postures) tilt tendencies (9:4:3). This suggests that subjects' assumption of the supine posture does not radically alter the type of tilt they feel. Instead, it evenly diminishes the frequency with which they feel any kind of tilt. One of the two subjects who exhibited extreme postural-adjustments and felt alternating-tilt most of the time she was upright, also felt alternating-tilt when she was supine.

5.4 Perceived Frontal Plane Orientation

Subject's sensations (i.e., full tumbling) could be classified as either vertical or horizontal. The subjects' haptic, vestibular, and gravireceptic senses were believed to play an important role in inducing vertical-frontal-plane illusions (VFPI) -- the sensation of being in a plane that is vertical with gravity. A subject had a VFPI whenever she specified a "vertical" (with or without forward or backward tilt) as the dominant frontal-plane orientation for the run, even if the subject did not report any other illusions (i.e., vertical with no vection and tilt).

The following stimuli should induce VFPI in subjects: strong gravitational or haptic down cues directed along the subjects' idiotropic vector, or the rolling of visually mono-oriented objects in a plane perpendicular to the subjects' visual axis. In this experiment, the probability of a subject having a VFPI was more sensitive to her vestibular and/or haptic than visual cues

Four types of VFPI behavior were noted among subjects (See Figure 5-18 and Figure 5-19). The first group (1,2,3,9,12,14) tended to have VFPI's every time they were erect, and never when they were supine. Hence, with respect to VFPI's, this group was highly sensitive to posture, but highly insensitive to scene content. The second group (10,11) exhibited the opposite of this. They exhibited a slight sensitivity to scene content and a relative insensitivity to posture. The third group (4,6,7,8,13,16) exhibited a mixture of both posture and scene sensitivity. The fourth group consisted subjects 5 and 15, who reported no VFPI's, feeling horizontal-frontal-plane illusions for every run. Thus, with respect to VFPI's, the fourth group exhibited no sensitivity to either posture or scene content. The reader must be reminded, however, that several of the subjects also reported brief moments in which they felt an instantaneous VFPI during a run, usually when they were having a reorientation (this result was anecdotal, so it was not clear whether this behavior was unique to any group). Thus, while a subject might be reporting a dominant horizontal-frontal-plane orientation, she might feel brief moments of VFPI due to the strong visual cues that caused the reorientation.

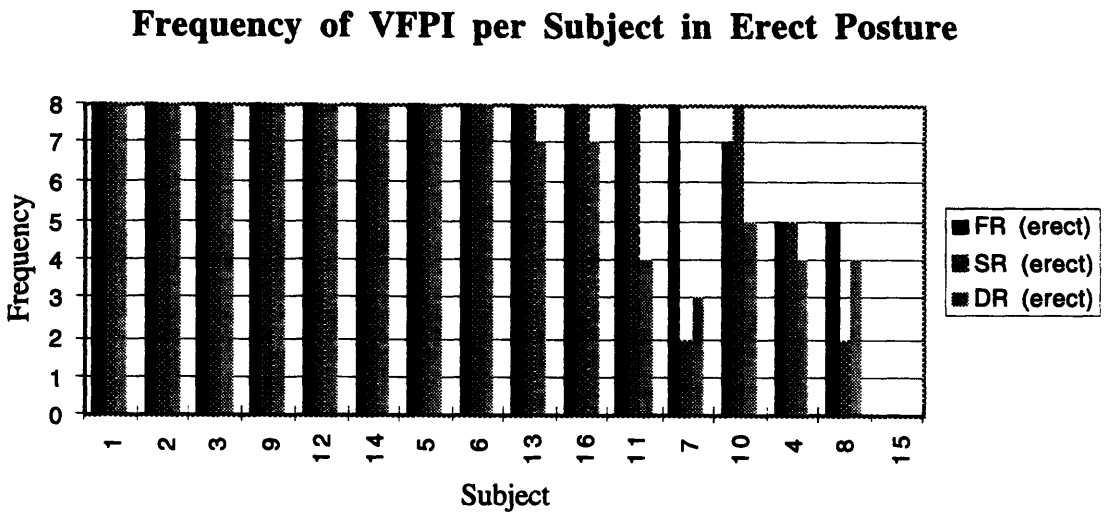

Figure 5-18: Frequency of VFP per Subject (Erect).

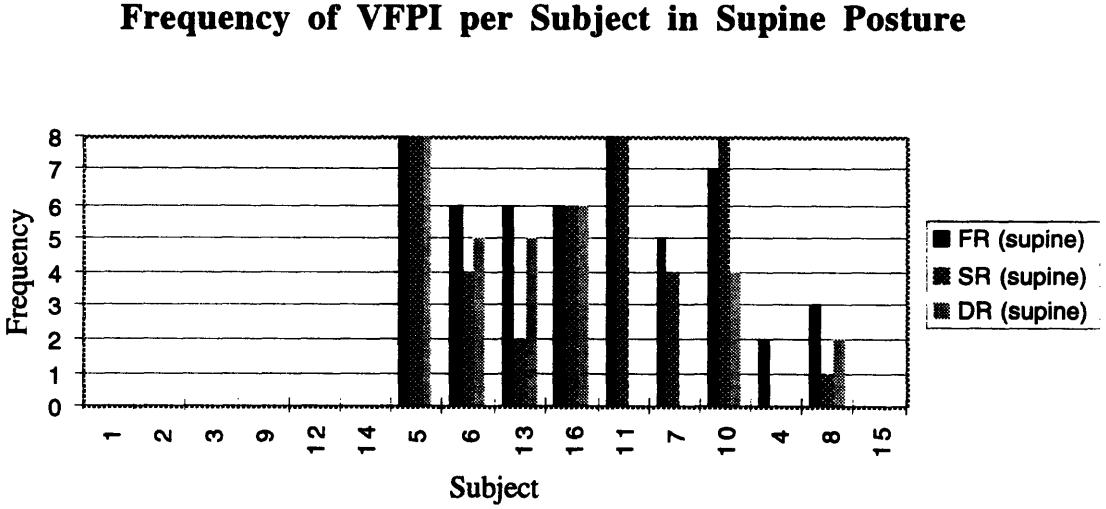

Figure 5-19: Frequency of VFP per Subject (Supine).

The most significant effect on perceived plane of rotation was, not surprisingly, posture. Only one subject ever felt vertical more often in the supine than erect posture. Table 5.14 lists the number of subjects reporting differences in frequency of VFPI between the two postures. By the WSRT, subjects reported significantly more frequent

VFPI's in the erect than supine posture for the FR (WSRT z = -3.11; p = 0.0018), SR (WSRT z = -2.9; p = 0.0037), and DR (WSRT z = -3.3; p = 0.0009). This was expected, and it suggested that it was harder to induce in a horizontal subject the illusion of being vertical in a gravitational field. On the other hand, it showed that at least some subjects will experience a VFPI when they are supine, despite the conflicting vestibular input.

Table 5.14: Number of Subjects Feeling Vertical More Often in One Posture.

# subjects who felt vertical more often in...	FR	SR	DR
erect posture	12	11	14
supine posture	0	1	0

Although subjects reported horizontal sensations more often when supine, several (as described earlier) felt instantaneously upright (with respect to gravity) at the moment they underwent a reorientation. Subjects felt this combined reorientation-upright sensation with reorientations roughly close to $\theta=0°$ (based on discussions with subjects). While this phenomenon was not reported by all subjects, nor was it recorded in detail, it seemed to occur more often with the furnished and symmetric room, than with the dotted room. This suggested that the reorientation-vertical illusions were due to a momentarily strong visual-down cue caused by polarity of the scene, and that this polarity made subjects feel "upright" for a moment, despite their really being supine.

6. Conclusions

The development of the flexible experimental control software used for this research makes completion of future experiments related to visual-vestibular interaction easier. Another researcher [Markmiller, 1996] in the laboratory in which this experiment was conducted recently finished his experiments, using the experimental control software developed for this study. Markmiller's research was on perceived *linear* self-motion under conflicting visual, haptic and vestibular stimuli. Markmiller spent around one week designing the scenes for his experiment, using 3D Studio by Autodesk, and then imported the 3D Studio model into World Tool Kit for use in his experiments. He spent only one more week expanding the experimental control software to allow linear as well as rotational motion. The modular design of the software made it necessary for him to modify only a small portion of the C source code. The software was provided to NASA at its request, and is being evaluated by NASA as a possible way of controlling experiments on the NASA Virtual Environment Generator which is to be flown on the 1998 Space Shuttle mission, Neurolab.

6.1 Vection

When erect and viewing the FR, 9/16 (56%) subjects eventually felt vertical full tumbling (VFT) but when supine only 6/16 (38%) did. These percentages were roughly comparable to the ones DeSouza observed in his subjects. In this experiment, however, it was not possible to show that individually subjects did feel VFT more often when they were erect than supine while they were viewing the FR. Posture had a significant effect on frequency of VFT only for the dotted room (DR), where polarity was completely absent. When

viewing the DR, 9 subjects felt VFT more often (2 less often) when erect than when supine. Subjects were dependent on vestibular cues when the visual field was not saturated by mono-oriented objects (e.g., as in the DR).

Consistent with the above explanation, 6 subjects when supine felt VFT more often (1 less often) when viewing the FR than DR. For subjects in the erect posture, however, there was no significant increase in frequency of VFT from the DR to the FR. Subjects, when supine, will therefore feel vertical more often when exposed to a highly polarized visual field perpendicular to their visual axis than when exposed to a non-polarized visual field. When viewing the FR and symmetric room (SR), subjects did not feel VFT more frequently in any one posture, probably because they were exposed to visual polarity that compensated for the lack of vestibular down cues aligned with their head-to-toe axis.

While VFT illusions were generally due to the dominance of highly polarized visual cues, horizontal full-tumbling illusions (HFT) -- the illusion of full 360° rotation in a gravitationally horizontal plane -- were more often driven by vestibular dominance. HFT illusions were reported very infrequently among the runs in which subjects were erect, suggesting that vertical vestibular cues were keeping subjects from feeling horizontal. It was easier for a subject to feel horizontal when she was actually supine, but when viewing highly polarized scenes, like the FR and the SR, she would additionally need to discount the polarity of the scene in order to feel HFT. When supine, the subject might also reinterpret the surfaces of the SR and the DR so as to resolve sensory conflict. If, when supine, the subject perceived the far wall of the SR as a ceiling and its windows as a skylight, she could resolve all sensory conflict when feeling HFT.

When exposed to a moving strong visual polarity stimulus, the human central nervous system (HCNS) may be convinced earlier than it normally would be that it

is moving. In effect, strong polarity should reduce latency. The results were consistent with this hypothesis, since the FR, SR, and DR induced vection on average within 7.2, 7.6 and 9.2 seconds respectively. This result, however does prove that polarity was the main cause shorter latencies. It was also possible, that the shorter latencies were due instead to different spatial frequencies of the scene. The larger number of moving lines in the FR and SR, for example, may have had a partial effect on vection latency.

It was surmised that putting the subject in the supine posture would significantly decrease latencies since the subject's visual axis would be aligned with the rotational axis. This trend did not manifest itself in the data, suggesting that the "vection" sensor in the HCNS was not as sensitive to the otolith cues as it was to the visual input. The HCNS may also have quickly learned to "discount" the vestibular reports of "no-motion" after repeated exposure to sensory-conflict.

Subjects followed a trend of faster perceived vection velocities for more polarized rooms ($SAT_{FR} = 47.2\% > SAT_{SR} = 42.9\% > SAT_{DR} = 35.1\%$). Thus, while the onset of vection was not as sensitive to scene polarity as previously thought, the intensity of vection was. Subjects were using the mono-oriented objects as reference points to determine their own vection velocities. When many polarized objects were rolling in their visual frames (the egocentric frames), subjects felt faster velocities of self-roll in the exocentric frame. The SR, not having any mono-oriented objects, may have induced greater vection saturation because it had a larger number of parallel lines than the dotted room. As with latency, posture only slightly effected saturation ($SAT_{ERECT} = 40.1\% < SAT_{SUPINE} = 42.6\%$), probably for the same reason as why posture did not significantly influence vection latency. The HCNS may have quickly learned to discount the conflicting vestibular input when determining self-roll velocities. For each scene, however, subjects

reported vection saturation on average a few percent more when supine than when erect, suggesting that posture may have a slight but not significant effect on saturation.

6.2 Reorientations

The reorientation behavior of subjects was by far the most interesting result of this research. Based on their mean reorientation angle in the FR when the real floor was reinterpreted as a subjective floor (reference surface), subjects were placed into two categories: "early" and "on-time." Subjects who reoriented early in this situation tended to do so as well when the subjective floors were the FR ceiling and the SR floor/ceilings. They exhibited mixed early and on-time behavior when the walls of the FR and SR and all surfaces of the DR were becoming subjective floors. In each scene the on-time subjects tended to reorient at around the same angle when each surface was becoming a subjective floor, and these angles were usually later than those of the early subjects.

The early subjects seemed to respond to visual polarity in a scene by reorienting sooner to the highly polarized orientations of the scene. They tended to reorient for all 4 subjective "upright" orientations of a scene about 2/3 of the time. In the DR, where polarity was absent, early subjects did not exhibit any significant difference in reorientation angles for the different subjective "upright" orientations but still reoriented sooner, on average, than on-time subjects did. This suggested that early subjects were more attracted to reorienting to highly polarized orientations of a scene, i.e., when the real floors and ceilings were in the appropriate places.

On-time subjects responded to lower levels of polarity not by reorienting later, but by reorienting less frequently. Overall, on-time subjects reorientd two-thirds as often as early subjects did. Many on-time subjects reoriented at least 2 times as often for orientations of the scene in which the FR and SR floor and ceilings were the subjective

floors than when they were the walls. They reoriented more often for the highly polarized orientations of the rooms. Therefore, their overall mean reorientation angles for a scene were more sensitive to the scenes polarity. The early subjects reoriented equally as often for each subjective "upright" orientation of the scene. Hence, their mean reorientation angles for scenes were not as sensitive to scene polarity. Also, the decreased frequency of reorientations among on-time subjects might be related to an increase in frequency of full-tumbling.

It was possible to explain early subjects' behavior using either a "competing-walls" or a "competing-floors" model. In the former, pairs of parallel surfaces in a scene (i.e., wall-wall or floor-ceiling) compete to become subjective walls in the subject's interpretation of the scene. The real walls of the FR and SR do indeed feel more like walls. Hence, subjects will believe the actual walls are the subjective walls of the scene earlier than they would the floor and ceiling. Subjects would feel reoriented sooner when the real walls were approaching their appropriate places. The competing-floor model would work similarly, but the subject relied on her interpretation of which surface was the subjective floor rather than which surfaces were the walls. The actual floors of the FR and SR were more readily interpreted than walls as subjective floors. Add to this, the assumption that the subject did not feel comfortable with the wall as a subjective floor, and one would hypothesize that the subject would interpret the ceiling early as a subjective floor in order to reduce discomfort. Both models can explain early subjects -- and with a few modifications on-time -- subjects' behavior. Perhaps the early subjects have expectations as to what types of reorientations will occur as the scene rotates about them, whereas the on-time subjects do not. The more naive on time subjects would be asking, "could it be a wall?" or "could it be a floor?" instead of "has it become a wall yet ?" or "has it become a floor yet?".

6.3 Future Research

Future experiments should explore the reorientation behavior of a larger pool of subjects. Subjects will probably fall into two categories of behavior. Roughly half will tend to reorient for all 4 subjective "upright" orientations in a rolling room but will reorient sooner for more polarized orientations. The others will react to less polarized orientations of the room by not reorienting at all, and when they do reorient, they will at later angles than for early subjects.

Testing subject's reactions to different versions of the furnished room would be interesting. If one were to make each of the 4 subjective "upright" orientations of the FR equally polarized, it would be possible to test whether subjects reorient sooner for some orientations because of greater overall scene polarity of that orientation or because subjects track specific objects and determine their orientation based on just a small component of the scene. If the mono-oriented furnishings were randomly assigned to a surface (i.e., the table and computer were on the wall, the book shelf on its side, the door on the floor) then a FR with 4 equally polarized upright orientations could be created.

"Wall-less" and "floor-less" furnished rooms could help determine whether subjects are following a "competing-floors" or "competing-walls" model. If subjects reoriented at similar angles for all reorientations of the "wall-less" FR it would imply that subjects used a competing-floor model. If, on the other hand, subjects reoriented at similar angles for all reorientations of the "floor-less" room, it would mean subjects followed a competing-walls model. A floor-less room can be made by removing all the objects (e.g., carpet) from the FR's floor. Walls can then be emphasized by adding objects like paintings, doors or book shelves. Following a similar strategy one can design a wall-less FR.

Research should be done to explore why the FR did not induce subjects to feel vection dramatically sooner than for the SR. It may be that rolling of a highly polarized

scene does not in itself induce a subject to feel vection sooner, but rather the rolling of *stationary* objects (e.g., windows or frames) in any orientation (e.g., scattered or aligned) convinces the subject's HCNS that it and not the scene is moving. Again, a FR with randomly oriented furnishings could help determine whether the alignment of mono-oriented objects significantly effects the HCNS when determining self-orientation

While full-tumbling illusions need to be studied in more detail, a careful distinction must be made between their horizontal and vertical counterparts. Each illusion represents a different form of conflict resolution. More subjects need to be tested to determine whether visual polarity has any significant effect on the frequency of VFT illusions in the erect posture. Also, does the training of subjects to feel reorientations make them less susceptible to full-tumbling?

Future researchers that study illusory tilt in erect subjects should carefully record direction of physical tilt and illusory tilt to see if a significant correlation exists. While originally thought that subjects would tilt in one direction with respect to the scene rotation direction, many subjects physically tilted and/or felt illusory tilt in both directions with respect to scene motion.

7. References Cited

[1] Allison, R., Zacher, J., Howard, I.P., *Report on the Effect of Field Size, Head Motion and Rotational Velocity on Roll Vection and Illusory Self-Tilt in a Tumbling Room.* unpublished report. 1995.

[2] Anderson, G.J., Dyre, B.P. Spatial orientation from optic flow in the central visual field. *Perception & Psychophysics*, 1989 **45**, 453-458.

[3] DeSouza, J., *A Virtual Environment System for Spatial Orientation Research.* Masters Thesis: Massachusetts Institiute of Technology, 1995.

[4] Dichgans, J., Brandt, T., Visual-vestibular interaction: effects of self-motion perception and postural control. In R. Held, H.Leibowitz, & H.L. Tueber (eds.), Handbook of Sensory Phsyiology: Vol. 8. Perception (pp. 756-804). New York: Springer-Verlag.

[5] Held, R., Dichgans, J., Bauer, J., *Characteristics of moving visual areas influencing spatial orientation*, Science, 144, 1975.

[6] Howard, I.P. *Human Visual Orientation.* (Chichester, Sussex: John Wiley) 1982.

[7] Howard, I.P., Cheung, B., Landolt J, *Influence of vection axis and body posture on visually-induced self rotation*, Advisory Group for Aerospace Research and Development, 433, 1988.

[8] Howard, I.P., Childerson, L., *The contribution of motion, the visual frame, and polarity to sensations of body tilt,* Perception, 1994, volume 23.

[9] Kitaharo, M., and Uno, R. "Equilibrium and vertigo in a tilting environment." Ann. Otol., 1967 **76**, 166-78.

[10] Kleint, H., "Versuche über die Wahrnehmung. I. Über Bewegung" *Zeitschrift für Psychologie* 1937 141 9-44.

[11] Markmiller, M. *Sensory Interaction in Human Perception.* Masters Thesis: Massachusetts Institute of Technology, 1996.

[12] Mittelstaedt, H., "A new solution to the problem of the subjective vertical" *Naturwissenchaften* 1983 **70**, 272-281

[13] Oman C.M., A heuristic mathematical model for the dynamics of sensory conflict and motion sickness. Acta. Otol. Suppl. 1982; 392.

[14] Oman, C.M., Howard, I.P., Carpenter-Smith, T., *Role of Visual Cues in Microgravity Spatial Orientation, Investigation Description.* 1995.

[15] Oman, C.M., Lichtenberg, B.K., Money, K.E., *Canadian vestibular experiments on the Spacelab-1 mission*, Experimental Brain Research 64, 1986.

[16] Oman, C.M., *The role of static visual orientation cues in the etiology of space motion sickness*, Pro. Symposium on Vestibular Organs and Altered Force Environment, 1987.

[17] Witkin, H. A., Asch S. E., Studies in space orientation: II, Perception of the upright with displaced visual fields and with body tilted. *Journal of Experimental Psychology* 1948; 38:455-77.

[18] Wood, R.W., "The haunted swing illusion" *Psychological Review* 2 pp .277-278, 1895.

[19] Young, L.R., Jackson, K., Groleau, N., Modestino., S, "Spatial Orientations and Posture During and Following Weightlessness: Human Experiments on Spacelab Life Sciences 1" *Journal of Vestibular Research*, Vol 3., pp. 231-239, 1993.

[20] Young, L.R., Oman, C.M., Dichgans, J.M., *Influence of head orientation on visually induced pitch and roll sensations,* Aviation, Space and Environmental Medicine 1975 **46** 264-268.

8. Appendix: ANOVAs

All ANOVAs cited in this thesis are listed below.

8.1 Latency

```
DEP VAR: TIMELAT      N:     372  MULTIPLE R: 0.629  SQUARED MULTIPLE R: 0.396
ANALYSIS OF VARIANCE

SOURCE        SUM-OF-SQUARES   DF  MEAN-SQUARE     F-RATIO       P

SUBJECT          3808.4086    15     253.8939     14.0525      0.0000
SCENE             298.5149     2     149.2575      8.2611      0.0003
POSTURE             5.1273     1       5.1273      0.2838      0.5946
SCENE
*POSTURE            1.2560     2       0.6280      0.0348      0.9658

ERROR            6341.6870   351      18.0675
```

8.2 Saturation

Note: DOVECT is a variable that specifies whether the subject was using the joystick trigger to report vection (1) or reorientations (0).

```
DEP VAR: VECTSAT      N:     768  MULTIPLE R: 0.695  SQUARED MULTIPLE R: 0.483
ANALYSIS OF VARIANCE

SOURCE        SUM-OF-SQUARES   DF  MEAN-SQUARE     F-RATIO       P

SUBJECT        167486.2196    15   11165.7480     41.3341      0.0000
SCENE           19347.3132     2    9673.6566     35.8106      0.0000
POSTURE           600.3138     1     600.3138      2.2223      0.1365
DOVECT           2296.1484     1    2296.1484      8.5000      0.0037
SCENE
*POSTURE          394.1608     2     197.0804      0.7296      0.4825

ERROR          201519.9909   746     270.1340
```

8.3 Reorientation Angle

8.3.1 All Subjects

```
DEP VAR:EFOLDDEG      N:    2828  MULTIPLE R: 0.604  SQUARED MULTIPLE R: 0.364
ANALYSIS OF VARIANCE
```

SOURCE	SUM-OF-SQUARES	DF	MEAN-SQUARE	F-RATIO	P
DAY	0.4704	1	0.4704	0.0013	0.9717
DIRECTIO	2446.3165	1	2446.3165	6.5336	0.0106
SUBJNBR	422080.9911	15	28138.7327	75.1532	0.0000
SCENE	1524.9476	2	762.4738	2.0364	0.1307
PLANE	2143.1743	1	2143.1743	5.7240	0.0168
POSTURE	1218.8293	1	1218.8293	3.2553	0.0713
SURFACE	108413.7695	3	36137.9232	96.5175	0.0000
SCENE *SURFACE	36236.4542	6	6039.4090	16.1301	0.0000
ERROR	.104725E+07	2797	374.4183		

8.3.2 Early Subjects

DEP VAR:EFOLDDEG N: 1835 MULTIPLE R: 0.582 SQUARED MULTIPLE R: 0.338
ANALYSIS OF VARIANCE

SOURCE	SUM-OF-SQUARES	DF	MEAN-SQUARE	F-RATIO	P
DAY	1451.6093	1	1451.6093	3.6355	0.0567
DIRECTIO	1849.9137	1	1849.9137	4.6331	0.0315
SUBJNBR	138805.5636	8	17350.6955	43.4545	0.0000
SCENE	491.1643	2	245.5822	0.6151	0.5407
PLANE	3351.0838	1	3351.0838	8.3927	0.0038
POSTURE	2001.3325	1	2001.3325	5.0123	0.0253
SURFACE	129425.3684	3	43141.7895	108.0479	0.0000
SCENE *SURFACE	39459.9251	6	6576.6542	16.4711	0.0000
ERROR	723103.4391	1811	399.2841		

8.3.3 On-Time Subjects

ANALYSIS OF VARIANCE

SOURCE	SUM-OF-SQUARES	DF	MEAN-SQUARE	F-RATIO	P
DAY	1913.9218	1	1913.9218	6.7186	0.0097
DIRECTIO	1010.4378	1	1010.4378	3.5470	0.0600
SUBJNBR	51222.8788	6	8537.1465	29.9686	0.0000
SCENE	8269.3142	2	4134.6571	14.5142	0.0000
PLANE	16.1846	1	16.1846	0.0568	0.8117
POSTURE	2.5425	1	2.5425	0.0089	0.9248
SURFACE	1569.6417	3	523.2139	1.8367	0.1388
SCENE *SURFACE	2943.0913	6	490.5152	1.7219	0.1127

ERROR	276608.5141	971	284.8697

8.3.4 Frequency of Reorientations

TREND is 1 for early subjects, 0 for on-time subjects (i.e., TREND=subgroup)
DEP VAR: COUNT N: 192 MULTIPLE R: 0.465 SQUARED MULTIPLE R: 0.216
ANALYSIS OF VARIANCE

SOURCE	SUM-OF-SQUARES	DF	MEAN-SQUARE	F-RATIO	P
SCENE	59.1133	2	29.5566	0.4746	0.6230
SURFACE	383.3151	3	127.7717	2.0515	0.1086
TREND	1262.6045	1	1262.6045	20.2722	0.0000
TREND*SCENE	33.1133	2	16.5566	0.2658	0.7669
TREND *SURFACE	769.8776	3	256.6259	4.1204	0.0075
SCENE *SURFACE	289.1943	6	48.1990	0.7739	0.5915
TREND*SCENE *SURFACE	230.4443	6	38.4074	0.6167	0.7168
ERROR	10463.4603	168	62.2825		

9. Appendix: Recruitment and Training

Subjects were solicited via electronic mail advertisements targeted at the MIT community. Subjects were then trained using two scenes (Figure 9-1 and Figure 9-2), a large dotted wall and a furnished room (different from the one used to collect data).

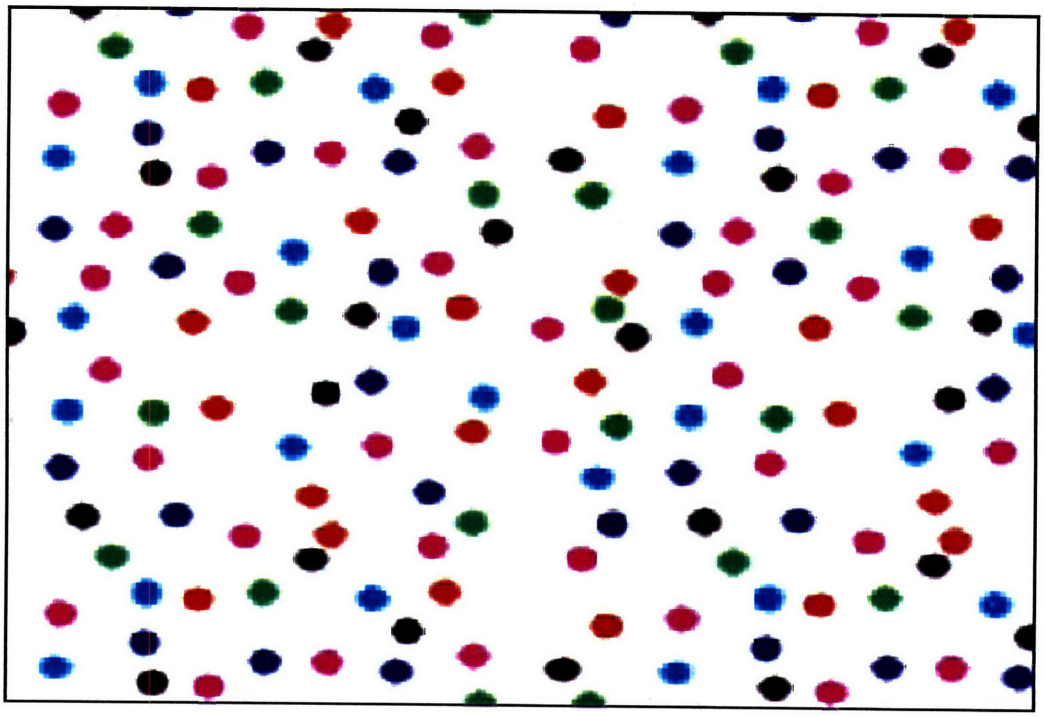

Figure 9-1: Dotted Wall Training Stimulus.

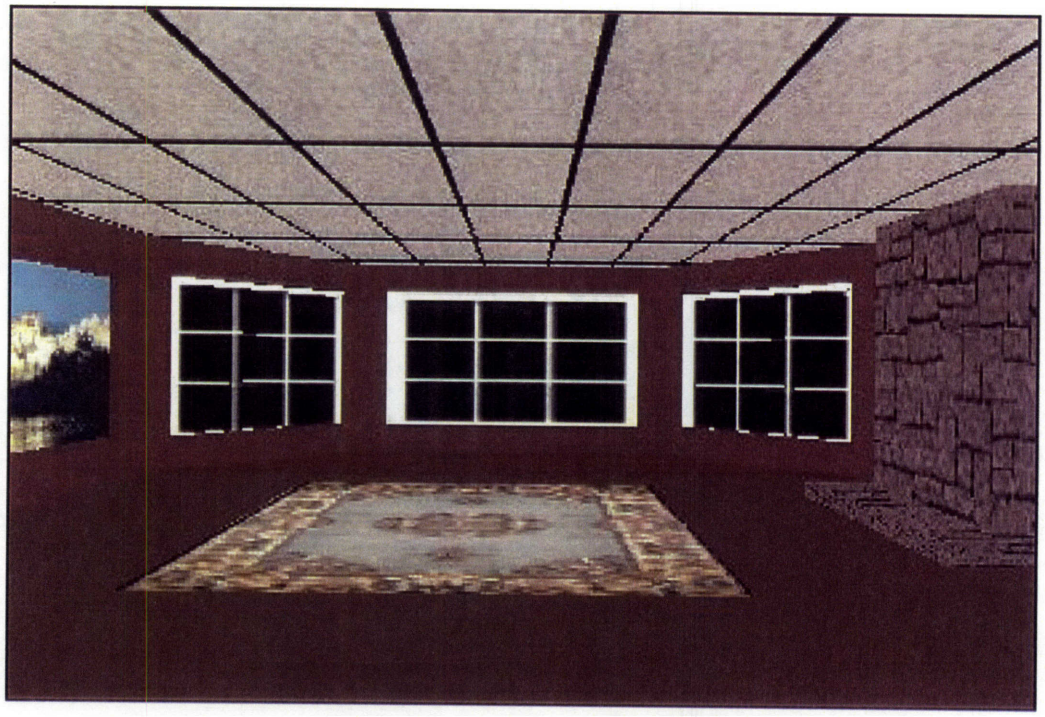

Figure 9-2: Furnished Room Training Stimulus.

The following information-packet/questionaire was given to all applicants:

Virtual Tumbling Room Experiment

Introduction

This experiment, which takes about an hour, studies how scene content influences our sense of rotation and tilt when we use "virtual reality" head mounted displays. Our goal is to understand how people use different objects in the visual scene to know how much they are tilted or rolled, and how fast they are rotating. The illusions you may experience in the experiment relate to the striking "visual reorientation illusions" experienced by astronauts: When the "down" arrow of gravity is absent, astronauts floating upside down in the spacecraft cabin say they often feel that the spacecraft ceiling

and the floor have somehow exchanged identities. Results of our experiment may allow us to improve the design of spacecraft interiors and certain types of "virtual reality" displays, so that users have a better "sense of where they are".

In order to do this experiment properly, we need to train you to distinguish and report on different aspects of your subjective sensations.

As the first step in your training, we ask you to read these instructions and fill out the questionnaire before you come in for the experiment. Results from the experiment and your answers to the questionaire will be kept completely confidential.

If you have any questions that you would like answered before your scheduled experiment session, I can be reached by Email at ----------, or leave a message with ----------.

When you come in for the experiment, we'll continue your training, and then perform the experiment itself.

Wearing the Head Mounted Display

For these tests you will wear a Head Mounted Display (HMD). In half of the runs, you will be sitting upright. For the remainder you will be lying on a bed, facing up.

The HMD resembles a large diving mask, with a sand bag on the strap behind your head to help balance the weight when your head is upright. Two color LCD TVs with lenses in front are mounted inside. All the scenes you will be seeing are created by the computer. The computer draws a slightly different scene for each of your eyes, producing an illusion of depth.

You won't be able to wear glasses. Once you have donned the HMD, help us check the following things:

1. Is the display reasonably comfortable ? (Are the straps too tight ?)
2. Does the scene seem in acceptable focus in each eye ?
3. When you look with both eyes, are you seeing double ? (We can adjust the offset between the two eyes).
4. Do both LCDs are about the same brightness ? (if not, adjust the display up or down, or left and right your head bit).
5. If you look around the scene, can you still see all of it ? (if not, try adjusting the display left or right a bit.)
6. Is stray light entering the headset. (is it pretty dark when the TV's are off?)

Training:

We'll first train you to report on various aspects of your sense of rotation and tilt. To do this, we first need to familiarize you with visual motion illusions, and teach you a vocabulary for describing them by means of three demonstrations.

Demonstration 1 - "Vection"

The term "vection" refers to sensation of self-motion - the sensation that **you**, not the room, are moving. In the first demonstration, we'll position you lying down, looking up at a polka-dotted surface that will start to rotate. After a few moments, you will may experience the illusion that *you* are moving (even just a little), in the opposite direction of the dots. The sensation probably will be of angular motion, which we call "circular-vection", though you may also experience a component of "linear vection", i.e. illusory motion in a straight line. The character of the vection sensation may remind you of illusory motion sensations you have felt in wide screen movies, like IMAX.

You will report the onset of the illusion of vection with a joystick. Hold the handle, and push-and hold the index finger trigger when you feel **yourself** starting to move. Try to distinguish self-motion from scene motion. The illusion of self motion often takes some time to develop. Once it does, you may feel that the dots in the scene are moving more slowly, or even seem stationary.

While the scene continues to rotate, it is also possible that your vection sensation will disappear momentarily. If this happens, release the trigger to report this "drop out", and press the trigger again when you feel motion once again. This way we can measure the latency (delay) of your vection sensation after the scene starts to move, and also the fraction of the time that you feel vection during the run.

After each run, we will ask you some follow up questions: How much of the relative motion you saw seemed to be associated with motion of the scene (as opposed to your own motion in the opposite direction?) Try to judge this saturation on a percentage scale from 0 to 100. 0% means that you feel completely **stationary** and the scene seems to be moving around you. At the other end of the scale, 100% means you feel as though **you** are moving, and the scene is completely stationary. A saturation of 50% means you feel that you are moving in one direction, and the scene seems to be moving in the opposite direction at about the same speed. Practice estimating your vection saturation as the scene spins, and give us verbally an estimate of the average level of vection saturation at the end of each trial.

We will also ask you how the plane of your illusory motion seemed oriented with respect to gravity. Did you feel that all your motion was in an earth-horizontal plane, parallel to the floor of the laboratory ?

Demonstration 2 - Visually Induced Tilt

In this next demonstration, you'll be looking at the same scene, this time while sitting upright. Once the scene starts to rotate, you'll probably feel some vection in the direction opposite to the scene motion, and begin to feel tilted away from the erect position. However, within a few moments, your sensation of tilt may stop increasing -i.e. you'll feel tilted at an approximately constant angle with respect to the laboratory vertical. If so, try to judge your final angle of tilt with respect to the vertical. How many degrees is it ?

Notice that even though you are tilted at approximately a constant angle, you may still have a sense of movement. You feel vection, and you may even be able to estimate its percentage of saturation as in experiment 1, but paradoxically your tilt angle is limited - you feel you should be tumbling head over heels, but you aren't !. It may be that your angle of tilt will fluctuate. One moment you feel tilted, and the next you feel upright again. We call this oscillating tilt sensation "*alternating tilt*".

Try to judge the plane of your circular vection and tilt with respect to gravity. Does your motion and tilt seem to be in an *earth vertical* plane ? Or is the axis tilted somewhat *forward* (face down) or *backward* (face up) ? By how many degrees ? To help you describe your tilt angle after the run, we have an artist's mannequin that you can use to demonstrate your sensation after the test.

Demonstration 3 - Visual Reorientation Illusions

In this demonstration, we'll ask you to look at a scene showing a furnished room while sitting upright. Notice the familiar objects in the room. The experimenter will go through these with you. Next, the experimenter will roll the scene by successive small increments around your line of sight. Try to keep your gaze centered on the middle of the far wall, but try to take in the whole scene. Notice that as the room eventually turns upside down, there is a natural tendency for you to feel that the ceiling of the room somehow seems like a generic floor, and the floor seems like a ceiling. Of course you know which is which if you think about it, because you remember details about "correct" orientation, but nonetheless, the ceiling somehow seems like a floor. Instead of feeling upside down in a right side up room, you may suddenly feel right side up in a room which is similar to the original, except that objects that you remember were on your right now are found on your left.. It may also be that the walls will sometimes seem like floors or ceilings, depending on your orientation to them.

We call these changes in perceived identities of the walls, ceiling, and floor "visual orientation illusions". Notice the top button (not the hat) on the joystick, beneath your thumb. As the experimenter moves the room around give the that button a brief click (press-and-release) as soon as you notice that a ceiling, floor, or wall has exchanged identities. This will tell us the scene orientation at which your "visual reorientation illusions" occurred.

When the experimenter moves the image of the room you may feel some illusory sensations of tilt. This is to be expected, and will be explored in the main experiment.

That's basically all the training you need. If you have any questions about the terminology, the experiment procedures, or anything else you'd like to clarify with the experiment, this is a good time to do so.

EXPERIMENT

Based on your training experiences, we will now ask you to view a variety of different rotating scenes in several different head positions, indicating vection onset and dropout with your index finger on the joystick trigger, and any reorientation illusions (changes in subjective identity of the ceiling, walls, or floor) with the thumb button. After each run, we'll ask you to describe:

a) the percentage of saturation of your vection, if you can, on the 0-100% scale.

b) categorize your vection and tilt as:

none

constant tilt,

alternating tilt (feeling tilted and then suddenly upright again)

full tumbling

c) categorize the plane of your illusory vection and tilt with respect to the vertical. Was your motion and tilt in:

a supine plane,

an earth vertical plane,

a head tilted backwards plane, or

head tilted forward plane ?

Even though you've had some training, it will probably take you a couple of runs to get comfortable with all this. Don't worry about it. We've allowed for this in the statistical experiment design. Just do the best you can. If you realize you've made a reporting mistake, tell the experimenter about it after the run. If you're uncertain about any of the definitions, just ask again !

We really appreciate your taking the time to be a subject in this experiment.

Virtual Environment Questionaire

Do you have medical conditions that would be aggravated if you became motion sick? (yes,no) ***If you said "yes," you should not be a subject for this experiment and you should stop right now. Otherwise, please continue...***
Have you ever experienced dizzy spells? (yes,no) ***If yes, can you please describe these experiences?***
or ... motion sickness? (yes,no) ***If yes, can you please explain some of your experiences?***
What is you dominant eye? (left,right) ***To find your dominant eye, hold your index finger up about 10 inches from your eyes and close each eye one at a time. If you close one eye and your finger seems to move to to the side, the other eye is dominant.***
Do you have normal peripheral vision? (yes,no) **Do you have normal depth perception? (yes,no)** **Do you need corrective lenses? (yes,no)**

Check all that apply. I have...	
☐astigmatism	☐dyslexia type(s):
☐near sightedness	☐blind spots where:
☐far sightedness	☐phoria ☐wall eye
☐color-blindness color(s):	☐strabismus

Do you have any hearing loss? (yes,no)

If yes, please explain how you have/are lost/losing your hearing.

Do you have any balance problems? (yes,no)

If yes, please describe the nature of your balance problem(s)?

Do you have a history of chronic ear infections? (yes,no)

If yes, can you please elaborate?

What is your gender?(female,male)

Previous Experience

Please describe any experience you have had with Virtual Reality systems.

10. Appendix : Experimental Software

This section describes the design and operation of the experiment software. The software proved itself to be valuable in conducting the experiments in this thesis, and it has the potential to help others who are performing research with WTK and virtual environments. The software was written as a combination of modules using C, and the package is portable to all compilers that have ANSI support. The software can also be adapted to other virtual reality toolkits by simply rewriting the experiment-driver module.

10.1 Software Design

This section will discuss the design of the experiment software. First, the motivation and requirements for the software are described. Then, the module-based implementation is detailed. That section also outlines the advantages and disadvantages of the implementation. Finally, the various modules in the program are expounded.

10.1.1 Background and Requirements

The software was required to be flexible to the point where the researcher could alter the design of an experiment without having to recompile the executable. This flexibility was balanced with the need for a dummy-proof method of conducting the experiments. For example, rather than having a user input rotation speed, duration, scene, etc., each time an experiment is run, the software reads them from text files, or scripts. When the software relies on scripts, it can control the experiment with greater consistency and reliability.

A software interface that requires minimal interaction with a human experiment controller makes it easier to conduct experiments in space. Astronauts would require less time handling the software and more time performing tests on themselves. In addition, the experiment could run itself, without needing an extra person to control the

experiment. If the software takes the role of the controller, it reduces the number of work hours required run the experiment and makes it possible to do more of them.

To run an experiment, the software needs the order in which to test different conditions. An experiment schedule is made up of a series of runs in which a set of conditions (or independent variables) is tested. In this experiment, a scene, velocity-profile, direction and posture were specified in each run. Many variables can be specified with an integer, i.e., direction can be 1 for CW and -1 for CCW rotation, but others, such as scene content, require an amount of information that is better kept in its own file.

10.1.2 Implementation

The software's requirements indicate that an module-based design would improve its reliability and enhance its ease of use. Most perceptual research is performed with a balanced schedule of conditions, and, to do this, the software must repeat presentations of a variable. In this experiment, for example, the schedule must periodically repeat a scene. Rather than repeat the scene descriptions in the schedule, it would be simpler to describe the scene in one place and have the schedule make references to it. The virtual motion profile that is repeated for each run is also easier to reference rather than repeat. The scene and profile, therefore, are components that can be combined to describe a specific run.

This module-based framework makes it simpler not only to schedule a set of conditions, but also to modify an existing condition without making drastic changes to the schedule itself. Suppose, for example, after creating a creating a schedule, you decide to modify one of the scenes; a Monet makes a more interesting impression than the picture of the space-shuttle launch. With a modular design, you would only need to change the scene description in one place, and the entire schedule is changed automatically. Without a modular design, however, you would need to go through the entire schedule, changing the

room description each time it appears.

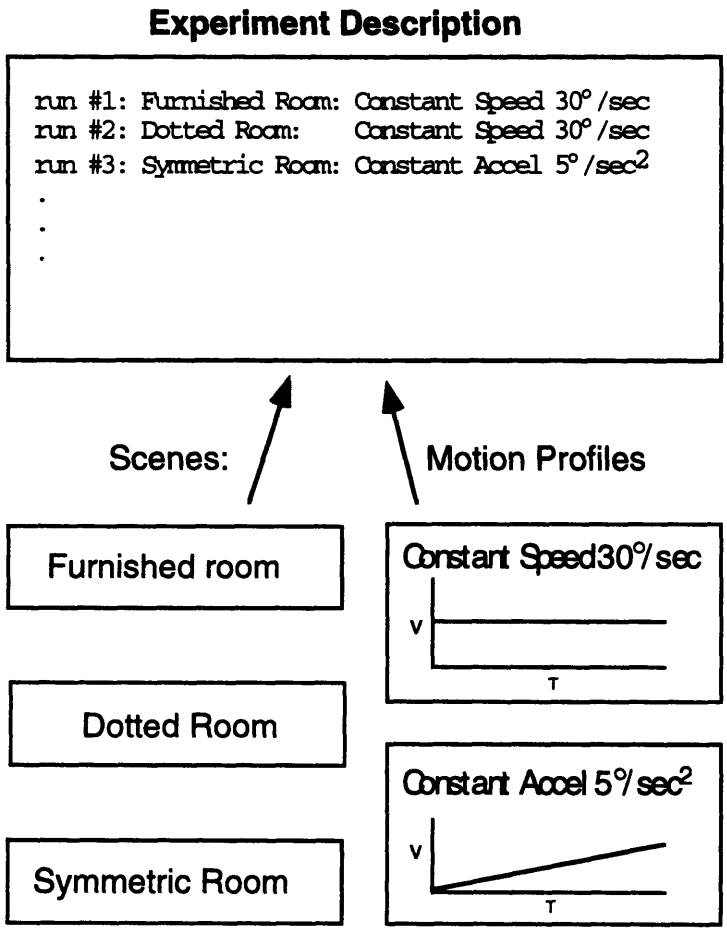

Figure 10-1: Modular Experiment Description.

The scheduling is specified using a set of text files. Scenery, for example, is described in one file, and velocity-profiles in another. The scheduling itself is placed in its own file. The framework, shown in Figure 10-1, can therefore be easily altered with the use of a text-editor. In the event that the experiment needs to be quickly redesigned (i.e., rapid experiment prototyping or mid-flight mission alterations), it can be done with minimal knowledge of how the experiment software is written.

There are, however, some drawbacks with the implementations. Ideally, the experiment-designer would want to specify all characteristics of the scene in one file.

Besides the visual description, these characteristics include information such as expected frame-rate. The nature of Sense8's World Toolkit and our hardware, however, make it impossible to completely control frame-rate. The system is free-running, meaning that it cycles as fast as it can through rendering and other tasks. Frame-rate, therefore, varies slightly throughout the experiment.

When a large number of textures become visible on the head-mounted-display's screens, it causes rendering time to increase, frame-rate to decrease, the scene rotation to slow, and the subject to experience virtual acceleration. How this effects immersion and fidelity is not completely known. Typically, vestibular excitation accompanies acceleration, so when a subject is virtually accelerated without the appropriate confirmation it causes a sensory conflict that may reduce the realism of the virtual environment. One subject reported saturated vection while the scene was slowing down, because, for a brief moment, he perceived that the room had stopped rotating so all his apparent velocity was due only to his rolling. Various ways to minimize the variation are being researched, but current hardware makes this difficult.

10.1.3 Modules

The software package is made up of 6 modules. The **Menu Module** handles the user-interface and sends commands to all the other modules. The **Subject Module** handles all access to the subject database and keeps track of what each subject has been through already. Experiment control is handled by the **Experiment Module**. This module determines the set of conditions to test, given a subjects treatment type, and it sends this information to the **Run Module** which runs an experiment by interacting with a virtual-reality tool kit such as Sense8's WTK. To adapt the software to another tool kit, one would only need to rewrite the **Run Module** to use it. A **Movement Module** implements

programmed velocity profiles. The software uses the **Data Module** to handle all storage, retrieval and conversion of data.

10.2 Using the Software: Instructions

The user-interface is menu driven, and Figure 10-2 shows the menu hierarchy. A human controller loads in a experiment description and performs various tasks including performing experiments, manipulating subject information, and exporting of data into text format.

To perform experiments, the controller must first select an existing subject or create new one using the subject database. To support Latin-square designs, subjects are given a treatment number that specifies where in the schedule the subject begins her experiment. Future versions of the software will let the user rapidly design experiment schedules using a powerful **Create Experiment** module. Currently, schedules must be manually entered using a text-editor.

The options in the **Data Menu** allow one to export data into a text format that is readable by most spreadsheet packages. This feature is currently limited to producing reports of a specific format, but future versions will support user-definable formats.

10.3 Source Code

The source code is listed in the following sections, starting with makefiles, continuing with the headers (*.h), and ending with source (*.c) files. The code was compiled on a Packard Bell Legend (90Mhz Pentium), using the WTK 2.0 virtual reality tool kit, High C development system, and the Phar-Lap DOS extensions.

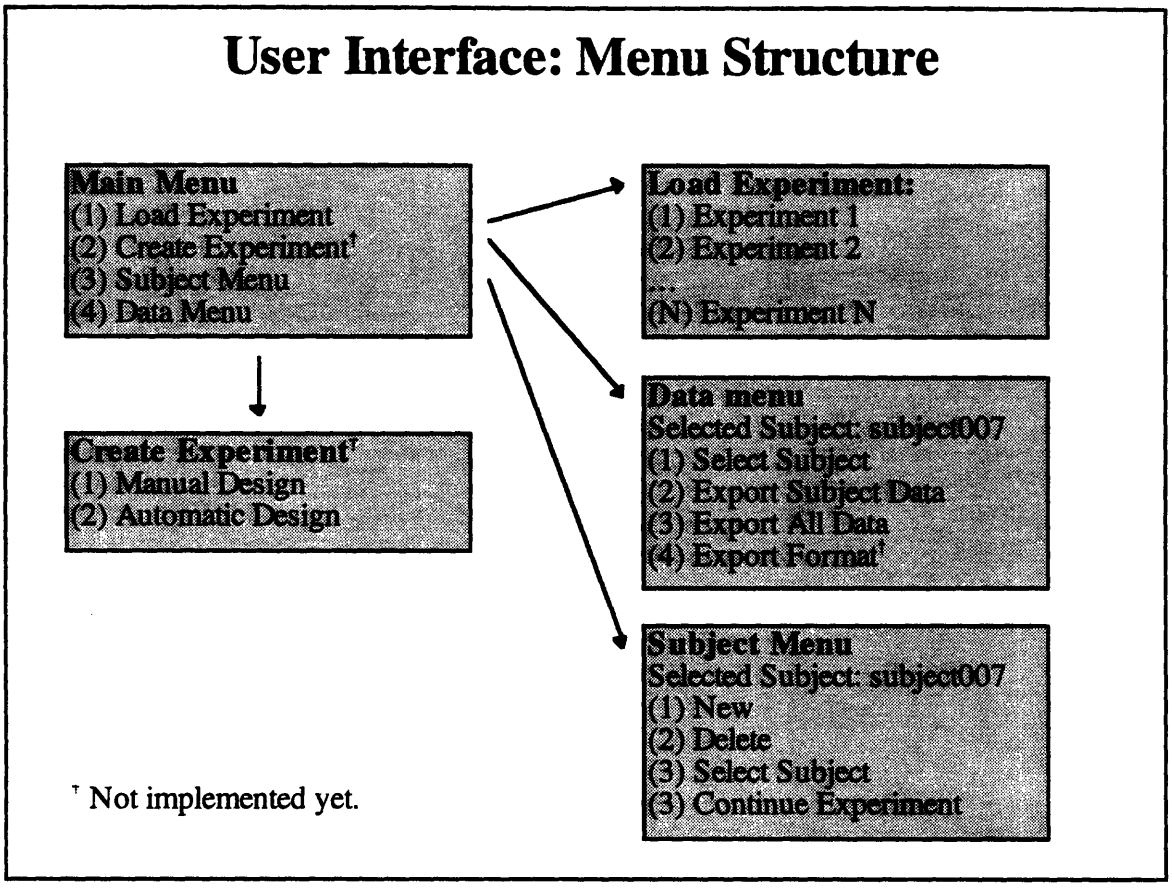

Figure 10-2: Experimental Software User Interface.

10.3.1 central.mak

```
#Makefile to build central.exe : the experiment control software.

SRC    = .
OBJ    =

OBJECTS =   ${OBJ}statmod.obj \
      ${OBJ}subjdb.obj \
      ${OBJ}runexp.obj \
      ${OBJ}joyutil.obj \
      ${OBJ}spin.obj \
      ${OBJ}myprint.obj \
      ${OBJ}fplan.obj \
      ${OBJ}move.obj \
      ${OBJ}expmod.obj \
      ${OBJ}datarec.obj \
      ${OBJ}dataread.obj \
      ${OBJ}track.obj \
      ${OBJ}joysens.obj \
      ${OBJ}dirutils.obj
```

```
#     link any other wtk demo

all: central.exe

central.exe: ${OBJECTS}
      386link -cvsymbols @central.lnk -EXE central.exe

.c.obj: joysens.h datarec.h central.mak
      hc386 -c -g -586 $*.c -Hpro=\wtk860\metaware.pro
```

10.3.2 central.lnk

```
! This link file is for MetaWare version 3.x compiler.
!
! This file is used by MAKE.BAT to link WTK.EXP using the Phar Lap
! Version 5.0 386|Link program
! The resulting protected mode executable WTK.EXP can be run under
! the Phar Lap 386|DOS-Extender using the RUN386.EXE command
!
subjdb.obj runexp.obj joyutil.obj spin.obj myprint.obj fplan.obj dataread.obj
move.obj expmod.obj datarec.obj track.obj joysens.obj dirutils.obj statmod.obj
-exeformat pe
-markphar
-subsystem dosstyle
-defstubname gotnt
-pebias 2000h
-STACK 131072
!-OFFSET 4000h
-maxdata 200000h
-twocase
-LIB \wtk860\lib\wtkspea
-LIB \wtk860\lib\libsp3dp
-LIB \wtk860\lib\libspp
-LIB \wtk860\lib\hcmlibp
-LIB \highc\small\HC586
-LIB \highc\small\HC387
-LIB \highc\small\HCNA
-attributes class CODE er
-attributes group CGROUP er
-attributes class DATA rw
-attributes class CONST rw
-attributes class BSS rw
-attributes class STACK rw
-attributes group DGROUP rw
```

10.3.3 subjdb.h

```
#ifndef _H_SUBJDB
#define _H_SUBJDB

#include"runexp.h"

#define SUBJDB_MAXWDLEN 100
typedef struct {
      int   numTreats;
```

```
      char  expName[50];
      char  expDir[SUBJDB_MAXWDLEN];
      TreatmentInfoRec *Treatments;
} ExpInfoRec;

#define SUBJDB_MAXNAMELEN 40
#define SUBJDB_MAXTREATMENTS 16

typedef struct {
      char  subjName[SUBJDB_MAXNAMELEN];
      int   treatment;
      int         finished;
/*
   int      gender;
   int  eyedata;
   etc...
*/
} SubjectInfo;

#define MAX_SUBJS 40
typedef struct {
      int   numSubjs;
      int   treatmentBound[SUBJDB_MAXTREATMENTS];
      SubjectInfo Subjects[MAX_SUBJS];
} SubjectListRec;

void subjdbMain(void);


void subjdbReadSubjectInfo(FILE *subfile,SubjectInfo *aSub);
int subjdbReadAllSubjs(void);
void subjdbWriteSubjects(void);
void subjdbWriteSubjectInfo(FILE *subfile, SubjectInfo *aSub);
void subjdbNewSubject(void);
void subjdbInitialize(void);
void subjdbMainMenu(void);
void subjdbSubjectMenu(void);
void subjdbStatMenu(void);
void subjdbPrintMenu(int which);
void subjdbLoadExperiment(void);
void subjdbNewExperiment(void);
void subjdbSaveExpInfo(void);
void subjdbDeleteSubject(void);
void subjdbContinueSubject(void);
void subjdbLoadSubjectList(void);
void subjdbResetSubject(void);
void subjdbPrintAllSubjects(void);
void subjdbChooseSubject();
void subjdbUpdateFinished(SubjectInfo *a);
void subjdb_PerformStatsOnChosen(void);
#endif
```

10.3.4 expmod.h

```
#ifndef _H_EXPMOD
#define _H_EXPMOD
#include "subjdb.h"
#ifndef FNAMESIZE
```

```
 #define  FNAMESIZE 80
#endif
void expmod_main(SubjectInfo *SInfo);
void expmod_setconfigfile(char *configfile);
int expmod_getnumtreats(void);

#endif
```

10.3.5 runexp.h

```
#ifndef _H_RUNEXP
#define _H_RUNEXP

#ifndef  FNAMESIZE
 #define  FNAMESIZE 80
#endif

enum {
       POSTURE_ERECT=1,
       POSTURE_SUPINE
};


typedef struct {
      char fplanfile[FNAMESIZE];
      char scenefile[FNAMESIZE];
      char triggfile[FNAMESIZE];
      char triggdir[FNAMESIZE];
      char treatcode[FNAMESIZE];
      float expected_fps;
      int  direction;
      int  posture;
} TreatmentInfoRec;

void runexp(TreatmentInfoRec *info,char *subname);
#endif
```

10.3.6 fplan.h

```
#define FPLAN_MAX_PARAMS 4

typedef struct {
      int   type;             /*Type of movement*/
      int   NumUps;                        /*Expected number of updates*/
      float p[FPLAN_MAX_PARAMS];    /*parameters for movement*/
} movement_rec;

extern int fplan_num_movements,all_done;
extern movement_rec *FlightPlan;
extern fplan_actual_updates;

#define GetNM()   fplan_num_movements
#define FP(i)     FlightPlan[i]
#define fplan_NumUps(i) FP[i].NumUps
#define fplan_isdone() all_done
#define fplan_get_actualupdates() fplan_actual_updates
```

```
/*first line is for featured, second is featureless scale 2, third
 featureless scaled 1*/

extern float EXPECTED_FPS;

/*#define EXPECTED_FPS 19.4*/
/*#define EXPECTED_FPS 12.5*/
/*#define EXPECTED_FPS 15.0*/

void  fplan_load(char *name);
void  fplan_picknload(void);
void  fplan_read_params(int numparams);
void  fplan_read_next_movement(void);
void  fplan_start_next_movement(void);
void  fplan_exit(void);
int   fplan_expups(void);
void  fplan_domove(void);
void  fplan_setexpectedfps(float a);
```

10.3.7 move.h

```
#include "wt.h"

#define DIRECTION_TOO 1

void  move_init(void);
int   move_isdone(void);
float move_getangle(void);
int   move_getrot_count(void);
void  move_setobject(WTobject *anobject);
void  move_accelerate(void);
void  move_jerk(void);
void  move_domove(void);
void  move_reset(void);
void    move_const_velocity(float v,float t);
void  move_const_accel(float v,float a,float t);
void  move_const_jerk(float a, float v, float j, float t);
void  move_getkinetics(float *v, float *a, float *j);
void  move_setrotdirection(int rot);

enum {
      DIRECTION_CW=1,
      DIRECTION_CCW
};
```

10.3.8 datarec.h

```
/*datarec.h*/

/*RECTYPE will probably also include rotation angle*/
#ifndef _H_DATAREC
#define _H_DATAREC
#include <time.h>
typedef     struct {
      clock_t time;
      float angle;
```

```
} StampRec,*StampPtr,**StampHandle;

enum {
      data_rec_trigger1,
      data_rec_trigger2
};

#define RECTYPE StampRec
#define RECTYPE2 RECTYPE
#define HedRec int

int     datarec_openexp(char *expname,int maxentries);
int   datarec_closeexp(void);
int   datarec_recdata(char *info,int which);
#endif
```

10.3.9 dataread.h

```
#ifndef _H_DATAREAD
#define _H_DATAREAD

#define GetDNE() data_numentries
#define GetDNE2() data_numentries2
#define TimeData(i) dataread[(i)].time
#define AngleData(i) dataread[(i)].angle
#define TimeData2(i) dataread2[(i)].time
#define AngleData2(i) dataread2[(i)].angle
#define StampData(i) dataread[(i)]
#define StampData2(i) dataread2[(i)]

int   openexpread(char *name);
int   closeexpread(void);


extern RECTYPE *dataread;
extern RECTYPE *dataread2;
extern int data_numentries,data_numentries2;
extern int data_numexps;
#endif
```

10.3.10 joysense.h

```
/* joysens.h
   joystick sensor update function detects button presses
*/

extern void WTjoystick_detbutt(WTsensor *sensor);
```

10.3.11 statmod.h

```
#ifndef _H_STATMOD
#define _H_STATMOD 1
#include "runexp.h"

void  printstats(char *expname);
void  performstats_1(void);
```

```
void  performstats2_1(void);
void  statmod_init(void);
#ifdef INDEPENDENT
int   performstats(char *name);
#else
int   performstats(char *name,TreatmentInfoRec *arec);
#endif
void  do_analysis(char *name);
void  statmod_init(void);
#endif
```

10.3.12 dirutils.h

```
#ifndef _H_DIRUTILS
#define _H_DIRUTILS
#include <dirent.h>

int   SetDirectory(char *dirname);
void  ListDirectory(int withnumbers);
struct _dirent *GetNthEntry(int N);
int   GetNthEntryName(int N,char *name);
void dirwalk(char *dir, void (*fcn) (char *));
#endif
```

10.3.13 myprint.h

```
#ifndef _H_MYPRINT
#define _H_MYPRINT

#include <stdarg.h>

/*#define ADAM_DEBUGGING 1*/
int myprint(char *fmt, ...);

#endif
```

10.3.14 subjdb.c

```
#include"wt.h"
#include<stdio.h>
#include<stdlib.h>
#include<errno.h>
#include<direct.h>
#include<string.h>
#include"subjdb.h"
#include<sys\types.h>
#include<sys\stat.h>
#include"dirutils.h"
#include"expmod.h"
#include"statmod.h"


#define GET_CHAR newgetchar

enum {
      ksubjmenuMAIN,
```

```
      ksubjmenuSUBJECT,
      ksubjmenuNEWSUBJ,
      ksubjmenuDELETE,
      ksubjmenuCONTMENU,
      ksubjmenuSTATMENU,
      ksubjmenuRESETSUBJ,
      ksubjmenuCHOOSESUBJ
};

extern TreatmentInfoRec Treatments[];
static char moduleName[25]="Subject Database";

/*subjdb    globals                                  */

extern char             SubjectDir[80];
static int        subjectsLoaded;
static SubjectListRec   subjdbAllSubjects;
static ExpInfoRec subjdbExpInfo;
static int        subjdb_initialized=0;
static char             subjdbExpFileName[1024];
static int        theChosen=0;
char              SubjectDir[80];

/*Syntactic Sugar*/
#define subjects(j)     (subjdbAllSubjects.Subjects[(j)])
#define num_subjs       subjdbAllSubjects.numSubjs
#define subjname  ((theChosen) ? subjects(theChosen-1).subjName : "a subject.")

/*Decls*/
void DoQuit(void);
char newgetchar(void);
void subjdbInit(void);


/* a dirwalk function for finding all the cfg files */

#define MAX_CFGS 20
char cfg_files[MAX_CFGS][128];
int numcfgfiles=0;

void findcfg(char *name){
      if (strstr(name,".CFG"))
            strcpy(cfg_files[numcfgfiles++],name);
}

/*    For Choosing Subjects         */



void main(int argc,char *argv[]){
      subjdbMain();
}

void subjdbMain(void){
      /*Main entry point into this module       */
      /*perform any initialization if necessary */
      if (!subjdb_initialized)
            subjdbInitialize();
```

```
        subjdbMainMenu();
        /*This is the last you should do when running a WTKapp*/
        printf("does this get run?\n");
        WTuniverse_delete();
}

void subjdbInitialize(void){
        /*Create the universe to be used*/
        WTuniverse_new(WTDISPLAY_STEREO, WIWINDOW_DEFAULT);
        subjectsLoaded=0;
        subjdbAllSubjects.numSubjs=0;
        subjdbExpInfo.numTreats=0;
        subjdbExpInfo.expName[0]=0;
        subjdbExpInfo.expDir[0]=0;
        subjdb_initialized=1;
}

/* Menus */
void subjdbMainMenu(){
        char choice=0;
        while (1) {
                subjdbPrintMenu(ksubjmenuMAIN);
                choice = GET_CHAR();
                switch (choice) {
                    case 'N':
                    case 'n':
                      subjdbNewExperiment();
                      break;
                    case 'L':
                    case 'l':
                      subjdbLoadExperiment();
                      break;
                    case 'S':
                    case 's':
                      subjdbSubjectMenu();
                      break;
                    case 'T':
                    case 't':
                      subjdbStatMenu();
                      break;
                    case 'Q':
                    case 'q':
                      DoQuit();
                      break;
                    default:
                      break;
                }
        }
}

void subjdbSubjectMenu(){
        char choice=0;
        while ((choice!='X') && (choice!='x')){
                subjdbPrintMenu(ksubjmenuSUBJECT);
                choice = GET_CHAR();
                switch (choice){
                    case 'D':
                    case 'd':
```

```
                subjdbDeleteSubject();
                break;
            case 'C':
            case 'c':
                subjdbContinueSubject();
                break;
            case 'N':
            case 'n':
                subjdbNewSubject();
                break;
            case 'R':
            case 'r':
                subjdbResetSubject();
                break;
            case 'P':
            case 'p':
                subjdbChooseSubject();
                break;
            default:
                break;
        }
    }
}


void subjdbStatMenu(){
    char choice=0;
    while ((choice!='X') && (choice!='x')){
        subjdbPrintMenu(ksubjmenuSTATMENU);
        choice = GET_CHAR();
        switch(choice){
            case 'C':
            case 'c':
                subjdbChooseSubject();
                break;
            case 'S':
            case 's':
                if (!theChosen){
                    subjdbChooseSubject();
                }
                if (theChosen){
                    subjdb_PerformStatsOnChosen();
                }
                break;
            case 'A':
            case 'a':
                {   /*ALL*/
                    int savedChosen=theChosen;
                    subjdbLoadSubjectList();
                    for (theChosen=1;theChosen<=num_subjs;theChosen++)
                        subjdb_PerformStatsOnChosen();
                    theChosen=savedChosen;
                }
                break;
            default:
                break;
        }
    }
```

```
}

void subjdbPrintMenu(int which){
      switch (which) {
          case ksubjmenuMAIN:
            printf("%s: Main Experiment Menu\n",moduleName);
            printf("Current Experiment is %s\n",subjdbExpInfo.expName);
            printf("(N)ew Experiment\n");
            printf("(L)oad other Experiment\n");
            printf("(S)ubject menu\n");
            printf("s(T)atistics menu\n");
            printf("(Q)uit\n");
            break;
          case ksubjmenuSUBJECT:
            printf("%s: Subject Menu\n",moduleName);
            if (theChosen) printf("Chosen subject %s\n",subjects(theChosen-
1).subjName);
                  printf("(N)ew subject\n");
            printf("(D)elete %s\n", subjname);
            printf("(C)ontinue/Begin testing %s\n",subjname);
            printf("(R)eset %s\n",subjname);
            printf("(P)ick a subject.\n");
            printf("e(X)it, or back.\n");
            break;
          case ksubjmenuNEWSUBJ:
            printf("%s: New Subject Menu\n",moduleName);
            printf("(1-%d) Treatment type.\n",subjdbExpInfo.numTreats);
            printf("or 0 to exit back.");
            break;
          case ksubjmenuRESETSUBJ:
            printf("%s: Reset which subject?\n(0 to cancel)",moduleName);
            subjdbPrintAllSubjects();
            break;
          case ksubjmenuDELETE:
            printf("%s: Delete which subject?\n(0 to cancel)",moduleName);
            subjdbPrintAllSubjects();
            break;
          case ksubjmenuCONTMENU:
            printf("%s: Continue/Begin testing which subject?(0 to cancel)\n",
                  moduleName);
            subjdbPrintAllSubjects();
            break;
          case ksubjmenuCHOOSESUBJ:
            printf("%s: Choose a subject.\n",moduleName);
            subjdbPrintAllSubjects();
            break;
          case ksubjmenuSTATMENU:
            printf("%s: Stats Menu\n",moduleName);
            if (theChosen) {
                  printf("Chosen subject %s\n",subjects(theChosen-1).subjName);
            }
            printf("Perform (S)tats on %s.\n",subjname);
            printf("(C)hoose a subject.\n");
            printf("Perform Stats on (A)ll subjects.\n");
            printf("e(X)it\n");
            break;
          default:
            printf("%s: Unknown Menu", moduleName);
```

```
            break;
      }
}


void subjdbLoadExperiment(void){
      char  *cwd=NULL;
      int   whichexp;
      int   i;
      numcfgfiles = 0;
      cwd = getcwd(NULL,SUBJDB_MAXWDLEN);
      dirwalk(cwd,findcfg);
      if (numcfgfiles) {
            printf("Choose an experiment...\n");
again:            for (i=1;i<=numcfgfiles;i++)
                  printf("(%d) %s\n",i,cfg_files[i-1]);
            scanf("%d",&whichexp);
            if (whichexp<=numcfgfiles) {
                  strcpy(subjdbExpFileName,cfg_files[whichexp-1]);
                  expmod_setconfigfile(subjdbExpFileName);

            }
            else {
                  printf("You must choose a number between 1 and %d\n",
                        numcfgfiles);
                  goto again;
            }
      }
      else {
            printf("There are no experiments in this directory\n");
            printf("You must first create a new one.\n");
            return;
      }
      if (cwd) free(cwd);
}

void subjdbNewExperiment(void){
      /*not implemented yet*/
      printf("Not yet implemented\n");
}


void subjdbSaveExpInfo(void){
      /*not implemented yet*/
      printf("This isnt implemented yet.\n");
}

void subjdbDeleteSubject(void){
      int i,start,last;
      if (!theChosen) {
            printf("You have not chosen a subject to delete\n");
            return;
      }
      printf("Are you sure you want to delete %s(Y/N)?\n",subjname);
      start = theChosen;
      last = num_subjs--;
      for (i=start;i<last;i++)
```

```
            subjects(i-1) = subjects(i);
      subjdbWriteSubjects();
}

void subjdbChooseSubject(){
      /*entry point for choosing a subject*/
      theChosen = choosesubject(ksubjmenuCHOOSESUBJ);
}

void subjdbContinueSubject(){
      if (theChosen==0)
            theChosen = choosesubject(ksubjmenuCONTMENU);
      if (theChosen>0) {
            expmod_main(&subjects(theChosen-1));
      }

}

void subjdbUpdateFinished(SubjectInfo *subjinfo){
      subjects(theChosen-1).finished = subjinfo->finished;
      subjdbWriteSubjects();
}

void subjdbResetSubject(){
      if (theChosen ==0)
            theChosen= choosesubject(ksubjmenuRESETSUBJ);
      subjects(theChosen-1).finished=0;
      subjdbUpdateFinished(&subjects(theChosen-1));
      printf("%s has been reset.\n",subjects(theChosen-1).subjName);

}
/* Subject Lists */

void subjdbLoadSubjectList(void){
      /*Loads a list of ALL test subjects (for a given
        experiment into an indexed datastructure
        for easy retrieval. Information loaded
        is only administrative-related info. It wont
        load in actual test results, only whether a
        user has completed a certain test and other
        relevant info. */

      /*make a quick exit if already loaded subjects, and
        delegate the work to another function*/

      if (!subjectsLoaded) subjectsLoaded = subjdbReadAllSubjs();
}

int   subjdbReadAllSubjs(void){
      printf("Readling all Subjs\n");
      char fname[512];
      FILE *subjfile=NULL;
      strcpy(fname,SubjectDir);
      strcat(fname,"subject.lst");

      printf("looking for subjects in %s\n",fname);
      if (subjfile = fopen(fname,"r")){
            int i;
```

```
            fscanf(subjfile,"%d",&subjdbAllSubjects.numSubjs);
            /*make sure num_subjs is less than maximum allowable*/
            num_subjs = (num_subjs<MAX_SUBJS) ? num_subjs : MAX_SUBJS;
            printf("Num subjects is %d\n",num_subjs);
            for (i=0;i<subjdbAllSubjects.numSubjs;i++)
                  subjdbReadSubjectInfo(subjfile,&subjdbAllSubjects.Subjects[i]);
            fclose(subjfile);
            return 1;
      }
      else {
            printf("Couldnt open %s\n",fname);
            return 0;
      }
}

void  subjdbReadSubjectInfo(FILE *subfile,SubjectInfo *aSub){
      /*read subject info into the subject record pointed to
        by *aSub. */
      fscanf(subfile,"%s",aSub->subjName);
      fscanf(subfile,"%d",&aSub->treatment);
      fscanf(subfile,"%d",&aSub->finished);
}

void subjdbWriteSubjects(void){
      char fname[512];
      FILE *subjfile=NULL;
      strcpy(fname,SubjectDir);
      strcat(fname,"subject.lst");


      if (subjfile = fopen(fname,"w")){
            int i;
            fprintf(subjfile,"%d\n",subjdbAllSubjects.numSubjs);
            for (i=0;i<subjdbAllSubjects.numSubjs;i++)
                  subjdbWriteSubjectInfo(subjfile,&subjdbAllSubjects.Subjects[i]);
            fclose(subjfile);
      }
}

void  subjdbWriteSubjectInfo(FILE *subfile,SubjectInfo *aSub){
      /*write subject info from the subject record pointed to
      by *aSub. */
      fprintf(subfile,"%s\n",aSub->subjName);
      fprintf(subfile,"%d\n",aSub->treatment);
      fprintf(subfile,"%d\n",aSub->finished);

}

void subjdbPrintAllSubjects(void){
      /*reads in all subjects into DB if necessary and then
        print out a list of them, with a number prepended so
        that the user can choose that subject. The number
        prepended corresponds to the index+1 in the loaded
        data structure. */
      int i;
      printf("Printing subjects, num_subjs = %d, subjectsLoades= %d\n",
            num_subjs,subjectsLoaded);
      if (!subjectsLoaded)
```

```
            subjdbLoadSubjectList();
      for (i=1;i<=num_subjs;i++)
            printf("(%2d) %8s(%1d,%1d)\n",i,subjects(i-1).subjName,
                  subjects(i-1).treatment,subjects(i-1).finished);

}

void subjdbNewSubject(void){
      char fname[512];
      SubjectInfo newsubj;
      if (num_subjs == MAX_SUBJS) {
            printf("Sorry, I dont have any room for more subjects.\n");
            printf("Complain to the author!\n");
            return;
      }
      subjdbLoadSubjectList();
      printf("Subject's name: ");
      scanf("%s",newsubj.subjName);
retry:      printf("%s's treatment(1-%d): ",newsubj.subjName,
             expmod_getnumtreats());
      scanf("%d",&newsubj.treatment);
      if (newsubj.treatment>expmod_getnumtreats() ||
          newsubj.treatment<1){
            printf("You must choose a value between 1-%d.\n",
                  expmod_getnumtreats());
            goto retry;
      }
      printf("New subject, %s, treatment %d, created.\n",
             newsubj.subjName, newsubj.treatment);
      newsubj.finished = 0;
      subjdbAllSubjects.Subjects[num_subjs++]=newsubj;
      /*NB: NEED TO CREATE A FOLDER IF NECESSARY!*/
      strcpy(fname,SubjectDir);
      strcat(fname,newsubj.subjName);
      mkdir(fname);
      subjdbWriteSubjects();
}

void subjdb_PerformStatsOnChosen(void){
      /*Create Text Data files corresponding to the
        subject specified by "theChosen" */

      int i;
      char temp[FNAMESIZE],expname[FNAMESIZE];
      printf("Doing Stats on %s...",subjname);
      for (i=1;i<=24;i++){
            strcpy(temp,SubjectDir);
            strcat(temp,subjname);
            temp[strlen(temp)+1]=0;
            temp[strlen(temp)]='\\';
            sprintf(expname,"%sf%d.dat",temp,i);
            statmod_init();
            if (!performstats(expname,&Treatments[i-1]))
                  printf("Could not open %s\n");

      }
      printf("...done.\n");
}
```

```
void DoQuit(void){
      /*If we need to do anything before quitting,
        do it here*/

      /*...nothing to do...*/
}

/*This picks the first character off a line*/

char newgetchar(void){
      char line[80];
      int i=0;
      scanf("%s",line);
      while (i<80 && (line[i]==' ' || line[i++]=='\t'));
      return (line[i-1]);
}

int  choosesubject(int menu){
      int choice;
      char line[80];
retry:      subjdbPrintMenu(menu);
      scanf("%s",line);
      choice = atoi(line);
      if (choice<0 || choice>num_subjs){
            printf("You must choose a number between 0-%d\n",
                  num_subjs);
            goto retry;
      }
      printf("You chose %d\n",choice);
      return choice;
}
```

10.3.15 expmod.c

```
/*Experiment Module: Entry point into runexp*/
#include "wt.h"
#include<stdio.h>
#include "runexp.h"
#include "expmod.h"
#include "move.h"
#include <direct.h>
#include "myprint.h"
#define NUM_TREATS 100
/*#define INDEPENDENT 1*/
int noisy = 0;

void loadexperimentinfo(void);
void expmod_readtreatment(FILE *tfile,TreatmentInfoRec *Treat);
void doexperimentmenu(void);
void load_configuration(char *config);
void makeifnecessary(char *name);
void donexttreatment(void);

TreatmentInfoRec Treatments[NUM_TREATS];
static int  NumTreats;
static char ExpFileName[80];
static char       SceneDir[80];
```

```
extern char SubjectDir[80];
static char ProfileDir[80];
static  char      cwd[100];
static  char      itsConfigFile[1024];

void expmod_setconfigfile(char *afile){
      strcpy(itsConfigFile,afile);
      load_configuration(afile);
      loadexperimentinfo();
}
int expmod_getnumtreats(){
      return NumTreats;
}

SubjectInfo itsSInfo;
#ifdef INDEPENDENT
void main(int argc, char *argv[]){

#else
void expmod_main(SubjectInfo *SInfo){
      if (SInfo) itsSInfo = *SInfo;
      else return;
#endif
/*load configuration*/

#ifdef INDEPENDENT
      load_configuration("vectexp.cfg");
#else
      load_configuration(itsConfigFile);
#endif

#ifdef INDEPENDENT
      WTuniverse_new(WTDISPLAY_STEREO, WTWINDOW_DEFAULT);
#endif
        loadexperimentinfo();

/*Do the treatment*/
#ifdef  INDPENDENT
        doexperimentmenu();
#else
      donexttreatment();
#endif

/*cleanup*/
#ifdef INDEPENDENT
      WTuniverse_delete();
#endif

}

void  makeifnecessary(char *name){
      char  testdir[120];
      strcpy(testdir,cwd);          /*copy wd into a string*/
      strcat(testdir,name);    /*append the dirname to check*/
      testdir[strlen(testdir)-1]=0; /*remove the last '\'*/
      mkdir(testdir);               /*make it if need to*/
}
```

```
void expmod_readtreatment(FILE *tfile,TreatmentInfoRec *Treat){

      strcpy(Treat->fplanfile,ProfileDir);
      fscanf(tfile,"%s",Treat->fplanfile + strlen(ProfileDir) );
      myprint("fplan file is in %s\n",Treat->fplanfile);
      strcpy(Treat->scenefile,SceneDir);
      fscanf(tfile,"%s",Treat->scenefile + strlen(SceneDir));
      strcpy(Treat->triggdir,SubjectDir);
      fscanf(tfile,"%s",Treat->triggfile);
      fscanf(tfile,"%s",Treat->treatcode);
      fscanf(tfile,"%f",&Treat->expected_fps);

#ifdef DIRECTION_TOO
      fscanf(tfile,"%d",&Treat->direction);
#endif
      fscanf(tfile,"%d",&Treat->posture);
      /*Create the directories if they dont already exist*/

      makeifnecessary(ProfileDir);
      makeifnecessary(SceneDir);
      makeifnecessary(SubjectDir);
}

void loadexperimentinfo(void){
      FILE *expfile=NULL;
      int   cnt;


      if (expfile = fopen(ExpFileName,"r")){
            fscanf(expfile,"%d",&NumTreats);
            printf("Number of treatments: %d\n",NumTreats);
            for (cnt=0;cnt<NumTreats;cnt++)
                  expmod_readtreatment(expfile,&Treatments[cnt]);
            fclose(expfile);
      }
}

void  doexperimentmenu(){
      unsigned int treatment=0,cnt;
/*    noisy = 1;*/
#if   INDEPENDENT
      while (treatment<=NumTreats){
            printf("Choose a treatment\n");
            for (cnt=1;cnt<=NumTreats;cnt++)
                  printf("(%d) %s\n",cnt,Treatments[cnt-1].treatcode);
            printf("(%d) Exit\n",NumTreats+1);
            scanf("%d",&treatment);
            if (treatment<=NumTreats)
                  runexp(&Treatments[treatment-1],"janedoe");
            /*runexp(&Treatments[treatment-1],itsSInfo.subjName);*/
      }
#else
      /*code to choose appropriate experiment to run*/
      donexttreatment();
#endif
}
```

```
void load_configuration(char *config){
      FILE *cfgfile=NULL;

      /*cfgfile = fopen("vectexp.cfg","r");*/
      cfgfile = fopen(config,"r");
      if (cfgfile){
            fscanf(cfgfile,"%s",SceneDir);
            fscanf(cfgfile,"%s",ProfileDir);
            fscanf(cfgfile,"%s",SubjectDir);
            fscanf(cfgfile,"%s",ExpFileName);
            getcwd(cwd,100);
            cwd[strlen(cwd)+1]=0;
            cwd[strlen(cwd)]='\\';
      }
      else {
            printf("Error: Could not open configuration file.\nIs \"vectexp.cfg\"
            really there?\n");
            exit(0);
      }
}

void donexttreatment(void){
      int next;
      char s[10];
      next = ((itsSInfo.treatment-1)+(itsSInfo.finished)) % NumTreats;
      printf("Now doing experiment %d of treatment %d (finished %d)\n",
            next+1,itsSInfo.treatment,itsSInfo.finished);
      printf("Subject's posture should be %s\n",
            (Treatments[next].posture==POSTURE_ERECT)?"ERECT":"SUPINE");
      printf("Type 'G' to begin or 'X' to go back (and RETURN)\n");
      itsSInfo.finished++;
      scanf("%10s",s);

      if (s[0]!='X' && s[0]!='x'){
            runexp(&Treatments[next],itsSInfo.subjName);
            subjdbUpdateFinished(&itsSInfo);
      }
}
```

10.3.16 runexp.c

```
/*runexp.c*/

#include "wt.h"
#include <strings.h>
#include <stdio.h>
#include <time.h>

#include "runexp.h"
#include "joysens.h"
#include "spea2d.h"
#include <time.h>
#include "datarec.h"
#include "fplan.h"
#include "move.h"
#include "myprint.h"
```

```
#define NUMUPS    886
/*#define CHANGEORDER 1*/
void spin(WTobject *object);
void SetupSpin(int howmany);
void CleanupSpin(void);
static void display_menu();
static void setparameters();
void draw2d(short eye);
void toggle_joystick();
void record_joystick();
FILE *infile = NULL;
FILE *outfile = NULL;

int num_rotation, num_tests, num_rotupdate;
char rotate_axis = Y;


WTpq lobbyview;
extern WTpq *spinView;
WTobject *lobby = NULL;
extern WTobject *spinObject;
extern int noisy;
/* frame rate checker */
static clock_t currtime;
static clock_t prevtime;
extern char       SceneDir[80];
extern char SubjectDir[80];
extern char ProfileDir[80];


/*joystic stuff*/

void benchit();
void (*actionfn) ();
void WTuniverse_setactions(actionfn);
void datastamp();
TreatmentInfoRec TInfo;

void runexp(TreatmentInfoRec *Info, char *subjectname)
{

      WTp3 centre;
      FILE *infile = NULL;
      int rend;
      short myevents[3];
      float fr,totime;
      char temp[90];
      myevents[1]=WTEVENT_ACTIONS;
      myevents[0]=WTEVENT_OBJECTSENSOR;
      myevents[2]=WTEVENT_TASKS;



      if (Info) TInfo = *Info;
      else return;

      printf("Fplanfile is %s\t",TInfo.fplanfile);
      printf("scenefile is %s\n",TInfo.scenefile);
```

```
      strcpy(temp,TInfo.triggdir);
      strcat(temp,subjectname);
      temp[strlen(temp)+1]=0;
      temp[strlen(temp)]='\\';
      strcat(temp,TInfo.triggfile);
      strcpy(TInfo.triggfile,temp);
      printf("Trigg file is %s\n",TInfo.triggfile);
      printf("Treatment Code is %s\n",TInfo.treatcode);
      printf("Expected FPS is %f\n",TInfo.expected_fps);
#ifdef DIRECTION_TOO
      printf("Direction is %s\n", (TInfo.direction==DIRECTION_CW)?"CW ":"CCW");
      move_setrotdirection(TInfo.direction);
#endif
      /*Initialize the universe, name it rotate, and set background to black*/
      /*WTdraw_setactions(draw2d);*/
      /*WTuniverse_new(WTDISPLAY_STEREO, WTWINDOW_DEFAULT);*/
      WTuniverse_setname("rotate");
      WTuniverse_setactions(benchit);

#ifdef CHANGEORDER
      if (WTuniverse_seteventorder(3,myevents))
            printf("changed order\n");
#endif

      WTuniverse_setbgcolor(0xFFF);

      WTuniverse_setresolution(WTRESOLUTION_LOWNTSC);
      /* Load the room as an object and assign the spin procedure to it.
         Set viewpoint to that loaded from file*/

      WTpq_init(&lobbyview);

      lobby = WTobject_new(TInfo.scenefile,&lobbyview,1.0,TRUE,TRUE);
      if (lobby) printf("Loaded the scene in/\n");
      spinObject = lobby;
      spinView = &lobbyview;
      /*printf("Number of polygons: %d\n",WTuniverse_npolygons());*/
      /*printf("Number of polygons in object %d\n",WTobject_npolygons(lobby));*/
      WTviewpoint_moveto(WTuniverse_getviewpoint(), &lobbyview);
      WTobject_settask(lobby,spin);

      /*Set the pivot point as the centre of the room*/
      WTuniverse_getmidpoint(centre);
      WTobject_setpivot(lobby,centre,WTFRAME_WORLD);

      /* Set default stereo viewing paramenters */
      WTviewpoint_setconvergence(WTuniverse_getviewpoint(),-63);
      WTviewpoint_setparallax(WTuniverse_getviewpoint(),2.034);
      WTviewpoint_setaspect(WTuniverse_getviewpoint(),0.81);
      WTviewpoint_setviewangle(WTuniverse_getviewpoint(),0.655);

      /* Set rendering for textured with perspective
         correction and gouraud shading. */
      rend = WTuniverse_getrendering();
      rend = rend | WTRENDER_TEXTURED | WTRENDER_PERSPECTIVE;
      if (noisy){
            printf("rendering is %ld\n",rend);
            printf("shading is %ld\n",WTRENDER_SHADED);
```

```
    }
    WTuniverse_setrendering(WTRENDER_V_TEXTURE | WTRENDER_PERSPECTIVE );
    if (noisy && 0){
          printf("Parallax is set at:%f.\n",
                WTviewpoint_getparallax(WTuniverse_getviewpoint()));
          printf("Converegence is set at: %d.\n",
                WTviewpoint_getconvergence(WTuniverse_getviewpoint()));
    }

    /* load lights from file */
    Wtlight_load("mylights");

    /*Set ambient light */
    WTlight_setambient(0.46);

    /* open the keyboard */
    WTkeyboard_open();
    num_tests=1;
    if (!datarec_openexp(TInfo.triggfile,NUMUPS))
          printf("Problem opening the daq module");

    else printf("DAQ open and waiting for data\n");

    WTobject_remove(lobby);
    /* Perform the test */


    setparameters();
    WTobject_add(lobby);
    prevtime = clock ();
    WTuniverse_ready();
    toggle_joystick();
    /*initial data stamp*/
    datastamp(data_rec_trigger1);
    WTuniverse_go();
    currtime = clock ();
    /*one last datastamp*/
    datastamp(data_rec_trigger1);
    WTuniverse_setbgcolor(0x000);
    WTuniverse_go1();
    num_rotupdate = fplan_get_actualupdates();
    fplan_exit();
    datarec_closeexp();
    totime=((float) currtime-prevtime)/((float) CLOCKS_PER_SEC);
    fr=num_rotupdate/totime;
    printf("Total time: %5.2f\tframe rate %5.2f\n", totime,fr);
    CleanupSpin();

    printf("Vacuuming the universe...swooooooop....");
    toggle_joystick();
    /* all done; clean everything up.*/
    WTuniverse_vacuum();
    printf("Turning off the LIGHTS!\n");
    WTlight_deleteall();
    WTlight_setambient(0.0);
    printf("<slurp> <slurp>\n");
}
```

```
/*********************************************
        Set parameters for rotation
*********************************************/
static void setparameters()
{
      int num=13;

      /*NUMBER OF ROTATIONS*/
      /*ROTATION SPEED*/
      /*rotate_angle = ((float) num)*PI/1440.0; */
      /*num_rotupdate = round((2*PI*((float)num_rotation))/(rotate_angle));*/

      fplan_setexpectedfps(TInfo.expected_fps);
      fplan_load(TInfo.fplanfile);

      num_rotupdate=fplan_expups();
      printf("Number of expected rotation updates: %d\n",num_rotupdate);
      SetupSpin(num_rotupdate+GetNM()*5);

}

/*********************************************
              Display option menu
*********************************************/

void display_menu()
{
      printf("    'q' Quit:\n");
      printf("    '?' Print this info:\n");
      printf("    'k' Increase parallax:\n");
      printf("    'l' Decrease parallax:\n");
      printf("    'a' Increase convergence:\n");
      printf("    's' Decrease convergence:\n");
}
```

10.3.17 spin.c

```
#include "wt.h"
#include "stdio.h"
#include "time.h"
#include "fplan.h"
#include "datarec.h"
#include "myprint.h"
#include "spea2d.h"
#include "..\tracker\track.p"

/*********************************************
   Check for key presses and rotate room
*********************************************/
static int zeros=0;
static clock_t *cycle_time;
static float frame_rate;
static int cycle_count=0;
static short port = COM2;
WTobject *spinObject=NULL;
WTpq *spinView = NULL;
WTsensor *track;  /* the Head Tracker */
```

```
/*for 2d text display, display every 5th update*/
int rots_to_upd=5,upd_int=5;

/*function declarations*/
void toggle_joystick();
void record_joystick();

void spin(WTobject *object)
{

      short key,conv;
      float parallax;

      /* get key presses, if any */
      key = WTkeyboard_getlastkey();

      switch(key){
          case'?':
            printf("    'q' Quit:\n");
            printf("    '?' Print this info:\n");
            printf("    'k' Increase parallax:\n");
            printf("    'l' Decrease parallax:\n");
            printf("    'a' Increase convergence:\n");
            printf("    's' Decrease convergence:\n");
            printf("    'j' Turn joystick on/off:\n");
            break;

          case'j':
            toggle_joystick();
            break;
          case'k':
            parallax = WTviewpoint_getparallax(WTuniverse_getviewpoint());
            parallax = parallax + 0.1;
            WTviewpoint_setparallax(WTuniverse_getviewpoint(),parallax);
            break;
          case'l':
            parallax = WTviewpoint_getparallax(WTuniverse_getviewpoint());
            parallax = parallax - 0.1;
            WTviewpoint_setparallax(WTuniverse_getviewpoint(),parallax);
            break;
          case'a':
            conv = WTviewpoint_getconvergence(WTuniverse_getviewpoint());
            conv = conv + 1;
            WTviewpoint_setconvergence(WTuniverse_getviewpoint(),conv);
                        printf("%d\n",conv);

            break;
          case's':
            conv = WTviewpoint_getconvergence(WTuniverse_getviewpoint());
            conv = conv - 1;
            WTviewpoint_setconvergence(WTuniverse_getviewpoint(),conv);
            break;
          case'o':
            /* Initialize a Tracker sensor */
            printf("trying to open head tracker\n");
            track = WTsensor_new(WTtrack_open,WTtrack_close,WTtrack_update,
                              WTserial_new(port,(port==COM1?4:(port==COM2?3:-1)),
```

```
                                (BAUD192|NP|DATA8|STOP1),89),1,WTSENSOR_DEFAULT);
            if (!track) WTerror("Couldn't find head tracker\n");
            /* Scale sensor sensitivity with the size of the universe. */
              printf("found head tracker. it is on");
            WTsensor_setsensitivity(track, 1.0);
            break;
          case'p':
            WTsensor_delete(track);
            WTviewpoint_moveto(WTuniverse_getviewpoint(), spinView);
          default:
            break;
        }
      record_joystick();
      if (fplan_isdone()) {
                  frame_rate = WTuniverse_framerate();
                  WTobject_remove(spinObject);
                  WTuniverse_stop();
            }
      else fplan_domove();
}

char astr[40];

void draw2d(short eye){

      WTsprgbcolor(255,255,255);
      WTsprgbbcolor(0,0,0);
      WTsptransp(0);
      WTspselfont(11);
      if (!rots_to_upd){
            sprintf(astr,"%5.2f",WTuniverse_framerate());
                  /*
                   *WTuniverse_npolygons());
                   */
            WTsptext(0.40,0.50,astr);
            rots_to_upd=upd_int;
      }
      else {
            rots_to_upd--;
            WTsptext(0.40,0.50,astr);
      }
}

void benchit(){
      cycle_time[cycle_count]=clock();
      if (cycle_count)
            if (cycle_time[cycle_count]==cycle_time[cycle_count-1])
            {
                  zeros++;
                  myprint("%d Zero cycle time at cycle %d!\n",zeros,cycle_count);
            }
      cycle_count++;
}

void  SetupSpin(int howmany){
      cycle_time = malloc(sizeof(clock_t)*(howmany));
      cycle_count = 0;
      zeros = 0;
```

```
}

void  CleanupSpin(void){
      if (cycle_time)
            free(cycle_time);
}
```

10.3.18 fplan.c

```
#include <stdio.h>
#include "fplan.h"
#include "move.h"
#include "myprint.h"

movement_rec      *FlightPlan=NULL;       /*array of movements*/
int         fplan_num_movements;    /*number of movements in array*/
int         fplan_index;            /*an index into the FlightPlan. Used for two
purposes*/
FILE        *fplan_file=NULL; /*file used to load from file*/
int         all_done;
int         fplan_actual_updates;
float       EXPECTED_FPS;
extern            int         noisy;
/*enum system is as follows: two digit number
  first digit is which is constant, the second
  is which parameter specifies the end of the motion.
  disp,veloc, accel, jerk, time = 0, 1, 2, 3, 4 respectively
  numbers greater 100 are for linear motion
*/

enum {
      CONST_VELOC_TIME = 14,
      CONST_ACCEL_DISP = 20,
      CONST_ACCEL_TIME = 24

};

void  fplan_load(char *name){
      fplan_index = 0;
      fplan_actual_updates = 0;
      all_done=0;
      move_init();
      printf("trying to open %s...",name);
      if (fplan_file = fopen(name,"r")){
            int cnt;
            fscanf(fplan_file,"%d",&fplan_num_movements);
            printf("%d movements.\n",GetNM());
            FlightPlan = (movement_rec *) malloc(sizeof(movement_rec)*GetNM());
            for (cnt=0;cnt<fplan_num_movements;cnt++)
                  fplan_read_next_movement();
      }
      else
            printf("failed!\n");
      if (fplan_file) fclose(fplan_file);
      fplan_file=NULL;
      fplan_index=-1;
      fplan_start_next_movement();
}
```

```
void fplan_picknload(void) {
      fplan_load("fplan.txt");
}

void fplan_read_params(int numparams){
      int cnt;
      if (noisy) printf("parameters: ");
      for (cnt=0;cnt<numparams;cnt++) {
            fscanf(fplan_file,"%f",&(FP(fplan_index).p[cnt]));
            if (noisy) printf("%f\t",FP(fplan_index).p[cnt]);
      }
      if (noisy) printf("\n");
      /*zero the rest*/
      while (cnt<FPLAN_MAX_PARAMS)
            FP(fplan_index).p[cnt++]=0.0;

}

void fplan_read_next_movement(void){
      fscanf(fplan_file,"%d",&(FP(fplan_index).type));
      if (noisy)
            printf("movement %d of type %d\n",fplan_index,FP(fplan_index).type);

      switch (FP(fplan_index).type) {
          case CONST_VELOC_TIME:
            fplan_read_params(2);
            FP(fplan_index).NumUps = EXPECTED_FPS*FP(fplan_index).p[1]+1;
            break;
          case CONST_ACCEL_DISP:
            fplan_read_params(3);
            break;
          case CONST_ACCEL_TIME:
            fplan_read_params(2);
            FP(fplan_index).NumUps = EXPECTED_FPS*FP(fplan_index).p[1]+1;
            break;
          default:
            /*oops, unsupported movement*/
            break;
      }
      fplan_index++;
}

void  fplan_start_next_movement(void){
      float v,a,j;
      fplan_index++;
      switch(FP(fplan_index).type){
          case CONST_VELOC_TIME:
            move_const_velocity(FP(fplan_index).p[0],FP(fplan_index).p[1]);
            break;
          case CONST_ACCEL_DISP:
            /*assume V0=0 for now*/
            move_getkinetics(&v,&a,&j);
            move_const_accel(v*180.0/PI,FP(fplan_index).p[0],
                        sqrt(2*FP(fplan_index).p[1]/
                              FP(fplan_index).p[0]));
            break;
          case CONST_ACCEL_TIME:
```

```
            move_getkinetics(&v,&a,&j);
            move_const_accel(v*180.0/PI,FP(fplan_index).p[0],
                        FP(fplan_index).p[1]);
            break;
          default:
            /*not supported, dont do anything*/
            break;
      }
}

void  fplan_exit(void){
      printf("in fplan_exit...\n");
      /*fplan_file is usually closed in fplan_load*/
      if (fplan_file)
            fclose(fplan_file);
      fplan_file=NULL;
      if (FlightPlan)
            free(FlightPlan);
      FlightPlan=NULL;
}

int   fplan_expups(void){
      /*number of expected updates*/
      int cnt;
      int expups=0;
      for (cnt=0;cnt<fplan_num_movements;cnt++)
            expups+=FP(cnt).NumUps;
      return (expups);
}

void fplan_domove(void){
      if (FP(fplan_index).NumUps==0) {
            if ((fplan_index+1)<fplan_num_movements)
                  fplan_start_next_movement();
            else (all_done=1);
      }
      fplan_actual_updates++;
      move_domove();
      FP(fplan_index).NumUps--;
      myprint("%d ",FP(fplan_index).NumUps);
}

void  fplan_setexpectedfps(float efps){
      EXPECTED_FPS = efps;
}
```

10.3.19 move.c

```
#include "wt.h"
#include "move.h"
#include "fplan.h"


int ROTATE_VIEWPOINT;

float velocity;         /*angular velocity in rad/sec       */
int is_accelerating;    /*is it constant accelerating (not jerk)*/
float acceleration;     /*angular_accel in rad/sec/sec            */
```

```
int is_jerking;          /*is it const jerking? d(accel)/dt != 0 */
float jerk;        /*angular jerk in rad/sec/sec/sec    */

int num_cycles;          /*number of cycle in the 'path'*/
int cur_cycle;           /*current point in the path    */

int rot_count;
/*wont use angle_path*/
float *angle_path=NULL; /*an array of angles we want  */

float view_angle=0.0;    /*the current rotation angle (might not be actual viewing
angle with head tracker on)*/
float rotate_angle;      /*the amount (rads) to rotate view/object in a cycle*/
float rotationdirection=1.0;
WTobject *itsObject=NULL;/*the room object in case we need it*/

extern int noisy;

void move_init(void){
      view_angle=0.0;
      rot_count=0;
/*    move_setrotdirection(DIRECTION_CW);*/
      move_reset();
}

int move_isdone(void){
      return (cur_cycle>num_cycles);
}

float move_getangle(void){
      return (view_angle);
}

int move_getrot_count(void){
      return rot_count;
}

void move_setobject(WTobject *anobject){
      itsObject=anobject;
}

void move_accelerate(void){
      rotate_angle = acceleration/(2*EXPECTED_FPS*EXPECTED_FPS) +
                        velocity/EXPECTED_FPS;
      velocity += acceleration/(EXPECTED_FPS);
}
void move_jerk(void){
      rotate_angle = jerk/(6*EXPECTED_FPS*EXPECTED_FPS*EXPECTED_FPS)+
            acceleration/(2*EXPECTED_FPS*EXPECTED_FPS) + velocity/EXPECTED_FPS;
      velocity += acceleration/(EXPECTED_FPS) + jerk/(2*EXPECTED_FPS*EXPECTED_FPS);
      acceleration += jerk/(EXPECTED_FPS);
}

void move_domove(void){
      if (is_accelerating)
            move_accelerate();
      else if (is_jerking)
```

```
            move_jerk();

      view_angle += rotate_angle;
      /*Sense8 says that rotating an object is less efficient than just rotating
      the viewpoint. It has something to do with reordering a data-structure*/

      /*WTobject_rotate(object,rotate_axis,rotate_angle,WTFRAME_WORLD);*/
      WTviewpoint_rotate(WTuniverse_getviewpoint(),Z,
                        rotationdirection*rotate_angle,WTFRAME_VPOINT);
      rot_count++;

}

void  move_reset(void){
      is_jerking=is_accelerating=0;
      acceleration=jerk=0.0;
      if (angle_path){
            free(angle_path);
            angle_path=NULL;
      }
      cur_cycle=num_cycles=0;
}

void  move_const_velocity(float veloc,float time){
      /*velocity in ~degs/sec*/
      /*WTK wants it in rads*/
      if (noisy)
            printf("move_const_velocity: veloc=%f time=%f",veloc,time);
      move_reset();
      rotate_angle = veloc * PI/180 / EXPECTED_FPS;
      velocity = veloc * PI/180;
      num_cycles=(int) EXPECTED_FPS*time;
      if (noisy) printf("acceleration = %f, num_cycles = %d, velocity = %f,
rotate_angle = %f",
            acceleration,num_cycles,velocity, rotate_angle);
}

void  move_const_accel(float V0, float accel,float time){
      /*accelertion in deg/sec/sec, convert to rad/sec/sec  */
      /*V0 initial velocity in deg/sec                    */
      if (noisy) printf("move_const_accel: V0=%f accel=%f time=%f\n",V0,accel,time);
      move_reset();
      acceleration = accel*PI/180;
      is_accelerating = 1;
      num_cycles= time*EXPECTED_FPS;
      velocity = V0 * PI/180;
      rotate_angle = velocity / EXPECTED_FPS;
      if (noisy) printf("acceleration = %f, num_cycles = %d, velocity = %f,
                        rotate_angle = %f",acceleration,num_cycles,velocity,
                        rotate_angle);
}

void  move_const_jerk(float A0,float V0, float jk, float time){
      /*A0, V0 initial accel and veloc in deg   */
      /*jk is jerk in deg/sec/sec/sec           */
      move_reset();
      jerk = jk*PI/180;
      is_jerking = 1;
```

```
        num_cycles = time/EXPECTED_FPS;
        velocity = V0 * PI/180;
        rotate_angle = velocity / EXPECTED_FPS;
        acceleration = A0 * PI/180;
}

void    move_getkinetics(float *v,float *a,float *j){
        *v=velocity;
        *a=acceleration;
        *j=jerk;
}

void    move_setrotdirection(int rotdir){
        rotationdirection = (rotdir==DIRECTION_CW)?(1.0):(-1.0);
        printf("Setting rotation direction to %f\n",rotationdirection);
}
```

10.3.20 datarec.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "datarec.h"
#include "runexp.h"

static FILE *expfile=NULL,*expfile2=NULL;
static RECTYPE     *data =NULL,*data2=NULL;
static char
        expname[FNAMESIZE],expname2[FNAMESIZE],defname[FNAMESIZE]="expdata.dat";
static int   expmaxentries,exppos,exppos2;
extern int   noisy;

int     datarec_openexp(char *name, int maxentries){
        char *period,*slash;
        /*pass name = NULL to get default name 'defname'
          must pass maxentries > 0
          returns 0 if there was a problem.
          */
        if (noisy)
              printf("openexp: name = %30s, maxentries = %d\n",name,maxentries);

        if (maxentries>0){
              data = (RECTYPE *) malloc(sizeof(RECTYPE)*maxentries);
              data2 = (RECTYPE2 *) malloc(sizeof(RECTYPE2)*maxentries);
        }
        else {
              printf("openexp: Must pass a greater than 0 for openexp\n");
              return 0;
        }
        if (!data) {
              printf("openexp: Could not allocate trigger 1 memory\n");
              if (data2) free(data2);
              data2=NULL;
              return 0;
        }
          else if (!data2) {
              printf("openexp: Count not allocate trigger 2 memory\n");
              if (data) free(data);
```

```
            data = NULL;
            return 0;
      }
      exppos=0;
      exppos2=0;
      if (name) {

            strncpy(expname,name,FNAMESIZE);
            strncpy(expname2,name,FNAMESIZE);
      }
      else {

            strcpy(expname,defname);
            strcpy(expname2,defname);
      }
      period = strrchr(expname2,'.');
      slash = strrchr(expname2,'\\');
      if ((period-slash)>5)
            *(period-1)='2';
      else
            strcpy(period,"t2.dat");
      printf("filename 2 is %s\n",expname2);
      return (expmaxentries = maxentries);
}

int datarec_recdata(char *info,int which){

      if ((which==data_rec_trigger1) && data && (exppos<expmaxentries)) {
            data[exppos++]=*((RECTYPE *) info);
            if (noisy)
                  printf("recdata: recording info: time %d angle %f \n",
                        ((RECTYPE *)info)->time,
                        ((RECTYPE *)info)->angle);
      }
      if ((which==data_rec_trigger2) && data2 && (exppos2<expmaxentries)) {
            data2[exppos2++]= *((RECTYPE2 *) info);
            if (noisy)
                  printf("recdata: recording info: time %d angle %f \n",
                        ((RECTYPE2 *)info)->time,
                        ((RECTYPE2 *)info)->angle);
      }

      else return 0;
      return 1;
}

int datarec_closeexp(void){
      char response[30];
      int  printout=0,cnt;
      if (data && data2) {
            HedRec header=exppos,header2=exppos2;
            int hd= 1,hd2=1;
            printf("trigger 1 has %d, and trigger 2 has %d records\n",
                  exppos,exppos2);
            printf("Do you want to see the stamps before saving them?\n");
            scanf("%30s",response);
            printout = (response[0]=='y' || response[0]=='Y');
            if (expfile = fopen(expname,"wb")){
```

```
                printf("closeexp: Saving %d records\n",exppos);
                if (printout) {
                      for (cnt=0;cnt<exppos;cnt++)
                            printf("Stamp %d\tangle=%5.2f\ttime=%ld\n",
                            cnt,data[cnt].angle,data[cnt].time);
                }
                /*write the header info*/
                fwrite(&hd,sizeof(int),1,expfile);
                fwrite(&header,sizeof(HedRec),1,expfile);
                /*write the experiment data*/
                fwrite(data,sizeof(RECTYPE),exppos,expfile);

                free(data);
                data=NULL;
                fclose(expfile);
                expfile = NULL;
          }
          else {
                printf("Could not open %s for saving trigger file!\n",expname);
                goto didntwork;
          }
          if (expfile2 = fopen(expname2,"wb")){
                printf("closeexp: Saving %d records\n",exppos2);
                if (printout) {
                      for (cnt=0;cnt<exppos2;cnt++)
                            printf("Stamp %d\tangle=%5.2f\ttime=%ld\n",
                            cnt,data2[cnt].angle,data2[cnt].time);
                }
                fwrite(&hd2,sizeof(int),1,expfile2);
                fwrite(&header2,sizeof(HedRec),1,expfile2);
                fwrite(data2,sizeof(RECTYPE2),exppos2,expfile2);
                free(data2);
                data2=NULL;
                fclose(expfile2);
                expfile2=NULL;
                return 1;
          }
          else {
                printf("Could not open %s for saving trigger 2 file!\n",expname2);
          }
    }
    else {
          printf("closeexp: No open experiment\n");

    }
    didntwork:
          free (data);
          free (data2);
          return 0;

}
```

10.3.21 dataread.c

```
/*Dataread.c */

#include <stdio.h>
#include <stdlib.h>
```

```
#include <string.h>
#include "datarec.h"
#include "dataread.h"
#include "myprint.h"
#include "statmod.h"

static FILE       *expfile=NULL,*expfile2=NULL;
RECTYPE     *dataread = NULL;
RECTYPE2    *dataread2 = NULL;
static char       expname[20],defname[20]="expdata.dat";
int         data_numentries,data_numentries2,data_numexps;

int   openexpread(char *name);
int   closeexpread(void);
void  secondfilename(char *a, char *b);

int   openexpread(char *name){
      /*1/11/95 Changed it so that it opens both trigger files*/
      /*Initially it will only load in the first experiment */
      char name2[90];
      char *period, *slash;
      int num_exp;
      /* Open second trigger file too */
      strcpy(name2,name);
      period = strrchr(name2,'.');
      slash = strrchr(name2,'\\');
      if ((slash) && (period-slash)>5)
            *(period-1)='2';
      else
            strcpy(period,"t2.dat");
      if (!dataread){
            myprint("openexpread: open file %s\n",name);

            if (expfile=fopen(name,"rb")){
                  fread(&num_exp,sizeof(int),1,expfile);
                  fread(&data_numentries,sizeof(HedRec),1,expfile);

                  /*printf("openxpread: %d entries being read\n",GetDNE());*/
                  dataread = malloc(sizeof(RECTYPE)*data_numentries);
                  fread(dataread,sizeof(RECTYPE),data_numentries,expfile);
                  if (expfile2 = fopen(name2,"rb")){
                        fread(&num_exp,sizeof(int),1,expfile2);
                        fread(&data_numentries2,sizeof(HedRec),1,expfile2);
                        /*printf("openexpread: %d entries being read\n",
                        data_numentries2);*/
                        dataread2 = malloc(sizeof(RECTYPE2)*data_numentries2);
                        fread(dataread2,sizeof(RECTYPE2),data_numentries2,expfile2);

                        myprint("openexpread: done\n");
                        strncpy(expname,name,20);
                        fclose(expfile);
                        fclose(expfile2);
                        return GetDNE();
                  }
                  else {
                        myprint("openexpread: could not open second trigger
                              file\n");
                        fclose(expfile);
```

```
                        free(dataread);
                        return 0;
                  }
            }
            else {
                  myprint("openexpread: could not open %s\n",name);
                  return 0;
            }
      }
      else {
            myprint("openexpread: already an experiment file open\n");
            return 0;
      }
}

int closeexpread(){
      if (dataread){
            free(dataread);
            if (dataread2) free (dataread2);
            dataread=dataread2=NULL;
            myprint("closeexpread: closed experiment\n");
            return 1;
      }
      else {
            myprint("closeexpread: nothing to close\n");
            return 0;
      }
}
```

10.3.22 joysens.c

```
/*joysens.c: Joystick Update Driver */

#include <stdio.h>
#include <stdlib.h>
#include "mathlib.h"
#include "wt.h"

#include "mathlib.p"

void WTjoystick_detbutt(WTsensor *sensor);

/*
 * read joystick
 */

void WTjoystick_detbutt(WTsensor *sensor)
{
       /*Dont need any work done here. Just call the
         raw up date function and let the WT_action
         function do all the work */
      WTjoystick_rawupdate(sensor);
}
```

10.3.23 joyutil.c

```
#include "wt.h"
```

```
#include <stdio.h>
#include "joysens.h"
#include "datarec.h"
#include "move.h"
void datastamp(int a);
void toggle_joystick();
void record_joystick();

/*Joystick utilities*/
WTsensor *joystick=NULL;
FLAG  prevtrigger=0,prevtrigger2=0;

/*assume joystick starts out with trigger
  released*/

void toggle_joystick(){
      if (!joystick){
            joystick=WTsensor_new(WTjoystick_open,WTjoystick_close,
                        WTjoystick_rawupdate,NULL,1,WTSENSOR_DEFAULT);
                  /*joystick=WTmouse_new();*/
            if (!joystick) printf("Warning, couldnt open joystick!\n");
            else {
                  printf("Joystick sensor created...running\n");
                  printf("sensitivity %f\n",WTsensor_getsensitivity(joystick));
                  WTsensor_setsensitivity(joystick,1.0);
                  WTviewpoint_addsensor(WTuniverse_getviewpoint(),joystick);
            }
      }
      else {
            WTsensor_delete(joystick);
            joystick=NULL;
            printf("Joystick sensor destroyed.\n");
      }
}

void record_joystick(){
      if (joystick){
            FLAG triggerdown,triggerdown2;
            long buttons;
            buttons = WTsensor_getmiscdata(joystick);

            triggerdown = buttons & WTJOYSTICK_TRIGGERDOWN;
            triggerdown2 = buttons & WTJOYSTICK_TOPDOWN;

            if (triggerdown != prevtrigger){
                  /*Stamp when pressed and released only*/
                  datastamp(data_rec_trigger1);
                  prevtrigger = triggerdown;
            }
            if ((prevtrigger2 == 0) && (triggerdown2) )
                  /*Stamp when pressed (not when released) */
                  datastamp(data_rec_trigger2);
            prevtrigger2=triggerdown2;
      }
}

static RECTYPE dummyRec;
void datastamp(int which){
```

```
        dummyRec.time=clock();
        dummyRec.angle=move_getangle();
        datarec_recdata(( char *) &dummyRec,which);
        printf("stamping at %lf, time %ld on %d\n",dummyRec.angle, dummyRec.time,
              which);
}
```

10.3.24 statmod.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "datarec.h"
#include "dataread.h"
#include "myprint.h"
#include "statmod.h"
#include <math.h>

#define PI 3.14159
#define THRESHOLD 0.500
#define TOTAL_ROTATIONS 3
#define RESOLUTION 36
#define MAX_CELLS 20
#define ORESOLUTION RESOLUTION*TOTAL_ROTATIONS
#define SECANGLEBEG(i) (i*2.0*PI/RESOLUTION)
#define SECANGLEEND(i) ((i+1)*2.0*PI/RESOLUTION)
#define THRESHOLD_ARC (2.0*PI/RESOLUTION*THRESHOLD)

#define RAD90      (RAD180/2.0)
#define RAD180     PI
#define RAD270   (RAD180*3.0/2.0)
#define RAD360     (2.0*RAD180)
#define REORIENT_MARGIN (RAD180/6.0)  /*30.0 degrees*/
#define WANT_BINS 1
#define WANT_RAW 1
#define IN_SECONDS 1
#define WANT_VERIFICATION 1
#define PLAN_A 1

int
vectionbins[RESOLUTION],dropinbins[RESOLUTION],dropoutbins[RESOLUTION],reorientbins[
RESOLUTION];
int
ovectionbins[ORESOLUTION],odropinbins[ORESOLUTION],odropoutbins[ORESOLUTION],oreorie
ntbins[ORESOLUTION];
StampRec dropins[MAX_CELLS];
StampRec dropouts[MAX_CELLS];
StampRec reorients[MAX_CELLS];
int reorient_wall[MAX_CELLS];
int num_dropins;
int num_dropouts;
TreatmentInfoRec *TreatInfo=NULL;

float percentvection;
float latency;
float angle_latency;
int total_vection=0,total_experiment=0,total_reorients;
int begin_time,end_time;
```

```
int relative(float a, int b);
int whichsector(float a);
int owhichsector(float a);
void IncrAllBins(int z[],int a);
void IncrBins(int z[],int a, int b, int c);
int pastthreshold(float a);
void fixangles(float *, float *, int *);
void perform_vection_stats(float begangle,float endangle,
                    int secnumb,int secnume,int numrevs,int overall);
int   find_wall(float angle);

int   statmod_done;
void statmod_menu(void){
      char choice;
      statmod_done = 0;
      while (!statmod_done){
            printf("Statistics Module Menu\n");
            printf("Perform analysis on (I)ndividual run\n");
            printf("Perform analysis on (S)ubject\n");
            printf("E(X)it\n");
            scanf("%c",&choice);
            switch (choice) {
                case 'I':
                case 'i':
                  /*AnalyseIndividualRun();*/
                  break;
                case 'S':
                case 's':
                  /*AnalyseSubject();*/
                  break;
                case 'X':
                case 'x':
                  statmod_done = 1;
                  break;
            }
      }
}

void statmod_init(void){
      int cnt;
      StampRec zerostamp;
      zerostamp.time = 0;
      zerostamp.angle = 0.0;

      total_vection=0;
      total_experiment=0;
      total_reorients=0;
      begin_time= end_time = 0;
      for (cnt=0;cnt<RESOLUTION;cnt++){
            vectionbins[cnt]=0;
            dropinbins[cnt]=0;
            dropoutbins[cnt]=0;
            reorientbins[cnt]=0;
      }
      for (cnt=0;cnt<ORESOLUTION;cnt++){
            ovectionbins[cnt]=0;
            odropinbins[cnt]=0;
            odropoutbins[cnt]=0;
```

```
            oreorientbins[cnt]=0;
      }
      for (cnt=0;cnt<MAX_CELLS;cnt++){
            dropins[cnt]=zerostamp;
            dropouts[cnt]=zerostamp;
            reorients[cnt]=zerostamp;
            reorient_wall[cnt]=0;
      }
      num_dropins=0;
      num_dropouts=0;
}

void do_analysis(char *name){
      statmod_init();
      /*performstats(name);*/
}

int   find_wall(float angle){
      float phi,psi;
      int n,wall;
      phi=fmod(angle,RAD360);
      psi=fmod(phi,RAD90);
      n=floor(phi/RAD90);
      if (psi>REORIENT_MARGIN) {
            /*printf("adding one\n");*/
            wall=(n+1) % 4;
      }
      else
            wall=n % 4;
      return wall;
}
void performstats2_1(void){
      int exppos;
      int secnum;
      int removed=0;

      if (0) printf("Sorting reorients: %d\n",GetDNE2());
      if (oreorientbins[0]) {
            printf("orientbins[0]==1 and I havent even started yet!\n");
            getch();
      }
      for (exppos=0;exppos<GetDNE2();exppos++){
            float angle;
            removed=0;
            angle = AngleData2(exppos);
            if ((angle==AngleData(GetDNE()-1)) && (exppos==0)) {
                  printf("ignoring reorient %d at %f\n",exppos+1,angle);
                  removed=1;
                  break;
            }
            /*now find out which wall its fallen on*/
            if (exppos==0)
                  reorient_wall[exppos]=find_wall(angle);
            else {
                  float diff=angle-AngleData2(exppos-1);
#ifdef PLAN_A
                  reorient_wall[exppos]=find_wall(angle);
                  if ((reorient_wall[exppos]==reorient_wall[exppos-1])
```

```
                        && (diff<(RAD90+REORIENT_MARGIN))){

/*If two reorientations have the same surface, we need to check
      and make sure that they are not too close, i.e., that the user
      did not accidentally push the reorientation button several times.
      Also we need to move the reorientation to the next surface if the
      previous one occurred early enough. Note: settin the reorientation
      to the negative of its designated surface is tantamount to "removing"
      it. The reorientations are not outright removed so that the researcher
      can see where subjects reorientations are being "removed"*/

                        if (fmod(angle,RAD90)>REORIENT_MARGIN){
/*Two angles in the first section of the region*/
                              reorient_wall[exppos]=-reorient_wall[exppos];
                        }

                        else if ((fmod(AngleData2(exppos-1),RAD90)<=REORIENT_MARGIN)
                              && (diff<REORIENT_MARGIN)) {
/*Two angles in the second part of the region
  or two too close in the first part of the region
 */
                              reorient_wall[exppos]=-reorient_wall[exppos];
                        }
                        else if ((fmod(AngleData2(exppos-1),RAD90)<=REORIENT_MARGIN)
                              && (fmod(angle,RAD90)>REORIENT_MARGIN)){
/*Two in the first part of the region. This time, on between -60 and -30
      and one between -30 and 0*/
                              reorient_wall[exppos]=-reorient_wall[exppos];
                        }
                        else {
/*Two reorientations that are far enough apart for the second one to
  be moved to the next surface*/
                              reorient_wall[exppos]=(reorient_wall[exppos]+1)%4;
                        }
                  }
#endif
/*note: PLAN_B was a previously used surface-scheme but was improved upon
in PLAN_A. */

#ifdef PLAN_B

                  diff=fmod(diff,RAD360);
                  if ((diff>REORIENT_MARGIN) && (diff<=(RAD90+REORIENT_MARGIN)))
                        reorient_wall[exppos]=(reorient_wall[exppos-1]+1)%4;
                  else if (diff<=(RAD180+REORIENT_MARGIN))
                        reorient_wall[exppos]=(reorient_wall[exppos-1]+2)%4;
                  else if (diff<=(RAD270+REORIENT_MARGIN))
                        reorient_wall[exppos]=(reorient_wall[exppos-1]+3)%4;
                  else  /*same wall as before*/
                        reorient_wall[exppos]=reorient_wall[exppos-1];
#endif
            }
            /*fix angle, make sure between 0 and 2*PI*/
            /*printf("angle is %f\n",angle);*/
            if (reorient_wall[exppos]>=0 && !removed){
                  secnum = owhichsector(angle);
                  oreorientbins[secnum]++;
                  while (angle>2.0*PI) angle -= 2.0*PI;
```

```
                while (angle<-2.0*PI) angle += 2.0*PI;
                /*what sector it falls in*/
                secnum = whichsector(angle);
                reorientbins[secnum]++;
            }
            if (!removed) {
                reorients[exppos]=StampData2(exppos);
                total_reorients++;
            }
        }
}


enum {
        NOT_OVERALL = 0,
        OVERALL = 1
};

void performstats_1(void){
        int exppos;
        int latencyset=0;
        int secnume,secnumb,osecnume,osecnumb;
        int numrevs;
        latency=0.0;
        angle_latency=0.0;
        if (0) printf("analyzing dropin/dropouts...\n");
        for (exppos=1;exppos<(GetDNE()-1);exppos+=2){
            float begangle,endangle,obegangle,oendangle;
            /* this is for percent of time trigger down*/

            if ((exppos==1) && (TimeData(exppos)==TimeData(exppos-1)) &&
                (AngleData(exppos)==AngleData(exppos-1))){
                  exppos++;
            }
            /*printf("%s\t%d\n",TreatInfo->treatcode,exppos);*/

            if (exppos>=(GetDNE()-1)) {
                  printf("breaking!\n");
                  break;
            }
            total_vection=total_vection +
                  (TimeData((exppos+1))-
                   TimeData(exppos));

            /*fix the angles. Change angles to be within 0 and 2PI
              and return number of complete revolutions in numrev*/
            obegangle=begangle = AngleData((exppos));
            oendangle=endangle = AngleData((exppos+1));
            /*first do the "overall" bins*/
            osecnumb = owhichsector(obegangle);
            osecnume = owhichsector(oendangle);
            fixangles(&begangle,&endangle,&numrevs);
            secnumb = whichsector(begangle);
            secnume = whichsector(endangle);
            /*numrevs = 0 for overall since there is no folding*/

            /*
               printf("calling perform o stats %f,%f,%ld,%ld\n",begangle,
```

```
                endangle,secnumb,secnume);
            */
            perform_vection_stats(obegangle,oendangle,osecnumb,osecnume,0,OVERALL);
            /*

               getch();
            */
            /*
               printf("calling perform stats %f,%f,%ld,%ld\n",begangle,
                  endangle,secnumb,secnume);
            */

      perform_vection_stats(begangle,endangle,secnumb,secnume,numrevs,NOT_OVERALL);

            if (paststhreshold(oendangle-obegangle)){
                  odropinbins[osecnumb]++;
                  dropinbins[secnumb]++;
                  dropins[num_dropins++]=StampData(exppos);

                  if (!latencyset){
                        latency = ((float) (TimeData(exppos)-TimeData(0)))/
                              ((float) CLOCKS_PER_SEC)-1.0;

                        angle_latency=AngleData(exppos);
                        latencyset=1;
                  }
                  if (((exppos+2)<GetDNE()) &&
                      !((exppos==(GetDNE()-3)) &&
                        (AngleData(exppos+1)==AngleData(exppos+2)))){
                        /*last condition weeds out those 'dropouts'
                          that occur when the actual scene motion
                          has stopped*/
                              odropoutbins[osecnume]++;
                              dropoutbins[secnume]++;
                              dropouts[num_dropouts++]=StampData(exppos+1);
                        }
            }
            myprint("secnumb=%d\tsecnume=%d\tbeg=%f\tend=%f\n",secnumb,secnume,
                  begangle,endangle);
      }

      end_time = TimeData((GetDNE()-1));
      begin_time = TimeData(0);

      total_experiment=end_time - begin_time + CLOCKS_PER_SEC*2.0;
      percentvection = ((float) total_vection)/((float) total_experiment);
      if (0) printf("...done\n");
}

void  perform_vection_stats(float begangle,float endangle, int secnumb,int
secnume,int numrevs,int overall){
      if (begangle<endangle) {      /*case I*/

            myprint("case I .. ");

            if (secnumb==secnume) { /*case a or b*/
                  /*increment secnumb/e bin if past threshold*/
```

```
            myprint("case a or b..");
            if (pastthreshold(endangle-begangle)) {
                  if (overall)
                        ovectionbins[secnumb]++;
                  else
                        vectionbins[secnumb]++;
            }
      }
      else {                              /*case c or d*/
            /*incr range from scnumb+1 to secnume-1*/
            /*and check if threshold is met in
              secnume and secnumb*/

            myprint("case c or d..");

            if (overall)
                  IncrBins(ovectionbins,secnumb+1,secnume-1,1);
            else
                  IncrBins(vectionbins,secnumb+1,secnume-1,1);
            if (pastthreshold(SECANGLEEND(secnumb)-begangle)){
                  if (overall)
                        ovectionbins[secnumb]++;
                  else
                        vectionbins[secnumb]++;
            }
            /*if endangle>SECANGLEBEG(secnume) then it means
              we are dealing with unfolded angles and the angle
              went beyond the ORESOLUTION, so this condition
              ensures that the last sector is included in the
              vection reports for unfolded angles.
              */
            if ((endangle>SECANGLEBEG(secnume)) ||
                 pastthreshold(endangle-SECANGLEBEG(secnume))){
                  if (overall)
                        ovectionbins[secnume]++;
                  else
                        vectionbins[secnume]++;
            }
      }
      if (numrevs){           /*If any complete revolutions
                              deal with them now*/
            if (overall)
                  IncrAllBins(ovectionbins,numrevs);
            else
                  IncrAllBins(vectionbins,numrevs);
      myprint("revs = %d\n",numrevs);
      }
}
else        {     /* case II */
      /*unfolded angles should never reach this point*/
      if (overall){
            printf("I'm not supposed to be here!\n");
            printf("%f\t%f\t%d\t%d\n",begangle,endangle,secnumb,secnume);
            /*getch();*/
      }
      myprint("case II...");

      if (secnumb==secnume) { /*case a or b*/
```

```
                /*increment secnumb/e bin if past threshold*/

                myprint(" a or b..");

                if ((begangle!=endangle) &&
                     pastthreshold(2.0*PI/RESOLUTION-(begangle-endangle))){
                      if (overall)
                            ovectionbins[secnumb]++;
                      else
                            vectionbins[secnumb]++;
                }
          }
          else {                    /*case c or d*/
                /*incr range from scnumb+1 to secnume-1*/
                /*and check if threshold is met in
                  secnume and secnumb*/

                myprint(" c or d..");

                if (overall) {
                      IncrBins(ovectionbins,secnumb+1,RESOLUTION-1,1);
                      IncrBins(ovectionbins,0,secnume-1,1);
                      if (pastthreshold(SECANGLEEND(secnumb)-begangle))
                            ovectionbins[secnumb]++;
                      if (pastthreshold(endangle-SECANGLEBEG(secnume)))
                            ovectionbins[secnume]++;
                }
                else {
                      IncrBins(vectionbins,secnumb+1,RESOLUTION-1,1);
                      IncrBins(vectionbins,0,secnume-1,1);
                      if (pastthreshold(SECANGLEEND(secnumb)-begangle))
                            vectionbins[secnumb]++;
                      if (pastthreshold(endangle-SECANGLEBEG(secnume)))
                            vectionbins[secnume]++;
                }
          }
          if (numrevs) {
                if (overall)
                      IncrAllBins(ovectionbins,numrevs);
                else
                      IncrAllBins(vectionbins,numrevs);
          }

          myprint(" revs = %d\n",numrevs);

      }
      /*done with the Vection Bins*/

#if 0
      /*do drop in and drop out bins*/
      if (overall){
            if (pastthreshold(endangle-begangle)){
                  odropinbins[owhichsector(begangle)]++;
                  odropoutbins[owhichsector(endangle)]++;
            }
#if 0
            if (owhichsector(begangle)==0)
                  printf("adding a dropin at 0, angle is %lf\n",begangle);
```

```
            if (owhichsector(endangle)>=ORESOLUTION){
                  printf("hey, my end sector %ld (angle=%lf) is gte
ORESOLUTION!\n",owhichsector(endangle),endangle);
                  getch();
            }
#endif

      }
      else {

            if (((endangle>begangle) && (pastthreshold(endangle-begangle))) ||
                ((begangle>endangle) && (pastthreshold(endangle-begangle+2.0*PI))))
            {
                  dropinbins[whichsector(begangle)]++;
                  dropoutbins[whichsector(endangle)]++;
            }
      }
#endif
}

void  printstats(char *expname){
      FILE  *out=NULL;
      char  outname[90];
      char  *period;
      int cnt;
      strcpy(outname,expname);
      if (period=strstr(outname,".")) *period=0;
      strcat(outname,".cht");
      out = fopen(outname,"w");
#ifndef INDEPENDENT
      if (out) {
            fprintf(out,"%s\n",TreatInfo->treatcode);
            fprintf(out,"%f\n",2.0*(1.5-((float) TreatInfo->direction)));
            fprintf(out,"%d\n",TreatInfo->posture);
            /*(TreatInfo->posture==POSTURE_ERECT) ? "ERECT" : "SUPINE");*/
            fprintf(out,"%f\n", latency);
            fprintf(out,"%f\n", angle_latency);
      }
#endif

#ifdef INDEPENDENT
      printf("TotaTrigTime\t%d\nTotalExpTime\t%d\nReorients %d\n",
             total_vection,total_experiment,total_reorients);
      printf("Bin #\trange\t\tvection\tdropins\tdropouts\treorients\n");
#endif
      if (out) {
            fprintf(out,"%d\n%d\n",total_vection,total_experiment);
#ifdef WANT_BINS
            fprintf(out,"Bin #\trange\t\tvection\tdropins\tdropouts\treorients\n");
#endif /*WANT_BINS*/

      }

      for (cnt=0;cnt<RESOLUTION;cnt++) {
#ifdef INDEPENDENT
            printf("%d\t%6.3f\t%6.3f\t%d\t%d\t%d\t%d\n",cnt,SECANGLEBEG(cnt),
                   SECANGLEEND(cnt),vectionbins[cnt],dropinbins[cnt],
                   dropoutbins[cnt],reorientbins[cnt]);
```

```
#endif
#ifdef WANT_BINS
            if (out)
fprintf(out,"%d\t%6.3f\t%6.3f\t%d\t%d\t%d\t%d\n",cnt,SECANGLEBEG(cnt),
                        SECANGLEEND(cnt),vectionbins[cnt],dropinbins[cnt],
                        dropoutbins[cnt],reorientbins[cnt]);
#endif
      }
      for (cnt=0;cnt<ORESOLUTION;cnt++) {
#ifdef INDEPENDENT
            printf("%d\t%6.3f\t%6.3f\t%d\t%d\t%d\t%d\n",cnt,SECANGLEBEG(cnt),
                  SECANGLEEND(cnt),ovectionbins[cnt],odropinbins[cnt],
                  odropoutbins[cnt],oreorientbins[cnt]);
#endif
#ifdef WANT_BINS
            if (out)
fprintf(out,"%d\t%6.3f\t%6.3f\t%d\t%d\t%d\t%d\n",cnt,SECANGLEBEG(cnt),
                  SECANGLEEND(cnt),ovectionbins[cnt],odropinbins[cnt],
                  odropoutbins[cnt],oreorientbins[cnt]);
#endif
      }
      if (out){
#ifdef WANT_RAW
            fprintf(out,"%d\n",num_dropins);
            for (cnt=0;cnt<MAX_CELLS;cnt++){

#ifdef IN_SECONDS
                  fprintf(out,"%9.6f\t%9.6f\n",
                        (dropins[cnt].time==0) ? 0.0 :
                        ((float) dropins[cnt].time-TimeData(0))/
                              ((float)CLOCKS_PER_SEC)-1.0,
                        dropins[cnt].angle);
#else
                  fprintf(out,"%ld\t%9.6f\n",
                        dropins[cnt].time,dropins[cnt].angle);
#endif
            }
            fprintf(out,"%d\n",num_dropouts);
            for (cnt=0;cnt<MAX_CELLS;cnt++){
#ifdef IN_SECONDS
                  fprintf(out,"%9.6f\t%9.6f\n",
                        (dropouts[cnt].time==0) ? 0.0 :
                        ((float) dropouts[cnt].time-TimeData(0))/
                              ((float)CLOCKS_PER_SEC)-1.0,
                        dropouts[cnt].angle);
#else
                  fprintf(out,"%ld\t%9.6f\n",
                        dropouts[cnt].time,dropouts[cnt].angle);
#endif
            }
            fprintf(out,"%ld\n",total_reorients);
            for (cnt=0;cnt<MAX_CELLS;cnt++){
#ifdef IN_SECONDS
                  fprintf(out,"%9.6f\t%9.6f\t%d\n",
                        (reorients[cnt].time==0) ? 0.0 :
                        (float) (reorients[cnt].time-TimeData(0))/
                              ((float)CLOCKS_PER_SEC)-1.0,
                        reorients[cnt].angle,reorient_wall[cnt]);
```

```
#else
                  fprintf(out,"%ld\t%9.6f\t%d\n",

      reorients[cnt].time,reorients[cnt].angle,reorient_wall[cnt]);
#endif      /*IN_SECONDS*/
            }
#endif

#ifdef WANT_VERIFICATION
            fprintf(out,"Vection\n");
            fprintf(out,"%ld\n",GetDNE());
            for (cnt=0;cnt<GetDNE();cnt++){
                  fprintf(out,"%ld\t%9.6f\n",
                        TimeData(cnt),AngleData(cnt));
            }
            fprintf(out,"Reinterpretations\n");
            fprintf(out,"%ld\n",GetDNE2());
            for (cnt=0;cnt<GetDNE2();cnt++){
            fprintf(out,"%ld\t%9.3f\n",
                  TimeData2(cnt),AngleData2(cnt));
            }
#endif
      }
      if (out) fclose(out);
}

int   relative(float angle, int secnum){
      /* -1 means lower, 0 means inside, 1 means higher */
      if (angle<SECANGLEBEG(secnum))
            return -1;
      else if (angle<SECANGLEEND(secnum))
            return 0;
      else return 1;
}

int   whichsector(float angle){
      /* find out which sector it falls in            */
      /* can use binary search mechanism        */
      /* this works if Resolution is a multiple of 4  */
      /* Dont think it works in any other case  */

      int which = RESOLUTION/2-1,minw=0,maxw=RESOLUTION-1,stop=0,rel;
      while (!stop) {
#if 0
            printf("angle, which, min, max = %f, %d, %d,
%d\n",angle,which,minw,maxw);
#endif
            /*which is the middle sector*/
            if ((rel=relative(angle,which))==0) return which;

            else if (rel<0) {
                  maxw=which-1;
                  which=maxw-(maxw-minw)/2;
            }
            else if (rel>0) {
                  minw=which+1;
                  which=minw+(maxw-minw)/2;
            }
```

```
	}
}

int owhichsector(float angle){
	/* find out which sector it falls in		*/
	/* can use binary search mechanism	*/
	/* this works if Resolution is a multiple of 4	*/
	/* Dont think it works in any other case	*/
	/* The code is identical to whichsector(...)	*/
	/* since all that changes is the maxw and	*/
	/* initial which			*/

	int which = ORESOLUTION/2-1,minw=0,maxw=ORESOLUTION-1,stop=0,rel;
	while (!stop) {
#if 0
		printf("angle, which, min, max = %f, %d, %d,
%d\n",angle,which,minw,maxw);
#endif
		/*which is the middle sector*/
		/*Some angles are just beyond the last sector, but lets just
		  relegate them to the last sector
		  */
/*		if (which>=ORESOLUTION) {printf("which >= resolution\n");getch();}*/
		if ((rel=relative(angle,which))==0) return (which<ORESOLUTION) ? which :
(ORESOLUTION-1);
		else if (rel<0) {
			maxw=which-1;
			which=maxw-(maxw-minw)/2;
		}
		else if (rel>0) {
			minw=which+1;
			which=minw+(maxw-minw)/2;
		}
	}
}


void fixangles(float *beg, float *end, int *numrevs){
	float diff=*end-*beg;	/*diff might be greater than 2PI but it is not
				always positive (depends on rotation
				direction*/
	*numrevs=0;
	if (diff<0) diff *= -1.0; /*use the absolute difference*/
	while (diff>2.0*PI){
		diff-=2.0*PI;
		(*numrevs)++;
	}

	myprint("fixangles beg = %f\t end = %f\n",*beg,*end);

	/* make sure angles are within 0 to 2*PI */
	while (*beg>2.0*PI) (*beg)-=2.0*PI;
	while (*beg<-2.0*PI) (*beg)+=2.0*PI;
	while (*end>2.0*PI) (*end)-=2.0*PI;
	while (*end<-2.0*PI) (*end)+=2.0*PI;
}

void IncrAllBins(int bins[],int numrevs){
```

```
	int count;
	for (count=0;count<RESOLUTION;count++)
		bins[count]+=numrevs;
}
void IncrBins(int bins[],int from,int to,int numrevs){
	int cnt;
	for (cnt=from;cnt<=to;cnt++) bins[cnt]+=numrevs;
}

int pastthreshold(float arc){
	/*
	   if arc is greater or equal to the threshold arc then return 1
	   else return 0
	   */
	return (arc>=THRESHOLD_ARC);
}


int   performstats(char *name
#ifdef INDEPENDENT
		){
#else
	,TreatmentInfoRec *atreatinfo){
#endif
	/* 1/11/95 Now performs stats on dataread and dataread2 variables*/
	/* need to put experiment name support in*/
	int cnt_exp;
	FILE *expfile;
	int nents;
	RECTYPE *dread=NULL;
#ifndef INDEPENDENT
	TreatInfo = atreatinfo;
	/*
	   printf("TreatInfo->treatcode = %s\n",TreatInfo->treatcode);
	   printf("%d\n",TreatInfo->direction);
	   printf("%s\n",(TreatInfo->posture==POSTURE_ERECT) ? "ERECT" : "SUPINE");
	   */
#endif
	openexpread(name);
#ifdef INDEPENDENT
	printf("doing stats on experiment %s, press a key\n",name);
	getch();
#endif
	if (dataread && dataread2) {
		performstats_1();
		performstats2_1();
		printstats(name);
		closeexpread();
		return 1;
	}
	else {
		printf("openexpread: data not available%s\n",name);
		return 0;
	}
}
```

10.3.25 myprint.c

```
#include <stdio.h>
#include "myprint.h"
int myprint(char *fmt, ...){
      va_list args;
#ifdef ADAM_DEBUGGING
      va_start(args,fmt);
      return(vprintf(fmt,args));
      va_end(args);
#else
      return 0;
#endif
}
```

10.3.26 dirutils.c

```
#include<stdio.h>
#include<string.h>
#include"dirutils.h"
#define MAX_PATH 1024
_DIR *directory=NULL;

int   SetDirectory(char *dirname){
      if (directory) _closedir(directory);
      directory = _opendir(dirname);
      if (!directory) {
            printf("Unable to open directory %s\n",dirname);
            return 0;
      }
      else return 1;
}

void ListDirectory(int withnumbers){
      int   cnt=1;
      struct _dirent *dir_entry;

      dir_entry = _readdir(directory);
      while (dir_entry=_readdir(directory)){
            if (withnumbers) printf("%d.\t",cnt);
            cnt++;
            printf("%s\n",dir_entry->d_name);
      }
}

struct _dirent *GetNthEntry(int N){
      struct _dirent *nthdir=NULL;
      if (!directory) {
            printf("DIRUTILS: Error, directory not set.\n");
            return NULL;
      }
      _rewinddir(directory);
      nthdir = _readdir(directory);
      N--;
      while (N && nthdir) {
            nthdir= _readdir(directory);
            N--;
```

```
        }
        return nthdir;
}

int     GetNthEntryName(int N,char *name){
        struct _dirent      *dir = GetNthEntry(N);
        if (dir) {
                strcpy(name,dir->d_name);
                return 1;
        }
        else return 0;
}

void dirwalk(char *dir, void (*fcn) (char *)){
        char name[MAX_PATH];
        struct _dirent *dp;
        _DIR *dfd;

        if ((dfd = _opendir(dir)) == NULL) {
                fprintf(stderr, "dirwalk: can't open %s\n",dir);
                return;
        }
        while ((dp = _readdir(dfd)) != NULL){
                if (strcmp(dp->d_name,".")==0
                    || strcmp(dp->d_name,"..")==0)
                        continue; /*skip self and parent*/
                if (strlen(dir)+strlen(dp->d_name)+2>sizeof(name))
                        fprintf(stderr,"dirwalk: name %s/%s too long\n",
                                dir,dp->d_name);
                else {
                        sprintf(name,"%s/%s",dir,dp->d_name);
                        (*fcn) (name);
                }
        }
        _closedir(dfd);
}
```