Behavioural morphisms in virtual environments

This item was submitted to Loughborough University's Institutional Repository by the/an author.

Additional Information:

• A Doctoral Thesis. Submitted in partial fulfilment of the requirements for the award of the degree of Doctor of Philosophy at Loughborough University.

Metadata Record: https://dspace.lboro.ac.uk/2134/36075

Publisher: © Simon Peter Nee

Rights: This work is made available according to the conditions of the Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International (CC BY-NC-ND 4.0) licence. Full details of this licence are available at: https://creativecommons.org/licenses/by-nc-nd/4.0/

Please cite the published version.
Behavioural Morphisms In Virtual Environments

by

Simon Peter Nee

A Doctoral Thesis

Submitted in partial fulfilment of the requirements for the award of

Doctor of Philosophy of Loughborough University

September 2001

(c) By Simon Peter Nee 2001
Abstract

One of the largest application domains for Virtual Reality lies in simulating the Real World. Contemporary applications of virtual environments include training devices for surgery, component assembly and maintenance, all of which require a high fidelity reproduction of psychomotor skills. One extremely important research question in this field is:

"How closely does our facsimile of a real task in a virtual environment reproduce that task?"

At present the field of Virtual Reality is answering this question in subjective terms by the concept of presence and in objective terms by measures of task performance or training effectiveness ratios. Rather than taking this approach this thesis considers a different perspective based within a HCI framework. The task representation is a 'model world metaphor' that uses 'direct manipulation' as its paradigm of interaction. By using these HCI terms the current research question, restated, becomes:

"How narrow are the gulfs of evaluation and execution in this interface?"

This thesis starts a new research thread by presenting a behavioural fidelity evaluation methodology that allows researchers to assess the fidelity of motor skill tasks in a virtual environment by focusing on task performance and human behaviour. By using human behaviour the problem of generalising results across varying experimental platforms is resolved as the datum is independent of the underlying technology.

The methodology used is called 'behavioural morphism'; behavioural morphism can be considered a vector consisting of three terms; behavioural correlation, error and performance. This thesis details the methods and experiments that were used to evaluate the use of behavioural morphism as an assessment methodology.
The main hypothesis behind behavioural morphism is that a psychological model and the original data used to support the establishment of that model or empirically collected data can be used to determine how closely a virtual task reproduces the real world task. The studies in this thesis show that there are significant differences between the real and virtual behaviour that would not become apparent using non-behavioural assessment methodologies.

This thesis shows that it is possible for real and virtual psychomotor performance and behaviour to vary significantly. Studies by other researchers have demonstrated kinaesthetic adaptation to virtual environments. This indicates that an assumption that virtual training environments provide an isomorphic task mapping due to a similarity in virtual interaction and real world interaction is incorrect.
Dedication

In Memory Of

George, my grandfather who left early in my research.

And

Josephine, my grandmother who left the month before I finished.

In Celebration Of

Mala, my wife.

And

Lorcan, my son and most original contribution
Acknowledgments

I want to thank Laurie and Gill, my parents who have been unwavering sources of support for me. I’d also like to thank my Aunt Olive for all her help and Roy Kalawsky, my Ph.D. supervisor for all his help, advice, encouragement and friendship over the last few years.

There are stages in any prolonged project when you inevitably start to wonder if you will ever reach the end, for these periods I’d like to thank all of my friends for being there with food, beer or whatever it was I needed to recharge my batteries. I would also like to thank the technicians in the Human Sciences Department, especially Dave Harris, for their invaluable contribution to my research in constructing the electronic circuits used in this thesis.

Finally, I would like to thank and acknowledge the invaluable financial support I received from British Aerospace Ltd (now BAe Systems Ltd) and to thank Professor Steve Grigg, Mr Nic Bealey and Mr Norman Logie personally for their commitment to my research.
### TABLE OF CONTENTS

1 **CHAPTER 1: THESIS INTRODUCTION** ................................................................. 8
   1.1 **THESIS INTRODUCTION** ............................................................................ 9
   1.2 **BACKGROUND** ......................................................................................... 11
      1.2.1 **Definitions** ....................................................................................... 11
      1.2.2 **Virtual Environment Interface Modes** ............................................. 12
      1.2.3 **Immersive** .......................................................................................... 13
      1.2.4 **Semi-immersive** ................................................................................ 14
      1.2.5 **Non-immersive** ................................................................................. 15
      1.2.6 **Computer Graphics** .......................................................................... 16
      1.2.7 **Object Representation** ...................................................................... 18
      1.2.8 **Surface Geometry** ............................................................................. 19
      1.2.9 **Surface Material** ............................................................................... 21
      1.2.10 **Surface Texture Maps** ...................................................................... 21
      1.2.11 **Vertex Normals** ............................................................................... 24
      1.2.12 **Scene Representation** ...................................................................... 25
      1.2.13 **How It All Fits Together** ................................................................ 26
      1.2.14 **The Graphics Pipeline** .................................................................... 26
      1.2.15 **Algorithms** ....................................................................................... 27
   1.3 **HCI BACKGROUND** .................................................................................. 27
      1.3.1 **Norman's execution and evaluation cycle** .......................................... 27
      1.3.2 **The Model World Metaphor** .............................................................. 29
      1.3.3 **Direct Manipulation Interfaces** ......................................................... 30
      1.3.4 **Mental Models** .................................................................................. 32
      1.3.5 **The Complexity of Platform** ............................................................. 33
   1.4 **BACKGROUND DISCUSSION** ................................................................. 35
   1.5 **CONCLUSION** ........................................................................................ 36

2 **CHAPTER 2: LITERATURE REVIEW** .............................................................. 37
   2.1 **INTRODUCTION** ...................................................................................... 38
   2.2 **GENERAL READING** ............................................................................. 39
   2.3 **RELATED RESEARCH AREAS AND WORK** .......................................... 41
      2.3.1 **Virtual Presence** ............................................................................... 41
      2.3.2 **Usability** ............................................................................................ 47
      2.3.3 **Simulation Fidelity** ........................................................................... 49
      2.3.4 **Human Factors** ................................................................................ 52
      2.3.5 **Adaptation and After Effects** ............................................................. 56
      2.3.6 **Other Related Work** ......................................................................... 56
      2.3.6.1.1 **Specific Studies** ......................................................................... 56
   2.4 **DISCUSSION** .......................................................................................... 64
   2.5 **CONCLUSION** ........................................................................................ 66

3 **CHAPTER 3: BEHAVIOURAL MORPHISM** ................................................ 68
   3.1 **INTRODUCTION** ..................................................................................... 69
   3.2 **BEHAVIOURAL MORPHISM** ................................................................. 70
      3.2.1 **Definition of Terms** .......................................................................... 70
         Definition of Behaviour ............................................................................... 70
         Definition of Performance ......................................................................... 70
         Definition of Morphism .............................................................................. 70

Simon Nee 24/05/2002
5.2.4 Results Discussion ................................................................. 127
5.3 IMPLEMENTATION ................................................................... 128
  5.3.1 Abstraction and Implementation ....................................... 128
  5.3.2 Logic implementation ....................................................... 130
    5.3.2.1 Configuration Logic .................................................... 130
    5.3.2.2 Data Collection Logic and Output ............................... 135
  5.3.3 Verification and Validation .............................................. 138
5.4 CHAPTER CONCLUSION.......................................................... 140

6 CHAPTER 6: ROTARY PURSUIT TRACKING EXPERIMENT ........... 141
  6.1 INTRODUCTION ....................................................................... 142
  6.2 RATIONALE .............................................................................. 142
  6.3 EXPERIMENTATION ............................................................... 143
    6.3.1 Methodology ................................................................. 143
    6.3.2 Experimental Virtual Environment Design .................... 145
    6.3.3 Room Layout ................................................................. 146
    6.3.4 Apparatus and Calibration of Systems ......................... 147
      6.3.4.1 Virtual Set-Up ............................................................ 147
      6.3.4.2 Real Set-Up ............................................................... 147
    6.3.5 Data Collection ............................................................... 148
      6.3.5.1 Virtual ................................................................. 148
      6.3.5.2 Real ............................................................... 149
      6.3.5.3 Datasets ............................................................... 151
    6.3.6 Results ............................................................................. 160
      6.3.6.1 Homogeneity of Variance Assumptions .................... 163
      6.3.6.2 Repeated-Measures ANOVA Tables ......................... 165
        6.3.6.2.1 Performance ANOVA ........................................ 165
        6.3.6.2.2 Error ANOVA ................................................. 166
        6.3.6.2.3 Correlation ANOVA ....................................... 167
      6.3.7 Results Discussion ....................................................... 168
    6.4 IMPLEMENTATION ............................................................... 169
      6.4.1 Real Task ...................................................................... 169
        6.4.1.1 Design Abstraction and Implementation ................ 169
        6.4.1.2 Logic Implementation ............................................ 171
          6.4.1.2.1 Configuration Logic ....................................... 171
          6.4.1.2.2 Real Pursuit Tracking Data Collection Logic and Output 171
          6.4.1.2.3 User Tracking Implementation .......................... 171
        6.4.1.3 Verification and Validation .................................. 176
      6.4.2 Virtual Task .................................................................. 177
        6.4.2.1 Abstraction and Implementation ......................... 177
        6.4.2.2 Logic Implementation ............................................ 179
          6.4.2.2.1 Environment Configuration Logic .................... 179
          6.4.2.2.2 Data Collection Logic and Output .................... 179
          6.4.2.2.3 Verification and Validation ............................... 182
    6.5 CHAPTER CONCLUSION ....................................................... 183

7 CHAPTER 7: DATA REVIEW .......................................................... 185

Simon Nee 24/05/2002
7.1 INTRODUCTION ................................................................. 186
7.2 THESIS REVIEW ............................................................. 186
7.3 EVIDENCE FOR HYPOTHESIS SUPPORT ....................... 188
  7.3.1 Evidence for Support of Hypothesis 1 ......................... 188
  7.3.2 Evidence for Support of Hypothesis 2 ......................... 189
    7.3.2.1 Evidence for Support of Hypothesis 2:a .................. 189
    7.3.2.1.1 Choice Reaction .............................................. 189
    7.3.2.1.2 Rapid Aimed Movement ..................................... 190
    7.3.2.1.3 Rotary Pursuit Experiment ............................... 190
  7.3.2.2 Evidence for Support of Hypothesis 2:b .................... 191
  7.3.2.3 Evidence for Support of Hypothesis 2:c .................... 191
  7.3.3 Hypothesis Support Conclusion .................................. 192
8 CHAPTER 8: THESIS CONCLUSION ........................................... 193
  8.1 INTRODUCTION ............................................................. 194
  8.2 ORIGINAL CONTRIBUTION TO KNOWLEDGE ................. 194
  8.3 FURTHER WORK ............................................................ 195
  8.4 THESIS CONCLUSION .................................................... 197
9 REFERENCES ........................................................................ 200
10 APPENDIX 1 EXPERIMENTAL SOFTWARE ............................ 210
  10.1.1 Experimental Software Design .................................. 211
    10.1.1.1 Base Virtual Environment Software Architecture ....... 211
    10.1.1.1.1 Run-time Virtual Environment ......................... 212
    10.1.1.1.2 Software API ................................................. 213
    10.1.1.1.3 Geometry Tools ............................................. 214
11 APPENDIX 2 CHOICE REACTION ENVIRONMENT CODE .......... 217
12 APPENDIX 3 FITTS TAPPING ENVIRONMENT CODE ............... 229
13 APPENDIX 4 ROTARY PURSUIT ENVIRONMENT CODE ............ 241
14 APPENDIX 5 FITTS LAW RAW DATA OUTPUT ....................... 252
15 APPENDIX 6 CHOICE REACTION RAW DATA OUTPUT ............ 253
16 APPENDIX 7 ROTARY PURSUIT RAW DATA OUTPUT ............... 256
17 APPENDIX 8 ROTARY PURSUIT RAW DATA OUTPUT ............... 257
<table>
<thead>
<tr>
<th>FIGURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>A TECHNOLOGICALLY MEDIATED EXPERIENCE</td>
</tr>
<tr>
<td>1.2</td>
<td>A HEAD MOUNTED DISPLAY</td>
</tr>
<tr>
<td>1.3</td>
<td>SHUTTER GLASSES FOR FIELD SEQUENTIAL VIEWING (STEREO VIEWING)</td>
</tr>
<tr>
<td>1.4</td>
<td>VR PROJECTION SCREEN FOR SEMI-IMMERSIVE VES</td>
</tr>
<tr>
<td>1.5</td>
<td>A TYPICAL DESKTOP WORKSTATION (COURTESY SGI)</td>
</tr>
<tr>
<td>1.6</td>
<td>HIGH LEVEL CONCEPTUAL FRAMEWORK FOR COMPUTER GRAPHICS</td>
</tr>
<tr>
<td>1.7</td>
<td>ABSTRACT REPRESENTATION OF GEOMETRIC DATA</td>
</tr>
<tr>
<td>1.8</td>
<td>GEOMETRIC DATA</td>
</tr>
<tr>
<td>1.9</td>
<td>POLYGONAL REPRESENTATION OF A SPHERE</td>
</tr>
<tr>
<td>1.10</td>
<td>A 3D UNIT SQUARE PLANE</td>
</tr>
<tr>
<td>1.11</td>
<td>TEXTURED AND UNTEXTURED OBJECTS</td>
</tr>
<tr>
<td>1.12</td>
<td>A TEXTURE MAP OF THE EARTH</td>
</tr>
<tr>
<td>1.13</td>
<td>A 3D PLANES WITH VERTEX AND SURFACENormals</td>
</tr>
<tr>
<td>1.14</td>
<td>A SCENE DATABASE FOR A CHAIR</td>
</tr>
<tr>
<td>1.15</td>
<td>NORMAN'S GULF OF EXECUTION AND EVALUATION</td>
</tr>
<tr>
<td>1.16</td>
<td>BRIDGING THE GULFS OF EXECUTION AND EVALUATION</td>
</tr>
<tr>
<td>2.1</td>
<td>IVAN SUTHERLAND'S SKETCHPAD ON AN MIT TX-2 COMPUTER</td>
</tr>
<tr>
<td>2.2</td>
<td>IVAN SUTHERLAND'S SWORD OF DAMOCLES HMD</td>
</tr>
<tr>
<td>2.3</td>
<td>ZELTSER'S SALIENT COMPONENTS OF PRESENCE</td>
</tr>
<tr>
<td>2.4</td>
<td>KALAWSKY'S MULTIDIMENSIONAL PRESENCE MEASURE</td>
</tr>
<tr>
<td>2.5</td>
<td>GABBARD'S VE USABILITY TAXONOMY (TAKEN FROM GABBARD)</td>
</tr>
<tr>
<td>2.6</td>
<td>AREAS SURVEYED BY PADMOS AND MILDERS</td>
</tr>
<tr>
<td>2.7</td>
<td>AREAS SURVEYED BY RINALDUCCI</td>
</tr>
<tr>
<td>2.8</td>
<td>VEPAP SUITE OF BENCHMARKING TASKS</td>
</tr>
<tr>
<td>2.9</td>
<td>VRMAT MEASURES</td>
</tr>
<tr>
<td>2.10</td>
<td>INSTRUMENTED OBJECTS</td>
</tr>
<tr>
<td>2.11</td>
<td>TOWERS OF HANOI MOVEMENT PATHS</td>
</tr>
<tr>
<td>2.12</td>
<td>KUHLEN'S VIRTUAL/REAL CUBE GRASPING EXPERIMENT</td>
</tr>
<tr>
<td>2.13</td>
<td>KUHLEN'S REAL-VIRTUAL COMPARISONS</td>
</tr>
<tr>
<td>3.1</td>
<td>TYPE I AND TYPE II ERROR TABLE</td>
</tr>
<tr>
<td>3.2</td>
<td>EXAMPLE GRAPH OF PREDICTOR VERSUS MEASURED VALUE</td>
</tr>
<tr>
<td>3.3</td>
<td>BEHAVIOURAL MORPHISM STUDY SEQUENCE OUTLINE</td>
</tr>
<tr>
<td>3.4</td>
<td>REAL AND VIRTUAL PERFORMANCE FUNCTIONS</td>
</tr>
<tr>
<td>3.5</td>
<td>INTEGRATED REAL AND VIRTUAL PERFORMANCE FUNCTIONS</td>
</tr>
<tr>
<td>4.1</td>
<td>REACTION-TIME VIRTUAL ENVIRONMENT EXPERIMENTAL LAYOUT</td>
</tr>
<tr>
<td>4.2</td>
<td>STIMULUS AND RESPONSE BLOCK SETS (STIMULI = N = 8, CHOICE = 7)</td>
</tr>
<tr>
<td>4.3</td>
<td>VIRTUAL TRIAL (STIMULI = N = 8, CHOICE = 3)</td>
</tr>
<tr>
<td>4.4</td>
<td>THE BLOCK-TRIAL CYCLE</td>
</tr>
<tr>
<td>4.5</td>
<td>REGRESSION ANALYSIS (HOME BLOCK REACTION)</td>
</tr>
<tr>
<td>4.6</td>
<td>REGRESSION ANALYSIS (RESPONSE BLOCK REACTION)</td>
</tr>
<tr>
<td>4.7</td>
<td>CHOICE REACTION TRIAL ABSTRACTION</td>
</tr>
<tr>
<td>4.8</td>
<td>FILLARRAY() FUNCTION</td>
</tr>
<tr>
<td>4.9</td>
<td>RANDOMISEARRAY() FUNCTION</td>
</tr>
<tr>
<td>4.10</td>
<td>STIMULUS BLOCK ARRANGEMENT FUNCTION</td>
</tr>
<tr>
<td>4.11</td>
<td>USED BLOCK ARRANGEMENT FUNCTION</td>
</tr>
<tr>
<td>4.12</td>
<td>TOUCH FUNCTION CALLBACK</td>
</tr>
<tr>
<td>4.13</td>
<td>DATA FUNCTION CALLBACK</td>
</tr>
</tbody>
</table>

Simon Nee 24/05/2002
FIGURE 5.2 EXPERIMENTAL ENVIRONMENT WITH A COLLISION STIMULUS .................. 118
FIGURE 5.3 GRAPH OF CELL MEANS FOR CORRELATION ACROSS COLLISION CUE .... 122
FIGURE 5.4 GRAPH OF CELL MEANS FOR PERFORMANCE ACROSS COLLISION CUE .... 124
FIGURE 5.5 GRAPH OF CELL MEANS FOR ERROR RATE ACROSS COLLISION CUE .... 126
FIGURE 5.6 THE RAPID-AIMED MOVEMENT ENVIRONMENT LAYOUT ..................... 128
FIGURE 5.7 FITTS' LAW TRIAL ABSTRACTION ................................................. 129
CODE SECTION 5.8 RAPID-AIMED MOVEMENT TRIAL CONFIGURATION ................. 134
CODE SECTION 5.9 TRIAL DATA COLLECTION FUNCTION ................................... 136
CODE SECTION 5.10 RAPID-AIMED MOVEMENT DATA OUTPUT FUNCTION ............... 138
FIGURE 6.1 TRACKING TARGET ROTATION ..................................................... 143
FIGURE 6.2 REAL TRIAL .............................................................................. 144
FIGURE 6.3 VIRTUAL TRIAL .......................................................................... 144
FIGURE 6.4 STEREO-PAIR IMAGE OF 3D PURSUIT TESTER MODEL .................. 145
FIGURE 6.5 STEREO-PAIR IMAGE OF 3D PROBE MODEL .................................. 145
FIGURE 6.6 EXPERIMENTAL AREA SET-UP ................................................... 146
FIGURE 6.7 INSTRUMENTATION SYSTEM ...................................................... 150
FIGURE 6.8 OPTICAL ENCODER AND TRIGGER STIMULUS ......................... 151
FIGURE 6.9 REAL-WORLD PROBE DATASET ................................................. 152
FIGURE 6.10 REAL-WORLD ROTOR DATASET .............................................. 153
FIGURE 6.11 LOOK-UP TABLE RAW DATA SAMPLE, SINGLE PERIOD .......... 156
FIGURE 6.12 FINAL CALIBRATION SET-UP ...................................................... 159
FIGURE 6.13 REPEATED MEASURES PERFORMANCE ANOVA FACTORS CELL MEANS. 160
FIGURE 6.14 REPEATED MEASURES ERROR ANOVA CELL MEANS GRAPH ....... 161
FIGURE 6.15 REPEATED MEASURES BEHAVIOUR ANOVA CELL MEANS ............ 162
FIGURE 6.16 A REAL ENVIRONMENT ROTARY PURSUIT TRIAL ................. 169
FIGURE 6.17 REAL ROTARY PURSUIT TRIAL ABSTRACTION ......................... 170
FIGURE 6.18 ROTARY PURSUIT DATA COLLECTION SYSTEM .......................... 172
FIGURE 6.19 DATA COLLECTION LAYOUT .................................................... 173
FIGURE 6.20 OPTOELECTRONIC SENSOR POSITIONING .................................. 173
FIGURE 6.21 LASER PRINTER 'BAR CODE' ................................................... 174
FIGURE 6.22 PROBE AND TRACKER SENSOR ASSEMBLY .............................. 174
FIGURE 6.23 POLHEMUS FASTRAK SENSOR ASSEMBLIES AND FIXING ........ 175
FIGURE 6.24 PC INTERFACE CIRCUIT ......................................................... 175
FIGURE 6.25 LABVIEW PROGRAM ............................................................. 176
FIGURE 6.26 VIRTUAL ENVIRONMENT TRIAL SET UP ................................. 177
FIGURE 6.27 VIRTUAL PURSUIT TRACKING ABSTRACTION ........................... 178
CODE SECTION 6.28 STIMULUS TRACKING FUNCTION ............................... 179
CODE SECTION 6.29 VIRTUAL PURSUIT TRACKING DATA COLLECTION FUNCTION . 180
CODE SECTION 6.30 VIRTUAL PURSUIT TRACKING DATA OUTPUT FUNCTION ...... 182
FIGURE 10.1 dVSI/dVISE ARCHITECTURE .................................................. 213
FIGURE 10.2 EXAMPLE TRISTRIP .................................................................. 214
FIGURE 10.3 EXAMPLE TRIFAN .................................................................. 215
FIGURE 10.4 POLYGON DECIMATION .......................................................... 215
INDEX OF EQUATIONS

SCHLOERB'S OBJECTIVE PRESENCE MEASURE EQUATION ........................................ 44
SIMULATION UNORDERED PROPERTIES ................................................................. 51
SIMULATION ORDERED PROPERTIES ................................................................. 51
PROPERTY AND PURPOSES ASSOCIATION ......................................................... 51
A FIDELITY DIFFERENTIAL .................................................................................... 52
TOTAL SYSTEM FIDELITY .................................................................................... 52
NEWTON'S FIRST LAW OF MOTION ................................................................ 73
THE PEARSON PRODUCT MOMENT CORRELATION ........................................ 75
A LINEAR EQUATION .......................................................................................... 75
THE BEHAVIOURAL MORPHISM MODEL COMPONENT DEFINITION ................ 78
THE BEHAVIOURAL MORPHISM PERFORMANCE COMPONENT DEFINITION .... 80
THE BEHAVIOURAL MORPHISM ERROR COMPONENT DEFINITION .................. 82
THE HICK-HYMAN LAW ...................................................................................... 86
FITTS' LAW ......................................................................................................... 96
INDEX OF DIFFICULTY ....................................................................................... 115
INDEX OF PERFORMANCE .................................................................................. 115
TRANSFORMATION MATRIX ................................................................................ 154
ROTOR ANGLE TO POSITION MAPPING ............................................................. 155

INDEX OF TABLES

TABLE 4-1 CORRELATION TABLE FOR LEAVING HOME BLOCK ............................... 94
TABLE 4-2 CORRELATION TABLE FOR HITTING RESPONSE BLOCK ....................... 95
TABLE 4-3 UNTRANSFORMED SUBJECT CORRELATION COEFFICIENTS ............. 96
TABLE 4-4 TRANSFORMED SUBJECT CORRELATION COEFFICIENTS ................ 97
TABLE 4-5 HYPOTHESIS TEST (CORRELATION) .................................................... 97
TABLE 4-6 HYPOTHESIS TEST (PERFORMANCE) .................................................. 98
TABLE 4-7 CHOICE POSITION RANDOMLY CHOSEN ............................................. 104
TABLE 4-8 RANDOMISED BLOCK SET OF RANDOM CHOICE POSITIONS ............. 105
TABLE 5-1 T-TEST COMPARISON OF CORRELATION COEFFICIENTS ............... 121
TABLE 5-2 REPEATED MEASURES ANOVA TABLE FOR CORRELATION ............. 121
TABLE 5-3 REPEATED MEASURES ANOVA TABLE FOR PERFORMANCE .......... 123
TABLE 5-4 REPEATED MEASURES ANOVA TABLE FOR ERROR RATE ................. 125
TABLE 6-1 ASSUMPTION OF SPHERICITY TEST, PERFORMANCE ......................... 163
TABLE 6-2 ASSUMPTION OF SPHERICITY TEST, ERROR ...................................... 164
TABLE 6-3 ASSUMPTION OF SPHERICITY TEST, CORRELATION ......................... 164
TABLE 6-4 REPEATED-MEASURES ANOVA TABLE, PERFORMANCE .................. 165
TABLE 6-5 REPEATED-MEASURES ANOVA TABLE, ERROR ............................... 166
TABLE 6-6 REPEATED-MEASURES ANOVA, CORRELATION ................................. 167
TABLE 10-1 DVS/DVISE MAIN COMPONENTS ..................................................... 211

Simon Nee 24/05/2002
1 Chapter 1: Thesis Introduction
1.1 Thesis Introduction

This thesis is about human behaviour in virtual environments. It deals solely with virtual environments that are meant to mimic the real-world. These are known as mimetic virtual environments (Cutting 1997). The area of interest to this thesis is the closeness or fidelity of tasks carried out in a real environment to tasks such as pressing a button or tapping between two targets when the world is represented by a 3D computer model and the tasks are carried out in a virtual environment.

In essence this thesis is interested in psychological measures of human behaviour and performance and the study of those measures in virtual environments. This thesis shows how to measure the closeness of tasks carried out in the real-world and in a virtual-world by using the real-world psychological measures as a datum or reference mark with which to compare our virtual facsimiles.

This thesis uses a general review of virtual environments and a specific literature review of virtual environment performance measures to construct a problem statement. It then proposes a solution to this problem statement and provides several research hypotheses to test the solution. After that original experimentation will be described that test the research hypotheses and a conclusion about the problem solution is drawn.

The next section is an introduction to the field of virtual environments and it describes the software, hardware and art components of a virtual environment. The background section will also describe the different types of virtual environments and give a working definition of virtual environment for this thesis.

The next section provides a background to virtual reality and virtual environments. The field of virtual environments and virtual reality is not new, research related to it has been undertaken for at least 4 decades (Sutherland 1965), primarily by research communities within Computer graphics, human factors, the military and robotics. What is new is the affordability of state-of-the-art computing equipment and the
associated technology necessary to carry out research in the field. At present even affordable high performance computing equipment can become swamped by the computational, memory and I/O needs of a real-time virtual environment.

The constraints of available memory and computer instruction cycles mean that the judicious allocation of resource is critical to the success of a virtual reality project. Designing systems today requires the application of art rather than science; projects rely on the designer's experiential knowledge (Padmos 1992; Schloerb 1995; Rinalducci 1996; Stytz 1996; Stanney et al. 1998; Bowman 2001; Bowman et al. 2001) rather than a set of human factors requirements, heuristics rather than quantitative experimental evidence.
1.2 Background

1.2.1 Definitions

There are as many different definitions of virtual reality and of virtual environments as there are researchers in the field. The definition that most strikes a chord with this thesis since it explicitly mentions human behaviour and implies that the system must have a system model of the user is the following.

"Virtual Environments are three-dimensional, computer-generated, simulated environments that are rendered in real-time according to the behaviour of the user" (Loeffler 1994)

One must make the distinction at this point that virtual reality and a virtual environment are not synonymous. Virtual reality is a type of human computer interface to a virtual environment, this involves the tracking of limb movement and the projection of the computer generated environment in such a manner that the user has a subjective feeling of egocentric immersion or 'presence' (Heeter 1992; Sheridan 1992; Snow 1996; Draper et al. 1998) in the virtual environment. Virtual Reality is one of a number of virtual environment human computer interfaces detailed later in this chapter.

Some commentators try to extend the definition of virtual reality via a presence argument.

"Purists at one extreme will define virtual reality in terms of an experience, which can encompass dreams, hallucinations, trompes-l'oeil and even books and films which absorb attention. Purists at the other extreme will insist on using definitions which derive from technology, such as a computer-generated environment, or a head-mounted display." (Carr and England 1995)

There is some merit to Carr's definition if one takes the view that creating the subjective feeling of immersion or 'presence' is the overriding aim of a virtual environment. One can see books, films and television all have the capacity to absorb
and focus attention. However, it could be argued that these are examples of flow experiences (Csikszentmihalyi 1975; Latta and Oberg 1994) that have, in terms of a simulation environment, a high psychological fidelity (Meister 1990).

In a paper that provided a taxonomy for synthetic experiences Robinett (Robinett 1992) discusses virtual experiences and concludes that they are technologically mediated experiences (Figure 1.1). This thesis takes a pragmatic approach to virtual environments and its definition of a virtual environment clearly falls in with the later group of purists mentioned by Carr. However, no pragmatic approach can be purist.

![Figure 1.1 A technologically mediated experience](image)

1.2.2 Virtual Environment Interface Modes

There are three main types of virtual environment human computer interface (Kalawsky 1993) immersive, semi-immersive and desktop. This thesis concentrates on the desktop variety but the methods of behavioural morphism developed in this thesis are applicable to all types of VE (Virtual Environment). These categorizations are loose and blur the technology and experience to some degree but are a good reference point when understanding the hardware that is available to present a virtual environment to a participant.
1.2.3 Immersive

This type of interface is characterised by the egocentric immersion of the user within a 3D virtual environment. This immersion is usually achieved using a HMD (Head Mounted Display) or CAVE (CAVE Automatic Virtual Environment) that presents a constrained, sampled approximation of the ambient optic array (Gibson 1979) that would be presented by the VE were it real. Characteristics of an immersive system are as follows.

- Visually coupled
- Large Field of view from 30° x 30° - 140° x 140° dependent on HMD
- 360° field of regard
- Stereoscopic/Monoscopic
- 6DOF Trackers

Figure 1.2 A Head Mounted Display

Immersing oneself in a virtual environment using datagloves, HMD's and 3D tracking equipment is the experience many call Virtual Reality. Ones limbs and head would be tracked via 6DOF (Six Degree of freedom.) sensors such as the ‘Flock of Birds’ or ‘Fastrak’ systems
1.2.4 Semi-immersive

Semi-immersive virtual environments are generally projected onto large screens. Images are presented in a field sequential manner, this means by using LCD shutter glasses different images are presented to each and enable the user to create a stereo image. A LCD cell in the eyepiece of the glasses achieves the blocking of the image from the eye, when a voltage is applied to a cell it becomes opaque because of an effect similar to two polarising filters being twisted in front of each other. Characteristics of semi-immersive environments are listed below.

- Stereoscopic
- Large screen projection
- Large Field of view up to 180° X 180°
- Field of regard ~ 150°
- Motion parallax with head movement if head tracked.

Figure 1.3 Shutter glasses for field sequential viewing (Stereo viewing)
1.2.5 Non-immersive

Non-immersive virtual environments are also known as desktop or fish-tank virtual environments. There is argument about whether a degree of immersion is evident in the user of a desktop virtual environment. The desktop VE is based on a graphics workstation. Presently, a high performance PC can deliver enough power to allow a reasonably good virtual environment to be implemented. Typically the user will interact using the conventional mouse, however, other devices such as spaceballs, datagloves and 3D mice can be used to interact. Desktop VEs may be observed with or without stereo. Typical non-immersive environment characteristics are:

- Stereoscopic or monoscopic
- Field of view = field of regards
- Field of regard ~ 50°
- No visual coupling
For a comprehensive overview of the technology used in the presentation of virtual environments in the late 1990’s see (Youngblut, Johnson et al. 1996).

1.2.6 Computer Graphics

In order to understand many different issues that may be involved in constructing a virtual environment one must first understand how to construct computer graphic models and how the technology creates the virtual environment. The following sections will deal with the federation of data objects and technology that underpin a virtual environment. The system dealt with will be a generic stereo desktop environment however this federation is a subset of the immersive and semi-immersive federations. To understand how a computer displays a virtual environment we need to understand that a virtual environment displayed on a desktop computer screen is the result of a series of algorithms operating on some dataset held within the computer's memory. The information needed to display a virtual environment is a structured arrangement of single object data, these data are organised in a manner that allows rapid transformations to occur.

In the next section we deal with how individual objects and scene databases are represent, for a more comprehensive overview of 3D computer graphics and 3D graphics in virtual environments virtual environments consult (Hearn and Baker 1986; Foley et al. 1990; Green and Sun 1995).
Figure 1.6 High level conceptual framework for computer graphics

Figure 1.6 above shows a high level conceptual framework for a virtual environment. The application model is the data that the virtual environment being displayed represents; this could be a CAD model, a flight simulation database or any other type of 3D database. The application model consists of all the geometric entities in the virtual environment such as object data and scene data; these data describe the objects and their position in the virtual environment. The application program determines how the objects in the scene behave by controlling their physics and interactivity with the user. The graphics system is then responsible for drawing any part of the scene that is visible to the display.
1.2.7 Object Representation

Objects in current VEs tend to be modelled using polygonal surface geometry, surface material data, surface texture data and vertex normals. Figure 1.7 below is an abstract representation of the data needed to represent a single object in computer graphics.

Figure 1.7 Abstract representation of geometric data entity

Figure 1.8 Geometric data
1.2.8 Surface Geometry

At present the fastest way to model the surface shape of an object is to use a polygonal representation, as a result most modern graphics systems are optimised for functional primitives such as triangles and quads (Kovach 1999). This is mainly the result of the need to optimise data structures so algorithms can act as quickly as possible. An example of a polygonal representation is the cube. It is made up of six sides each containing 4 vertices. Other methods of representing surfaces do exist such as splines, these are mathematically parametric in nature using piecewise polynomial functions (Hearn and Baker 1986; Foley et al. 1990; Hill jnr 2000). Splines are not generally optimised in computer hardware and are not used for time critical applications. The geometry of an object defines both its size and shape.

![Polygonal representation of a sphere](image)

**Figure 1.9 Polygonal representation of a sphere**

The polygon is represented within the computer as a set of 3D co-ordinates in Cartesian space that bound a 3D plane. The dataset of a unit square could be held in the computer as \((0,0,0),(1,0,0),(1,1,0),(0,1,0)\). This dataset is referred to as the
geometry of the object and as seen in (Figure 1.9) above can have varying polygonal complexity. The number of polygons or vertices used to represent the surface can be critical for the effective representation of certain types of surface.

![Figure 1.10 A 3D unit square plane](image)

From (Figure 1.10 A 3D unit square plane) above we can see that a unit square polygon can be represented by an ordered set of vertices.

```
000 010 110 100
```

A 3d unit cube consists of six polygons with some common vertices i.e.

```
000 100 110 010
000 100 101 001
000 010 011 001
100 101 111 110
010 110 111 011
101 001 011 111
```
This is how geometry models are stored within the computer. Combinations of the polygon primitives are used to represent higher order real-world objects.

1.2.9 Surface Material

As we can see in Figure 1.9 above the sphere has a material associated with it, in this case a purple smooth shaded, a blue flat shaded and a dark blue wire frame materials have been applied. This material attribute is stored in the objects dataset as well and defines the way the surface interacts with light sources. The main material attributes are:

- Ambient
- Diffuse
- Specular
- Emissive
- Opacity

The data provided to the computer via these attributes allow the scene rendering algorithms to be lit and shaded correctly.

1.2.10 Surface Texture Maps

Virtual environments with a great degree of realism are only possible through the use of surface texture maps. Without surface texture maps the fine or high frequency detail of an object would have to be provided via the polygon model increasing the polygonal complexity and computational cost of the object. The realism is achieved is by applying a texture map to the surface of an object.
Amongst the object dataset is an attribute that relates a texture and a method of mapping it to the surface of an object. Texture mapping is a way we can increase the
realism of an object without increasing its polygonal complexity Figure 1.11. A texture map is shown in Figure 1.12 above.
1.2.11 Vertex Normals

Vertex normals are data entities that are stored with the surface normal data, they are vectors that are perpendicular to the surface at that vertex position they are associated with.

Figure 1.13 A 3D planes with vertex and surface normals

Vertex normals are used in shading objects at run time. The data merely provides the computer with the information to achieve shading, whether the computer uses it therefore, is related to the software being run. The data are either inferred from or specified with the vertex data. There are several shading techniques (Hearn and Baker 1986; Foley et al. 1990; Hill jnr 2000).

- Flat
- Gouraud (Smooth)
- Phong (Specular)

The Gouraud and Phong shading provide more realistic graphics but at a performance penalty that impacts on overall computer system performance.
1.2.12 Scene Representation

The most popular technique to represent scenes is a hierarchical tree like data structure known as a scene database (Foley et al. 1990; Hill jnr 2000). Other methods do exist such as volumetric or voxel (Volume element cf. (picture element or pixel) based modelling but these are not used in the systems available for this project.

The scenegraph is the abstract representation of a series of objects and their geometric relations. The scenegraph data structure is a tree structure and can therefore be represented by a directed acyclic graph (Gersting 1993). This abstract representation is of objects organised in a manner that allows complicated objects to be built up of simpler objects. For instance a chair could be made up as follows.

![Figure 1.14 A scene database for a chair](image)

From this basic data structure entire virtual worlds can be created, manipulated and organised. At the most basic level this database organises the object data in the form shown in Figure 1.14 in computer memory.

Each object in the database is a node in the scenegraph tree. The object database will also allow objects other than physical objects to be stored. The database can include
transformation matrices, sound objects, and light sources to be placed into the scene. Transformation matrices allow the designer to manipulate object positions and orientations in the VE. Sound objects allow waveforms to be incorporated and light sources allow multiple light sources to be positioned in various parts of the scene if needed.

1.2.13 How It All Fits Together

Once a designer has designed a virtual environment using a design tool such as MultiGen (1994) or 3DStudioMax the data are saved in a database as described above. A visual simulation application program loads the database; this application provides the virtual environment at run-time. The application will enable the data to be passed to the computer's graphics pipeline and the virtual environment drawn to a display device.

1.2.14 The Graphics Pipeline

The generic graphics pipeline for virtual environment can be thought of as consisting of three functions: App, cull and draw.

- App
  Application accepts inputs, simulates system dynamics, evaluates interactions between objects and updates the visual database with the users' actions.

- Cull
  Traverses the visual database and determines what is visible to the user. Selects levels of detail sorts, optimises the state details of the objects and generated a display list for the draw functions.

- Draw
  Traverses the display list and issues graphics commands to a geometry pipeline that creates an image for display on the requisite device.

We can see from the above descriptions that to represent an object we need the following data.

- Geometry
Behavioural Morphisms In Virtual Environments

- Size
- Shape
- Material
- Surface light interactions
- Texture
- Realistic appearance without polygonal complexity
- Vertex normals
- Shading of objects

These data and attributes when passed from memory to algorithms enable the computer system to draw the objects within a virtual environment. The data are then used by the computer system to draw a virtual environment; this means the performance of the environment is reliant on both the data and efficiency of the algorithms.

1.2.15 Algorithms

The software supporting a virtual environment system is complex. There will be algorithms controlling the scene database, rendering, culling, simulation and input-output. Beneath the application software will be another layer of software, the operating system.

1.3 HCI Background

1.3.1 Norman’s execution and evaluation cycle

Norman’s execution and evaluation cycle of interaction (Norman 1986; Norman 1988) is a ubiquitous model of interaction in the field of HCI. Summarised, the model states that the user of a computer formulates a plan of action and executes that plan, the user then observes the computer interface to evaluate the result of the executed plan. A more detailed breakdown of the cycle is given below.

- Execution
  - Establishing the goal.
  - Forming the intention.
• Specifying the action sequence.
• Executing the action.

• Evaluation
  • Perceiving the system state
  • Interpreting the system state
  • Evaluating the system state with respect to goals and intentions

To hypothesise why some interfaces are difficult to use Norman introduces two concepts: the *Gulf of Execution* and the *Gulf of Evaluation*. The Gulf of Execution is present due to a mismatch in the task language of the user and the core language of the system. The Gulf of Execution can be thought of as a notional distance between the collection of intended actions of the user and the collection of available actions the system allows.

![Figure 1.15 Norman’s Gulf Of Execution and Evaluation](image)

The gulf of evaluation is a measure of the distance between the representation given to the user of the physical state of the system and the user’s expectation. If the physical representation of the system given to the user allows easy evaluation of the result of an action in terms of the user’s goals then the gulf of evaluation is small. An appropriate output metaphor enables the user to build a good conceptual model of the physical system.
As pointed out by Norman the designer or user of a system can bridge the gulfs of execution and evaluation. The designer can bridge the gulfs by designing the input and output of the system to reflect the psychological requirements of the user of the system. The user bridges the gulfs by creating plans, strategies and by learning to interpret the system state efficiently.

One can see that the interaction language needed to delete a file in a MS-DOS based computer system is not as intuitive as the interaction language needed for the task of deleting a file under the Windows 2000 system. The interaction language under MS-DOS contains complicated lexical, syntactic and semantic structures that are not present in the action of dragging a file to the Trashcan.

### 1.3.2 The Model World Metaphor

The *model world metaphor* (Hutchins 1986) describes a computer interface in which the objects and actions mirror the objects and actions in some real-world domain. Instead of the interaction between the user and the computer being based on a conversational it is based on the user ‘acting’ upon the object within the model. The world changes because of actions of the user on interface objects that represent objects in the model world.
Using the model world metaphor means that the input and languages represent the subject of the interaction in a manner that allows the user to assume that the input and output objects are the entities they refer to in the model world.

"In a system built on the model world metaphor, the interface itself is a world where the user can act, and which changes state in response to user actions. The world of interest is explicitly represented and there is no intermediary between user and world. Appropriate use of the model-world metaphor can create the sensation in the user of acting upon the objects of the task domain themselves. We call this aspect of directness direct engagement." (Norman 1986)

The model world metaphor gives the interface a new role; instead of mediating between systems the user can view the interface as the system or model world in its entirety.

1.3.3 Direct Manipulation Interfaces

Shneiderman coined the term ‘direct manipulation’ to describe a set of interfaces that share the following characteristics (Shneiderman 1983; Shneiderman 1998).

- Continuous representation of the objects and actions of interest with meaningful visual metaphors
- Reversible, incremental actions whose effect on the object of interest is rapidly visible
- Replacement of complex command language syntax by a direct manipulation of the object of interest

The earliest example of an interface that alluded to direct manipulation is thought to be ‘Sketchpad’, a basic graphical design program. (Sutherland 1963). Ivan Sutherland is also attributed with the original idea of immersing the user in computer-generated environments in his paper ‘The Ultimate Display’. (Sutherland 1965) There are two main concepts of importance in the ‘directness’ of a direct manipulation interface. (Hutchins et al. 1986)

- Directness = Distance + Engagement
- Distance
• The notional distance between the task in the user’s mind and the way the task will be accomplished by using the interface provided. (cf. Gulf of Execution and Evaluation)

• Engagement
  • The qualitative feeling one is directly manipulating the object of interest (cf. Presence)

An example of direct manipulation interaction is the deletion of a file using the Windows 2000 desktop interface (The immediate file deletion option must be chosen). The user selects a file by clicking on its visual metaphor, holds the right mouse button down and drags it the file visual metaphor to the action metaphor (visual) of the trashcan.

Distance relates to the interface language that the interface uses. Whenever we interact with an input device we are using the input language and whenever we observe the output we are using the output language. Moving a mouse so a cursor has selected a file is an example of using the input language, observing the cursor and stopping it from moving once the goal has been reached is an example of using the output language.

An interface language has two properties that describe it in terms of the input and output languages of the system.

• Semantic directness
• Articulatory directness

Semantic directness refers to the relationship between what a user needs to convey and the set of expressions available at the interface. If the interface expressions require the user to alter their conceptual model of the task to achieve a goal or does not allow them to express intention concisely then the semantic directness of the interface is poor.
Articulatory directness concerns the meanings of the expressions at the interface and the physical form required to ‘act’ them out. (Preece et al. 1994) and (Hutchins et al. 1986) give the same example of articulatory directness in the form of a moped turn indicator switch that is located on the left-hand handlebar. To turn left the switch must be pulled backwards, and to turn right the switch must be pushed forward. The switch is semantically direct as it has a single state for each of the possible goals the user may want to communicate to the motorcycles electrical system. The switch is also articulatorially direct if one understands that the direction the switch moves in mirrors the direction the handlebars must be moved to turn in the indicated direction.

1.3.4 Mental Models

A user has expectations of a system’s behaviour due to their ‘mental model’ of the system they are using. A system will have an expectation of the behaviour of the user called the ‘user model’. Allen (1997) has written a good review of the main concepts found in the literature on mental models.

The mental model is made up of the user’s analogs of the real-world processes the system is carrying out. Clearly, the mental model of a user is not observable and can only be modified indirectly by training the user.

The user model is held in the computer software and can be changed readily. The user model can be constructed to take parameters that allow the computer to distinguish between users; these input parameters can be set by the user or inferred by the system through behaviour or responses. The degree of personalisation available will vary from baserate predictions derived from statistical populations to individualised models.

User models are being developed for virtual environments but these have tended to concentrate on perceptual models of the Human and not behavioural or cognitive (Rix 1999; Kalawsky 2000). We can consider the perceptual model as the programmer attempting to bridge the gulf of evaluation by adjusting the systems user model, by including a behavioural aspect to the user models the gulf of execution may also be bridged. Some models, which contain behavioural and perceptual facets, do exist such
as the 'Model Human Processor' (Card et al. 1983) but their applicability to virtual environments is yet to be explored.

1.3.5 The Complexity of Platform

Reproducibility is one of the cornerstones of modern science and as such is a fundamental problem for the field of computer-mediated experiences. Establishing a common virtual environment platform for research is a difficult if not impossible task. At present no commonly used software or hardware system is available to the virtual environment researchers. This means that most researchers use differing platforms for their experiments, in a strict scientific sense this means that the reproducibility of experiments must presently be called into question. On first inspection the problem is solvable by defining a common system of hardware and software, however there are two main problems with this.

- Synchronization of systems.
- Will the researchers really have the same system?

The first problem is a practical one. Who will oversee the platform and administer its use? This is a logistical problem and whilst theoretically solvable it is practically very difficult to administer. Whilst this is not impossible it is difficult to see an international effort to do this being acceptable to researchers, a body willing to accept the responsibility and cost of the role and the research finance needed being readily available.

The second problem may be even more difficult to overcome. In effect any base platform would require researchers to use identical systems. In a field that relies on commercially available software and hardware this is problematical, supposedly 'identical' systems in information technology may suffer from subtle but nevertheless real differences that may not be obvious from the outset.

Software versioning is already a problem in the IT industry. Two 'identical' virtual computer systems could have differing version of the operating systems on due to bugfixes, different BIOS due to dates of manufacture, differing ROM in the tracking
systems, different motherboards etc. However, it goes further than this when we consider virtual environments as, in part, bespoke systems. Different researchers may use different algorithms to achieve the same goal; there could be differences in calculation precision, the data structures used for geometric representation and other virtual entities. In the case of distributed systems processes may operate across processor boundaries and be subject to the bandwidth and temporal characteristics of the network and have the overhead of whatever remote process call procedure is present. Even the display that the user encounters will differ from system to system in terms of gamma values.

One commercially available VE (Division dVS 1996) system allows polygonal, bounding or bounding sphere based collision detection, textured or non-textured modes, geometry to be representation via polygons, tristrips or trifans, a choice of geometry representation may therefore affect performance depending on the graphics hardware in place.

Combining the possible impacts of the factors above make generalising task performance on a single performance parameter problematic. Take for instance a hypothetical link between frame rate and performance at a pick and place task (Ware and Balakrishnan 1994), if we try to compare the performance (reproduce the results) on a different system there may be a different underlying collision algorithm that decreases or increases the computers collision detection performance in the very range we are interested in. We could for instance improve the performance of the collision detection algorithm on an Intel Pentium III processor by decreasing the processor precision from IEEE double to IEEE single precision with the _controlfp() Win32 API instruction whilst we carry out the testing. This would virtually double the speed of square root and divide operations and the algorithm since collision detection algorithms will use a lot of square root and divide operations. One researcher may be aware of this and another may not.

There may be subtle complexity based parameters present as well, both Stanney and Groen (Stanney and Kennedy 1997; Groen and Werkhoven 1998) have reported kinaesthetic adaptation to systems that mean varying initial conditions such as the calibration or gain of trackers may have an effect. Indeed, Zhai (Zhai and Milgram 1994) and Nixon (Nixon et al. 1998) have shown that there are asymmetries in tracker magnetic field due to many reasons. The magnetic field of a tracker is determined by
the windings of the conductors around a ferrite core differences in the field windings will cause differences in the magnetic field it produces. One way to affect the field of a magnetic tracker is to place it near an already existing magnetic field; the tracker's field is then modified by the principle of superposition. Metal objects can also modify the magnetic fields of a tracker (Nixon 1998).

When one combines the flexibility of configuration, software, hardware, operating system with the number of major and dot releases of this software and the different methods of programming an implementation we can see that the combinations of differences between supposedly 'identical' systems is huge. It is probably impracticable therefore to try and resolve these differences. All these possible differences may confound reproducibility of results cross platform. If our intention in virtual environment research is to compare cross platform then we need a method of assessing the comparability of systems.

1.4 Background discussion

We see from the above discussion that virtual environments can be considered as direct manipulation interfaces that use the model world metaphor. We also see that users approach computers with a mental model of how they work. The argument on platform complexity shows that using technology to define a base is fraught with confounding variables. To compare experiment with experiment on the basis of technological similarity needs a large investment in 'under the bonnet' investigations.

If we consider the real-world as a 'perfect virtual environment' we see that there are no gulfs of evaluation and execution in the real-world (Other than the inherent interface of the object itself) and that the user has a highly learned mental model of the system. A fairly simple deduction then tells us that a comparison of real-world human behaviour and virtual world behaviour can inform us about our gulfs of evaluation and execution and how the mental model of the real-world is being affected. If we accept this argument we can see that any mismatch between the real-world predictive models, errors and performances and a virtual environment is indicative of the introduction of gulfs of execution and evaluation.
1.5 Conclusion

This thesis investigates the possibility of using a human behaviour based methodology to quantify the fidelity of a virtual task with respect to its real counterpart. Indeed the research hypothesis is that human behaviour can be used as a datum to compare real and virtual tasks. A side effect of this comparison is the ability to indirectly compare the quality of differing virtual environments by comparing their behaviour-based measures. Finally, this thesis seeks to remove any technological based bias of virtual-real comparisons by using human behaviour as a cross system datum point.
2 Chapter 2: Literature Review
2.1 Introduction

The field of virtual reality draws inspiration from fields as diverse as computer science, optics, and psychology. This disparate and wide-ranging nature means that any literature review will not be a straightforward database review of particular terms or keywords. This literature review considers papers written by psychologists, computer scientists, human factors practitioners and other fields with interests in virtual reality. This inclusion is required as all these researchers have differing methods but allied perspectives on the research problems that face the virtual environment practitioner and all perspectives, at present, are equally valid since the field is yet to mature.

This review is intended to be a comprehensive analysis of the literature that is related directly to or has bearing upon the research and research direction of this thesis. To bound the literature review within a manageable subset of the literature related to virtual reality, the following problem statement was formulated. Using this question as a guideline for relevance allowed the literature review to close in on the important issues.

"How closely does our virtual task resemble the real-world task?"

The literature review was the driving force behind the identification of the research opportunities that this thesis seeks to exploit. It covers the general literature on the subject of virtual environments, issues of presence, usability, simulation fidelity, human factors and any specific studies of relevance.
2.2 General Reading

The two most important pieces of literature in the field of virtual environments are the paper by Ivan Sutherland (Sutherland 1963) that introduced interactive computer graphics and the GUI (Graphical User Interface) to the world and the paper Sutherland (Sutherland 1965) that proposed a totally reconfigurable room-sized display. This paper was about the ‘Ultimate Display’ and through it Ivan Sutherland is credited with inventing the concept of a computer generated virtual environment or virtual reality.

In 1966 Sutherland built the first ever HMD (Head-Mounted Display Figure 2.2) known as the “Sword of Damocles”. The world’s first interactive computer graphics and GUI ‘Sketchpad’ can be seen in Figure 2.1 below.

Figure 2.1 Ivan Sutherland’s Sketchpad on an MIT TX-2 Computer
After the initial work by Sutherland many years passed before NASA researchers (Fisher et al. 1986) recognised that the availability of cheap LCD screens and powerful computer systems could be utilized to recreate the ideas that Sutherland had envisaged and create graphically sophisticated virtual environments. After this period virtual environments and virtual reality became a research field in its own right.

Some works of special note to the researcher interested in the broad range of issues that are associated with virtual environments are (Ellis et al. 1993; Kalawsky 1993; Barfield and Furness 1995; Carr and England 1995). Kalawsky gives a coherent history of virtual environment systems and then looks at physiology, perception, virtual environment systems and their enabling technology. Ellis deals with visual and spatial perception, telerobotics, manual and supervisory control, cartography, scientific visualisation, cartography, and medical illustration. Ellis’ work is a collection of papers that provide many insights into the issues related to HCI issues that impact upon researchers in the field of virtual reality and virtual environments. Carr collects together some valuable insights into the current thinking on the perceptual processes involved in experiencing virtual environments. Barfield, as with Carr and Ellis, provides a collection of papers from the some of the most prominent
researchers in the field on subjects as varied as the origins of virtual environments, cognition, performance, presence, interaction and eye tracking. These four publications are an ideal introduction to the issues related to the design, construction and analysis of virtual environment systems.

2.3 Related Research Areas and Work

The problem statement presented in Section 2.1 helped focus the majority of the effort of the literature survey directly upon performance measures within virtual environments and virtual reality. Reviewing the literature shows that whilst it is difficult to categorise all virtual environment research into performance in virtual environments generally the field splits along three lines simulation fidelity, presence and usability. The division of researchers using each measure tends to reflect the aims and background of the researcher using that particular measure.

To give the reader an idea of which measures are applied by the differing fields one could, informally, state that the interest in presence has a psychological slant, simulation fidelity has a computer science and mathematical simulation perspective and usability comes within the realm of human factors and software engineering. However, there is much cross-fertilization within virtual environment researchers.

2.3.1 Virtual Presence

Virtual presence is probably the single most researched area in the field of virtual environments. Qualitatively virtual presence refers to a participant’s subjective feeling of ‘being there’ (Heeter 1992) within a virtual environment. Virtual presence emerges from the research carried out into ‘telepresence’ and remote operation (See (Ellis et al. 1993)). (Held and Durlach 1992) determines telepresence as occurring when the following condition is satisfied (attributed to (Akin et al. 1983)).

“At the worksite the manipulators have the dexterity to allow the operator to perform normal human functions. At the control station, the operator receives sufficient quantity and quality of sensory feedback to provide a feeling of actual presence at the worksite.”
One of the reasons presence is the most researched area of virtual reality is because it deals, in a large part, with the users subjective feelings. This is the area in which the media and other external commentators have decided is the most important part of the technology. (Heeter 1992) describes three parameters or dimensions of presence.

- Subjective personal presence
- Social presence
- Environmental presence

(Loomis 1992) deals with the definition of presence by bringing in an alternative perspective from psychology. Loomis uses Self and Non-Self as an introduction to the concept of ‘distal attribution’. Distal attribution is the conceptualisation of objects as external to the self via distal stimuli. Loomis draws a comparison between the sensations of presence and perceptual act of distal attribution.

(Steuer 1992) discusses presence with respect to the vividness and interactivity of the content and medium used to present the environment. He describes vividness as the ‘ability to produce a sensorially rich mediated environment’ and interactivity as ‘the degree to which users can influence the form or content of the mediated environment.’ Steuer also groups all perception based mediation technologies under a group of media systems. Media systems he includes range from smoke signals to diorama to Paramount’s Starship Enterprise’s Holodeck, real and fictional systems.

Zeltzer (1992) provides a highly cited taxonomy of graphic simulation systems (Figure 2.3) using the AIP (Autonomy Interaction and Presence) cube.

- **Autonomy** - Extent to which the VE is more than just a geometric entity
- **Interaction** - Extent to which the VE can be modified at runtime
- **Presence** - Used as a measure for the number and fidelity of sensory channels

Whilst the AIP cube is useful to roughly categorise the types of mediated experiences that are available via today’s computer hardware it is not a system performance assessment or comparison device since it offers no quantitative measures or qualitative assessment methodologies in itself.
(Sheridan 1992a; Sheridan 1992b; Sheridan 1996) notes in his earliest paper on the subject that the contemporary literature on presence 'offers us no useful measure' of presence and instead offers three determinants of presence and thus implies that presence is a function of these three components.

- Control of sensors
- Extent of sensory information
- Ability to modify environments

Sheridan also mentions the degradations of afference and efference stimuli having some impact on the subjective feeling of presence.

Implicit in all the papers quoted so far was the assumption that the maximization of presence was the main goal of designing virtual interfaces. (Ellis 1996) questioned this aim and instead argued that efficient communication between the operator and the simulations should be the defining measurement. Ellis argues that if presence cannot be shown to correlate positively with this efficient communication, in other words the task can be completed satisfactorily, then presence is a secondary concern.
(Schloerb 1995) provides an interesting proposal for a quantitative measure of presence that uses both subjective and objective measures. The objective measure utilizes task completion expectation values or 'likelihoods that a task will be completed' to define itself. The subjective measure that is used by Schloerb is a probabilistic measure ('likelihood') of whether the user will feel present in an environment. The work uses signal detection theory in the form of Receiver Operator Characteristics (ROC) (Wickens 1992).

\[
\nu(x, t) = \sum_{t \in \tau} P_t \left[ \sum_i P_{ix} V_{it} \right]
\]

Where

\[
\nu(x, t) = \text{expected value of performance of operator } x \text{ for a given set of tasks } t
\]

\[
P_t = \text{probability of task } t
\]

\[
P_{ix} = \text{probability that outcome } i \text{ will occur given task } t \text{ is performed by operator } x \text{ (i.e. the level of objective presence)}
\]

\[
V_{it} = \text{value outcome}
\]

**Equation 2.1 Schloerb's objective presence measure equation**

Other interesting and relevant work in presence includes the effects of pictorial realism on presence (Welch et al. 1996) which concludes that pictorial realism is less important in generating presence than visual feedback delay and interactivity. A paper on the methods of determining a causal link between presence and task performance (Welch 1999) demonstrates the paucity of evidence of a link between presence and task performance in 1999.

More recently the multi-dimensionality of presence has been further recognised by (Witmer and Singer 1998). Their paper provides a presence questionnaire (PQ) and immersive tendencies questionnaire (ITQ) that is used to demonstrate a weak but positive relation between presence and task performance. (Hendrix and Barfield 1996) report positive relationships with presence for (Geometric field of view) GFOV, visual coupling and stereopsis. The paper also discusses the multidimensional nature of presence.

Slater and Wilbur revisit presence and factor it into two parts (Slater and Wilbur 1997), 'Presence and Immersion'. The definition of presence remains roughly the
same but the new measure of the immersion of the virtual environment is defined using technological factors such as FOV and resolution. Slater (Slater 1994) develops the concept of immersion further by using the idea of a stacking depth. Stacking virtual environments is achieved by a participant donning virtual HMDs (Head-Mounted Display). The user entering another virtual environment by putting on a virtual HMD whilst in a previous virtual environment would achieve a stacking depth of 2. The results from this paper (Slater 1994) show a positive relationship between stacking depth and presence.

(Hendrix and Barfield 1996) reports on an experiment that demonstrates the importance of spatialised sound in increasing the sense of presence of a virtual environment. In another related paper (Gilkey and Weisenberger 1995) the effect of sudden deafness in adults is related to a sense of detachment with the real-world, the implication being that ambient noise has a large psychophysical scaling impact on presence in virtual environments.

Despite no strong positive correlation being shown to date between presence and task performance. (Bystrom and Barfield 1999) presents the IPP (Immersion, Presence, Performance) conceptual model for use in ‘developing a theoretical framework for research into presence and for interpreting the results of empirical studies.

Interestingly a recent paper by (Freeman et al. 2000) answered a conjecture by Loomis (Loomis 1992) that behavioural responses such as the object ‘looming’ startling response may be good measures of presence. Freeman found that the postural and subjective measures were not significantly correlated.

Kalawsky (Kalawsky et al. 1999; Kalawsky 2000) suggests that presence may be represented by a measure depending on multi-dimensional factor. The parameters for this measure being split into 4 factor areas (Figure 2.4). (Rix 1999) could also be considered as approaching this by building multi-modal perceptual models.
### Behavioural Morphisms In Virtual Environments

<table>
<thead>
<tr>
<th>Factors</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Demand of attentional resources</td>
<td>$\alpha = \text{demand on attentional resource}$</td>
</tr>
<tr>
<td>Supply of attentional resources</td>
<td>$\beta = \text{supply of attentional resource}$ $\chi = \text{concentration of attention}$ $\beta_d = \text{division of attention}$ $C = \text{spare mental capacity}$</td>
</tr>
<tr>
<td>Understanding of situation</td>
<td>$SA, \Delta SA = \text{Situation awareness and change in situation awareness}$ $\gamma = \text{understanding of situation}$ $\mu = \text{complexity of situation}$ $\tau, \Delta \tau = \text{Spatial awareness and change in spatial awareness}$ $\Psi = \text{familiarity of situation}$</td>
</tr>
<tr>
<td>Information</td>
<td>$I_{qy} = \text{information quantity}$ $I_{qau} = \text{information quality}$ $T_e = \text{elapsed time in environment}$</td>
</tr>
<tr>
<td>Technological factors</td>
<td>$\kappa = \text{sensory modality}$ $\sigma = \text{degree of immersion}$ $\theta = \text{field of view subtended by the participant's eye}$ $\phi = \text{field of regard of participant}$ $d_m = \text{Display mode (binocular, monocular)}$ $t = \text{update rate}$ $\Delta t = \text{time lag (propagation delay between event and consequential action)}$</td>
</tr>
</tbody>
</table>

Figure 2.4 The parameters of Kalawsky’s multidimensional presence measure

(Flach and Holden 1998) have contributed a paper to the presence discussion with a psychological perspective; it reviews the science of perception with particular respect to virtual environments. It covers the main subjects of immersion, presence and perception but with respect to various modern perception theories including to the ecological theory of perception expounded by Gibson (Gibson 1979) that is cited by many researchers in the field of virtual reality.

For the reader who may wish to carry out further investigation into presence research both (Snow 1996) and (Schuemie et al. 2001) provide excellent review material and synopsis of presence research to date.
As can be seen from the above publications on virtual presence and the numerous attempts to define and redefine it, it is a poorly understood phenomenon that has been in a constant flux since its first proposal as a quality measure for virtual environments.

### 2.3.2 Usability

Usability is an area of HCI that is becoming more and more important to virtual reality and virtual environment designers and developers. It allows the designer to focus the limited resources of the technology onto the most pertinent area of the system design. Usability is *concerned with making systems easy to learn and easy to use.* (Preece et al. 1994)

A literature review in a recent thesis by Kaur (Kaur 1998) identified that the area of usability in virtual environments systems is not well established, that the causal factors have not been identified and the impact of varying those causal factors is not well understood (cf. Presence research). Kaur also notes that whilst there are various studies that could be used within a usability research context the application domains studied have been too isolated from each other to be of comparative value. For instance, it would be difficult to draw a general usability conclusion between a Near-Earth space simulator (Stytz and Kunz 1996) and virtual reality laparoscopic surgical simulator (Tendick 2000) since the application domains are so far apart. Usability studies such as these have concentrated on the underlying virtual environment hardware of application domain specific interactions.

One of the first to identify a need for overall rather than individual usability analysis techniques for virtual environments was (Kalawsky 1999). Kalawsky carried out research into both the rationale and application of the usability methods expounded by (Preece et al. 1994; Dix et al. 1998). He presents details of a computer based questionnaire evaluation that incorporate some of the multi-dimensional factors mentioned in (Kalawsky et al. 1999; Kalawsky 2000). Kalawsky’s Usability analysis (VRUSE) describes the use of a computer based usability questionnaire that is used to collect data on ten usability factors.
(Gabbard and Hix 1997) delivered a report on a "taxonomy of usability characteristics in virtual environments" to the office of Naval research that is an excellent in depth investigation into the area. This report identified as with Kaur (Kaur 1998) and Kalawsky (Kalawsky 1999) that little work had been carried out in the area. An overview of the areas that Gabbard covers is shown in below.

Figure 2.5 Gabbard's VE usability taxonomy (Taken from Gabbard)

Gabbard (Gabbard et al. 1999) also investigated the usability of eye tracking in a medical visualisation simulation. This novel experiment uses experts in VE to
construct a ‘Heuristic Evaluation’ which is intended to identify usability problems against their collected knowledge and a ‘Formative Usability Evaluation’ in which representative users perform tasks in an attempt to identify usability issues. In essence this is a rapid iterative prototyping methodology (Preece et al. 1994; Dix et al. 1998) formalized using usability methods.

(Lindeman et al. 1999) carried out a usability study of user interfaces for immersive virtual environments, the study was confined to using differing 2D interface metaphors (widgets) within an immersive virtual environment with or without haptic feedback. From the usability within virtual environments perspective the experiment was very tightly focused on the devices investigated and general conclusions about virtual environment usability is not covered and probably not possible due to the specific nature of the interface metaphors.

2.3.3 Simulation Fidelity

Many researchers in virtual environments have identified simulation fidelity as an area requiring more research (Zeltzer 1992; Kalawsky 1993; Ditmar and Hale 1994; Rinalducci 1996; Stytz 1996; Eggleston and Janson 1997). Stytz mentions a need for research into the human factors of simulation fidelity. Zeltser calls for research into 'selective simulation fidelity' whilst Kalawsky focuses on the 'human factors inevitably associated with a complex man-machine interface'.

"While VR systems have been around for some time (e.g., aircraft and automobile simulators), only recently has the capability existed to directly manipulate objects in a VE. Consequently, the data are generally scarce in addressing how properties of the VE effect [sic] human performance, ..." (Eggleston and Janson 1997)

However, although a reasonable consensus exists about the need for simulation fidelity measures there remains no general 'catch all' measure of simulation fidelity (Pace 1998; Roza et al. 1999). Indeed it seems to be accepted that fidelity requirements are highly task dependent. Simulation fidelity has three recognised parameters (Meister 1990; Rinalducci 1996).
Stytz (Stytz 1996) proposes that the accepted definition must be extended due to the blurring of these measures in a virtual environment. For instance, in a totally computer generated virtual environment where is the division between equipment and environment fidelity? Stytz proposes the following seven fidelities.

- Sensory Fidelity
- Physics Fidelity
- Modelling Fidelity
- Information Fidelity
- Time Fidelity
- System Fidelity
- Input Device Fidelity

Clearly, Stytz is starting to factor out the multiple underlying factors that will need to be examined in any simulation fidelity measure for virtual environment based problems. This is a promising development as is recognises the possibility that many sub-fidelities may need to be taken into consideration if one also considers the complexity of the high level psychological fidelities that may come into play.

Recent work by Simulation Interoperability Standards Organization (SISO) members revisits the question of simulation fidelity within distributed virtual environments (Clark and Duncan 1997; Schow et al. 1997; Fay 1998; Pace 1998; Roza et al. 1999; Roza, et al. 2000) Of particular interest is the definition of fidelity.

'\textit{The degree to which a model or simulation reproduces the state and behavior (sic) of a real-world object or perception of a real-world object, feature, condition, or standard in a measurable perceivable manner.}'
The major proposed components of fidelity in (Roza et al. 1999) cf. Stytz fidelities) are:

- Resolution
- Error
- Accuracy
- Sensitivity
- Precision
- Capacity

(Schow et al. 1997) goes further than other SISO members and follows Stytz in factoring out the individual fidelities of a simulation. He even presents a calculus for calculating fidelity differentials. (Schow et al. 1997) starts with set of unordered A priori properties that represent the simulation.

\[ P = \{P_1, P_2, P_3, ..., P_n\} \quad \text{Equation 2.2 Simulation unordered properties} \]

These properties may be elements of the simulation or the entities within the simulations. For instance, if our task performance were causally linked to presented texture frequency then texture frequency must be a property in the set above since it is a variable of interest. The set is then sub classed into an ordered set of properties, the ordering being related to the most important property attributes of the simulation i.e. q1 most important, q2 next most important etc.

\[ Q = \{q_1, q_2, q_3, ..., q_n\} \quad \text{Equation 2.3 Simulation Ordered Properties} \]

These ordered properties are then aligned with their purposes into a set of ‘experimental frames’. So each property has a corresponding purpose. Both property and purpose are now ordered by importance.

\[ q_1 \rightarrow e_1, q_2 \rightarrow e_2, ..., q_n \rightarrow e_n \quad \text{Equation 2.4 Property and purposes association} \]

The measure of the fidelity of a simulation is then some dependent measure of the properties against the purposes. An example of a fidelity measure could be a correlation of simulation flight path with intended flight path. After initial exploration
of the components of fidelity the paper defines candidates for differential and total fidelity measures.

\[ \sum |F_i - F_j| \] Where \( F = \) Component Fidelity

Equation 2.5 A fidelity differential

\[ \sum F_i W_i \] Equation 2.6 Total system fidelity

These fidelities are essentially a set of measures related to the simulation properties and weighted according to the level of importance of that simulation property. These at present are mainly visual fidelity criteria but behavioural aspects have been proposed.

As we can see from the above some of the research is starting to touch on areas of interest to virtual environments. The reduction of simulations into low-level fidelity entities resonates with the need to assess the task fidelity aspects of virtual environments.

In the future the SISO work in fidelity could be of great importance to the virtual reality community since it allows a 'divide and conquer' methodology to be applied to the very complex problem of assessing how to model virtual environments. Breaking tasks into sub-tasks as a method of defeating complexity has precedents in computer science, both the GOMS model (Goals, Operators, Methods, and Selection)(Card et al. 1983) and top down stepwise refinement use this methodology (Sommerville 1992) to reduce the complexity of the task at hand.

2.3.4 Human Factors

There have been no comprehensive empirical studies that have dealt with the human factors of virtual environments and virtual reality. However many review papers do exist which seek to provide human factors specifications for the design of virtual environments. (Padmos 1992) surveys the literature in the area of the human factors of simulator imaging. Starting with a review of the areas (Padmos 1992) reports on the necessary visual cues for a number of different types of simulators (See Figure 2.6).
Another literature survey by (Barfield et al. 1995) investigates the human abilities to discriminate visual, tactile, auditory and kinaesthetic information and compares these with the abilities of modern technology to deliver these stimuli in the different modalities. This survey identifies the large gap between the human abilities in many areas and the reality of what modern technology (1995) can deliver. Most of the information is presented in the form of specification tables a’la (Boff and Lincoln 1988). However, most of this information is related to the information processing side of human perception and omits completely the cognitive aspect of human perception.

<table>
<thead>
<tr>
<th>Areas Investigated</th>
</tr>
</thead>
<tbody>
<tr>
<td>Binocular Cues</td>
</tr>
<tr>
<td>Monocular Cues</td>
</tr>
<tr>
<td>Image Field Size</td>
</tr>
<tr>
<td>Image Collimation</td>
</tr>
<tr>
<td>Image Projection</td>
</tr>
<tr>
<td>Image Viewing Region and Distance</td>
</tr>
<tr>
<td>Image Luminance</td>
</tr>
<tr>
<td>Image Spatial Resolution</td>
</tr>
<tr>
<td>Image Colour Palette</td>
</tr>
<tr>
<td>Update Frequency</td>
</tr>
<tr>
<td>Refresh Rate</td>
</tr>
<tr>
<td>Image Delay</td>
</tr>
<tr>
<td>Image Anti-aliasing</td>
</tr>
<tr>
<td>Number Of Polygons Per Channel</td>
</tr>
<tr>
<td>Depth Of Field</td>
</tr>
<tr>
<td>Level Of Detail</td>
</tr>
<tr>
<td>Texture</td>
</tr>
<tr>
<td>Transparency</td>
</tr>
<tr>
<td>Shadows</td>
</tr>
<tr>
<td>Meteorological Effects</td>
</tr>
<tr>
<td>Time Of Day</td>
</tr>
<tr>
<td>Light Points</td>
</tr>
</tbody>
</table>

Figure 2.6 Areas surveyed by Padmos and Milders
or its possible ecological nature, they tacitly acknowledge this in the following remark.

'The glaring omission in our tables, and arguably in the entire field of teleoperation and virtual environments, is the lack of a conceptual framework for understanding how and why less than perfect interfaces slow down task completion or reduce the sense of presence.' (Barfield et al. 1995)

Rinalducci's (1996) survey paper on visual fidelity adds further to the collection of facts and figures of human factors specifications. Unlike the previous papers, however, Rinalducci explains the psychological background behind aspects of visual cue choices in virtual environments. Areas covered by (Rinalducci 1996) are shown below.

<table>
<thead>
<tr>
<th>Areas Investigated</th>
</tr>
</thead>
<tbody>
<tr>
<td>Motion</td>
</tr>
<tr>
<td>Colour</td>
</tr>
<tr>
<td>Binocular Vision and Stereopsis</td>
</tr>
<tr>
<td>Pictorial, Secondary and physiological cues to depth</td>
</tr>
<tr>
<td>Texture</td>
</tr>
<tr>
<td>Vertical Development (Height In Field)</td>
</tr>
<tr>
<td>Luminance</td>
</tr>
<tr>
<td>Field Size</td>
</tr>
<tr>
<td>Spatial Resolution</td>
</tr>
</tbody>
</table>

Figure 2.7 Areas surveyed by Rinalducci

Rinalducci (Rinalducci 1996) concludes his paper by remarking that future work should, amongst other recommendations, research into.

'determinations of the simulation fidelity (i.e., how well they model the situation) of virtual environment visual displays for various related tasks. In other words, how does performance in a simulated environment compare to that in the real-world situation.'
The human factors of three-dimensional display systems have been investigated by (McKenna and Zeltser 1993). The paper is a study of stereoscopic, lenticular, parallax barrier, slice-stacking and holographic display systems that investigates both the technology used to display images, the human eye response, stereopsis, FOV, viewing pupils and bandwidth.

The human factors of the use of virtual environments in visualising scientific data were studied by (Baker and Wickens 1995). Baker reaches no conclusions. This is a survey paper that collects the relevant human factors issues and psychological issues into a single repository but provides no forward momentum or pathway for the field. A more recent paper by (Stanney et al. 1998) again surveys the human factors of virtual environment. The areas covered include human performance efficiency, simulator sickness, human sensory considerations, health and safety and the direct effects/indirect after effects of virtual environment immersion. The paper quotes Wann and Mon-Williams (Wann and Mon Williams 1996)

"the goal is to build environments that minimize the learning required to operate within them, but maximize the information yield."

This quote can be compared directly with the definition of usability used by Preece (Preece et al. 1994). This assumption is clearly incorrect if, as this thesis aims to, we wish to reproduce the real-world task as well as possible. However this paper does provide a useful categorisation of the research threads that virtual environment research should be broken down into. This may seem a trivial point however the focus is needed since the field is so diverse and cross-discipline. These categories are sufficiently diverse and yet informative as to allow the area of most researchers to be pigeonholed usefully.

- Human Performance Efficiency
- Heath and Safety
- Social Implications
2.3.5 Adaptation and After Effects

Human beings have an innate plasticity to their nervous systems; this plasticity is the basis for leaning and adaptation. However, this wonderful gift has a side effect when it occurs within virtual environments. People adapt to virtual environments and the after effects can be problematical. Many researchers have investigated these adaptations.

Changes in oculomotor function have been reported by Howarth (Howarth 1999) and Hasebe (Hasebe et al. 1996). Rolland and Biocca (Rolland and Biocca 1995; Biocca and Rolland 1998) report adaptation in hand to eye co-ordination that is a temporary after effect of using a HMD in an augmented reality experiment. (Groen and Werkhoven 1998) present results that demonstrate an adaptation to a discrepant hand position in a virtual environment, this is an important paper since it shows that motor skill training could be subject to adaptation effects. A method to quantify adaptation to proprioceptive and kinaesthetic discrepancies and results that show the effects of adaptation using these measures are described by (Stanney and Kennedy 1997).

2.3.6 Other Related Work

2.3.6.1.1 Specific Studies

This section predominately deals with literature that has reference to this thesis because the experiment under study involved an assessment of task performance or human interaction in a virtual environment.

A study by (McGee et al. 1997) into evaluating input devices in a virtual environment with Fitts Law showed that target width, distance of target and angle of target had significant performance differences between the input devices. The title of this paper is slightly misleading since no virtual environment or virtual reality is involved in the experiment. The experiment used HMD to display a Fitts task that would normally be based on a computer screen and carried out with an input device such as a mouse. The task involved tapping between two rectangles displayed on the flat computer screen surface. It is difficult to see how these results could be generalised to a virtual
Behavioural Morphisms In Virtual Environments

environment. This experiments proposes itself as validation of Fitts Law in a virtual environment.

Using a peg insertion task (Nemire 1997) investigated the effects of visual and auditory cues. The independent variables in this experiment were stereopsis, binocular viewing, movement amplitude, target width, audio cues and visual cues. The results for this experiment demonstrated that there were no benefit for stereopsis cues or audio and visual cues for the easier tasks indexes. (Barfield et al. 1999) investigated the effects of stereopsis-and head tracking on a tracking task. The task in question was to keep a virtual stylus centred on a computer-generated wire. Barfield reports that the number of times the wire was overstepped, a measure related to time-on-target, was significantly reduced by the head-tracking cues but not the stereopsis.

An investigation into Fitts' Law in virtual environments by (Eggleston and Janson 1997) looked at the effects of FOV on the performance of a Fitts' tapping task. This experiment compared the real and virtual tasks and how performance was affected by differing sizes of FOV. The results showed that smaller FOV affected performance in the virtual environment. However the study only investigated two indexes of difficulty and accepts a' priori the validity of Fitts' Law in virtual environments.

A more recent study performed by (Mason et al. 2001) show that Fitts' Law does not always hold in a virtual environment when haptic feedback is removed. The study investigated a 'reach-to-grasp' task where visual and haptic feedbacks were present or not. The task carried out took place in a virtual environment and an augmented environment. The study demonstrates very strongly that visual and haptic stimuli can affect human performance in a virtual environment.

(Arthur et al. 1993) used a tree-tracing task to evaluate the effects of stereo-enabled visually coupled HMD's on task performance. The results showed that lag and frame-rate had a significant negative effect on the task performance. (Ware and Balakrishnan 1994) used Fitts' Law to study the effects of tracker lag and display frame rate on an object placement task. The study found that whilst head-tracking lag was not critical, hand tracking was. Low frame rates were also found to be performance degrading but this was concluded to be a result of the hand lag that was due to low frame rates. The
possible lags present in virtual reality systems are investigated by (Wloka 1995). Wloka explains that input devices, application program processing, image rendering, synchronization and frame rate can introduce lags. He also discusses lag prediction, reduction and measurement systems.

An often-cited paper into the trade off between resolution and interactivity in spatial task performance is (Smets and Overbeeke 1995). The experiment investigated a search and act 'jigsaw puzzle' problem using varying levels of degraded vision for the user. The users vision of the puzzle was through HMD, the HMD then had varying levels of degraded image passed into it. The other independent variable was 'interactivity' which was defined implicitly as 'some' difference between still camera, passive camera, and head-coupled camera. The results showed that spatial resolution could be traded-off interactivity, i.e. the head-coupled situation was the best performing.

The differing human performance between 3D mice and virtual hand control was studied via manipulation performance in an important paper by (Werkhoven 1994). This study used a die manipulation task in which the participant had to grasp, pitch, roll and position virtual die. Werkhoven (Werkhoven 1994) concludes that retinal disparity is a large cue in assessing an object's position relative to a virtual hand or a 3-D cursor and that its presence is a performance enhancer. Stereopsis did not aid the pitch and roll tasks in this study. Positioning in the depth coordinate was the worst performing positioning task with participants performing best using the virtual-hand. The absence of stereo cues affected the cursor-controlled task the most. Size or subtended visual angle is also concluded by Werkhoven be a large factor in the assessment of object distances.

In an attempt to create a standard platform for virtual environment training research (Lampton et al. 1994) constructed the VEPAB (Virtual Environment Performance Assessment Battery). The battery consists of a suite of tasks that can be used to benchmark human performance, investigate side and after effects, and subjective reactions across virtual environment platform. The tasks used by VEPAB are shown in Figure 2.8 below.
## VEPAB Tasks

<table>
<thead>
<tr>
<th>Vision</th>
<th>Color</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acuity</td>
<td></td>
</tr>
<tr>
<td>Object Recognition</td>
<td></td>
</tr>
<tr>
<td>Size Estimation</td>
<td></td>
</tr>
<tr>
<td>Distance Estimation</td>
<td></td>
</tr>
<tr>
<td>Locomotion</td>
<td>Straight-away</td>
</tr>
<tr>
<td>Backup</td>
<td></td>
</tr>
<tr>
<td>Turns</td>
<td></td>
</tr>
<tr>
<td>Figure-8</td>
<td></td>
</tr>
<tr>
<td>Doorways</td>
<td></td>
</tr>
<tr>
<td>Windows</td>
<td></td>
</tr>
<tr>
<td>Elevator</td>
<td></td>
</tr>
<tr>
<td>Manipulation</td>
<td>Slide</td>
</tr>
<tr>
<td>Dial</td>
<td></td>
</tr>
<tr>
<td>Bins</td>
<td></td>
</tr>
<tr>
<td>Tracking</td>
<td>Head Control</td>
</tr>
<tr>
<td>Stationary Target</td>
<td></td>
</tr>
<tr>
<td>Moving Target</td>
<td></td>
</tr>
<tr>
<td>Device Control</td>
<td></td>
</tr>
<tr>
<td>Stationary Target</td>
<td></td>
</tr>
<tr>
<td>Moving Target</td>
<td></td>
</tr>
<tr>
<td>Reaction Time</td>
<td>Simple Reaction</td>
</tr>
<tr>
<td>Choice Reaction</td>
<td></td>
</tr>
</tbody>
</table>

**Figure 2.8 VEPAB suite of benchmarking tasks**

The experiments carried out in the VEPAB paper were used as an evaluation of the battery of tests. The visual test section of the battery uses direct comparisons with *real-world measure*, i.e., Snellen charts for acuity and Ishihara plates for colour vision. The visual acuity figures are even reported as a Snellen ratio's (20/860). The VEPAB methodology identifies that to compare implementations and experiments a common framework must be in place. However the method is confounded by the
framework being based on the identical implementation of that technology across researchers. As argued in the background of this thesis this is very difficult to achieve.

(Poupyrev et al. 1996; Poupyrev et al. 1997) describe a framework for studying interactions with immersive virtual reality using a task analysis breakdown of the operations carried out during virtual manipulations.

<table>
<thead>
<tr>
<th>Task</th>
<th>Independent Variable</th>
<th>Metric</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select</td>
<td>Distance to target</td>
<td>Virtual cubits</td>
</tr>
<tr>
<td></td>
<td>Horizontal and vertical directions to target</td>
<td>Degrees of arc</td>
</tr>
<tr>
<td></td>
<td>Horizontal and vertical size of non-occluded portion</td>
<td>Degrees of arc or percentage</td>
</tr>
<tr>
<td></td>
<td>Distance to occluding object</td>
<td>Virtual cubits</td>
</tr>
<tr>
<td></td>
<td>Direction of occlusion</td>
<td>Left/right/up/down</td>
</tr>
<tr>
<td></td>
<td>Horizontal and vertical size of target</td>
<td>Degrees of arc</td>
</tr>
<tr>
<td>Position</td>
<td>Initial distance</td>
<td>Virtual cubits</td>
</tr>
<tr>
<td></td>
<td>Initial vertical and horizontal direction</td>
<td>Degrees of arc</td>
</tr>
<tr>
<td></td>
<td>Final distance</td>
<td>Virtual cubits</td>
</tr>
<tr>
<td></td>
<td>Final horizon and vertical directions</td>
<td>Degrees of arc</td>
</tr>
<tr>
<td></td>
<td>Vertical precision</td>
<td>Percent of overlap</td>
</tr>
<tr>
<td></td>
<td>Horizontal precision</td>
<td>Percent of overlap</td>
</tr>
<tr>
<td>Orient</td>
<td>Distance</td>
<td>Virtual cubits</td>
</tr>
<tr>
<td></td>
<td>Horizontal and vertical directions</td>
<td>Degrees of arc</td>
</tr>
<tr>
<td></td>
<td>Initial orientation</td>
<td>Degrees of arc</td>
</tr>
<tr>
<td></td>
<td>(3 angles)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Final orientation</td>
<td>Degrees of arc</td>
</tr>
<tr>
<td></td>
<td>(3 angles)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Accuracy</td>
<td>Degrees of arc</td>
</tr>
</tbody>
</table>

**Figure 2.9 VRMAT Measures**
As can be seen this is a manipulation evaluation only and it is not intended to be a generic evaluation methodology for virtual environment task comparison or benchmarking.

Another test bed for virtual interaction is reported by (Tendick 2000). This project looks at the use of virtual environments for use with laparoscopic surgery. Tendick states.

'It is still poorly understood how surgeons learn and adapt to the unusual perceptual motor relationships in minimally invasive surgery.'

Because of this poor understanding methods of 'instrumented motions', i.e. tracking movements are proposed. The data from these motions can then be analysed for movement components. Presumably this comparison will be between real and virtual environment training scenarios.

In two papers back to back in the same issue of Presence (Bowman 2001; Bowman, Johnson et al. 2001) introduce a test bed for the evaluation of interaction techniques in virtual environments and a set of design guidelines for the construction of virtual environments. In the framework paper Bowman states

'Quantifying the performance of VE interaction is a difficult task, because performance is not well defined. It is relatively easy to measure and quantify time for task completion and accuracy but these are no the only requirements of real VE applications. VE developers are also concerned with notions such as the naturalism of the interaction (how closely it mimics the real-world) ...

...Thus it is essential that we focus on user centric performance measures.'

The framework however is designed for non real-world interactions and most of the interaction techniques used are novel and have no or little performance measures to compare against. It is difficult to see how a framework like this can 'compare with the real-world' when it uses non real-world interactions.
The design guideline paper (Bowman 2001) also deals with non real-world interaction devices that make it difficult to extend the guidelines to real-world mimetic virtual environments.

In a paper on using virtual reality as a tool for training assembly (Boud et al. 2000) actually instrumented the motions of experimental participants via instrumented objects (cf. instrumented motions) whilst carrying out a ‘Towers of Hanoi’ task (See Figure 2.10 and Figure 2.11 below). This experiment compared directly real and virtual tasks.

Figure 2.10 Instrumented objects

Figure 2.11 Towers of Hanoi movement paths
It can be seen from the path motion that the real-world condition (Circles) of the experiment is much smoother. The aim of this experiment was to assess the effect of haptic feedback on a task.

![Figure 2.12 Kuhlen's virtual/real cube grasping experiment](image)

Another real /virtual comparison was undertaken by (Kuhlen et al. 2000). The task was to grasp a real or virtual cube suspended in front of the participant. The set up of the experiment can be seen in Figure 2.12 above and some typical results in Figure 2.13 below. The conclusions of the experiment are deep-rooted in the movement organization of the brain that this thesis is not concerned with, but the experiment does allow the writers to contradict some existing hypotheses of human movement. This paper is important since the data for real and virtual tasks are collected in real-time, treated and compared. The writers realise that the behaviours are different and use that difference to support their arguments.
2.4 Discussion

The literature survey in this chapter is necessarily broad; this is because virtual environment research is an area comprised of many differing fields investigating many differing areas. All these fields have their own particular modus operandi in research terms due to the history and main aims of the area. This section will attempt to bring out the most salient points in the various areas. This is done as a justification for the subsequent research within this thesis. The first thing to note from the literature review is that none of the methods mentioned or papers presented provide an answer to the research question stated in Section 2.1, yet most of them at some point mention explicitly or imply that real/virtual performance and behaviour comparison is an important research area.

Presence, with the exception of (Schloerb 1995) is a solely subjective measure with virtually every measure being questionnaire based. It is difficult to see how one may arrive at any objective assumptions about human behaviour using it. From the contrary perspective it is also difficult to see how the introduction of an objective behavioural measure would benefit Presence. Adding a behaviour measure would merely introduce more variables into a multivariate function that is still too vague in its definition to be used in any rigorous fashion. Whilst presence is undoubtedly an important research area its utility is in the psychological importance scaling of the cues available for manipulation by system designers. Presence is at the moment a
phenomenon waiting for a theory; there is no agreement about what it is or how to measure it. All of the papers mentioned in the presence section of this thesis had a different conceptual model of Presence and a different way of measuring it.

(Schloerb 1995) mentions the concept of 'objective presence'. Objective presence is a candidate measure for answering our research question. However, objective presence is an analysis of task performance that uses probable outcomes rather than the behaviour of the subject to assess quality. Some tasks cannot be described simply by a categorical measures but require continuous values.

Usability has both subjective and objective measures at its heart but at present has no method or methodology to assess behavioural fidelity. As such usability techniques will be looking for methods of behavioural assessment. No behavioural assessments have been presented within the area of usability.

The simulation fidelity measures of 3D computer generated environments mentioned above focuses on a very small subset of visual criteria with no real effort to look at human behaviour. However if we view the overall measure of fidelity of a virtual environment as a sum of the weighted sub-fidelities of interest to the simulation as in Equation 2.6 we see that simulation has the capacity to absorb behavioural aspects into its methodology by adding valid a behavioural fidelity measures. No extension or re-evaluation of the proposed methodology would be needed. In the end this method may provide and answer to the research question being posed. The only research that requires to be carried out is the method of behavioural fidelity measurement.

The majority of human factors studies ((Padmos 1992; Barfield et al. 1995; Rinalducci 1996; Stanney et al. 1998)) carried out have mainly been fact-finding missions into human sensory capabilities. No comprehensive systematic empirical study has been carried out. This is probably because of the magnitude of the task that would be involved.

Most of the papers within the specific studies section above mention that there is insufficient understanding of the essential characteristics and parameters of virtual interfaces yet most of these papers used task performance measures based upon real-
world tasks to assess their experiments. Task performance is not the whole story of human behaviour. Humans adapt and adopt learned behaviours. Any experiments adopting ‘a priori’ the validity of any real-world models could be criticized for ignoring this. Most of the studies presented accepted ‘a priori’ that real-world psychological models apply to the virtual environments that they are using. The user could be behaving completely differently from the way they do in the real-world but using a learned behaviour or some form of strategy behaviour to obtain real-world performance. Three notable exceptions to this are (Boud et al. 2000), (Kuhlen et al. 2000), and (Mason et al. 2001).

2.5 Conclusion

There is a clear need for a method of comparison of real and virtual world tasks for virtual environments that are required to mimic the real-world. Most researchers agree on this. Inferring task fidelity via presence or task performance is difficult at best and may be fundamentally impossible or confounded by adaptation and learned behaviours.

What is clear throughout the research into virtual environments is that the only common factor across virtual environment platforms is the presence of a human participant. Therefore it makes sense to use the Human as the datum of measurement. (Lampton et al. 1994) VEPAB came very close to reaching this assumption by providing a battery of comparison tasks that used in part real word datums such as Snellen tests or the Ishihara colour plate. However the fact that they were also looking at non real-world interactions obscured the path and they included relative non real-world interaction task measures as well.

No method of real/virtual task comparison is apparent at present and none of the present measures or methods of measuring the quality of virtual environments are usable in this context. Without such a measure how can one demonstrate that a trainee is learning to use a laparoscope or a trainee is learning to use a simulation of a laparoscope? Are we training the trainees to use the tools or training them to use the simulation’s tools? We can show a procedural training transfer fairly simply but how do we extend that to motor skill transfer?
Any methodology that supposes to compare real and virtual tasks must be able to show differences, not only in task performance, but human behaviour itself. Otherwise the method could be confounded by learned behaviour, adaptation and strategy. This means that measure must be able to provide feedback about behaviour itself, task performance and errors.

The next chapter will propose a candidate for the methodology that is required called ‘Behavioural Morphism’. Behavioural morphism relies on only one thing; the availability of mathematical models of behaviour or the ability to collect data with respect to the task. The methodology could be incorporated into a usability analysis or simulation fidelity analysis as part of a weighted set of fidelities for the task.
3 Chapter 3: Behavioural Morphism
3.1 Introduction

In the background and literature review chapters of this thesis it was argued that there are many factors that can prevent researchers from generalizing results across virtual environments platforms or at least be considered as confounding factors within comparative experiments. The lack of a reproducible experimental platform was argued to be a result of the differing underlying technological and design factors conspiring through complexity. Some of these factors are shown below.

i. HCI Metaphor Factors
   i. Gulf of evaluation
   ii. Gulf of execution
   iii. Cognitive distance
   iv. Mental Model

ii. Virtual Environment Design Factors
   i. Cue modality
   ii. Interaction style
   iii. Adaptation
   iv. Simulator Sickness
   v. Cue absence

iii. Technological Factors
   i. Differing algorithm implementations

iv. Complexity Factors
   i. Operating System variances
   ii. Software release
   iii. Platform related bugs
   iv. Platform related performance issues

Reproducibility is taken by the scientific community to be a ‘necessary and sufficient’ condition for the general acceptance of scientific hypotheses. The lack of a reproducible platform is argued by this thesis to be a fundamental problem within the field of virtual environment research. Without an identical platform researchers would be only be able to compare individual cases on a platform-by-platform basis and then infer the general result. Clearly this is a far from ideal scientific methodology, such comparisons are weak and do not provide a basis for robust analysis. Someone
wishing to dispute the validity of this argument may suggest a contrary perspective by proposing that these factors have negligible impact between experiments. However to support this argument they must show differences to be negligible, at present no one has.

In order to reproduce experiments some method must be found to provide a reliable platform and provide robust experimental techniques that will quantify the effects of possible unknowns.

### 3.2 Behavioural Morphism

#### 3.2.1 Definition of Terms

**Definition of Behaviour**

Defined broadly, behaviour is a term used to describe how organisms act in response to environmental stimuli. For the purpose of this thesis the behaviour under investigation is human task solving behaviour and the stimuli are presented via a virtual environment or the real-world.

More specifically this thesis defines behaviour as being described by a mathematical model that predicts it. In other words the mathematical model describes the response of a human to some stimuli. As such behaviour and mathematical behaviour are synonymous.

**Definition of Performance**

The term performance is used in this thesis to describe some measure of a mathematical behaviour that is used to define the accomplishment of a participant at the task under investigation.

**Definition of Morphism**

Morphism is the condition or quality of having a specified form. For instance, an isomorphism is a one-to-one and onto mapping of a pattern (structure) between domain and range such that there is no loss of information. A homomorphism is
many-to-one mapping of a pattern between domain and range where the pattern may be simplified. If one makes a simplified model of some system a homomorphism is being applied.

### 3.2.2 Rationale

One can suggest that a 'perfect' virtual environment or virtual reality would be perceptually indistinguishable from the real-world (Zeltzer 1992). Following this line of thought we can see that for the purposes of a mimetic virtual environment evaluation we can consider the real-world as a form of virtual environment with no confounding factors. A mimetic virtual environment is a virtual environment that seeks to mimic the real-world. Once we make this fairly obvious assumption we can then see that real-world behaviour could and should be used as a metric to quantify the quality of any virtual environment. This may seem to be a trivial statement however a large amount of 'presence' research has been carried out without a firm understanding of the phenomena and without a firm psychological footing. (Heeter 1992; Held and Durlach 1992; Sheridan 1992; Zeltzer 1992; Slater 1994; Rolland 1995; Schloerb 1995) are some of the more lucid accounts of the presence phenomenon, the research into presence continues apace with numerous papers published. This research thread seems to have had the monopoly of interest and has yet to provide a workable framework for virtual environment comparison.

A solution to these issues has been developed by the author called 'Behavioural Morphism'. As its name implies it uses the 'form' of user real-world behaviour to quantify the quality or fidelity of a virtual task. Behavioural morphism is a concept that compares existing mathematical models of real-world behaviour with virtual environment behaviour. The behavioural morphism methodology allows the researcher comparison methods that abstract away from technological by using human behaviour itself as the datum of comparison. Behavioural morphism is a measure that is by definition free of technological dependence because it makes the platform of comparison human behaviour and not technology.

### 3.2.3 Mathematical Models

Mathematical models are used to predict many things, for instance, Newton's three laws of motion are examples of mathematical models that are so established they were
given special place as the fundamental ‘Laws’ of Physics for hundreds of years. The scientific power of a model comes from the predictive ability of the mathematical equation that is used to describe it. However not all mathematical models in Science can be explained in theoretical terms, Quantum Physics is far from being understood yet it has been used to calculate the most precise results (9 decimal places) ever known to Science (Feynman 1988). The fact that Quantum Physics can predict results to such accuracy demonstrates that one can use a model to describe the behaviour of a system without fully understanding the underlying principles of the system. As long as one can mathematically model something one can predict its behaviour, if the prediction is incorrect the model is refined. A mathematical model is a high level description of the system that can be validated or supported by the provision of empirical evidence. In psychology many cases exist where we describe high-level behaviour but do not understand the workings of a system, consciousness is a prime example (Penrose 1990).

Science creates mathematical models of a system by undertaking experiments where an independent variable is varied and any change in a dependent variable is observed. The validity of the derived mathematical model is established by using the derived model to predict a result and then showing that the predicted result occurred. After this an explanation of the mathematical model in theoretical terms can be offered. Normally statistical tests are applied to prevent type I and type II errors in rejecting or accepting the null hypothesis due the presence of random variables. A type I error occurs when a researcher claims support for a research hypothesis when the results are due to randomness or luck and the null hypothesis should have been accepted. A type II error occurs when a researcher dismisses a research hypothesis and accepts the null hypothesis when in fact the research hypothesis is correct, this again is due to the presence of random variables.

<table>
<thead>
<tr>
<th>Null Hypothesis is:</th>
<th>Accepted</th>
<th>Rejected</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>Correct</td>
<td>Type I error</td>
</tr>
<tr>
<td>False</td>
<td>Type II error</td>
<td>Correct</td>
</tr>
</tbody>
</table>

Figure 3.1 Type I and type II error table
For instance Newton hypothesised that the acceleration of a body is equal to the product of the force applied and reciprocal of its mass (Equation 3.1).

\[ \text{acceleration} = \frac{\text{Force}}{\text{Mass}} \]

Equation 3.1 Newton's first law of motion

If an experiment were carried out to measure the acceleration of an object using a linearly increasing series of forces we would see a very strong correlation between the force and the acceleration, providing we were accurate in our measurements. To prove that such a relationship really exists Newton's null hypothesis \((H_0)\) would state that there is no relationship between the dependent acceleration variable and the independent force variable; the alternative or research hypothesis would state \((H_1)\) that there is a relationship. Newton would reject \(H_0\) and support for \(H_1\) by demonstrating that any correlation is statistically significant to some level of significance. A significance result less that 1\% is highly significant and a level less than 5\% is significant.

3.2.4 Sources of Human Behavioural Models

In the fields of engineering psychology, ergonomics, human factors and human-computer interaction there are many works that provide the data and mathematical models by which the modern ergonomist, engineering psychologist or HCI practitioner can model human behaviour in their designs. Excellent examples are (Card et al. 1983; McCormick 1987; Boff and Lincoln 1988; Wilson 1990; Helander 1997). Wickens (Wickens 1992) describes the possible uses of such models and equations;

"Here a set of equations to represent some aspect of human performance can be calculated or ‘run’ (often on a computer) as the model simulation receives input that would characterise the system under development. The computer-simulated
'performance' of the model-simulated human operator and system can be examined, and if it is unsatisfactory, changes in the design concept can be implemented.”

By tuning this rationale to our problem domain of fidelity in virtual environments we see that many models of human behaviour could and should be used in validating the 'fidelity' of a virtual environment in terms of human behaviour. To validate a virtual environment task we should be able to compare the behaviour of the user in the real-world with the behaviour in the virtual world. The fidelity can then be quantified by assessing the distance between the behaviours using some objective and quantifiable measure.

3.2.5 Comparing Real and Virtual behaviour

The most commonly compared objective data in virtual environment experiments are task performance and error rate. Many studies and methodologies have used these as the basis of comparison (Arthur et al. 1993; Lampton et al. 1994; Ware and Balakrishnan 1994; Smets and Overbeeke 1995; Poupyrev et al. 1996; Eggleston and Janson 1997; Poupyrev et al. 1997). However, none of these studies have taken the step of considering behaviour in comparison with the real-world. All have accepted 'a priori' the validity of the real-world models within the virtual environment they are using. One can see that this is a large assumption considering the perceptual conflicts that can occur due to the lack of various cues, one only has to look at the considerable research into simulation sickness and vection induced motion sickness (VIMS) to see this.

Since we have concluded that behaviour should be a factor in evaluation we have to determine a manner in which to compare behaviours, to do this we use the Pearson Product Moment Correlation (Godfrey et al. 1988). Mathematical models are expressed as equations and this allows one to test the strength of relationship between a predictor (dependent) and criterion (independent) variable with correlation. Using the following formulae correlation between predictor and criterion variables can be established.
Behavioural Morphisms In Virtual Environments

\[ r = \frac{\operatorname{cov}(x, y)}{s_x s_y} \]

Where

\[ \operatorname{cov}(x, y) = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{n} \]

\[ s_x = \text{standard deviation in } x \]

\[ s_y = \text{standard deviation in } y \]

**Equation 3.2 The Pearson product moment correlation**

In statistical terms this correlation gives us a measure of the variance in y that occurs when introducing a variance in x. **Transforming** any model’s equation into a linear form shown in **Equation 3.3**, with a predictor variable mapping onto x, enables us to perform a correlation calculation using **Equation 3.2**.

\[ y = mx + c \]

**Equation 3.3 A linear equation**

The correlation coefficient then tells us the strength of the relationship and the predictor variable gives us the form of the relationship whether the underlying behaviour is linear, logarithmic, exponential or power in nature. Most mathematical models that use correlation to establish their validity will use a function such as the one above to test their original research hypotheses. It should be noted that the Pearson product moment correlation coefficient is only applicable for multivariate normally distributed data and measures only linear relationships.

![Graph of Function versus Predictor](image)

**Figure 3.2 Example graph of predictor versus measured value**
To illustrate this point Figure 3.2 shows a graph of three functions against a predictor variable. As we can see from this the shape of the curves with respect to the predictor variables is different. The predictor variable in this case is F(x) = log(x) where x is monotonically increasing towards positive infinity. The graphs shows that a predictor variable graphed against itself will give a straight line or, using the Pearson product moment a correlation value of 1.0. The correlations with the predictor value in this case are as follows.

1. F(x) is Logarithmic = 1.0
2. F(x) is Cosinosoidal = -0.85
3. F(x) is Linear = 0.95

As we can see this measure can be considered a quantitative analysis of the form the relationship between a predictor and a criterion variable. We can also see that although the difference between the linear and predictor values is small and would sum close to zero in the round the correlation measure picks up a difference.

### 3.2.6 Behavioural Morphism Components

There are three component measures that are used in behavioural morphism.

- The model behavioural correlation component. \( R \)
- The model error component. \( E \)
- The model performance component. \( Y \)

The model correlation component is an objective quantitative measure of the difference in *behaviour* of the real and virtual tasks. The model error component is an objective and quantitative measure of the difference in the real task *error* and the virtual task *error*. The model performance component is an objective and quantitative measure of the real task *performance* and the virtual task *performance*.

Behavioural morphism relies on the presence of an established psychological model of the task or a set of control data that are collected from the real-world. In sections 3.2.6.1, 3.2.6.3 and 3.2.6.2 below each measure is further defined and its use demonstrated. The flowchart in Figure 3.3 shows a typical sequence for performing a behavioural morphism study into a task.
Figure 3.3 Behavioural morphism study sequence outline
3.2.6.1 The Behavioural Correlation Component \( R \)

To calculate the model correlation component the data or model from the original experiment should be researched. The original correlation coefficient (or a calculation of the correlation coefficient) should then be taken as the real-world value for the strength of validity of the real-world correlation mathematical model. Next an experiment must be undertaken to collect data on the task using the target virtual environment platform. This experiment must vary the same independent variable and measure the same dependent variable as in the original experiment. Once the data is collected the correlation between the independent and dependent variables should be calculated using the collected dataset.

The morphism measure for behaviour uses the square of the correlation coefficient or ‘coefficient of non-determination’ (Howell 1997) as the comparison factor and is shown in Equation 3.4 below. The coefficient of non-determination is defined as the variance in dependent variable not attributable to variance in independent variable and is a quantitative percentage value, in other words a quantitative measure of the error in the models predictive ability. We use this error as the comparison measure.

\[
R = \frac{1 - (r_{\text{virtual}}^2)}{1 - (r_{\text{real}}^2)}
\]

where

\[1 - (r_{\text{real}}^2) = \text{Real coefficient of non - determination}\]
\[1 - (r_{\text{virtual}}^2) = \text{Virtual coefficient of non - determination}\]

**Equation 3.4 The behavioural morphism model component definition**

Below, is an illustrative example of using the correlation measure using data obtained from one of the experiments detailed later in this thesis.

Real world model correlation \( r_{\text{real}} = 0.98 \)

Virtual world correlation \( r_{\text{virtual}} = 0.90 \)

\[
\therefore R = \frac{1 - (0.92)}{1 - (0.982)}
\]

\[
R = \frac{0.19}{0.0396}
\]

\[R \approx 4.8\]
What this measure indicates is any introduced variances by showing differences in correlation strengths. If the tasks are the same in the real and virtual worlds the correlations collected should not vary significantly, if they do this measure can quantify the amount of variance.

Using illustrative data the example shown above indicates that 4.8 times more of the variance in \( y \) cannot be explained by the variance of \( x \) in the virtual situation. This illustrative example shows that the model component quantifies any unknowns that the virtual environment has in comparison to the real-world by showing that the model dependent variable has a different variance in real and virtual worlds. In practice there are more statistical tests that should be carried out to validate the fact that any difference is not due to chance. Any case where \( R=1 \) could not be proved to be different using this methodology, i.e. we cannot prove the tasks are different.

There is a special case where the real-world correlation is less than the virtual world correlation or the negative of the real-world case. This is easily handled. Not all mathematical models correlate perfectly with the systems they are modelling; this is due to many factors such as a naivety about the underlying system in construction of the model. However when correlation is high these models still predict the system well. If the virtual correlation is larger than the real-world correlation we can interpret this as the model predicting the virtual world system better than it predicts the real-world system. Either we can revisit the model, which is the job of the Psychologist, or we treat the error as is and consider it as, again, a difference from the real-world. If a negative correlation exists the systems are trivially different and not really comparable, this too is a valid result and of use to the researcher.
3.2.6.2 The Model Performance Component \( Y \)

Whilst the model component makes use of the correlation coefficient the performance measure uses the native mathematical model task performance measure. For instance this could be reaction time in a reaction time model or search time in a visual search task. The measure integrates the absolute difference between real and virtual tasks across levels and then normalizes this value using the integrated real-world performance. Equation 3.5 shows the measure used for comparing performance in behavioural morphism.

\[
Y = \frac{\int \text{abs}(f_{\text{real}}(x) - f_{\text{virtual}}(x)) \, dx}{\int f_{\text{real}}(x) \, dx}
\]

where
\[
\begin{align*}
&f_{\text{real}}(x) = \text{Real performance} \\
&f_{\text{virtual}}(x) = \text{Virtual performance}
\end{align*}
\]

**Equation 3.5 The behavioural morphism performance component definition**

To illustrate the derivation of this measure consider the graph in Figure 3.4 which shows two abstract functions that represent the user performance at some task across levels. To gain an overall measure it is necessary to measure the area under each curve. This is done via integration and is shown in Figure 3.5.

![Figure 3.4 Real and Virtual performance functions](image-url)
Figure 3.5 Integrated real and virtual performance functions

We can then divide the absolute difference of these functions by the real value to give a performance measure that is normalized with respect to the real performance. This ratio is a measure of difference of the virtual performance against real performance; by dividing by the integrated real-world performance we gain a relative measure of the virtual performance in terms of the real-world over all the levels measured.

The illustrative calculations shown below demonstrate that the user performed with a degradation effect of 2.95. In other words the real-world performance in virtual was nearly three times as good as the virtual world performance. The example uses real data collected from one of the experiments described later in this thesis. The isomorphic or ideal case would be $Y = 1$. ID is the Fitts’ Law Index of difficulty term detailed later in this thesis.
Virtual Performance Component

\[ \int_{\text{Bottom}}^{\text{Top}} f_{\text{virtual}}(ID) \, dID \]

\[ \therefore \text{Since } ID \rightarrow x \]

\[ \int_{\text{Bottom}}^{\text{Top}} mx + c \, dx \]

Substituting collected data

\[ \int_{2}^{6} 384.4x - 456.3 \, dx \]

= 5131.44

Real Performance Component

\[ \int_{\text{Bottom}}^{\text{Top}} f_{\text{real}}(ID) \, dID \]

\[ \therefore \text{Since } ID \rightarrow x \]

\[ \int_{\text{Bottom}}^{\text{Top}} mx + c \, dx \]

Substituting collected data

\[ \int_{8}^{3} 98.7x + 12.8 \, dx \]

= 1739.98

\[ \therefore Y = \frac{5131.44}{1739.98} \approx 2.95 \]

3.2.6.3 The Model Error Component E

The measure error is measured directly using the standard error measure of the system at the tested levels of the independent variable. In the case of discrete modal errors (missing a button for instance) using a measure of the errors committed as a rate at a specific level of the independent variable. The integration (In a case where a discrete number of levels are tested this would be a summation operation.) is intended to allow the variance in error over all levels to have an effect on the end measure.

\[ E = \frac{\int e_{\text{virtual}}(x) \, dx}{\int e_{\text{real}}(x) \, dx} \]

where

\[ e_{\text{virtual}}(x) = \text{Virtual error} \]

\[ e_{\text{real}}(x) = \text{Real error} \]

Equation 3.6 The behavioural morphism error component definition.
An example of the use of the error component measure is shown below. It shows that the user had 5.25 times more error than in the real-world. Again this data is taken directly from one of the experiments detailed later in this thesis.

\[ \text{Real World Error } e_{\text{real}} = 0.04 \]
\[ \text{Virtual World Error } e_{\text{virtual}} = 0.21 \]
\[ \therefore E = \frac{e_{\text{virtual}}}{e_{\text{real}}} = 5.25 \]

3.2.7 The <R,E,Y> Vector

The REY vector is a shorthand notation for representing the fidelity of a virtual environment for allowing humans to perform a real-world task. Combining the illustrative results shown above into an illustrative <R,E,Y> (cf. AIP Cube (Zeltzer 1992)) vector gives us a vector <4.8,5.25,2.95>. This vector can be easily read, gives an intuitive understanding and tells us exactly how our environment allows subjects on average to behave at the task investigated. This vector is (approximately) interpreted as the average user having 4.8 times the unexplained variance in their behaviour in comparison with the real-world case, 5.25 times as much error and a performance that is nearly three times worse.

The ultimate aim for a virtual environment that is meant to reflect the real-world would be an isomorphism, however we cannot prove an isomorphism exists at a low level we can only show that no isomorphism exists by showing a non-unit REY vector or that a homomorphism, that can be quantified via the values in the REY vector, exists. A unit REY vector shows that we cannot prove that behaviour, error or performance varies from the real-world case. One can see that these measures could be used to address the impact system variables have on them by direct measurement. This allows a clear methodology with which to attempt to create a unit REY vector. At present no method such as this exists that takes into consideration behaviour. Indeed, no method such as this has been proposed. Using this vector an experiment could take place on differing machines and the impact of any platform differences.
would show up as a differing REY vectors. Once we have a REY vector we can then decide if the behaviour was close enough to allow comparison.

Fidelity difference could be considered as the differences in REY vectors. We can see that any system in which the error and performance was inferior to the real-world would result in a vector that was larger than the <1,1,1> and any system with better error and performance would result in a vector that was smaller. This measure is not intended to be a measure of each of the thousands possible factors that effect virtual environments what it is however is a datum with which to start comparing and identifying those factors and identifying them across platform. Further work is needed however into identifying whether these measures are linearly independent.

3.3 Conclusion

Behavioural morphism offers a way of characterising and quantifying human behaviour differences between real and virtual tasks. Since no other methodology has been proposed to carry out this measurement, an investigation into behavioural morphisms validity as a methodology should be investigated and reported.

The remaining part of this thesis now proposes that behaviour morphism is a valid method for assessing the fidelity of a task carried out within a virtual environment. Evidence supporting this thesis will be provided by experimentally (and in the first hypothesis case evidentially) by testing following hypotheses.

1. Human behaviour varies between real and virtual tasks.

2. Behavioural morphism is sensitive enough to compare the fidelity of real and virtual tasks

   (a) Behavioural morphism is sensitive enough to identify different behaviour between the real and virtual tasks

   (b) Behavioural morphism is sensitive enough to measure the fidelity of a task within a virtual environment where a mathematical model exists.

   (c) Behavioural morphism is sensitive enough to measure the fidelity of a task within a virtual environment where no mathematical model exists.
Chapter 4: Choice-Reaction Experiment
4.1 Introduction

The Hick-Hyman law is a model of human psychomotor behaviour that describes human performance at choice-reaction tasks. The model was developed in parallel by Hick (Hick 1952) and by Hyman (Hyman 1953); both investigated applied information theory in an attempt to quantify the uncertainty present in stimulus events. The Hick-Hyman law is stated below.

\[ RT = a + b \log_2 (N_{choice}) \]

Equation 4.1 The Hick-Hyman Law

Both the constant terms (a and b) in Equation 4.1 are dependent on the individual performance of a participant and can be collected statistically and used to infer a population through parametric statistics. Parametric statistics are used to determine behaviour across a population rather than on an individual basis. The Hick-Hyman law has been used extensively over the last thirty years within Human Factors, Psychology and Computer Science and as such has proved to be very robust. The Hick-Hyman Law is based on Shannon’s original work on information theory (Shannon 1949). Good accounts of the use of the Law and it’s applications can be found in (McCormick 1987), (Card et al. 1983), (Wilson 1990) and (Wickens 1992), all of these publications detail the use of the Hick-Hyman law and imply that it is extremely robust in describing human behaviour at choice reaction tasks.

4.2 Experimentation

4.2.1 Rationale

The experiment described in this chapter was carried out to identify the utility of using behavioural morphism to analyse a one-shot discrete motor skill task within a virtual environment, and to test the hypothesis that the psychological model known as the Hick-Hyman Law holds within a virtual environment. This task was chosen because of its fundamental nature within human psychomotor behaviour. Choice
reaction is a sub-task of almost all other movement tasks and therefore an important fundamental measure to validate.

This experiment tests the hypothesis that choice reaction behaviour varies between the real and virtual world. To support this hypothesis this experiment must show that the mean correlation coefficient gathered in the original experiment varies, in a statistically significant manner, from the sample of correlation coefficients gathered at a choice reaction task in a virtual environment. In other words this experiment tests to see if virtual choice reaction behaviour varies significantly from the accepted behavioural model (Hick 1952 and Hyman 1953).

This experiment is also used as supporting argument for the hypothesis that behavioural methods can be used to assess the fidelity of a virtual environment by showing that behaviour in virtual environments can be measured in a quantitative fashion and therefore used as a comparison measure.

4.2.2 Methodology

The experiment used a within-subjects repeated measures (8 x 20) factorial design that required the participants to respond as quickly as possible to a series of stimuli presented in a virtual environment. A within subject design was used so the participants behaviour could be tested several times at the same level and also compared across levels, a within subject design cancels out differences across subjects and allows small individual differences to become apparent. The reaction time that related response with stimuli was recorded using a virtual environment. The experimental layout consisted of three sets of blocks placed on a table (Figure 4.1) within a virtual environment. The blocks were organized into a stimulus block set, a response block set and a home block (Figure 4.2). The arrangement was such that the movement distance from the home block to all of the response blocks was the same.
The number of blocks present in the stimulus and response block sets was varied to give a number of stimulus and response choices from 1 to 8. Each choice level was tested 20 times in eight different trial blocks. To avoid order effects each trial was presented in a pseudo-random order by an automated lottery process that was programmed into the virtual environment controlling software (Coolican 1996; Howell 1997).

The reaction time measured was a totally “virtual reaction” where the response duration was measured when a virtual block had been pressed by a virtual hand in response to the virtual stimulus. This is unlike other studies (Lampton et al. 1994) where the reaction was measured using the physical interaction device itself and is still a real-world reaction.
All the models were prepared using 3DStudioMax R2.0 and conventional room and furniture dimensions. The size of the blocks and distance of movement of the subject's hand during the experiments were established during the experimental pilot phase. A full description of the creation and design of this virtual environment are available in Section 4.3.

4.2.2.1 Experimental Procedure

Each trial consisted of the participant being presented with a trigger stimulus (a lit stimulus block) and responding by carrying out an action that was dependent upon the initial stimulus (touching a response block); the classic choice reaction. In this experiment the participant was required to press the response block that had the same number on it as the stimulus block. All participants were asked to carry out the response as quickly as possible with an emphasis on speed.
All participants undertook identical tutorial exercises prior to the experiment. This tutorial consisted of a random but single presentation of each level; this ensured the participants were fully aware of the experimental procedure and familiar with the 6DOF controller.

When the participant was comfortable and ready to proceed with a trial they placed the virtual hand so it touched the home block and turned it green (Figure 4.3). After a pseudo-random period of between 1 and 3 seconds one of the stimulus blocks was illuminated, the participant then responded by attempting to press the correct button from the response block set arranged on the desk as quickly as possible. The pseudo-random period was present since a similar pseudo random period was present in the original experiments.

In order to maintain a consistent start point the participant had to start each trial with their virtual hand touching a 'home' block. Their hand had to return to the home block before the computer would allow the next trial to begin. The stimulus and response
blocks were arranged so that the stimulus-response mapping was reasonably strong (Boff and Lincoln 1988; Wickens 1992). The block-trial cycle is shown in Figure 4.4 below. This cycle was repeated 20 times during a block to fulfil data collection normality requirements. The number of required samples was derived during the pilot process by evaluating the power of the experiment using statistical tests available in the SPSS statistical software.

Figure 4.4 The block-trial cycle.

4.2.2.2 Order Effects

The algorithm used to control the experiment used the ANSI 'C' library pseudo-random number generator to decide the order of presentation in each block thus simply counterbalancing the results for order effects. For each participant the algorithm generated 8 arrays of 20 values, the first array contained 1's, the second 2's the third 3's etc. The arrays represented the blocks and the elements in the arrays represented the trials, the pseudo-number generator was then used to reorganise the order of the arrays by swapping elements. This randomisation method is a variant of the 'lottery' method mentioned by many statistical references (Coolican 1996; Howell 1997). Using this method ensured that for each of the eight rounds of trials there were potentially $B \times N!$ alternative orderings that could be presented, this effectively
statistically randomised the order of presentation between subject and trial. A full account of the counterbalancing technique is detailed in section 4.3.2.1.

4.2.2.3 Apparatus

The experimental apparatus consisted of a Silicon Graphics Inc Indigo² Max Impact with R4400 MIPS processor 64MB RAM and 4 MB TRAM. The software was programmed for this experiment using Division's dVS/dVISE API v 6.0, the C programming language and the Silicon Graphics' IRIX v6.2 operating system. The stereo condition was provided via a pair of commercially available shutter glasses. All trials were run with a screen frame update rate of 60Hz with the Polhemus tracker running at 100Hz and the display set at 1280x1024 pixel resolution. All participants used a 6DOF controller that contained a Polhemus Fastrak device to control their virtual hand.

4.2.2.4 Subjects

Twelve unpaid participants volunteer were used for the experiment. None of these had any previous experience of desktop virtual environments and they were all drawn from the Loughborough University population. The age ranged from 20 – 35 (Mean = 24.37 Std.Dev = 4.28), all were right-handed and male. Prior to the experiment the subjects' eyesight was tested for colour blindness, acuity, stereo-acuity and FOV. The subjects also had their inter-pupillary distance measured to enable the correct stereo condition configuration to be set in the experimental software prior to exposure. A consent form was used to screen participants for health conditions that may make it unacceptable for them to take part in the trials. Each subject was asked to read and sign the consent form prior to participating.

4.2.2.5 Data Collection

The data were collected using collision algorithms provided by the Division API, the software generated a software callback when particular buttons were pressed. The data were collected via a linked list programming technique (Cormen et al. 1989; Gersting 1993) within the programs themselves. The list was then written out to file after the experiment. The linked list technique enabled a dynamic random access memory based structure to collect the data without the need to write to disk during the experiment, this negated as far as possible disk accessing which may have affected the
virtual environment system performance by causing the rendering pipeline to be interrupted. The data that were collected for each trial were:

- Time of virtual hand uncolliding with the home button
- Time of virtual hand collision with the target button
- Button Pressed
- Button Highlighted
- Position in Array
- Number of Choices

For further information about the data collection algorithms and methods see section 4.3.2.3.

4.2.2.6 Data Analysis

The choice of statistical analysis software was SPSS v9.0. For initial ease the experimental program had been programmed to automatically generate C.S.V. (Comma Separated Value) Excel compatible files. To analyse the data they first had to be manipulated from their Excel form into a form that SPSS could recognise, this was done using an Excel macro, this delegated the majority of the hard data translation work to the computer. Statistical outliers were then removed from the data by converting the data to Z scores and eliminating pertinent data with Z-scores higher than three (Coolican 1996; Howell 1997). Not removing the outliers would have skewed the results that were collected by allowing spurious values to be included in any calculations. For further information about the data output methods see section 4.3.2.3.

4.2.3 Results

The graph in Figure 4.5 shows the results of mean reaction duration between stimulus presentation and the point at which the participants' virtual hand stopped touching the home block i.e., the choice reaction. This shows, as the original experiment showed, a highly significant correlation (Table 10-1) \((p < 0.01)\) between mean subject reaction time and the predictor value of \(\log_2(N_{\text{choices}})\). A significant correlation \((p < 0.05)\) is also shown in (Figure 4.6 and Table 4-2), this correlation shows the duration between stimulus presentation and the participant pressing the correct response block.
Behavioural Morphisms In Virtual Environments

Choice Reaction Regression (Home Block, Mean Subject)

Figure 4.5 Regression analysis (Home block reaction)

Hick-Hyman Law Regression Model

<table>
<thead>
<tr>
<th>Model</th>
<th>R</th>
<th>R Square</th>
<th>Adjusted R Square</th>
<th>Std. Error of the Estimate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Log(N+1)</td>
<td>.971a</td>
<td>.942</td>
<td>.933</td>
<td>.0173</td>
</tr>
</tbody>
</table>

*Predictors: (Constant), INFORM

Table 4-1 Correlation table for leaving home block

Figure 4.5 above shows the mean reaction time (the period between stimulus lighting and hand leaving the home block) against the Bit per Stimulus \( \log_2(\text{N}_{\text{choices}}) \) where \( N\) = number of choices. The graph also shows a regression line fitting the reaction time curve. A relationship can clearly be seen. This correlation of the predictor and criterion variables is shown in Table 4-1.
Figure 4.6 Regression analysis (Response block reaction)

Hick-Hyman Law Regression Model

<table>
<thead>
<tr>
<th>Model</th>
<th>R</th>
<th>R Square</th>
<th>Adjusted R Square</th>
<th>Std. Error of the Estimate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Log(N+1)</td>
<td>.916(^a)</td>
<td>.840</td>
<td>.813</td>
<td>0.0379</td>
</tr>
</tbody>
</table>

\(^a\) Predictors: (Constant), INFORM

Table 4-2 Correlation table for hitting response block

Figure 4.6 above shows the mean reaction time (the period between stimulus lighting and hand hitting the response block) against the Bit per Stimulus (\(\log_2(N_{choices})\)) where \(N=\) number of choices). This graph also shows a regression line fitting the reaction time curve. This correlation of the predictor and criterion variables is shown in Table 4-2.

From Figure 4.5 we can see that the Hick-Hyman is a good predictor of human behaviour at choice-reaction tasks in a virtual environment. But we wish to know if the real-world situation and the virtual situation are the same. We do this by
comparing the mean correlation coefficient of the original experiment with the sample we have collected. Table 4-3 below shows the correlation coefficients collected for our participant sample in the virtual environments. However a straightforward comparison is not possible since the population of all correlation coefficients are negatively skewed (Howell 1997).

![Table 4-3 Untransformed subject correlation coefficients](image)

Table 4-3 Untransformed subject correlation coefficients

To enable to compare the original mean correlation this sample must be transformed into a Normal distribution using the 'Fisher transform' (Howell 1997). This formula redistributes a negatively skewed formula into a normal distribution.

$$R' = 0.5 \times \log_2 \left( \frac{1 + r}{1 - r} \right)$$  \textbf{Equation 4.2 Fisher Transform}
The Fisher transformed correlation coefficients are shown in Table 4-4. It is these data that were used for the statistical comparison with the original Hick-Hyman experiment.

Table 4-5 shows a one-sample T-Test that compares the means of the sample in correlation of the original experiment in (Hyman 1953) with the sample in Table 4-4 that were collected during the virtual choice reaction experiment. The tests show highly significant difference ($p < 0.01$) between the original mean correlation coefficient for the experiment and the virtual environment data.
Paired Samples T-Test Comparison of Performance

<table>
<thead>
<tr>
<th>Pairs</th>
<th>Mean</th>
<th>Std</th>
<th>Std Err</th>
<th>Lower</th>
<th>Upper</th>
<th>t</th>
<th>df</th>
<th>Sig (2-tailed)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pair 1</td>
<td>.1698</td>
<td>.020</td>
<td>.0325</td>
<td>.0929</td>
<td>.2467</td>
<td>5.219</td>
<td>7</td>
<td>.001</td>
</tr>
<tr>
<td>Pair 2</td>
<td>.6075</td>
<td>.028</td>
<td>.0293</td>
<td>.5383</td>
<td>.6767</td>
<td>20.76</td>
<td>7</td>
<td>.000</td>
</tr>
</tbody>
</table>

Table 4-6 Hypothesis Test (Performance)

Table 4-6 shows the comparison of the mean performance of the original experiments with the mean performance recorded during the virtual experiment; again there is a significant difference (p< 0.01) in comparison with the original experiment. These tests show a significant difference between performance and behaviour in comparison with the original experiments.

4.2.4 Results Discussion

4.2.4.1 Behaviour

By comparing the mean correlation coefficient of the original experiments with the samples we have collected within our virtual experiment it has been shown that there are significant differences in model correlation between our results and the experiment that provided the accepted model of choice reaction. That is not to say that the Hick-Hyman model is invalid within a virtual environment merely that we have shown that the strength of correlation between the predictor and criterion variables is significantly less than in the real-world. This allows us to reject the null hypothesis of isomorphism between real and virtual world behaviour and accept the alternative hypothesis that there may well be difference in behaviour. By showing that the correlation coefficient is different from the original experiment in a statistically significant manner we have allowed the possibility of differing behaviour in a virtual environment to appear.
4.2.4.2 Performance

The performance of participants at the choice-reaction task showed significant differences ($p<0.01$) between the real-world experiments and our virtual experiment. This allows us to reject the null hypothesis of isomorphic performance between real and virtual experiments. Regressing to the simple reaction case the performance is nearly 2.5 times as long as the real-world. However the difference decreases as the information content rises, this again supports the hypothesis of a non-isomorphic relationship between real and virtual tasks by demonstrating a statistically significant variance in performance. However, the original experiment used different motor skills to indicate a reaction and this may nullify this particular result. One can argue this may or may not be the case.

4.2.4.3 Error

There were no errors committed during the course of this experiment. This is probably due to the simple nature of the task and the over learning of skills such as this in the real-world.
4.3 Implementation

4.3.1 Abstraction and Implementation

It had been decided whilst reviewing the literature on choice reaction experiments (Hick 1952; Hyman 1953; Poulton 1974; Card et al. 1983; Boff and Lincoln 1988; Wickens 1992) that the choice reaction experiment task should take the form of a participant selecting a response button from an array of response buttons after a stimulus button had been enlightened. The participant should be instructed to carry out the task quickly and accurately with the emphasis on speed. The task virtual environment layout is shown in Figure 4.1 and Figure 4.2 above. This layout was chosen because it allowed a symmetrical use of flexor and extensor movements depending on response button and meant that the distance between the home and response blocks were identical. This choice meant that using statistically random presentation of the movements the effects of performance differences between flexor and extensor movements could be reduced.

The trial procedure was designed so that the user was in charge of the initiation of the individual trials. The participant indicated their readiness to proceed to the next trial by placing their virtual hand onto a home block in the virtual environment to trigger the start of the next trial. A representation of a typical choice reaction trial cycle is shown in Figure 4.4 above. A functional decomposition of the trial cycle was derived and is shown in Figure 4.7. The functional decomposition was used as a base for the software implementation of the virtual trials. The complete program listing for the block and trial control code implementation is available in Appendix 2 Choice Reaction Environment Code.

There are main aspects of the choice reaction implementation that are of special interest. First is the configuration of the trial environment stimulus and response blocks in space programmatically. The second is the method that was employed to counterbalance the experimental order of the blocks and trials programmatically. The third is the data collection methodology. All of these areas will be discussed in the following section.
Figure 4.7 Choice reaction trial abstraction
4.3.2 Logic implementation

The control logic of the choice reaction experiment had to oversee the trial configuration, trial ordering and data collection so the experimental requirements were met. These requirements were driven by the need to counterbalance the experiment and the need to collect an acceptable amount of data with respect to the central limits theorem (Howell 1997). These requirements were assessed during the pilot phase by using power tests within SPSS. The main requirements were:

- An array of reaction blocks from 1 – 8 must be presented randomly.
- The positions of the stimulus block within the array must be random.
- Each position in an array must be equally tested.
- Reaction time at each number of choices must be tested 10 times.
- Record the reaction time.
- Record the stimulus button.
- Record the Response button pressed.
- Pause after each trial until the participant indicates readiness.
- Output the data in a format that could be easily manipulated.

4.3.2.1 Counter Balancing Technique

The choice reaction trial logic was required to randomise out the biases that could be present due to presentation of trial order and position. This is known as experimental counterbalancing (Coolican 1996) and is done to prevent ordering and learning effects from confounding the experiment. This process was carried out automatically by the software at the start of each experiment and guaranteed a completely different set of trial orderings for each participant. Conceptually, the counterbalancing operation can be viewed as three stages.

2) Create a 2D Array (Block x Trials) with
   a) A 1D Array for each number of Choices level (1,2,3,4,5,6,7,8)
   b) 1 Cell for each trial (20)
3) Randomise the position tested by running a lottery in each trial cell
   a) Number of lottery entrants = Number of choices
4) Randomise the array as a whole by randomly swapping pairs of trial cells
The functions that carried out this algorithm are presented in Code Section 4.8 and Code Section 4.9 below. `fillArrayO` filled the 2D array so each 1D array held the upper limit of the number of choices, `randomizeArrayO` then ran the lottery and pair swapping process.

```c
void fillArrayO
{
    int x,y;
    srand((unsigned)time(NULL));
    for (x=0; x<BLOCKS; x++)
    {
        for (y=0; y<TRIALS; y++)
        {
            position[x][y] = x+1;
        }
    }
}
```

**Code Section 4.8 fillArrayO Function**

```c
void randomizeArrayO
{
    int x, y, tempswap, pos1x,pos2x,pos1y,pos2y;
    for (x=0; x<NUMRAND; x++)
    {
        pos1x = (int)(rand()% BLOCKS);
        pos1y = (int)(rand()% TRIALS);
        pos2x = (int)(rand()% BLOCKS);
        pos2y = (int)(rand()% TRIALS);
        tempswap = position[pos1x][pos1y];
        position[pos1x][pos1y] = position[pos2x][pos2y];
        position[pos2x][pos2y] = tempswap;
    }
}
```

**Code Section 4.9 randomiseArrayO function**
The results of these two functions can be seen in Table 4-7 and Table 4-8. Table 4-7 shows the state of the 2D array after the lottery has been carried out on each 1D array (Columnar Array) and Table 4-8 shows the state of the array after the 10,000 swaps have taken place. The result of this operation is a 400 cell randomised array that can be addressed monotonically to derive a trial configuration. Using this technique experimental biases due to ordering, stimulus position, and motor movement preference due to response position have been reduced as far as is possible by randomisation. Using the ANSI standard ‘C’ library Srand(NULL) function (Code Section 4.8) to seed the current system ensured the random number generator was secure and not repeating within its period.

<p>| | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>5</td>
<td>6</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td>4</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>5</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>6</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>7</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>6</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>1</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>2</td>
<td>8</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>7</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>6</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>3</td>
<td>4</td>
<td>2</td>
<td>6</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>5</td>
<td>4</td>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>5</td>
</tr>
</tbody>
</table>

Table 4-7 Choice position randomly chosen
Table 4-8 Randomised block set of random choice positions

The 2D array was then used throughout the experiment to look up the correct trial configuration. Once the participant had started a trial the virtual environment software obtained the trial configuration directly from this array and configured the virtual environment accordingly. The virtual environment configuration process is described in the next section.

4.3.2.2 Environment Configuration Logic

At the start of each trial the control software was responsible for configuring the positions of the geometric objects (Table, participant, stimulus and response blocks etc) in the virtual environment. This was done to enable the correct trial scheduled by the block/trial array mentioned in the section above to take place. After this look-up phase the software translated the stimulus response blocks into the correct position using the following function.
void arrangeStimulusBlocks(int number)
{
    int x; dmPoint pos;
    char mat[] = "Letters:A";

    for (x=0;x<number;x++)
    {
        mat[8]=(char)(x+65);
        pos[0]=(x*0.05)-(0.025*(number-1));
        pos[1]=0.76; pos[2]=-0.22;
        VCEntity_SetPositionPoint(Stimulus[x],pos);
        VCVisual_SetFrontMaterial(stimulusVis[x],mat);
    }
}

Code Section 4.10 Stimulus block arrangement function

Using this function the array of stimulus blocks were positioned correctly and had the correct letter assigned as a material. The response blocks were then generated and translated into position. These block positions were calculated using the following function. This positioning ensures that the distance to each response block is the same from the block that the user rests their virtual hand on to start the trial i.e. arranged in a semi circle.
void arrangeResponseBlocks(int number)
{
    int x;
    char mat[]="Letters:A";
    dmPoint pos;

    for(x=0;x<number;x++)
    {
        mat[8]=(char)(x+65);

        if(number!=1)
        {
            pos[0]=0.27*sin((3*PI/2)+((PI/(number-1))*x));
            pos[1]=0.75;
            pos[2]=(-0.27*cos((3*PI/2)+((PI/(number-1))*x)))+0.15;
            VCEntity_setpositionPoint(plate[x],pos);
        }
        else
        {
            pos[0]=0.0;
            pos[1]=0.75;
            pos[2]=-0.135;
            VCEntity_setpositionPoint(plate[x],pos);
        }
        VCVisual_SetFrontMaterial(plateVis[x],mat);
    }
}

Code Section 4.11 Used block arrangement function

4.3.2.3 Data Collection Logic and Output

For each trial the function in Code Section 4.12 below was called once a response block had been touched. This function first sets the response block colour to red to indicate to the user that the button has been touched and then records the time in microseconds, the number of the button pressed and the number of the button that was the actual stimulus.
void touch(VCBodyAppCallbackData *cd, void *data)
{
    struct timespec stopTime;
    VCAudio_Start(click);
    printf("touched\n");
    if (cd->entity==Button)
    {
        touchButtonFlag=1;
        VCVirtual_SetFrontMaterial(ButtonVis, "Room:Red");
        return;
    }
    if (WaitingForReaction==1)
    {
        clock_gettime(CLOCK_REALTIME,&stopTime);
        current->pressStop.tv_sec=stopTime.tv_sec;
        current->pressStop.tv_nsec=stopTime.tv_nsec;
    }
    if (cd->entity==plate[0])
    {
        current->pressedCharacter='A';
        VCVirtual_SetFrontMaterial(plateVis[0], "Room:Red");
    }
    if (cd->entity==plate[1])
    {
        current->pressedCharacter='B';
        VCVirtual_SetFrontMaterial(plateVis[1], "Room:Red");
    }
    if (cd->entity==plate[2])
    {
        current->pressedCharacter='C';
        VCVirtual_SetFrontMaterial(plateVis[2], "Room:Red");
    }
    if (cd->entity==plate[3])
    {
        current->pressedCharacter='D';
        VCVirtual_SetFrontMaterial(plateVis[3], "Room:Red");
    }
    if (cd->entity==plate[4])
    {
        current->pressedCharacter='E';
        VCVirtual_SetFrontMaterial(plateVis[4], "Room:Red");
    }
}
if (cd->entity==plate[5])
{
    current->pressedCharacter='F';
    VCVisual_SetFrontMaterial(plateVis[5], "Room:Red");
}
if (cd->entity==plate[6])
{
    current->pressedCharacter='G';
    VCVisual_SetFrontMaterial(plateVis[6], "Room:Red");
}
if (cd->entity==plate[7])
{
    current->pressedCharacter='H';
    VCVisual_SetFrontMaterial(plateVis[7], "Room:Red");
}
if (WaitingForReaction==1)
{
    WaitingForReaction=0;
    removeUnwantedBlocks(0);
    checkForDataOrExit();
}

**Code Section 4.12 Touch function callback**

This data was then added to a linked list structure and the linked list structure extended by one link. Once all the experimental trials had been exhausted this linked list was then written to disk using the function in Code Section 4.13. This function outputs an ASCII file that can be imported into Excel using the inserted delimiters to provide the data structure within Excel.

```c
void End()
{
    /*
    This function outputs the data in the linked list to a file
    */
    FILE *fp;
    char str[80];
```
int epochvalue;
double secstart, secstop, nanostart, nanostop;
current = head;

/*Calculate durations involves transforming from long ints to double floats*/

while (current->nextCollision != NULL)
{
    secstart=(current->start.tv_sec);
    secstop=(current->pressStop.tv_sec);
    nanostart=(current->start.tv_nsec);
    nanostop=(current->pressStop.tv_nsec);
    nanostart=(nanostart/1000000000);
    nanostop=(nanostop/1000000000);
    secstart = secstart+nanostart;
    secstop = secstop+nanostop;
    current->pressButtonDuration = secstop-secstart;

    secstart=(current->start.tv_sec);
    secstop=(current->leaveStop.tv_sec);
    nanostart=(current->start.tv_nsec);
    nanostop=(current->leaveStop.tv_nsec);
    nanostart=(nanostart/1000000000);
    nanostop=(nanostop/1000000000);
    secstart = secstart+nanostart;
    secstop = secstop+nanostop;
    current->leaveButtonDuration = secstop-secstart;

    current = current->nextCollision;
}

/*Repoint to head of list to output data*/
current = head;

/*
This is where the data is put into the file
 copy directory path and append filename from command line*/

strcpy(str, "data/choice/");
strcat(str, filename);
if((fp = fopen(str,"w"))==NULL) {
    printf("Error opening file\n"); 
    exit(1); 
}

fprintf(fp,"B,T,C\tStart\tReact\tPress\tReactDur\tPressDur\tSearch\tPreChar\tPos\n");

while (current->nextCollision !=NULL) {
    fprintf(fp,"%d@%d@%d\t",current->Block,current->Trial,current->NumberOfBlocks); 
    fprintf(fp,"%u.%u\t",current->start.tv_sec,current->start.tv_nsec); 
    fprintf(fp,"%u.%u\t",current->leaveStop.tv_sec,current->leaveStop.tv_nsec); 
    fprintf(fp,"%u.%u\t",current->pressStop.tv_sec,current->pressStop.tv_nsec); 
    fprintf(fp,"%f\t",current->leaveButtonDuration); 
    fprintf(fp,"%f\t",current->pressButtonDuration); 
    fprintf(fp,"%c\t",current->searchCharacter); 
    fprintf(fp,"%c\t",current->pressedCharacter); 
    fprintf(fp,"%2d\n",current->Position); 
    current=current->nextCollision; 
}

fprintf(fp,"%s\n",filename); 
fclose(fp); 
VCVisual_SetFrontMaterial(ButtonVis, "Room:Orange"); 

Code Section 4.13 Data output function
Once imported into Excel this ASCII file produces a worksheet that has columns of data that can then be treated by prewritten macro operations. The macro operations organise the data and produce the data required for the statistical analysis.

### 4.3.3 Verification and Validation

To verify and validate the virtual environment design for the choice reaction experiment was undertaken. This involved three pilot participants undertaking the experiment and running the data produced through the data collection, data output, data manipulation and statistical test functions that had been constructed. This process ensured the following:

- *The experimental control was straightforward and usable*
- *The participant could control the trial execution*
- *The data output was correct*
- *The data was in a format that could be analysed*

After the pilot process had been undertaken the results were investigated only for methodological reasons. These results were not included in the final experimental study.

The only problem encountered during the pilot phase of the choice reaction experiment was in the import stage of the Excel spreadsheet. This problem was to do with the use of the ‘@’ symbol in the input stream as a delimiter, this function clashed with a native Excel delimiter and caused data formatting problems. This problem was solved by swapping the ‘@’ delimiter with the comma (,) symbol in the data output function of the virtual environment software.
4.4 Chapter Conclusion

The experiment detailed in this chapter demonstrates a significant difference \((p<0.01)\) in the strength of correlation between dependent and independent variables between the experiment that defined the accepted model and the virtual. This allows us to support the hypothesis that the behaviour between real and virtual may be different. It shows that some of the variance in reaction time in a virtual environment is not due to the predictor variable of number of choices. What this experiment does not demonstrate clearly is an actual behavioural difference, only the possibility of one. The correlation coefficient is still significant for the Hick-Hyman law in a virtual environment and it still predicts human choice reaction behaviour well. This experiment also showed a significant difference in performance when the virtual task is compared with the original and subsequent real-world experiments; however, the performance measure is called into question due to the fact that the original experiments and the virtual experiment used different motor skills to indicate a reaction.

The choice reaction experiment showed that the behavioural morphism measure for behaviour (The correlation component) could be applied to an open-loop discrete ballistic motor skill and pick up the possibility of differences in behaviour from the real-world situation. This in turn still supports the thesis that behavioural morphism measures are necessary due to differences in behaviour that may not be picked up by analysing conventional measure such as performance and error rates. This experiment also supports the thesis that behavioural morphism can utilize accepted psychological models to determine possible behavioural differences in open-loop discrete ballistic motor skills.

From a practical point of view however, this experiment has also demonstrated that the method of behavioural morphism where a mathematical model exists requires significant investment in the form of algorithm design, subject testing, statistical analysis and data manipulation.
5 Chapter 5: Rapid-Aimed Movement Experiment
5.1 Introduction

Fitts Law (Fitts 1954) is a very successful model of human psychomotor behaviour. Fitts Law is a predictive model that allows model user performance in rapid-aimed movements with varying movement amplitudes and target widths. Fitts’ law has been used to determine the effectiveness of interaction devices in many studies (MacKenzie 1992; Arthur et al. 1993; McGee et al. 1997; Plamondon 1997).

MacKenzie (1992) has pointed out that Fitts’ law ‘remains an analogy waiting for a theory’. This statement strikes a chord since as with the Hick-Hyman law mentioned in the experiments above the base model Fitts’ law relies upon Shannon’s communication theory and not a theorem of human movement. In Fitts’ law the human is considered an information processor with a maximum bandwidth and it is this bandwidth or information capacity that determines human performance at rapid aimed movements. Equation 1 below gives the fundamental form of Fitts’ law. There are a few variations on this form most of which are covered by MacKenzie or Plamondon. Mackenzie also provides an excellent account of the underlying theory.

\[ MT = a + b \log_2 \left( \frac{2A}{W} \right) \]

Equation 5.1 Fitts' Law

\[ A = \text{Amplitude} \]
\[ W = \text{Width} \]

The independent variable of interest is known as the index of difficulty (ID) and its derivation is shown in Equation 5.2 below.

\[ ID = \log_2 \left( \frac{2A}{W} \right) \]

Equation 5.2 Index of difficulty

\[ IP = \frac{ID}{MT} \]

Equation 5.3 Index of performance
From the above equations it can be seen that calculating the MT over differing ID allows us to form a regression line with slope and intercept coefficients. The datum of performance in used in this experiment is the based on the interpretation that devices are interacting with identical systems.

5.2 Experimentation

5.2.1 Rationale

This experiment is identical in methodology to the original experiment that was carried out by Fitts. Unlike the choice reaction experiment which established that the behavioural measures can be used to pick up possible behavioural differences between the real and virtual tasks this experiment looks at whether the behavioural morphism measures can pick up differences between tasks in different virtual environments by using the real-world model as a datum for comparison.

The null hypothesis for this experiment is that the behaviour, performance and error at a Fitts’ tapping task do not vary when cues such as collision stimulus and retinal disparity (stereo vision) are varied.

5.2.2 Methodology

5.2.2.1 Design

The experiment was 2x3 factorial within-subjects repeated measures design that was modelled on the original experiment. The 2x3 factorial design was chosen to allow comparison of stereo cues and collision stimulus cues without the need for the large amount of subjects that would have been needed for a between subjects design. The participant carried out a discrete serial tapping task between two plates. The subject moved a virtual stylus between two virtual plates via a 6DOF joystick. The plates were varied through ten width/amplitude combinations (indexes of difficulty) in a desktop virtual environment. Ten levels were chosen to enable any correlation measures to be based upon a reasonable number of candidate points that included overlapping indexes of difficulty. The aim was to tap as quickly as possible between the two tapping plates.
The factors of the experiment were the availability to the subject of retinal disparity (binocular) and collision notification stimuli. Two viewing conditions existed one with stereo-glasses providing a depth cue and one with them switched off, the participants wore the glasses for both experiments. Three virtual environment designs were used, these environments were identical in all but the stimuli that notified the participant, in the absence of haptic feedback, that they had collided with the tapping plate. One environment gave a visual cue, one an audible cue and one gave no cue at all. The environments were identical in every other respect. Overall, 25200 taps were observed.

After the data were collected the mean movement times were calculated and a regression line constructed between mean movement time (Dependent variable) and Fitts ID (Independent variable). Each subject would end up with six values, one for each experimental condition. These were then placed into a two way repeated measures ANOVA.
All subjects undertook a tutorial exercise before they could progress to the experiment. Training was stopped once the performance at the tutorial exercise was not significantly different between following trials. The order of presentation of the conditions was counterbalanced to reduce the impact of any residual learning effect that the tutorial had not cancelled. For full information about the design and implementation of the experimental virtual environment see section 5.3.

### 5.2.2.2 Stimuli

In each viewing condition three collision notification stimuli conditions were presented to the subject to indicate a collision between the probe and the contact plate.

- An audible warning was emitted.
- Probe turns red plates turn green.
- No stimulus given.
Since there was no possibility of controlling the haptic/kinaesthetic feedback from the joystick the choices of collision feedback consisted of the visual and auditory modalities. The visual stimulus turned the plate red and the probe green. These stimuli were chosen because they would reflect the type of substitute stimulus available to the virtual environment designer in the absence of force feedback systems and the type of stimuli available within commercial VE systems.

5.2.2.3 Apparatus

The experimental apparatus consisted of a Silicon Graphics Inc Indigo\textsuperscript{2} Max Impact with R4400 MIPS processor 64MB RAM and 4 MB TRAM. The hand position was detected using a 6DOF (Degree Of Freedom) joystick from Division that contained a Polhemus Fastrak tracker at 60Hz. The software was programmed using Division's dVS/dVISE API, the C programming language and the Silicon Graphics' IRIX 6.2 operating system. The stereo condition was provided via a pair of commercially available shutter glasses. For the stereo condition the Silicon Graphics Inc monitor was set up to interface the images using the IRIX 6.2 'setrnon' command. Both stereo and mono trials were run with a screen frame update rate of 60Hz and with the Polhemus tracker running at 60Hz and the display set at 1280x1024 pixel resolution.

5.2.2.4 Subjects

Twelve unpaid participants were used in the experiment. None of these had any previous experience of desktop virtual environments and they were all drawn from the University population. The age ranged from 22 - 40, all were right-handed and male. Prior to the experiment the subjects' eyesight was tested for colour blindness, acuity, stereo-acuity and FOV. The subjects also had their IPD (Inter-Pupillary Distance) measured to enable the correct stereo condition configuration to be set in the experimental software.

5.2.2.5 Data Collection

The data were collected by using collision algorithms provided by Division to trigger bespoke collection algorithms. The data were recorded in a linked list data structure within the programs themselves. The linked list was then written out to file after the experiment was completed. The linked list technique enabled a dynamic RAM based
structure to collect the data without the need to write to disk during the experiment negating as far as possible disk accessing which may have affected the virtual environment system performance. Statistical outliers were removed from the data by converting the data to z scores and any points with scores higher than three were eliminated.

5.2.3 Results

Once the data were collected and treated for outliers they were then fed into Excel spreadsheets and a regression analysis carried out to identify the slope of the line between MT (movement time) and ID. Outliers were identified using the techniques discussed in (Howell, 1997 and Coolican, 1996). This in essence allows outliers with a Z score of more that 3 to be classed as spurious points and omitted from the statistical analysis.

The reciprocal of the slope is the measure used in this experimental treatment to compare the different conditions, it is known as the 'Index of Performance' (IP). This method has been used by many studies before to assess the performance of interaction devices. The IP for each subject for each condition was calculated and then analysed using a repeated-measures ANOVA. This process was repeated for the correlation coefficients collected for the adherence to Fitts' Law.

The tables below show the statistical test for the experiment. The extra terms in the tables such as 'Roy's Largest Root' or 'Wilks' Lambda' (SPSS Users Manual Version 9.0) relate to statistical weighting factors that provide a method of handling data that is non-normal. Since our data was normal these are identical. If the data was non-normal we would choice a factor depending on how conservative we wished the test to be.

5.2.3.1 Behaviour
The first test to be carried out was a 'T' test comparison of the mean correlation coefficient gathered in the original Fitts experiment with the correlation coefficients gathered in this experiment. The results can be seen in Table 5-1, it shows that the strength of correlation for the Fitts' model varies significantly with the original
experiment (p< 0.01 on all comparisons). This is taken as evidence to support the notion of differing behaviour between real and virtual tasks.

### One-Sample T Test Against Original Fitts' Correlation Coefficient

<table>
<thead>
<tr>
<th></th>
<th>t</th>
<th>df</th>
<th>Sig. (2-tailed)</th>
<th>Mean Difference</th>
<th>95% Confidence Interval of the Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Audio Stereo</td>
<td>-19.720</td>
<td>11.000</td>
<td>6.217E-10</td>
<td>-7.84</td>
<td>-8.49 to -7.23</td>
</tr>
<tr>
<td>Audio Mono</td>
<td>-9.481</td>
<td>11.000</td>
<td>1.258E-06</td>
<td>-7.01</td>
<td>-7.64 to -6.38</td>
</tr>
<tr>
<td>Visual Stereo</td>
<td>-9.510</td>
<td>11.000</td>
<td>1.218E-08</td>
<td>-8.42</td>
<td>-9.03 to -7.83</td>
</tr>
<tr>
<td>Visual Mono</td>
<td>-12.615</td>
<td>11.000</td>
<td>6.946E-08</td>
<td>-7.59</td>
<td>-8.10 to -7.08</td>
</tr>
<tr>
<td>Null Stereo</td>
<td>-13.922</td>
<td>11.000</td>
<td>2.493E-08</td>
<td>-8.66</td>
<td>-9.23 to -8.09</td>
</tr>
<tr>
<td>Null Mono</td>
<td>-11.480</td>
<td>11.000</td>
<td>1.832E-07</td>
<td>-7.61</td>
<td>-8.07 to -7.15</td>
</tr>
</tbody>
</table>

### Table 5-1 T-Test comparison of correlation coefficients

The correlation coefficients also varied significantly across the stereo condition within our experiment. This can be seen in Table 5-2 and more clearly Figure 5.3 as a vertical displacement in the cell mean lines. This shows an interface device such as stereo glasses can directly affect task behaviour; the evidence for this is in the significantly differing strength of correlation between the stereo conditions. There was no significant difference in correlation when different collision cues were applied.

### Correlation Repeated-measures Anova

<table>
<thead>
<tr>
<th>Effect</th>
<th>Value</th>
<th>F</th>
<th>Hypothesis df</th>
<th>Error df</th>
<th>Sig.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>STIM</strong></td>
<td>Pillai's Trace</td>
<td>.174</td>
<td>1.055&lt;sup&gt;a&lt;/sup&gt;</td>
<td>2.000</td>
<td>10.00</td>
</tr>
<tr>
<td>Wilks' Lambda</td>
<td>.826</td>
<td>1.055&lt;sup&gt;a&lt;/sup&gt;</td>
<td>2.000</td>
<td>10.00</td>
<td>.384</td>
</tr>
<tr>
<td>Hotelling's Trace</td>
<td>.211</td>
<td>1.055&lt;sup&gt;a&lt;/sup&gt;</td>
<td>2.000</td>
<td>10.00</td>
<td>.384</td>
</tr>
<tr>
<td>Roy's Largest Root</td>
<td>.211</td>
<td>1.055&lt;sup&gt;a&lt;/sup&gt;</td>
<td>2.000</td>
<td>10.00</td>
<td>.384</td>
</tr>
<tr>
<td><strong>STER</strong></td>
<td>Pillai's Trace</td>
<td>.356</td>
<td>0.068&lt;sup&gt;b&lt;/sup&gt;</td>
<td>1.000</td>
<td>11.00</td>
</tr>
<tr>
<td>Wilks' Lambda</td>
<td>.544</td>
<td>0.068&lt;sup&gt;b&lt;/sup&gt;</td>
<td>1.000</td>
<td>11.00</td>
<td>.031</td>
</tr>
<tr>
<td>Hotelling's Trace</td>
<td>.552</td>
<td>0.068&lt;sup&gt;b&lt;/sup&gt;</td>
<td>1.000</td>
<td>11.00</td>
<td>.031</td>
</tr>
<tr>
<td>Roy's Largest Root</td>
<td>.552</td>
<td>0.068&lt;sup&gt;b&lt;/sup&gt;</td>
<td>1.000</td>
<td>11.00</td>
<td>.031</td>
</tr>
<tr>
<td><strong>STIM X STER</strong></td>
<td>Pillai's Trace</td>
<td>.011</td>
<td>.053&lt;sup&gt;b&lt;/sup&gt;</td>
<td>2.000</td>
<td>10.00</td>
</tr>
<tr>
<td>Wilks' Lambda</td>
<td>.989</td>
<td>.053&lt;sup&gt;b&lt;/sup&gt;</td>
<td>2.000</td>
<td>10.00</td>
<td>.948</td>
</tr>
<tr>
<td>Hotelling's Trace</td>
<td>.011</td>
<td>.053&lt;sup&gt;b&lt;/sup&gt;</td>
<td>2.000</td>
<td>10.00</td>
<td>.948</td>
</tr>
<tr>
<td>Roy's Largest Root</td>
<td>.011</td>
<td>.053&lt;sup&gt;b&lt;/sup&gt;</td>
<td>2.000</td>
<td>10.00</td>
<td>.948</td>
</tr>
</tbody>
</table>

<sup>a</sup> Exact statistic  
<sup>b</sup> Design: Intercept  
Within Subjects Design: STIM+STER+STIM*STER

Table 5-2 Repeated measures ANOVA table for correlation
Figure 5.3 Graph of cell means for correlation across collision cue
5.2.3.2 Performance

The performance varied significantly in the collision cue condition (p<0.01) (Table 5-3 and Figure 5.4) but just failed to reach significance across the stereo condition (p=0.06). The order of performance can be explained by the notion that auditory reaction is faster than visual reaction (Wickens 1992). There were no interactions present in this statistical test.

**Performance Repeated-measures Anova**

<table>
<thead>
<tr>
<th>Effect</th>
<th>Value</th>
<th>F</th>
<th>Hypothesis df</th>
<th>Error df</th>
<th>Sig.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>STIM</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pillai's Trace</td>
<td>.635</td>
<td>8.707a</td>
<td>2.000</td>
<td>10.000</td>
<td>.006</td>
</tr>
<tr>
<td>Wilks' Lambda</td>
<td>.365</td>
<td>8.707a</td>
<td>2.000</td>
<td>10.000</td>
<td>.006</td>
</tr>
<tr>
<td>Hotelling's Trace</td>
<td>1.741</td>
<td>8.707a</td>
<td>2.000</td>
<td>10.000</td>
<td>.006</td>
</tr>
<tr>
<td>Roy's Largest Root</td>
<td>1.741</td>
<td>8.707a</td>
<td>2.000</td>
<td>10.000</td>
<td>.006</td>
</tr>
<tr>
<td><strong>STER</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pillai's Trace</td>
<td>.285</td>
<td>4.395a</td>
<td>1.000</td>
<td>11.000</td>
<td>.060</td>
</tr>
<tr>
<td>Wilks' Lambda</td>
<td>.715</td>
<td>4.395a</td>
<td>1.000</td>
<td>11.000</td>
<td>.060</td>
</tr>
<tr>
<td>Hotelling's Trace</td>
<td>.400</td>
<td>4.395a</td>
<td>1.000</td>
<td>11.000</td>
<td>.060</td>
</tr>
<tr>
<td>Roy's Largest Root</td>
<td>.400</td>
<td>4.395a</td>
<td>1.000</td>
<td>11.000</td>
<td>.060</td>
</tr>
<tr>
<td><strong>STIM * STER</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pillai's Trace</td>
<td>.248</td>
<td>1.647a</td>
<td>2.000</td>
<td>10.000</td>
<td>.241</td>
</tr>
<tr>
<td>Wilks' Lambda</td>
<td>.752</td>
<td>1.647a</td>
<td>2.000</td>
<td>10.000</td>
<td>.241</td>
</tr>
<tr>
<td>Hotelling's Trace</td>
<td>.329</td>
<td>1.647a</td>
<td>2.000</td>
<td>10.000</td>
<td>.241</td>
</tr>
<tr>
<td>Roy's Largest Root</td>
<td>.329</td>
<td>1.647a</td>
<td>2.000</td>
<td>10.000</td>
<td>.241</td>
</tr>
</tbody>
</table>

\[a. \text{Exact statistic}\]
\[b. \text{Design: Intercept}\]
\[\text{Within Subjects Design: STIM+STER+STIM*STER}\]

Table 5-3 Repeated measures ANOVA table for Performance
Figure 5.4 Graph of cell means for performance across collision cue
5.2.3.3 Error

Error also showed significance across collision cue condition (Table 5-4 and Figure 5.5). But there is an interaction present that means we cannot take this significant result at face value. Tracing through the result we see that the interaction is due to a particularly high error in the null collision stimulus cue condition. This is where the participant was not informed by the virtual environment that they had collided with a plate and had to basically guess if they had touched it or not. The significant result in the error is thus due to this interaction. Therefore this experiment showed that the lack of a collision stimulus had significant effect on the error but the modality of any stimulus that is present did not.

<table>
<thead>
<tr>
<th>Effect</th>
<th>Value</th>
<th>F</th>
<th>Hypothesis df</th>
<th>Error df</th>
<th>Sig</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>STIM</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pillai's Trace</td>
<td>.881</td>
<td>37.064a</td>
<td>2.000</td>
<td>10.000</td>
<td>.000</td>
</tr>
<tr>
<td>Wilks' Lambda</td>
<td>.119</td>
<td>37.064a</td>
<td>2.000</td>
<td>10.000</td>
<td>.000</td>
</tr>
<tr>
<td>Hotelling's Trace</td>
<td>7.413</td>
<td>37.064a</td>
<td>2.000</td>
<td>10.000</td>
<td>.000</td>
</tr>
<tr>
<td>Roy's Largest Root</td>
<td>7.413</td>
<td>37.064a</td>
<td>2.000</td>
<td>10.000</td>
<td>.000</td>
</tr>
<tr>
<td><strong>STER</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pillai's Trace</td>
<td>.124</td>
<td>1.561a</td>
<td>1.000</td>
<td>11.000</td>
<td>.237</td>
</tr>
<tr>
<td>Wilks' Lambda</td>
<td>.876</td>
<td>1.561a</td>
<td>1.000</td>
<td>11.000</td>
<td>.237</td>
</tr>
<tr>
<td>Hotelling's Trace</td>
<td>.142</td>
<td>1.561a</td>
<td>1.000</td>
<td>11.000</td>
<td>.237</td>
</tr>
<tr>
<td>Roy's Largest Root</td>
<td>.142</td>
<td>1.561a</td>
<td>1.000</td>
<td>11.000</td>
<td>.237</td>
</tr>
<tr>
<td><strong>STIM * STER</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pillai's Trace</td>
<td>.531</td>
<td>5.656a</td>
<td>2.000</td>
<td>10.000</td>
<td>.023</td>
</tr>
<tr>
<td>Wilks' Lambda</td>
<td>.469</td>
<td>5.656a</td>
<td>2.000</td>
<td>10.000</td>
<td>.023</td>
</tr>
<tr>
<td>Hotelling's Trace</td>
<td>1.131</td>
<td>5.656a</td>
<td>2.000</td>
<td>10.000</td>
<td>.023</td>
</tr>
<tr>
<td>Roy's Largest Root</td>
<td>1.131</td>
<td>5.656a</td>
<td>2.000</td>
<td>10.000</td>
<td>.023</td>
</tr>
</tbody>
</table>

a. Exact statistic
b. Design: Intercept
   Within Subjects Design: STIM+STER+STIM*STER

Table 5-4 Repeated measures ANOVA table for error rate
Behavioural Morphisms In Virtual Environments

Figure 5.5 Graph of cell means for error rate across collision cue
5.2.4 Results Discussion

All conditions showed a significant difference with the original model correlation. Since this experiment was identical in methodology to the original experiment (Fitts, 1954) this is taken to be firm proof that the virtual behaviour at a rapid-aimed movement task is different from the real-world behaviour. The performance across the cue stimulus condition varied significantly showing that design consideration such as collision cues in the absence of haptic feedback can affect human performance. Significant differences in the strength of correlation across stereo condition show that the cues presented at the hardware interface can also significantly affect behaviour.

The tapping task was identical across conditions apart from the collision stimulus and the presence stereo cues. The fact that the stereo condition had a significant effect on the behaviour and the collision stimulus had a significant effect on the error and performance proves that design decisions as trivial as including an object touching click have a direct impact on human performance and behaviour.

These results demonstrate that behavioural morphism can identify differences in virtual environment performance due to design consideration and interface design. They also demonstrate that using real-world human behaviour is an effective way of extracting hidden behavioural differences between supposedly identical tasks. The virtual and real tasks are not the same simply because they look the same and require the same movement skills.
5.3 **Implementation**

5.3.1 **Abstraction and Implementation**

After reading the literature related to rapid-aimed movement literature the original Fitts’ law experiment (Fitts 1954) was chosen as the model for procedure of the virtual environment in the rapid-aimed movement experiment. To create a virtual environment that reflected the original experiment involved modelling two ‘tapping’ plates on a table and a probe that is used to tap as quickly as possible between them.

![Figure 5.6 The Rapid-aimed movement environment layout](image)

The trial procedure was designed so that the user was in charge of the individual trial execution and indicated their readiness by starting to tap. Once a predetermined number of taps had been successfully completed the tapping plates disappeared for thirty seconds indicating to the user that the trial had completed. The functional flow
Behavioural Morphisms in Virtual Environments

is shown in Figure 5.7 below. As with the choice reaction the order or the blocks and trials was randomised to prevent order and learning effects. The method used was identical to that detailed in the previous section (Section 4.3.2.1). This reuse of the software speeded up the development process.

Figure 5.7 Fitts' Law trial abstraction
5.3.2 Logic implementation

The main constraints of the virtual environment design were the need to mimic the original experiment and the need to collect the requisite data for analysis. The top-level requirements were:

- A pair of tapping plates must be presented randomly.
- Each amplitude and width level must be equally tested.
- Each level must be tested 35 times
- Record the time to tap between the plates
- Record the width and amplitude at the start of each trial
- Pause after each trial until the participant indicates readiness
- Output the data in a format that could be easily manipulated

All of these requirements we derived from experimental requirements, for instance the requirement in bullet point 3 above was driven by the need to reduce the standard error. This was also tested using the power tests available in SPSS.

The control logic of this experiment was straightforward since the only configuration needed for each trial was the adjustment of the width and amplitude of the plates. This was provided by the following function.

5.3.2.1 Configuration Logic

The code shown below is the configuration logic for the virtual environment it is included to demonstrate the programmatical method used to configure the virtual environment.

```cpp
void adjPlates()
{
    dmPoint p;
    dmEuler euler;
    dmScale s;

    switch(epochCounter)
    {
```
case 1:
    dmPointSet (p, -0.04375, 0, 0);
dmScaleSet (s, 0.01, 0.1, 0.1);
dmEulerSetD (euler, 0, 0, 0);
VCEntity_SetPositionPointEulerScale (LPlate, p, euler, s);
dmPointSet (p, 0.04375, 0, 0);
dmScaleSet (s, 0.01, 0.1, 0.1);
dmEulerSetD (euler, 0, 0, 0);
VCEntity_SetPositionPointEulerScale (RPlate, p, euler, s);
current->epoch = epochCounter;
epochCounter = epochCounter++;
trialCounter = 0;
plateamplitude = 0.0875;
platewidth = 0.025;
break;

case 2:
    dmPointSet (p, -0.06875, 0, 0);
dmScaleSet (s, 0.01, 0.1, 0.1);
dmEulerSetD (euler, 0, 0, 0);
VCEntity_SetPositionPointEulerScale (LPlate, p, euler, s);
dmPointSet (p, 0.06875, 0, 0);
dmScaleSet (s, 0.01, 0.1, 0.1);
dmEulerSetD (euler, 0, 0, 0);
VCEntity_SetPositionPointEulerScale (RPlate, p, euler, s);
current->epoch = epochCounter;
epochCounter = epochCounter++;
trialCounter = 0;
plateamplitude = 0.1375;
platewidth = 0.025;
break;

case 3:
    dmPointSet (p, -0.09375, 0, 0);
dmScaleSet (s, 0.01, 0.1, 0.1);
dmEulerSetD (euler, 0, 0, 0);
VCEntity_SetPositionPointEulerScale (LPlate, p, euler, s);
dmPointSet (p, 0.09375, 0, 0);
dmScaleSet (s, 0.01, 0.1, 0.1);
dmEulerSetD (euler, 0, 0, 0);
Behavourial Morphisms In Virtual Environments

```c
VCEntity_SetPositionPointEulerScale (RPlate, p, euler, s);
current->epoch = epochCounter;
epochCounter = epochCounter++;
trialCounter = 0;
plateamplitude = 0.1875;
platewidth = 0.025;
break;

case 4:
dmPointSet (p, -0.14375, 0, 0);
dmScaleSet (s, 0.01, 0.1, 0.1);
dmEulerSetD (euler, 0, 0, 0);
VCEntity_SetPositionPointEulerScale (LPlate, p, euler, s);
dmPointSet (p, 0.14375, 0, 0);
dmScaleSet (s, 0.01, 0.1, 0.1);
dmEulerSetD (euler, 0, 0, 0);
VCEntity_SetPositionPointEulerScale (RPlate, p, euler, s);
current->epoch = epochCounter;
epochCounter = epochCounter++;
trialCounter = 0;
plateamplitude = 0.2875;
platewidth = 0.025;
break;

case 5:
dmPointSet (p, -0.19375, 0, 0);
dmScaleSet (s, 0.01, 0.1, 0.1);
dmEulerSetD (euler, 0, 0, 0);
VCEntity_SetPositionPointEulerScale (LPlate, p, euler, s);
dmPointSet (p, 0.19375, 0, 0);
dmScaleSet (s, 0.01, 0.1, 0.1);
dmEulerSetD (euler, 0, 0, 0);
VCEntity_SetPositionPointEulerScale (RPlate, p, euler, s);
current->epoch = epochCounter;
epochCounter = epochCounter++;
trialCounter = 0;
plateamplitude = 0.3875;
platewidth = 0.025;
break;

case 6:
dmPointSet (p, -0.071875, 0, 0);
```

Simon Nee 24/05/2002
dmScaleSet (s, 0.005, 0.1, 0.1);
dmEulerSetD (euler, 0, 0, 0);
VCEntity_SetPositionPointEulerScale (LPlate, p, euler, s);
dmPointSet (p, 0.071875, 0, 0);
dmScaleSet (s, 0.005, 0.1, 0.1);
dmEulerSetD (euler, 0, 0, 0);
VCEntity_SetPositionPointEulerScale (RPlate, p, euler, s);
current->epoch = epochCounter;
epochCounter = epochCounter++;
trialCounter = 0;
plateamplitude = 0.14375;
platewidth = 0.0125;
break;

case 7: dmPointSet (p, -0.096875, 0, 0);
  dmScaleSet (s, 0.005, 0.1, 0.1);
dmEulerSetD (euler, 0, 0, 0);
VCEntity_SetPositionPointEulerScale (LPlate, p, euler, s);
dmPointSet (p, 0.096875, 0, 0);
dmScaleSet (s, 0.005, 0.1, 0.1);
dmEulerSetD (euler, 0, 0, 0);
VCEntity_SetPositionPointEulerScale (RPlate, p, euler, s);
current->epoch = epochCounter;
epochCounter = epochCounter++;
trialCounter = 0;
plateamplitude = 0.19375;
platewidth = 0.0125;
break;

case 8: dmPointSet (p, -0.146875, 0, 0);
  dmScaleSet (s, 0.005, 0.1, 0.1);
dmEulerSetD (euler, 0, 0, 0);
VCEntity_SetPositionPointEulerScale (LPlate, p, euler, s);
dmPointSet (p, 0.146875, 0, 0);
dmScaleSet (s, 0.005, 0.1, 0.1);
dmEulerSetD (euler, 0, 0, 0);
VCEntity_SetPositionPointEulerScale (RPlate, p, euler, s);
current->epoch = epochCounter;
epochCounter = epochCounter++;
trialCounter = 0;
plateamplitude = 0.29375;
platewidth = 0.0125;
break;
case 9:
dmPointSet (p, -0.196875, 0, 0);
dmScaleSet (s, 0.005, 0.1, 0.1);
dmEulerSetD (euler, 0, 0, 0);
VCEntity_SetPositionPointEulerScale (LPlate, p, euler, s);
dmPointSet (p, 0.196875, 0, 0);
dmScaleSet (s, 0.005, 0.1, 0.1);
dmEulerSetD (euler, 0, 0, 0);
VCEntity_SetPositionPointEulerScale (RPlate, p, euler, s);
current->epoch = epochCounter;
epochCounter = epochCounter++;
trialCounter = 0;
plateamplitude = 0.39375;
platewidth = 0.0125;
break;
case 10:
dmPointSet (p, -0.296875, 0, 0);
dmScaleSet (s, 0.005, 0.1, 0.1);
dmEulerSetD (euler, 0, 0, 0);
VCEntity_SetPositionPointEulerScale (LPlate, p, euler, s);
dmPointSet (p, 0.296875, 0, 0);
dmScaleSet (s, 0.005, 0.1, 0.1);
dmEulerSetD (euler, 0, 0, 0);
VCEntity_SetPositionPointEulerScale (RPlate, p, euler, s);
current->epoch = epochCounter;
epochCounter = epochCounter++;
trialCounter = 0;
plateamplitude = 0.59375;
platewidth = 0.0125;
break;
default:
    (void) printf("Unknown option\n");
    exit(2);
This function uses the scheduled trial number (epochCounter) as an input parameter to position and scale the plates on the table. It also initialises the trial counters to start counting the trials in a block.

5.3.2.2 Data Collection Logic and Output

The function in Code Section 5.9 is one of the two data collection algorithms used for the trial data collection. It is called each time the right hand plate is touched. There is also a function to handle the data collection for the left hand tapping plate; it differs only in the name and where it is called.

```c
void rplateprocessCollisions(VCCollision_CallbackData *cd, void *
data)
{ /*
   Right plate collision callback handler
 */
    VCCollisionReportData *report;
    VAttribute *hitNoise=(VAttribute *)data;
    struct timespec starttime;

    clock_gettime(CLOCK_REALTIME,&starttime);
    report = VCCollision_GetFirstCollisionReport(cd->collision,NULL);

    if(report == NULL)
    {
        /* Restore to normal visual stimuli here as probe uncollides ( NULL report) */
        VCVisual_SetGeometry(probeVis, "geometry/ExplFitts12");
        VCVisual_SetGeometry(RPlateVis,"geometry/ExplFitts6");
    }
    else
    {
        if(report->direction[1]==-1)
        {
            /* Stimili presented here as correct ( unit vector in -Z
```
direction) collision takes place */

VCAudio_Start(hitNoise);
VCCVisual_SetGeometry(ProbeVis,"geometry/Exp1Fitts13");
VCCVisual_SetGeometry(RPlateVis,"geometry/Exp1Fitts9");

current->collision = *report;
trialCounter=trialCounter++;
current->counter=trialCounter;
current->epoch=epochCounter+1;
current->targetAmplitude = plateamplitude;
current->targetWidth = plateWidth;
current->plate=1;

current->collision.time.secs=starttime.tv_sec;
current->collision.time.uSecs=starttime.tv_nsec;
current->nextCollision=(struct collisionList
*)malloc(sizeof(struct collisionList));
current->nextCollision->previousCollision=current;
current->nextCollision->counter=current->counter;
current = current->nextCollision;

} else
{
  errors=errors+1;
}
}

if (trialCounter == MAXTRIALS && epochCounter == MAXEPOCHS-1 )
  OnMaxTrials();
if (trialCounter == MAXTRIALS && epochCounter != MAXEPOCHS)
  adjPlates();

---

**Code Section 5.9 Trial data Collection function**

The trial data collected by the above function is appended to a linked list structure and the linked list structure that stores all of the data. Once all the experimental blocks had been exhausted this linked list was then written to disk using the function in Code
Section 5.10. This function outputs an ASCII file that can be imported into Excel using the inserted delimiters to provide the data structure within Excel.

```c
void OnMaxTrials()
{ /*
   This function outputs the data in the linked list to a file
 */

   FILE *fp;
   char str[80];
   int epochvalue;
   double realtime,starttime;

   /*
   This is where the data is put into the file
   */
   current = head ;

   /*
   copy directory path and append filename from command line
   */
   strcpy(str,"data/Audio/");
   strcat(str,filename);

   if((fp = fopen(str,"w"))==NULL)
   { printf("Error opening file\n");
     exit(1);
   }

   fprintf(fp,"Epoch\tTrial\tTime\tX Position\tY Position\n");

   while ( current->nextCollision !=NULL)
   {
     realtime = current->collision.time.secs+(current->collision.time.uSecs/1000000000.0);
     fprintf(fp,"%3d\t",current->epoch);
```

Simon Nee 24/05/2002
Behavioural Morphisms In Virtual Environments

fprintf(fp,"%4.d\t",current->counter);
fprintf(fp,"%f\t",realtime);
fprintf(fp,"%f\t",current->collision.point[0]);
fprintf(fp,"%f\n",current->collision.point[2]);

if (current->counter == MAXTRIALS)
{
    fprintf(fp, "\n");
    fprintf(fp, "Plate Width\tPlate Amplitude\t(2A/W)\n");
    fprintf(fp, "%f\t%f\t%f\n",current->targetWidth,current->
targetAmplitude,((2*current->targetAmplitude)/current->
targetWidth));
    fprintf(fp, "\n");
    current= current->nextCollision;
}

fprintf(fp, "%s\n",filename);

fclose(fp);
exit(1);

Code Section 5.10 Rapid-aimed movement data output function

5.3.3 Verification and Validation

To verify and validate the virtual environment design for the rapid-aimed movement experiment a pilot study was undertaken. This involved three pilot participants undertaking the experiment and running the data produced through the data collection, data output, data manipulation and statistical test functions that had been constructed. This process ensured the following

- The experimental control was straightforward and usable
- The participant could control the trial execution
- The data output was correct
- The data was in a format that could be analysed

Simon Nee 24/05/2002
138
After the pilot process had been undertaken the results were investigated only for methodological reasons. These results were not included in the final experimental study due to the risks of comparing non-like data, if any differences occurred between the pilot study and the real study the pilot data could be considered to be suspect. No problems were encountered during the pilot
5.4 Chapter Conclusion

This experimental chapter on a Fitts tapping task has demonstrated significant differences between real-world and virtual world behaviour. Since the experiment was identical in methodology to the original experiment this is taken as proof of behavioural variations between real and virtual tasks. Since we also have evidence of collision cues and stereo cues affecting performance and behaviour we may go further and say that the virtual and real tasks are not the same. Indeed the fact that stereo cues modify behaviour suggests that a virtual stereo tapping task is different from a mono tapping task.

The experiment carried out here demonstrated that the behaviour, performance and error of real and virtual tasks can be compared and inferences drawn about the design of the virtual environment. For instance it is clear from the results of the experiment that a collision cue is vital if one wishes to reduce error and increase performance in the absence of haptic feedback.

All of these results support the thesis that behavioural morphism can utilize accepted psychological models to determine behavioural differences in open-loop serial ballistic motor skills. The experiment demonstrates the morphism measures can be applied to an open-loop serial ballistic motor skill and identify differences in behaviour that would not be have been picked up by analysing using simple single parameters such as performance or error. The fact that behaviour varies is troubling for some training applications such as training motor skills for surgery this would present real difficulties. What exactly are we training - the real task or the simulations version of the task? Behavioural morphism measures may help us address this issue by providing a quantitative analysis of the task behaviour we are training and the task we wish to train.

However, like the previous chapter this research shows that behavioural morphism requires considerable effort in the terms of experimentation, subject resources, programming, data manipulation and statistical analysis to enable a comparison.
Chapter 6: Rotary Pursuit Tracking Experiment
6.1 Introduction

This thesis uses well-established human factors/psychological models as a datum for real-world comparison. This is because the models have been tested over time by the psychological community and found to be reasonably robust. However, there are tasks where no mathematical models exist or where the models that exist do not adequately describe human behaviour for the purpose of generalization. In this case the only alternative is to collect data from the real task and compare it with the virtual task.

The experiment that is reported in this section is a study of a real-world rotary pursuit task versus virtual world rotary pursuit task. The participant's performance, error and behaviour were compared between a real pursuit task and a virtual pursuit task. These comparisons are carried out using datasets alone. The real-world dataset is used as the datum for comparison the same way as well-founded mathematical models are where they exist. This task was chosen because of it is a highly used skill that is used in many manual operations and is a fundamental sub-task in many other tasks.

6.2 Rationale

This experiment tested the hypothesis that real and virtual tasks are isomorphic in behaviour, error and performance at a rotary pursuit task. Being able to reject the null hypotheses would support one of the arguments of this thesis, namely that behavioural morphism techniques can identify differences in real and virtual behaviour where no mathematical models exists for comparison. The experiment if successful will also demonstrate the utility behavioural morphism within a closed-loop motor control task.
6.3 Experimentation

6.3.1 Methodology

The experiment conducted used a within-subjects repeated measures design (2x8 levels). This design method was chosen to allow the experiment to record individuals without the need for the larger number of participants that a between subjects design would require. The task was a rotary pursuit task the aim of which was to keep a probe tip centred on a visible light stimulus that tracked clockwise along a circular track as in Figure 6.1 below. The measure of performance for both experiments was the RMS (Root Mean Square Error) (Poulton 1974; Boff and Lincoln 1988; Wickens 1992; Zhai and Milgram 1994) between the probe tip and the stimulus square centre.

The experimental procedure was divided into two blocks of eight trials. One block was carried out using a real rotary pursuit tester as seen in Figure 6.2 below. The other block used an equivalent virtual representation of the pursuit tester presented in a stereoscopic manner through the use of LCD (Liquid Crystal Display) shutter glasses as seen in Figure 6.3 below. The assumption of equivalence was taken to be good because of the closeness of the subjective visual fidelity of the virtual models (Padmos 1992; Rinalducci 1996; Welch et al. 1996). The order of the blocks was counterbalanced across the users to alleviate, as far as possible, learning effects.

Figure 6.1 Tracking Target Rotation.
Figure 6.2 Real Trial.

Figure 6.3 Virtual Trial.
6.3.2 Experimental Virtual Environment Design

A virtual environment model of a rotary pursuit tester was designed, programmed and constructed so it would replicate the real rotary pursuit tester. The virtual model was constructed by modelling the real pursuit tester and probe using 3DStudioMax R2.0 and by measuring the real object, ± 0.5 mm. Textures were captured from the real-world using a flatbed scanner and applied to the virtual models. The finished models can be seen in Figure 6.4 and Figure 6.5 below. The models were incorporated into a real-time virtual environment using geometry format conversion tools and the Division dVS/dVISE virtual environment API.

![Figure 6.4 Stereo-pair image of 3D pursuit tester model](image1)

![Figure 6.5 Stereo-pair image of 3D probe model](image2)

Both the real and virtual blocks of the experiment relied on a Polhemus Fastrak 6DOF magnetic tracking sensor that was fixed to the back end of the probe. The function of the sensor was to report its position when requested to do so. On the first trial condition of each experiment the position of the pursuit tester surface and the offset of
the magnetic sensor from the probe tip were measured. The position of the sensor on the probe can be seen in Figure 6.9. Using the reported position and the measured tip offset the probe’s tip position could be calculated via post processing of the collected data.

6.3.3 Room Layout

Both block conditions of the experiment were carried out in the same session and in the same room, this was meant to ensure no difference could occur due to extraneous environmental conditions. However, the participants were required to change position when carrying out the second block of the experiment. This movement was required due to the logistical limitations of moving the computer and tracking system around during the experiment and was unavoidable. The set up of the area can and the differing positions between the real and virtual trials can be seen in Figure 6.6 above.
6.3.4 Apparatus and Calibration of Systems

6.3.4.1 Virtual Set-Up

For the virtual section of the experiment the apparatus consisted of a Silicon Graphics Inc Indigo2 Max Impact with a R4400 MIPS processor 64MB RAM and 4 MB TRAM. For this part of the experiment a Polhemus Fastrak magnetic tracker was used to transfer the user’s hand movement to the user’s virtual hand/probe representation. The participant had full 6DOF control over the virtual probe.

The environment was a bespoke environment programmed in C using Division’s dVS/dVISE virtual environment API (Application Programmers Interface) and the Silicon Graphics’ IRIX 6.2 operating system. The stereo condition was provided via a pair of commercially available LCD shutter glasses and by presenting frames in a field sequential manner on a Silicon Graphics Inc 21 " monitor.

The IRIX 6.2 command “setmon strıect” command was used to set up the monitor to interlace the alternate images. This alternating of images combined with the shutter glasses is what provides the participant with depth information since each eye receives a different image. The virtual environment display monitor was set up to display 1280x1024 pixel frames. The frame update was fixed at 60Hz and the tracker-sampling rate fixed at 100 Hz in the runtime dVS virtual environment. The software was programmed to trigger the collection of data using an expiring callback timer. The callback was controlled via the operating system and thus the position of the target and probe tip was sampled at a constant 30 Hz over the course of the experiment. The data collected was the virtual position of the target stimulus square and the virtual position of the probe tip. Full details about the design and implementation of the virtual environment for section of the experiment can be found in section 6.4.2

6.3.4.2 Real Set-Up

The experimental set up for the real part of this study consisted of a specially instrumented rotary pursuit tracker attached to a PC with an Intel Pentium 233 MHz II processor, 64mb RAM, Microsoft Windows 95™ OSR2, National Instruments LabView software and a National Instruments LabPC+ IO (Input/Output) card in the
ISA slot. A Polhemus Fastrak magnetic tracker was used to determine the position of the probe. For this portion of the experiment the position of the target stimulus and the position of the magnetic tracker had to be measured. From these measurements the distance between the target stimulus centre and the probe tip was calculated. The rotary pursuit tester that was used for this experiment did not give a measurement for the position of the target stimulus so to gain this measurement instrumentation was added to the system (Figure 6.7). The instruments were designed, built and fitted to the tester prior to running the experiment.

The instrumentation system consisted of an optical encoder fixed to the pursuit tester and a software program that controlled the sampling of the magnetic tracker (Figure 6.8). As the turntable turned the optical encoder triggered the software to read the magnetic tracker at 30 Hz, the software then time stamped the position data and appended the turntable position to it. This data was stored in computer RAM until the end of the experiment where it was finally written to an ASCII text file. From this ASCII data file the distance between the probe tip and target stimulus could be calculated in retrospect using a look up table and some mathematical transformations. Full details about the design and implementation of the real-world section of the experiment can be found in section 6.4.1

6.3.5 Data Collection

6.3.5.1 Virtual

The collection of the virtual block data presented little difficulty. The virtual environment had been specially programmed so the data collection algorithms and data handling formed part of the program. The position of the centre of the virtual probe and centre of the virtual target stimulus were collected through the environment API.

The position of target centre and probe tip centre positions were read within the duration of one function call of each other making the difference in time between their sampling in the order of tens of microseconds. This meant that the data samples were very tightly time-stamped together. The program stored the collected data in a dynamic link list structure in RAM, this structure was written to an ASCII disk file at
the end of the experiment. Using a RAM based data collection methodology meant that the virtual environment was not handicapped or interrupted by the operating system writing to disk. The technical details regarding this structure can be seen in 6.4.2.2.2.

6.3.5.2 Real

The collection of the real-world data presented several problems. The first problem was recording the position of the stimulus target during the experiment without being able to directly attach a position sensor to it. Recording the position of the tip of the probe during the experiment was the second problem. The final problem was synchronising and calibrating the two positions. No commercial software could provide a solution to both these problems so a special instrumentation system was built and fitted to the pursuit tester.

The instrumentation system counted the angular position of the pursuit rotor via reference and index stripe bars attached to its outer circumference as seen in Figure 6.8 below. The reference and index marks were read via an optoelectronic sensor and a computer-interfacing device (Figure 6.7). A computer system that was interfaced to the instrumentation initialised the magnetic tracker at the start of each experiment and counted the generated pulses of the indexing system. The design and development of this bespoke electronics system that was developed for this experiment is described in detail in section 6.4.1.2.3.

When the computer system received a signal that an index mark was in front of the optical sensor it triggered a reading via LabView of the position of the magnetic tracker. When the tracker returned the data it was stored in RAM along with the number of the index mark that triggered the reading. This method of collection meant that the position of the tracker was recorded within a very small latency (Approximately 4-8 milliseconds depending on the measurement cycle of the tracker.) of the angular position of the turntable. Keeping the data sampling latency this small meant that the probe sensor position and tracker position sampling coherency was very high and any the error was small and systematic. An account of the errors present in this system is featured in section 6.3.5.5 below. The data collected were then stored.
in a RAM based data structure until the end of the trial when it was written to an ASCII file. There were 180 index marks in total providing a 30 Hz sampling rate with the turntable period set at $T = 6$ seconds providing the same sampling rate as the virtual data. A full description of the software and the instrumentation used is available in section 6.4.1.2.3.

Figure 6.7 Instrumentation system
6.3.5.3 Datasets

During each virtual trial the position of the centre of the probe and the centre of the target in the virtual environment were collected at 30 Hz (X, Y, Z) via the virtual environment software. During each real trial the position and orientation (X, Y, Z, Yaw, Pitch, and Roll) of a magnetic sensor was measured at 30 Hz, as was the angular position of the rotor position. In each trial the target described the track ten times with a period of six seconds.

In all 23400 data were collected for each participant. The same probe was used for real and virtual tracking tasks and the tip offset calculated at the start of each experiment. The experimental set-up detailed above produces 2 datasets.

Virtual dataset

Target position (XTARGET, YTARGET, ZTARGET)

Probe tip position (XTIP, YTIP, ZTIP)

Real dataset

Rotor angle (ϕROTOR)
Behavioural Morphisms In Virtual Environments

Sensor position (XORIGIN, YORIGIN, ZORIGIN)
Sensor orientation (φYAW, θPITCH, φROLL)
Probe tip offset (XTIP, YTIP, YTIP)

Figure 6.9 Real-world probe dataset.
6.3.5.4 Data Processing

An acknowledged measure of performance used in this type of tracking experiment and the one used in this experiment is the root mean square error (RMS) (Poulton 1974; Boff and Lincoln 1988; Wickens 1992; Zhai and Milgram 1994) between the centre of the tracking stimulus and the centre of the tracking reticule, in this case the centre of the probe tip. Clearly, only the virtual dataset provides a direct method of using this measure. The data from the real portion of the experiment had to be post-processed to provide the Cartesian position of the probe tip and target centre.

Obtaining the tip position in real Cartesian space for the probe in the real block required calculating it from the sensor position, orientation and tip offset. This was achieved by using the following transformation matrix (Equation 6.1 Transformation matrix) for Eulerian angles (Ascension Flock of Birds User Manual).

Figure 6.10 Real-world rotor dataset.
Equation 6.1 Transformation matrix

Obtaining the position of the target stimulus for the real pursuit tester utilised look-up tables for the turntable indexes. The look-up tables were computed prior to the experiment by physically attaching a sensor to the turntable stimulus light and recording its position with respect to the index and reference marks. The reconstruction used an average of 100 rotations of the target stimulus and was achieved by curve fitting to a cosine function. The cosine function was chosen since the target stimulus described a circle and the curve fitting required calculating only the amplitude and phase arguments. The cosine function was chosen because once fitted it would accurately describe the position of the stimulus point since the stimulus point describes a circle. Once the look-up tables had been computed the magnetic transmitter and pursuit tester were fixed in position for the duration of the experiment. Any initial variance in the calibration could then be cancelled out via measuring known calibration points. The target stimulus position could then be found by inputting the rotor index into the reconstructed waveform function.
Equation 6.2 Rotor angle to position mapping
6.3.5.5 Errors

Error due to bar code

The bar code attached to the rotor is the driver for all measurements. Measurements are triggered by the leading and trailing edges of a barcode segment. The bar code was printed out at 200 dpi on a laser printer and attached to the circumference of the rotor. The barcode had approximately 2 mm of play that was taken out to balance up the error. Thus the positional error in the bar code was approximately 2/180 mm spread in 4 segments over the circumference of the rotor. Since the position of stimulus light was fixed with respect to the bar code we know for each of the 180 positions where the light is to a high accuracy via our look up table.
Error due to optoelectronic read of the bar code.
The bespoke electronics only role was to transfer a signal representing the bar code (square wave) to a PCI data acquisition board (DAQ) within the monitoring computer. The signal to the DAQ board from the bespoke electronics was analogue signal and any temporal error in it was due to the response of the optosensors and TTL within it, this is in the order of at most tens of nanoseconds.

Error due to sampling frequency
The DAQ board sampled the (square wave) waveform into two of its internally thresholded digital counters the sampling frequency was 40 kHz. This means that the position of the rotor was sampled at 40 kHz as the rotor rotates.

Error due to DAQ interrupts
On the monitoring computer was a copy of LabView which monitored the DAQ counters and ran an interrupt routine to read the tracker each time the index position digital counter incremented. The interrupt latency is in the order of microseconds or tens of microseconds.

Error due to tracker read.
The tracker was read with a latency of 4-8 ms (as per the manufacturer's specification) depending on the tracker measurement cycle.

Total Error
The only significant error was in fact the tracker latency. Running the rotor at 1/6 Hz led to tracker sampling of 30 Hz (A highly accurate 30 Hz due to underlying 40 KHz sampling and accuracy in the bar code) this meant that between samples there was 33 ms to make the measurement of the tracker so the tracker read latency was small and systematic and carried out easily within the sampling period. The tracker latency was by far the largest error and swamped the other errors described above. It is taken as the overall error for the system, assuming the participants hand is moving at roughly the same velocity as the stimulus the error in hand position is approximately ±1 mm.
6.3.5.6 Data Analysis

All the post experimental data manipulation and transformation was undertaken using specially written Microsoft Excel 2000 macros. This made processing the 280,800 data possible without extensive hand manipulation. The final statistical analyses were carried in SPSS 9.0 for Windows.

For each real-world trial the data was inserted into an Excel spreadsheet and a macro applied that calculated the RMS via Pythagoras’ Theorem. The real data RMS data were calculated using an alternative Excel macro. The macro achieved this by bringing together the calibration data, the tracking data and the look-up table. The Excel macro sheets also carried out the transformation given in Equation 6.1. The data were then treated for outliers and extreme values. The spreadsheets also calculated the standard deviation of the performance and the correlation of tip position and target position. The resultant RMS data were then placed into SPSS for the experimental comparison. Figure 6.11 shows the calibration data, the data look-up table and trial data sets for a typical trial. The common calibration point for all the datasets can be seen at approximately 7.00 pm point on the circle in Figure 6.12 over.
Example Data Set

Figure 6.12 Final Calibration Set-Up
6.3.6 Results

The null hypotheses under scrutiny for this experiment were:

- Task performance is isomorphic (RMS Error)
- Task error is isomorphic (RMS Error Variance)
- Task behaviour is isomorphic (Correlation between target and probe position)

Since no simple behaviour function exists the behaviour measure was defined as a correlation between the target and stimulus position. This is a simple behavioural model that the target position predicts the probe tip position.

The hypotheses were tested using repeated-measures ANOVA's statistical tests. These tests are typically used for repeated measures within-subjects experiments (Howell 1997). Before the statistical tests are presented and give an intuitive understanding for the datasets collected the cell means are presented for each of them below in the graphs in Figure 6.13, Figure 6.14 and Figure 6.15 below. It can be readily seen from these graphs that there is a significant difference between the real and virtual trial sets.

**Cell Marginal Means: Performance ANOVA**

![Graph showing repeated measures performance ANOVA factors cell means.]

**Figure 6.13 Repeated measures performance ANOVA factors cell means.**
Figure 6.13 shows the average performance of the participants at the task in the real and virtual environments. As can be seen from this graph there is a pronounced learning effect present in the virtual component of the experiment.

![Cell Marginal Means of Variance of RMS Error](image)

**Figure 6.14 Repeated measures error ANOVA cell means graph.**

Figure 6.14 shows the average error of the participants at the task in the real and virtual environments. Again there is a pronounced learning effect present in the virtual component of the experiment.
Figure 6.15 Repeated measures behaviour ANOVA cell means.

Figure 6.15 shows the average behaviour correlation of the participants at the task in the real and virtual environments. Note that our initial behavioural model is validated by the very high correlation coefficient of the target position with probe tip position in the real-world.
6.3.6.1 Homogeneity of Variance Assumptions

The performance measure, as mentioned earlier, is the RMS error between the probe tip centre and the centre of the stimulus target. The results of the ANOVA test on the null hypothesis are shown in Table 6-4 below. As with all ANOVA's this test is sensitive to assumptions of homogeneity of variance (Coolican 1996; Howell 1997), this assumption was tested using SPSS's Mauchly's assumption of sphericity test (Table 6-1).

Mauchly's Test of Sphericity

<table>
<thead>
<tr>
<th>Within Subjects Effect</th>
<th>Mauchly's W</th>
<th>Approx. Chi-Square</th>
<th>df</th>
<th>Sig.</th>
<th>Epsilon^a</th>
</tr>
</thead>
<tbody>
<tr>
<td>METHOD</td>
<td>1.000</td>
<td>.000</td>
<td>9</td>
<td>.000</td>
<td>1.000</td>
</tr>
<tr>
<td>TRIAL</td>
<td>.003</td>
<td>48.667</td>
<td>27</td>
<td>.010</td>
<td>3.12</td>
</tr>
<tr>
<td>METHOD * TRIAL</td>
<td>.000</td>
<td>81.190</td>
<td>27</td>
<td>.000</td>
<td>.319</td>
</tr>
</tbody>
</table>

Tests the null hypothesis that the error covariance matrix of the orthonormalized transformed dependent variables is proportional to an identity matrix.

a. May be used to adjust the degrees of freedom for the averaged tests of significance. Corrected tests are displayed in the Tests of Within-Subjects Effects table.

b. Design: Intercept
   Within Subjects Design: METHOD*TRIAL

Table 6-1 Assumption of sphericity test, performance.

The table above shows that the assumption of sphericity for our performance data is violated, so a correction factor (Greenhouse-Geisser correction factor) must be applied to the F statistic to ensure the test's validity. The sphericity assumptions were also broken on the other two ANOVA tests, these are shown in Table 6-2 and Table 6-3 close. As a result of this all ANOVA tests used the Greenhouse-Geisser correction factor to ensure that the integrity of the significance in the tests is retained. The Greenhouse-Geisser correction factor is the most conservative factor with respect to significance.
**Behavioural Morphisms In Virtual Environments**

**Mauchly's Test of Sphericity**

<table>
<thead>
<tr>
<th>Within Subjects Effect</th>
<th>Mauchly's W</th>
<th>Approx. Chi-Square</th>
<th>df</th>
<th>Sig.</th>
<th>Greenhouse-Geisser</th>
<th>Huynh-Feldt</th>
<th>Lower-bound</th>
</tr>
</thead>
<tbody>
<tr>
<td>METHOD</td>
<td>1.000</td>
<td>0.00</td>
<td>0</td>
<td></td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
</tr>
<tr>
<td>TRIAL</td>
<td>0.031</td>
<td>29.366</td>
<td>27</td>
<td>.000</td>
<td>0.532</td>
<td>0.638</td>
<td>0.143</td>
</tr>
<tr>
<td>METHOD * TRIAL</td>
<td>0.000</td>
<td>76.516</td>
<td>27</td>
<td>.000</td>
<td>0.339</td>
<td>0.439</td>
<td>0.143</td>
</tr>
</tbody>
</table>

Tests the null hypothesis that the error covariance matrix of the orthonormalized transformed dependent variables is proportional to an identity matrix.

a. May be used to adjust the degrees of freedom for the averaged tests of significance. Corrected tests are displayed in the Tests of Within-Subjects Effects table.

b. Design: Intercept
   Within Subjects Design: METHOD+TRIAL+METHOD*TRIAL

**Table 6-2 Assumption of sphericity test, error**

<table>
<thead>
<tr>
<th>Within Subjects Effect</th>
<th>Mauchly's W</th>
<th>Approx. Chi-Square</th>
<th>df</th>
<th>Sig.</th>
<th>Greenhouse-Geisser</th>
<th>Huynh-Feldt</th>
<th>Lower-bound</th>
</tr>
</thead>
<tbody>
<tr>
<td>METHOD</td>
<td>1.000</td>
<td>0.00</td>
<td>0</td>
<td></td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
</tr>
<tr>
<td>TRIAL</td>
<td>0.000</td>
<td>151.119</td>
<td>27</td>
<td>.000</td>
<td>0.172</td>
<td>0.161</td>
<td>0.143</td>
</tr>
<tr>
<td>METHOD * TRIAL</td>
<td>0.000</td>
<td>159.534</td>
<td>27</td>
<td>.000</td>
<td>0.172</td>
<td>0.161</td>
<td>0.143</td>
</tr>
</tbody>
</table>

Tests the null hypothesis that the error covariance matrix of the orthonormalized transformed dependent variables is proportional to an identity matrix.

a. May be used to adjust the degrees of freedom for the averaged tests of significance. Corrected tests are displayed in the Tests of Within-Subjects Effects table.

b. Design: Intercept
   Within Subjects Design: METHOD+TRIAL+METHOD*TRIAL

**Table 6-3 Assumption of sphericity test, correlation**
6.3.6.2 Repeated-Measures ANOVA Tables

6.3.6.2.1 Performance ANOVA

The performance ANOVA in Table 6-4 below shows highly significant main effects in both F(1,11), p < 0.01) method and trial number F(2,1,24,1), p <0.01. A highly significant interaction between method and trial F(1.9,20.5) p< 0.01 was also present. This ANOVA is a statistical test of the null hypothesis that the performance is the same in real and virtual tasks.

Tests of Within-Subjects Effects

<table>
<thead>
<tr>
<th>Source</th>
<th>Type III Sum of Squares</th>
<th>df</th>
<th>Mean Square</th>
<th>F</th>
<th>Sig.</th>
</tr>
</thead>
<tbody>
<tr>
<td>METHOD</td>
<td>Sphericity Assumed</td>
<td>14.837</td>
<td>1</td>
<td>14.837</td>
<td>164.055</td>
</tr>
<tr>
<td></td>
<td>Greenhouse-Geisser</td>
<td>14.837</td>
<td>1</td>
<td>14.837</td>
<td>164.055</td>
</tr>
<tr>
<td></td>
<td>Huynh-Feldt</td>
<td>14.837</td>
<td>1</td>
<td>14.837</td>
<td>164.055</td>
</tr>
<tr>
<td></td>
<td>Lower-bound</td>
<td>14.837</td>
<td>1</td>
<td>14.837</td>
<td>164.055</td>
</tr>
<tr>
<td>Error(METHOD)</td>
<td>Sphericity Assumed</td>
<td>.995</td>
<td>11</td>
<td>9.044E-02</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Greenhouse-Geisser</td>
<td>.995</td>
<td>11</td>
<td>9.044E-02</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Huynh-Feldt</td>
<td>.995</td>
<td>11</td>
<td>9.044E-02</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Lower-bound</td>
<td>.995</td>
<td>11</td>
<td>9.044E-02</td>
<td></td>
</tr>
<tr>
<td>TRIAL</td>
<td>Sphericity Assumed</td>
<td>2.232</td>
<td>7</td>
<td>.319</td>
<td>24.801</td>
</tr>
<tr>
<td></td>
<td>Greenhouse-Geisser</td>
<td>2.232</td>
<td>2.184</td>
<td>1.022</td>
<td>24.801</td>
</tr>
<tr>
<td></td>
<td>Huynh-Feldt</td>
<td>2.232</td>
<td>2.746</td>
<td>.813</td>
<td>24.801</td>
</tr>
<tr>
<td></td>
<td>Lower-bound</td>
<td>2.232</td>
<td>1</td>
<td>2.232</td>
<td>24.801</td>
</tr>
<tr>
<td>Error(TRIAL)</td>
<td>Sphericity Assumed</td>
<td>.990</td>
<td>77</td>
<td>1.286E-02</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Greenhouse-Geisser</td>
<td>.990</td>
<td>24.027</td>
<td>4.120E-02</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Huynh-Feldt</td>
<td>.990</td>
<td>30.210</td>
<td>3.277E-02</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Lower-bound</td>
<td>.990</td>
<td>11</td>
<td>9.000E-02</td>
<td></td>
</tr>
<tr>
<td>METHOD * TRIAL</td>
<td>Sphericity Assumed</td>
<td>1.310</td>
<td>7</td>
<td>.187</td>
<td>11.581</td>
</tr>
<tr>
<td></td>
<td>Greenhouse-Geisser</td>
<td>1.310</td>
<td>1.864</td>
<td>.703</td>
<td>11.581</td>
</tr>
<tr>
<td></td>
<td>Huynh-Feldt</td>
<td>1.310</td>
<td>2.230</td>
<td>.587</td>
<td>11.581</td>
</tr>
<tr>
<td></td>
<td>Lower-bound</td>
<td>1.310</td>
<td>1</td>
<td>1.310</td>
<td>11.581</td>
</tr>
<tr>
<td>Error(METHOD*TRIAL)</td>
<td>Sphericity Assumed</td>
<td>1.244</td>
<td>77</td>
<td>1.616E-02</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Greenhouse-Geisser</td>
<td>1.244</td>
<td>20.506</td>
<td>6.067E-02</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Huynh-Feldt</td>
<td>1.244</td>
<td>24.527</td>
<td>5.072E-02</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Lower-bound</td>
<td>1.244</td>
<td>11</td>
<td>1.113</td>
<td></td>
</tr>
</tbody>
</table>

Table 6-4 Repeated-measures ANOVA table, performance

This table demonstrates that the behaviour between real and virtual varied significantly. The interaction is due to the presence of a learning effect in the virtual condition and not in the real condition.
6.3.6.2.2 Error ANOVA

The error ANOVA in Table 6-5 below shows highly significant main effects in both F(1,11), p < 0.01 method and trial number F(3.7,40.9), p <0.01 . As in the performance ANOVA a highly significant interaction between method and trial F(2.4,26.1) p< 0.01 was also present. This ANOVA is a statistical test of the null hypothesis that the error is the same in real and virtual tasks.

Tests of Within-Subjects Effects

<table>
<thead>
<tr>
<th>Measure: MEASURE_1</th>
<th>Source</th>
<th>Type III Sum of Squares</th>
<th>df</th>
<th>Mean Square</th>
<th>F</th>
<th>Sig.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>METHOD</strong></td>
<td>Sphericity Assumed</td>
<td>7.463</td>
<td>1</td>
<td>7.463</td>
<td>180.573</td>
<td>.000</td>
</tr>
<tr>
<td></td>
<td>Greenhouse-Geisser</td>
<td>7.463</td>
<td>1</td>
<td>7.463</td>
<td>180.573</td>
<td>.000</td>
</tr>
<tr>
<td></td>
<td>Huynh-Feldt</td>
<td>7.463</td>
<td>1</td>
<td>7.463</td>
<td>180.573</td>
<td>.000</td>
</tr>
<tr>
<td></td>
<td>Lower-bound</td>
<td>7.463</td>
<td>1</td>
<td>7.463</td>
<td>180.573</td>
<td>.000</td>
</tr>
<tr>
<td><strong>Error(METHOD)</strong></td>
<td>Sphericity Assumed</td>
<td>.455</td>
<td>11</td>
<td>.041</td>
<td>180.573</td>
<td>.000</td>
</tr>
<tr>
<td></td>
<td>Greenhouse-Geisser</td>
<td>.455</td>
<td>11</td>
<td>.041</td>
<td>180.573</td>
<td>.000</td>
</tr>
<tr>
<td></td>
<td>Huynh-Feldt</td>
<td>.455</td>
<td>11</td>
<td>.041</td>
<td>180.573</td>
<td>.000</td>
</tr>
<tr>
<td></td>
<td>Lower-bound</td>
<td>.455</td>
<td>11</td>
<td>.041</td>
<td>180.573</td>
<td>.000</td>
</tr>
<tr>
<td><strong>TRIAL</strong></td>
<td>Sphericity Assumed</td>
<td>1.236</td>
<td>7</td>
<td>.177</td>
<td>36.080</td>
<td>.000</td>
</tr>
<tr>
<td></td>
<td>Greenhouse-Geisser</td>
<td>1.236</td>
<td>3.724</td>
<td>.332</td>
<td>36.080</td>
<td>.000</td>
</tr>
<tr>
<td></td>
<td>Huynh-Feldt</td>
<td>1.236</td>
<td>5.888</td>
<td>2.211</td>
<td>36.080</td>
<td>.000</td>
</tr>
<tr>
<td></td>
<td>Lower-bound</td>
<td>1.236</td>
<td>1</td>
<td>1.236</td>
<td>36.080</td>
<td>.000</td>
</tr>
<tr>
<td><strong>Error(TRIAL)</strong></td>
<td>Sphericity Assumed</td>
<td>.377</td>
<td>77</td>
<td>.005</td>
<td>36.080</td>
<td>.000</td>
</tr>
<tr>
<td></td>
<td>Greenhouse-Geisser</td>
<td>.377</td>
<td>40.967</td>
<td>.009</td>
<td>36.080</td>
<td>.000</td>
</tr>
<tr>
<td></td>
<td>Huynh-Feldt</td>
<td>.377</td>
<td>64.543</td>
<td>.006</td>
<td>36.080</td>
<td>.000</td>
</tr>
<tr>
<td></td>
<td>Lower-bound</td>
<td>.377</td>
<td>11</td>
<td>.034</td>
<td>36.080</td>
<td>.000</td>
</tr>
<tr>
<td><strong>METHOD * TRIAL</strong></td>
<td>Sphericity Assumed</td>
<td>7.32</td>
<td>7</td>
<td>.105</td>
<td>16.398</td>
<td>.000</td>
</tr>
<tr>
<td></td>
<td>Greenhouse-Geisser</td>
<td>7.32</td>
<td>3.723</td>
<td>.308</td>
<td>16.398</td>
<td>.000</td>
</tr>
<tr>
<td></td>
<td>Huynh-Feldt</td>
<td>7.32</td>
<td>3.070</td>
<td>.238</td>
<td>16.398</td>
<td>.000</td>
</tr>
<tr>
<td></td>
<td>Lower-bound</td>
<td>7.32</td>
<td>1</td>
<td>.732</td>
<td>16.398</td>
<td>.002</td>
</tr>
<tr>
<td><strong>Error(METHOD*TRIAL)</strong></td>
<td>Sphericity Assumed</td>
<td>4.91</td>
<td>77</td>
<td>.006</td>
<td>16.398</td>
<td>.000</td>
</tr>
<tr>
<td></td>
<td>Greenhouse-Geisser</td>
<td>4.91</td>
<td>26.107</td>
<td>.019</td>
<td>16.398</td>
<td>.000</td>
</tr>
<tr>
<td></td>
<td>Huynh-Feldt</td>
<td>4.91</td>
<td>33.765</td>
<td>.015</td>
<td>16.398</td>
<td>.000</td>
</tr>
<tr>
<td></td>
<td>Lower-bound</td>
<td>4.91</td>
<td>11</td>
<td>.045</td>
<td>16.398</td>
<td>.000</td>
</tr>
</tbody>
</table>

Table 6-5 Repeated-measures ANOVA table, error

This table shows that the error was significantly different between the real and virtual conditions and that as with performance the virtual task error had a learning effect that produced the significant interaction.

Simon Nee 24/05/2002
6.3.6.2.3 Correlation ANOVA

The error ANOVA in Table 6-5 below shows a highly significant main effect for method $F(1,11), \ p < 0.01$ and a significant main effect for trial number $F(1.2,13.2), \ p < 0.05$. As in the performance and error ANOVAs a significant interaction between method and trial $F(1.2,13.2), \ p < 0.05$ was also present. This ANOVA is a statistical test of the null hypothesis that the correlations are the same in the real and virtual tasks.

**Tests of Within-Subjects Effects**

<table>
<thead>
<tr>
<th>Measure: MEASURE_1</th>
<th>Type III Sum of Squares</th>
<th>df</th>
<th>Mean Square</th>
<th>F</th>
<th>Sig</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>METHOD</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sphericity Assumed</td>
<td>3.324E-03</td>
<td>1</td>
<td>3.324E-03</td>
<td>99.686</td>
<td>.000</td>
</tr>
<tr>
<td>Greenhouse-Geisser</td>
<td>3.324E-03</td>
<td>1</td>
<td>3.324E-03</td>
<td>99.686</td>
<td>.000</td>
</tr>
<tr>
<td>Huynh-Feldt</td>
<td>3.324E-03</td>
<td>1</td>
<td>3.324E-03</td>
<td>99.686</td>
<td>.000</td>
</tr>
<tr>
<td>Lower-bound</td>
<td>3.324E-03</td>
<td>1</td>
<td>3.324E-03</td>
<td>99.686</td>
<td>.000</td>
</tr>
<tr>
<td><strong>Error(METHOD)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Huynh-Feldt</td>
<td>3.668E-04</td>
<td>11</td>
<td>3.334E-05</td>
<td>8.059</td>
<td>.011</td>
</tr>
<tr>
<td><strong>TRIAL</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sphericity Assumed</td>
<td>1.187E-03</td>
<td>7</td>
<td>1.696E-04</td>
<td>8.059</td>
<td>.013</td>
</tr>
<tr>
<td>Greenhouse-Geisser</td>
<td>1.187E-03</td>
<td>7</td>
<td>1.696E-04</td>
<td>8.059</td>
<td>.013</td>
</tr>
<tr>
<td>Huynh-Feldt</td>
<td>1.187E-03</td>
<td>7</td>
<td>1.696E-04</td>
<td>8.059</td>
<td>.013</td>
</tr>
<tr>
<td>Lower-bound</td>
<td>1.187E-03</td>
<td>7</td>
<td>1.696E-04</td>
<td>8.059</td>
<td>.013</td>
</tr>
<tr>
<td><strong>Error(TRIAL)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sphericity Assumed</td>
<td>1.621E-03</td>
<td>77</td>
<td>2.105E-05</td>
<td>7.045</td>
<td>.000</td>
</tr>
<tr>
<td>Greenhouse-Geisser</td>
<td>1.621E-03</td>
<td>77</td>
<td>2.105E-05</td>
<td>7.045</td>
<td>.000</td>
</tr>
<tr>
<td>Huynh-Feldt</td>
<td>1.621E-03</td>
<td>77</td>
<td>2.105E-05</td>
<td>7.045</td>
<td>.000</td>
</tr>
<tr>
<td>Lower-bound</td>
<td>1.621E-03</td>
<td>77</td>
<td>2.105E-05</td>
<td>7.045</td>
<td>.000</td>
</tr>
<tr>
<td><strong>METHOD * TRIAL</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sphericity Assumed</td>
<td>1.052E-03</td>
<td>7</td>
<td>1.503E-04</td>
<td>7.045</td>
<td>.013</td>
</tr>
<tr>
<td>Greenhouse-Geisser</td>
<td>1.052E-03</td>
<td>7</td>
<td>1.503E-04</td>
<td>7.045</td>
<td>.013</td>
</tr>
<tr>
<td>Huynh-Feldt</td>
<td>1.052E-03</td>
<td>7</td>
<td>1.503E-04</td>
<td>7.045</td>
<td>.013</td>
</tr>
<tr>
<td>Lower-bound</td>
<td>1.052E-03</td>
<td>7</td>
<td>1.503E-04</td>
<td>7.045</td>
<td>.013</td>
</tr>
<tr>
<td><strong>Error(METHOD*TRIAL)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sphericity Assumed</td>
<td>1.643E-03</td>
<td>77</td>
<td>2.134E-05</td>
<td>7.045</td>
<td>.013</td>
</tr>
<tr>
<td>Greenhouse-Geisser</td>
<td>1.643E-03</td>
<td>77</td>
<td>2.134E-05</td>
<td>7.045</td>
<td>.013</td>
</tr>
<tr>
<td>Huynh-Feldt</td>
<td>1.643E-03</td>
<td>77</td>
<td>2.134E-05</td>
<td>7.045</td>
<td>.013</td>
</tr>
<tr>
<td>Lower-bound</td>
<td>1.643E-03</td>
<td>77</td>
<td>2.134E-05</td>
<td>7.045</td>
<td>.013</td>
</tr>
</tbody>
</table>

**Table 6-6 Repeated-measures ANOVA, correlation**

This table shows that the correlation between stimulus position and hand position was significantly different between the real and virtual conditions and that as with performance the correlation was subject to a learning effect that produced the significant interaction.

Simon Nee 24/05/2002 167
6.3.7 Results Discussion

It can be seen from the graphs above, Table 6-4, Table 6-5 and Table 6-6 above that there are significant main effects of method and trial in the measures of behaviour, error and performance, there are also significant interactions in method and trial in all conditions. These main effects therefore cannot simply be taken at face value because a significant interaction is present that may distort, conceal or exaggerate them (Coolican 1996; Howell 1997). We must first identify the source of the interaction before making any assumption about the main effects.

All the interactions shown in the tables can be traced to, and explained by, the presence of a pronounced learning effect in the virtual pursuit task that is not present in the real task. This interaction effect is clearly visible in the factor plots Figure 6.13, Figure 6.14 and Figure 6.15. The interaction shows in greater relief that learning is present in the virtual condition and not the real condition and therefore the tasks are unlikely to be the same task despite the obvious similarity in motor activity.

The learning effect is the source of the interaction effect and can safely be ignored when interpreting the main effect of the method of training i.e. real or virtual. However it does show that the main effect for trial cannot be accepted for the real-world situation since clearly all the significance in this measure is due to the learning effects in the virtual level.

In short, there are significant differences in all the measures between real and virtual situations $p<0.01$, F(1,11). From this result all three null hypotheses can be rejected, none of these tasks are isomorphic. From this and the fact that an interaction is present due to learning effects we can infer that the tasks are not the same although it is easy to assume they are due to the similarity of the motor skill undertaken.
6.4 Implementation

6.4.1 Real Task

6.4.1.1 Design Abstraction and Implementation

The real-world rotary pursuit was the driving design for the experiment. The device available for experimental construction was originally used to demonstrate motor skill learning. Normally the device would be used over several trials and a read-out of time on target performance recorded, this data could then be analysed by the students to identify a learning effect. However, for the purposes of the rotary pursuit experiment the ‘time on target’ measure it recorded itself was not the ideal performance measurement parameter for this experiment. The preferred performance measure was the RMS error between probe tip and the target stimulus. Using the RMS measure meant that both probe tip and stimulus target positions had to captured somehow, obtaining this data was non-trivial and required the construction of a bespoke data capture system. The trial abstraction is shown in Figure 6.17. The construction of the bespoke instrumentation system is discussed in the next section.

Figure 6.16 A real environment rotary pursuit trial
Figure 6.17 Real rotary pursuit trial abstraction
6.4.1.2 Logic Implementation

The trial logic in this phase of the pursuit tracking experiment was controlled via the experimental supervisor. The supervisor's role was scripted and entailed ensuring the following.

- The data collection software was activated
- The trials were started
- Enforce a minutes rest between trials
- The trials were stopped
- The data collection software was stopped and the data saved

6.4.1.2.1 Configuration Logic

The environment was static once set-up. The rotor was fixed to rotate at 1/6 Hz and to describe the track 10 times leading to each trial lasting 60 seconds. The only variable that changed within this experiment was the trial number or subject exposure to the rotary pursuit tracking task.

6.4.1.2.2 Real Pursuit Tracking Data Collection Logic and Output

The data collection and output was carried out by the LabView program that was constructed to control the instrumentation system and the functionality can be seen in Figure 6.25.

6.4.1.2.3 User Tracking Implementation

After an initial period of reflection upon the problem of data collection a design was struck upon that would solve the problem. The design (Figure 6.18 and Figure 6.19) used reference and index pulses generated by two infrared optoelectronic sensors (Figure 6.20) reading a 'bar code' arrangement attached to the rotor (Figure 6.21) to trigger the reading of the position of a sensor attached to the tracking probe (Figure 6.23 and Figure 6.22). The pulses were 'read' by a National Instrument PC1200 IO card that was programmed via LabView v5.0 to count the pulses and trigger a reading of the tracking system. The software also outputted the data in a form that was easily importable in Microsoft Excel. This data could then be post-processed to get the position of the tracker and the target stimulus to a high precision. The Polhemus Fastrak user manual quotes an accuracy of ±0.08cm spatially and ±4ms temporally. The LabView programs are shown in Figure 6.25
Figure 6.18 Rotary Pursuit data collection system
Figure 6.19 Data collection layout

Figure 6.20 Optoelectronic sensor positioning
Behavioural Morphisms In Virtual Environments

Figure 6.21 Laser printer 'Bar Code'

Figure 6.22 Probe and tracker sensor assembly

Laser written index marks applied to pursuit rotor

Polhemus Fasttrack Tracking Sensor

Simon Nee 24/05/2002
Figure 6.23 Polhemus Fastrak sensor assemblies and fixing

Figure 6.24 PC Interface circuit
6.4.1.3 Verification and Validation

As with all the other experiments the set-up was testing by running pilot subjects through the experiments. This process ensured the following. No problems were encountered during the pilot phase:

- The experimental control was straightforward and usable
- The supervisor was able to control the experiment easily
- The participant could complete the task
- The data output was correct
- The data was in a format that could be analysed
6.4.2 Virtual Task

6.4.2.1 Abstraction and Implementation

The main aim of the virtual phase of the rotary pursuit experiment was to mimic the trials carried out in the real phase. To do this the virtual environment used virtual models derived from the real-world equipment. Since the user was controlling a virtual version of the probe and the users movement were being collected via a 6DOF tracker the collection of the RMS information about virtual probe tip position and stimulus target position could be easily obtained from the virtual environment software.

Figure 6.26 Virtual environment trial set up

The trial abstraction for this phase of the rotary pursuit tracking experiment is shown in Figure 6.27. As with the other virtual environment experiments the supervisor started the experiment and the user was in control of the trial progressions.
Figure 6.27 Virtual pursuit tracking abstraction
6.4.2.2 Logic Implementation

The pursuit tracking experimental virtual environment task for the final experiment was driven by the following requirements.

- Start tracking the stimulus when the participant indicates readiness
- Collect RMS data at the same rate as the real-world experiment
- Stop tracking after 10 rotations

The trial procedure was different from the preceding experiments since the participant would undertake only two blocks, one real one virtual. The counterbalancing was across the block level, and this was easily achieved by alternately starting subjects on the virtual or real blocks. Of the twelve subjects six started on the real blocks and six started on the virtual blocks.

6.4.2.2.1 Environment Configuration Logic

Since this experiment was looking at learning effects the environment was constant across and required no configuration during the experiment. The environment was constructed once only and the bespoke software merely collected data during the trials. The only work carried out by the software during the trial was that of describing the track with the stimulus as shown in Code Section 6.28 below.

```c
void processTimer(VCTimer_CallbackData *callbackData, void *data)
{
    VCEntity_RotateY(target,-0.0418);
    VCEntity_RotateZ(target,-0.0418);
}
```

**Code Section 6.28 Stimulus tracking function**

6.4.2.2.2 Data Collection Logic and Output

The data was collected during the trials using a callback timer. The function that collected the data is shown in Code Section 6.29 below. The data output function is shown in Code Section 6.30.
void dataCollectorTimer(VCTimer_CallbackData *callbackData, void *data)
{
    dmPoint    targetPoint, tipPoint;
    dmMatrix  targetPos, tipPos;

    VCEntity_GetAbsolutePosition(target, targetPos);
    VCEntity_GetAbsolutePosition(tip, tipPos);

    dmPointFromMat(targetPoint, targetPos);
    dmPointFromMat(tipPoint, tipPos);

    current->targetX = targetPoint[0];
    current->targetY = targetPoint[1];
    current->targetZ = targetPoint[2];

    current->tipX = tipPoint[0];
    current->tipY = tipPoint[1];
    current->tipZ = tipPoint[2];

    current->nextCollision = (struct collisionList *)malloc(sizeof(struct collisionList));
    current->nextCollision->previousCollision = current;
    current = current->nextCollision;
}

Code Section 6.29 Virtual pursuit tracking data collection function
void TrialEnd()
{
  /*
  This function outputs the data in the linked list to a file
  */

  FILE *fp;
  char str[80];
  int epochvalue;
  double secstart, secstop, nanostart, nanostop;

  current = head;
  /*Calculate durations involves transforming from long ints to double floats*/

  /*
  This is where the data is put into the file
  copy directory path and append filename from command line
  */
  printf("Top Of Write Out\n");

  strcpy(str,"data/Rotary/");
  strcat(str,filename);

  if((fp = fopen(str,"w")) == NULL)
  {
    printf("Error opening file\n");
    exit(1);
  }
Behavioural Morphisms In Virtual Environments

```c
fprint(fp,"Target X Y Z, Tip X Y Z\n");

while (current->nextCollision !=NULL)
{
    fprintf(fp,"%f,%f,%f," ,current->targetX,current->targetY,current->targetZ);
    fprintf(fp,"%f,%f,%f,\n",current->tipX,current->tipY,current->tipZ);
    current=current->nextCollision;
}

fprintf(fp, "%s\n", filename);
fclose(fp);
```

Code Section 6.30 Virtual pursuit tracking data output function

**6.4.2.3 Verification and Validation**

To verify and validate the virtual environment design for the virtual rotary pursuit movement experiment a pilot study was undertaken. This involved three pilot participants undertaking the experiment and running the data produced through the data collection, data output, data manipulation and statistical test functions that had been constructed. This process ensured the following

- The experimental control was straightforward and usable
- The participant could control the trial execution
- The data output was correct
- The data was in a format that could be analysed

After the pilot process had been undertaken the results were investigated only for methodological reasons. These results were not included in the final experimental study. No problems were encountered during the pilot.
6.5 Chapter Conclusion

The aim of this experiment was to test the hypothesis that behavioural morphism measures could be used to identify behavioural, performance and error differences between real and virtual tasks where no mathematical model exists or is usable in the context. The results show it can and does provide a method to do this.

This experiment showed very effectively that between the real and virtual task the error varied significantly \((F(1,11) = 180.6, \ p < 0.01)\), the behaviour varied significantly \((F(1,11)=99.7, \ p < 0.01)\) and the performance varied significantly \((F(1,11)=164.1, \ p < 0.01)\). These results support the thesis that behavioural morphism measures can isolate differences in behaviour that may remain hidden within conventional measures of task performance and error, and allow human behaviour to be used as a datum for virtual-real and virtual-virtual comparisons.

This experiment demonstrates differing behaviour between a real-world task and a virtual facsimile of that real-world task. Virtual environments are presently being proposed as training systems for many different motor skill tasks by many commentators. The behavioural differences shown in this experiment would not be apparent had the experiment not checked for them, again this calls into question the validity of using virtual environments to train motor skills without first validating the simulation in terms of the task to be trained.

It is easy to argue that if we used the virtual environment in this experiment to train rotary pursuit testing we would indeed be teaching trainees a virtual environment rotary pursuit task and not a real-world rotary pursuit task. The results in this chapter also support the thesis in arguing that behavioural morphism measures can be used to compare systems for varying behaviour when no mathematical model exists or is easily used and again support the thesis that behaviour is a usable performance datum.

To carry out the rotary pursuit experiment a virtual environment had to be constructed and programs written to collect data from it. Special software, bespoke instrumentation and hardware had to be used to obtain the same data from the real-world. Both these data sets had then to be post processed to obtain the results needed.
to allow a comparison of real and virtual tasks. This demonstrates again, that while behavioural morphism measures deliver a valuable insight into normally unseen behavioural changes the work needed to investigate involves considerable investment in design, development, implementation and testing. Whether researchers would want to undertake this method each time they wish to compare results or validate a training task is open for discussion.
Chapter 7: Data Review
7.1 Introduction

This chapter summarizes and discusses the work reported thus far in this thesis. It presents a review of the both the literary material covered and original experiments undertaken. After the review of the material covered so far has been covered this chapter will then identify the thesis support, experimental and evidential, for the research hypothesis presented in Section 3.3. It will individually discuss each hypothesis and identify the most salient parts of the research carried out.

7.2 Thesis Review

In chapters one and two of this thesis the aims of this research were derived. These aims were driven by the identification, through literature review and platform complexity arguments, that no method exists for measuring the fidelity of tasks within virtual environments and that there was a clear consensus amongst researchers that such a method was needed. From these aims a problem statement was formed to enable a focus of attention onto the most salient aspects of the topic.

"How closely does our virtual task resemble the real-world task?"

This research driver was derived from evidence gathered in a general review of research into virtual environment construction, human computer interaction, other related fields, and a detailed study of the literature on the measurement of human performance within virtual environments. The general and specific reviews demonstrated that no method exists, or has been proposed, that can be used to quantitatively compare real and virtual tasks for 'fidelity'. The literature review showed that a large consensus exits amongst researchers and that no such methodology has, as of the preparation of this thesis, been proposed. The literature review also demonstrated that phenomena such as visual adaptation and kinaesthetic adaptation have been shown to affect the behaviour of participants in experiments post virtual environment exposure. These adaptation effects were argued to be confounding factors in motor skill training taking into consideration that modern virtual environments are being used and being proposed as of use in the training of
perceptual motor skills. Chapter 2 proposed that the variation of behaviour of a human at these skills in a virtual environment needs to be measured and its effects investigated before any training transfer can be attributed to the virtual environment. One can see that some critical motor skills such as those of pilots or surgeons may be profoundly affected until adaptation effects have been dissipated. If these effects are left unidentified one cannot attribute training transfer to the training system. Whilst procedurally the virtual task may appear the same as the real-world in behavioural terms the task may be different.

Chapter 3 presented the ideas behind behavioural morphism. Behavioural morphism was proposed by the author as a solution to the problem statement and as the measure needed for assessing real-virtual world task fidelities. The methodology proposes using three components, behavioural correlation, error and performance to measure the similarities between real and virtual tasks. Chapter 3 also proposed that the following research hypotheses needed to be tested to provide support or dispute the two general hypotheses that behavioural morphism could and should be used to compare real and virtual tasks.

1. Human behaviour varies between real and virtual tasks.

2. Behavioural morphism is sensitive enough to compare the fidelity of real and virtual tasks
   (a) Behavioural morphism is sensitive enough to identify different behaviour between the real and virtual tasks
   (b) Behavioural morphism is sensitive enough to measure the fidelity of a task within a virtual environment where a mathematical model exists.
   (c) Behavioural morphism is sensitive enough to measure the fidelity of a task within a virtual environment where no mathematical model exists.

Chapters 4, 5, and 6 detailed the experiments carried out to test these hypotheses. The experiments carried out in this thesis used a choice-reaction task, a rapid-aimed movement task and a rotary pursuit task to investigate various aspects of using behavioural morphism. The implementation of each experiment was explained in detail including the modelling of the virtual environments, the programming of the
experimental software, the data collection methods and the design of any hardware that was utilized.

7.3 Evidence for Hypothesis Support

This section looks at the evidence for the support or rejection of the research hypothesis presented in Section 3.3. It does this on a hypothesis-by-hypothesis basis. To support the hypothesis that behavioural morphism can be used to compare real and virtual world tasks this thesis must show support from the hypotheses that were to be tested above. This section argues that this support is present and presents the evidence for it.

7.3.1 Evidence for Support of Hypothesis 1

Hypothesis 1

*Human behaviour varies between real and virtual tasks*

This hypothesis is the general validation of the research carried out. It answers the question “Is there really a problem?” Support is proposed for accepting this thesis in two forms. The first is the evidence that adaptations such as the kinaesthetic adaptation and virtual adaptations take place post exposure to virtual environments (Stanney and Kennedy 1997; Groen and Werkhoven 1998; Howarth 1999). This is strong evidence that the behaviour of the human is different between real and virtual tasks. Secondly, experimental evidence in all three experiments undertaken show that the human task behaviour varies significantly with the real-world model or in the case of the rotary pursuit experiment the real-world dataset, this is shown in Table 4-5 Hypothesis Test (Correlation), and Table 6-6 Repeated-measures ANOVA, correlation. This is again strong evidence that behaviour varies in closely modelled virtual environment tasks. The presence of such distinct and significant interactions in Table 6-4 Repeated-measures ANOVA table, performance, Table 6-5 Repeated-measures ANOVA table, error and Table 6-6 Repeated-measures ANOVA, correlation that were traced to learning effects are also strong supportive data.

To support the hypothesis 1 this thesis presents the evidence that humans adapt and humans behave differently between real and virtual environments. Therefore, any
method of comparing real and virtual must take into consideration human behaviour or it could be argued that there are confounding factors of behaviour in the measurement. This hypothesis is therefore accepted within the context of this thesis.

In short, there really is a problem with assuming virtual task are the same as real tasks because of similarity in the user interaction.

7.3.2 Evidence for Support of Hypothesis 2

<table>
<thead>
<tr>
<th>Hypothesis 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Behavioural morphism is sensitive enough to compare the fidelity of real and virtual tasks</td>
</tr>
</tbody>
</table>

This hypothesis is the main thesis of this research, namely, can behavioural morphism be used to compare real and virtual tasks. To do this it must be able to identify differences in behaviour, performance and error, it must be able to work in the presence of a mathematical model and where no mathematical model exists. If the method can fulfil all these requirements then it is can be successfully argued that the method is valid for the purpose. To validate whether the method would fulfil these requirements the following hypotheses were tested.

7.3.2.1 Evidence for Support of Hypothesis 2:a

<table>
<thead>
<tr>
<th>Hypothesis 2:a</th>
</tr>
</thead>
<tbody>
<tr>
<td>Behavioural morphism is sensitive enough to identify different behaviour between the real and virtual tasks</td>
</tr>
</tbody>
</table>

This hypothesis tested the ability of behavioural morphism to identify behavioural differences in comparison with the original experiments and behavioural differences within an experimental design. The evidence for these differences is shown in 7.3.2.1.1, 7.3.2.1.2 and section 7.3.2.1.3 immediately below.

7.3.2.1.1 Choice Reaction

In this experiment the experimental correlation coefficient varied significantly from the original experiment (p <0.01) Table 4-5 Hypothesis Test (Correlation). This
demonstrated that the participant's behaviour was varying from the model significantly more than the original experiment. This shows behavioural differences being identified.

7.3.2.1.2 Rapid Aimed Movement

The correlation coefficients at each experimental level varied significantly (p<0.01) with original experiment in all conditions, Table 5-1 T-Test comparison of correlation coefficients this shows behavioural differences being identified. The correlation varied significantly across the stereo condition (p<0.05) this shows that the behaviour of the participant varied significantly within the experimental design due to the presence or lack of presence of a visual stereo cue. This demonstrates that the correlation measure can be used to identify behavioural differences across experimental conditions. Performance varied across the collision stimulus condition only (p<0.01) Table 5-3 Repeated measures ANOVA table for Performance since there was no significant behaviour change across this condition for the stereo condition this is taken as evidence that a behaviour may change but the performance may vary. The presence, at the same time, of a varying behaviour and a non-varying performance supports the argument that behavioural morphism identifies behavioural differences that would not be picked using a simple task performance measure. If we had only measured the task performance in this experiment then we would not know that the behaviour varied. This arguments also follow for the error measure since that only varied across the collision stimulus condition (p<0.01) Table 5-4 Repeated measures ANOVA table for error rate and not the stereo condition. This is important since identical performance could indicate comparable tasks when in fact differing behaviours occur and training could be confounded.

7.3.2.1.3 Rotary Pursuit Experiment

The rotary pursuit experiment showed differences in performance (p >0.01) Table 6-4 Repeated-measures ANOVA table, performance, error Table 6-5 Repeated-measures ANOVA table, error and behaviour Table 6-6 Repeated-measures ANOVA, correlation. Hypothesis 2:a was also supported by the presence of a learning effect in all of the measures.
All of the above results are taken to be evidence that supports hypothesis 2:a and this thesis thus supports the hypothesis that behavioural morphism can be used to identify differing behaviour between real and virtual tasks.

7.3.2.2 Evidence for Support of Hypothesis 2:b

Hypothesis 2:b

Behavioural morphism is sensitive enough to measure the fidelity of a task within a virtual environment where a mathematical model exists.

The argument for support of hypothesis 2:a using the data from the choice reaction and rapid aimed movement experiments is also valid for this experimental hypothesis. By using the original model the data not only identified a variance with the original model but in the case of the rapid aimed movement task it identified behavioural differences within the experimental design that would not have been apparent using task performance or error measures alone. This allows us to accept the hypothesis that behavioural morphism measures can be used to compare tasks where a mathematical model of the task exists. This argument applies to error, performance and behaviour.

7.3.2.3 Evidence for Support of Hypothesis 2:c

Hypothesis 2:c

Behavioural morphism is sensitive enough to measure the fidelity of a task within a virtual environment where no mathematical model exists.

As with hypothesis 2:a the data from the pursuit rotary experiment allows us to accept the hypothesis that behavioural morphism can be used where no mathematical model exists or it is not possible to use the model for the comparison due to some other reason. In this experiment the presence of a learning effect is very good indication that behavioural morphism measures are needed to validate virtual training environment. In this experiment the virtual condition was definitely training
something other than the real-world pursuit task; the presence of the learning effect makes this indisputable.

7.3.3 Hypothesis Support Conclusion

In all, the experiments and literature review undertaken in this thesis supported each other in this thesis identifying or demonstrating the following:

- Differing behaviour with the original model
- Differing behaviour across identical tasks using differing interfaces
- Differing performance across identical tasks using differing interfaces
- No significant difference in performance whilst a significant difference in behaviour.
- Learning effects in virtual conditions not present in real conditions.
- Adaptation to virtual-real visual and kinaesthetic mismatches.

All of the above results support the hypothesis that behavioural morphism can be successfully used for comparing real and virtual tasks, that is identified differences in performance, error and behaviour. The method identified differences due to stereo conditions and collision cues in identical experiments in the rapid aimed movement experiments and a learning effect in the rotary pursuit experiment that affected behaviour.

In conclusion, the behavioural morphism method succeeded in identifying behavioural, performance and error differences for the tasks investigated, it also provided a valuable insight into the differences between apparently identical tasks in real and virtual worlds. The aim of this thesis was to enable real and virtual tasks to be compared in terms of a faithful reproduction; this research has shown that behavioural morphism can provide one way of achieving this.
Chapter 8: Thesis Conclusion
8.1 Introduction

This chapter completes this thesis by presenting the areas of contribution to knowledge it makes and the further work that it recommends. In the final section of this chapter this thesis will be concluded with a summary of the work.

8.2 Original Contribution to Knowledge

Now that the research that this thesis undertook has been summarised and any hypothesis support discussed we are in a position to state explicitly where this thesis makes an original contribution to knowledge. This thesis extends human knowledge in the following key areas.

1. It conceives and proposes the first quantitative methodology for comparing real and virtual tasks for 'fidelity' incorporating human behaviour.

2. It identifies, proposes and demonstrates that human behaviour, performance and error can be used as a real-virtual fidelity 'metric' or datum.

3. It provides experimental evidence of behavioural differences between real and virtual tasks for choice reaction, rapid aimed movement and rotary pursuit tasks.

4. It provides evidence of virtual task performance remaining the same when behaviour changes significantly and thus the possibility of standard performance measures such as task performance and error being confounded for real-virtual comparisons.

5. It provides evidence of learning effects and behavioural differences between closely modelled real and virtual rotary pursuit tasks that demonstrate modern virtual motor skill training needs a behavioural validation and verification design review.

The main contribution this thesis has made is declared in point 1 in the list above. This contribution is in providing a method for comparing real and virtual tasks where none exists. It has been proposed for many years by many researchers that a comparison methodology such as this needs to be defined however, the normal research slants have been geared towards comparison of human sensory abilities, subjective presence and task performance rather than human behaviour.

Other contributions have also been made by this thesis, these include a full literature review of human behaviour performance measures within virtual environments,
evidence of behaviour modification due to stimulus modality of touch collision stimuli, behavioural modification due to lack of binocular cues. This thesis also demonstrates how to collect and compare real and virtual tasks using bespoke equipment and software.

8.3 Further Work

All of the experiments in this thesis should be carried out in fully immersive and semi-immersive virtual environments. The results should then be compared with this thesis and conclusions drawn as to the utility of behavioural morphism for cross-researcher validity. This is seen as an important area of research if the techniques described in this thesis are to be of use to the general field. Some of the experiments that have been undertaken previous to this thesis and which accept a 'priori models of human behaviour should be investigated for possible behavioural variation to lend further strength to the argument that behavioural aspects need to be taken into consideration in training or fidelity studies.

Other tasks that can be mathematically modelled should be tested to identify areas of weakness within the methodology. Questions of sampling effect in the correlation coefficient need addressing; if the original study used few points to construct the correlation how valid are its results? The effects of the original experimental methodology also need investigation, for instance the original choice reaction experiment by Hick used only four participants and would these days be argued as being statistically weak due to sample size, however the model has been tested time and time again over the last fifty years and has yet to be found wanting.

Most interesting future work would be in the use of immersive virtual workbenches with haptic feedback. This would enable the effects of kinaesthetic adaptation on behaviour to be investigated thoroughly. As noted in the experiments this methodology has identified behavioural and performance differences due to design factors such as provision of stereo cue and collision stimulus cue, a very important piece of future research would be the systematic study of the effects of all system dependent (FOV, resolution colour depth, haptic feedback etc) variables on task
behaviour using behavioural morphism to factor out the various effects into behaviour, performance and error effectors.

A study of training environments that are being proposed, implemented or used should be carried out to identify any possible weakness within the behavioural assumptions of the system. Again, any differences found would lend further support to the findings of this thesis and aid training providers in supplying systems that are functionally useful and whose motor skill task training characteristics can be quantified, verified and validated.
8.4 Thesis Conclusion

The experimental evidence in this thesis has supported the findings of the literature review; human behaviour is an issue in virtual interfaces that up until now has been ignored with the exception of a few important works into human adaptations to virtual environment interfaces. No task fidelity measures are available, at present, which take into consideration human behaviour. This allows, at the very least, an argument for the presence of undiagnosed negative training transfer due to training task mismatch.

There is no common virtual environment platform for comparative studies to be carried out in virtual environments, this means that researchers will have a problem generalising their results and may even make reproducibility an issue. The background chapter in this thesis demonstrated many of the problems with obtaining 'identical systems'. Even if we assume that we don't need identical systems how do we measure the closeness of two systems that vary in components? This thesis reports that stereo glasses modify behaviour at a tapping task; do different stereo glasses modify behaviour differently? The complexity argument must rule out comparisons based on using standardised technology, it is too difficult to factor out the constituent parts of a system and constrain them across studies and across researchers.

This thesis has provided evidence in the literature review of adaptations to virtual environment interfaces. It has also provided evidence through experimentation of behavioural differences between real and virtual tasks that appear at the outset to be identical. These two facts show in sharp relief that relying on a similarity in appearance between real and virtual tasks for training or software validation and verification is not a good idea and could even be counterproductive. For instance, in a virtual surgical training context it is risky to assume tasks are identical a’priori since adaptations take time to disappear and until they do disappear proprioception may be altered. How many people would want to be operated upon by a brain surgeon who thinks his or her right hand is two centimetres to the left of its real position because he or she has been rehearsing the operation in a virtual simulator? This is not a far-fetched scenario since virtual environments are used for rehearsal of surgery at present; some have even been reviewed in this thesis.
As we can see once behavioural differences between real and virtual tasks are identified it becomes a matter of necessity to quantify the fidelity of the real task with the virtual task, without such a measure we cannot validate or verify the training behaviour of the system. Using performance as a measure would not work since, as this thesis shows, performance can be static whilst behaviour varies. Clearly, this means that any performance-based measure would be incapable of identifying behavioural differences that did not affect performance. Humans use many strategies to overcome problems.

Once could use behavioural morphism to indirectly characterise systems with respect to each other; because we are using the real-world as a datum comparing system with system becomes a case of comparing the difference of each system with the real-world case. This datum based comparison process is technology independent and can be used as an indicator of whether systems a comparable or not. If one virtual environment system had a rapid-aimed movement behavioural coefficient of 0.98 and another had one of 0.75 we can see that the human is clearly behaving differently in each environment and any comparison would be fraught with confounding arguments. However, if they both had correlations of 0.98 the behaviour could be assumed to be close enough to be comparable.

The work detailed in this document has proposed one solution to the problem of measuring the fidelity of real and virtual tasks. Behavioural morphism may not be the definitive answer to all problems but it brings a novel perspective to comparative human behaviour in virtual environments and the psychology of virtual environment interactions.
This page is intentionally blank
9 References


Behavioural Morphisms In Virtual Environments


Simon Nee 24/05/2002


*Simon Nee 24/05/2002*
Behavioral Morphisms in Virtual Environments


10 Appendix 1 Experimental Software

To enable the experiments described in the previous chapter to take place three completely different virtual environments had to be constructed. A simple requirements analysis of the experimental methodology was undertaken. After a study of the requirements analysis and the available software solutions it was concluded that no commercial package provided solutions to all the requirements identified and that a hybrid solution was required. Using a COTS (Commercial off-the-shelf) product could not have provided an overall solution because the virtual environment systems were also required to arbitrate the control of the experiment, schedule the data collection within the experiments and output the collected data. However, the requirements could be met by augmenting a COTS product with bespoke software and combining them into a hybrid solution.

Creating the experimental virtual environments also necessitated integrating both commercial available hardware and software with proprietary/bespoke hardware and software. For instance, all the experiments carried out had a requirement for visual stereo cues to be present however the stereo glasses available and the computer system used were not compatible so an electronic "black box" had to be used that would allow the stereo glasses to function by providing an interface between the computer and the glasses. The data collection system for the pursuit rotor experiment had to be built from the ground up since no COTS solution existed.

The construction of software and hardware to fulfil the experimental requirements specification for this thesis was treated as a software engineering process. However, no software engineering development lifecycle paradigm such as the waterfall model, spiral model or others investigated (Sommerville 1992; Bennett, McRobb et al. 1999) appeared to appropriate since they were designed for larger projects with more that a single developer, instead a small-scale rapid prototyping iterative methodology (Dix, Finlay et al. 1998) was adopted and the software verified via user piloting (Preece, Sharp et al. 1994). The utilization of user piloting coincidentally fulfilled a requirement for an experimental methodology validation technique (Coolican 1996).
This chapter deals with the design and implementation of the experimental virtual environments. It pays special attention to the difficulties that were encountered and shows how they were overcome. This chapter demonstrates the wide range of skills needed to implement a simple real-world example of a highly targeted virtual training environment. This chapter also provides further evidence of how difficult it is to develop a static research platform due to the innate complexity involved in designing, developing and producing virtual environments.

### 10.1.1 Experimental Software Design

#### 10.1.1.1 Base Virtual Environment Software Architecture

All of the experimental virtual environments in this thesis utilized as their base building block a commercially available development software platform produced by Division PLC called dVS/dVISE. This development software provides a development infrastructure abstracts into three main component, these components are listed in below.

- Run-time virtual environment - *dVS*
- Software API – *VCLib*
- Geometry conversion tools

**Table 10-1 dVS/dVISE Main Components**

These three components enable customized virtual environments to be developed without a major bottom-up software development process being undertaken. The dVISE component of the system is not featured in this document since it is a high-level design component designed to allow non-programmers to use the system. dVISE was not utilized during development or experimentation.

The other reasons for using the Division development software were its' performance, the supported available peripherals and the supported target computing platforms. The software ran on the SGI Indigo² platform that was available and gave the 60Hz graphics performance required. This meant that the platform natively supported the field sequential graphics presentation method needed to provide stereo images.
through stereo glasses and the Polhemus Fastrak 6DOF tracking system required to track the users hand movements during experiments.

10.1.1.1 Run-time Virtual Environment

The run-time virtual environment component of the dVS/dVISE system (dVS) provides the graphical and simulation software primitives, in other words it renders the images of the entities in the virtual environment, keeps track of their positions, monitors user input and other base virtual environment functions. It does this by combining a number of configurable software ‘Actors’ or real-time object-oriented processing agents into a simulation federation. In all six actors are available, these actors are added to the virtual environment and configured by manipulating a file known as the “registry file”. The six actors and their broad responsibilities are as follows.

- **Physics** - Provides basic physics for simulation objects
- **Visual** - Rendering and visual output
- **Body** - Avatar Representation
- **Input** - Input controllers
- **Collision** - Collision detection
- **Audio** - Auditory localization of objects

Using actors like these allows many different hardware components to be supported simply by adding a version of the actor that provides the functionality of the required device. For instance, the visual actor can set to render using the OpenGL or Performa libraries under SGI by loading different versions of the visual actor. The same can be done for different tracking systems using the input actor or for different models of HMD again using the visual actor.

The run-time virtual environment was used to provide the base functionality mentioned above and relieve the researcher of the need to program these facilities from the bottom up. Programming these basic facilities at a sufficient performance would have been a substantial project in itself.
The VCLib software API provides a method of extending the functionality of the run-time system. It does this allowing standard the main program loop of the run-time system with our bespoke code. For instance, no experimental procedural control structure is present within the default run-time main program loop other than that which was provided via the physics actor, i.e. Newtonian physics. This meant that the procedural control that the experiments required had to be programmed into each environment. After this program code has been developed, the runtime environment is recompiled into a completely new run-time virtual environment application. The high-level structure of the virtual environment applications developed using dVS is shown in Figure 10.1 below.

There is functionality that need to be added but is event driven and therefore does not belong in the main program loop. This code is added by inserting event callback functions that ‘fire’ when an event of interest occurs. A callback function is a piece of computer code that is called in response to some system event.
10.1.1.1.3 **Geometry Tools**

The geometry tools supplied with dVS/dVISE allowed the geometrical/visual aspects of a virtual environment to be constructed in professional modelling tools such as MultiGen and 3DStudioMax and then transformed into a format that is compatible with the run-time software detailed above. This allowed suitable tools to be used to construct the 3D models within the virtual environments easing the process of model construction and assisting in the process of building accurate 3D models. Without these state-of-the-art tools the process of modelling the 3D models would have been complex and time consuming.

The geometry tools also allow the optimisation of the supplied 3D models for real-time display by creating flattened hierarchy of objects that have been organized into triangular strips and fans to assist the geometric transformation operations that are out within the system.

Figure 10.2 Example Tristrip

Tri-stripping (Figure 10.2) reduces the number of computations needed to recalculated vertex positions by allowing vertex sharing to occur between polygons i.e. if two triangle polygons share a single vertex only five vertex transformation need take place as opposed to six without sharing. Tri-fanning (Figure 10.3) is a similar operation
that uses a fan shape rather than a strip structure to take advantage of vertex redundancy within polygonal models.

Figure 10.3 Example Trifan

There are also polygon decimation facilities that allow the number of polygons within polygonally complex models to be reduced in number. Figure 10.4 below shows an example of polygon decimation on a simple object. However, these decimation algorithms can be very complex due to the infinite variety of shapes that can be constructed.

Figure 10.4 Polygon decimation
The geometry tools aided the development by allowing the "tailoring" of geometric models to a level where the visual appearance and system performance were acceptable for real time interaction. This was tested via pilot experimentation.
#include <dvs/vc.h>
#include <stdio.h>
#include <string.h>
#include "include/col.h" /* Collision List data structure */
#include "include/settings.h"

#define PI 3.1415926
#define TRIALS 20
#define BLOCKS 8
#define WARNMAX 5
#define NUMRAND 2000 /* Number of random swaps in block trial array */
#define MAXBLOCKS 8

void arrangeStimulusBlocks(int blockNumber);
void arrangeResponseBlocks(int blockNumber);
void touch(VCBodyAppCallbackData *cd, void *data);
void untouch(VCBodyAppCallbackData *cd, void *data);
void removeUnwantedBlocks(int number);

These are global data structures for experimental data, it's simpler this way.
/*+++++++++++++++++++*/
struct collisionList *head,*current;
char *filename;
int trialCounter=0,blockCounter=0,errors=0;
int position[BLOCKS][TRIALS];
int WaitingForReaction=0,numberOfBlocks;
int touchButtonFlag;
/*
Global geometry stuff so call backs can reach them easily
*/
/*Objects*/
VCEntity *Button,*Stimulus[8],*plate[8],*collisionNoise;
/*Visuals*/
VCAttribute *ButtonVis,*stimulusVis[8],*plateVis[8];
/*Boundaries*/
VCAttribute *ButtonBoundary,*plateBoundary[8];

VCAttribute *click;

VCColor white={1,1,1};

void End()
{
/*
This function outputs the data in the linked list to a file
*/

FILE *fp;
char str[80];
int epochvalue;
double secstart, secstop, nanostart, nanostop;

current = head;
/*Calculate durations involves transforming from long ints to double floats*/

while (current->nextCollision != NULL)
{
    secstart=(current->start.tv_sec);
    secstop=(current->pressStop.tv_sec);
    nanostart=(current->start.tv_nsec);
    nanostop=(current->pressStop.tv_nsec);
    nanostart=(nanostart/1000000000);
    nanostop=(nanostop/1000000000);
    secstart=secstart+nanostart;
    secstop=secstop+nanostop;
    current->pressButtonDuration=secstop-secstart;

    secstart=(current->start.tv_sec);
    secstop=(current->leaveStop.tv_sec);
    nanostart=(current->start.tv_nsec);
    nanostop=(current->leaveStop.tv_nsec);
    nanostart=(nanostart/1000000000);
    nanostop=(nanostop/1000000000);
    secstart=secstart+nanostart;
    secstop=secstop+nanostop;
    current->leaveButtonDuration=secstop-secstart;

    current=current->nextCollision;
}

/*Repoint to head of list to output data*/
current = head;

/*
 This is where the data is put into the file
 copy directory path and append filename from command line
*/

strcpy(str,"data/choice/");
strcat(str, filename);

if((fp = fopen(str,"w")) == NULL)
{
    printf("Error opening file\n");
    exit(1);
}

fprintf(fp,"B.T.C	Start	React	Press	ReactDur	PressDur	SeaChar	PreChar	Pos\n");

while (current->nextCollision != NULL)
{
    fprintf(fp,"%d@%d@%d	",current->Block,current->Trial,current->NumberOfBlocks);
    fprintf(fp,"%u.%u	"," current->start.tv_sec,current->start.tv_nsec);
    fprintf(fp,"%u.%u	",current->leaveStop.tv_sec,current->leaveStop.tv_nsec);
}
fprintf(fp,"%u.%u	", current->pressStop.tv_sec, current->pressStop.tv_nsec);
fprintf(fp,"%f	", current->leaveButtonDuration);
fprintf(fp,"%f	", current->pressButtonDuration);
fprintf(fp,"%c	", current->searchCharacter);
fprintf(fp,"%c	", current->pressedCharacter);
fprintf(fp,"%d
", current->Position);

current=current->nextCollision;

fprintf(fp,"%s\n", filename);
fclose(fp);

VCVisual_SetFrontMaterial(ButtonVis, "Room:Orange");

}

void doTrial(int StimulusPos)
{
    struct timespec startTime;
    char name[]="Letters:A";

    name[8]=(char)(StimulusPos+64+8);
    printf("Name %s\n",name);
sleep(3);
    VCVisual_SetFrontMaterial(stimulusVis[StimulusPos-1],name);
    clock_gettime(CLOCK_REALTIME,&startTime);
current->start.tv_sec=startTime.tv_sec;
current->start.tv_nsec=startTime.tv_nsec;
WaitingForReaction=1;
}

void fillArray()
{
    int x,y;
srand( (unsigned)time( NULL ) );

    for (x=0;x<BLOCKS;x++)
    {
        for (y=0;y<TRIALS;y++)
        {
            position[x][y]=x+1;
        }
    }
}

void randomizeArray()
{
    int x,y, tempswap, pos1x,pos2x,pos1y,pos2y;
    /* Randomize array contents */

    for (x=0;x<NUMRAND;x++)
    {
        pos1x=(int)(rand()% BLOCKS);
p0s1y=(int)(rand()% TRIALS);
pos2x=(int)(rand()% BLOCKS);
}
pos2y = (int) (rand() % TRIALS);

tempswap = position[poslx][posly];

position[poslx][posly] = position[pos2x][pos2y];
position[pos2x][pos2y] = tempswap;

void checkForDataOrExit()
{
    current->nextCollision = (struct collisionList *)malloc(sizeof(struct
    collisionList));
    current->nextCollision->previousCollision = current;
    current = current->nextCollision;

    if((blockCounter + 1 == BLOCKS) && (trialCounter + 1 == TRIALS))
    {
        printf("Experiment Completed
");
        End();
    }
    else
    {
        if(trialCounter + 1 == TRIALS)
        {
            trialCounter = 0;
            blockCounter = blockCounter + 1;
            VCVisual_SetFrontMaterial(ButtonVis, "Room:Orange");
            sleep(60);
            VCVisual_SetFrontMaterial(ButtonVis, "Room:Green");
        }
        else
        {
            trialCounter = trialCounter + 1;
        }
    }
}

void setTrial(int blockNumber)
{
    numberOfBlocks = blockNumber;
    current->NumberOfBlocks = blockNumber;
    current->Trial = trialCounter + 1;
    current->Block = blockCounter + 1;
    arrangeStimulusBlocks(blockNumber);

Simon Nee 24/05/2002
arrangeResponseBlocks(blockNumber);
removeUnwantedBlocks(blockNumber);

if (blockNumber==1)
{
current->searchCharacter='A';
current->Position=1;
pos= 1;
printf("pos %d\n",pos);
}
else
{
pos=((rand()%blockNumber)+1);
current->searchCharacter=(char)(pos+64);
current->Position=pos;
printf("pos %d\n",pos);
}
doTrial(pos);

void removeUnwantedBlocks(int number)
{
int x;

for(x=number;x<8;x++)
{
VCEntity_Translate(plate[x],100,100,100);
VCEntity_Translate(stimulus[x],100,100,100);
printf("Translating %d\n",x);
}
}

void replaceUnwantedBlocks(int number)
{
int x;

for(x=number;x<numberOfBlocks;x++)
{
VCEntity_Translate(plate[x],-100,-100,-100);
VCEntity_Translate(Stimulus[x],-100,-100,-100);
printf("Detranslating %d\n",x);
}
}

void arrangeResponseBlocks(int number)
{
int x;
char mat[]="Letters:A";
dmPoint pos;

for(x=0;x<number;x++)
{
mat[x]=(char)(x+65);

if(number!=1)
behavioural morphisms in virtual environments

```c
{ 
    pos[0] = .27*sin(3*PI/2 + ((PI/(number-1))*x));
    pos[1] = 0.75;
    pos[2] = (-.27*cos(3*PI/2 + ((PI/(number-1))*x)))*0.15;

    VCEntity_SetPositionPoint(plate[x],pos);
}

else
{
    pos[0] = 0.0;
    pos[1] = 0.75;
    pos[2] = -0.135;

    VCEntity_SetPositionPoint(plate[x],pos);
}

VCVisual_SetFrontMaterial(plateVis[x],mat);
}

void arrangeStimulusBlocks(int number)
{
    int x;
    dmPoint pos;
    char mat[]="Letters:A"

    for(x=0;x<number;x++)
    {
        mat[8] = (char)(x+65);
        pos[0] = (x*0.05)-(0.025*(number-1));
        pos[1] = 0.76;
        pos[2] = -0.22;
        VCEntity_SetPositionPoint(Stimulus[x],pos);
        VCVisual_SetFrontMaterial(stimulusVis[x],mat);
    }
}

/*#..............................................................*/

void createBlocks()
{
    int x;
    char mat[]="Letters:A"

    for(x=0;x<8;x++)
    {
        mat[8] = (char)(x+65);
        plate[x] = VCEntity_Create(NULL,0);
        plateVis[x] =
            VCVisual_Create("geometry/Button",NULL,VC_VISUAL_ENABLE,NULL,NULL,NUL,
            0);  
        VCEntity_AttachAttribute(plate[x],plateVis[x]);
        plateBoundary[x] = VCEntity_AddBoundaryGeometry(plate[x],"geometry/Button");
        VCEntity_Scale(plate[x],0.003,0.001,0.003);
        VCEntity_RotateX(plate[x],3.142);
        Stimulus[x] = VCEntity_Create(NULL,0);
```

Simon nec 24/05/2002 222
stimulusVis[x] =
VCVisual_Create("geometry/Button", NULL, VC_VISUAL_ENABLE, NULL, NULL, NULL, 0);
    VCEntity_AttachAttribute(Stimulus[x], stimulusVis[x]);
    VCEntity_Scale(Stimulus[x], 0.003, 0.001, 0.003);
    VCEntity_RotateX(Stimulus[x], 4.142);
    VCVisual_SetFrontMaterial(plateVis[x], mat);
    VCVisual_SetFrontMaterial(stimulusVis[x], mat);
{ 
touchButtonFlag=1;
VCVisual_SetFrontMaterial(ButtonVis, "Room:Red");
return;
}

if(WaitingForReaction==1)
{
clock_gettime(CLOCK_REALTIME, &stopTime);
current->pressStop.tv_sec=stopTime.tv_sec;
current->pressStop.tv_nsec=stopTime.tv_nsec;
}

if(cd->entity==plate[0])
{
current->pressedCharacter='A';
VCVisual_SetFrontMaterial(plateVis[0], "Room:Red");
}
if(cd->entity==plate[1])
{
current->pressedCharacter='B';
VCVisual_SetFrontMaterial(plateVis[1], "Room:Red");
}
if(cd->entity==plate[2])
{
current->pressedCharacter='C';
VCVisual_SetFrontMaterial(plateVis[2], "Room:Red");
}
if(cd->entity==plate[3])
{
current->pressedCharacter='D';
VCVisual_SetFrontMaterial(plateVis[3], "Room:Red");
}
if(cd->entity==plate[4])
{
current->pressedCharacter='E';
VCVisual_SetFrontMaterial(plateVis[4], "Room:Red");
}
if(cd->entity==plate[5])
{
current->pressedCharacter='F';
VCVisual_SetFrontMaterial(plateVis[5], "Room:Red");
}
if(cd->entity==plate[6])
{
current->pressedCharacter='G';
VCVisual_SetFrontMaterial(plateVis[6], "Room:Red");
}
if(cd->entity==plate[7])
{
current->pressedCharacter='H';
VCVisual_SetFrontMaterial(plateVis[7], "Room:Red");
}

if(WaitingForReaction==1)
{
WaitingForReaction=0;
removeUnwantedBlocks(0);
}
void untouch(VCBodyAppCallbackData *cd, void *data) {
    struct timespec stopTime;
    VCAudio_Start(click);
    printf("untouched\n");
    if (cd->entity==Button) {
        touchButtonFlag=0;
        if (WaitingForReaction==1) {
            clock_gettime(CLOCK_REALTIME, &stopTime);
            current->leaveStop.tv_sec=stopTime.tv_sec;
            current->leaveStop.tv_nsec=stopTime.tv_nsec;
        }
    }
    VCVVisual_SetFrontMaterial(ButtonVis, "Room:Green");
}

if (cd->entity==plate[0]) {
    VCVVisual_SetFrontMaterial(plateVis[0], "Letters:A");
}
if (cd->entity==plate[1]) {
    VCVVisual_SetFrontMaterial(plateVis[1], "Letters:B");
}
if (cd->entity==plate[2]) {
    VCVVisual_SetFrontMaterial(plateVis[2], "Letters:C");
}
if (cd->entity==plate[3]) {
    VCVVisual_SetFrontMaterial(plateVis[3], "Letters:D");
}
if (cd->entity==plate[4]) {
    VCVVisual_SetFrontMaterial(plateVis[4], "Letters:E");
}
if (cd->entity==plate[5]) {
    VCVVisual_SetFrontMaterial(plateVis[5], "Letters:F");
}
if (cd->entity==plate[6]) {
    VCVVisual_SetFrontMaterial(plateVis[6], "Letters:G");
}
if (cd->entity==plate[7]) {
    VCVVisual_SetFrontMaterial(plateVis[7], "Letters:H");
}

/*##################################################################*/
void startTrials(VCBodyInput_CallbackData *cd, void *data) {
    Simon Nee 24/05/2002
}
printf("Start Trials\n");
if (touchButtonFlag==1)
{
    getTrialSetup();
}
else
{
    printf("User is not touching the start position button\n");
}
/**

        This grabs hold of the hand so we can pick the probe up via the data parameter
        */
    char *partName;
    VCBodyPartGetName(partData->bodyPart, &partName);
    printf("\n%s\n",partName);
    if (strcmp(partName, "rightHand") == 0)
    {
        /*
         * VCEntity_Pick((VCEntity *)data,partData->bodyPart,NULL,NULL);
         */
    }
    /*
    Grab hold of hand when it is created by searching each create callback
    When a body part is created it generated this callback.
    */
}
void positionBody(VCBodyCreate_CallbackData *cd, void *data)
{
    /*
    Function positioning of body in front of table
    */
    dmPoint position = {0.0,1.2,0.7};
    dmEuler orientation = {-4.0,0.0};
    dmScale scale = {1,1,1};
    printf("\nGot here! ! 3 \n");
    VCBody_setPosition(cd->body,NULL,position,orientation,scale,NULL,0);
    VCBody_AttachBodyPartCreateCallback(cd->body,processBodyParts,data);
}
int main (int argc, char **argv)
{
    ActorId actor;

    *Walls,*Ceiling,*Floor,*TableTop,*Legs,*light1,*light2,*light3;
    VCCollision
    *ButtonCollision,*HoldButtonRightCollision,*HoldButtonLeftCollision;
    static VCColour ambient = {0.45, 0.45, 0.45};
    static VCColour directional = {0.9, 0.9, 0.9};

    /* Fill array*/
    fillArray();
    randomizeArray();

    /*Initialize timer */
    /*Initialise linked list and pointers.*/

    head = current = (struct collisionList *)malloc(sizeof(struct collisionList));

    #if DEBUG >= 3
    /*
    Debug output for checking initialization of list
    */
    printf(" head->counter \d\n",head->counter);
    printf(" head \d current \d\n", head,current);
    #endif
    actor=VC_InitApplication(&argc, argv, NULL);

    if (actor != VC_OK)
        exit (1);

    /*
    This is where the environment is created
    */

    light1=VCEntity_create(NULL, 0);
    light2=VCEntity_create(NULL, 0);
    light3=VCEntity_Create(NULL, 0);
    VCEntity_AddLightAmbient(light1,ambient);
    VCEntity_AddLightPoint(light2,directional );
    VCEntity_Translate(light2,0,2,0);

    Walls = VC_ConstructVisualGeometry("geometry/Walls",0,NULL,NULL);
    Ceiling = VC_ConstructVisualGeometry("geometry/Ceiling",0,NULL,NULL);
    Floor = VC_ConstructVisualGeometry("geometry/Floor",0,NULL,NULL);
Behavioural Morphisms In Virtual Environments

TableTop = VC_ConstructVisualGeometry("geometry/TableTop", 0, NULL, NULL);
VCEntity_Scale(TableTop, 0.1, 0.1, 0.1);
Legs = VC_ConstructVisualGeometry("geometry/Legs", 0, NULL, NULL);
VCEntity_Scale(Legs, 0.1, 0.1, 0.1);

Button = VCEntity_Create(NULL, O);
ButtonVis = VCVisual_Create("geometry/Button", NULL, VC_VISUAL_ENABLE, NULL, NULL, NULL, 0);
VCEntity_AttachAttribute(Button, ButtonVis);
ButtonBoundary = VCEntity_AddBoundaryGeometry(Button, "geometry/Button");
VCBoundary_GetCollision(ButtonBoundary, &ButtonCollision);
VCBoundary_ModifyMode(ButtonBoundary, VC_COLLISION_ENABLE, VC_COLLISION_NO_POSITION);
VCEntity_Scale(Button, 0.005, 0.001, 0.005);
VCEntity_RotateX(Button, 3.142);
VCEntity_Translate(Button, 0.0, 0.75, 0.15);
VCVisual_SetFrontMaterial(ButtonVis, "Room:Green");

/* Set start Width and amplitude here case 1 epoch */
/* name file from second command line parameter */
createBlocks();
filename = argv[1];
collisionNoise = VC_ConstructAudioVoice("down", &click);
VC_AttachBodyCreateCallback(positionBody, NULL);
arrangeResponseBlocks(8);
arrangeStimulusBlocks(8);

VCBody_AttachInteractionCallbacks(NULL, touch, untouch, NULL, NULL, NULL, NULL);
VCBody_AttachInputCallback(NULL, NULL, (VC_INPUT_PRESS | 't'), 0, startTrials, NULL);
printf("Here\n");
VC_MainLoop();

#include <dvs/vc.h>
#include <stdio.h>
#include <string.h>
#include <sys/time.h>
#include "col.h" /* Collision List data structure */
#include "settings.h"

These are global data structures for experimental data, it's simpler
this way.

*/
int errors=0;
float plateamplitude = 0;
float platewidth = 0;
struct itimerval realtimer,virtuatimer,prooftimer,setvalue,optvalue;
struct collisionList *head,*current;
char *filename;
int trialCounter=0,epochCounter=1;

/*
Global geometry stuff so call backs can reach them easily
*/
VCEntity *LPlate,*RPlate,*Probe;
VCAtribute *LPlateVis,*RPlateVis,*ProbeVis;
VCColor white={
    1,1,1};

void adjPlates()
{
    dmPoint p;
dmEuler euler;
dmScale s;

    switch(epochCounter)
    {
        case 1:
            dmPointSet (p, -0.04375, 0, 0);
dmScaleSet (s, 0.01, 0.1, 0.1);
dmEulerSetD (euler, 0, 0, 0);
VCEntity_setpositionPointEulerScale (LPlate, p, euler, s);
dmPointSet (p, 0.04375, 0, 0);
dmScaleSet (s, 0.01, 0.1, 0.1);
dmEulerSetD (euler, 0, 0, 0);
VCEntity_setpositionPointEulerScale (RPlate, p, euler, s);
    current->epoch = epochCounter;
    epochCounter=epochCounter++; 
    trialCounter = 0 ;
    plateamplitude = 0.0875;
    platewidth = 0.025;
case 2:
  dmPointSet (p, -0.06875, 0, 0);
  dmScaleSet (s, 0.01, 0.1, 0.1);
  dmEulerSetD (euler, 0, 0, 0);
  VCEntity_SetPositionPointEulerScale (LPlate, p, euler, s);
  dmPointSet (p, 0.06875, 0, 0);
  dmScaleSet (s, 0.01, 0.1, 0.1);
  dmEulerSetD (euler, 0, 0, 0);
  VCEntity_SetPositionPointEulerScale (RPlate, p, euler, s);
  current->epoch = epochCounter;
  epochCounter = epochCounter++;
  trialCounter = 0;
  plateAmplitude = 0.1375;
  plateWidth = 0.025;
  break;

case 3:
  dmPointSet (p, -0.09375, 0, 0);
  dmScaleSet (s, 0.01, 0.1, 0.1);
  dmEulerSetD (euler, 0, 0, 0);
  VCEntity_SetPositionPointEulerScale (LPlate, p, euler, s);
  dmPointSet (p, 0.09375, 0, 0);
  dmScaleSet (s, 0.01, 0.1, 0.1);
  dmEulerSetD (euler, 0, 0, 0);
  VCEntity_SetPositionPointEulerScale (RPlate, p, euler, s);
  current->epoch = epochCounter;
  epochCounter = epochCounter++;
  trialCounter = 0;
  plateAmplitude = 0.1875;
  plateWidth = 0.025;
  break;

case 4:
  dmPointSet (p, -0.14375, 0, 0);
  dmScaleSet (s, 0.01, 0.1, 0.1);
  dmEulerSetD (euler, 0, 0, 0);
  VCEntity_SetPositionPointEulerScale (LPlate, p, euler, s);
  dmPointSet (p, 0.14375, 0, 0);
  dmScaleSet (s, 0.01, 0.1, 0.1);
  dmEulerSetD (euler, 0, 0, 0);
  VCEntity_SetPositionPointEulerScale (RPlate, p, euler, s);
  current->epoch = epochCounter;
  epochCounter = epochCounter++;
  trialCounter = 0;
  plateAmplitude = 0.2875;
  plateWidth = 0.025;
  break;

case 5:
  dmPointSet (p, -0.19375, 0, 0);
  dmScaleSet (s, 0.01, 0.1, 0.1);
  dmEulerSetD (euler, 0, 0, 0);
  VCEntity_SetPositionPointEulerScale (LPlate, p, euler, s);
  dmPointSet (p, 0.19375, 0, 0);
  dmScaleSet (s, 0.01, 0.1, 0.1);
  dmEulerSetD (euler, 0, 0, 0);
  VCEntity_SetPositionPointEulerScale (RPlate, p, euler, s);
  current->epoch = epochCounter;
  epochCounter = epochCounter++;
  trialCounter = 0;

Simon Nee 24/05/2002
Behavioural Morphisms In Virtual Environments

plateamplitude = 0.3875;
platewidth = 0.025;
break;

case 6: dmPointSet (p, -0.071875, 0, 0);
dmScaleSet (s, 0.005, 0.1, 0.1);
dmEulerSetD (euler, 0, 0, 0);
VCEntity_SetPositionPointEulerScale (LPlate, p, euler, s);
dmPointSet (p, 0.071875, 0, 0);
dmScaleSet (s, 0.005, 0.1, 0.1);
dmEulerSetD (euler, 0, 0, 0);
VCEntity_SetPositionPointEulerScale (RPlate, p, euler, s);

current->epoch = epochCounter;
epochCounter=epochCounter++;
trialCounter = 0;
plateamplitude = 0.14375;
platewidth = 0.0125;
break;

case 7: dmPointSet (p, -0.096875, 0, 0);
dmScaleSet (s, 0.005, 0.1, 0.1);
dmEulerSetD (euler, 0, 0, 0);
VCEntity_SetPositionPointEulerScale (LPlate, p, euler, s);
dmPointSet (p, 0.096875, 0, 0);
dmScaleSet (s, 0.005, 0.1, 0.1);
dmEulerSetD (euler, 0, 0, 0);
VCEntity_SetPositionPointEulerScale (RPlate, p, euler, s);

current->epoch = epochCounter;
epochCounter=epochCounter++;
trialCounter = 0;
plateamplitude = 0.19375;
platewidth = 0.0125;
break;

case 8: dmPointSet (p, -0.146875, 0, 0);
dmScaleSet (s, 0.005, 0.1, 0.1);
dmEulerSetD (euler, 0, 0, 0);
VCEntity_SetPositionPointEulerScale (LPlate, p, euler, s);
dmPointSet (p, 0.146875, 0, 0);
dmScaleSet (s, 0.005, 0.1, 0.1);
dmEulerSetD (euler, 0, 0, 0);
VCEntity_SetPositionPointEulerScale (RPlate, p, euler, s);

current->epoch = epochCounter;
epochCounter=epochCounter++;
trialCounter = 0;
plateamplitude = 0.29375;
platewidth = 0.0125;
break;

case 9: dmPointSet (p, -0.196875, 0, 0);
dmScaleSet (s, 0.005, 0.1, 0.1);
dmEulerSetD (euler, 0, 0, 0);
VCEntity_SetPositionPointEulerScale (LPlate, p, euler, s);
dmPointSet (p, 0.196875, 0, 0);
dmScaleSet (s, 0.005, 0.1, 0.1);
dmEulerSetD (euler, 0, 0, 0);
VCEntity_SetPositionPointEulerScale (RPlate, p, euler, s);

current->epoch = epochCounter;
epochCounter=epochCounter++;
trialCounter = 0;
plateamplitude = 0.39375;
platewidth = 0.0125;
break;

case 10: dmPointSet (p, -0.296875, 0, 0);
dmScaleSet (s, 0.005, 0.1, 0.1);
dmEulerSetD (euler, 0, 0, 0);
VCEntity_SetPositionPointEulerScale (LPlate, p, euler, s);
dmPointSet (p, 0.296875, 0, 0);
dmScaleSet (s, 0.005, 0.1, 0.1);
dmEulerSetD (euler, 0, 0, 0);
VCEntity_SetPositionPointEulerScale (RPlate, p, euler, s);
current->epoch = epochCounter;
epochCounter = epochCounter++;
trialCounter = 0;
plateamplitude = 0.59375;
platewidth = 0.0125;
break;

default :
    { void printf("Unknown option\n");
      exit(2);
    }

}"# ##################################################################
/ void OnMaxTrials()
{
    /*
    This function outputs the data in the linked list to a file
    */
    FILE *fp;
    char str[80];
    int epochvalue;
    double realtime,starttime;
#if DEBUG >~ 1
    /*
    This is all for debug purposes
    */
    current = head ;
    /*
    point current pointer at the head of the list
    */
    {
        printf(" head %d current %d \n", head,current);
    }
printf("###Collision List Report###\n");
printf("Epoch\Trial\tTime\tX Position\tY Position\n");
while ( current->nextCollision !=NULL) {

realtime = current->collision.time.secs+(current->collision.time.uSecs/1000000000.0);

printf("%3d\t", ((current->epoch)-1));
printf("%4.d\t", current->counter);
printf("%f\t", realtime);
printf("%f\t", current->collision.point[0]);
printf("%f\n", current->collision.point[2]);

if (current->counter == MAXTRIALS)
{
    printf("\n");
    printf("Plate Width	Plate Amplitude\t(A/W)\n");
    printf("%f\t%f\t%f\n", current->targetWidth,current->targetAmplitude,(current->targetAmplitude)/current->targetWidth);
    printf("\n");
}

current= current->nextCollision;

printf("\n");
#endif

/*
This is where the data is put into the file
*/
current = head;

/*
copy directory path and append filename from command line
*/
strcpy(str,"data/Tutorial/");
strcat(str,filename);
if((fp = fopen(str,"w"))==NULL)
{
    printf("Error opening file\n");
    exit(1);
}

fprintf(fp,"Epoch\tTrial\tTime\tX Position\tY Position\n");
while ( current->nextCollision !=NULL)
{
    realtime = current->collision.time.secs+(current->collision.time.uSecs/1000000000.0);

    fprintf(fp,"%3d\t", (current->epoch)-1);
    fprintf(fp,"%4.d\t", current->counter);
    fprintf(fp,"%f\t", realtime);
    fprintf(fp,"%f\t", current->collision.point[0]);
    fprintf(fp,"%f\n", current->collision.point[2]);
if (current->counter == MAXTRIALS)
{
    fprintf(fp, "n");
    fprintf(fp, "Plate Width\tPlate Amplitude\t(A/W)\n")
    fprintf(fp, "%f\t%f\t%f\n", current->targetWidth, current->targetAmplitude, ((current->targetAmplitude)/current->targetWidth)+.5);
    fprintf(fp, "n");
    current= current->nextCollision;
}

fprintf(fp, "%s\n", filename);
fclose(fp);
exit(1);

void rplateprocesscollisions(VCCollision_CallbackData *cd, void *data)
{
    /* Right plate collision callback handler */

    VCCollisionReportData *report;
    VCAtribute *hitNoise=(VCAtribute *)data;
    struct timespec starttime;

    clock_gettime(CLOCK_REALTIME,&starttime);
    report = VCCollision_GetFirstCollisionReport(cd->collision,NULL);
    if(report == NULL)
    {
        /* Restore to normal visual stimuli here as probe uncollides ( NULL report) */
        VCVvisual_SetGeometry(ProbeVis, "geometry/Exp1Fitts12");
        VCVvisual_SetGeometry(RPlatevis,"geometry/Exp1Fitts6");
    }
    else
    {
        if(report->direction[1]==-1)
        {
            /* Stimili presented here as correct ( unit vector in -Z direction) collision takes place */
            VCAudio_Start(hitNoise);
            VCVvisual_SetGeometry(ProbeVis,"geometry/Exp1Fitts13");
            VCVvisual_SetGeometry(RPlateVis,"geometry/Exp1Fitts9");
        }
        current->collision = *report;
    }

Simon Nee 24/05/2002
Behavioural Morphisms In Virtual Environments

```c
trialCounter = trialCounter + i;
current -> counter = trialCounter;
current -> epoch = epochCounter;

#if DEBUG >= 2

/*
This get printed out BEFORE we change the current pointer Otherwise
current points at the empty next record and you print out GIGO
*/

    printf("##RPlate Collision##\n");
    printf("**Data in report**\n");
    printf("Collision %d\n", current -> counter);
    printf("Flags %d\n", report -> flags);
    printf("Points %f %f %f\n", report -> point[0], report -> point[1], report -> point[2]);
    printf("Direction %f %f %f\n", report -> direction[0], report -> direction[1], report -> direction[2]);
    printf("%d %d\n", report -> time.secs);
    printf("%u\n", report -> time.uSecs);
    printf("**Data in List**\n");
    printf("RPlate Collision\n");
    printf("Collision %d\n", current -> counter);
    printf("Flags %d\n", current -> collision.flags);
    printf("Points %f %f %f\n", current -> collision.point[0], current -> collision.point[1], current -> collision.point[2]);
    printf("Direction %f %f %f\n", current -> collision.direction[0], current -> collision.direction[1], current -> collision.direction[2]);
    printf("Rplate at %u.%u\n\n", starttime.tv_sec, starttime.tv_nsec);
#endif

current -> targetAmplitude = plateamplitude;
current -> targetWidth = platewidth;
current -> plate = 1;

    current -> collision.time.secs = starttime.tv_sec;
current -> collision.time.uSecs = starttime.tv_nsec;
current -> nextCollision = (struct collisionList *) malloc(sizeof(struct collisionList));
current -> nextCollision -> previousCollision = current;
current -> nextCollision -> counter = current -> counter;
current = current -> nextCollision;
}

else
{
    errors = errors + 1;
}

if (trialCounter == MAXTRIALS && epochCounter == MAXEPOCHS + 1)
    OnMaxTrials();
if (trialCounter == MAXTRIALS && epochCounter != MAXEPOCHS + 1)
    adjPlates();
```

Simon Nee 24/05/2002 235
void lplateprocessCollisions(VCCollision_CallbackData *cd, void *data)
{
    /*
    Left plate collision callback handler
    */
    VCCollisionReportData *report;
    VCAtribute *hitNoise=(VCAtribute *)data;
    struct timespec starttime;

    clock_gettime(CLOCK_REALTIME, &starttime);
    report = VCCollision_GetFirstCollisionReport(cd->collision, NULL);

    if(report == NULL)
    {
        /* Restore to normal visual stimuli here as probe uncollides ( NULL report */
        VCVVisual_SetGeometry(ProbeVis, "geometry/Exp1Fitts12");
        VCVVisual_SetGeometry(LPlateVis, "geometry/Exp1Fitts5");
    }
    else
    {
        if (report->direction[1]==-1)
        {
            /* Stimuli presented here as correct ( unit vector in -Z direction) collision takes place */
            VCAudio_Start(hitNoise);
            VCVVisual_SetGeometry(ProbeVis, "geometry/Exp1Fitts13");
            VCVVisual_SetGeometry(LPlateVis, "geometry/Exp1Fitts8");

            current->collision = *report;
            trialCounter=trialCounter++;  
            current->counter=trialCounter;
            current->epoch=epochCounter;

            #if DEBUG >= 2
            /*
            This get printed out BEFORE we change the current pointer Otherwise current
            points at the empty next record and you print out GIGO
            */
        
    //Simon Nee 24/05/2002
    //236
Behavioural Morphisms In Virtual Environments

printf("##RPlate Collision##\n");
printf("**Data in report**\n");
printf("Collision %d \n",current->counter);
printf("Flags %d\n",report->flags);
printf("Points %f %f %f\n",report->point[0],report->point[1],report->point[2]);
printf("Direction %f %f %f\n",report->direction[0],report->direction[1],report->direction[2]);
printf("Secs %u \n",report->time.secs);
printf("Msecs %u \n",report->time.usecs);
printf("**Data in List**\n");
printf("RPlate Collision\n");
printf("Collision %d \n",current->counter);
printf("Flags %d\n",current->collision.flags);
printf("Points %f %f %f\n",current->collision.point[0],current->collision.point[1],current->collision.point[2]);
printf("Direction %f %f %f\n",current->collision.direction[0],current->collision.direction[1],current->collision.direction[2]);
printf("Lplate at %u.%u\n",starttime.tv_sec,starttime.tv_nsec);
#if defined
current->targetAmplitude=plateamplitude;
current->targetWidth=platewidth;
current->plate=2;
current->collision.time.secs=starttime.tv_sec;
current->collision.time.usecs=starttime.tv_nsec;
current->nextCollision=(struct collisionList *)malloc(sizeof(struct collisionList));
current->nextCollision->previousCollision=current;
current->nextCollision->counter=current->counter;
current = current->nextCollision;
#endif
else
{
    errors=errors+1;
}
}
if (trialCounter == MAXTRIALS && epochCounter == MAXEPOCHS+1 )
    OnMaxTrials();
if (trialCounter == MAXTRIALS && epochCounter != MAXEPOCHS+1)
    adjPlates();

/*##################################################################*/
void processBodyparts(VCBodyPartCreate_CallbackData *partData, void *data)
{
    /*
    This grabs hold of the hand so we can pick the probe up via the data parameter
    */
    char *partName;
VCBodyPartGetName(partData->bodyPart, &partName);

if (strcmp(partName, "hand") == 0)
    VCEntity_Pick((VCEntity *)data, partData->bodyPart, NULL, NULL);
/*
Grab hold of hand when it is created by searching each creat callback
When a body part is created it generated this callback.
*/
}

void positionBody(VCBodyCreate_CallbackData *cd, void *data)
{
    /*
    Function positioning of body in front of table
    */
    dmPoint position = {
        0.0, 1.0, 0.9
    };
    dmRuler orientation = {
        0.0, 0.0
    };

    VCBODY_SetPosition(cd->body, NULL, position, orientation, NULL, NULL, NULL);
    VCBODY_AttachBodyPartCreateCallback(cd->body, processBodyParts, data);
}

/*####################################################
##############*/
int main (int argc, char **argv)
{
    ActorId actor;
    VCEntity *Walls,*Ceiling,*Floor;
    VCEntity *TableTop,*Legs;
    VCEntity *HotSpot;
    VCEntity *light1,*light2,*noise;
    VCCollision *rplatecollision,*lplatecollision;
    VCAbstract *rplateboundary,*lplateboundary,*audio;
    static VCColour ambient = {
        0.35, 0.35, 0.35
    };
    static VCColour directional = {
        0.9, 0.9, 0.9
    };

    /*Initialize timer */

    /*Initialise linked list and pointers.*/
    head = current = (struct collisionList *)malloc(sizeof(struct collisionList));
    head->counter= trialCounter = 0; /* Ensure counter zeroed at start */
    head->epoch=epochCounter;
    #if DEBUG >= 3

Simon Nee 24/05/2002

238
/*
Debug output for checking initalization of list
*/
printf(" head->counter %d\n",head->counter);
printf(" head %d current %d\n", head, current);
#endif

actor=VC_InitApplication(&argc, argv, NULL);
if (actor != VC_OK)
exit(1);

/*
This is where the environment is created
*/
light1=VCEntity_create(NULL, 0);
light2=VCEntity_Create(NULL, 0);
VCEntity_AddLightAmbient(light1, ambient);
VCEntity_AddLightDirectional(light2, directional );
VCEntity_RotateX(light2, dmDegToRad(-45));

Walls = VC_ConstructVisualGeometry("geometry/Exp1Fitts1",0,NULL,NULL);
Ceiling = VC_ConstructVisualGeometry("geometry/Exp1Fitts2",0,NULL,NULL);
Floor = VC_ConstructVisualGeometry("geometry/Exp1Fitts3",0,NULL,NULL);
TableTop = VC_ConstructVisualGeometry("geometry/Exp1Fitts7",0,NULL,NULL);
VCEntity_Scale(TableTop,0.1,0.1,0.1);

Legs = VC_ConstructVisualGeometry("geometry/Exp1Fitts4",0,NULL,NULL);
VCEntity_Scale(Legs,0.1,0.1,0.1);

LPlate = VCEntity_Create(NULL,0);
LPlateVis=VCVisual_Create("geometry/Exp1Fitts5",NULL,VC_VISUAL_ENABLE,NULL,NULL,0);
VCEntity_AttachAttribute(LPlate,LPlateVis);
lplateboundary = VCEntity_AddBoundaryGeometry(LPlate,"geometry/Exp1Fitts10");
VCBoundary_GetCollision(lplateboundary,&lplatecollision);
VCBoundary_ModifyMode(lplateboundary,VC_COLLISION_ENABLE,VC_COLLISION_NO_POSITION);

RPlate = VCEntity_Create(NULL,0);
RPlateVis=VCVisual_Create("geometry/Exp1Fitts6",NULL,VC_VISUAL_ENABLE,NULL,NULL,0);
VCEntity_AttachAttribute(RPlate,RPlateVis);
rplateboundary = VCEntity_AddBoundaryGeometry(RPlate,"geometry/Exp1Fitts11");
VCBoundary_GetCollision(rplateboundary,&rplatecollision);
VCBoundary_ModifyMode(rplateboundary,VC_COLLISION_ENABLE,VC_COLLISION_NO_POSITION);
/* Set start Width and amplitude here case 1 epoch */
    adjPlates();
/* name file from second command line parameter */
    filename = argv[1];

#if DEBUG >=3
    printf("Lplate Collision Flags %u\n", lplateboundary->mode);
    printf("Rplate Collision Flags %u\n", rplateboundary->mode);
    printf(" Filename: %s.\n", argv[1]);
#endif

Probe = VCEntity_Create(NULL, 0);
ProbeVis = VCVisual_Create("geometry/Exp1Fitts12", NULL, VC_VISUAL_ENABLE, NULL, NULL, NULL, 0);
    VCEntity_AttachAttribute(Probe, ProbeVis);
    VCEntity_AddBoundaryGeometry(probe,"geometry/Exp1Fitts12");
    VCEntity_Translate(Probe, 0, 0.17, 0.9);

noise = VC_ConstructAudioVoice("pop", &audio);
    VC_AttachBodyCreateCallback(positionBody, Probe);
    VCCollision_AttachUpdateCallback(rplatecollision, rplateprocessCollisions, (void *)audio);
    VCCollision_AttachUpdateCallback(lplatecollision, lplateprocessCollisions, (void *)audio);

    VC_MainLoop();
}
# Behavioural Morphisms In Virtual Environments

## 13 Appendix 4 Rotary Pursuit Environment Code

```c
#include <dvs/vc.h>
#include <stdio.h>
#include <string.h>
#include "include/col.h"
/* Collision List data structure */
#include "include/settings.h"

#define TRIALS 20
#define BLOCKS 8
#define WARNMAX 5
#define NUMRAND 2000 /* Number of random swaps in block trial array */

/* These are global data structures for experimental data, it’s simpler this way. */
struct collisionList *head,*current;
char *filename;
int trialCounter=0,blockCounter=0,errors=0;
int position[BLOCKS][TRIALS];
int WaitingForReaction=0;
int touchButtonFlag;

/* Global geometry stuff so call backs can reach them easily */

/*Objects*/
VCEntity *Button,*HoldButtonRight,*HoldButtonLeft,*Stimulus,*plate[25],*plateOlder;

VCEntity *collisionNoise;
/*Visuals*/
VCAttribute *ButtonVis,*HoldButtonRightVis,*HoldButtonLeftVis,*StimulusVis,*plateVis[25];
/*Boundaries*/
VCAttribute *ButtonBoundary,*HoldButtonRightBoundary,*HoldButtonLeftBoundary;

VCAttribute *click;
VCColor white={1,1,1};

void End()
{
/*
 This function outputs the data in the linked list to a file
*/
    FILE *fp;
    char str[80];
    int epochvalue;
    double secstart,secstop,nanostart,nanostop;
```
current = head;
/*Calculate durations involves transforming from long ints to double floats*/

while (current->nextCollision != NULL)
{
    secstart=(current->start.tv_sec);
    secstop=(current->stop.tv_sec);
    nanostart=(current->start.tv_nsec);
    nanostop=(current->stop.tv_nsec);
    nanostart=(nanostart/1000000000);
    nanostop=(nanostop/1000000000);
    secstart=secstart+nanostart;
    secstop=secstop+nanostop;
    current->duration=secstop-secstart;
    secstart=(current->start.tv_sec);
    secstop=(current->lrstop.tv_sec);
    nanostart=(current->start.tv_nsec);
    nanostop=(current->lrstop.tv_nsec);
    nanostart=(nanostart/1000000000);
    nanostop=(nanostop/1000000000);
    secstart=secstart+nanostart;
    secstop=secstop+nanostop;
    current->lrduration=secstop-secstart;
}
/*Repoint to head of list to output data*/
current = head;

/*This is where the data is put into the file
 copy directory path and append filename from command line*/

strcpy(str,"data/read/");
strcat(str, filename);

if((fp = fopen(str,"w")) == NULL)
{
    printf("Error opening file\n");
    exit(1);
}

fprintf(fp,"Block\tTrial\tStartSec\tStartNan\tStopSec\tStopNan\n	Duration\tCharacter\tArray Position\n");

while (current->nextCollision != NULL)
{
    fprintf(fp, "%d@%d\t", current->Block, current->Trial);
    fprintf(fp, "%u.%u\t", current->start.tv_sec, current-
>start.tv_nsec);
    fprintf(fp, "%u.%u\t", current->lrstop.tv_sec, current-
>lrstop.tv_nsec);
    fprintf(fp, "%u.%u\t", current->stop.tv_sec, current-
>stop.tv_nsec);
    fprintf(fp, "%f\t", current->lrduration);
}
fprintf(fp, "%f\t", current->duration);
fprintf(fp, "%c\t", current->searchCharacter);
fprintf(fp, "%d\t", current->position);
fprintf(fp, "%-26.25s\n", current->characterSet);
current = current->nextCollision;
}

fclose(fp);
VCVisual_SetFrontMaterial(ButtonVis, "Room:Orange");
VCVisual_SetFrontMaterial(HoldButtonRightVis, "Room:Orange");
VCVisual_SetFrontMaterial(HoldButtonLeftVis, "Room:Orange");

void fillArray()
{
    int x, y;
    srand( (unsigned) time( NULL ) );
    for (x=0;x<BLOCKS;x++)
    {
        for (y=0;y<TRIALS;y++)
        {
            position[x][y] = x + 1;
        }
    }
}

void randomizeArray()
{
    int x, y, temp_swap, pos1x, pos2x, pos1y, pos2y;
    /* Randomize array contents */
    for (x=0; x<NUMRAND; x++)
    {
        pos1x = (int) (rand() % BLOCKS);
        pos1y = (int) (rand() % TRIALS);
        pos2x = (int) (rand() % BLOCKS);
        pos2y = (int) (rand() % TRIALS);
        temp_swap = position[pos1x][pos1y];
        position[pos1x][pos1y] = position[pos2x][pos2y];
        position[pos2x][pos2y] = temp_swap;
    }
}

void checkForDataOrExit()
{
    current->nextCollision = (struct collisionList *)malloc(sizeof(struct collisionList));
    current->nextCollision->previousCollision = current;
current = current->nextCollision;

if((blockCounter+1==BLOCKS)&&(trialCounter+1==TRIALS))
{
    printf("Experiment Completed\n");
    End();
}
else
{
    if(trialCounter+1==TRIALS)
    {
        trialCounter=0;
        blockCounter=blockCounter+1;
        VCVVisual_SetFrontMaterial(ButtonVis,
        "Room:Orange");
        sleep(60);
        VCVVisual_SetFrontMaterial(ButtonVis, "Room:Green");
    }
    else
    {
        trialCounter=trialCounter+1;
    }
}

/*----------------------------------------------------------------------------------------*/
/* void setTrial(int blockNumber) */
/* This places numbers on a 5x5 board in random order. */
/* It then grabs a number from a specific position and stores it */

int pos,ReactChar;
char FindChar,material[25];
char name[9]="Letters:";

for(pos=0;pos<25;pos++)
{
    material[pos]= (char)((rand()%26)+65);
    name[8]=material[pos];
    VCVVisual_SetFrontMaterial(plateVis[pos],name);
    if ((pos+1)==blockNumber)
    {
        FindChar=material[pos];
        VCVVisual_SetFrontMaterial(StimulusVis,name);
    }
}

/* This section checks that no other occurrences of the chosen number and replaces them if there are*/

for(pos=0;pos<25;pos++)
{
name[8]=material[pos];
if(pos+1!=blockNumber)
{
    while(material[pos]==FindChar)
    {
        material[pos]=(char)((rand()%26)+65);
        name[8]=material[pos];
    }
    VCVI_sual_setFrontMaterial(plateVis[pos],name);
}
printf("%s ",name);
printf("%d",blockNumber);
if((pos+1)%5==0)
printf("\n");
current->Position=blockNumber;
current->Trial=trialCounter+1;
current->Block=blockCounter+1;
for(pos=0;pos<25;pos++)
{
    current->characterSet[pos]=material[pos];
}
current->searchCharacter=FindChar;

/* The above should provide a 5x5 set of numbers that can be mapped
onto the alphabet (0-25,26)*/
/*The numbers are random and will contain duplicates there will be
only one occurrence of the chosen*/
/*Assign material to Subject notice.*/
/*Block with Subject Notice*/
}

void doTrial()
{
    struct timespec startTime;
    VCEntity_Translate(Stimulus,-20.0,-20.0,-20.0);
    sleep(3);
    VCEntity_Translate(Stimulus,20.0,20.0,20.0);
    clock_gettime(CLOCK_REALTIME,&startTime);
    current->start.tv_sec=startTime.tv_sec;
    current->start.tv_nsec=startTime.tv_nsec;
    WaitingForReaction=1;
}

/* The above should provide a 5x5 set of numbers that can be mapped
onto the alphabet (0-25,26)*/
/*The numbers are random and will contain duplicates there will be
only one occurrence of the chosen*/
/*Assign material to Subject notice.*/
/*Block with Subject Notice*/

void getTrialSetup()
{
    printf("Block: %d Trial: %d Condition:
%d\n",blockCounter+1,trialCounter+1,position[blockCounter][trialCounter]);
switch(position[blockCounter][trialCounter])
{
    case 1:
        setTrial(1);
        break;
    case 2:
        setTrial(2);
        break;
    case 3:
        setTrial(6);
        break;
    case 4:
        setTrial(10);
        break;
    case 5:
        setTrial(14);
        break;
    case 6:
        setTrial(18);
        break;
    case 7:
        setTrial(22);
        break;
    case 8:
        setTrial(25);
        break;
    default:
        printf("An error occurred....\n");
        End();
}

void startTrials(VCBodyInput_CallbackData *cd, void *data)
{
    printf("Start Trials\n");
    if(touchButtonFlag==1)
    {
        getTrialSetup();
        doTrial();
    }
    else
    {
        printf("User is not touching the start position button\n");
    }
}

void processBodyParts(VCBodyPartCreate_CallbackData *partData, void *data)
{
    /*
    This grabs hold of the hand so we can pick the probe up via the data parameter
    */
    char *partName;
VCBodyPart_GetName(partData->bodyPart, &partName);
printf("\n%s\n", partName);

if (strcmp(partName, "rightHand") == 0)
{
    /*
    VCEntity_Pick((VCEntity *)data, partData->bodyPart, NULL, NULL);
    */
}

/*
Grab hold of hand when it is created by searching each create
callback
When a body part is created it generated this callback.
*/

void positionBody(VCBodyscale_CallbackData *cd, void *data)
{
    /*
    Function positioning of body in front of table
    */
    dmPoint position = {0.0, 1.1, .834};
    dmEuler orientation = {-0.174, 0, 0};
    dmScale scale = {1, 1, 1};
    printf("\nGot here! \n\n");
    VCBodyscale_SetPosition(cd->body, NULL, position, orientation, scale, NULL, 0);
    VCBodyscale_AttachBodyPartCreateCallback(cd->body, processBodyParts, data);
}

void touch(VCBodyscaleAppCallbackData *cd, void *data)
{
    struct timespec stopTime;
    int counter;

    VCAudio_Start(click);
    printf("Touched \n");

    if (cd->entity==Button)
    {
        VCBodyscale_SetFrontMaterial(ButtonVis, "Room:Red");
        if(WaitingForReaction==1)
        {
            clock_gettime(CLOCK_REALTIME, &stopTime);
            current->stop.tv_sec=stopTime.tv_sec;
            current->stop.tv_nsec=stopTime.tv_nsec;
            WaitingForReaction=0;
            /*Remove Stimulus*/
            for(counter=0;counter<25;counter++)
            {
                VCBodyscale_SetFrontMaterial(plateVis[counter], "Room:Red");
            }
        }
    }
}

Simon Nee 24/05/2002
Behavioural Morphisms In Virtual Environments

checkForDataOrExit();

if (cd->entity==HoldButtonRight)
{
    VCVisual_SetFrontMaterial(HoldButtonRightVis, "Room:Red");
touchButtonFlag=1;
}

if (cd->entity==HoldButtonLeft)
{
    VCVisual_SetFrontMaterial(HoldButtonLeftVis, "Room:Red");
touchButtonFlag=1;
}

void untouch(VCBodyAppCallbackData *cd, void *data)
{
    struct timespec stopTime;
    VCAudio_Start(click);
    if (cd->entity==Button)
    {
        VCVisual_SetFrontMaterial(ButtonVis, "Room:Green");
    }
    if (cd->entity==HoldButtonRight)
    {
        if(WaitingForReaction==1)
        {
            clock_gettime(CLOCK_REALTIME,&stopTime);
            current->lrstop.tv_sec=stopTime.tv_sec;
            current->lrstop.tv_nsec=stopTime.tv_nsec;
        }
        VCVisual_SetFrontMaterial(HoldButtonRightVis, "Room:Green");
touchButtonFlag=0;
    }
    if (cd->entity==HoldButtonLeft)
    {
        if(WaitingForReaction==1)
        {
            clock_gettime(CLOCK_REALTIME,&stopTime);
            current->lrstop.tv_sec=stopTime.tv_sec;
            current->lrstop.tv_nsec=stopTime.tv_nsec;
        }
        VCVisual_SetFrontMaterial(HoldButtonLeftVis, "Room:Green");
touchButtonFlag=0;
    }

/*##################################################################*/
int main (int argc, char **argv)
{
    ActorId actor;
    VCEntity *Walls,*Ceiling,*Floor,*TableTop,*Legs,*light1,*light2,*light3,*collisionNoise;
    VCCollision *ButtonType, *HoldButtonRightCollision, *HoldButtonLeftCollision;
    VCAttribute *pop;
    static VCColour ambient = {0.45, 0.45, 0.45};
    static VCColour directional = {0.9, 0.9, 0.9};
    int x;

    /* Fill array*/
    fillArray();
    randomizeArray();

    /*Initialize timer */
    /*Initialise linked list and pointers.*/
    head = current = (struct collisionList *)malloc(sizeof(struct collisionList));

    #if DEBUG >= 3
    /*
    Debug output for checking initalization of list
    */
    printf("head->counter \d\n",head->counter);
    printf("head %d current %d\n",head,current);
    #endif
    actor=VC_InitApplication(&argc, argv, NULL);

    if (actor != VC_OK)
        exit (1);

    /*
    This is where the environment is created
    */
    light1=VCEntity_Create(NULL, 0);
    light2=VCEntity_Create(NULL, 0);
    light3=VCEntity_Create(NULL, 0);
    VCEntity_AddLightAmbient(light1,ambient);
    VCEntity_AddLightPoint(light2,directional );
    VCEntity_Translate(light2,0,2,0);

    Walls = VC_ConstructVisualGeometry("geometry/Walls",0,0,0,0);,
    Ceiling = VC_ConstructVisualGeometry("geometry/Ceiling",0,0,0,0);,
    Floor = VC_ConstructVisualGeometry("geometry/Floor",0,0,0,0);,
Behavioural Morphisms In Virtual Environments

TableTop =
VC_ConstructVisualGeometry("geometry/TableTop", 0, NULL, NULL);
VCEntity_Scale(TableTop, 0.1, 0.1, 0.1);
Legs = VC_ConstructVisualGeometry("geometry/Legs", 0, NULL, NULL);
VCEntity_Scale(Legs, 0.1, 0.1, 0.1);

for(x=0;x<25;x++)
{
  plate[x] = VCEntity_Create(NULL, 0);
  plateVis[x] =
  VCVisual_Create("geometry/Button", NULL, VC_VISUAL_ENABLE, NULL, NULL, NULL, 0);
  VCEntity_AttachAttribute(plate[x], plateVis[x]);
  VCEntity_RotateX(plate[x], 3.142);
  VCEntity_Translate(plate[x], ((-40) + ((x % S) * 20)), 0, ((-40) + ((x / 5) * 20)));
  VCEntity_Scale(plate[x], 0.005, 0.001, 0.005);
  VCEntity_RotateX(plate[x], 1);
  VCEntity_Translate(plate[x], 0.0, 0.94, -0.3);
  printf("%d", x);
}

Button = VCEntity_Create(NULL, 0);
ButtonVis=
VCVisual_Create("geometry/Button", NULL, VC_VISUAL_ENABLE, NULL, NULL, NULL, 0);
VCEntity_AttachAttribute(Button, ButtonVis);
ButtonBoundary =
VCEntity_AddBoundaryGeometry(Button, "geometry/Button");
VCBoundary_GetCollision(ButtonBoundary, &ButtonCollision);
VCBoundary_ModifyMode(ButtonBoundary, VC_COLLISION_ENABLE, VC_COLLISION_NO_POSITION);
VCEntity_Scale(Button, 0.005, 0.001, 0.005);
VCEntity_RotateX(Button, 3.142);
VCEntity_Translate(Button, 0.0, 0.75, 0.15);
VCVisual_SetFrontMaterial(ButtonVis, "Room:Green");

HoldButtonRight = VCEntity_Create(NULL, 0);
HoldButtonRightVis=
VCVisual_Create("geometry/Button", NULL, VC_VISUAL_ENABLE, NULL, NULL, NULL, 0);
VCEntity_AttachAttribute(HoldButtonRight, HoldButtonRightVis);
HoldButtonRightBoundary =
VCEntity_AddBoundaryGeometry(HoldButtonRight, "geometry/Button");
VCBoundary_GetCollision(HoldButtonRightBoundary, &HoldButtonRightCollision);
VCBoundary_ModifyMode(HoldButtonRightBoundary, VC_COLLISION_ENABLE, VC_COLLISION_NO_POSITION);
VCEntity_Scale(HoldButtonRight, 0.005, 0.001, 0.005);
VCEntity_RotateX(HoldButtonRight, 3.142);
VCEntity_Translate(HoldButtonRight, 0.3, 0.75, 0.15);
VCVisual_SetFrontMaterial(HoldButtonRightVis, "Room:Green");

HoldButtonLeft = VCEntity_Create(NULL, 0);
HoldButtonLeftVis=
VCVisual_Create("geometry/Button", NULL, VC_VISUAL_ENABLE, NULL, NULL, NULL, 0);
VCEntity_AttachAttribute(HoldButtonLeft,HoldButtonLeftVis);
HoldButtonLeftBoundary =
VCEntity_AddBoundaryGeometry(HoldButtonLeft,"geometry/Button");
VCBoundary_GetCollision(HoldButtonLeftBoundary,&HoldButtonLeftCollision);
VCBoundary_ModifyMode(HoldButtonLeftBoundary,VC_COLLISION_ENABLE,VC_COLLISION_NO_POSITION);
VCEntity_Scale(HoldButtonLeft,0.005,0.001,0.005);
VCEntity_RotateX(HoldButtonLeft,3.142);
VCEntity_Translate(HoldButtonLeft,-0.3,0.75,0.15);
VCVisual_SetFrontMaterial(HoldButtonLeftVis,"Room:Green");

Stimulus = VCEntity_Create(NULL,0);
StimulusVis =
VCVisual_Create("geometry/Button",NULL,VC_VISUAL_ENABLE,NULL,NULL,NULL,0);
VCEntity_AttachAttribute(Stimulus,StimulusVis);
VCEntity_Scale(Stimulus,0.025,0.002,0.025);
VCEntity_RotateX(Stimulus,4.142);
VCEntity_Translate(Stimulus,0.0,0.95,-0.25);
VCEntity_Translate(Stimulus,20.0,20.0,20.0); /* Move so not visible*/

/* Set start Width and amplitude here: case 1 epoch */
/* name file from second command line parameter */

filename = argv[1];
collisionNoise = VC_ConstructAudioVoice("down", &click);
VC_AttachBodyCreateCallback(positionBody,NULL);
VCBody_AttachInputCallback(NULL,NULL,(VC_INPUT_PRESS |
't'),0,startTrials,NULL);
VCBody_AttachInteractionCallbacks(NULL,touch,untouch,NULL,NULL,NULL,NULL);

VC_MainLoop();
}
### Appendix 5: Fitts Law Raw Data Output

<table>
<thead>
<tr>
<th>Epoch</th>
<th>Trial</th>
<th>Time</th>
<th>X Position</th>
<th>Y Position</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>889136960.576318</td>
<td>0.051424</td>
<td>0.202274</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>889136961.715054</td>
<td>-0.034732</td>
<td>0.195365</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>889136962.653957</td>
<td>0.054171</td>
<td>0.183774</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>889136963.553365</td>
<td>-0.042960</td>
<td>0.193151</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>889136964.392203</td>
<td>0.049266</td>
<td>0.188539</td>
</tr>
<tr>
<td>1</td>
<td>6</td>
<td>889136965.391812</td>
<td>-0.044120</td>
<td>0.204430</td>
</tr>
<tr>
<td>1</td>
<td>7</td>
<td>889136966.181273</td>
<td>0.045434</td>
<td>0.197578</td>
</tr>
<tr>
<td>1</td>
<td>8</td>
<td>889136967.121018</td>
<td>-0.041817</td>
<td>0.211046</td>
</tr>
<tr>
<td>1</td>
<td>9</td>
<td>889136967.920889</td>
<td>0.053920</td>
<td>0.198977</td>
</tr>
<tr>
<td>1</td>
<td>10</td>
<td>889136968.719944</td>
<td>-0.046706</td>
<td>0.203381</td>
</tr>
<tr>
<td>1</td>
<td>11</td>
<td>889136969.459563</td>
<td>0.039498</td>
<td>0.193176</td>
</tr>
<tr>
<td>1</td>
<td>12</td>
<td>889136970.278963</td>
<td>-0.048554</td>
<td>0.205529</td>
</tr>
<tr>
<td>1</td>
<td>13</td>
<td>889136970.978824</td>
<td>0.042248</td>
<td>0.186207</td>
</tr>
<tr>
<td>1</td>
<td>14</td>
<td>889136971.927431</td>
<td>-0.040947</td>
<td>0.196941</td>
</tr>
<tr>
<td>1</td>
<td>15</td>
<td>889136972.657218</td>
<td>0.049022</td>
<td>0.184761</td>
</tr>
<tr>
<td>1</td>
<td>16</td>
<td>889136973.547105</td>
<td>-0.047877</td>
<td>0.201429</td>
</tr>
<tr>
<td>1</td>
<td>17</td>
<td>889136974.315944</td>
<td>0.047624</td>
<td>0.187443</td>
</tr>
<tr>
<td>1</td>
<td>18</td>
<td>889136975.106909</td>
<td>-0.036702</td>
<td>0.194805</td>
</tr>
<tr>
<td>1</td>
<td>19</td>
<td>889136975.884603</td>
<td>0.044087</td>
<td>0.184756</td>
</tr>
<tr>
<td>1</td>
<td>20</td>
<td>889136976.654171</td>
<td>-0.046406</td>
<td>0.194440</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>889136978.414024</td>
<td>0.039676</td>
<td>0.177238</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>889136979.106140</td>
<td>-0.046365</td>
<td>0.197160</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>889136979.851993</td>
<td>0.043491</td>
<td>0.186800</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>889136980.644049</td>
<td>-0.036633</td>
<td>0.194032</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>889136981.413390</td>
<td>0.049286</td>
<td>0.187199</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>889136982.182172</td>
<td>-0.035262</td>
<td>0.196845</td>
</tr>
<tr>
<td>2</td>
<td>7</td>
<td>889136983.079941</td>
<td>0.046977</td>
<td>0.188780</td>
</tr>
<tr>
<td>2</td>
<td>8</td>
<td>889136984.841938</td>
<td>-0.037582</td>
<td>0.193203</td>
</tr>
<tr>
<td>2</td>
<td>9</td>
<td>889136984.641372</td>
<td>0.035411</td>
<td>0.187306</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>889136985.380736</td>
<td>-0.041553</td>
<td>0.196698</td>
</tr>
<tr>
<td>2</td>
<td>11</td>
<td>889136986.108347</td>
<td>0.050355</td>
<td>0.189323</td>
</tr>
<tr>
<td>2</td>
<td>12</td>
<td>889136986.957149</td>
<td>-0.042027</td>
<td>0.200786</td>
</tr>
<tr>
<td>2</td>
<td>13</td>
<td>889136987.679014</td>
<td>0.045746</td>
<td>0.187401</td>
</tr>
<tr>
<td>2</td>
<td>14</td>
<td>889136988.471916</td>
<td>-0.043894</td>
<td>0.198955</td>
</tr>
<tr>
<td>2</td>
<td>15</td>
<td>889136989.155671</td>
<td>0.040669</td>
<td>0.182089</td>
</tr>
</tbody>
</table>

Plate Width Plate Amplitude (2A/W)

<table>
<thead>
<tr>
<th>Width</th>
<th>Amplitude</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.025000</td>
<td>0.087500</td>
</tr>
<tr>
<td>4.000000</td>
<td></td>
</tr>
</tbody>
</table>

Simon Nee 24/05/2002
### 15 Appendix 6 Choice Reaction Raw Data Output

<table>
<thead>
<tr>
<th>B.T.C</th>
<th>Start</th>
<th>React</th>
<th>Press</th>
<th>ReactDur</th>
<th>PressDur</th>
<th>SeaChar</th>
<th>PreChar</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Pos</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1@1@1</td>
<td>928873338.527494000</td>
<td>928873339.338395000</td>
<td></td>
<td></td>
<td></td>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>1@2@5</td>
<td>928873340.297429000</td>
<td>928873348.290193000</td>
<td></td>
<td></td>
<td></td>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>1@3@1</td>
<td>928873351.177252000</td>
<td>928873358.309443000</td>
<td></td>
<td></td>
<td></td>
<td>D</td>
<td>D</td>
</tr>
<tr>
<td>1@4@7</td>
<td>928873358.699418000</td>
<td>928873363.914245000</td>
<td></td>
<td></td>
<td></td>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>1@5@2</td>
<td>928873379.489401000</td>
<td>928873378.931459000</td>
<td></td>
<td></td>
<td></td>
<td>G</td>
<td>G</td>
</tr>
<tr>
<td>1@6@6</td>
<td>928873384.643620000</td>
<td>928873385.363516000</td>
<td></td>
<td></td>
<td></td>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>1@7@6</td>
<td>928873393.187692000</td>
<td>928873393.775183000</td>
<td></td>
<td></td>
<td></td>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>1@8@3</td>
<td>928873413.168188000</td>
<td>928873412.347420000</td>
<td></td>
<td></td>
<td></td>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>1@9@2</td>
<td>928873419.829021000</td>
<td>928873420.450244000</td>
<td></td>
<td></td>
<td></td>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>1@10@2</td>
<td>928873426.353477000</td>
<td>928873426.954761000</td>
<td></td>
<td></td>
<td></td>
<td>B</td>
<td>B</td>
</tr>
</tbody>
</table>

Simon Nee 24/05/2002
<table>
<thead>
<tr>
<th>Time</th>
<th>User</th>
<th>Start Time</th>
<th>End Time</th>
<th>Duration</th>
<th>Distance</th>
<th>Speed</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1@11@7</td>
<td>928873431.85990900</td>
<td>928873432.633524000</td>
<td>1.218083</td>
<td>B B</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1@12@3</td>
<td>928873439.409466000</td>
<td>928873440.181166000</td>
<td>0.771700</td>
<td>2.060493</td>
<td>C C</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>1@13@2</td>
<td>928873462.527818000</td>
<td>928873463.976670000</td>
<td>1.530299</td>
<td>A A</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1@14@8</td>
<td>928873468.956621000</td>
<td>928873469.982661000</td>
<td>0.771700</td>
<td>2.060493</td>
<td>C C</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>1@15@6</td>
<td>928873476.483459000</td>
<td>928873477.368285000</td>
<td>0.884826</td>
<td>1.167070</td>
<td>C C</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>1@16@3</td>
<td>928873482.669080000</td>
<td>928873483.528184000</td>
<td>0.859104</td>
<td>2.027702</td>
<td>C C</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>1@17@6</td>
<td>928873490.559808000</td>
<td>928873491.460537000</td>
<td>0.900729</td>
<td>1.409658</td>
<td>F F</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>1@18@5</td>
<td>928873496.703913000</td>
<td>928873497.494904000</td>
<td>0.790991</td>
<td>1.020390</td>
<td>C C</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>1@19@6</td>
<td>928873502.318930000</td>
<td>928873503.202410000</td>
<td>0.733191</td>
<td>1.131011</td>
<td>C C</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>1@20@2</td>
<td>928873508.532280000</td>
<td>928873509.112775000</td>
<td>0.580495</td>
<td>1.409875</td>
<td>A A</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2@1@4</td>
<td>928873576.336601000</td>
<td>928873577.697920000</td>
<td>0.733191</td>
<td>1.131011</td>
<td>C C</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>2@2@1</td>
<td>928873581.980959000</td>
<td>928873582.471696000</td>
<td>0.490737</td>
<td>1.031471</td>
<td>A A</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Behavioural Morphisms In Virtual Environments</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2@3@5  928873588.126541000  928873588.877243000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>928873589.995572000  0.750702  1.869031 A A 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2@4@5  928873594.361026000  928873595.896990000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>928873595.514264000  0.728673  1.153238 C C 3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2@5@1  928873600.662855000  928873601.144186000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>928873601.475866000  0.481331  0.813011 A A 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2@6@8  928873605.838883000  928873606.738821000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>928873607.628919000  0.899938  1.790036 G F 7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2@7@5  928873612.204872000  928873612.953085000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>928873613.873593000  0.748213  1.668721 A A 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Simon Nee 24/05/2002
Behavioural Morphisms In Virtual Environments

16 Appendix 7 Rotary Pursuit Raw Data Output
Target X Y Z, Tip X Y Z
0.139776,0.589664,0.600341,0.137329,0.598102,0.594512,
0.137992,0.579081,0.610961,0.137102,0.597950,0.594348,
0.136437,0.573871,0.616204,0.137511,0.598156,0.594618,
0.132031,0.563705,0.626462,0.137730,0.597353,0.594923,
0.129194,0.558785,0.631441,0.137105,0.595710,0.594868,
0.122305,0.549357,0.641013,0.137990,0.596643,0.596752,
0.118276,0.544883,0.645573,0.136367,0.594661,0.598389,
0.113876,0.540588,0.649959,0.136497,0.592628,0.599945,
0.104025,0.532600,0.658154,0.132777,0.587284,0.605540,
0.092891,0.525506,0.665481,0.129396,0.583636,0.609390,
0.086891,0.522326,0.668787,0.122963,0.574794,0.618059,
0.074130,0.516754,0.674626,0.121514,0.568889,0.623559,
0.067414,0.514382,0.677139,0.100874,0.554592,0.637942,
0.053428,0.510508,0.681306,0.077104,0.546265,0.645340,
0.046207,0.509020,0.682945,0.060221,0.541502,0.650098,
0.031436,0.506966,0.685309,0.036064,0.533047,0.659524,
0.023938,0.506408,0.686024,0.006296,0.523991,0.668270,
0.008847,0.506239,0.686509,-0.025523,0.518534,0.672951,
0.001308,0.506628,0.686278,-0.054551,0.517544,0.675288,
-0.013626,0.508349,0.684870,-0.066145,0.517304,0.675407,
-0.020969,0.509674,0.683698,-0.075549,0.517887,0.675381,
-0.035277,0.513231,0.680441,-0.094255,0.519856,0.673785,
-0.042193,0.515450,0.678367,-0.101947,0.521505,0.670939,
-0.055424,0.520730,0.673364,-0.103556,0.522444,0.669861,
-0.061694,0.523773,0.670453,-0.107594,0.527110,0.667368,
-0.073431,0.530611,0.663861,-0.108790,0.528745,0.665926,
-0.078858,0.534382,0.660204,-0.110786,0.533938,0.658886,
-0.088732,0.542562,0.652230,-0.111976,0.539574,0.653371,
-0.093145,0.546943,0.647943,-0.110211,0.542647,0.650125,
-0.100845,0.556207,0.638840,-0.106609,0.547959,0.644311,
-0.104105,0.561059,0.634056,-0.106964,0.551708,0.641707,
-0.109388,0.571116,0.624110,-0.105521,0.560214,0.632555,
-0.111392,0.576286,0.618982,-0.103959,0.563169,0.630687,
-0.114092,0.586819,0.608506,-0.102290,0.569845,0.624791,
-0.114778,0.592144,0.603194,-0.102388,0.571724,0.621653,
-0.114809,0.602821,0.592518,-0.102191,0.578993,0.616725,
-0.114155,0.608135,0.587190,-0.098306,0.587935,0.607298,
-0.111517,0.618620,0.576650,-0.098040,0.590968,0.604050,
-0.109542,0.623754,0.571475,-0.097362,0.597091,0.598609,
-0.104318,0.633717,0.561403,-0.093787,0.601378,0.594771,
-0.101086,0.638510,0.556542,-0.088566,0.610490,0.586723,
-0.093440,0.647636,0.547256,-0.082314,0.625540,0.573291,
-0.089053,0.651937,0.542863,-0.081232,0.629621,0.568645,
-0.079226,0.659940,0.534655,-0.075840,0.641097,0.559129,
-0.073821,0.663613,0.530868,-0.074196,0.645425,0.555142,
-0.062123,0.670240,0.523996,-0.069373,0.657287,0.544499,
-0.055871,0.673170,0.520935,-0.068613,0.662630,0.539504,
-0.042670,0.678212,0.515616,-0.056605,0.675242,0.527505,
-0.028695,0.682107,0.511429,-0.043477,0.681544,0.522377,
-0.021479,0.683606,0.509779,-0.035533,0.683418,0.520134,
-0.006714,0.6S5682,0.507394,-0.019503,0.685619,0.518867,
0.000782,0.686251,0.506668,-0.012713,0.684751,0.518281,
0.015871,0.686442,0.506161,0.001131,0.684582,0.517875,
0.023412,0.686063,0.506381,0.009698,0.685563,0.518357,
0.038351,0.684364,0.507767,0.026001,0.683245,0.519618,

I Simon Nee 24/05/2002


17 Appendix 8 Rotary Pursuit Raw Data Output

<table>
<thead>
<tr>
<th>5</th>
<th>3</th>
<th>01</th>
<th>10.62</th>
<th>-28.64</th>
<th>8.59</th>
<th>-22.11</th>
<th>-7.29</th>
<th>-0.89</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>5</td>
<td>01</td>
<td>10.64</td>
<td>-28.64</td>
<td>8.61</td>
<td>-21.95</td>
<td>-7.21</td>
<td>-0.82</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>01</td>
<td>10.58</td>
<td>-28.69</td>
<td>8.61</td>
<td>-22.08</td>
<td>-7.13</td>
<td>-0.68</td>
</tr>
<tr>
<td>5</td>
<td>7</td>
<td>01</td>
<td>10.59</td>
<td>-28.68</td>
<td>8.61</td>
<td>-22.01</td>
<td>-7.14</td>
<td>-0.71</td>
</tr>
<tr>
<td>5</td>
<td>8</td>
<td>01</td>
<td>10.59</td>
<td>-28.68</td>
<td>8.61</td>
<td>-22.12</td>
<td>-7.24</td>
<td>-0.83</td>
</tr>
<tr>
<td>5</td>
<td>9</td>
<td>01</td>
<td>10.62</td>
<td>-28.66</td>
<td>8.60</td>
<td>-22.05</td>
<td>-7.24</td>
<td>-1.00</td>
</tr>
<tr>
<td>5</td>
<td>10</td>
<td>01</td>
<td>10.62</td>
<td>-28.68</td>
<td>8.53</td>
<td>-22.11</td>
<td>-7.13</td>
<td>-1.08</td>
</tr>
<tr>
<td>5</td>
<td>11</td>
<td>01</td>
<td>10.62</td>
<td>-28.66</td>
<td>8.54</td>
<td>-22.13</td>
<td>-7.19</td>
<td>-1.23</td>
</tr>
<tr>
<td>5</td>
<td>11</td>
<td>01</td>
<td>10.59</td>
<td>-28.65</td>
<td>8.56</td>
<td>-22.24</td>
<td>-7.31</td>
<td>-1.27</td>
</tr>
<tr>
<td>5</td>
<td>12</td>
<td>01</td>
<td>10.59</td>
<td>-28.64</td>
<td>8.56</td>
<td>-22.23</td>
<td>-7.27</td>
<td>-1.36</td>
</tr>
<tr>
<td>5</td>
<td>13</td>
<td>01</td>
<td>10.60</td>
<td>-28.65</td>
<td>8.53</td>
<td>-22.25</td>
<td>-7.20</td>
<td>-1.26</td>
</tr>
<tr>
<td>5</td>
<td>14</td>
<td>01</td>
<td>10.64</td>
<td>-28.62</td>
<td>8.56</td>
<td>-22.13</td>
<td>-7.26</td>
<td>-1.43</td>
</tr>
<tr>
<td>5</td>
<td>15</td>
<td>01</td>
<td>10.60</td>
<td>-28.63</td>
<td>8.56</td>
<td>-22.28</td>
<td>-7.29</td>
<td>-1.43</td>
</tr>
<tr>
<td>5</td>
<td>16</td>
<td>01</td>
<td>10.61</td>
<td>-28.62</td>
<td>8.52</td>
<td>-22.22</td>
<td>-7.16</td>
<td>-1.34</td>
</tr>
<tr>
<td>5</td>
<td>18</td>
<td>01</td>
<td>10.60</td>
<td>-28.60</td>
<td>8.59</td>
<td>-22.28</td>
<td>-7.23</td>
<td>-1.41</td>
</tr>
<tr>
<td>5</td>
<td>19</td>
<td>01</td>
<td>10.62</td>
<td>-28.61</td>
<td>8.57</td>
<td>-22.21</td>
<td>-7.15</td>
<td>-1.33</td>
</tr>
<tr>
<td>5</td>
<td>20</td>
<td>01</td>
<td>10.62</td>
<td>-28.60</td>
<td>8.58</td>
<td>-22.16</td>
<td>-7.17</td>
<td>-1.39</td>
</tr>
<tr>
<td>5</td>
<td>21</td>
<td>01</td>
<td>10.57</td>
<td>-28.61</td>
<td>8.61</td>
<td>-22.29</td>
<td>-7.22</td>
<td>-1.45</td>
</tr>
<tr>
<td>5</td>
<td>22</td>
<td>01</td>
<td>10.58</td>
<td>-28.62</td>
<td>8.59</td>
<td>-22.31</td>
<td>-7.18</td>
<td>-1.32</td>
</tr>
<tr>
<td>5</td>
<td>23</td>
<td>01</td>
<td>10.63</td>
<td>-28.62</td>
<td>8.57</td>
<td>-22.10</td>
<td>-7.11</td>
<td>-1.46</td>
</tr>
<tr>
<td>5</td>
<td>24</td>
<td>01</td>
<td>10.66</td>
<td>-28.66</td>
<td>8.57</td>
<td>-22.09</td>
<td>-7.08</td>
<td>-1.25</td>
</tr>
<tr>
<td>5</td>
<td>25</td>
<td>01</td>
<td>10.74</td>
<td>-28.71</td>
<td>8.55</td>
<td>-22.07</td>
<td>-6.97</td>
<td>-1.02</td>
</tr>
</tbody>
</table>
## Behavioural Morphisms In Virtual Environments

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>26</td>
<td>01</td>
<td>10.87</td>
<td>-28.74</td>
<td>8.67</td>
<td>-22.01</td>
</tr>
<tr>
<td>5</td>
<td>26</td>
<td>01</td>
<td>11.03</td>
<td>-28.85</td>
<td>8.68</td>
<td>-22.04</td>
</tr>
<tr>
<td>5</td>
<td>27</td>
<td>01</td>
<td>11.34</td>
<td>-28.91</td>
<td>8.79</td>
<td>-21.80</td>
</tr>
<tr>
<td>5</td>
<td>28</td>
<td>01</td>
<td>11.69</td>
<td>-28.98</td>
<td>8.88</td>
<td>-21.48</td>
</tr>
<tr>
<td>5</td>
<td>29</td>
<td>01</td>
<td>12.01</td>
<td>-29.07</td>
<td>9.08</td>
<td>-21.61</td>
</tr>
<tr>
<td>5</td>
<td>30</td>
<td>01</td>
<td>12.28</td>
<td>-29.19</td>
<td>9.26</td>
<td>-21.68</td>
</tr>
<tr>
<td>5</td>
<td>31</td>
<td>01</td>
<td>12.65</td>
<td>-29.27</td>
<td>9.39</td>
<td>-21.47</td>
</tr>
<tr>
<td>5</td>
<td>32</td>
<td>01</td>
<td>12.97</td>
<td>-29.37</td>
<td>9.53</td>
<td>-21.26</td>
</tr>
<tr>
<td>5</td>
<td>33</td>
<td>01</td>
<td>13.23</td>
<td>-29.44</td>
<td>9.56</td>
<td>-21.13</td>
</tr>
<tr>
<td>5</td>
<td>34</td>
<td>01</td>
<td>13.46</td>
<td>-29.47</td>
<td>9.63</td>
<td>-21.12</td>
</tr>
<tr>
<td>5</td>
<td>35</td>
<td>01</td>
<td>13.78</td>
<td>-29.52</td>
<td>9.65</td>
<td>-20.88</td>
</tr>
<tr>
<td>5</td>
<td>36</td>
<td>01</td>
<td>14.03</td>
<td>-29.54</td>
<td>9.73</td>
<td>-20.50</td>
</tr>
<tr>
<td>5</td>
<td>37</td>
<td>01</td>
<td>14.36</td>
<td>-29.56</td>
<td>9.78</td>
<td>-20.24</td>
</tr>
<tr>
<td>5</td>
<td>38</td>
<td>01</td>
<td>14.67</td>
<td>-29.51</td>
<td>9.85</td>
<td>-19.73</td>
</tr>
<tr>
<td>5</td>
<td>39</td>
<td>01</td>
<td>14.73</td>
<td>-29.62</td>
<td>9.91</td>
<td>-19.92</td>
</tr>
<tr>
<td>5</td>
<td>39</td>
<td>01</td>
<td>15.04</td>
<td>-29.58</td>
<td>9.98</td>
<td>-19.37</td>
</tr>
<tr>
<td>5</td>
<td>40</td>
<td>01</td>
<td>15.15</td>
<td>-29.62</td>
<td>9.98</td>
<td>-19.26</td>
</tr>
<tr>
<td>5</td>
<td>41</td>
<td>01</td>
<td>15.22</td>
<td>-29.65</td>
<td>9.97</td>
<td>-18.97</td>
</tr>
<tr>
<td>5</td>
<td>42</td>
<td>01</td>
<td>15.33</td>
<td>-29.65</td>
<td>9.96</td>
<td>-18.76</td>
</tr>
<tr>
<td>5</td>
<td>43</td>
<td>01</td>
<td>15.47</td>
<td>-29.61</td>
<td>9.98</td>
<td>-18.25</td>
</tr>
<tr>
<td>5</td>
<td>44</td>
<td>01</td>
<td>15.70</td>
<td>-29.50</td>
<td>10.03</td>
<td>-17.21</td>
</tr>
<tr>
<td>5</td>
<td>45</td>
<td>01</td>
<td>15.99</td>
<td>-29.34</td>
<td>10.07</td>
<td>-16.30</td>
</tr>
<tr>
<td>5</td>
<td>46</td>
<td>01</td>
<td>16.12</td>
<td>-29.26</td>
<td>10.11</td>
<td>-15.91</td>
</tr>
<tr>
<td>5</td>
<td>47</td>
<td>01</td>
<td>16.22</td>
<td>-29.27</td>
<td>10.05</td>
<td>-15.77</td>
</tr>
<tr>
<td>5</td>
<td>48</td>
<td>01</td>
<td>16.29</td>
<td>-29.28</td>
<td>10.10</td>
<td>-15.67</td>
</tr>
</tbody>
</table>

Simon Nee 24/05/2002